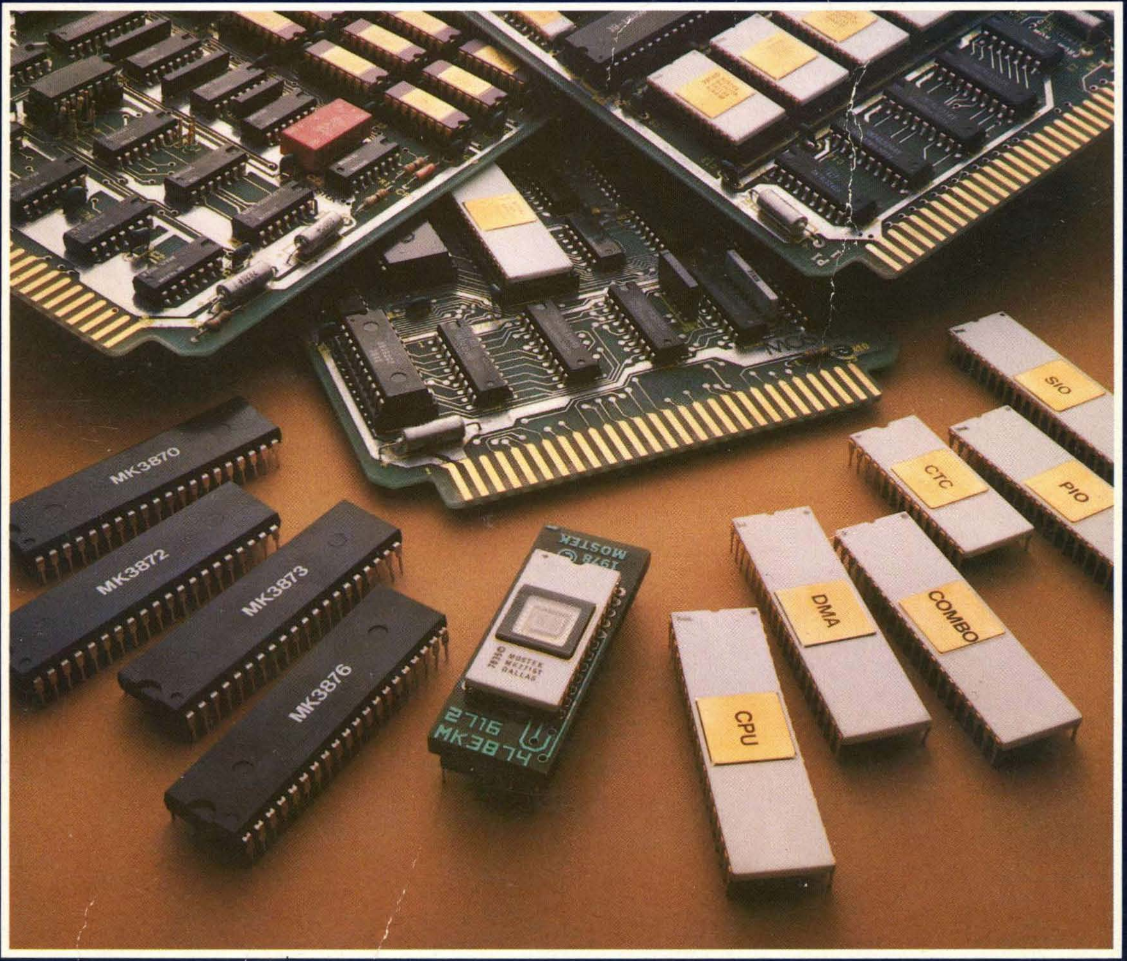
MOSTEK 1979

MICROCOMPUTER DATA BOOK



1979
Microcomputer Products
Data Book

Mostek reserves the right to make changes in specifications at any time and without notice. The information furnished by Mostek in this publication is believed to be accurate and reliable. However, no responsibility is assumed by Mostek for its use; nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights of Mostek.

The "PRELIMINARY" designation on a Mostek data sheet indicates that the product is not characterized. The specifications are subject to change, are based on design goals or preliminary part evaluation, and are not guaranteed. Mostek Corporation or an authorized sales representative should be consulted for current information before using this product. No responsibility is assumed by Mostek for its use; nor for any infringements of patents and trademarks or other rights of third parties resulting from its use. No license is granted under any patents, patent rights, or trademarks of Mostek. Mostek reserves the right to make changes in specifications at any time and without notice.

PRINTED IN USA June 1979
Publication Number MK79707

1979 MICROCOMPUTER DATA BOOK

1979 MICROCOMPUTER DATA BOOK

1979 MICROCOMPUTER DATA BOOK

FUNCTIONAL INDEX

Product Specifications

Application Note

Product Specifications

Device Specifications

Application Note

Device Specifications

Device Specifications

Application Notes

Product Specifications

Product Specifications

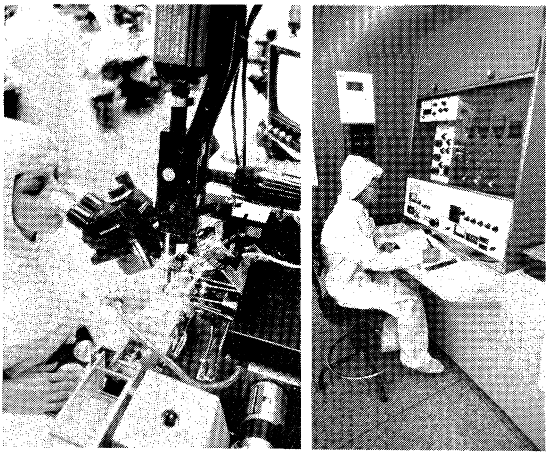
Product Specifications

Product Specifications

1979 MICROCOMPUTER DATA BOOK

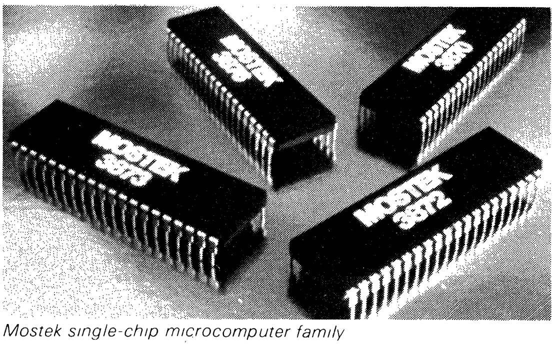
MOSTEK 1969-1979
Ten Years of Technology Leadership

Mostek Technology. Technology links the past, present, and future of Mostek. Innovations in both circuit and system design, and wafer processing have accounted for our rapid growth and for the strong acceptance of Mostek as a technology leader.



The proven process technology in the semiconductor industry is N-Channel silicon-gate MOS. Mostek is recognized as an important innovator in this process because of the continuing development of new techniques and enhancements which allow significant performance breakthroughs in our products. Competing technologies have not yet been able to approach either the performance or producibility of N-Channel MOS. Therefore, it appears that NMOS silicon-gate will continue to lead industry developments for several years to come.

Microcomputer Components. Mostek's microcomputer products cover the full spectrum of microprocessor applications worldwide.

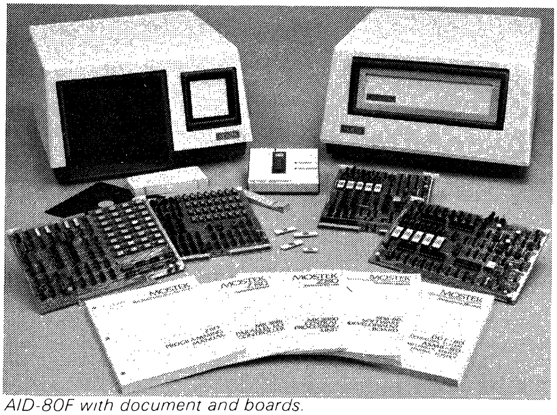


Mostek single-chip microcomputer family

Mostek's Z80 is the most powerful 8-bit microcomputer available. It is software compatible with the 8080A yet has some significant system advantages — an increased instruction set, reduced dynamic memory interfacing costs, reduced I/O costs and reduced support circuitry costs.

Mostek's 3870 Family of single-chip microcomputers allow system flexibility and expansion while retaining the design and economic advantages of single-chip construction. Software compatible with the F8, Mostek's 3870 family is the answer to a wide range of low-cost microcomputer applications.

Microcomputer Systems. Mostek's microcomputer line is supported by a wide array of development aids. These include software development boards that may be used as software development aids or as stand-alone microcomputers. Add-on memory boards, application interface modules, and emulators assist in system design, debugging, and field testing.



AID-80F with document and boards.

Mostek's microcomputer line includes Mostek's MD Series™ of OEM microcomputer boards. The MD Series features both stand-alone boards (designated MD) and expandable boards (designated MDX) that are STD-Z80 BUS compatible. These powerful Z80-based boards are simple and economical to use.

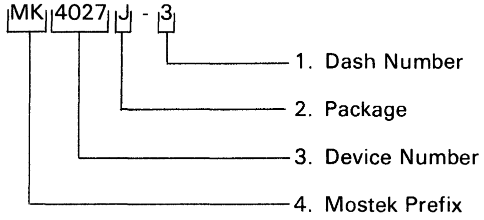
Also available is Mostek's AID-80F™, a dual floppy-disk development system that develops and debugs software for Mostek's entire microcomputer line.

Mostek provides a complete base of powerful software and software aids, complete documentation, and factory and field-application engineers.

ORDERING INFORMATION

Factory orders for parts described in this book should include a four-part number as explained below:

Example:



1. Dash Number

   One or two numerical characters defining spacific device performance characteristic.

2. Package

   P - Gold side-brazed ceramic DIP
   J - CER-DIP
   N - Epoxy DIP (Plastic)
   K - Tin side-brazed ceramic DIP
   T - Ceramic DIP with transparent lid
   E - Ceramic leadless chip carrier

3. Device Number

1XXX or 1XXXX	- Shift Register, ROM
2XXX or 2XXXX	- ROM, EPROM
3XXX or 3XXXX	- ROM, EPROM
38XX	- Microcomputer Components
4XXX or 4XXXX	- RAM
5XXX or 5XXXX	- Counters, Telecommunication and Industrial
7XXX or 7XXXX	- Microcomputer Systems

4. Mostek Prefix

   MK-Standard Prefix

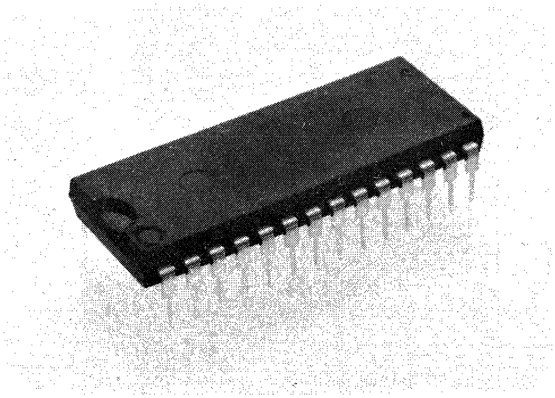
   MKB-100% 883B screening, with final electrical test at low, room and high-rated temperatures.

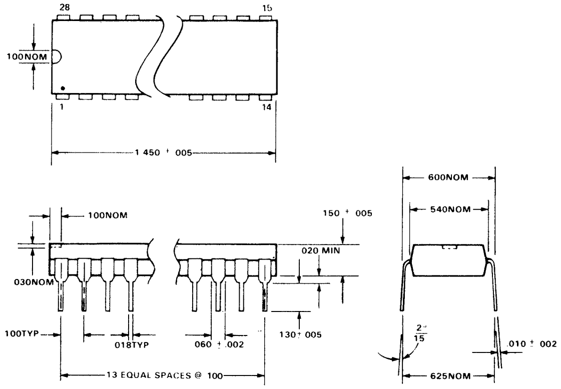
General

Package Descriptions

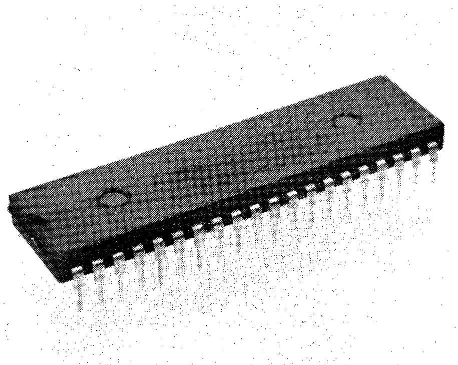
PLASTIC DUAL-IN LINE PACKAGING (N)

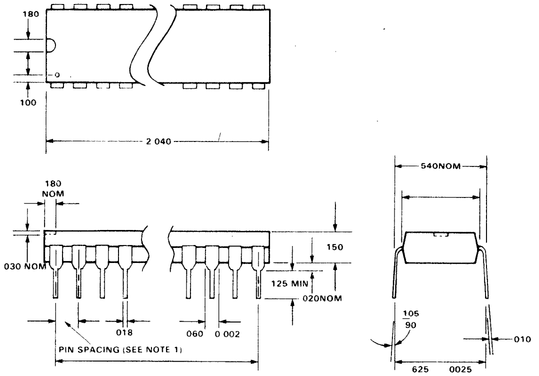
28 PIN





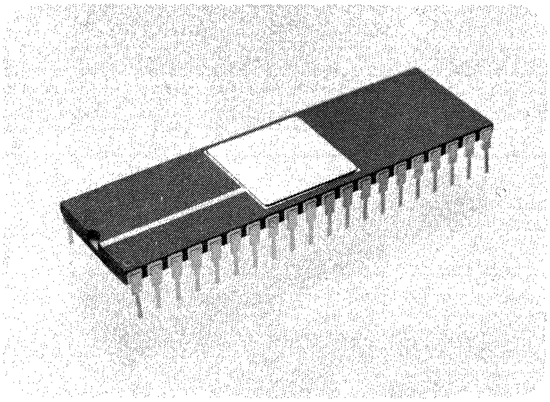
40 PIN

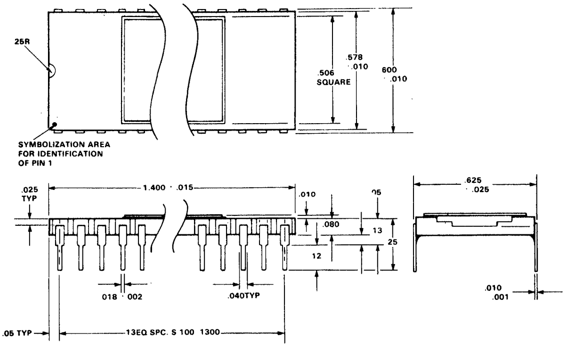




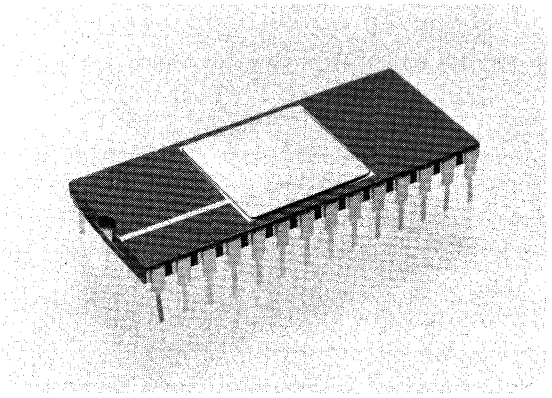
NOTES:
1. The true-position pin spacing is 0.100 between centerlines. Each pin centerline is located within ± 0.010 of its true longitudinal position relative to pins 1 and 40.

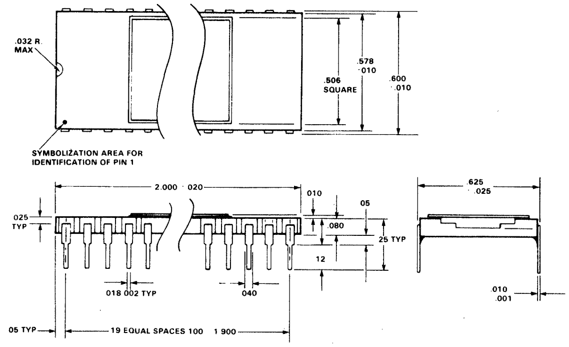
CERAMIC DUAL-IN-LINE HERMETIC PACKAGING (P)
28 PIN



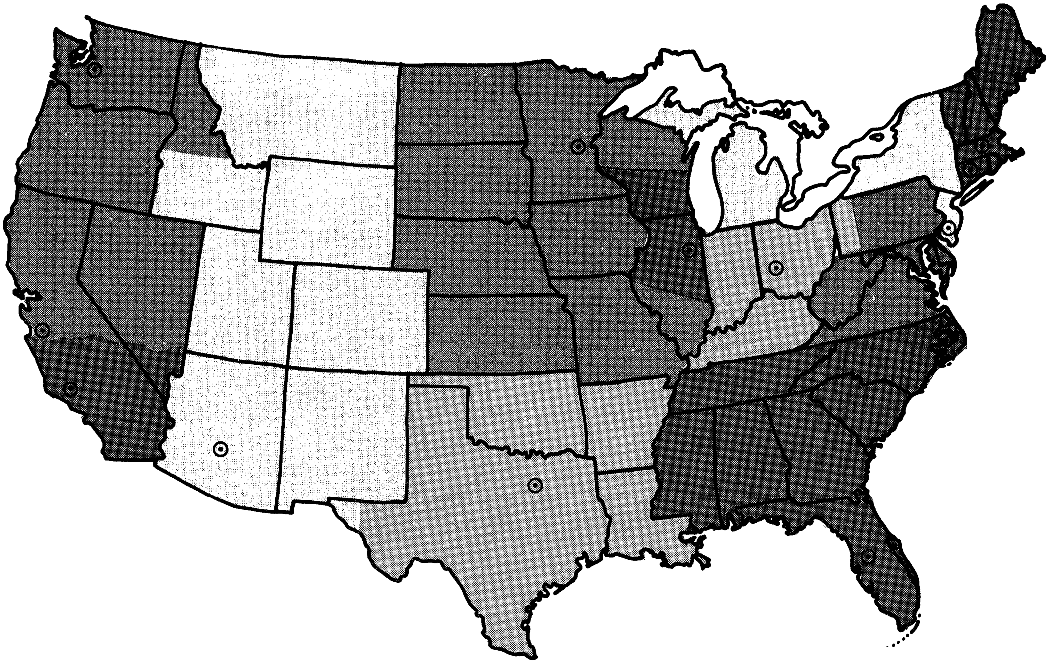


40 PIN





U.S. AND CANADIAN SALES OFFICES



CORPORATE HEADQUARTERS

Mostek Corporation
1215 W. Crosby Rd.
P. O. Box 169
Carrollton, Texas 75006

REGIONAL OFFICES

Mostek (Eastern U.S./Canada)
34 W. Putnam, 2nd Floor
Greenwich, Conn. 06830
203/622-0955
TWX 710-579-2928

Mostek (Northeast U.S.)
29 Cummings Park, Suite #426
Woburn, Mass. 01801
617/935-0635
TWX (Temp.) 710-332-0435

Mostek (Mid-Atlantic U.S.)
East Gate Business Center
125 Gaither Drive, Suite D
Mt. Laurel, New Jersey 08054
609/235-4112
TWX 710-897-0723

Mostek (Southeast U.S.)
Exchange Bank Bldg.
1111 N. Westshore Blvd.
Suite 414
Tampa, Florida 33607
813/876-1304
TWX 810-876-4611

Mostek (Central U.S.)
701 E. Irving Park Road
Suite 206
Roselle, Ill. 60172
312/529-3993
TWX 910-291-1207

Mostek (North Central U.S.)
6125 Blue Circle Drive, Suite A
Minnetonka, Mn. 55343
612/935-4020
TWX 910-576-2802

Mostek (South Central U.S.)
228 Byers Road
Suite 105
Miamisburg, Ohio 45342
513/866-3405
TWX 810-473-2976

Mostek (Michigan)
Livonia Pavillion East
29200 Vassar, Suite 815
Livonia, Mich. 48152
313/478-1470
TWX 810-242-2978

Mostek (Southwest U.S.)
4100 McEwen Road
Suite 237
Dallas, Texas 75234
214/386-9141
TWX 910-860-5437

Mostek (Northern California)
2025 Gateway Place
Suite 268
San Jose, Calif. 95011
408/287-5081
TWX 910-338-7338

Mostek (Southern California)
17870 Skypark Circle
Suite 107
Irvine, Calif. 92714
714/549-0397
TWX 910-595-2513

Mostek (Rocky Mountains)
8686 N. Central Ave.
Suite 126
Phoenix, Ariz. 85020
602/997-7573
TWX 910-957-4581

Mostek (Northwest)
1107 North East 45th Street
Suite 411
Seattle, Wa. 98105
206/632-0245

U.S. AND CANADIAN REPRESENTATIVES

ALABAMA
Beacon Elect. Assoc., Inc.
11309 S. Memorial Pkwy.
Suite G
Huntsville, AL 35803
205/881-5031
TWX 810-726-2136

ARIZONA
Summit Sales
7336 E. Shoeman Lane
Suite 116E
Scottsdale, AZ 85251
602/994-4587
TWX 910-950-1283

CALIFORNIA
Harvey King, Inc.
8124 Miramar Road
San Diego, CA 92126
714/566-5252
TWX 910-335-1231

COLORADO
Waugaman Associates
4800 Van Gordon
Wheat Ridge, CO 80033
303/423-1020
TWX 910-938-0750

CONNECTICUT
New England Technical Sales
33 Trotwood Drive
W. Hartford, CT 06117
203/236-4705

FLORIDA
Beacon Elect. Assoc., Inc.
6842 N.W. 20th Ave.
Ft. Lauderdale, FL 33309
305/971-7320
TWX 510-955-9834

Beacon Elect. Assoc., Inc.
P. O. Box 125
Ft. Walton Beach, FL 32548
904/244-1550

Beacon Elect. Assoc., Inc.
235 Maitland Ave.
P. O. Box 1278
Maitland, FL 32751
305/647-3498
TWX 810-853-5038

Beacon Elect. Assoc., Inc.
316 Laurie
Melbourne, FL 32901
305/259-0648
TWX 810-853-5038

GEORGIA
Beacon Elect. Assoc., Inc.*
6135 Barfield Rd.
Suite 112
Atlanta, GA 30328
404/256-9640
TWX 810-751-3165

ILLINOIS
Carlson Electronic Sales*
600 East Higgins Road
Elk Grove Village, IL 60007
312/956-8240
TWX 910-222-1819

INDIANA
Rich Electronic Marketing*
599 Industrial Drive
Carmel, IN 46032
317/844-8462
TWX 810-260-2631

Rich Electronic Marketing
3448 West Taylor St.
Fort Wayne, IN 46804
219/432-5553
TWX 810-332-1404

IOWA
Cahill Associates
226 Sussex Dr. N.E.
Cedar Rapids, IA 52402
319/377-4018

Carlson Electronic Sales
204 Collins Rd. N.E.
Cedar Rapids, IA 52402
319/377-6341

KANSAS
Rush & West Associates*
107 N. Chester Street
Olathe, KN 66061
913/764-2700
TWX 910-749-6404

KENTUCKY
Rich Electronic Marketing
5910 Bardstown Road
P. O. Box 91147
Louisville, KY 40291
502/239-2747

MASSACHUSETTS
New England Technical Sales*
135 Cambridge Street
Burlington, MA 01803
617/272-0434
TWX 710-332-0435

MICHIGAN
A.P.J. Associates, Inc.
496 Ann Arbor Trail
Plymouth, MI 48170
313/459-1200
TWX 810-242-6970

MINNESOTA
Cahill Associates*
315 N. Pierce
St. Paul, MN 55104
612/646-7217
TWX 910-563-3737

MISSOURI
Rush & West Associates
481 Melanie Meadows Lane
Ballwin, MO 63011
314/394-7271

NEW MEXICO
Waugaman Associates
9004 Menaul N.E.
Suite 7
P. O. Box 14894
Albuquerque, NM 87111

NORTH CAROLINA
Beacon Elect. Assoc., Inc.
1207 West Bessemer Ave.
Suite 112
Greensboro, NC 27408
919/275-9997
TWX 510-925-1119

NEW YORK
E R A (Engrg. Rep. Assoc.)
One DuPont Street
Plainview, NY 11803
516/822-9890
TWX 510-221-1849

Precision Sales Corp.
5 Arbustus Ln., MR-97
Binghamton, NY 13901
607/648-3686

Precision Sales Corp.*
1 Commerce Blvd.
Liverpool, NY 13088
315/451-3480
TWX 710-541-0483

Precision Sales Corp.
3594 Monroe Avenue
Rochester, NY 14534
716/381-2820

PENNSYLVANIA
CMS Marketing
121A Lorraine Avenue
P.O. Box 300
Oreland, PA 19075
215/885-5106
TWX 510-665-0161

TENNESSEE
Beacon Elect. Assoc., Inc.
100 Tulsa Road
Oak Ridge, TN 37830
615/482-2409
TWX 810-572-1077

Rich Electronic Marketing
1128 Tusculum Blvd.
Suite D
Greenville, TN 37743
615/639-3139

TEXAS
West & Associates, Inc.
8403 Shoal Creek Road
Austin, TX 78758
512/451-2456

West & Associates, Inc.*
4300 Alpha Road, Suite 106
Dallas, TX 75234
214/661-9400
TWX 910-860-5433

West & Associates, Inc.
9730 Town Park #101
Houston, TX 77036
713/777-4108

UTAH
Waugaman Associates
445 East 2nd South
Suite 304
Salt Lake City, UT 84111
801/363-0275
TWX 910/925-5607

WISCONSIN
Carlson Electronic Sales
Northbrook Executive Ctr.
10701 West North Ave.
Suite 209
Milwaukee, WI 53226
414/476-2790
TWX 910-222-1819

CANADA
Cantec Representatives Inc.*
17 Bentley Avenue
Ottawa, Ontario
Canada K2E 6T7
613/225-0363
TWX 610-562-8967

Cantec Representatives Inc.
15737 Rue Pierrefonds
Ste. Genevieve, P.Q.
Canada H9H 1G3
514/694-4049
TELEX 05-822790

Cantec Representatives Inc.
83 Galaxy Blvd., Unit 1A
(Rexdale)
Toronto, Canada M9W 5X6
416/675-2460
TWX 610-492-2655

*Home Office

U.S. AND CANADIAN DISTRIBUTORS

ARIZONA
Kierulff Electronics
4134 E. Wood St.
Phoenix, AZ 85040
602/243-4104
TWX 910/951-1550

CALIFORNIA
Bell Industries
1161 N. Fair Oaks Avenue
Sunnyvale, CA 94086
408/734-8570
TWX 910/339-9378

Arrow Electronics
720 Palomar Avenue
Sunnyvale, CA 94086
408/739-3011
TWX 910/339-9371

Intermark Electronics
1802 E. Carnegie Avenue
Santa Ana, CA 92705
714/540-1322
TWX 910/595-1583

Intermark Electronics
4125 Sorrento Valley Blvd.
San Diego, CA 92121
714/279-5200
TWX 910/335-1515

Intermark Electronics
1020 Stewart Drive
Sunnyvale, CA 94086
408/738-1111
TWX 910/339-9312

Kierulff Electronics
2585 Commerce Way
Los Angeles, CA 90040
213/725-0325
TWX 910/580-3106

Kierulff Electronics
3969 E. Bayshore Road
Palo Alto, CA 94303
415/968-6292
TWX 810/379-6430

Kierulff Electronics
8797 Balboa Avenue
San Diego, CA 92123
714/278-2112
TWX 910/335-1182

Kierulff Electronics
14101 Franklin Avenue
Tustin, CA 92680
714/731-5711
TWX 910/595-2599

Schweber Electronics
17811 Gillette Avenue
Irvine, CA 92714
714/556-3880
TWX 910/595-1720

COLORADO
Bell Industries
8155 W. 48th Avenue
Wheatridge, CO 80033
303/424-1985
TWX 910/938-0393

Kierulff Electronics
10890 E. 47th Avenue
Denver, CO 80239
303/371-6500
TWX 910/932-0169

CONNECTICUT
Arrow Electronics
295 Treadwell
Hamden, CT 06514
203/248-3801
TWX 710/465-0780

Schweber Electronics
Finance Drive
Commerce Industrial Park
Danbury, CT 06810
203/792-3500
TWX 710/456-9405

FLORIDA
Arrow Electronics
1001 N.W. 62nd St.
Suite 108
Ft. Lauderdale, FL 33309
305/776-7790
TWX 510/955-9456

Arrow Electronics
115 Palm Bay Road, N.W.
Suite 10 Bldg. 200
Palm Bay, FL 32905
305/725-1480
TWX 510/959-6337

Diplomat Southland
2120 Calumet
Clearwater, FL 33515
813/443-4514
TWX 810/866-0436

Kierulff Electronics
3247 Tech Drive
St. Petersburg, FL 33702
813/576-1966
TWX 810/863-5625

GEORGIA
Arrow Electronics
3406 Oakcliff Road
Doraville, GA 30340
404/455-4054
TWX 810/757-4213

Schweber Electronics
4126 Pleasantdale Road
Atlanta, GA 30340
404/449-9170

ILLINOIS
Arrow Electronics
492 Lunt Avenue
P. O. Box 94248
Schaumburg, IL 60193
312/893-9420

Bell Industries
3422 W. Touhy Avenue
Chicago, IL 60645
312/982-9210
TWX 910/223/4519

Kierulff Electronics
1536 Lanmeier
Elk Grove Village, IL 60007
312/640-0200
TWX 910/222-0351

INDIANA
Advent Electronics
8505 Zionsville Road
Indianapolis, IN 46268
317/297-4910
TWX 810/341-3228

Ft. Wayne Electronics
3606 E. Maumee
Ft. Wayne, IN 46803
219/423-3422
TWX 810/332-1562

Graham Electronics
133 S. Pennsylvania St.
Indianapolis, IN 46204
317/634-8202
TWX 810/341-3481

IOWA
Advent Electronics
682 58th Avenue
Court South West
Cedar Rapids, IA 52404
319/363-0221

LOUISIANA
Sterling Electronics
4613 Fairfield Avenue
Metairie, LA 70005
504/887-7610
Telex 58-328

MASSACHUSETTES
Kierulff Electronics
13 Fortune Drive
Billerica, MA 01821
617/667-8331
TWX 710/390-1449

Lionex Corporation
1 North Avenue
Burlington, MA 01803
617/272-9400
TWX 710/332-1387

Schweber Electronics
25 Wiggins Avenue
Bedford, MA 01730
617/275-5100
TWX 710/326-0268

Arrow Electronics
96D Commer Way
Woburn, MA 01801

MARYLAND
Arrow Electronics
4801 Benson Avenue
Baltimore, MD 21227
301/247-5200
TWX 710/236-9005

Cramer Electronics
16021 Industrial Drive
Gaithersburg, MD 20760
301/948-0110
TWX 710/828-0082

MICHIGAN
Arrow Electronics
3921 Varsity Drive
Ann Arbor, MI 48104
313/971-8220
TWX 810/223-6020

Schweber Electronics
33540 Schoolcraft Road
Livonia, MI 48150
313/525-8100

MINNESOTA
Arrow Electronics
5251 W. 73rd Street
Edina, MN 55435
612/830-1800
TWX 910/576-3125

MISSOURI
Olive Electronics
9910 Page Blvd.
St. Louis, MO 63132
314/426-4500
TWX 910/763-0710

Semiconductor Spec
3805 N. Oak Trafficway
Kansas City, MO 64116
816/452-3900
TWX 910/771-2114

NEW JERSEY
Arrow Electronics
Pleasant Valley Avenue
Morrestown, NJ 08057
609/235-1900
TWX 710/897-0892

Arrow Electronics
285 Midland Avenue
Saddlebrook, NJ 07662
201/797-5800
TWX 710/988-2206

Kierulff Electronics
3 Edison Place
Fairfield, NJ 07006
201/575-6750
TWX 710/734-4372

Schweber Electronics
18 Madison Road
Fairfield, NJ 07006
201/227-7880
TWX 710/734-3405

NEW MEXICO
Bell Industries
11728 Linn N.E.
Albuquerque, NM 87123
505/292-2700
TWX 910/989-0625

NEW YORK
Arrow Electronics
900 Broad Hollow Rd.
Farmingdale, L.I., NY 11735
516/694-6800
TWX 510/224-6494

Cramer Electronics
7705 Maltage Drive
P. O. Box 370
Liverpool, NY 13088
315/652-1000
TWX 710/545-0230

Cramer Electronics
3000 S. Winton Road
Rochester, NY 14623
716/275-0300
TWX 510/253-4766

Lionex Corporation
415 Crossway Park Drive
Woodbury, NY 11797
516/921-4414
TWX 510/221-2196

Schweber Electronics
2 Twin Line Circle
Rochester, NY 14623
716/424-2222

Schweber Electronics
Jericho Turnpike
Westbury, NY 11590
516/334-7474
TWX 510/222-3660

NORTH CAROLINA
Arrow Electronics
1369G South Park Drive
Kernersville, NC 27282
919/996-2039

Hammond Electronics
2923 Pacific Avenue
Greensboro, NC 27406
919/275-6391
TWX 510/925-1094

OHIO
Arrow Electronics
3100 Plainfield Road
Kettering, OH 45432
513/253-9176
TWX 810/459-1611

Arrow Electronics
10 Knoll Crest Drive
Reading, OH 44139
513/761-5432
TWX 810/461-2670

Arrow Electronics
6238 Cochran Road
Solon, OH 44139
216/248-3990
TWX 810/427-9409

Schweber Electronics
23880 Commerce Park Road
Beachwood, OH 44122
216/464-2970
TWX 810/427-9441

OKLAHOMA
Sterling Electronics
9810 E. 42nd Street
Suite 229
Tulsa, OK 74145
918/663-2410
Telex 49-9440

PENNSYLVANIA
Schweber Electronics
101 Rock Road
Horsham, PA 19044
215/441-0600

SOUTH CAROLINA
Hammond Electronics
1035 Lown Des Hill Rd.
Greenville, SC 29602
803/233-4121
TWX 810/281-2233

TEXAS
Arrow Electronics
13740 Midway Road
P.O. Box 401068
Dallas, TX 75240
214/661-9300
TWX 910/861-5495

Quality Components
10201 McKalla
Suite D
Austin, TX 78758
512/838-0551

Quality Components
4303 Alpha Road
Dallas, TX 75240
214/387-4949
TWX 910/860-5459

Quality Components
6126 Westline
Houston, TX 77036
713/772-7100

Schweber Electronics
7420 Harwin Drive
Houston, TX 77036
713/784-3600
TWX 910/881-1109

Sterling Electronics
2800 Longhorn Blvd.
Suite 101
Austin, TX 78759
512/836-1341
Telex - 776-407

Sterling Electronics
2875 Merrell Road
P.O. Box 29317
Dallas, TX 75229
214/357-9131
Telex - 025

Sterling Electronics
4201 Southwest Freeway
Houston, TX 77027
713/627-9800
TWX 910/881-5042

UTAH
Bell Industries
2258 S. 2700 W.
Salt Lake City, UT 84119
801/972-6969
TWX 910/925-5686

Kierulff Electronics
3695 W. 1987 South St.
Salt Lake City, UT 84104
801/973-6913

WASHINGTON
Kierulff Electronics
1005 Andover Park East
Seattle, WA 98188
206/575-4420
TWX 910/444-2034

WISCONSIN
Arrow Electronics
434 Rawson Avenue
Oak Creek, WI 53154
414/764-6600
TWX 910/262-1192

CANADA
Prelco Electronics
2767 Thames Gate Drive
Mississauga, Ontario
Toronto L4T 1G5
416/678-0401
TWX 610/492-8974

Prelco Electronics
480 Port Royal St. W.
Montreal 357 P.Q. H3L 2B9
514/389-8051
TWX 610/421-3616

Prelco Electronics
1770 Woodward Drive
Ottowa, Ontario K2C 0P8
613/226-3491
TWX 610/562-8724

R.A.E. Industrial
3455 Gardner Court
Burnaby, B.C. V5G 4J7
604/291-8866
TWX-604/291-8866

W.E.S. Ltd.
1515 King Edward St.
Winnipeg, Manitoba R3H 0R8
204/632-1260
Telex - 07-57347

INTERNATIONAL MARKETING OFFICES

EUROPEAN HEAD OFFICE
Mostek International
150 Chausee de la Hulpe
B-1170 Brussels
Belgium
32 2-660.69.24
Telex - 62011

Austria
Transistor-Vertriebs GmbH
Auhofstrasse 41 A
A-1130 Vienna
43 222-829.45.12
Telex - 13738

Belgium
Sotronic
14, Rue Pere de Deken
B-1040 Brussels
32 2-736.10.07
Telex - 25141

Denmark
Semicap APS
Gammel Kongevej 184.5
DK-1850 Copenhagen
45 1-22.15.10
Telex - 15987

Finland
S.W. Instruments
Karstulantie 4B
SF-00550 Helsinki 55
358-0-73.82.65
Telex - 122411

France
Mostek France s.a.r.l
30 Rue de Morvan
SILIC 505
F-94623 Rungis Cedex
33 1-687.34.14
Telex - 204049

I.P.C.
113, Rue Artistide Briand
F-91400 Orsay
33 10-19.27
Telex - 691451

P.E.P.
4, Rue Barthelemy
F-92120 Montrouge
33 1-735.33.20
Telex - 204534

SCAIB
80, Rue d'Arcuil
SILIC 137
F-94150 Rungis Cedex
33 1-687.23.12
Telex - 204674

Germany
Mostek GmbH
Talstrasse 172
D-7024 Filderstadt 1
49 711-70.10.45
Telex - 7255792

Mostek GmbH
Friedlandstrasse 1
d-2085 Quickborn
49 4106-2077/78
Telex - 213685

Neye Enatechnik GmbH
Schillerstrasse 14
D-2085 Quickborn
49 4106-61.22.95
Telex - 213.590

Dr Dohrenberg
Bayreuther Strasse 3
D-1 Berlin 30
49 30-213.80.43
Telex - 184860

Raffel-Electronic GmbH
Lochnerstrasse 1
D-4030 Ratingen
49 2102-280.24
Telex - 8585180

Siegfried Ecker
Konigsberger Strasse 2
D-6120 Michelstadt
49 6061-2233
Telex - 4191630

Matronic GmbH
Lichtenberger Weg 3
D-7400 Tubingen
49 7071-24.43.31
Telex - 726.28.79

Dema-Electronic GmbH
Blutenstrasse 21
D-8 Munchen 40
49 89-288018
Telex - 28345

Italy
Mostek Italia S.p.A
Via G. da Procida, 10
I-20149 Milano
39 2-349.26.96
Telex - 333601

Comprel S.r.L.
Viale Romagna, 1
I-20092 Cinisello Balsamo
39 2-928.08.09/928.03.45
Telex - 332484

The Netherlands
Nijkerk Elektronika BV
Drentestraat 7
1083 HK Amsterdam
020.428.933
Telex - 11625

Norway
Hefro Tekniska A/S
Postboks 6596
Rodelkka
Oslo 5
47 2-38.02 86
Telex - 16205

Sweden
Mostek Scandinavia AB
Magnusvagen 1, 8 tr.
S-17531 Jarfalla
46 758-343.38
Telex - 12997

Interelko AB
Strandbergsg. 47
S-11251 Stockholm
46 8-13.21.60
Telex - 10689

Spain
Comelta S.A.
Cia Electrinica Tecnicas
Aplicadas
Consejo de Ciento, 204
Entlo 3A.
Barcelona 11
34 3-254.66.07/08
Telex - 51934

Switzerland
Memotec AG
CH-4932 Lotzwil
41 63-28.11.22
Telex - 68636

United Kingdom
Mostek U.K. Ltd.
Masons House
1-3 Valley Drive
Kingsbury Road,
London, N.W. 9
44 1-204.93.22
Telex - 25940

Celdis Limited
37-39 Loverock Road
Reading
Berks RG 31 ED
44 734-58.51.71
Telex - 848370

Distronic Limited
50-51 Burnt Mill
Elizabeth Way,
Harlow
Essex CM 202 HU
44 279-32.497/39.701
Telex - 81387

A.M. Lock co., Ltd.
Neville Street,
Chadderton,
Oldham, Lancashire
44 61-652.04.31
Telex - 669971

Pronto Electronic Systems Ltd.
645 High Road,
Seven Kings,
Ilford,
Essex IG 38 RA
44 1-599.30.41
Telex - 24507

Yugoslavia
Chemcolor
Inozemma Zastupstva
Proleterskih brigada 37-a
41001 Zagreb
41-513.911
Telex - 21236

Argentina
Rayo Electronics, S.R.L.
Belgrano 990, Pisos 6y2
1092 Buenos Aires
38-1779, 37-9476
Telex - 122153

Australia
Amtron Tyree Pty. Ltd.
176 Botany Street
Waterloo, N.S.W. 2017
61 69-89.666
Telex - 25643

Brasil
Cosele, Ltda.
Rua da Consolacao, 867
Conj. 31
01301 Sao Paulo
55 11-257.35.35/258.43.25
Telex - 1130869

Hong Kong
Cet Limited
1402 Tung Wah Mansion
199-203 Hennessy Road
Wanchai, Hong Kong
5-72.93.76
Telex - 85148

Israel
Telsys Limited
54 Jabotinsky Road
Ramat-Gan 52462
972 73.98.65
72.23.62
Telex - 32392

Japan
Systems Marketing, Inc.
4th Floor, Shindo Bldg.
3-12-5 Uchikanda,
Chiyoda-Ku,
Tokyo, 100
81 3-254.27.51
Telex - 25761

Teijin Advanced Products Corp.
1-1 Uchisaiwai-Cho
2-Chome Chiyoda-Ku
Tokyo, 100
81 3-506.46.73
Telex - 23548

Korea
Vine Overseas Trading Corp.
Room 303-Tae Sung Bldg.
199-1 Jangsa-Dong
Jongro-Ku
Seoul
26-1663, 25-9875
Telex - 24154

New Zealand
E.C.S. Div. of Airspares
P.O. Box 1048
Airport Palmerston North
77-047
Telex - 3766

South Africa
Radiokom
P.O. Box 56310
Pinegowrie
2123,
Transvaal
789-1400
Telex - 8-0838 SA

Taiwan
Dynamar Taiwan Limited
P.O. Box 67-445
2nd Floor, No. 14, Lane 164
Sung-Chiang Road
Taipei
5418251
Telex - 11064

1979 MICROCOMPUTER DATA BOOK

Z80 Central Processor Module (MDX-CPU1)

FEATURES

□ Z80 CPU
□ 4K x 8 EPROM (two 2716's, customer provided)
□ 256 x 8 Static RAM (compatible with DDT-80 debugger).
□ Flexible Memory decoding for EPROM and RAM
□ Four counter/timer channels
□ Restart to 0000H or E000H (strapping option)
□ Debug compatible for single step in DDT-80
□ 4MHz version available
□ +5V only
□ Fully buffered signals for system expandability
□ STD-Z80 BUS compatible

DESCRIPTION

The MD Series and the STD-Z80 BUS were designed to satisfy the need for low cost OEM microcomputer modules. The STD-Z80 BUS uses a motherboard interconnect system concept and is designed to handle any MD Series Card type in any slot. The modules for the STD-Z80 BUS are a compact 4.5 x 6.5 inches which provide for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities provide high functionality per module.

The MD Series of OEM microcomputer boards and the STD-Z80 BUS offer the most cost effective system configuration available to the OEM system designer.

MDX-CPU1 DESCRIPTION

The MOSTEK MDX-CPU1 is the heart of an MD Series Z80 system. Based on the powerful Z80 microprocessor, the MDX-CPU1 can be used with great versatility in an OEM microcomputer system application. This is done simply by inserting custom ROM or EPROM memories into the sockets provided on the board and configuring them virtually anywhere within the Z80 memory map.

On board memory is provided in the form of sockets for 4K of EPROM (2-2716's) and 256 bytes of scratchpad RAM as pictured in the block diagram. In addition, an MK3882 Counter Timer Circuit is included on the MDX-CPU1 to provide counting and timing functions for the Z80. Either 2716 EPROM can be located at any 2K boundary within any given 16K block in the Z80 memory map via a jumper arrangement.

The MDX-CPU1 can be used in conjunction with the MDX-DEBUG and MDX-DRAM modules to utilize DDT-80 and ASMB-80 in system development. This is accomplished by strapping the scratchpad RAM to reside at location FF00 so that it will act as the Operating System RAM for DDT-80.

The MDX-CPU1 is also available in 4MHz version (MDX-CPU1-4). In this version, one wait cycle is automatically inserted each time on-board memory is accessed by a read or write cycle. This is necessary to make the access times of the 2716 PROMs and the 3539 scratchpad RAM compatible with the MK3880-4 MHz Z80-CPU.

ELECTRICAL SPECIFICATIONS

WORD SIZE

Instruction:	8, 16, 24, or 32 bits
Data:	8 bits

CYCLE TIME

Clock period or T state = 0.4 microsecond @ 2.5MHz
or = 0.25 microsecond @ 4.00MHz
Instructions require from 4 to 23 T states

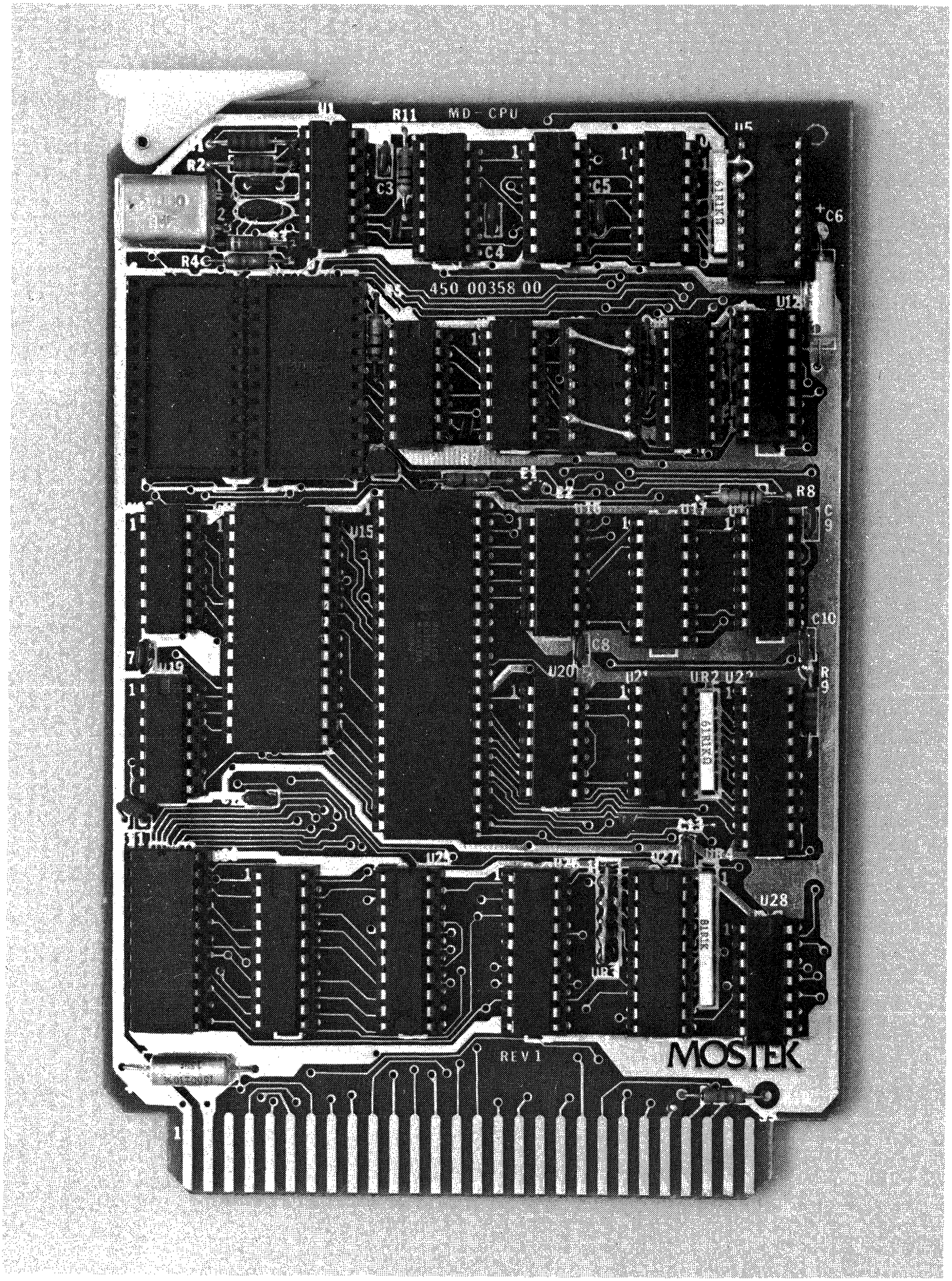
MEMORY ADDRESSING

On-Board EPROM: jumper selectable for any 2K boundary within a 16K block of Z80 memory map.
On-Board RAM: FF00-FFFF

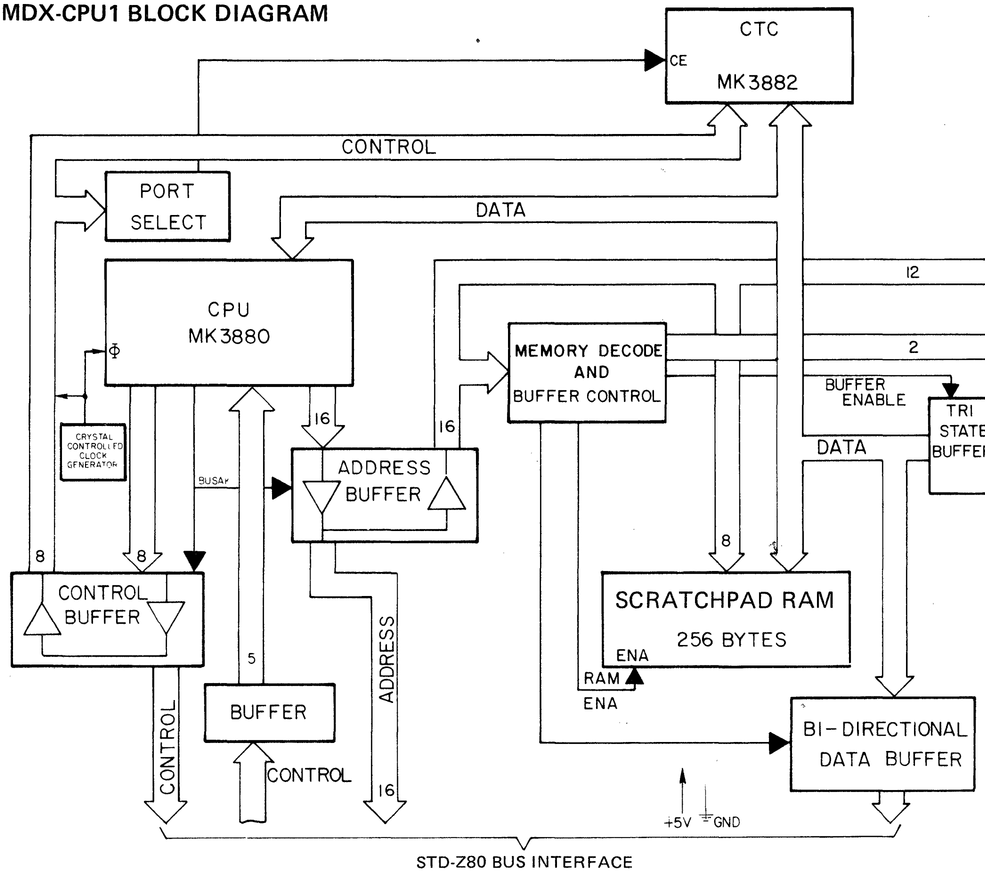
MEMORY CAPACITY

On-Board EPROM - 4K bytes (sockets only)
On-Board RAM - 256 bytes
Off-board Expansion - Up to 65,536 byte, with user-specified combinations of RAM, ROM, PROM.

MDX-CPU1 BOARD WITH OVERLAY



MDX-CPU1 BLOCK DIAGRAM



MEMORY SPEED REQUIRED

MEMORY	ACCESS TIME	CYCLE TIME
2716*	450nS	450nS

* Single 5 volt type required

I/0 ADDRESSING

On-Board Programmable Timer

PORT ADDRESS (HEX)	MK3882 CHANNEL
7C	0
7D	1
7E	2
7F	3

I/0 CAPACITY

Up to 252 port addresses can be decoded off board. Four port addresses are on board. 252 + 4 = 256 total I/0 ports.

INTERRUPTS

Multi-level with three vectoring mode (Mode 0, 1, 2). Interrupt requests may originate from user-specified I/0 or from the on-board MK3882 CTC.

PARALLEL BUS INTERFACE - STD-Z80 BUS COMPATIBLE

Input	One 74LS load max
Bus Outputs	I_{OH} = -15 mA min at 2.4 volts
	I_{OL} = 24 mA min at 0.5 volts

SYSTEM CLOCK

	MIN	MAX
MDX-CPU1	500 KHz	2.500 MHz
MDX-CPU1-4	500 KHz	4.000 MHz

POWER SUPPLY REQUIREMENTS

5V ± 5% at 1.1A maximum

OPERATING TEMPERATURE

0° C to 50° C

MECHANICAL SPECIFICATIONS

CARD DIMENSIONS

4.5 in (11.43cm) high by 6.50 in. (16.51cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD-Z80 BUS	56 pin dual read out	Printed Circuit Viking 3VH28/ ICE5
	0.125 in. centers	Wire Wrap Viking 3VH28/ 1CND5
		Solder Lug Viking 3VH28/ 1CN5

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MDX-CPU1	Module with Operation Manual less EPROMs and mating connectors. 2.5MHz version.	MK77850
MDX-CPU1-4	Module with Operations Manual less EPROMs and mating connectors. 4.0 MHz version.	MK77850-4
	MDX-CPU1 and -4 Operations Manual	MK79612
MDX-PROTO data sheet	MD Series Protyping package	MK79605
AID-80F data sheet	Disk based development system for MD Series	MK78568
AIM-80 data sheet	Z80 In-circuit Emulation module (2.5 MHz only)	MK78537

MOSTEK®

MD SERIES MICROCOMPUTER MODULES

Dynamic Ram Module (MDX-DRAM)

FEATURES

- □ Three memory sizes
  8K x 8 (MDX-DRAM8)
  16K x 8 (MDX-DRAM16)
  32K x 8 (MDX-DRAM32)
- □ Selectable addressing on 4K boundaries.
- □ 4MHz version available (MDX-DRAM-4)
- □ STD BUS compatible

DESCRIPTION

The MD Series and the STD-Z80 BUS were designed to satisfy the need for low cost OEM microcomputer modules. The STD-Z80 BUS uses a motherboard interconnect system concept and is designed to handle any MD Series card type in any slot. The modules for the STD-Z80 BUS are a compact 4.5 x 6.5 inches which provides for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities provide high functionality per module.

The MD Series of OEM microcomputer boards and the STD-Z80 BUS offer the most cost effective system configuration available to the OEM system designer.

MDX-DRAM DESCRIPTION

The MDX-DRAM is designed to be a RAM memory expansion board for the MOSTEK MD SERIES of Z80 based microcomputers. It is available in three memory capacities: 8K bytes (MDX-DRAM8), 16K bytes (MDX-DRAM16), and 32K bytes (MDX-DRAM32). Additionally, the MDX-DRAM16 and the MDX-DRAM32 are available in a 4MHz version. Thus, the designer can choose from the various options to tailor his add-on dynamic RAM directly to his system requirements.

The MDX-DRAM8 is designed using MOSTEK's MK4108, 8,192-bit dynamic RAM. The MDX-DRAM16 and MDX-DRAM32 utilize high-performance MK4116, 16,384-bit dynamic RAMs which allow 4MHz versions of these boards to be offered. No wait-state insertion circuitry is required on any of the RAM cards.

Address selection is provided on all MDX-DRAM cards for positioning the 8K, 16K, or 32K of memory to start on any 4K boundary.

ELECTRICAL SPECIFICATIONS

WORD SIZE

8 bits

MEMORY SIZE

MDX-DRAM8 - 8,192 bytes
MDX-DRAM16 - 16,384 bytes
MDX-DRAM32 - 32,768 bytes

ACCESS TIMES

SYSTEM CLOCK		MEMORY ACCESS TIMES	MEMORY CYCLE TIMES
MDX-DRAM	2.5 MHz	350ns max.	465ns min.
MDX-DRAM-4	4.0 MHz	200ns max.	325ns min.

ADDRESS SELECTION

Selection of 8K, 16K, or 32K contiguous memory blocks to reside at any 4K boundary

PARALLEL BUS INTERFACE-STD BUS COMPATIBLE

Inputs	One 74LS load max
Bus Outputs	I_{OH} = -15mA min. at 2.4 volts
	I_{OL} = 24mA min. at 0.5 volts

SYSTEM CLOCK

	Min	Max
MDX-DRAM	1.25MHz	2.5MHz
MDX-DRAM-4	1.25MHz	4.0MHz

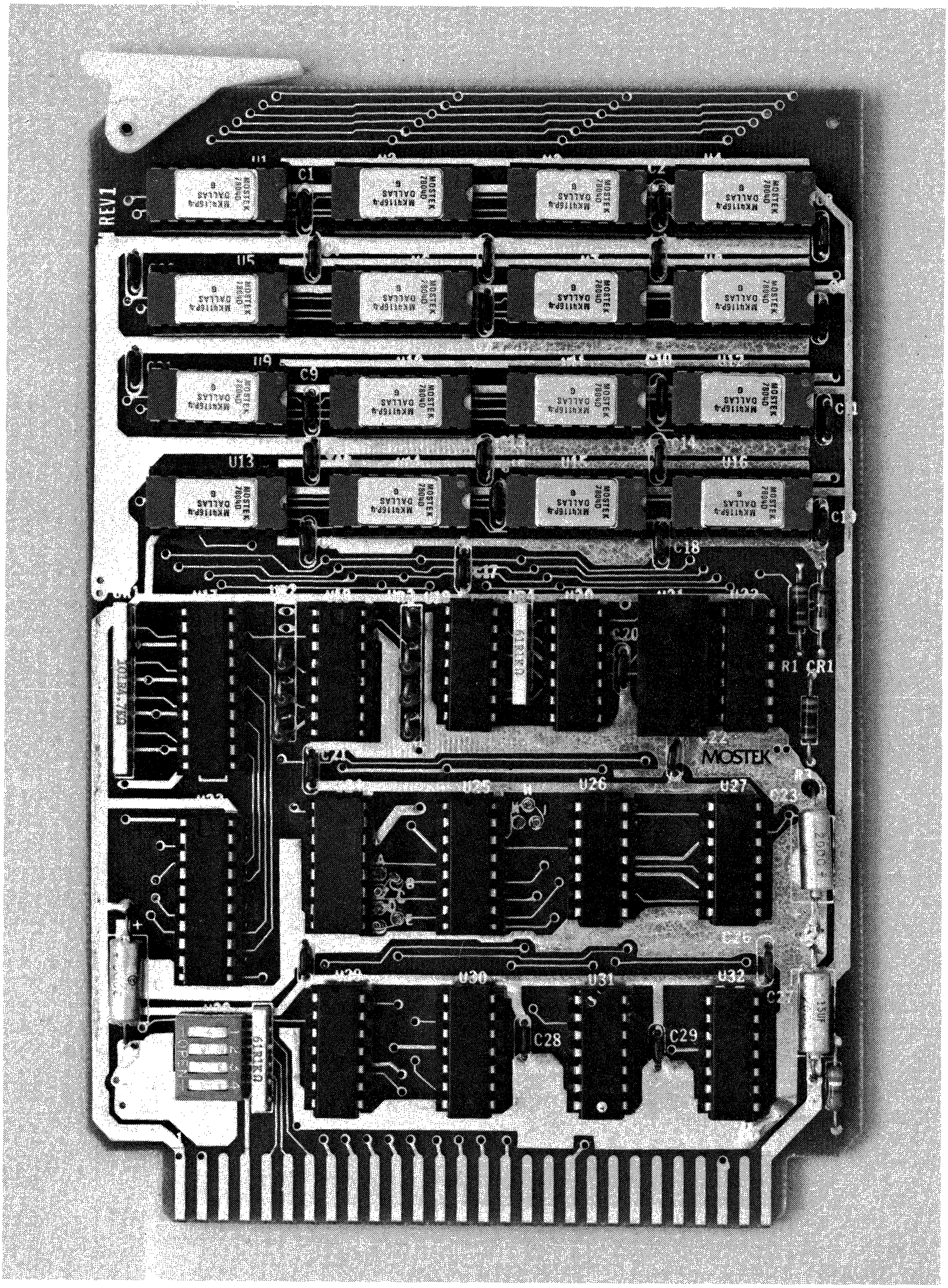
POWER SUPPLY REQUIREMENTS

+5V ± 5% at 0.6A max.
+12V ± 5% at 0.25A max.
-12V ± 5% at 0.03A max.

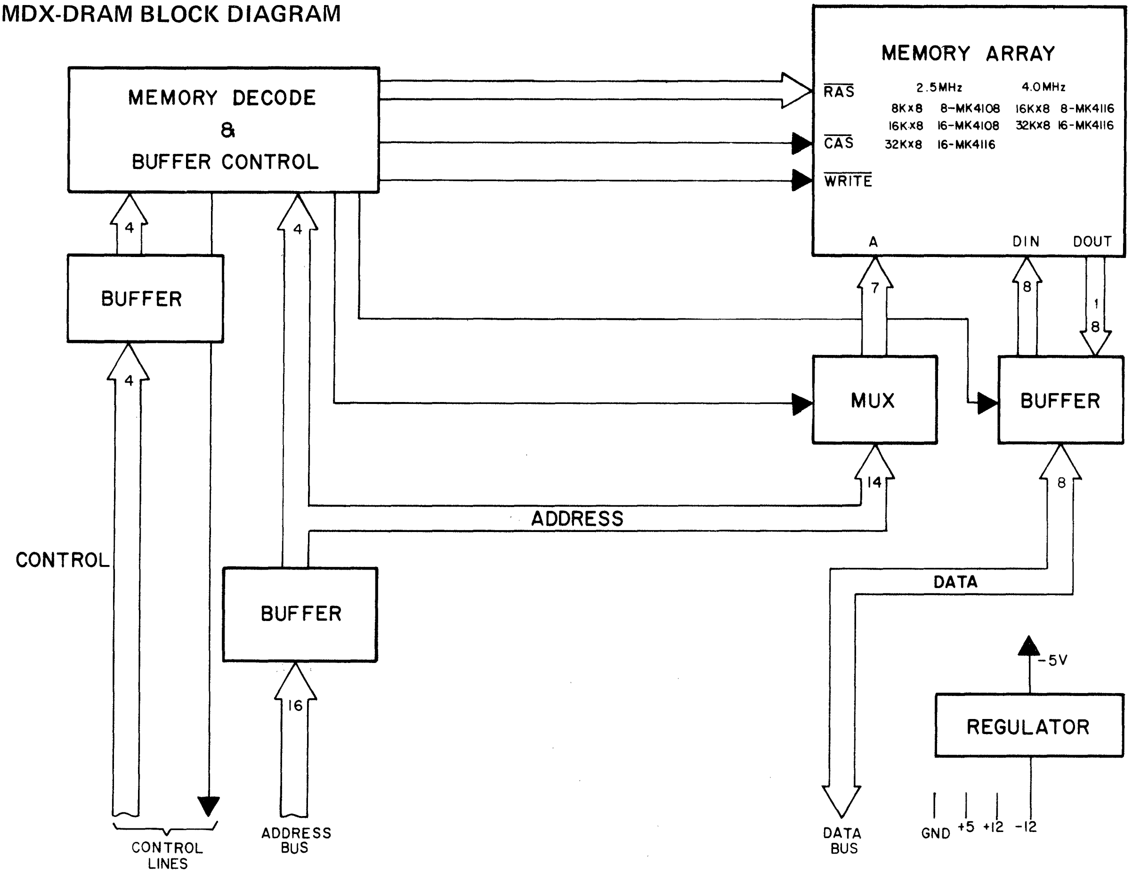
OPERATING TEMPERATURE

0°C to 50°C

MDX-DRAM BOARD



MDX-DRAM BLOCK DIAGRAM



MECHANICAL SPECIFICATIONS

CARD DIMENSION

4.5 in. (11.43cm) high by 6.50 in. (16.51cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD-Z80 BUS	56 pin dual read out	Printed Circuit Viking 3VH28/ ICE5
	0.125 in centers	Wire wrap Viking 3VH28/ 1CND5
		Solder Lug Viking 3VH28/ 1CN5

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
	Module with Operation Manual less mating connectors in the following memory capacities, 2.5MHz versions.	
MDX-DRAM8	8K Bytes (4108's)	MK77750
MDX-DRAM16	16K Bytes (4108's) (4116's)	MK77751 MK77754
MDX-DRAM32	32K Bytes (4116's)	MK77752
	Module with Operations Manual less mating connectors in the following memory capacities, 4.0 MHz version:	
MDX-DRAM16-4	16K Bytes (4116's)	MK77754-4
MDX-DRAM32-4	32K Bytes (4116's)	MK77752-4
MDX-PROTO Data Sheet	MD Series prototyping package	MK79605
AID-80F Data Sheet	Disk based development system for MD Series	MK78568
AIM-80 Data Sheet	Z80 In-circuit emulation module (2.5MHz only)	MK78537

EPROM/UART Module (MDX-EPROM/UART)

FEATURES

- □ 10K x 8 EPROM/ROM (2716's not included)
- □ Serial I/O channel
  RS - 232 and 20 mA interface
  Reader step control for Teletypes
  Baud rate generator 110-19200 Baud
- □ 4MHz version available (MDX-EPROM/UART-4)
- □ STD BUS compatible.

DESCRIPTION

The MD Series and the STD-Z80 BUS were designed to satisfy the need for low cost OEM microcomputer modules. The STD-Z80 BUS uses a motherboard interconnect system concept and is designed to handle any MD Series card type in any slot. The modules for the STD-Z80 BUS are a compact 4.5 x 6.5 inches which provides for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities provide high functionality per module.

The MD Series of OEM microcomputer boards and the STD-Z80 BUS offer the most cost effective system configuration available to the OEM system designer.

MDX-EPROM/UART DESCRIPTION

The MDX-EPROM/UART is one of MOSTEK's complete line of STD-Z80 BUS compatible Z80 microcomputer modules.

Designed as a universal EPROM add-on module for the STD-Z80 BUS, the MDX-EPROM/UART provides the system designer with sockets to contain up to 10K x 8 of EPROM memory (5-2716's) as shown in the Block Diagram.

The EPROM memories can be positioned to start on any 2K boundary within a 16K block of memory via a strapping option provided on the MDX-EPROM/UART.

Included on-board the MDX-EPROM/UART is a fully buffered asynchronous I/O port with a Teletype reader step control. A full duplex UART is used to receive and transmit data at the serial port. Operation and UART options are under software control. Once the unit has been programmed, no further changes are necessary unless there is a modification of the serial data format. Features of the UART include:

Full duplex operation
Start bit verification
Data word size variable from 5 to 8 bits
One or two stop bit selection
Odd, even, or no parity option
One word buffering on both transmit and receive

The MDX-EPROM/UART is also available in a 4MHz version. Circuitry is provided to force one wait state each time on board EPROMs or the UART are accessed.

ELECTRICAL SPECIFICATIONS

WORD SIZE

8 bits for PROM
5 to 8 bits for Serial I/O.

MEMORY ADDRESSING

ROM/EPROM
2K blocks jumper selectable for any 2K boundary within a given 16K boundary of Z80 memory map.

MEMORY CAPACITY

10K bytes of 2716 memory.
(2716's not included)

MEMORY SPEED REQUIRED

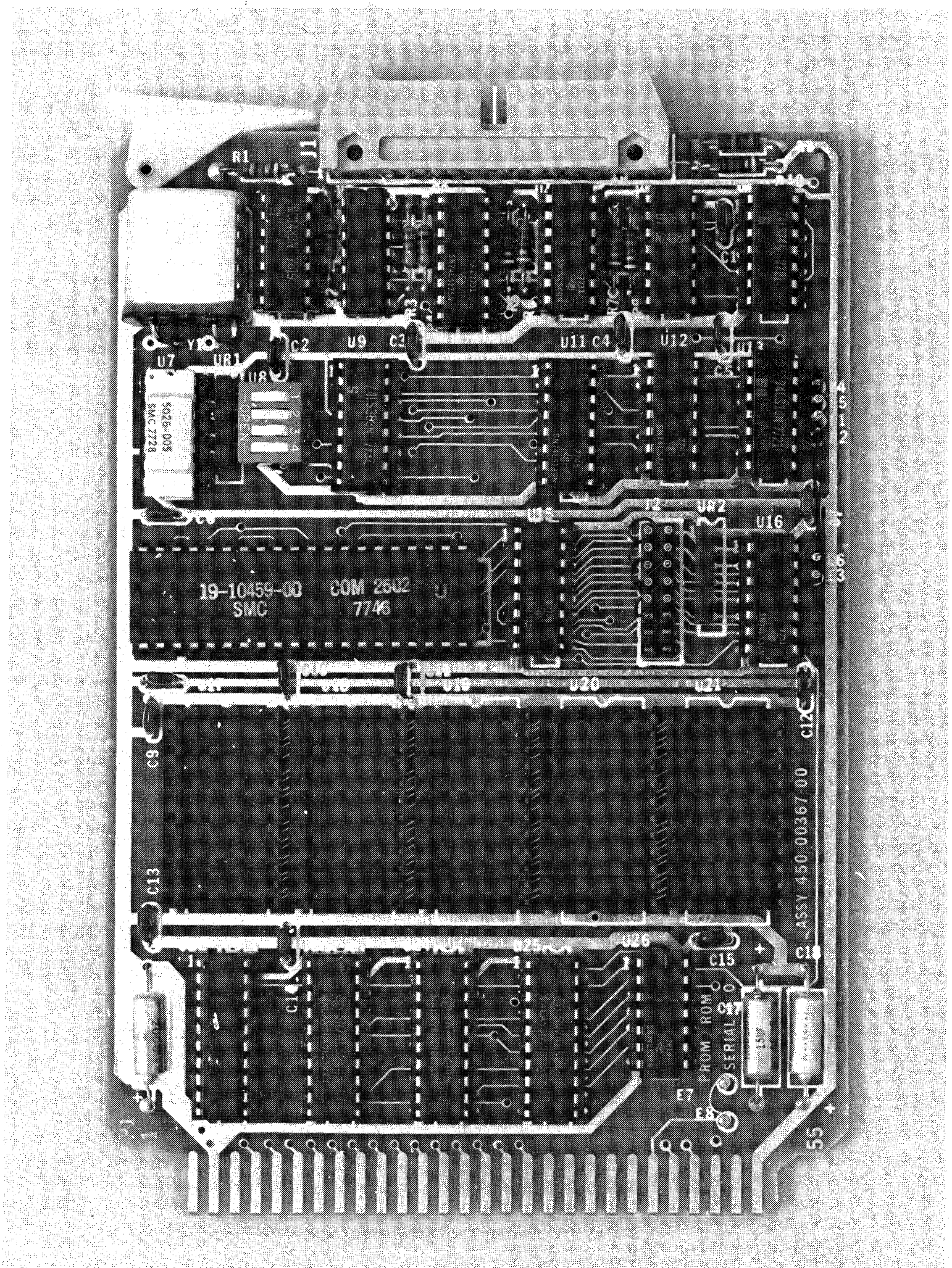
MEMORY	ACCESS TIME	CYCLE TIME
2716*	450ns	450ns

* Single 5 Volt type required

I/O ADDRESSING

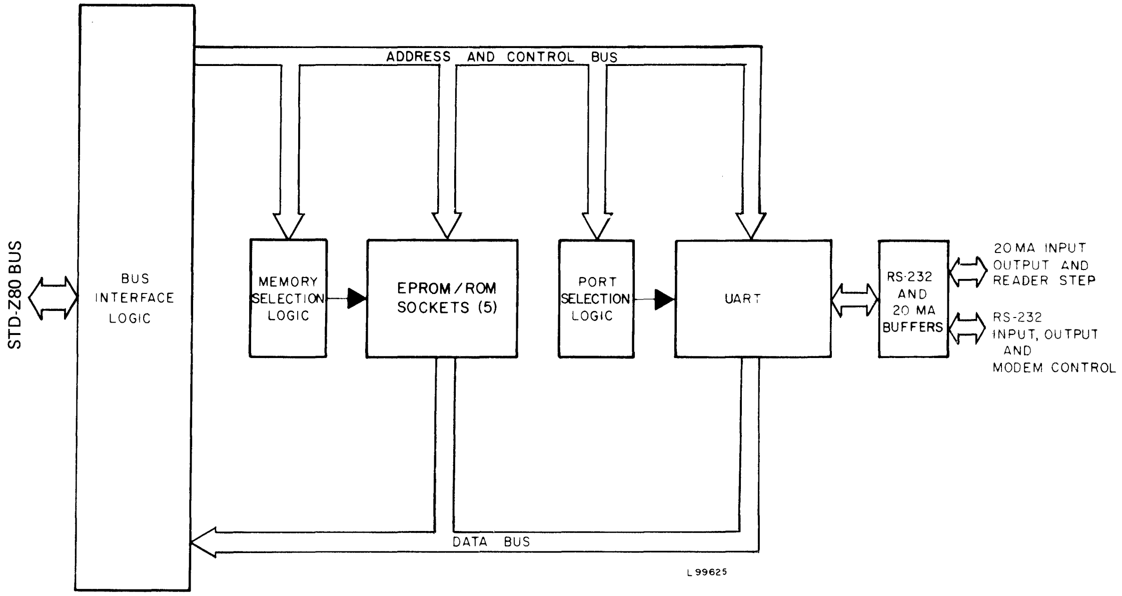
On-board Serial I/O Port
Control Port DDH
Data Port DCH
Modem and Reader Step Control DEH

BOARD PHOTO


R1
J1
C1
C2
U9
C3
U11
C4
U12
U13
C5
U7
VR1
U8
OPEN
1 2 3 4
5026-005
SMC 7728
4
5
1
2
19-10459-00
SMC
COM 2502
7746
J2
VR2
U16
6
3
U20
U21
C12
C9
C13
ASSY 450 00367 00
U25
U26
C14
C15
C17
C18
PROM
ROM
SERIAL
E7
E8
55

]

MDX-EPROM/UART BLOCK DIAGRAM



I/0 TRANSFER RATE

X16 BAUD RATE CLOCK	BAUD RATE (Hz)
1760	110
4800	300
9600	600
19200	1200
38400	2400
76800	4800
153600	9600
317200	19200

SERIAL COMMUNICATIONS CHARACTERISTICS

Asynchronous

- Full duplex operation
- Start bit verification
- Data word size variable from 5 to 8 bits.
- One or two stop bits
- Odd, even, or no parity
- One word buffering on both transmit and receive.

SYSTEM CLOCK

	MIN.	MAX.
MDX-EPROM/UART	250 KHz	2.5 MHz
MDX-EPROM/UART-4	250 KHz	4.0 MHz

SERIAL COMMUNICATIONS INTERFACE

SIGNAL	BUFFERED FOR 20mA Current Loop	RS-232
Transmitted data	Output	Output
Received data	Input	Input
Reader Step Relay (RSR)	Output (40mA)	
Data Terminal Ready (DTR)		Input
Request to Send (RTS)		Input
Carrier Detect (CDET)		Output
Clear to Send (CTS)		Output
Data Set Ready (DSR)		Output

PARALLEL BUS INTERFACE - STD-Z80 BUS COMPATIBLE

Bus	Inputs	One 74LS Load Max
	Outputs	I_{OH} = -15mA min at 2.4 Volts
		I_{OL} = 24mA min at 0.5 Volts

MECHANICAL SPECIFICATIONS

CARD DIMENSIONS

4.5 in. (11.43cm) high by 6.50 in. (16.51 cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD-Z80 BUS	56 pin dual 0.125 in. centers	Printed Circuit Viking 3VH28/ 1CE5 Wire Wrap Viking 3VH28/ 1CND5 Solder Lug Viking 3VH28/ 1CN5
Serial I/O	26 pin dual 0.100 in. grid	Flat Ribbon Ansley 609-2600M Discrete Wires Winchester PGB26A (housing) Winchester 100-70020S (contacts)

POWER SUPPLY REQUIREMENTS

+12 Volts ± 5% at 50 mA max.
-12 Volts ± 5% at 35 mA max.
+5 Volts ± 5% at 1.2 A max.

OPERATING TEMPERATURE RANGE

0° C to +50° C

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MDX-EPROM/UART	Module with Operation Manual Less EPROMs and mating connectors. 2.5MHz version.	MK77753
MDX-EPROM/UART-4	Module with Operation Manual less EPROMs and Mating connectors. 4.0 MHz version.	MK77753-4
	MDX-EPROM/UART Operations Manual only	MK79604
MDX-PROTO Data Sheet	MD Series Prototyping Package	MK79605
AID-80F Data Sheet	Data Sheet of disk based development system for MD Series	MK78568
AIM-80 Data Sheet	Z80 In-Circuit Emulation Module (2.5 MHz only)	MK78537

MOSTEK®

MD SERIES MICROCOMPUTER MODULES

Programmable Input/Output Unit (MDX-PIO)

FEATURES:

- □ Four 8-bit I/O ports with 2 handshake lines per port
- □ All I/O lines fully buffered
- □ I/O lines TTL compatible with provision for termination resistor networks
- □ Jumper options for inverted or non-inverted handshake
- □ Two 8-bit ports capable of true bidirectional I/O
- □ Programmable In only, Out only, or Bidirectional
- □ Output data buffers selectable to provide inverted or non-inverted drive capability
- □ Interrupt driven programmability
- □ Address strap selectable
- □ STD-Z80 BUS Compatible
- □ 4 MHz Option
- □ Fully buffered for MD Series expandability

DESCRIPTION

The MD Series and the STD-Z80 BUS were designed to satisfy the need for low cost OEM microcomputer modules. The STD-Z80 BUS uses a motherboard interconnect system concept and is designed to handle any MD Series card type in any slot. The modules for the STD-Z80 BUS are a compact 4.5 x 6.5 inches which provides for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities providing high functionality per module.

The MD Series of OEM microcomputer boards and the STD-Z80 BUS offer the most cost effective system configuration available to the OEM system designer.

MDX-PIO DESCRIPTION

The parallel I/O controller (MDX-PIO) is a highly versatile unit designed to provide a variety of methods for inputting and outputting data from the MD Series microcomputer system. The system is designed around two Mostek MK3881 Z80-PIO parallel I/O controllers which give four independent 8-bit I/O ports with two handshake (data transfer) control lines per port. The Z80-PIO's are designated PIO1 and PIO2. Each has an I/O port pair designated A and B. Each port pair of each PIO have similar output circuitry. All I/O lines are buffered and have provisions for termination resistors on board. All port lines are brought to two 26 pin connectors; two ports per connector.

Figure 1 illustrates in block diagram from the major functional elements of port pair A and B of PIO 1. These elements can be defined as the resistor termination networks, data buffers, port configuration control, MK3881 PIO, and address decode and data bus buffers. Input and output from the ports are provided through J1, a 26 pin connector. This connector provides data paths for the two ports and their respective handshake signals.

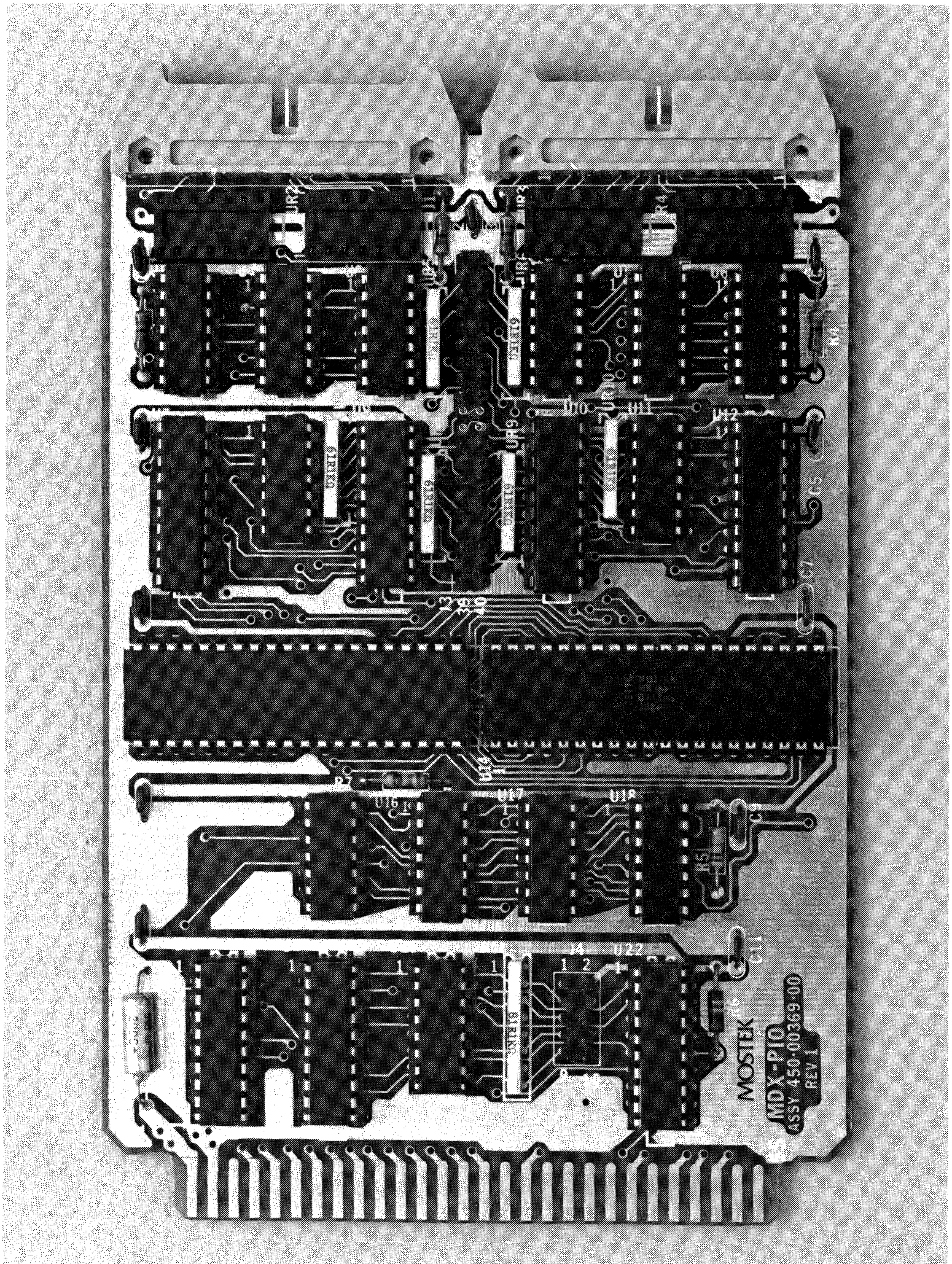
One 14-pin socket is provided per port for resistor dual inline packages so that terminations may be placed on the data lines. A parallel termination is provided for each 8-bit port data line plus the input strobe (STB) handshake line. The MDX-PIO is normally shipped with 1K pullup terminators. In addition to the parallel termination resistors, the ready (RDY) handshake output line is series terminated with a 47 Ω resistor. This is used to damp and reduce reflections on this output line.

Port A and B data bus lines are buffered using quadruple non-inverting transceivors. The buffers can be configured using port configuration jumpers to provide fixed Input, fixed Output or Bidirectional (Port A only) signals. Further the transceivers are configured such that port direction can be selected in 4-bit sections. The transceivers are mounted in sockets so that they can be easily replaced with their complements in order to achieve a polarity change if desired.

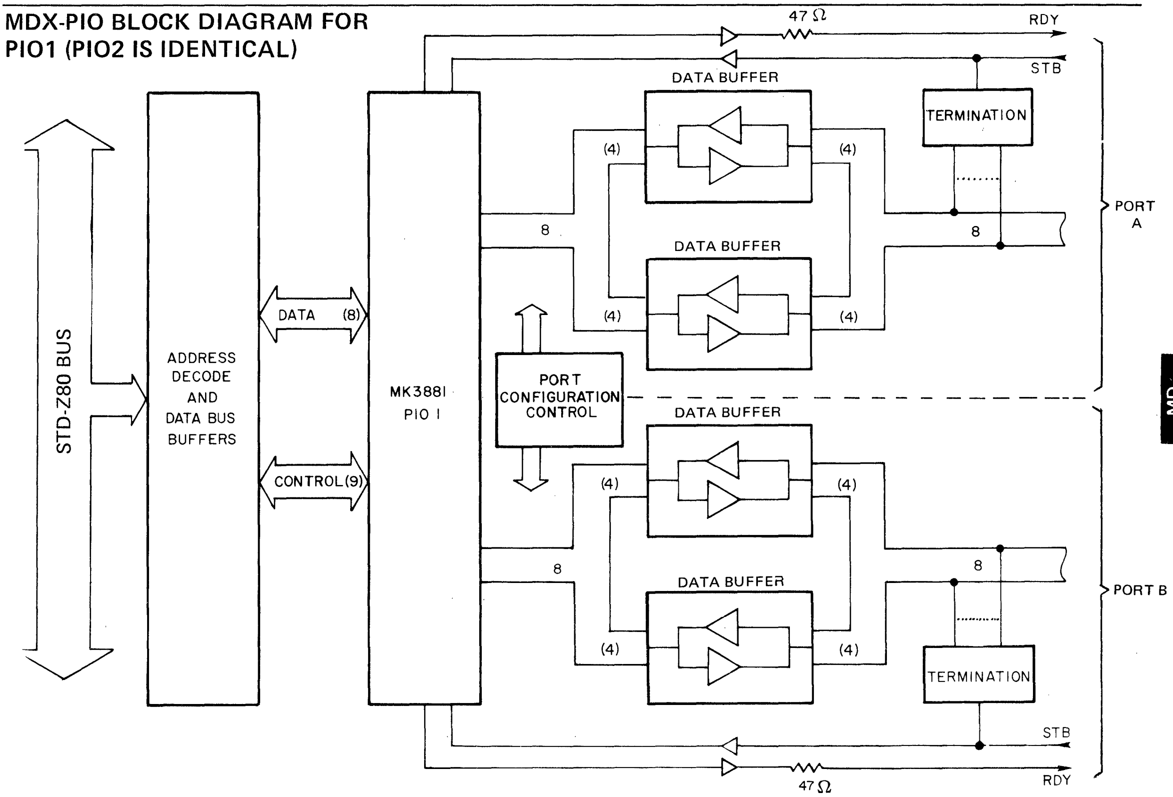
The handshake lines are also fully buffered. The port configuration control provides jumper options to independently control the polarity or "sense" of each handshake line so as to further ease the interfacing between the MDX-PIO and peripheral devices.

The MK3881, PIO parallel I/O controller is the heart of the module. This circuit is a fully programmable two port device which provides a wide range of configuration options. Any one of four distinct modes of operation can be selected for a port. They are byte output, byte input, byte bidirectional (Port A only) and bit control mode. The PIO also automatically generates all handshaking signals in all the above modes.

MDX-PIO BOARD



MDX-PIO BLOCK DIAGRAM FOR PIO1 (PIO2 IS IDENTICAL)



The PIO permits total interrupt control so that full usage of the MDX-CPU1 interrupt capabilities can be utilized during I/O transfers. Also the PIO can be programmed to interrupt the CPU on the occurrence of a specified status condition in a specific peripheral device. The PIO circuit will provide vectored interrupts and maintain the daisy chain priority interrupt logic compatible with the STD BUS.

The address decoding, interface and bus management for the board are performed by the address decode and data bus circuit. Each MDX-PIO port has two addresses, one for Control and one for Data. A total of eight addresses are utilized per board. These addresses are defined in the table below.

TABLE 1

	PIO 1		PIO 2	
	PORT A	PORT B	PORT A	PORT B
Data	$XX0_8$	$XX2_8$	$XX4_8$	$XX6_8$
Control	$XX1_8$	$XX3_8$	$XX5_8$	$XX7_8$

The XX symbols stand for the upper 5 bits of the I/O channel address. These bits are jumper selectable on the MDX-PIO board in order to provide address selectable,fully decoded ports.

The circuitry for the other two ports provided by PIO #2 is identical to PIO #1. The port configuration logic, buffers, termination and pin out on connector J-2 is duplicated for PIO #2.These two ports share the address decode and data bus buffer circuitry with PIO #1. The only differences are in the address decoding as given in the port address table, and PIO #2 is lower priority in the daisy chain interrupt structure.

ELECTRICAL SPECIFICATIONS

WORD SIZE:

Data: 8-bits
I/O Addressing: 8-bits

I/O ADDRESSING:

On-board programmable - See Table 1

I/O CAPACITY:

Four parallel 8-bit ports. On board jumper, selectable in 4 bit bytes as either In only, Out only, or Bidirectional. (Port 1A or 2A only) Automatic handshake provided with each port.

INTERRUPTS

Vectored interrupts generated. Interrupt vector programmable upon initialization. Daisy chained interrupt priority. Selected bit channels can be masked out under program control.

SYSTEM CLOCK

	MIN	MAX
MDX-PIO	250KHz	2.5 MHz
MDX-PIO-4	250KHz	4.0 MHz

I/O DRIVERS

The following line drivers and terminations are all compatible with the I/O driver sockets on the MDX-PIO.

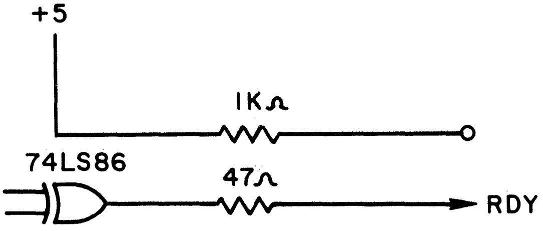
SIGNALS	TYPE	OUTPUT	SINK CURRENT (mA)
Address, Data Bus & Control	74LS245	NI Tri-State Bidirectional	24
I/O Ports 1A and 2A	*74LS244	NI Tri-State Bidirectional	24
	74LS241	I Tri-State Bidirectional	24
I/O Ports 1B and 2B	*74LS243	NI Tri-State Bidirectional	24
	74LS242	NI Tri-State Bidirectional	24
Handshake: RDY	74L586	I/NI (strap selectable)	8

Note: I = inverting N I = non-inverting

* These chips are supplied with the board. They may be exchanged with the other unit listed to provide the alternate signal polarity.

TERMINATORS:

1K ohm resistors on all I/O port lines.



PARALLEL BUS INTERFACE-STD-Z80 BUS COMPATIBLE

Inputs	One 74LS Load Max.
Bus Outputs	I_{OH} = -15mA min. at 2.4 volts I_{OL} = 24mA min. at 0.5 volts

POWER SUPPLY REQUIREMENTS

+5 volts ± 5% at 1.1A max.

OPERATING TEMPERATURE RANGE

0°C to 50°C

MECHANICAL SPECIFICATIONS

CARD DIMENSIONS

4.5 in. (11.43cm) high by 6.50 in. (16.51cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD-Z80 BUS	56 pin dual	Printed Circuit Viking 3VH28/1CE5
	0.125 in. centers	Wire Wrap Viking 3VH28/1CND5
		Solder Lug Viking 3VH28/1CN5
Parallel I/O	26 pin dual 0.100 in. center	Flat Ribbon Ansley 609-2600M Discrete Wires Winchester PGB26A (housing) Winchester 100-70020S (contacts)

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MDX—PIO	Module with Operation Manual less mating connectors. 2.5MHz version	MK77650
MDX-PIO-4	Module with Operation Manual less mating connectors. 4.0 MHz version	MK77650-4
	MDX-PIO Operations Manual only	MK79606
MDX-PROTO Data Sheet	MD Series prototyping package data sheet	MK79605
AID-80F Data Sheet	Data Sheet of disk based development system for MD Series	MK78568
AIM-80 Data Sheet	Z80 In-circuit Emulation module (2.5 MHz only)	MK78537

MOSTEK®

MD SERIES MICROCOMPUTER MODULES

Serial Input/Output Module (MDX-SIO)

MD Series Expand

FEATURES

- □ Two independent full-duplex channels
- □ Independent programmable Baud rate clocks
- □ Asynchronous data rates - 110 to 19.2K bits per second
- □ Receiver data registers quadruply buffered
- □ Transmitter data registers double buffered
- □ Asynchronous operation
- □ Binary synchronous operation
- □ HDLC or IBM SDLC operation
- □ Both CRC-16 and CRC-CCITT (-0 and -1) hardware implemented
- □ Modem control
- □ Operates as DTE or DCE
- □ Serial input and output as either RS-232 or 20mA current loop
- □ Current loop optically isolated
- □ Current loop selectable for either active or passive mode
- □ Address programmable
- □ 4 MHz option
- □ Compatible with STD-Z80 BUS

DESCRIPTION

The MD Series and the STD-Z80 BUS were designed to satisfy the need for low cost OEM microcomputer modules. The STD-Z80 BUS uses a motherboard interconnect system concept and is designed to handle any MD Series card type in any slot. The modules for the STD-Z80 BUS are a compact 4.5 x 6.5 inches which provides for system partitioning by function (RAM, EPROM, I/0). This smaller module size makes system packaging easier while increasing MOS-LSI densities providing high functionality per module.

The MD Series of OEM microcomputer boards and the STD-Z80 BUS offer the most cost effective system configuration available to the OEM system designer.

MDX-SIO DESCRIPTION

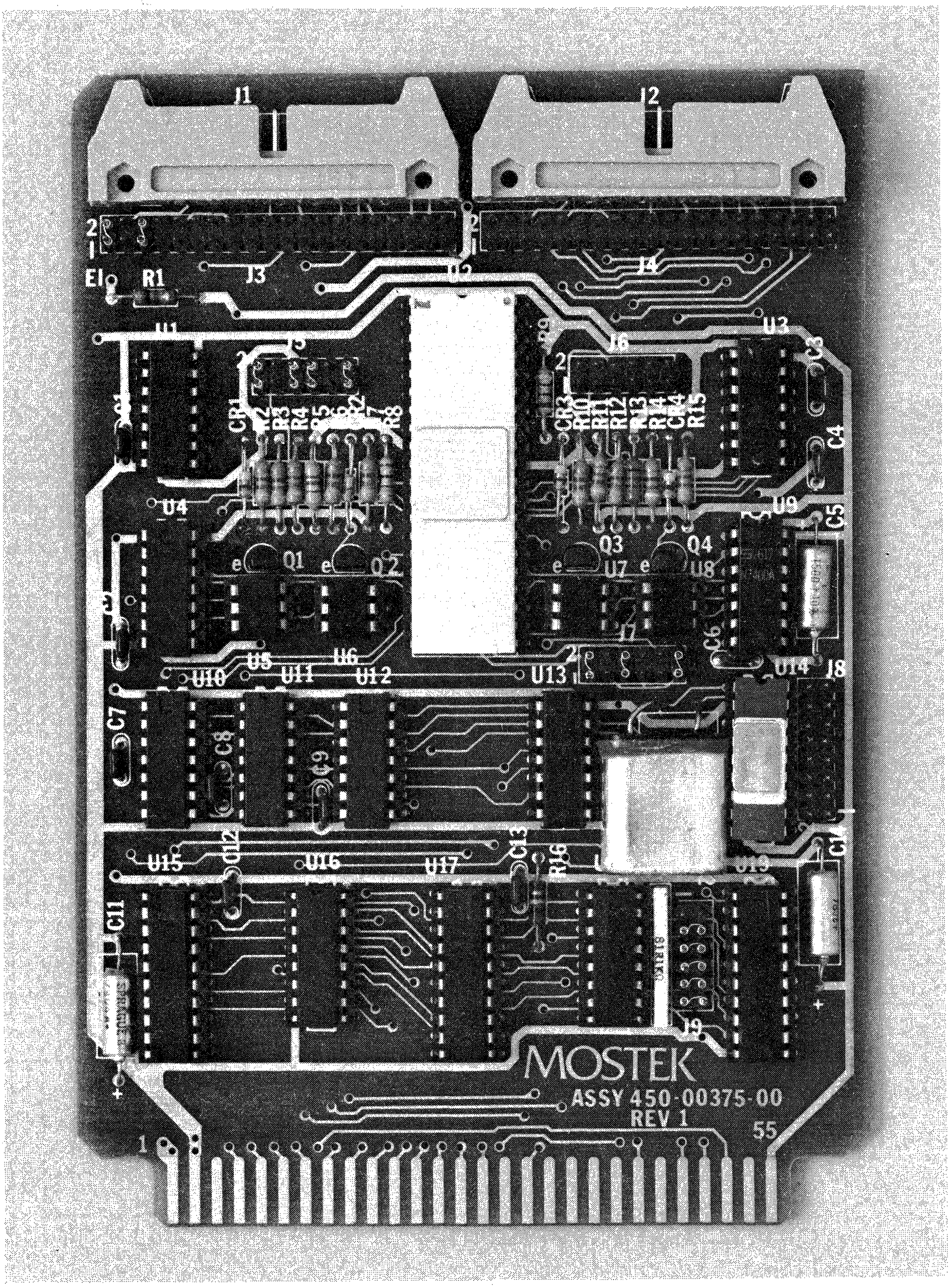
The Serial Input/Output Module, MDX-SIO, is designed to be a multiprotocol asynchronous or synchronous I/O module for the STD-Z80 Bus. The module is designed around the Mostek MK3884 Z80-SIO which provides two full duplex, serial data channels. Each channel has an independent programmable baud rate clock generator to increase module flexibility. The MDX-SIO is capable of handling asynchronous, synchronous, and synchronous bit oriented protocols such as IBM BiSync, IBM SDLC, HDLC and virtually any other serial protocol. It can generate CRC codes in any synchronous mode and can be programmed by the CPU for any traditional asynchronous format. The serial input and output data are fully buffered and are provided at the connector as either a 20mA current loop or RS-232-C levels. A modem control section is also provided for handshaking and status. The MDX-SIO module can be jumper configured as a data terminal (DTE) or as a modem (DCE) in order to facilitate a variety of interface configurations.

Figure 1 is a block diagram of the MDX-SIO module. It consists of five main elements. They are the channel configuration headers, line drivers and receivers, MK3884 Z80-SIO, programmable Baud rate generators, and address decode and data bus buffers. Input and output to the board is provided via two 26 pin connectors. One connector is dedicated for each channel.

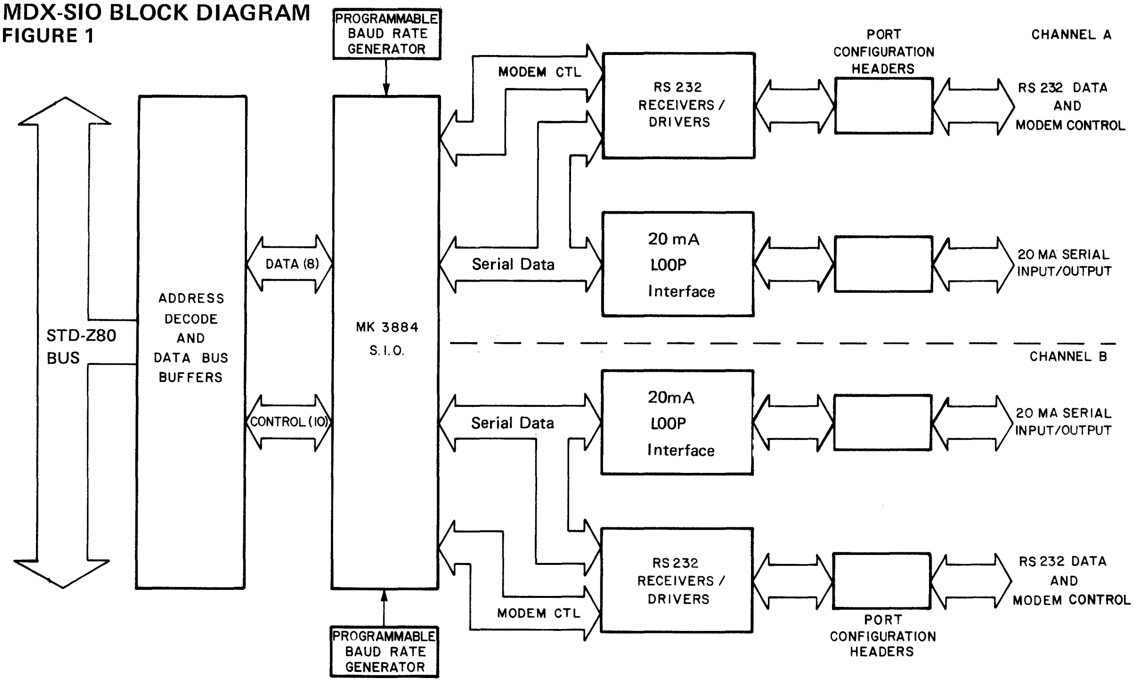
Several features are available as options that are selected via the channel configuration header. The headers are used to select the orientation of the data communication interface and the mode of the 20mA current loop. The MDX-SIO can be selected to act as either a terminal or processor (Data Terminal Equipment DTE) or as the modem (Data Communications Equipment - DCE). The header allows reconfiguration of both data interchange and modem control signals. This allows increased flexibility necessary to link different hardware elements in OEM data link systems and networks. The module is shipped from the factory wired as a DTE interface.

The MDX-SIO has different selectable options for the 20mA current loop. The receiver and transmitter functions can be reconfigured on the module to allow for reorientation of these signals. Also the receive and transmit circuits can be selected to function in either an active or passive mode. In the active mode, the MDX-SIO module provides the 20mA current source. In the passive mode, the module requires that the loop current be provided. The latter is the same mode as that of a Teletype.

MDX-SIO BOARD



MDX-SIO BLOCK DIAGRAM
FIGURE 1


PROGRAMMABLE BAUD RATE GENERATOR
MODEM CTL
RS 232 RECEIVERS / DRIVERS
PORT CONFIGURATION HEADERS
CHANNEL A
RS 232 DATA AND MODEM CONTROL
DATA (8)
Serial Data
20 mA LOOP Interface
20 MA SERIAL INPUT/OUTPUT
ADDRESS DECODE AND DATA BUS BUFFERS
STD-Z80 BUS
MK 3884 S.I.O.
CHANNEL B
CONTROL (IO)
Serial Data
20mA LOOP Interface
20 MA SERIAL INPUT/OUTPUT
RS 232 RECEIVERS / DRIVERS
RS 232 DATA AND MODEM CONTROL
MODEM CTL
PORT CONFIGURATION HEADERS
PROGRAMMABLE BAUD RATE GENERATOR

An EIA and 20mA current loop interface circuit is used to provide the necessary level shifting and signal conditioning between the MK3884 Z80-SIO and the connector. These line drivers and receivers provide the correct electrical signal levels, slew rate and impedance for interfacing RS-232C and 20mA current loop peripherals. Additionally, optical isolation is provided for both transmit and receive circuits in the 20mA current loop mode.

The Mostek MK3884 Z80-SIO is the central element of this module. This device is a multifunction component designed to satisfy a wide variety of serial data communications requirements in microcomputer systems. Its basic role is that of a serial to parallel, parallel to serial converter/controller but within that role it is configured by software programming so that its function can be optimized for a given serial data communications application. The MK3884 provides two independent full duplex channels; A and B.

- Asynchronous operation (Channel A and B)
  - 5, 6, 7, or 8 bits/character
  - 1, 1 ½ or 2 stop bits
  - Even, odd or no parity
  - x1, x16, x32 and x64 clock modes
  - Break generation and detection
  - Parity, Overrun and Framing error detection

- Binary Synchronous operation (Channel A only)
  - One or two Sync characters in separate registers
  - Automatic Sync character insertion
  - CRC generation and checking

- HDLC or IBM SDLC operation (Channel A only)
  - Automatic Zero insertion and deletion
  - Automatic Flag Insertion
  - Address field recognition
  - I-Field residue handling
  - Valid receive messages protected from overrun
  - CRC generation and checking

The MK3884 also provides modem control inputs and outputs as well as daisy chain priority interrupt logic. Eight different interrupt vectors are generated by the SIO in response to various conditions affecting the data communications channel transmission and reception.

Address decoding, STD-Z80 BUS interface and bus management for the module are performed by the Address Decode and Data Bus circuit. The MDX-SIO contains command registers that are programmed to select the desired operational mode. The addressing scheme is as follows:

XXXXX 00	Channel A Data
XXXXX 01	Channel A Control Status
XXXXX 10	Channel B Data
XXXXX 11	Channel B Control Status

The X indicates the binary code necessary to represent which of the 64 port addresses is selected by on board strapping.

Each channel has an individual programmable Baud rate generator. The X1 multiplier on the Z80-SIO must be used in the synchronous mode. The X16, X32, or X64 Z80-SIO clock rate can be specified for the asynchronous mode. Table 1 indicates the possible Baud rates available for both operation modes with the Z80-SIO Data Rate multipliers.

Figure 1

BAUD RATE (Hz)

SYNCHRONOUS	ASYNCHRONOUS		
X1	X16	X32	X64
800	50	25	12.5
1200	75	37.5	18.75
1760	110	55	27.50
2152	134.5	67.25	33.63
2400	150	75	37.50
4800	300	150	75
9600	600	300	150
19200	1200	600	300
28800	1800	900	450
32000	2000	1000	500
38400	2400	1200	600
57600	3600	1800	900
76800	4800	2400	1200
115200	7200	3600	1800
153600	9600	4800	2400
307200	19200	9600	4800

ELECTRICAL SPECIFICATIONS

WORD SIZE

Data: 8-bits
I/0 addressing: 8-bits

I/0 ADDRESSING

On board upper six bits programmable

I/0 CAPACITY

Serial - Two full duplex serial ports. Channel A is capable of synchronous and asynchronous operation. Channel B is asynchronous only. Special control registers and circuitry to permit implementation of SDLC, BiSync, MonoSync, HDLC. Other formats can be programmed on Channel A only.

SERIAL BAUD RATES

See Table 1

INTERRUPTS

Generates vectored interrupts to 8 different locations corresponding to conditions within both channels. Interrupt vector location programmable. Daisy chained priority hardware interrupt circuitry.

SYSTEM CLOCK

	MIN	MAX
MDX-SIO	250KHz	2.5 MHz
MDX-SIO-4	250KHz	4.0 MHz

SERIAL COMMUNICATION INTERFACE

SIGNAL	20mA LOOP	RS-232-C
Transmitted data	Output	Output
Received data	Input	Input
Data Terminal Ready (DTR)		Input/Output
Request to Send (RTS)		Input/Output
Clear to Send (CTS)		Output/Input
Carrier Detect (CDET)		Output/Input

PARALLEL BUS INTERFACE - STD-Z80 BUS COMPATIBLE

Inputs	One 74LS load max.
Bus Outputs	I_{OH} = -15mA min at 2.4 Volts I_{OL} = 24mA min at 0.5Volts

POWER SUPPLY REQUIREMENTS

+12 volts ± 5% at 72 mA max.
-12 volts ± 5% at 46 mA max.
+5 volts ± 5% at 650 mA max.

OPERATING TEMPERATURE

0°C to 50°C

MECHANICAL SPECIFICATIONS

CARD DIMENSIONS

4.5 in. (11.43cm) high by 6.50 in. (16.51cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD-Z80 BUS	56 pin	Printed Circuit Viking 3VH28/1CE5
	0.125 in.centers	Wire Wrap Viking 3VH28/1CND5
		Solder Lug Viking 3VH28/1CN5

CONNECTORS Cont'd.

SERIAL I/O	26 pin 0.100 in. center	Flat Ribbon Ansley 609-2600M Discrete Wires: WINCHESTER PGB26A (housing) WINCHESTER 100-70020S (contacts)

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MDX-SIO	Dual channel, Full-Duplex Serial I/O Module less mating connectors with Operations Manual. 2.5MHz version.	MK77651
MDX-SIO-4	Module with Operations Manual less mating connectors. 4.0MHz version	MK77651-4
	MDX-SIO Operations Manual	MK79608
MDX-PROTO Data Sheet	MD Series Prototyping Package	MK79605
AID-80F Data Sheet	Disk based development system for MD series	MK78568
AIM-80 Data Sheet	Z80 In-circuit Emulation module (2.5 MHz only)	MK78537

MOSTEK®

MD SERIES MICROCOMPUTER MODULES

Z80 Microcomputer Debug Module (MDX-DEBUG)

HARDWARE FEATURES

- □ STD-Z80 BUS compatible
- □ 4 MHz version available
- □ Serial I/O Channel
- □ 10K bytes of ROM contain the following firmware: DDT-80, ASMB-80

DEBUGGER FEATURES

- □ Z80 Operating System with debug capability
- □ Channelized I/O for versatility
- □ I/O peripheral drivers supplied
- □ ROM based

TEXT EDITOR FEATURES

- □ Input and modification of ASCII Text
- □ Line and character editing
- □ Alternate command buffers for pseudo-macro command capability
- □ ROM based

ASSEMBLER FEATURES

- □ Assembles all Z80 mnemonics
- □ Object output in industry standard hexadecimal format extended for Relocatable and Linkable Programs
- □ Over fifteen pseudo-ops
- □ Two pass assembly
- □ ROM based

LINKING LOADER FEATURES

- □ Loads into memory both relocatable and non-relocatable object output of the assembler
- □ Loads Relocatable modules anywhere in memory
- □ Automatically provides linkage of global symbols between object modules as they are loaded
- □ Prints system load map
- □ ROM based

MD SERIES GENERAL DESCRIPTION

The MD Series and the STD-Z80 BUS were designed to satisfy the need for low cost OEM microcomputer modules. The STD-Z80 BUS uses a motherboard interconnect system concept and is designed to handle any MD Series card type in any slot. The modules for the STD-Z80 BUS are a compact 4.5 x 6.5 inches which provides for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities provide high functionality per module.

The MD Series of OEM microcomputer boards and the STD-Z80 BUS offer the most cost effective system configuration available to the OEM system designer.

HARDWARE DESCRIPTION

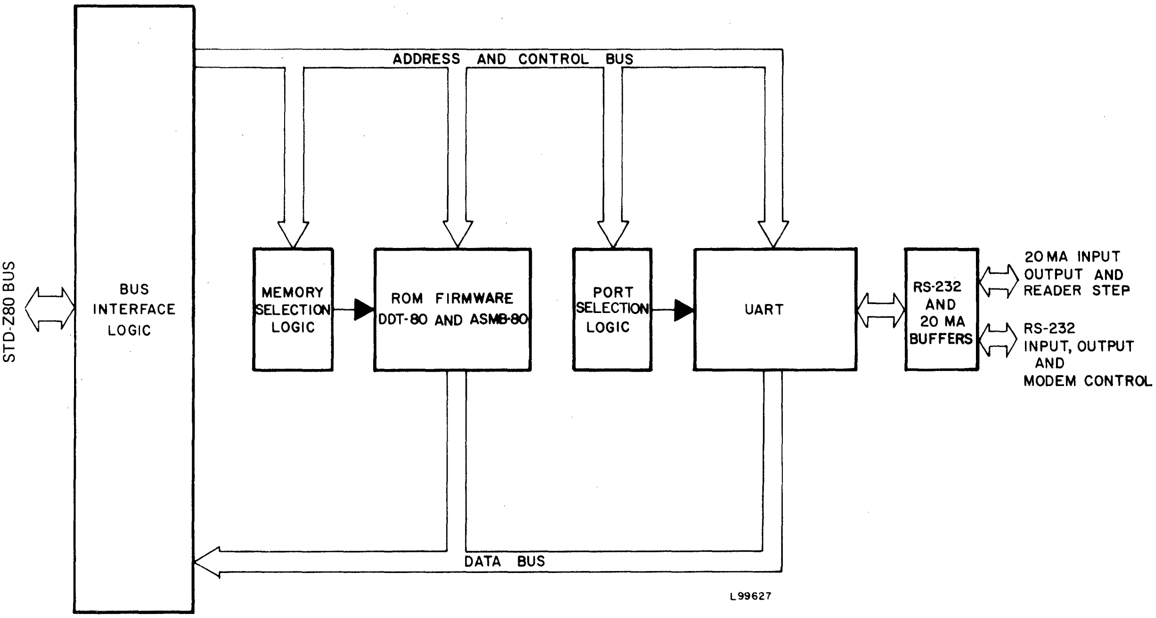
The MDX-DEBUG module has sockets for 10K bytes of masked ROM that are filled with a Z80 firmware package (DDT-80/ASMB-80). This module has a STD-Z80 BUS interface and is available in both 2.5MHz and 4.0MHz versions. Included on-board is a fully buffered asynchronous I/O port capable of 110-19200 Baud rates. Serial data interfaces are available for 20mA current loop (with reader step control) and RS-232. The on-board Baud Rate Generator is selectable to all common Baud rates from 110 to 19,200 Baud.

FIRMWARE DESCRIPTION

DEBUGGER DESCRIPTION

DDT-80 is the Operating System for the MDX-DEBUG Module. It resides in a 2K ROM (MK34000 series) resident on the MDX-DEBUG Module. It provides the necessary tools and techniques to operate the system, i.e., to efficiently and conveniently perform the tasks necessary to develop microcomputer software. DDT-80 is designed to support the user from initial design through production testing. It allows the user to display and update memory, registers, and ports, load and dump object files, set breakpoints, copy blocks of memory, and execute programs.

MDX-DEBUG BLOCK DIAGRAM



DDT-80 COMMAND SUMMARY

M s - Display and/or update the contents of memory location s.

M s, f - Tabulate the contents to memory locations s through f.

P s - Display and/or update the content of I/O port s.

D s, f - Dump the contents of memory locations s through f in a format suitable to be read by the L command.

L - Load, into memory, data which is in the appropriate format.

E s - Transfer control from DDT-80 to a user's program starting at location s.

H - Perform 16 bit hexadecimal addition and/or subtraction.

C s, f, d - Copy the contents of memory locations s through f to another location in memory starting at location d.

B s - Insert a breakpoint in the user's program (must be in RAM) at location s which transfers control back to DDT-80. This allows the user to intercept his program at a specific point (location s) and examine memory and CPU registers to determine if this program is working correctly.

R - Display the contents of the user registers.

The s, f, and d represent start, finish, and destination operands required for each command.

MEMORY, PORT AND REGISTER COMMANDS (M, P, R)

The M, P, and R commands provide the means for displaying the contents of specified memory locations, port addresses, or CPU registers. The M and P commands sequentially access memory locations or ports and display their contents. The user has the option of updating the content of the memory location or port. (Note some ports are output only and their contents cannot be displayed). The M command also gives the user access to the CPU registers through an area in RAM called the Register Map (discussed in the Execute, Breakpoint section below).

The M and R commands are used to tabulate blocks of memory locations (M) or the CPU registers (R). The M command will accept two operands, the starting and ending address of the memory block to be tabulated. The R command will accept either no operand or one. If no operand is specified, the CPU registers will be displayed without a heading. If an operand is specified then a heading which labels the registers contents will be displayed as well.

EXECUTE AND BREAKPOINT (E, B)

The E command is used to execute all programs, including aids such as the Assembler. The B command is used to set a breakpoint to exit from a program at some predetermined location for debugging purposes. At the instant of a breakpoint exit, the contents of all

CPU register are saved in a designated area of MDX-DEBUG RAM called the Register Map. In the Register Map, the register contents may be examined or modified using the M command and a predefined mnemonic (or absolute address) of the storage location for that register (example :PC, :A, . . ., :SP). The Register Map is also used to initialize the CPU registers whenever execution is initiated or resumed. Thus the E and B commands can be used together to initialize, execute, and examine the results of individual program segments.

The B command gives the user the option of having all CPU registers displayed when the breakpoint is encountered. This is done by entering a second operand to the B command. Otherwise DDT-80 defaults to displaying the PC and AF registers. When all CPU registers are displayed, the format is the same as for the R command previously discussed.

LOAD, DUMP, AND COPY (L, D, C)

The L and D commands load and dump object files through the object I/O channel in standard Intel Hex format. Checksums are used for error detection, and the addresses of questionable blocks are typed automatically while loading.

The C command will copy the contents of the memory block specified to another block of memory. There are no restrictions on the direction of the copy or on whether the blocks overlap.

HEXADECIMAL ARITHMETIC (H)

The H command is a dummy command used to allow hexadecimal addition and subtraction for expression evaluation without performing any other operation.

DDT-80 I/O CAPABILITIES

DDT-80 specifies I/O channels, designated 'Console', 'Object', and 'Source', to which any suitable devices may be assigned. The Channel Assignment Table is located in MDX-DRAM where it may be examined or modified using the M command. The table addresses correspond to the I/O channels and the table contents correspond to the addresses of the peripheral driver routines. A channel which has a device assignment may have that device assignment changed using the M command. This is accomplished by merely modifying the table contents of that channel's table address to correspond to the new peripheral driver routine. A set of peripheral driver routines is supplied and listed below. This scheme also allows the user to write a driver routine for his own peripheral, load it into memory, and easily configure that peripheral into the system.

DDT-80 I/O PERIPHERAL DRIVERS

1. A serial input driver (usually a keyboard).
2. A serial output driver (usually a CRT or teletype typehead).
3. A serial input driver which sends out a reader step signal (usually a teletype reader).
4. A serial output driver which forces a delay after a carriage return (usually a Silent 700 typehead).
5. A parallel input driver (usually for high speed paper tape input).
6. A parallel output driver (usually for high speed paper tape output).
7. A parallel output driver (usually for a line printer).

TEXT EDITOR DESCRIPTION

The Text Editor permits random access editing of ASCII character strings. It can be used as a line or character oriented editor. Individual characters may be located by position or context. The Editor works on blocks of characters which are typically read into memory from magnetic tape or paper tape. Each edited block can be output to magnetic tape or paper tape after editing is completed. While the primary application for the Text Editor is in editing assembly language source statements, it may be applied to any ASCII text delimited by "carriage returns".

The Editor has a macro command processing option. Up to two sets of commands may be stored and processed at any time during the editing process.

All I/O is done via the DDT-80 channels. The Editor can be used with the MOSTEK ASMB-80 Assembler and Loader to edit, assemble, and load programs in memory without the need for external media for intermediate storage.

The following commands are recognized by the Text Editor:

An -	Advance record pointer n records
Bn -	Backup record pointer n records
Cn dS1dS2d -	Change string S1 to string S2 for n occurrences
Dn -	Delete next n records
E -	Exchange current record with records to be inserted
I -	Insert records
Ln -	Go to line number n
Mn -	Enter command buffers (pseudo-macro)
N -	Print top, bottom, and current line number
Pn -	Punch n records from buffer
R -	Read source records into buffer
Sn dS1d -	Search for nth occurrence of string S1

ASSEMBLER DESCRIPTION

The Assembler reads Z80 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The assembly listing shows address, machine code, statement number, and source statement. The object code is in industry standard hexadecimal format modified for relocatable, linkable assemblies.

The Assembler supports conditional assemblies, global symbols, relocatable programs, and a printed symbol table. It can assemble any length program, limited only by a symbol table size which is user selectable. Expressions involving addition and subtraction are allowed. A global symbol is categorized as "internal" if it appears as a label in the program; otherwise it is an "external" symbol. The printed symbol table shows which symbols are internal and which are external. The Assembler allows the user to select relocatable or non-relocatable assembly via the "PSECT" pseudo-op. Relocation records are placed in the object output for relocatable assemblies (the MOSTEK object format is defined below). The Assembler can be run as a single pass assembler or as a learning tool. (In this mode, global symbols and forward references are not allowed).

The following pseudo-ops are recognized by the Assembler:

ORG -	program origin
EQU -	equate label
DEFL -	define label
DEFM -	define message
DEFB -	define byte
DEFW -	define word
DEFS -	define storage
END -	end statement
NAME -	program name definition
PSECT -	program section definition
GLOBAL -	global symbol definition Supports the following assembler pseudo-ops
EJECT -	eject a page of listing
TITLE -	place heading at top of each page
LIST -	turn listing on
NLIST -	turn listing off

RELOCATING LINKING LOADER DESCRIPTION

The MOSTEK Relocating Linking Loader provides state-of-the-art capability for loading programs into memory by allowing loading and linking of any number of relocatable and non-relocatable object modules. Non-relocatable modules are always loaded at their starting address as defined by the ORG pseudo-op during assembly. Relocatable object modules can be positioned anywhere in memory at an offset address.

The Loader automatically links and relocates global symbols which are used to provide communication or linkage between program modules. As object programs are loaded, a table containing global symbol references and definitions is built up. At the end of each module, the loader resolves all references to global symbols which are defined by the current or a previously loaded module. It also prints on the console device the number of defined global symbols that have been referenced. The symbol table can be printed to list all global symbols and their load address. The number of object modules which can be loaded by the Loader is limited only by the amount of MDX-RAM available for the modules and the symbol table. Space for the symbol table is allocated dynamically downward in memory from either the top of memory or from a specified address entered as an operand of the load command.

All I/0 is done via the DDT-80 channels. Assemblies can be done from source statements stored in memory (by the Editor). The object output can be directed to a memory buffer rather than to an external device. Thus, assembly and loading can be done without external storage media.

The Loader prints the beginning and ending address of each module as it is loaded. The transfer address as defined by the END pseudo-ops is printed for the first module loaded. The Loader execute command (E) can be used to automatically start execution at the transfer address.

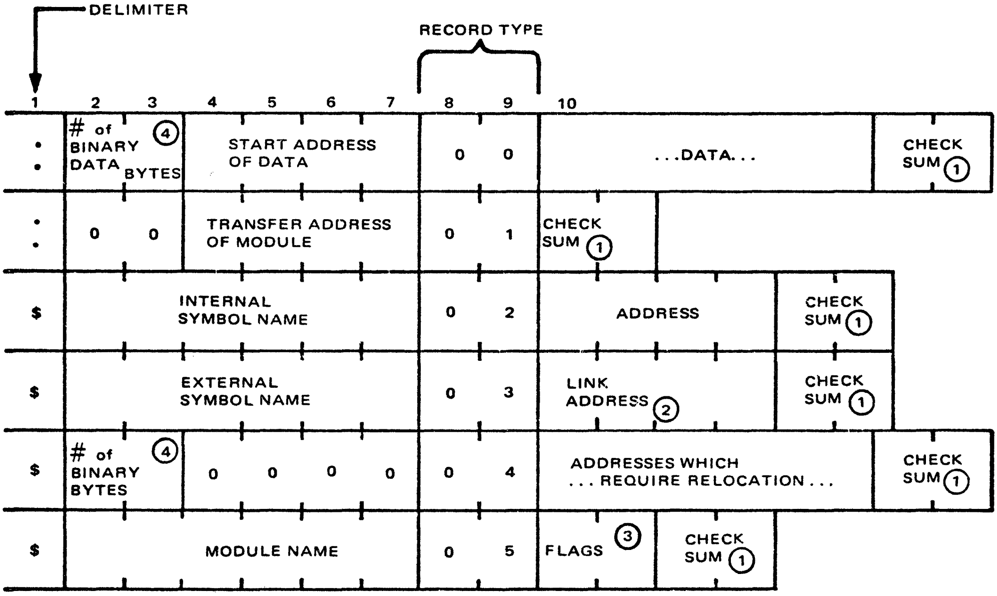
The Loader Commands are the following:

L offset -	load object module at address "offset" plus program origin address
E -	execute loaded program at transfer address of first module
T -	print global symbol table

MOSTEK OBJECT OUTPUT DEFINITION

Each record of an object module begins with a delimiter (colon or dollar sign) and ends with carriage return and line feed. A colon (:) is used for data records and the end-of-file record. A dollar sign ($) is used for records containing relocation information and linking information. All information is in ASCII. Each record is identified by "type". The type is determined by the 8th and 9th bytes of the record which can take the following values:

00	- data
01	- end-of-file
02	- internal symbol
03	- external symbol
04	- relocation information
05	- module definition


DELIMITER
RECORD TYPE
1 2 3 4 5 6 7 8 9 10
• • | # of BINARY DATA BYTES ④ | START ADDRESS OF DATA | 0 0 | . . .DATA. . . | CHECK SUM ①
• • | 0 0 | TRANSFER ADDRESS OF MODULE | 0 1 | CHECK SUM ①
$ | INTERNAL SYMBOL NAME | 0 2 | ADDRESS | CHECK SUM ①
$ | EXTERNAL SYMBOL NAME | 0 3 | LINK ADDRESS ② | CHECK SUM ①
$ | # of BINARY BYTES ④ | 0 0 0 0 | 0 4 | ADDRESSES WHICH . . . REQUIRE RELOCATION . . . | CHECK SUM ①
$ | MODULE NAME | 0 5 | FLAGS ③ | CHECK SUM ①

NOTES:

1. Check Sum is negative of the binary sum of all bytes except delimiter and carriage return/line feed.
2. Link Address points to last address in the data which uses the external symbol. This starts a backward link list through the data records for that external symbol. The list terminates at OFFFFH.
3. The flags are one binary byte. Bit 0 is defined as:
   0 - absolute module
   1 - relocatable module
4. Maximum of 64 ASCII bytes.

ELECTRICAL SPECIFICATIONS

I/0 TRANSFER RATE

X16 Baud Rate Clock	Baud Rate (Hz)
1,760	110
4,800	300
9,600	600
19,200	1,200
38,400	2,400
76,800	4,800
153,600	9,600
307,200	19,200

SERIAL COMMUNICATIONS CHARACTERISTICS

Asynchronous

Full duplex operation
Start bit verification
Data word size variable from 5 to 8 bits.
One or two stop bits
Odd, even, or no parity
One word buffering on both transmit and on receive.

SERIAL COMMUNICATIONS INTERFACE

SIGNAL	BUFFERED FOR: 20mA Current Loop	RS-232
Transmitted data	Output	Output
Received data	Input	Input
Data Terminal Ready (DTR)		Input
Request to Send (RTS)		Input
Carrier Detect (CDET)		Output
Clear to Send (CTS)		Output
Data Set Ready (DSR)		Output
Reader Step relay (RS)	Output (40mA)	

PARALLEL BUS INTERFACE-STD-Z80 BUS COMPATIBLE

Inputs	One 74LS load Max
Bus Outputs	I_{OH} = -15mA min at 2.4 Volts I_{OL} = 24mA min at 0.5 Volts

I/0 ADDRESSING

On-board Serial I/0 Port
Control Port DDH
Data Port DCH
Module and Reader Step Control Port DEH

SYSTEM CLOCK

MDX-DEBUG	1.25MHz	2.5MHz
MDX-DEBUG-4	1.25MHz	4.0MHz

POWER SUPPLY REQUIREMENT

+12 Volts ± 5% at 50 mA max.
-12 Volts ± 5% at 35 mA max.
+5 Volts ± 5% at 1.2 A max.

OPERATING TEMPERATURE

0°C to + 50°C

MECHANICAL SPECIFICATIONS

CARD DIMENSIONS

4.5 in. (11.43cm) high by 6.50 in. (16.51cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD BUS	56 pin dual read out	Printed Circuit Viking 3VH28/1CE5
	0.125 in. centers	Wire Wrap Viking 3VH28/1CND5
		Solder Lug Viking 3VH28/1CN5
Serial I/0	26 pin dual readout 0.100 in. grid	Flat Ribbon Ansley 609-2600M
		Discrete Wires Winchester PGB26A (housing) Winchester 100-70020S (contacts)

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MDX-DEBUG	Module with 10K bytes of firmware and Operations Manual. No mating connectors. 2.5MHz version.	MK77950
MDX-DEBUG-4	Module with 10K bytes of firmware and Operations Manual. No mating connectors. 4.0MHz version	MK77950-4
	MDX-DEBUG Operations Manual only	MK79611
	Program Source Listing of 10K byte firmware package (DDT/ASMB-80) including comments and flow charts. (Available free with purchase of either MDX-DEBUG Module).	MK78536 and MK78534
MDX-PROTO Data Sheet	MD Series Prototyping package	MK79605
AID-80F Data Sheet	Disk based development system for MD Series	MK78568
AIM-80 Data Sheet	Z80 In-circuit emulation module (2.5MHz only)	MK78537

* The DDT-80 and ASMB-80 listings are available directly from MOSTEK by filling out a copy of the Software Licensing Agreement printed on the opposite page of this data sheet and returning it with the appropriate payment of Customer Purchase Order to:

MOSTEK CORPORATION
Microcomputer Systems Div.
1215 West Crosby Road
Carrollton, Texas 75006

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold
on condition that the Purchaser agrees to
the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.
2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.
3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.
4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.
5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.
6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject to this Agreement:

Order Number Description Price*

____

____

____

Ship To: ____________________ Bill To: ____________________

____________________ ____________________

Method of Shipment: ____________________ Customer P.O. Number ____________________
Agreed To: ____________________

PURCHASER
By: ____________________
Title: ____________________
Date: ____________________

MOSTEK CORPORATION
By: ____________________
Title: ____________________
Date: ____________________

* Prices Subject to Change Without Notice

MOSTEK®

MD SERIES MICROCOMPUTER MODULES

Z80 Single Step Module (MDX-SST)

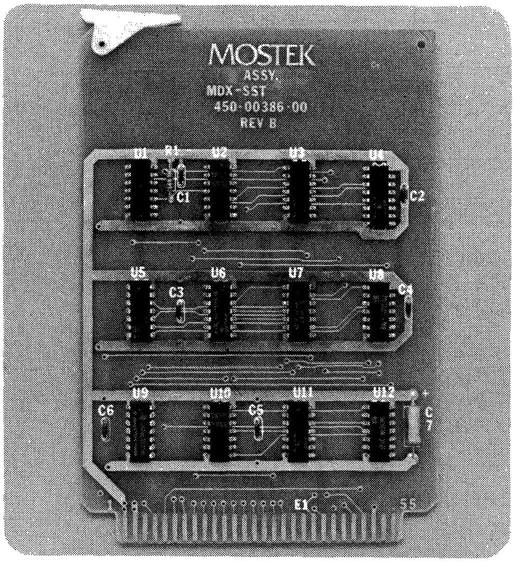
FEATURES

□ Hardware single-step capability
□ Compatible with DDT-80 Operating System
□ STD-Z80 BUS compatible

DESCRIPTION

The MD Series and the STD BUS were designed for low-cost OEM microcomputer modules. The STD BUS uses a motherboard interconnect system concept and is designed to handle any MD Series Card type in any slot. The modules for the STD BUS are a compact 4.5 x 6.5 inches which provide for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier, while increasing MOS-LSI densities provide high functionality per module.

The MD Series of OEM microcomputer boards and the STD BUS offer the most cost-effective system configuration available to the OEM system designer.



MDX-SST DESCRIPTION

The MOSTEK MDX-SST was designed to enhance the hardware and software debug capability for MD Series systems. The use of the MDX-SST with the MDX-CPU1 and MDX-DEBUG boards allows the user to single-step instructions through RAM and/or EPROM/ROM with the capability of displaying all of the MDX-CPU1 registers on each instruction execution.

The MDX-SST board is implemented using the MDX-CPU1's nonmaskable interrupt and is controlled by firmware from the keyboard. When the command to single step an instruction is given, the sequence of events is the same as executing a program except that a "1" is output to the single step control port (DF$_H$) instead of a "0". The circuit decodes the double M1 instructions (CB$_H$, DD$_H$, ED$_H$, or FD$_H$) and M1 is used to clock a shift register circuit which (if a "1" is output to port DF$_H$) generates a nonmaskable interrupt at the start of the instruction to be single stepped. The nonmaskable interrupt saves the address of execution on the stack and causes the next instruction to be fetched from address E066$_H$. The shift register is clocked twice after the nonmaskable interrupt, causing the signal DEBUG to go low, forcing "E" on the most significant address lines, and causing the instruction to be fetched from the E066$_H$ in the operating system DDT-80. The operating system then jumps to E069$_H$, clears the debug flip-flop by reading PORT DF$_H$, saves the MDX-CPU1 registers in the MDX-CPU1 scratch RAM, and waits for the next command.

The single-step command is implemented in DDT-80 which resides on the MDX-DEBUG board and has the following format:

S COMMAND, Single-step

This command allows the user to start single-stepping from a given location for a given number of instructions and to display the CPU registers after each step.

Format:

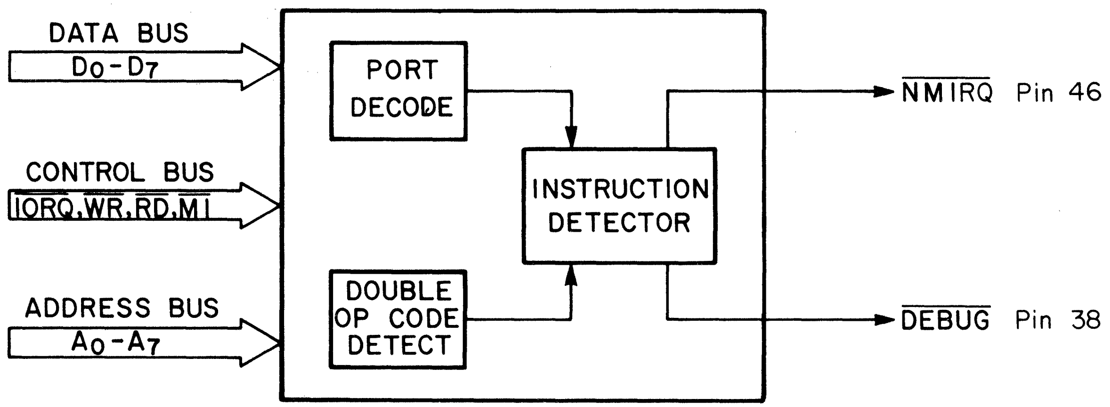
.S aaaa,nn,b(cr)	start single-stepping at location aaaa for nn steps or instructions. If b=0, display only the PC and AF registers, if b≠0, display all the CPU registers.
.S aaaa,nn (cr)	the same as above with b = 0 assumed.
.S aaaa (cr)	the same as above with nn=1 and b=0 assumed.
.S (cr)	the same as above with nn=1 and b=0 assumed; aaaa is set equal to the contents of the user's PC.

The use of the MDX-SST board requires the MDX-CPU1 and the MDX-DEBUG.

ELECTRICAL SPECIFICATIONS

PORT ADDRESS (HEX)
DF

MDX-SST BLOCK DIAGRAM



SYSTEM CLOCK

MIN	MAX
500KHz	4.00MHz

PARALLEL BUS INTERFACE

STD-Z80 BUS COMPATIBLE

POWER SUPPLY REQUIREMENTS

+5Vdc @ 85mA

OPERATING TEMPERATURE

0°C to 50°C

MECHANICAL SPECIFICATIONS

CARD DIMENSIONS

4.5 in (11.43cm) high by 6.50 in. (16.51cm) long
0.48 in (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD BUS	56 pin 0.125 in. centers	Printed Circuit Viking 3VH28/1CE5

CONNECTOR CONTINUED

Wire Wrap Viking 3VH28/1CND5

Solder Lug Viking 3VH28/1CN5

ORDERING INFORMATION		
DESIGNATOR	**DESCRIPTION**	**PART NO.**
MDX-SST	Single Step Module	MK77958
MDX-SST	Operations Manual	MK79638
MDX-PROTO data sheet	MD Series Prototyping package	MK79605
AIM-80 data sheet	Z80 In-circuit Emulation module (2.5 MHz only)	MK78537

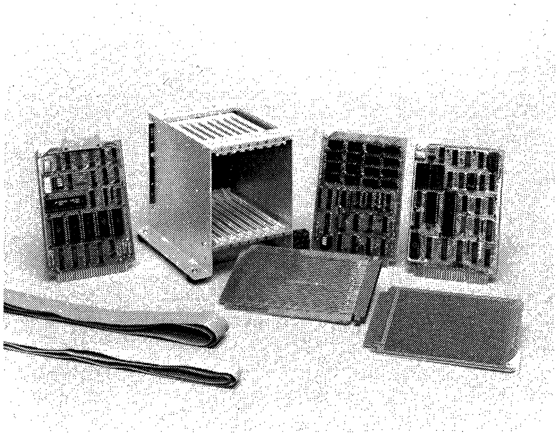
MOSTEK®

MD SERIES MICROCOMPUTER MODULES

Prototyping Package (MDX-PROTO)

FEATURES

- □ 8-slot card cage with mother board (MK77954)
- □ MDX-CPU1 module (MK77850)
- □ MDX-DRAM8 module (MK77750)
- □ MDX-DEBUG module (MK77950)
- □ MD-WW2 Wire wrap board (MK77952)
- □ MD-EXT Extender board (MK77593)
- □ Cables for RS232 device (MK77955) or TTY (MK77956)
- □ 4MHz option available (MDX-PROTO-4)
- □ STD BUS compatible



DESCRIPTION

The MD Series and the STD BUS were designed to satisfy the need for low-cost OEM microcomputer modules. The STD BUS uses a mother board interconnect system concept and is designed to handle any MD Series Card type in any slot. The modules for the STD BUS are a compact 4.5 x 6.5 inches which provide for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities provide high functionality per module.

The MD Series of OEM microcomputer boards and the STD BUS offer the most cost-effective system configuration available to the OEM system designer.

MDX-PROTO DESCRIPTION

HARDWARE DESCRIPTION

MDX-CPU1 DESCRIPTION

The MOSTEK MDX-CPU1 is the heart of an MD Series Z80 system. Based on the powerful Z80 microprocessor, the MDX-CPU1 can be used with great versatility in an OEM microcomputer system application. This is done simply by inserting custom ROM or EPROM memories into the sockets provided on the board and configuring them virtually anywhere within the Z80 memory map.

On-board memory is provided in the form of 4K of EPROM (2-2716's) and 256 bytes of scratchpad RAM as pictured in the block diagram. In addition, a MK3882 Counter Time Circuit is included on the MDX-CPU1 to provide counting and timing functions for the Z80. Either 2716 EPROM can be located at any 2K boundary within any given 16K block in the Z80 memory map via a jumper arrangement.

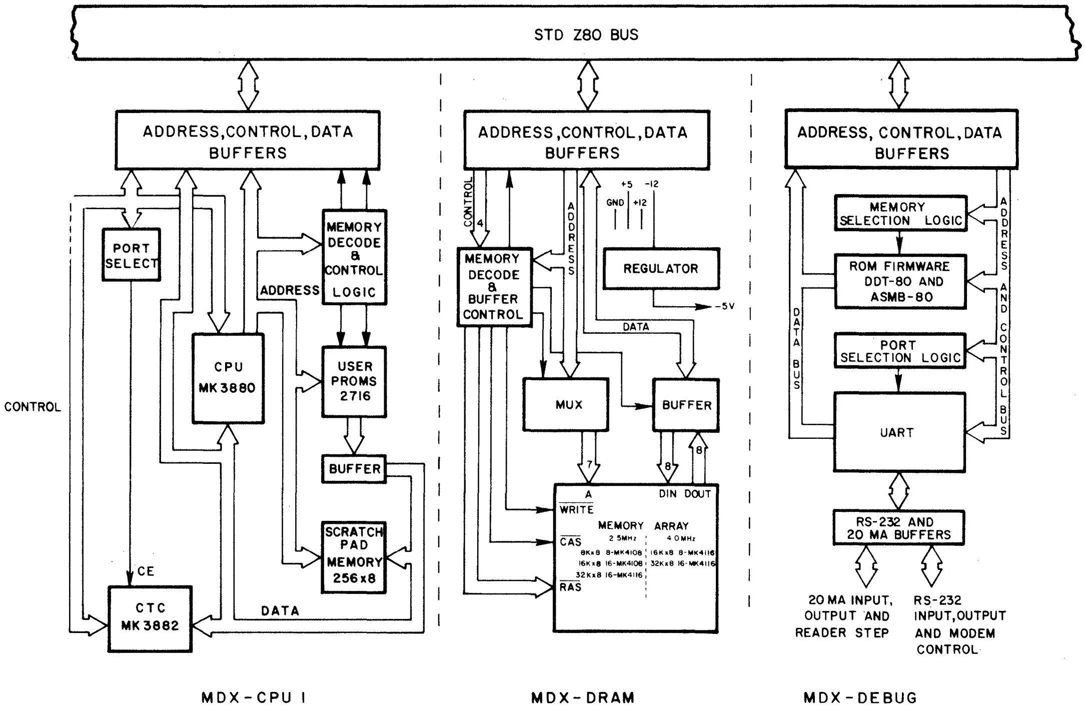
The MDX-CPU1 can be used in conjunction with the MDX-DEBUG and MDX-DRAM modules to utilize DDT-80 and ASMB-80 in system development. This is accomplished by strapping the scratchpad RAM to reside at location FF00 so that it will act as the Operating System RAM for DDT-80.

The MDX-CPU1 is also available in 4MHz version (MDX-CPU1-4). In this version, one wait cycle is automatically inserted each time on-board memory is accessed by a read or write cycle. This is necessary to make the access times of the 2716 PROMs and the 3539 scratchpad RAM compatible with MK3880-4 4MHz Z80-CPU.

MDX-DRAM DESCRIPTION

The MDX-DRAM is designed to be a RAM memory expansion board for the MOSTEK MD SERIES of Z80 based microcomputers. It is available in three memory capacities: 8K bytes (MDX-DRAM8), 16K bytes (MDX-

BLOCK DIAGRAM MDX-PROTO



DRAM 16), and 32K bytes (MDX-DRAM32). Additionally, the MDX-DRAM16 and the MDX-DRAM32 are available in a 4MHz version. Thus, the designer can choose from the various options to tailor his add-on dynamic RAM directly to his system requirements.

The MDX-DRAM8 is designed using MOSTEK's MK4108 8,192-bit dynamic RAM. The MDX-DRAM32 utilizes high-performance MK4116, 16K-bit dynamic RAMs which allow 4MHz versions of these boards to be offered. No wait-state insertion circuitry is required on any of the RAM cards.

Address selection is provided on all MDX-DRAM cards for positioning the 8K, 16K, or 32K of memory to start on any 4K boundary.

MDX-DEBUG DESCRIPTION

The MDX-DEBUG Module has sockets for 10K bytes of masked ROM that are populated with a Z80 firmware package (DDT-80/ASMB-80). This module has a STD BUS interface and is available in both 2.5MHz and 4.0MHz versions. Included on board is a fully buffered asynchronous I/O port capable of 110-19200 Baud Rates. Serial Data interfaces are available for 20mA current loop (with reader step control) and RS-232. The on-board Baud Rate Generator is selectable to all common Baud Rates from 110 to 19,200 Baud.

FIRMWARE DESCRIPTION

DEBUGGER DESCRIPTION

DDT-80 is the Operating System for the MDX-DEBUG Module. It resides in a 2K ROM (MK34000 series) resident on the MDX-DEBUG Module. It provides the necessary tools and techniques to operate the system, i.e., to efficiency and conveniently develop microcomputer software. DDT-80 is designed to support the user from initial design through production testing. It allows the user to display and update memory, registers, and ports, load and dump object files, set breakpoints, copy blocks of memory, and execute programs.

DDT-80 COMMAND SUMMARY

M s	- Display and/or update the contents of memory location s.
M s, f	- Tabulate the contents of memory locations s through f.
P s	- Display and/or update the contents of I/O port s.
D s,f	- Dump the contents of memory locations s through f in a format suitable to read by the L command.
L	- Load, into memory, data which is in the appropriate format.
E s	- Transfer control from DDT-80 to a user's program starting at location s.

H - Perform 16-bit hexadecimal addition and/or subtraction.

C s,f,d - Copy the contents of memory locations s through f to another location in memory starting at location d.

B s - Insert a breakpoint in the user's program (must be in RAM) at location s which transfers control back to DDT-80. This allows the user to intercept his program at a specific point (location s) and examine memory and CPU registers to determine if this program is working correctly.

R - Display the contents of user registers.

The s,f, and d represent start, finish, and destination operands required for each command.

MEMORY, PORT AND REGISTER COMMANDS (M,P,R)

The M, P, and R commands provide the means for displaying the contents of specified memory location, port addresses, or CPU registers. The M and P commands sequentially access memory locations or ports and display their contents. The user has the option of updating the content of the memory location or port. (Note some ports are output only and their contents cannot be read or displayed). The M command also gives the user access to the CPU registers through an area in RAM called the Register Map (discussed in the Execute, Breakpoint section below).

The M and R commands are used to tabulate blocks of memory locations (M) or the CPU registers (R). The M command will accept two operands, the starting and ending addresses of the memory block to tabulated. The R command will accept either no operand or one. If no operand is specified, the CPU registers will be displayed without a heading. If an operand is specified then a heading which labels the register contents will be displayed as well.

EXECUTE AND BREAKPOINT (E,B).

The E command is used to execute all programs, including aids such as the Assembler. The B command is used to set a breakpoint to exit from a program at some predetermined location for debugging purposes. At the instant of a breakpoint exit, the contents of all CPU registers are saved in a designated area of MDX-DEBUG RAM called the Register Map. In the Register Map, the register contents may be examined or modified using the M command and a predefined mnemonic (or absolute address) of the storage location for that register (Example: PC, :A,...,:SP). The Register Map is also used to initialized the CPU registers whenever execution is initiated or resumed. Thus the E and B commands can be used together to initialize, execute, and examine the results of individual program segments.

The B command gives the user the option of having all CPU registers displayed when the breakpoint is encountered. This is done by entering a second operand to the B command. Otherwise DDT-80 defaults to displaying the PC and AF registers. When all CPU registers are displayed, the format is the same as for the R command previously discussed.

LOAD, DUMP, AND COPY, (L,D,C)

The L and D commands load and dump object files through the object I/O channel in standard Intel Hex format. Checksums are used for error detection, and the addresses of questionable blocks are typed automatically while loading.

The C command will copy the contents of the memory block specified to another block of memory. There are no restrictions on the direction of the copy or on whether the blocks overlap.

HEXADECIMAL ARITHMETIC (H)

The H command is a dummy command used to allow hexadecimal addition and subtraction for expression evaluation without performing any other operation.

DDT-80 I/O CAPABILITIES

DDT-80 specifies I/O channels, designated 'Console', 'Object', and 'Source', to which any suitable devices may be assigned. The Channel Assignment Table is located in MDX-RAM where it may be examined or modified using the M command. The table addresses correspond to the I/O channels and the table contents correspond to the addresses of the peripheral driver routines. A channel which has a device assignment may have that device assignment changed using the M command. This is accomplished by merely modifying the table contents of that channel's table address to correspond to the new peripheral driver routine. A set of peripheral driver routines is supplied and listed below. This scheme also allows the user to write a driver routine for his own peripheral, load it into memory, and easily configure that peripheral into the system.

DDT-80 I/O PERIPHERAL DRIVERS

1. A serial input driver (usually a keyboard).
2. A serial output driver (usually a CRT or teletype typehead).
3. A serial input driver which sends out a reader step signal (usually a teletype reader).
4. A serial output driver which forces a delay after a carriage return (usually a Silent 700 typehead).
5. A parallel input driver (usually for high-speed paper tape input).
6. A parallel output driver (usually for high-speed paper tape output).
7. A parallel output driver (usually for a line printer).

TEXT EDITOR DESCRIPTION

The Text Editor permits random access editing of ASCII character strings. It can be used as a line or character-oriented editor. Individual characters may be located by position or context. The Editor works on blocks of characters which are typically read into memory from magnetic tape or paper tape. Each edited block can be output to magnetic tape or paper tape after editing is completed. While the primary application for the Text

Editor is in editing assembly language source statements, it may be applied to any ASCII text delimited by "carriage returns".

The Editor has a macro command processing option. Up to two sets of commands may be stored and processed at any time during the editing process. All I/O is done via the DDT-80 channels. The Editor can be used with the MOSTEK ASMB-80 Assembler and Loader to edit, assemble, and load programs in memory without the need for external media for intermediate storage.

The following commands are recognized by the Text Editor:

An	- Advance record pointer n seconds
Bn	- Backup record pointer n seconds
Cn dS1dS2D	- Change string S1 to string S2 for n occurences
Dn	- Delete n records
E	- Exchange current record with records to be inserted
I	- Insert records
Ln	- Go to line number n
Mn	- Enter command buffers (pseudo-macro)
N	- Print top, bottom and current line number
Pn	- Punch n records from buffer
R	- Read source records into buffer
Sn dS1d	- Search for nth occurrence of signal S1

ASSEMBLER DESCRIPTION

The Assembler reads Z80 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The assembly listing shows address, machine code, statement number, and source statement. The object code is in industry-standard hexadecimal format modified for relocatable, linkable assemblies.

The Assembler supports conditional assemblies, global symbols, relocatable programs and a printed symbol table. It can assemble any length program, limited only by a symbol table size which is user selectable. Expressions involving addition and subtraction are allowed. A global symbol is catagorized as "internal" if it appears as a label in the program; otherwise it is an "external" symbol. The printed symbol table shows which symbols are internal and which are external. The assembler allows the user to select relocatable or non-relocatable assembly via the "PSECT" pseudo-op. Relocation records are placed in the object output for relocatable assemblies. (The MOSTEK object format is defined below.) The Assembler can be run as a single-pass assembler or as a learning tool. (In this mode, global symbols and forward references are not allowed.) The following pseudo-ops are recognized by the Assembler:

EQU	- equate label
DEFL	- define label
DEFM	- define message
DEFB	- define byte
DEFW	- define word
DEFS	- define storage
END	- end statement
NAME	- program name definition
PSECT	- global symbol definition Supports the following assembler psuedo-ops
EJECT	- eject a page of listing
TITLE	- place heading at top of each page
LIST	- turn listing on
NLIST	- turn listing off

RELOCATING LINKING LOADER DESCRIPTION

The MOSTEK Relocating Linking Loader provides state-of-the-art capability for loading programs into memory by allowing loading and linking of any number of relocatable and non-relocatable object modules. Non-relocatable modules are always loaded at their starting address as defined by the ORG pseudo-op during assembly. Relocatable object modules can be positioned anywhere in memory at an offset address.

The Loader automatically links and relocates global symbols which are used to provide communication or linkage betweeen program modules. As object programs are loaded a table containing global symbol references and definitions is built up. At the end of each module, the loader resolves all references to global symbols which are defined by the current or a previously loaded module. It also prints on the console device the number of defined global symbols that have been referenced. The symbol table can be printed in order to list all global symbols and their load address. The number of object modules which can be loaded by the Loader is limited only by the amount of MDX-RAM available for the modules and the symbol table. Space for the symbol table is allocated dynamically downward in memory from either the top of memory or from a specified address entered as an operand of the load command.

All I/O is done via the DDT-80 channels. Assemblies can be done from source statements stored in memory (by the Editor). The object output can be directed to a memory buffer rather than to an external device. Thus, assembly and loading can be done without external storage media.

The Loader prints the beginning and ending address of each module as it is loaded. The transfer address as defined by the END pseudo-op is printed for the first module loaded. The Loader execute command (E) can be used to automatically start execution at the transfer address.

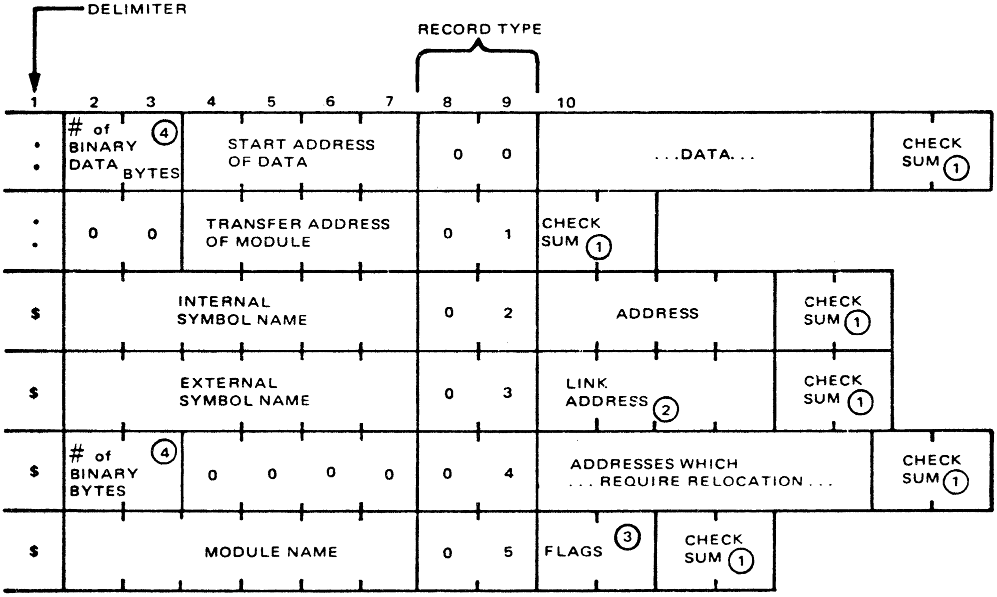
The Loader Commands are the following:

L offset	- load object module at address "offset" plus program origin address
E	- execute loaded program at transfer address of first module
T	- print global symbol table

MOSTEK OBJECT OUTPUT DEFINITION

Each record of an object module begins with a delimiter (colon or dollar sign) and ends with carriage return and line feed. A colon (:) is used for data records and end-of-file record. A dollar sign ($) is used for records containing relocation information and linking information. All information is in ASCII. Each record is identified by "type". The type is determined by the 8th and 9th bytes of the record which can take the following values:

00 - data
01 - end-of-file
02 - internal symbol
03 - external symbol
04 - relocation information
05 - module definition



NOTES:

1. Check Sum is negative of the binary sum of all bytes except delimiter and carriage return/line feed.
2. Link Address points to last address in the data which uses the external symbol. This starts a backward link list through the data records for the external symbol. The list terminates at OFFFFFH.
3. The flags are one binary byte. Bit 0 is defined as:
   0 - absolute module
   1 - relocatable module
4. Maximum of 64 ASCII bytes.

ELECTRICAL SPECIFICATIONS

MDX-CPU1

WORD SIZE

Instruction: 8, 16, 24, or 32 bits
Data: 8 bits

CYCLE TIME

Clock period or T state = 0.4 microsecond @ 2.5MHz
0.25 microsecond @ 4.00 MHz
Instructions require from 4 to 23 T states

MEMORY ADDRESSING

On-Board EPROM: jumper selectable for any 2K boundary within a 16K block of Z80 memory map.
On-Board RAM: FFOO-FFFF

MEMORY CAPACITY

On-Board EPROM - 4K bytes (sockets only)
On-Board RAM-256 bytes
Off-board Expansion - Up to 65,536 bytes with user-specified combinations of RAM, ROM, PROM.

MEMORY SPEED REQUIRED

MEMORY	ACCESS TIME	CYCLE TIME
2716*	450ns	450ns

*Single 5 volt type required

I/O ADDRESSING

On-Board Programmable Timer

PORT ADDRESS (HEX)	MK3882 CHANNEL
7C	0
7D	1
7E	2
7F	3

I/O CAPACITY

Up to 252 port address can be decoded off board. Four port addresses are on board. 252 + 4V = 256 total I/O ports.

INTERRUPTS

Multi-level with three vectoring modes (Mode 0,1,2). Interrupt requests may originate from user-specified I/O or from the on-board MK3882 CTC.

PARALLEL BUS INTERFACE - STD BUS COMPATIBLE

Inputs	One 74LS load max
Bus Outputs	I_{OH} = -3mA min at 2.4 volts
	I_{OL} = 24mA min at 0.5 volts

SYSTEM CLOCK

	MIN	MAX
MDX-CPU1	500 KHz	2.500MHz
MDX-CPU-4	500 KHz	4.000MHz

POWER SUPPLY REQUIREMENTS

5V ± 5% at 1.1A maximum

OPERATING TEMPERATURE

0°C to 50°C

MDX-DRAM

WORD SIZE

8 bits

MEMORY SIZE

MDX-DRAM8	- 8,192 bytes
MDX-DRAM16	- 16,384 bytes
MDX-DRAM32	- 32,768 bytes

ACCESS TIME

	SYSTEM CLOCK	MEMORY ACCESS TIMES	MEMORY CYCLE TIMES
MDX-DRAM	2.5MHz	350ns max.	465ns min.
MDX-DRAM-4	4.0MHz	200ns max.	325ns mih.

ADDRESS SELECTION

Selection of 8K, 16K, or 32K contiguous memory blocks to reside at any 4K boundary.

SYSTEM CLOCK

	MIN	MAX
MDX-DRAM	1.25MHz	2.5MHz
MDX-DRAM-4	1.25MHz	4.0MHz

PARALLEL BUS INTERFACE-STD BUS COMPATIBLE

Inputs	One 74LS load max
Bus Outputs	IOH =-15mA min. at 2.4 volts
	IOL = 24mA min at 0.5 volts

POWER SUPPLY REQUIREMENTS

+5V ± 5% at 0.6A max.
+12V ± 5% at 0.25A max.
-12V ± 5% at 0.03A max.

OPERATING TEMPERATURE

0°C to 50°C

MDX-DEBUG

I/O TRANSFER RATE

X 16 Baud Rate Clock	Baud Rate (Hz)
1,760	110
4,800	300
9,600	600
19,200	1,200
38,400	2,400
76,800	4,800
153,600	9,600
307,200	19,200

SERIAL COMMUNICATIONS CHARACTERISTICS

Asynchronous
- Full duplex operation
- Start bit verification
- Data word size variable from 5 to 8 bits
- One or two stops bits
- Odd, even, or no parity
- One word buffering on both transmit and on receive.

SERIAL COMMUNICATIONS INTERFACE

SIGNAL	BUFFERED FOR: 20mA Current Loop	RS-232
Transmitted data	Output	Output
Received data	Input	Input
Data Terminal Ready (DTR)		Input
Request to Send (RTS)		Input
Carrier Detect (CDET)		Output
Clear to Send (CTS)		Output
Data Set Ready (DSR)		Output
Reader Step relay (RS)	Output (20mA)	

PARALLEL BUS INTERFACE-STD BUS COMPATIBLE

Inputs	One 74LS load max
Bus Outputs	I_{OH} = -3mA min. at 2.4 volts
	I_{OL} = 24mA min. at 0.5 volts

I/O ADDRESSING

On-Board Serial I/O Port
Control Port DDH
Data Port DCH
Module and Reader Step Control Port DEH

SYSTEM CLOCK

MDX-DEBUG	1.25MHz	2.5MHz
MDX-DEBUG-4	1.25MHz	4.0MHz

POWER SUPPLY REQUIREMENT

+12 Volts ± 5% at 50 mA max.
-12 Volts ± 5% at 35 mA max.
+5 Volts ± 5% at 1.2 mA max.

OPERATING TEMPERATURE

0° to 50°C

MECHANICAL SPECIFICATIONS

CARD DIMENSIONS

4.5 in. (11.43cm) high by 6.50 in (16.51 cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (.016 cm) printed circuit board thickness

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD BUS	56 pin dual 0.125 in. centers	Printed Circuit Viking 3VH28/1CE5 Wire Wrap Viking 3VH28/1CND5 Solder Lug Viking 3VH28/1CN5

MD-CC8 STD BUSSED 1/4 rack (MK77954) bussed motherboard with eight connectors on 0.5 in. centers.

STD BUS Organization

RS232 Cable MD-RS232	26 pin socket connector ANSLEY #609-2061M 5 feet of 26 wire flatcable ANSLEY #171-26 25-pin standard EIA ANSLEY #609-25S
TTY Cable MD-TTY	26 pin socket connector ANSLEY #609-2061M 5 feet of 26 wire flatcable ANSLEY #171-26 TTY connector Molex 15 Pin Molex #03-09-2151

STD BUS

	COMPONENT SIDE				CIRCUIT SIDE			
	PIN	MNEMONIC	SIGNAL FLOW	DESCRIPTION	PIN	MNEMONIC	SIGNAL FLOW	DESCRIPTION
LOGIC POWER BUS	1	+5V	IN	+5 Volts DC (Bussed)	2	+5V	In	+5VDC (Bussed)
	3	GND	IN	Digital Ground (Bussed)	4	GND	In	Digital Ground (Bussed)
	5	-5V	IN	-5 Volts DC	6	-5V	In	-5 Volts DC
DATA BUS	7	D3	In/Out	Low Order Data Bus	8	D7	In/Out	High Order Data Bus
	9	D2	In/Out	Low Order Data Bus	10	D6	In/Out	High Order Data Bus
	11	D1	In/Out	Low Order Data Bus	12	D5	In/Out	High Order Data Bus
	13	D0	In/Out	Low Order Data Bus	14	D4	In/Out	High Order Data Bus
ADDRESS BUS	15	A7	Out	Low Order Address Bus	16	A15	Out	High Order Address Bus
	17	A6	Out	Low Order Address Bus	18	A14	Out	High Order Address Bus
	19	A5	Out	Low Order Address Bus	20	A13	Out	High Order Address Bus
	21	A4	Out	Low Order Address Bus	22	A12	Out	High Order Address Bus
	23	A3	Out	Low Order Address Bus	24	A11	Out	High Order Address Bus
	25	A2	Out	Low Order Address Bus	26	A10	Out	High Order Address Bus
	27	A1	Out	Low Order Address Bus	28	A9	Out	High Order Address Bus
	29	A0	Out	Low Order Address Bus	30	A8	Out	High Order Address Bus
CONTROL BUS	31	WR	Out	Write to Memory or I/O	32	RD	Out	Read to Memory or I/O
	33	IORQ	Out	I/O Address Select	34	MEMRQ	Out	Memory Address Select
	35	IOEXP	Out	I/O Expansion	36	MEMEX	In/Out	Memory Expansion
	37	REFRESH	Out	Refresh Timing	38	MC SYNCU	**	**
	39	STATUS 1	Out	**	40	STATUS 0	Out	CPU Status
	41	BUSAK	Out	Bus Acknowledge	42	BUSRQ	In	Bus Request
	43	INTAK	Out	Interrupt Acknowledge	44	INTRQ	In	Interrupt Request
	45	WAITRQ	In	Wait Request	46	NMIRQ	In	Non-Maskable interrupt
	47	SYSRESET	Out	System Reset	48	PBRESET	In	Push Button Reset
	49	CLOCK	Out	Clock from Processor	50	CNTRL	In	AUX Timing
	51	PCO	Out	Priority Chain Out	52	PCI	In	Priority Chain In
POWER BUS	53	AUXGND	In	AUX Ground (Bussed)	54	AUXGND	In	AUX Ground (Bussed)
	55	AUX+V	In	+12 Volts DC)	56	AUX-V	In	-12 Volts DC

**Refer to a STD-Z80 BUS Description

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MDX-PROTO	Prototyping package with Operations Manuals. 2.5 MHz version.	MK77951 Note: 2.5 MHz version includes 8K dynamic RAM board MDX-DRAM8 only.
MDX-PROTO-4	Prototyping package with Operations Manuals. 4.0 MHz version.	MK77951-4 NOTE: 4.0 MHz version includes 16K dynamic RAM board MDX-DRAM 16-4.
AID-80F data sheet	Disk-based development system for system for MD series.	MK78568
AIM-80 data sheet	Z80 In-Circuit Emulation module (2.5 MHz only).	MK78537

MOSTEK®

MD SERIES MICROCOMPUTER MODULES

Universal Memory Card (MDX-UMC)

FEATURES

□ Can be strapped to accept the following industry-standard memory devices:

EPROM	STATIC RAM	ROM
2758 (1K x 8)	MK4118 (1K x 8)	
2716 (2K x 8)	MK4802 (2K x 8)	MK34000 (2K x 8)
2732 (4K x 8)		

□ Memories can be mixed to form a combination memory board

□ Wait state generator for 4MHz operation

□ STD-Z80 BUS compatible

□ +5 Volt only

DESCRIPTION

The MD Series and the STD-Z80 BUS were designed to satisfy the need for low cost OEM microcomputer modules. The STD-Z80 BUS uses a motherboard interconnect system concept and is designed to handle any MD Series card type in any slot. The modules for the STD-Z80 BUS are a compact 4.5 x 6.5 inches which provides for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities provide high functionality per module.

The MD Series of OEM microcomputer boards and the STD-Z80 BUS offer the most cost effective system configuration available to the OEM system designer.

MDX-UMC DESCRIPTION

The MDX-UMC is one of MOSTEK's complete line of STD-Z80 BUS compatible microcomputer modules.

Designed as a universal memory card for the STD-Z80 BUS, the MDX-UMC provides the user with the capability of configuring the board to meet the system requirement of ROM/EPROM and/or RAM. By the use of strapping options, the user is able to configure pairs of sockets for ROM/EPROM/RAM to form a combination memory board.

Other MDX-UMC features include 4K boundary addressing and an optional wait-state generator to accomodate slower memories for 4MHz operations.

MECHANICAL SPECIFICATIONS

CARD DIMENSION

4.5 in. (11.43cm) high by 6.50 in. (16.51cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD-Z80 BUS	56 pin dual read out	Printed Circuit Viking 3VH28/ 1CE5
	0.125 in. centers	Wire Wrap Viking 3VH28/ 1CND5
		Solder Lug Viking 3VH28/ 1CN5

ELECTRICAL SPECIFICATIONS

WORD SIZE

8 bits

MEMORY ADDRESSING

4K boundaries

MEMORY CAPACITY

8 sockets
Sockets are strapped in pairs to accomodate the following memories:

EPROM	STATIC RAM	ROM
2758	MK4118	
2716	MK4802	MK34000
2732		

PARALLEL BUS INTERFACE - STD-Z80 BUS COMPATIBLE

Inputs:	One 74LS load max.
BUS Outputs:	I_{OH} = –15mA min at 2.4 Volts
	I_{OL} = 24mA min at 0.5 Volts

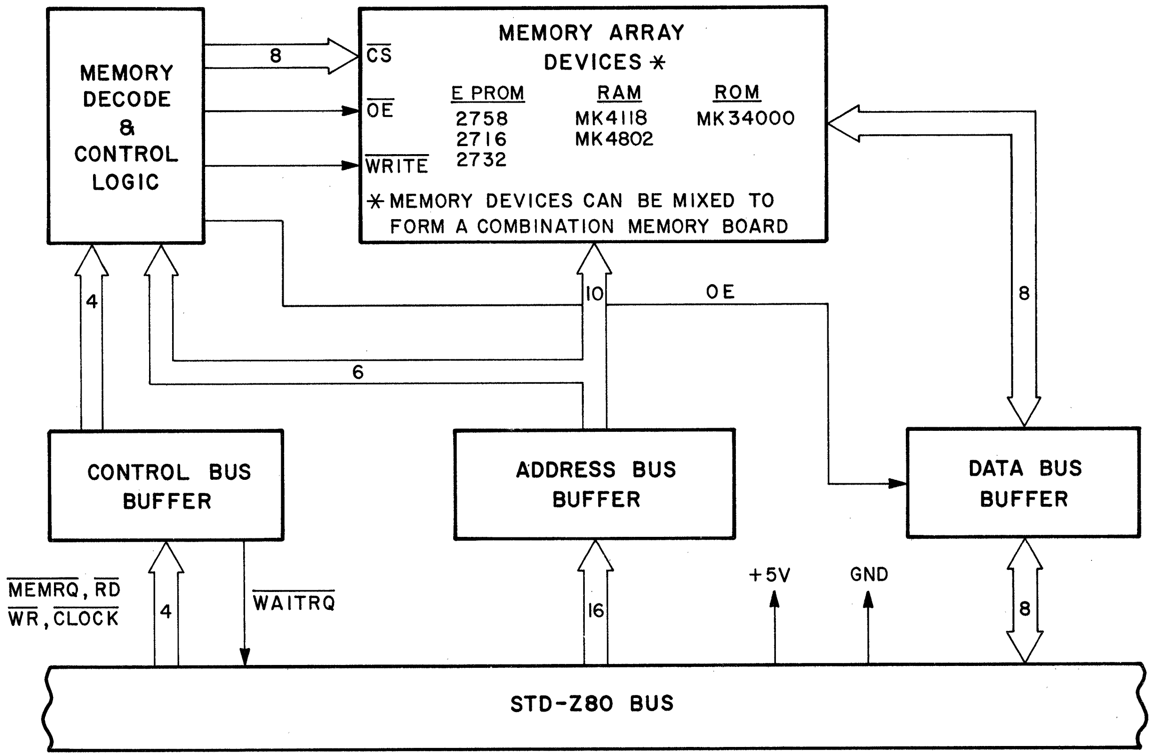
POWER SUPPLY REQUIREMENTS*

+5V ± 5% at 0.450 A max
*Does not include power for memories

OPERATING TEMPERATURE

0°C to 50°C

MDX-UMC BLOCK DIAGRAM



MDX-UMC BLOCK DIAGRAM ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MDX-UMC	Module with operation manual less mating connectors	MK77759
MDX-PROTO	MD Series prototyping package	MK77951
AID-80F	MD Series development system	MK78125
AIM-80	Z80 In-Circuit Emulation module (2.5MHz only)	MK78132

MOSTEK®

MD SERIES MICROCOMPUTER MODULES

EPROM Module (MDX-EPROM)

FEATURES

□ Accepts the following industry standard EPROMS:
2758 (1K x 8)
2716 (2K x 8)
2732 (4K x 8)

□ Eight EPROM sockets for maximum storage of:
8K x 8 using 2758's
16K x 8 using 2716's
32K x 8 using 2732's

□ Wait state generator for 4MHz operation

□ STD-Z80 BUS compatible

□ +5 Volt only

Description

The MD series and the STD-Z80 BUS were designed to satisfy the need for low cost OEM microcomputer modules. The STD-Z80 BUS uses a motherboard interconnect system concept and is designed to handle any MD Series card type in any slot. The modules for the STD-Z80 BUS are a compact 4.5 x 6.5 inches which provides for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities provide high functionality per module.

The MD Series of OEM microcomputer boards and the STD-Z80 BUS offer the most cost effective system configuration available to the OEM system designer.

MDX-EPROM DESCRIPTION

The MDX-EPROM is designed to be an EPROM memory expansion board for the MOSTEK MD SERIES™ of Z80-based microcomputers. The MDX-EPROM accepts the following EPROMS; 2758 (1K x 8), 2716 (2K x 8) and 2732 (4K x 8) which gives a maximum storage capacity of 8K, 16K, or 32K bytes respectively.

Starting address selection is provided for positioning the MDX-EPROM on any 4K boundary. A wait-state generator is also provided for optional 4MHz operation.

ELECTRICAL SPECIFICATIONS

WORD SIZE

8 bits

MEMORY CAPACITY

8K x 8 using eight 2758's
16K x 8 using eight 2716's*
32K x 8 using eight 2732's
*EPROMS included

REQUIRED ACCESS TIME

MEMORY TIME	MIN ACCESS TIME	CYCLE TIME
2758, 2716, 2732	450ns*	450ns

*One wait state must be added for 4MHz operation.

ADDRESS SELECTION

4K boundaries

BUS INTERFACE

STD-Z80 BUS compatible

Inputs:	One 74LS load max.
Bus Outputs:	I_{OH} = –15mA min at 2.4 Volts
	I_{OH} = 24mA min at 0.5 Volts

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD-Z80 BUS	56 Pin dual 0.125 in centers	Printed Circuit Viking 3VH28/1CE5 Wire Wrap Viking 3VH28/1CND5 Solder Lug Viking 3VH28/1CN5

MD Series Expand

MECHANICAL SPECIFICATIONS

CARD DIMENSION

4.5 in. (11.43cm) high by 6.50 in. (16.51cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

POWER SUPPLY REQUIREMENTS*

+5 Volts ± 5% at 0.45A

*Does not include EPROMs. Add 100 mA for each EPROM.

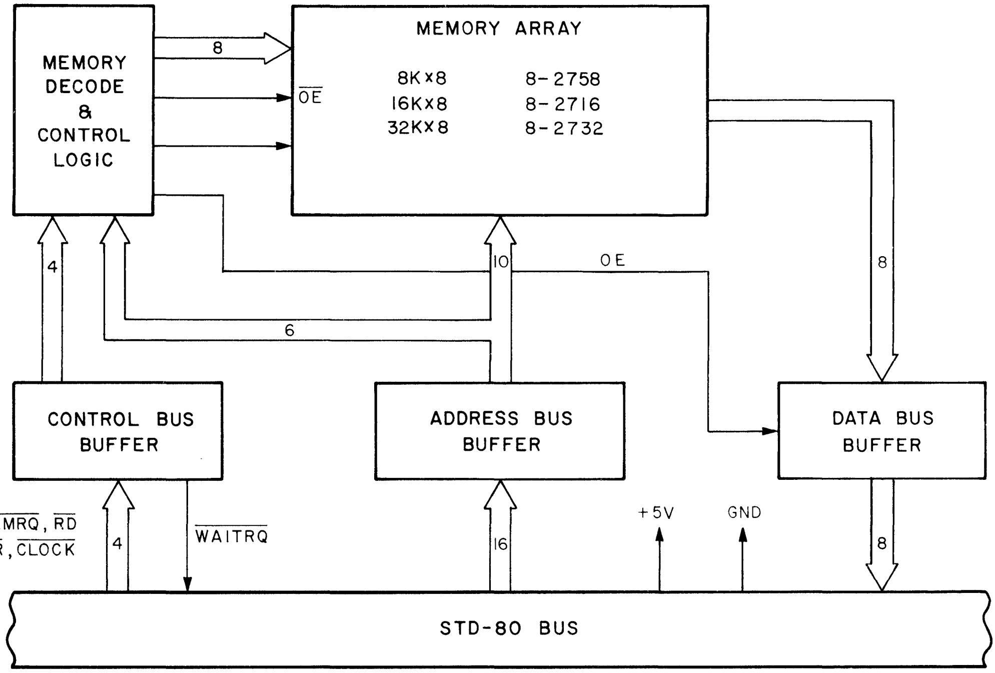
OPERATING TEMPERATURE

0°C to 50°C

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MDX-EPROM	Module with Operation Manual less mating connectors (does not include EPROMS).	MK77758
	MDX-EPROM Operations Manual only	MK79671✓
MDX-PROTO	MD Series prototyping package	MK77951
AID-80F	Disk-based development system for MD series	MK78125
AIM-80	Z80 In-Circuit-Emulation module (2.5 MHz only)	MK78132

MDX-EPROM BLOCK DIAGRAM



MD Series
Expand

MOSTEK®

MD SERIES MICROCOMPUTER MODULES

Static RAM Module (MDX-SRAM)

FEATURES

- □ Three memory sizes
  - 4K x 8 (MDX-SRAM4)
  - 8K x 8 (MDX-SRAM8)
  - 16K x 8 (MDX-SRAM16)
- □ Selectable starting adddress on 4K boundaries
- □ 2.5 MHz and 4.0 MHz compatible
- □ STD-Z80 BUS compatible
- □ +5 Volt only

DESCRIPTION

The MD Series™ and the STD-Z80 BUS were designed to satisfy the need for low-cost OEM microcomputer modules. The STD-Z80 uses a motherboard interconnect system concept and is designed to handle any MD Series card type in any slot. The modules for the STD-Z80 BUS are a compact 4.5 x 6.5 inches which provides for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities provide high functionality per module.

The MD Series of OEM microcomputer boards and the STD-Z80 BUS offer the most cost-effective system configuration available to the OEM system designer.

MDX-SRAM DESCRIPTION

The MDX-SRAM is designed to be a static RAM Memory expansion board for the MOSTEK MD SERIES of Z80 based microcomputers. It is available in three memory capacities; 4K bytes (MDX-SRAM4), 8K bytes (MDX-SRAM8), and 16K bytes (MDX-SRAM16). Additionally, all MDX-SRAM boards are 2.5MHz and 4.0MHz compatible. Thus, the designer can choose from three options available and tailor the add-on static RAM directly to the system requirements.

The MDX-SRAM is designed using the state of the art MK4118 (1Kx8) static RAM and MK4802 (2Kx8) static RAM memory devices. Because of the high speed of the MK4118 and MK4802, no wait states are necessary for operating the MDX-SRAM at 2.5MHz or 4.0MHz. Address selection is provided on all MDX-SRAM cards for positioning the 4K, 8K, or 16K of memory to start on any 4K boundary.

ADDRESS SELECTION

Selection of 4K, 8K, or 16K contiguous memory blocks to begin on any 4K boundary.

BUS INTERFACE

STD-Z80 BUS compatible
Inputs: One 74LS load max
Bus Outputs: I_{OH} = -15mA min at 2.4 Volts
I_{OL} = 24mA min at 0.5 Volts

POWER SUPPLY REQUIREMENTS

BOARDS	+5V ± 5%
MDX-SRAM4	0.8 A max
MDX-SRAM8	1.2 A max
MDX-SRAM16	1.2 A max

ELECTRICAL SPECIFICATIONS

WORD SIZE
8 bits

MEMORY SIZE

MDX-SRAM4	- 4,096 bytes
MDX-SRAM8	- 8,192 bytes
MDX-SRAM16	- 16,384 bytes

TIMING

	MEMORY ACCESS	MEMORY CYCLE
MDX-SRAM	250ns max.	250ns min.

OPERATING TEMPERATURE

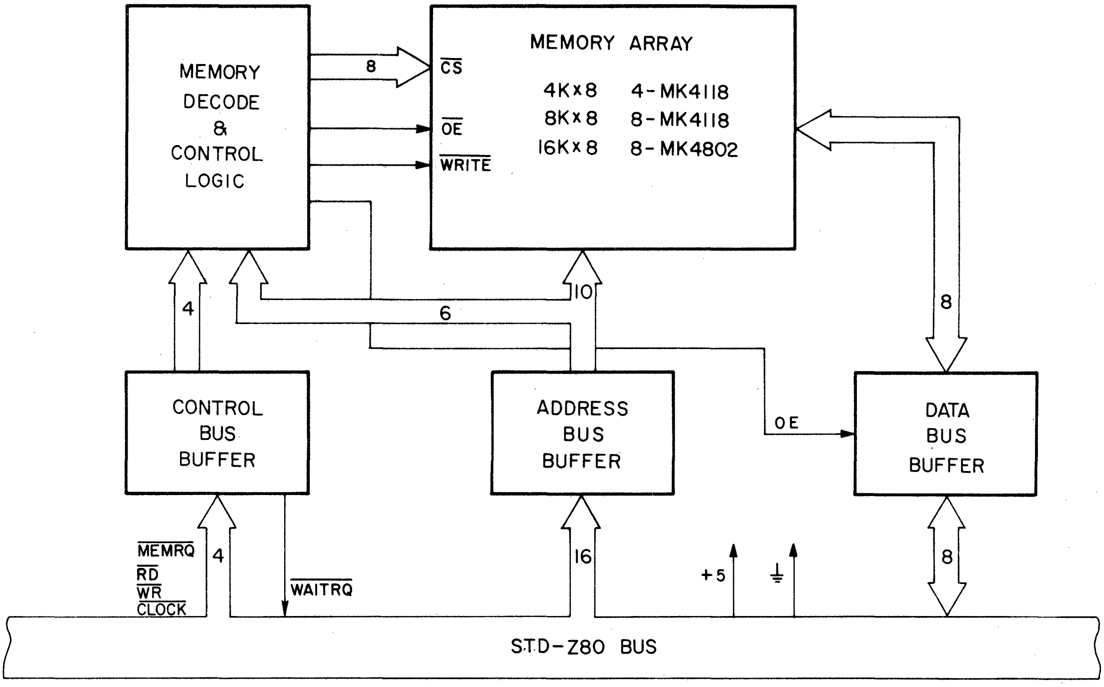
0°C to 50°C

MECHANICAL SPECIFICATIONS

CARD DIMENSION

4.5 in. (11.43 cm) high by 6.50 in. (16.51 cm) long
0.48 in. (1.22 cm) maximum profile thickness
0.062 in. (0.16 cm) printed circuit board thickness

MDX-SRAM BLOCK DIAGRAM



CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
		Printed Circuit
STD-Z80 BUS	56 pin dual read out	Viking 3VH28/ 1CE5 Wire Wrap
	0.125 in centers	Viking 3VH28/ 1CND5 Solder Lug Viking 3VH28/ 1CN5

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MDX-SRAM4	4K Bytes (4118's) module with operation manual less mating connectors	MK77755
MDX-SRAM8	8K Bytes (4118's) module with operation manual less mating connectors	MK77756
MDX-SRAM16	16K Bytes (4802's) module with operation manual less mating connectors	MK77757
MDX-PROTO Data Sheet	MD Series prototyping package	MK79605
AID-80F Data Sheet	Disk based development system for MD Series	MK78568
AIM-80 Data Sheet	Z80 In-circuit emulation module (2.5 MHz only)	MK78537

MOSTEK®

MD SERIES ACCESSORIES

MD-ACC

The following items are available as accessories to support design, development, and production of products designed around the MOSTEK MD Series Z80 microcomputer modules:

- WW1 wire wrap card with bussed power and ground
- WW2 wire wrap card without bussed power and ground
- MD-CC8 8-slot card cage
- MD-CC14 14-slot card cage
- MD-CC28 28-slot card cage
- MD-EXT Extender card.

Description

The STD BUS concept is a joint design between Mostek and Pro-Log to satisfy the need for cost-effective OEM Microcomputer Systems. The definition of the STD BUS and the MD Series of OEM microcomputer modules are a result of years of microcomputer component and module manufacturing experience. The STD BUS uses a motherboard interconnect system concept and is designed to handle any MD Series card in any card slot. Modules for the STD BUS range from CPU, RAM and EPROM Modules to Input, Output, A/D, and TRIAC control modules. A ROM-based DEBUG module provides users of the STD BUS with Edit, Assembly, and Debug capability using only an ASCII terminal.

Printed circuit modules for the STD BUS are a compact 4.5 x 6.5 inches providing for system partitioning by function (RAM, PROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities provide high functionality per module.

MECHANICAL SPECIFICATIONS

CARD DIMENSIONS

4.5 in (11.43cm) high by 6.50 in. (16.51cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

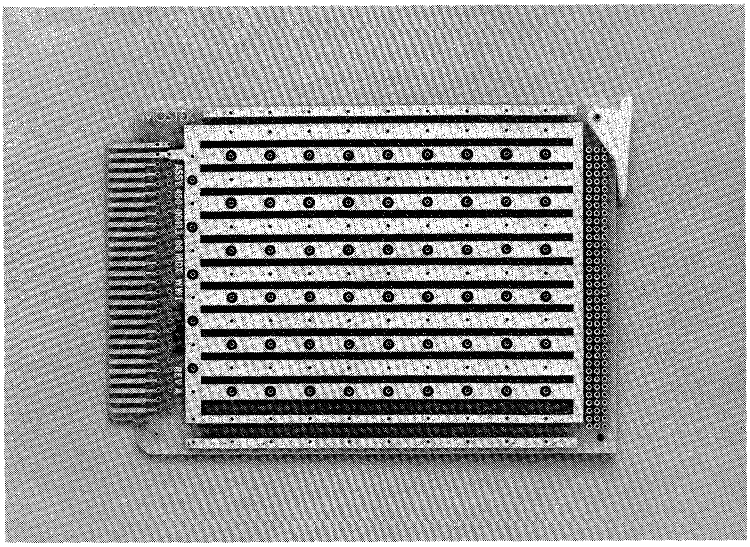
CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
STD-Z80 BUS	56 pin dual read out 0.125 in. centers	Printed Circuit Viking 3VH28/ICE5 Wire Wrap Viking 3VH28/1CND5 Solder Lug Viking 3VH28/1CN5

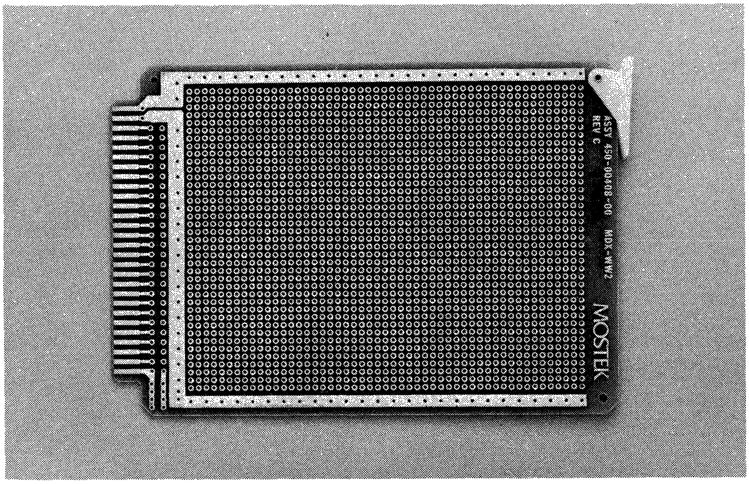
ORDER INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MD-WW1	MD Series wire wrap card with bussed power and ground	MK77959
MD-WW2	MD Series wire wrap card without bussed power and ground	MK77952
MD-EXT	MD Series extender card	MK77953
MD-CC8	MD Series 8-slot card cage with STD BUS motherboard	MK77954
MD-CC14	MD Series 14-slot card cage with STD BUS motherboard.	MK77960
MD-CC28	MD Series 28-slot card cage with STD BUS motherboard	MK77961

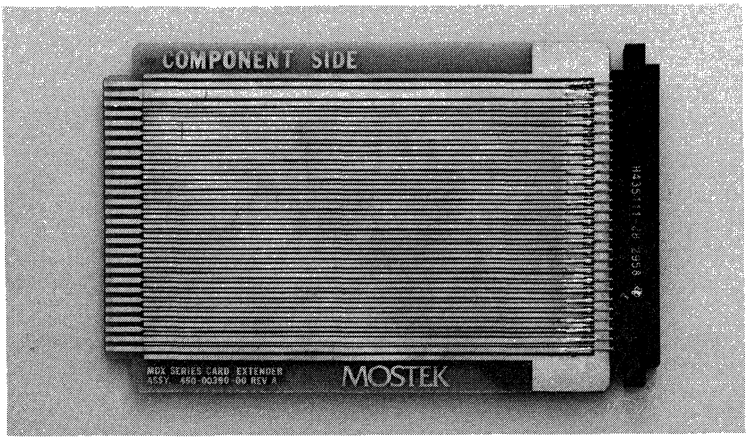
WW1 PHOTO MK77959



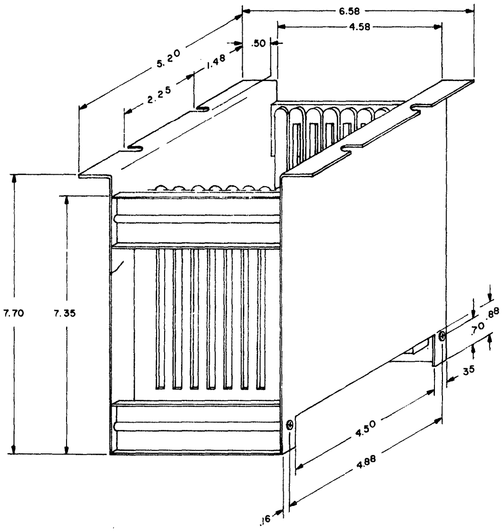
WW2 PHOTO MK77952



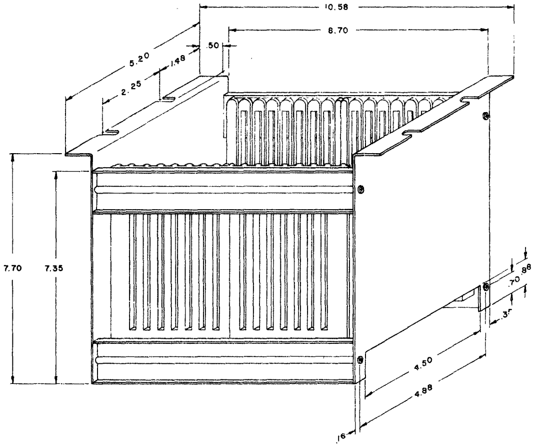
MD - EXT PHOTO MK77953



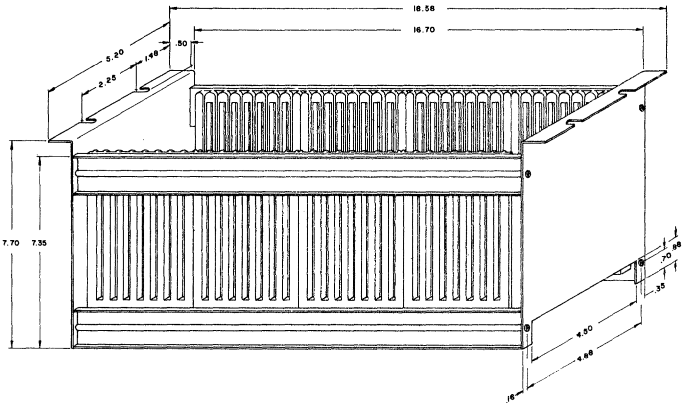
MD-CC 8 Drawing with Dimensions



MD-CC 14 Drawing with Dimensions

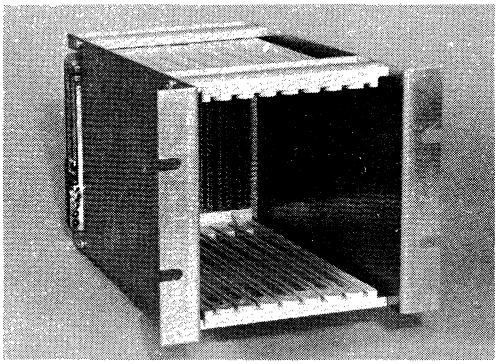


MD-CC 28 Drawing with Dimensions

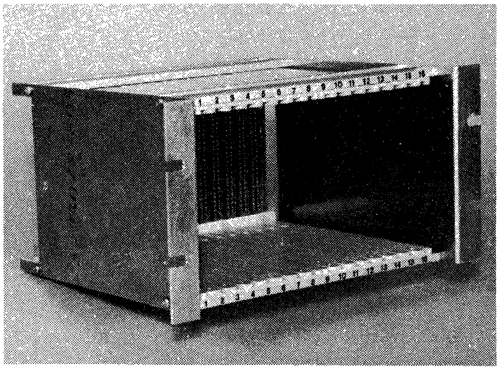


MD Series Expand

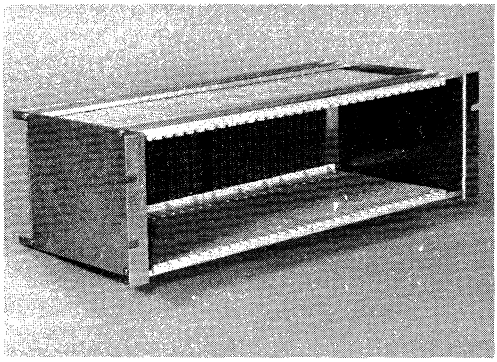
MD-CC8 8-Slot Card Cage



MD-CC14 14-Slot Card Cage



MD-CC28 28-Slot Card Cage



MOSTEK®

MD SERIES MICROCOMPUTER MODULES

Analog to Digital Conversion Module (MDX-A/D)

FEATURES

□ 8-Bit A/D converter with 16 single-ended analog inputs

□ 3 full-scale input ranges
- 0 to +1 Volts
- 0 to +2 Volts
- 0 to +5 Volts

□ Total unadjusted error < ± ½ LSB

□ Linearity error < ± ½ LSB

□ No missing codes

□ Guaranteed monotonicity

□ No zero adjust required

□ No full scale adjust required

□ Provisions for additional channel expansion

□ Optional sample and hold

□ Address programmable

□ 4MHz option

□ Compatible with STD-Z80 BUS

DESCRIPTION

The MD Series and the STD-Z80 BUS were designed to satisfy the need for low-cost OEM microcomputer modules. The STD-Z80 BUS uses a motherboard interconnect system concept and is designed to handle any MD Series card type in any slot. The modules for the STD-Z80 BUS are a compact 4.5 x 6.5 inches which provides for system partitioning by function (RAM, EPROM, I/O). This smaller module size makes system packaging easier while increasing MOS-LSI densities providing high functionality per module.

The MD Series of OEM microcomputer boards and the STD-Z80 BUS offer the most cost effective system configuration available to the OEM system designer.

MDX-A/D DESCRIPTION

The Analog to Digital Converter Module, MDX-A/D, is designed to be a 16 channel single-ended A/D module for the STD-Z80 BUS. The module is designed around the MOSTEK MK5160 8-bit A/D converter/16 channel analog multiplexer. Additional provisions have been included to allow further analog expansion if desired. Also, an optional Sample and Hold module (AD582) may be added to increase system performance. Figure 1 is a block diagram of the MDX-A/D showing the major elements of the module.

The first element of this board is the multiplexer. This 16-channel multiplexer can directly access any one of 16 single-ended analog channels and provides logic for additional channel expansion. All analog input lines contain a diode/resistor protection circuit to reduce damage potential from overvoltage and transient inputs.

The output of the multiplexer can either drive the A/D converter directly or a Sample and Hold (S/H) module version is available. The board is shipped normally without a Sample and Hold.

If an S/H function is required, an Analog Devices AD582 needs to be inserted and one jumper removed. This circuitry allow sampling of signals up to 5KHz with a nominal 150nsec aperture time.

The other half of the MK5160 is the A/D converter. The 8-bit A/D consists of 256 series resistors with an analog switch tree, a chopper stabilized comparator and a sucessive approximation register. The series resistor approach guarantees monotonicity and no missing codes. The need for external zero and full-scale adjustments has been eliminated and an absolute accuracy of ⩽ 1 LSB including quantizing error is provided. A start convert signal initiates the conversion process and can be jumper selected from either an external source or under program control. Upon completion, a DONE signal is generated to indicate end of conversion. This signal is used to flag the program as well as any external device.

The Data Bus Buffer and Interface Logic allows the MDX-A/D module to interface with the STD-Z80 BUS. It provides buffering for all signals as well as address decoding and A/D port control. A total of 4 port address locations are required and can start on any four-word boundary.

ELECTRICAL SPECIFICATIONS

WORD SIZE

Data: 8 bits
I/O Addressing: 8 bits

I/O ADDRESSING

On board programmable on 4-word boundaries
X X X X X X 0 0 A/D Port Configuration Data
X X X X X X 0 1 A/D Port Configuration Control
X X X X X X 1 0 A/D Data Input/Output Port
X X X X X X 1 1 Data Control Port

I/O CAPACITY

Eight bit analog to digital converter with up to sixteen single ended analog input channels. Channel expansion available. Start conversion and done handshake signals available at the edge connector.

Three full scale input ranges: 0 to +l Volt, 0 to +2 Volts and 0 to +5 Volts.

INTERRUPTS

Vectored interrupts generated. Interrupt vector programmable upon initialization. Daisy-chained interrupt priority. Interrupts are controlled by a MOSTEK MK3881 Parallel I/O controller chip.

SYSTEM CLOCK

	MIN	MAX
MDX-A/D	250 KHz	2.5 MHz
MDX-A/D-4	250 KHz	4.0 MHz

ELECTRICAL SPECIFICATIONS

POWER SUPPLY REQUIREMENTS

+12 Volts ±5% at 30 mA max
-12 Volts ±5% at 15 mA max
+5 Volts ± 5% at 0.6 A max

SAMPLE/HOLD OPTION DATA

DROOP RATE: 100mV at 25°C
APERTURE TIME: 150nsec
MAX INPUT FREQUENCY: 5KHz
APERTURE JITTER: 15 nsec

CONVERSION TIME

138 microseconds max

OPERATING TEMPERATURE RANGE

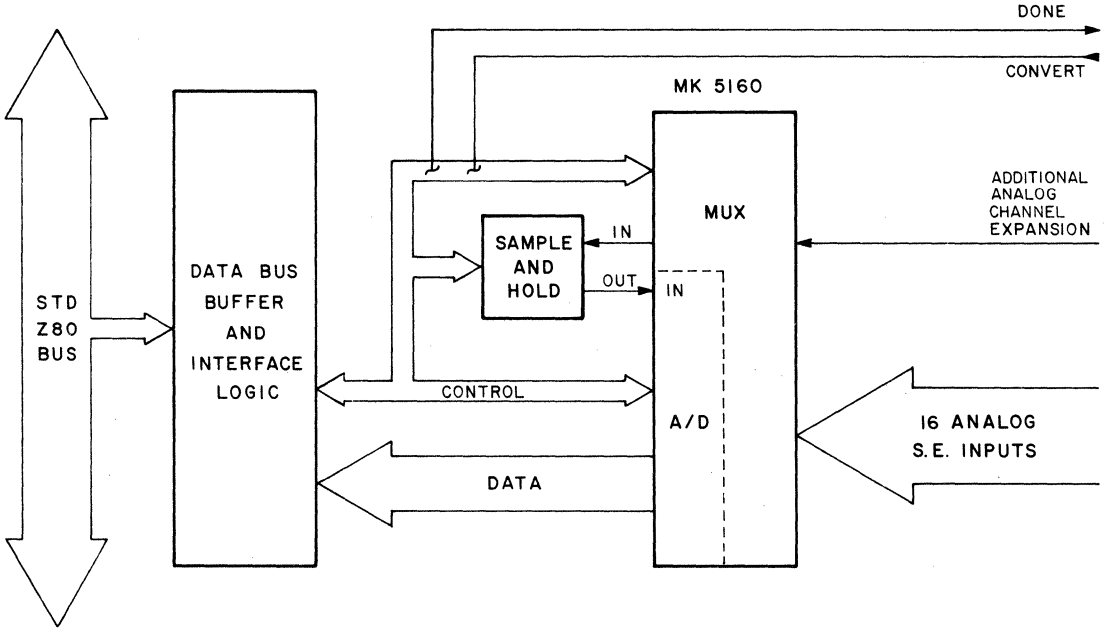
0° to +50°C

MECHANICAL SPECIFICATIONS

CARD DIMENSIONS

4.5 in. (11.43cm) high by 6.50 in. (16.51cm) long
0.48 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

MDX-A/D BLOCK DIAGRAM
Figure 1



CONNECTORS

FUNCTIONS	CONFIGURATION	MATING CONNECTOR
STD-Z80 BUS	56 pin dual 0.125 centers	Printed Circuit Viking 3VH28/1CE5 Wire Wrap Viking 3VH28/1CND5 Solder Lug Viking 3VH28/1CN5
Analog I/O	40 pin dual 0.100 centers	Ansley 609-4000

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MDX-A/D	Module with Operation Manual less mating connector: 2.5 MHz version	MK77653
MDX-A/D-4	Module with Operation Manual less mating connector: 4.0 MHz version	MK77653-4
	MDX-A/D Operations Manual only	MK79632

STANDARD LICENSE AGREEMENT AND REGISTRATION FORM

All Mostek Corporation software products are sold
on condition that the Purchaser agrees to
the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below, including, but not restricted to: object code, source code and program listings.
2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.
3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.
4. Mostek's sole obligation shall be to make available all published modifications of updates made by Mostek to licensed software products which are published within one (1) year from date of purchase, provided Purchaser has completed and returned the Software License Agreement and Registration Form.
5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.
6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

To be eligible for software product updates this form must be completed and returned to:

Mostek Corporation
Microcomputer Department
Software Librarian
MS503
P.O. Box 169
1215 W. Crosby Road
Carrollton, TX 75006

The following software products are subject to this agreement:

Part Number Description

____

____

Ship to: ________________________________

Customer

P.O. Number: ____________________ Purchase Date: ____________________

System Serial Number: ____________________

PURCHASER

By: ____________________

Title: ____________________

Company: ____________________

Date: ____________________

MOSTEK CORPORATION

By: ____________________

Title: ____________________

Date: ____________________

MOSTEK®

STD-Z80 BUS DESCRIPTION AND ELECTRICAL SPECIFICATIONS

Application Note

DESCRIPTION

The purpose of this application note is to provide the O.E.M. system designer with more information about the STD-Z80 BUS. The information presented is a bus description of the STD-Z80 BUS, the pin out, and the recommended BUS loading specifications.

In April of 1978, several meetings were held between MOSTEK CORP. and PROLOG CORP. to discuss the possibility of defining a new O.E.M. microcomputer board BUS. The goals for the new BUS were that it be simple to interface to, be well defined, and be able to use a standard 56 pin edge card connector. The results of these meetings were successful, and the STD BUS was defined.

The STD BUS was defined as a general purpose microprocessor bus which is capable of supporting the following processors: Z80, 8080, 8085, 6800, and 6809. It is possible to design simple function cards which will work with each of the processors, however it may be difficult or impossible to design an add on card which used one of the many peripheral chips and then have the card work with all of the STD BUS processors. It was for this reason that MOSTEK defined the STD-Z80 BUS. The STD-Z80 is a subset of the general purpose STD BUS and is defined exclusively for the Z80. By specifying the STD-Z80 bus, exact functional pin descriptions and bus timing can be given. Therefore, a STD-Z80 system will be guaranteed to work with all STD-Z80 designed boards.

The STD-Z80 backplane pin assignments are listed and described in Table 1. A table showing the BUS pins versus BUS signals is shown in Table 2.

STD-Z80 BUS DESCRIPTION

Table 1

BUS PIN	MNEMONIC	DESCRIPTION
1	+5V	+5Vdc system power
2	+5V	+5Vdc system power
3	GND	Ground-System signal ground and DC return
4	GND	Ground-System signal ground and DC return
5	-5V	-5Vdc system power
6	-5V	-5Vdc system power
7	D3	Data Bus (Tri—state, input/output, active high). D0-D7 constitute an 8-bit bi-directional data bus. The data bus is used for data exchange with memory and I/O devices.
8	D7	
9	D2	
10	D6	
11	D1	
12	D5	
13	D0	
14	D4	
15	A7	Address Bus (Tri-state, output, active high). A0-A15 make up a 16-bit address bus. The address bus provides the address for memory (up to 65K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the lower 8 address bits to allow the user to directly select up to 256 input or 256 output ports. A0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh refresh address for dynamic memories.
16	A15	
17	A6	
18	A14	
19	A5	
20	A13	
21	A4	
22	A12	
23	A3	
24	A11	
25	A2	
26	A10	
27	A1	
28	A9	
29	A0	
30	A8	
31	/WR	Write (Tri-state,output, active low) /WR indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.
32	/RD	Read (Tri-state, output, active low). RD indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.
33	/IORQ	Input/Output Request (Tri-state, output, active low). The /IORQ signal indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. An /IORQ signal is also generated with a /M1 signal when an interrupt is being acknowledged to indicate that an interrupt

		response vector can be placed on the data bus. Interrupt Acknowledge operations occur during /M1 time, while I/O operations never occur during /M1 time.
34	/MEMRQ	Memory Request (Tri-state, output, active low). The /MEMRQ signal indicates that the address bus holds a valid address for a memory read or memory write operation.
35	/IOEXP	I/O Expansion, not used on MDX cards. (Normally strapped to ground on the MOSTEK motherboard)
36	/MEMEX	Memory Expansion, not used on Mostek MDX cards.(Normally strapped to ground on the MOSTEK motherboard)
37	/REFRESH	REFRESH (Tri-state, output, active low). /REFRESH indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the /MEMRQ signal should be used to perform a refresh cycle for all dynamic RAMs in the system. During the refresh cycle A7 is a logic 0 and the upper 8 bits of the address bus contains the I register.
38	/MCSYNC	Not generated on the MOSTEK MDX-CPU1. Can be generated by gating the following signals:/RD+ /WR + /INTAK. By connecting a jumper on the MDX-CPU1, this line becomes /DEBUG (Input). /DEBUG is used in conjunction with the DDT-80 operating system on the MDX-DEBUG card, and the MDX-SST card for implementing a hardware single step function. When pulled low, the /DEBUG line will set an address modification latch which will force the upper three address lines A15, A14, and A13 to a logic 1. These address lines will remain at a logic 1 until reset by performing any I/O operation.
39	/STATUS 1	Machine Cycle One (Tri-state, output, active low). /M1 indicates that the current machine cycle is in the op code fetch cycle of an instruction. Note that during the execution of two-byte op-codes /M1 will be generated as each op-code is fetched. These two-byte op-codes always begin with a CBh, DDh, EDh, or FDh. /M1 also occurs with IORQ to indicate an interrupt acknowledge cycle.
40	/STATUS 0	Not used on Mostek MDX cards.
41	/BUSAK	Bus Acknowledge (Output, active low). Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus, and control bus signals have been set to their high impedance state and the external device can now control the bus.
42	/BUSRQ	Bus Request (Input, active low). The /BUSRQ signal is used to request the CPU address bus, data bus, and control signal bus to go to a high impedance state so that other devices can control those buses. When /BUSRQ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated and the /BUSAK signal is activated.
43	/INTAK	Interrupt Acknowledge (Tri-state, output, active low). The /INTAK signal indicates that an interrupt acknowledge cycle is in progress, and the interrupting device should place its response vector on the data bus. The /INTAK signal is equivalent to an IORQ during an /M1.
44	/INTRQ	Interrupt Request (Input, active low). The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip flop (IFF)

		is enabled and if the BUSRQ signal is not active. When the CPU accepts the interrupt, an interrupt acknowledge signal /INTAK (IORQ during an M1) is sent out at the beginning of the next instruction.
45	/WAITRQ	Wait Request (Input, active low). Wait Request indicates to the CPU that the addressed memory or I/O device is not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU. Use of this signal postpones refresh as long as it held active.
46	/NMIRQ	Non-Maskable Interrupt Request (Input, negative edge triggered). The Non-Maskable Interrupt Request line has a higher priority than the /INTRQ line and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. /NMIRQ automatically forces the CPU to restart to location 0066h. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous WAIT cycles can prevent the current instruction from ending and that a /BUSRQ will override a /NMIRQ.
47	/SYSRESET	System Reset (Output, active low). The System Reset line indicates that a reset has been generated either from an external reset or the power on reset circuit. The system reset will occur only once per reset and will be approximately 2 microseconds in duration. A system reset will also force the CPU program counter to zero, disable interrupts, set the I register to 00h, set the R register to 00h, and set Interrupt Mode 0.
48	/PBRESET	Push Button Reset (Input, active low). The Push Button Reset will generate a debounced system reset.
49	/CLOCK	Processor Clock (Output, active low). Single phase system clock.
50	/CNTRL	Not used on MOSTEK MDX cards.
*51	PCO	Priority Chain Output (Output, active high). The signal is used to form a priority-interrupt daisy chain when more than one interrupt-driven device is being used. A high level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.
*52	PCI	Priority Chain In (Input, active high). This signal is used to form a priority-interrupt daisy chain when more than one interrupt-driven device is being used. A high level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.
53	AUX GND	Auxiliary Ground (Bussed)
54	AUX GND	Auxiliary Ground (Bussed)
55	+12V	+12Vdc system power
56	-12V	-12Vdc system power

NOTES:

1. Input/Output references of each signal are made with respect to MDX-CPU1 module.
2. The following signals have pull-up resistors: /WR, /RD, /IORQ, /MEMRQ, /REFRESH, /DEBUG, /M1, /BUSRQ, /INTAK, /INTRQ, /WAITRQ, /NMIRQ, /SYSRESET, /PBRESET, and /CLOCK. The value of the pull-up resistors are 1K except for /WAITRQ which is 500 ohms and /PBRESET which is 10K ohms. These resistors are located on the MDX-CPU1 module.

*The Mostek card cage is prioritized from left to right as viewed from the top with component side of boards to the left.

STD—Z80 ELECTRICAL BUS SPECIFICATIONS

BUS RECEIVERS

Logical Low: 0.8V maximum at -0.36mA
Logical High: 2.0V minimum at 20μA

BUS DRIVERS

Logical Low: 0.5V maximum at 24mA
Logical High: 2.4V minimum at -15mA
Off State Output Current (tri-state): ±100μA

RECOMMENDED BUS DRIVERS AND RECEIVERS

Bus Drivers: 74LS240, 74LS241, 74LS373, 74LS374, and 74LS244.
Bus Receivers: 74LS240, 74LS241, and 74LS244.
Bus Transceivers: 74LS245, 74LS242, and 74LS243.

STD-Z80 BUS PIN-OUT

Table 2

Pin	Component Side Mnemonic	Pin	Circuit Side Mnemonic
1	+5V	2	+5V
3	GND	4	GND
5	-5V	6	-5V
7	D3	8	D7
9	D2	10	D6
11	D1	12	D5
13	D0	14	D4
15	A7	16	A15
17	A6	18	A14
19	A5	20	A13
21	A4	22	A12
23	A3	24	A11
25	A2	26	A10
27	A1	28	A9
29	A0	30	A8
31	/WR	32	/RD
33	/IORQ	34	/MEMRQ
35	/IOEXP	36	/MEMEX
37	/REFRESH	38	/MCSYNC
39	/STATUS 1	40	/STATUS 0
41	/BUSAK	42	/BUSRQ
43	/INTAK	44	/INTRQ
45	/WAITRQ	46	/NMIRQ
47	/SYSRESET	48	/PBRESET
49	/CLOCK	50	/CNTRL
51	PCO	52	PCI
53	AUX GND	54	AUX GND
55	+12V	56	-12V

1979 MICROCOMPUTER DATA BOOK

PRELIMINARY

MOSTEK®

MD SERIES MICROCOMPUTER MODULES

Z80 Single Board Computer (MD-SBC1)

FEATURES

- □ Z80 Microprocessor
- □ 2K byte RAM capacity with 1K included
- □ Sockets for 8K bytes 2716 EPROM
- □ Crystal Clock - 2.5 MHz
- □ Three TTL buffered 8-bit OUTPUT ports
- □ Two TTL buffered 8-bit INPUT ports
- □ Two Interrupt Inputs
- □ Single +5 volt power supply

DESCRIPTION

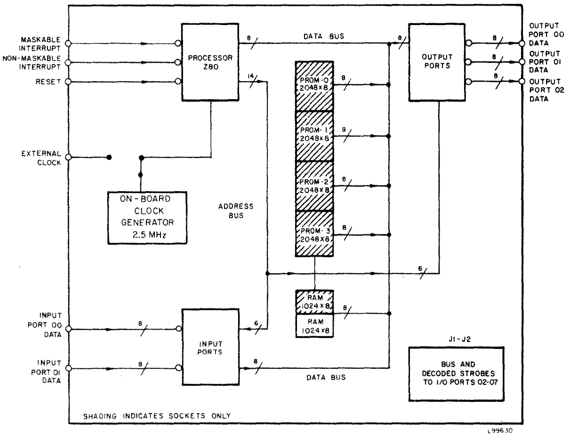
The MD-SBC1 is a complete Z80 based microcomputer on 4 ½ in. by 6 ½ in. circuit module. All I/O is fully TTL buffered and is brought to a 56 pin edge connector.

The smaller card size and the single power supply makes the MD-SBC1 easier to package and easier to use than most other modules. While the module size is small no compromises have been made in computing power due to increasing MOS-LSI densities and the use of the Z80 microcomputer. The 40 buffered TTL I/O lines and the 8K bytes of EPROM provide the capability to solve many control problems encountered by the OEM microcomputer user. The expandable MD Series (MDX) has the same form factor allowing easy expansion to a multi-board system with increased capability.

Figure 1 is a block diagram of the MD-SBC1. The basic module comes with 1K bytes of RAM expandable to 2K bytes by the addition of two 2114 type RAMs. Four 2716 sockets are provided for up to 8K bytes of EPROM, and are decoded in 2K blocks starting at address zero. The output ports are 74LS244 latches which are brought to the card cage connector. The input ports are 74LS240 Octal Buffers with 4.7K OHM pull-up resistors on the inputs. These input lines are also brought to the edge connector. The Z80-CPU is driven by a crystal clock at 2.5MHz (400nsec T-State).

Both the $\overline{\text{NMI}}$ and $\overline{\text{INT}}$ interrupt inputs to the Z80-CPU are terminated with 4.7K Ohm pull ups and brought to the card edge connector. An external clock can be used by changing strapping options on the board. Power on reset circuitry is included on the CPU's $\overline{\text{RESET}}$ input. Provision is made to expand the I/O capability through the use of on-board connectors.

MD-SBC1 BLOCK DIAGRAM
Figure 1



ELECTRICAL SPECIFICATIONS

WORD SIZE

INSTRUCTION 8,16, 24 or 32 bits
DATA 8 bits

CYCLE TIME

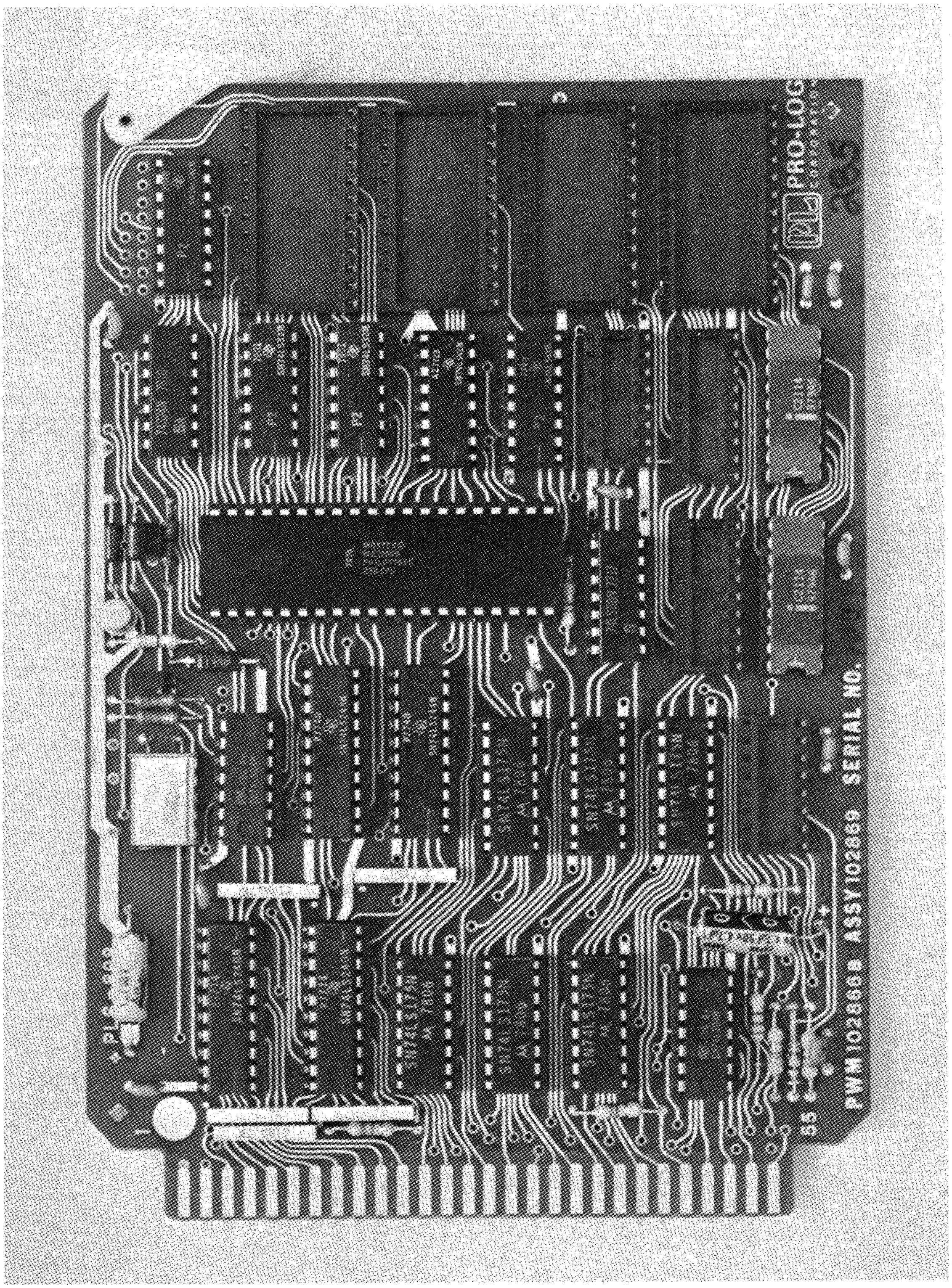
T-STATE = 400nSec, fastest instruction is 1.6 microsecond.

MEMORY ADDRESSING

EPROM NUMBER	HEX ADDRESS
0	0000-07FF
1	0800-0FFF
2	1000-17FF
3	1800-1FFF

RAM NUMBER	HEX ADDRESS
STANDARD	2000-23FF
OPTIONAL	2400-27FF

MD-SBC 1 BOARD PHOTO



MEMORY CAPACITY

8 K bytes of 2716 memory (none included)
2 K bytes of 2114 memory (1K bytes included)

MEMORY SPEED REQUIRED

Memory	Access Time Required	Cycle Time Required
2716*	450nSec	450nSec
2114	450nSec	450nSec

* Single 5 volt type required

I/O ADDRESSING AND CAPACITY

PORT TYPE	HEX ADDRESS	DATA CAPACITY
Input	00 and 01	16 lines
Output	00, 01, 02	24 lines

INTERRUPTS

Two active low; $\overline{NMI}$ and $\overline{INT}$. See Z80-CPU (MK3880) Technical Manual for a full description of Z80 interrupts.

I/O INTERFACES

Inputs - One 74LS load plus a 4.7K Ohm pull up resistor
Outputs - I_{OH} = -15mA at V_{OH} = 2.4 volts
I_{OL} = 24mA at V_{OL} = 0.5 volts

SYSTEM CLOCK

	MIN	MAX
MD-SBC1	250KHz	2.5MHz

POWER SUPPLY REQUIREMENTS

+5 volts ± 5% at 1.2A max (fully loaded)
(100mA per RAM, 100mA per EPROM)

OPERATING TEMPERATURE RANGE

0°C to +50°C

MECHANICAL SPECIFICATIONS

CARD DIMENSIONS

4.5 in. (11.43cm) high by 6.50 in. (16.51cm) long
048 in. (1.22cm) maximum profile thickness
0.062 in. (0.16cm) printed circuit board thickness

CONNECTORS

FUNCTION	CONFIGURATION	MATING CONNECTOR
Paralled I/O	56 pin (28 position)	Printed Circuit VIKING 3VH-28/1CE5
	0.125 in centers	Wire Wrap VIKING 3VH-28/1CND5
		Solder Lug VIKING 3VH-28/1CN5

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MD-SBC1	Complete Z80 Single Board Computer with Operations Manual less EPROMs and mating connector.	MK77851
	MD-SBC1 Operations Manual only.	MK79609
MDX-PROTO Data Sheet	MD Series prototyping package	MK78605
AID-80F Data Sheet	Disk based development system for MD Series	MK78568
AIM-80 Data Sheet	Z80 In-Circuit Emulation Module for AID-80F	MK78537

1979 MICROCOMPUTER DATA BOOK

MK 3880 CENTRAL PROCESSING UNIT

Z80
Family

TABLE OF CONTENTS

1.0 INTRODUCTION

The term "microcomputer" has been used to describe virtually every type of small computing device designed within the last few years. This term has been applied to everything from simple "microprogrammed" controllers constructed out of TTL MSI up to low end minicomputers with a portion of the CPU constructed out of TTL LSI "bit slices." However, the major impact of the LSI technology within the last few years has been with MOS LSI. With this technology, it is possible to fabricate complete and very powerful computer systems with only a few MOS LSI components.

The Mostek Z80 family of components is a significant advancement in the state-of-art of microcomputers. These components can be configured with any type of standard semiconductor memory to generate computer systems with an extremely wide range of capabilities. For example, as few as two LSI circuits and three standard TTL MSI packages can be combined to form a simple controller. With additional memory and I/O devices a computer can be constructed with capabilities that only a minicomputer could previously deliver. This wide range of computational power allows standard modules to be constructed by a user that can satisfy the requirements of an extremely wide range of applications.

The major reason for MOS LSI domination of the microcomputer market is the low cost of these few LSI components. For example, MOS LSI microcomputers have already replaced TTL logic in such applications as terminal controllers, peripheral device controllers, traffic signal controllers, point of sale terminals, intelligent terminals and test systems. In fact the MOS LSI microcomputer is finding its way into almost every product that now uses electronics and it is even replacing many mechanical systems such as weight scales and automobile controls.

The MOS LSI microcomputer market is already well established and new products using them are being developed at an extraordinary rate. The Mostek Z80 component set has been designed to fit into this market through the following factors:

1. The Z80 is fully software compatible with the popular 8080A CPU offered from several sources. Existing designs can be easily converted to include the Z80 as a superior alternative.

2. The Z80 component set is superior in both software and hardware capabilities to any other 8-bit microcomputer system on the market. These capabilities provide the user with significantly lower hardware and software development costs while also allowing him to offer additional features in his system.

3. A complete development and OEM system product line including full software support is available to enable the user to easily develop new products.

Microcomputer systems are extremely simple to construct using Z80 components. Any such system consists of three parts:

1. CPU (Central Processing Unit)

2. Memory

3. Interface circuits to peripheral devices

The CPU is the heart of the system. Its function is to obtain instructions from the memory and perform the desired operations. The memory is used to contain instructions and in most cases data that is to be processed. For example, a typical instruction sequence may be to read data from a specific peripheral device, store it in a location in memory, check the parity and write it out to another peripheral device. Note that the Mostek component set includes the CPU and various general purpose I/O device controllers, as well as a wide range of memory devices. Thus, all required components can be connected together in a very simple manner with virtually no other external logic. The user's effort then becomes primarily one of software development. That is, the user can concentrate on describing his problem and translating it into a series of instructions that can be loaded into the microcomputer memory. Mostek is dedicated to making this step of software generation as simple as possible. A good example of this is our assembly language in which a simple mnemonic is used to represent every instruction that the CPU can perform. This language is self documenting in such a way that from the mnemonic the user can understand exactly what the instruction is doing without constantly checking back to a complex cross listing.

2.0 Z80-CPU ARCHITECHURE

A block diagram of the internal architecture of the Z80-CPU is shown in Figure 2.0-1 The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.

Z80-CPU BLOCK DIAGRAM

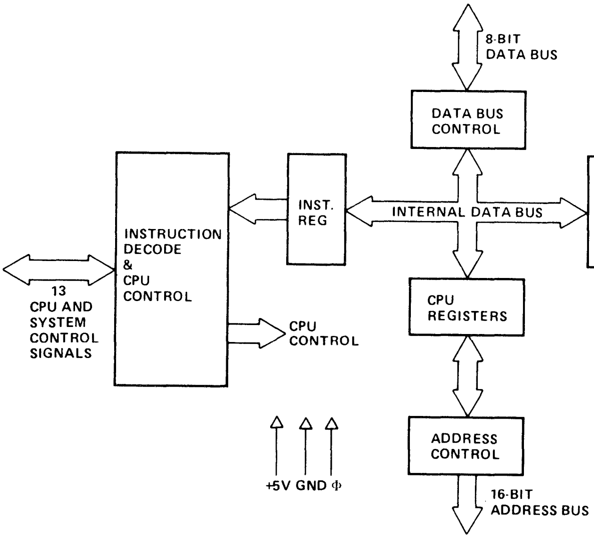


FIGURE 2.0-1

2.1 CPU REGISTERS

The Z80-CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 2.0-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z80 registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Special Purpose Registers

1. **Program Counter (PC).** The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.

2. **Stack Pointer (SP).** The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.

Z80-CPU REGISTER CONFIGURATION

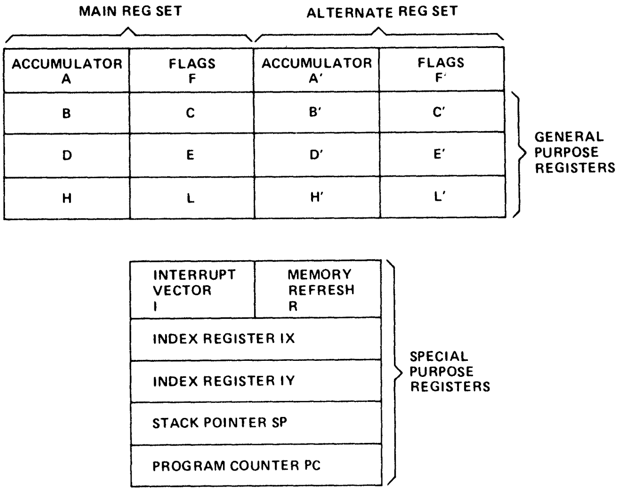


FIGURE 2.0-2

3. **Two Index Registers (IX & IY).** The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.

4. **Interrupt Page Address Register (I).** The Z80-CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I Register is used for this purpose to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

5. **Memory Refresh Register (R).** The Z80-CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. This 7-bit register is automatically incremented after each instruction fetch. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer.

Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the flag register indicates specific conditions for 8 or 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair that he wishes to work with with a single exchange instruction so that he may easily work with either pair.

General Purpose Registers

There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. One set is called BC, DE, and HL while the complementary set is called BD', DE' and HL'. At any one time the programmer can select either set of registers to work with through a single exchange command for the entire set. In systems where fast interrupt response is required, one set of general purpose registers and an accumulator/flag register may be reserved for handling this very fast routine. Only a simple exchange command need be executed to go between the routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general purpose registers are used for a wide range of applications by the programmer. They also simplify programming, especially in ROM based systems where little external read/write memory is available.

2.2 ARITHMETIC & LOGIC UNIT (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally the ALU communicates with the registers and the external data bus on the internal data bus. The type of functions performed by the ALU include:

Add

Subtract

Logical AND

Logical OR

Logical Exclusive OR

Compare

Left or right shifts or rotates (arithmetic and logical)

Increment

Decrement

Set bit

Reset bit

Test bit

2.3 INSTRUCTION REGISTER AND CPU CONTROL

As each instruction is fetched from memory, it is placed in the instruction register and decoded. The control section performs this function and then generates and supplies all of the control signals necessary to read or write data from or to the registers, controls the ALU and provides all required external control signals.

3.0 Z80-CPU PIN DESCRIPTION

The Z80-CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in Figure 3.0-1 and the function of each is described below.

Z80 PIN CONFIGURATION

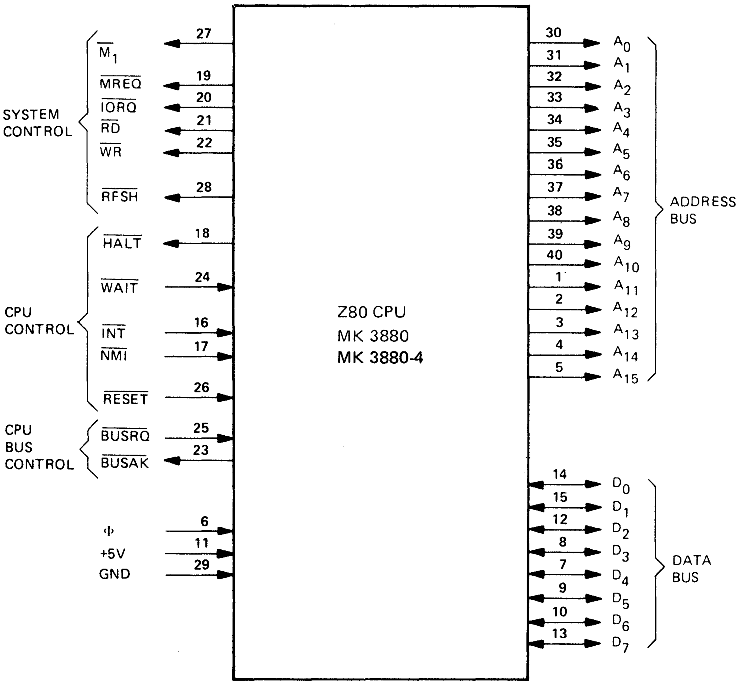


FIGURE 3.0-1

A_0-A_{15}
(Address Bus)

Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7
(Data Bus)

Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$
(Machine Cycle one)

Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M_1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH, or FDH. $\overline{M_1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$
(Memory Request)

Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{IORQ}$
(Input/Output Request)

Tri-state output, active low. The $\overline{IORQ}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An $\overline{IORQ}$ signal is also generated with an $\overline{M_1}$ signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during M_1 time while I/O operations never occur during M_1 time.

$\overline{RD}$
(Memory Read)

Tri-state output, active low. $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{WR}$
(Memory Write)

Tri-state output, active low. $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.

$\overline{RFSH}$
(Refresh)

Output, active low. $\overline{RFSH}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and current $\overline{MREQ}$ signal should be used to do a refresh read to all dynamic memories. A_7 is a logic zero and the upper 8 bits of the Address Bus contains the I Register.

$\overline{HALT}$
(Halt state)

Output, active low. $\overline{HALT}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.

$\overline{WAIT}$*
(Wait)

Input, active low. $\overline{WAIT}$ indicates to the Z80-CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU.

$\overline{INT}$
(Interrupt Request)

Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled and if the $\overline{BUSRQ}$ signal is not active. When the CPU accepts the interrupt, an acknowledge signal ($\overline{IORQ}$ during M_1 time) is sent out at the beginning of the next instruction cycle. The CPU can respond to an interrupt in three different modes that are described in detail in section 8.

$\overline{NMI}$

Input, negative edge triggered. The non maskable interrupt request line has a higher priority than $\overline{INT}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{NMI}$ automatically forces the Z80-CPU to restart to location 0066_H. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous WAIT cycles can prevent the current instruction from ending, and that a $\overline{BUSRQ}$ will override a $\overline{NMI}$.

$\overline{\text{RESET}}$

Input, active low. $\overline{\text{RESET}}$ forces the program counter to zero and initializes the CPU. The CPU initialization includes:

1) Disable the interrupt enable flip-flop
2) Set Register I = 00_H
3) Set Register R = 00_H
4) Set Interrupt Mode 0

During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state. No refresh occurs.

$\overline{\text{BUSRQ}}$
(Bus Request)

Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When $\overline{\text{BUSRQ}}$ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated.

$\overline{\text{BUSAK}}$*
(Bus Acknowledge)

Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Φ

Single phase system clock.

*While the Z80-CPU is in either a $\overline{\text{WAIT}}$ state or a Bus Acknowledge condition, Dynamic Memory Refresh will not occur.

4.0 CPU TIMING

The Z80-CPU executes instructions by stepping through a very precise set of a few basic operations. These include:

Memory read or write

I/O device read or write

Interrupt acknowledge

All instructions are merely a series of these basic operations. Each of these basic operations can take from three to six clock periods to complete or they can be lengthened to synchronize the CPU to the speed of external devices. The basic clock periods are referred to as T states and the basic operations are referred to as M (for machine) cycles. Figure 4.0-0 illustrates how a typical instruction will be merely a series of specific M and T cycles. Notice that this instruction consists of three machine cycles (M1, M2 and M3). The first machine cycle of any instruction is a fetch cycle which is four, five or six T states long (unless lengthened by the wait signal which will be fully described in the next section). The fetch cycle (M1) is used to fetch the OP code of the next instruction to be executed. Subsequent machine cycles move data between the CPU and memory or I/O devices and they may have anywhere from three to five T cycles (again they may be lengthened by wait states to synchronize the external devices to the CPU). The following paragraphs describe the timing which occurs within any of the basic machine cycles. In section 7, the exact timing for each instruction is specified.

BASIC CPU TIMING EXAMPLE

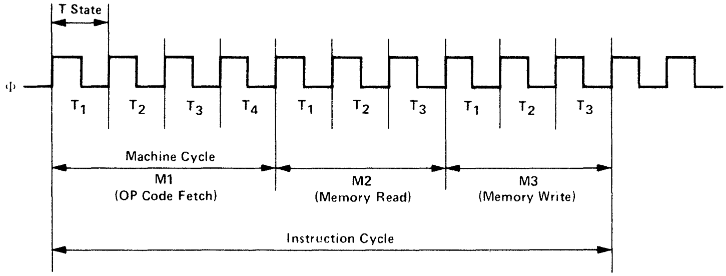


FIGURE 4.0-0

All CPU timing can be broken down into a few very simple timing diagrams as shown in Figure 4.0-1 through 4.0-7. These diagrams show the following basic operations with and without wait states (wait states are added to synchronize the CPU to slow memory or I/O devices).

4.0-1. Instruction OP code fetch (M1 cycle)

4.0-2. Memory data read or write cycles

4.0-3. I/O read or write cycles

4.0-4. Bus Request/Acknowledge Cycle

4.0-5. Interrupt Request/Acknowledge Cycle

4.0-6. Non maskable Interrupt Request/Acknowledge Cycle

4.0-7. Exit from a HALT instruction

INSTRUCTION FETCH

Figure 4.0-1 shows the timing during an M1 cycle (OP code fetch). Notice that the PC is placed on the address bus at the beginning of the M1 cycle. One half clock time later the $\overline{\text{MREQ}}$ signal goes active. At this time the address to the memory has had time to stabilize so that the falling edge of $\overline{\text{MREQ}}$ can be used directly as a chip enable clock to dynamic memories. The $\overline{\text{RD}}$ line also goes active to indicate that the memory read data should be enabled onto the CPU data bus. The CPU samples the data from the memory on the data bus with the rising edge of the clock of state T3 and this same edge is used by the CPU to turn off the $\overline{\text{RD}}$ and $\overline{\text{MREQ}}$ signals. Thus the data has already been sampled by the CPU before the $\overline{\text{RD}}$ signal becomes inactive. Clock state T3 and T4 of a fetch cycle are used to refresh dynamic memories. (The CPU uses this time to decode and execute the fetched instruction so that no other operation could be performed at this time). During T3 and T4 the lower 7 bits of the address bus contain a memory refresh address and the $\overline{\text{RFSH}}$ signal becomes active to indicate that a refresh read of all dynamic memories should be accomplished. Notice that a $\overline{\text{RD}}$ signal is not generated during refresh time to prevent data from different memory segments from being gated onto the data bus. The $\overline{\text{MREQ}}$ signal during refresh time should be used to perform a refresh read of all memory elements. The refresh signal can not be used by itself since the refresh address is only guaranteed to be stable during $\overline{\text{MREQ}}$ time.

INSTRUCTION OP CODE FETCH

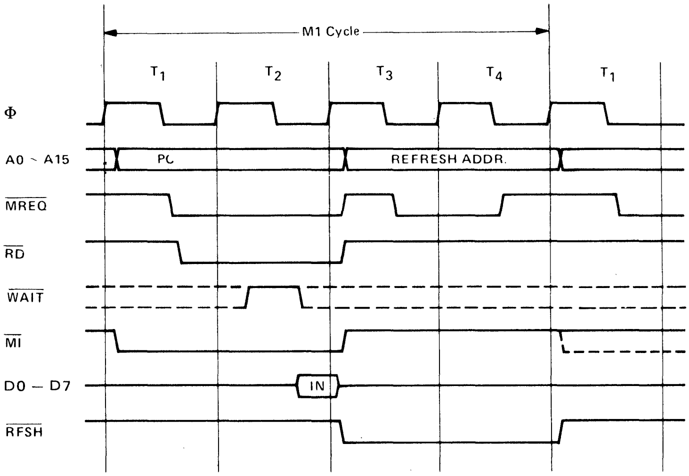


FIGURE 4.0-1

Figure 4.0-1A illustrates how the fetch cycle is delayed if the memory activates the $\overline{\text{WAIT}}$ line. During T2 and every subsequent Tw, the CPU samples the $\overline{\text{WAIT}}$ line with the falling edge of Φ. If the $\overline{\text{WAIT}}$ line is active at this time, another wait state will be entered during the following cycle. Using this technique the read cycle can be lengthened to match the access time of any type of memory device.

INSTRUCTION OP CODE FETCH WITH WAIT STATES

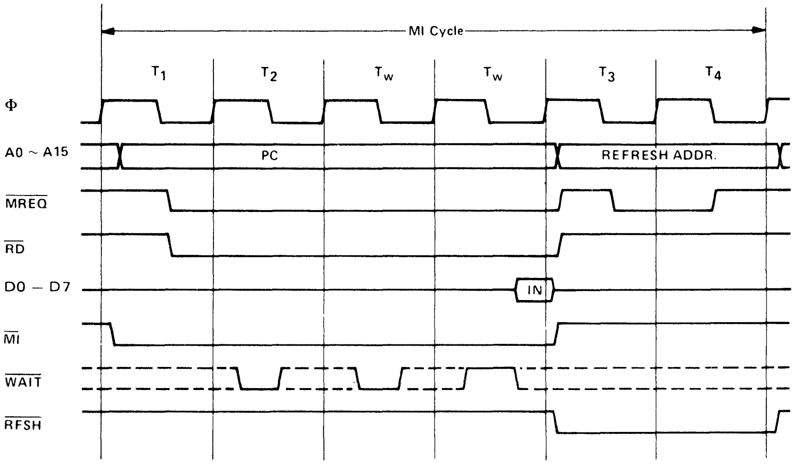


FIGURE 4.0-1A

MEMORY READ OR WRITE

Figure 4.0-2 illustrates the timing of memory read or write cycles other than an OP code fetch (M1 cycle). These cycles are generally three clock periods long unless wait states are requested by the memory via the WAIT signal. The MREQ signal and the RD signal are used the same as in the fetch cycle. In the case of a memory write cycle, the MREQ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The WR line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory. Furthermore the WR signal goes inactive one half T state before the address and data bus contents are changed so that the overlap requirements for virtually any type of semiconductor memory type will be met.

MEMORY READ OR WRITE CYCLES

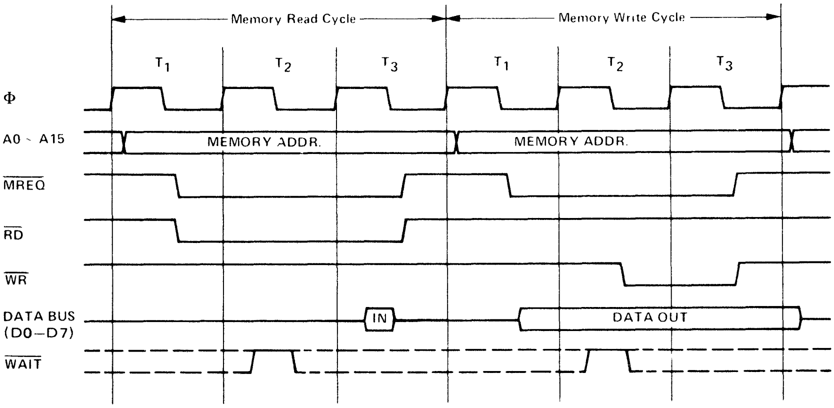


FIGURE 4.0-2

Figure 4.0-2A illustrates how a $\overline{\text{WAIT}}$ request signal will lengthen any memory read or write operation. This operation is identical to that previously described for a fetch cycle. Notice in this figure that a separate read and a separate write cycle are shown in the same figure although read and write cycles can never occur simultaneously.

MEMORY READ OR WRITE CYCLES WITH WAIT STATES

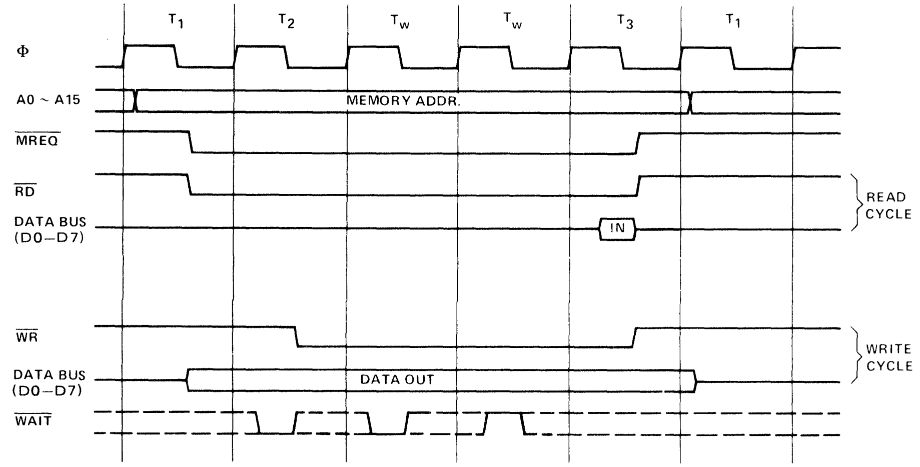


FIGURE 4.0-2A

INPUT OR OUTPUT CYCLES

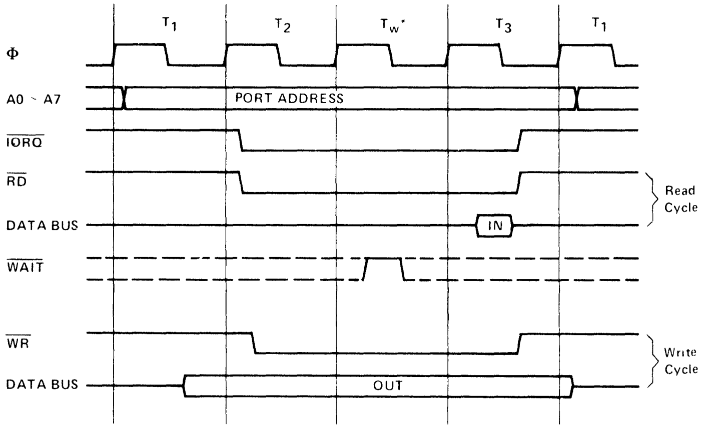
Figure 4.0-3 illustrates an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted. The reason for this is that during I/O operations, the time from when the $\overline{\text{IORQ}}$ signal goes active until the CPU must sample the $\overline{\text{WAIT}}$ line is very short and without this extra state sufficient time does not exist for an I/O port to decode its address and activate the $\overline{\text{WAIT}}$ line if a wait is required. Also, without this wait state it is difficult to design MOS I/O devices that can operate at full CPU speed. During this wait state time the $\overline{\text{WAIT}}$ request signal is sampled. During a read I/O operation, the $\overline{\text{RD}}$ line is used to enable the addressed port onto the data bus just as in the case of a memory read. For I/O write operations, the $\overline{\text{WR}}$ line is used as a clock to the I/O port, again with sufficient overlap timing automatically provided so that the rising edge may be used as a data clock.

Figure 4.0-3A illustrates how additional wait states may be added with the $\overline{\text{WAIT}}$ line. The operation is identical to that previously described.

BUS REQUEST/ACKNOWLEDGE CYCLE

Figure 4.0-4 illustrates the timing for a Bus Request/Acknowledge cycle. The $\overline{\text{BUSRQ}}$ signal is sampled by the CPU with the rising edge of the last clock period of any machine cycle. If the $\overline{\text{BUSRQ}}$ signal is active, the CPU will set its address, data and tri-state control signals to the high impedance state with the rising edge of the next clock pulse. At that time any external device can control the buses to transfer data between memory and I/O devices. (This is generally known as Direct Memory Access [DMA] using cycle stealing). The maximum time for the CPU to respond to a bus request is the length of a machine cycle and the external controller can maintain control of the bus for as many clock cycles as is desired. Note, however, that if very long DMA cycles are used, and dynamic memories are being used, the external controller must also perform the refresh function. This situation only occurs if very large blocks of data are transferred under DMA control. Also note that during a bus request cycle, the CPU cannot be interrupted by either a $\overline{\text{NMI}}$ or an $\overline{\text{INT}}$ signal.

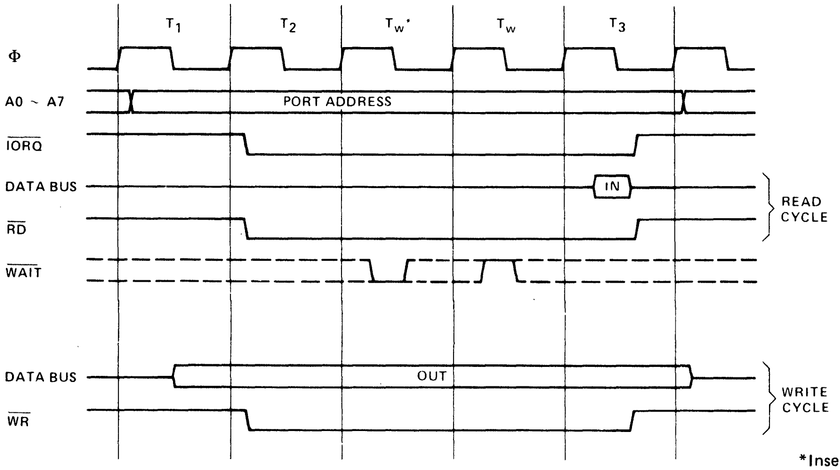
INPUT OR OUTPUT CYCLES



*Inserted by Z80 CPU

FIGURE 4.0-3

INPUT OR OUTPUT CYCLES WITH WAIT STATES



*Inserted by Z80 CPU

FIGURE 4.0-3A

BUS REQUEST/ACKNOWLEDGE CYCLE

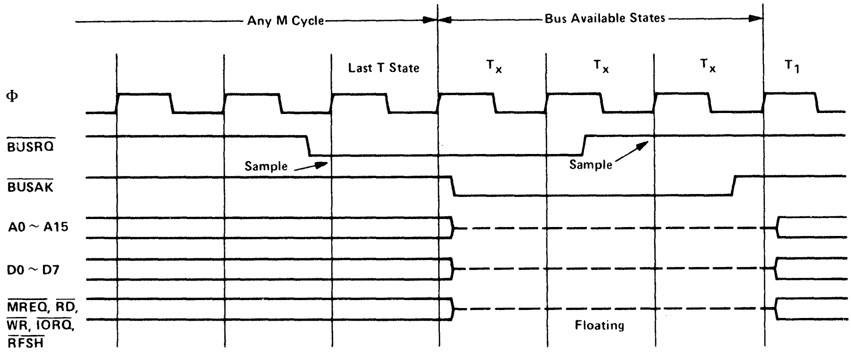
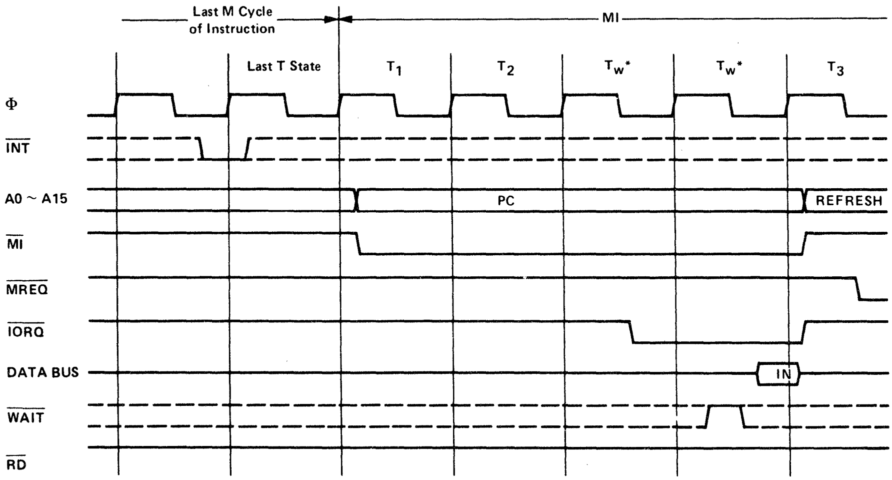


FIGURE 4.0-4

INTERRUPT REQUEST/ ACKNOWLEDGE CYCLE

Figure 4.0-5 illustrates the timing associated with an interrupt cycle. The interrupt signal (INT) is sampled by the CPU with the rising edge of the last clock at the end of any instruction. The signal will not be accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the BUSRQ signal is active. When the signal is accepted a special M1 cycle is generated. During this special M1 cycle the IORQ signal becomes active (instead of the normal MREQ) to indicate that the interrupting device can place an 8-bit vector on the data bus. Notice that two wait states are automatically added to this cycle. These states are added so that a ripple priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the ripple signals to stablilize and identify which I/O device must insert the response vector. Refer to section 8.0 for details on how the interrupt response vector is utilized by the CPU.

INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

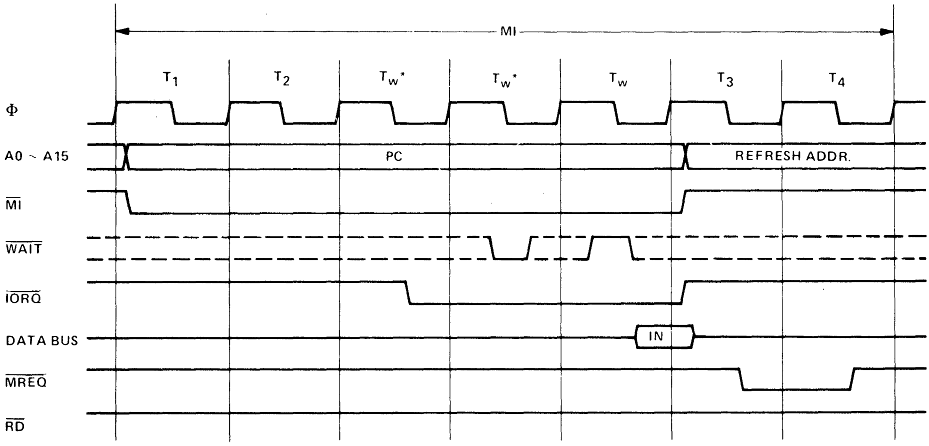


Mode 0 shown

FIGURE 4.0-5

Figure 4.0-5A illustrates how additional wait states can be added to the interrupt response cycle. Again the operation is identical to that previously described.

INTERRUPT REQUEST/ACKNOWLEDGE WITH WAIT STATES



Mode **0** shown

FIGURE 4.0-5A

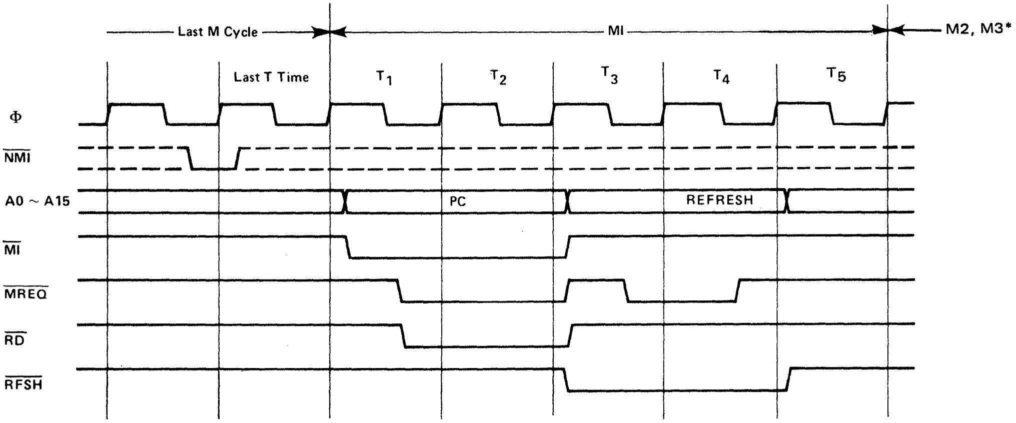
NON MASKABLE INTERRUPT RESPONSE

Figure 4.0-6 illustrates the request/acknowledge cycle for the non-maskable interrupt. A pulse on the $\overline{\text{NMI}}$ input sets an internal NMI latch which is tested by the CPU at the end of every instruction. This NMI latch is sampled at the same time as the interrupt line, but this line has priority over the normal interrupt and it can not be disabled under software control. Its usual function is to provide immediate response to important signals such as an impending power failure. The CPU response to a non maskable interrupt is similar to a normal memory read operation. The only difference being that the content of the data bus is ignored while the processor automatically stores the PC in the external stack and jumps to location 0066_H. The service routine for the non maskable interrupt must begin at this location if this interrupt is used.

HALT EXIT

Whenever a software halt instruction is executed the CPU begins executing NOP's until an interrupt is received (either a non-maskable or a maskable interrupt while the interrupt flip flop is enabled). The two interrupt lines are sampled with the rising clock edge during each T4 state as shown in Figure4.0-7. If a non-maskable interrupt has been received or a maskable interrupt has been received and the interrupt enable flip-flop is set, then the halt state will be exited on the next rising clock edge. The following cycle will then be an interrupt acknowledge cycle corresponding to the type of interrupt that was received. If both are received at this time, then the non maskable one will be acknowledged since it was highest priority. The purpose of executing NOP instructions while in the halt state is to keep the memory refresh signals active. Each cycle in the halt state is a normal M1 (fetch) cycle except that the data received from the memory is ignored and a NOP instruction is forced internally to the CPU. The halt acknowledge signal is active during this time to indicate that the processor is in the halt state.

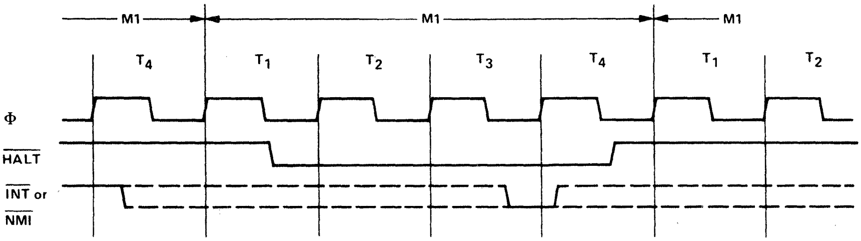
NON MASKABLE INTERRUPT REQUEST OPERATION



*M2 and M3 are stack write operations

FIGURE 4.0-6

HALT EXIT



HALT INSTRUCTION
IS RECEIVED
DURING THIS
MEMORY CYCLE

FIGURE 4.0-7

5.0 Z80-CPU INSTRUCTION SET

The Z80-CPU can execute 158 different instruction types including all 78 of the 8080A CPU. The instructions can be broken down into the following major groups:

- Load and Exchange
- Block Transfer and Search
- Arithmetic and Logical
- Rotate and Shift
- Bit Manipulation (set, reset, test)
- Jump, Call and Return
- Input/Output
- Basic CPU Control

5.1 INTRODUCTION TO INSTRUCTION TYPES

The load instructions move data internally between CPU registers or between CPU registers and external memory. All of these instructions must specify a source location from which the data is to be moved and a destination location. The source location is not altered by a load instruction. Examples of load group instructions include moves between any of the general purpose registers such as move the data to Register B from Register C. This group also includes load immediate to any CPU register or to any external memory location. Other types of load instructions allow transfer between CPU registers and memory locations. The exchange instructions can trade the contents of two registers.

A unique set of block transfer instructions is provided in the Z80. With a single instruction a block of memory of any size can be moved to any other location in memory. This set of block moves is extremely valuable when large strings of data must be processed. The Z80 block search instructions are also valuable for this type of processing. With a single instruction, a block of external memory of any desired length can be searched for any 8-bit character. Once the character is found the instruction automatically terminates. Both the block transfer and the block search instructions can be interrupted during their execution so as to not occupy the CPU for long periods of time.

The arithmetic and logical instructions operate on data stored in the accumulator and other general purpose CPU registers or external memory locations. The results of the operations are placed in the accumulator and the appropriate flags are set according to the result of the operation. An example of an arithmetic operation is adding the accumulator to the contents of an external memory location. The results of the addition are placed in the accumulator. This group also includes 16-bit addition and subtraction between 16-bit CPU registers.

The bit manipulation instructions allow any bit in the accumulator, any general purpose register or any external memory location to be set, reset or tested with a single instruction. For example, the most significant bit of register H can be reset. This group is especially useful in control applications and for controlling software flags in general purpose programming.

The jump, call and return instructions are used to transfer between various locations in the user's program. This group uses several different techniques for obtaining the new program counter address from specific external memory locations. A unique type of jump is the restart instruction. This instruction actually contains the new address as a part of the 8-bit OP code. This is possible since only 8 separate addresses located in page zero of the external memory may be specified. Program jumps may also be achieved by loading register HL, IX or IY directly into the PC, thus allowing the jump address to be a complex function of the routine being executed.

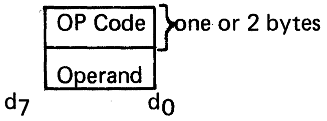
The input/output group of instructions in the Z80 allow for a wide range of transfers between external memory locations or the general purpose CPU registers, and the external I/O devices. In each case, the port number is provided on the lower 8 bits of the address bus during any I/O transaction. One instruction allows this port number to be specified by the second byte of the instruction while other Z80 instructions allow it to be specified as the content of the C register. One major advantage of using the C register as a pointer to the I/O device is that it allows different I/O ports to share common software driver routines. This is not possible when the address is part of the OP code if the routines are stored in ROM. Another feature of these input instructions is that they set the flag register automatically so that additional operations are not required to determine the state of the input data (for example its parity). The Z80-CPU includes single instructions that can move blocks or data (up to 256 bytes) automatically to or from any I/O port directly to any memory location. In conjunction with the dual set of general purpose registers, these instructions provide for fast I/O block transfer rates. The value of this I/O instruction set is demonstrated by the fact that the Z80-CPU can provide all required floppy disk formatting (i.e., the CPU provides the preamble, address, data and enables the CRC codes) on double density floppy disk drives on an interrupt driven basis.

Finally, the basic CPU control instructions allow various options and modes. This group includes instructions such as setting or resetting the interrupt enable flip flop or setting the mode of interrupt response.

5.2 ADDRESSING MODES

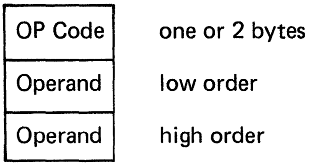
Most of the Z80 instructions operate on data stored in internal CPU registers, external memory or in the I/O ports. Addressing refers to how the address of this data is generated in each instruction. This section gives a brief summary of the types of addressing used in the Z80 while subsequent sections detail the type of addressing available for each instruction group.

Immediate. In this mode of addressing the byte following the OP code in memory contains the actual operand.



Examples of this type of instruction would be to load the accumulator with a constant, where the constant is the byte immediately following the OP code.

Immediate Extended. This mode is merely an extension of immediate addressing in that the two bytes following the op codes are the operand.



Examples of this type of instruction would be to load the HL register pair (16-bit register) with 16 bits (2 bytes) of data.

Modified Page Zero Addressing. The Z80 has a special single byte call instruction to any of 8 locations in page zero of memory. This instruction (which is referred to as a restart) sets the PC to an effective address in page zero. The value of this instruction is that it allows a single byte to specify a complete 16-bit address where commonly called subroutines are located, thus saving memory space.

OP Code	one byte

b_7 b_0 Effective address is $(00b_5b_4b_3000)$

Relative Addressing. Relative addressing uses one byte of data following the OP code to specify a displacement from the existing program to which a program jump can occur. This displacement is a signed two's complement number that is added to the address of the OP code of the following instruction.

OP Code	Jump relative (one byte OP code)
Operand	8-bit two's complement displacement added to Address (A+2)

The value of relative addressing is that it allows jumps to nearby locations while only requiring two bytes of memory space. For most programs, relative jumps are by far the most prevalent type of jump due to the proximity of related program segments. Thus, these instructions can significantly reduce memory space requirements. The signed displacement can range between +127 and -128 from A + 2. This allows for a total displacement of +129 to -126 from the jump relative OP code address. Another major advantage is that it allows for relocatable code.

Extended Addressing. Extended Addressing provides for two bytes (16 bits) of address to be included in the instruction. This data can be an address to which a program can jump or it can be an address where an operand is located.

OP Code	one or two bytes
Low Order Address or Low order operand	
High Order Address or High order operand	

Extended addressing is required for a program to jump from any location in memory to any other location, or load and store data in any memory location.

When extended addressing is used to specify the source or destination address of an operand, the notation (nn) will be used to indicate the content of memory at nn, where nn is the 16-bit address specified in the instruction. This means that the two bytes of address nn are used as a pointer to a memory location. The use of the parentheses always means that the value enclosed within them is used as a pointer to a memory location. For example, (1200) refers to the contents of memory at location 1200.

Indexed Addressing. In this type of addressing, the byte of data following the OP code contains a displacement which is added to one of the two index registers (the OP code specifies which index register is used) to form a pointer to memory. The contents of the index register are not altered by this operation.

OP Code	two byte OP code
OP Code	
Displacement	Operand added to index register to form a pointer to memory.

An example of an indexed instruction would be to load the contents of the memory location (Index Register + Displacement) into the accumulator. The displacement is a signed two's complement number. Indexed addressing greatly simplifies programs using tables of data since the index register can point to the start of any table. Two index registers are provided since very often operations require two or more tables. Indexed addressing also allows for relocatable code.

The two index registers in the Z80 are referred to as IX and IY. To indicate indexed addressing the notation:

(IX+d) or (IY+d)

is used. here d is the displacement specified after the OP code. The parentheses indicate that this value is used as a pointer to external memory.

Register Addressing. Many of the Z80 OP codes contain bits of information that specify which CPU register is to be used for an operation. An example of register addressing would be to load the data in register B into register C.

Implied Addressing. Implied addressing refers to operations where the OP code automatically implies one or more CPU registers as containing the operands. An example is the set of arithmetic operations where the accumulator is always implied to be the destination of the results.

Register Indirect Addressing. This type of addressing specifies a 16-bit CPU register pair (such as HL) to be used as a pointer to any location in memory. This type of instruction is very powerful and it is used in a wide range of applications.

OP Code
one or two bytes

An example of this type of instruction would be to load the accumulator with the data in the memory location pointed to by the HL register contents. Indexed addressing is actually a form of register indirect addressing except that a displacement is added with indexed addressing. Register indirect addressing allows for very powerful but simple to implement memory accesses. The block move and search commands in the Z80 are extensions of this type of addressing where automatic register incrementing, decrementing and comparing has been added. The notation for indicating register indirect addressing is to put parentheses around the name of the register that is to be used as the pointer. For example, the symbol

(HL)

specifies that the contents of the HL register are to be used as a pointer to a memory location. Often register indirect addressing is used to specify 16-bit operands. In this case, the register contents point to the lower order portion of the operand while the register contents are automatically incremented to obtain the upper portion of the operand.

Bit Addressing. The Z80 contains a large number of bit set, reset and test instructions. These instructions allow any memory location or CPU register to be specified for a bit operation through one of three previous addressing modes (register, register indirect and indexed) while three bits in the OP code specify which of the eight bits is to be manipulated.

ADDRESSING MODE COMBINATIONS

Many instructions include more than one operand (such as arithmetic instructions or loads). In these cases, two types of addressing may be employed. For example, load can use immediate addressing to specify the source and register indirect or indexed addressing to specify the source and register indirect or indexed addressing to specify the destination.

5.3 INSTRUCTION OP CODES

This section describes each of the Z80 instructions and provides tables listing the OP codes for every instruction. In each of these tables the shaded OP codes are identical to those offered in the 8080A CPU. Also shown is the assembly language mnemonic that is used for each instruction. All instruction OP codes are listed in hexadecimal notation. Single byte OP codes require two hex characters while double byte OP codes require four hex characters. The conversion from hex to binary is repeated here for convenience.

Hex		Binary		Decimal	Hex		Binary		Decimal
0	=	0000	=	0	8	=	1000	=	8
1	=	0001	=	1	9	=	1001	=	9
2	=	0010	=	2	A	=	1010	=	10
3	=	0011	=	3	B	=	1011	=	11
4	=	0100	=	4	C	=	1100	=	12
5	=	0101	=	5	D	=	1101	=	13
6	=	0110	=	6	E	=	1110	=	14
7	=	0111	=	7	F	=	1111	=	15

Z80 instruction mnemonics consist of an OP code and zero, one or two operands. Instructions in which the operand is implied have no operand. Instructions which have only one logical operand or those in which one operand is invariant (such as the Logical OR instruction) are represented by a one operand mnemonic. Instructions which may have two varying operands are represented by two operand mnemonics.

LOAD AND EXCHANGE

Table 5.3-1 defines the OP code for all of the 8-bit load instructions implemented in the Z80-CPU. Also shown in this table is the type of addressing used for each instruction. The source of the data is found on the top horizontal row while the destination is specified by the left hand column. For example, load register C from register B uses the OP code 48H. In all of the tables the OP code is specified in hexadecimal notation and the 48H (=0100 1000 binary) code is fetched by the CPU from the external memory during M1 time, decoded and then the register transfer is automatically performed by the CPU.

The assembly language mnemonic for this entire group is LD, followed by the destination followed by the source (LD DEST., SOURCE). Note that several combinations of addressing modes are possible. For example, the source may use register addressing and the destination may be register indirect, such as load the memory location pointed to by register HL with the contents of register D. The OP code for this operation would be 72. The mnemonic for this load instruction would be as follows: LD (HL), D

The parentheses around the HL means that the contents of HL are used as a pointer to a memory location. In all Z80 load instruction mnemonics the destination is always listed first, with the source following. The Z80 assembly language has been defined for ease of programming. Every instruction is self documenting and programs written in Z80 language are easy to maintain.

Note in Table 5.3-1 that some load OP codes that are available in the Z80 use two bytes. This is an efficient method of memory utilization since 8, 16, 24 or 32 bit instructions are implemented in the Z80. Thus often utilized instructions such as arithmetic or logical operations are only 8-bits which results in better memory utilization than is achieved with fixed instruction sizes such as 16-bits.

All load instructions using indexed addressing for either the source or destination location actually use three bytes of memory with the third byte being the displacement d. For example a load register E with the operand pointed to by IX with an offset of +8 would be written: LD E, (IX + 8)

The instruction sequence for this in memory would be:

Address A	DD	OP Code
A+1	5F	
A+2	08	Displacement operand

The two extended addressing instructions are also three byte instructions. For example the instruction to load the accumulator with the operand in memory location 6F32H would be written:

LD A, (6F 32H)

and its instruction sequence would be:

Address A	3A	OP Code
A+1	32	low order address
A+2	6F	high order address

Notice that the low order portion of the address is always the first operand.

The load immediate instructions for the general purpose 8-bit registers are two-byte instructions. The instruction load register H with the value 36H would be written:

LD H, 36H

and its sequence would be:

Address A	26	OP Code
A+1	36	Operand

Loading a memory location using indexed addressing for the destination and immediate addressing for the source requires four bytes. For example:

LD (IX - 15), 21H

would appear as:

Address A	DD	OP Code
A+1	36	
A+2	F1	displacement (-15 in signed two's complement)
A+3	21	operand to load

Notice that with any indexed addressing the displacement always follows directly after the OP code.

Table 5.3-2 specifies the 16-bit load operations. This table is very similar to the previous one. Notice that the extended addressing capability covers all register pairs. Also notice that register indirect operations specifying the stack pointer are the PUSH and POP instructions. The mnemonic for these instructions is "PUSH" and "POP". These differ from other 16-bit loads in that the stack pointer is automatically decremented and incremented as each byte is pushed onto or popped from the stack respectively. For example the instruction:

PUSH AF

is a single byte instruction with the OP code of F5H. When this instruction is executed the following sequence is generated:

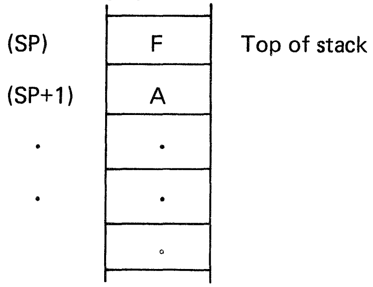
Decrement SP

LD (SP), A

Decrement SP

LD (SP), F

Thus the external stack now appears as follows:



8 BIT LOAD GROUP

		SOURCE															
		IMPLIED		REGISTER							REG INDIRECT			INDEXED		EXT. ADDR.	IMME.
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX + d)	(IY + d)	(nn)	n
REGISTER	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
	B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
	C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
	D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
	E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
	H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
	L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
DESTINATION REG INDIRECT	(HL)			77	70	71	72	73	74	75							36 n
	(BC)			02													
	(DE)			12													
INDEXED	(IX+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
EXT. ADDR	(nn)			32 n n													
IMPLIED	I			ED 47													
	R			ED 4F													

TABLE 5.3-1

The POP instruction is the exact reverse of a PUSH. Notice that all PUSH and POP instructions utilize a 16-bit operand and the high order byte is always pushed first and popped last. That is a:

PUSH BC is PUSH B then C
PUSH DE is PUSH D then E
PUSH HL is PUSH H then L
POP HL is POP L then H

The instruction using extended immediate addressing for the source obviously requires 2 bytes of data following the OP code. For example:

LD DE, 0659H

will be:

Address A	11	OP Code
A+1	59	Low order operand to register E
A+2	06	High order operand to register D

In all extended immediate or extended addressing modes, the low order byte always appears first after the OP code.

Table 5.3-3 lists the 16-bit exchange instructions implemented in the Z80. OP code 08H allows the programmer to switch between the two pairs of accumulator flag registers while D9H allows the programmer to switch between the duplicate set of six general purpose registers. These OP codes are only one byte in length to absolutely minimize the time necessary to perform the exchange so that the duplicate banks can be used to effect very fast interrupt response times.

BLOCK TRANSFER AND SEARCH

Table 5.3-4 lists the extremely powerful block transfer instructions. All of these instructions operate with three registers.

HL points to the source location.
DE points to the destination location.
BC is a byte counter.

After the programmer has initialized these three registers, any of these four instructions may be used. The LDI (Load and Increment) instruction moves one byte from the location pointed to by HL to the location pointed to by DE. Register pairs HL and DE are then automatically incremented and are ready to point to the following locations. The byte counter (register pair BC) is also decremented at this time. This instruction is valuable when blocks of data must be moved but other types of processing are required between each move. The LDIR (Load, increment and repeat) instruction is an extension of the LDI instruction. The same load and increment operation is repeated until the byte counter reaches the count of zero. Thus, this single instruction can move any block of data from one location to any other.

Note that since 16-bit registers are used, the size of the block can be up to 64K bytes (1K = 1024) long and it can be moved from any location in memory to any other location. Furthermore the blocks can be overlapping since there are absolutely no constraints on the data that is used in the three register pair.

The LDD and LDDR instructions are very similar to the LDI and LDIR. The only difference is that register pairs HL and DE are decremented after every move so that a block transfer starts from the highest address of the designated block rather than the lowest.

16 BIT LOAD GROUP 'LD' 'PUSH' AND 'POP'

		SOURCE									
		REGISTER							IMM. EXT.	EXT. ADDR.	REG. INDIR.
		AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
DESTINATION REGISTER	AF										F1
	BC								01 n n	ED 4B n n	C1
	DE								11 n n	ED 5B n n	D1
	HL								21 n n	2A n n	E1
	SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
	IX								DD 21 n n	DD 2A n n	DD E1
	IY								FD 21 n n	FD 2A n n	FD E1
EXT. ADDR.	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH INSTRUCTIONS → REG. IND.	(SP)	F5	C5	D5	E5		DD E5	FD E5			

POP INSTRUCTIONS ↑ (SP)

NOTE: The Push & Pop Instructions adjust the SP after every execution

TABLE 5.3-2

EXCHANGES 'EX' AND 'EXX'

		IMPLIED ADDRESSING				
		AF'	BC', DE' & HL'	HL	IX	IY
IMPLIED	AF	08				
	BC, DE & HL		D9			
	DE			EB		
REG. INDIR.	(SP)			E3	DD E3	FD E3

TABLE 5.3-3

BLOCK TRANSFER GROUP

			SOURCE	
			REG. INDIR.	
			(HL)	
DESTINATION	REG. INDIR.	(DE)	ED A0	'LDI' – Load (DE)←(HL) Inc HL & DE, Dec BC
			ED B0	'LDIR,' – Load (DE)←(HL) Inc HL & DE, Dec BC, Repeat until BC = 0
			ED A8	'LDD' – Load (DE)←(HL) Dec HL & DE, Dec BC
			ED B8	'LDDR' – Load (DE)←(HL) Dec HL & DE, Dec BC, Repeat until BC = 0

Reg	HL	points to source
Reg	DE	points to destination
Reg	BC	is byte counter

Table 5.3-4

Table 5.3-5 specifies the OP codes for the four block search instructions. The first, CPI (compare and increment) compares the data in the accumulator, with the contents of the memory location pointed to by register HL. The result of the compare is stored in one of the flag bits (see section 6.0 for a detailed explanation of the flag operations) and the HL register pair is then incremented and the byte counter (register pair BC) is decremented.

The instruction CPIR is merely an extension of the CPI instruction in which the compare is repeated until either a match is found or the byte counter (register pair BC) becomes zero. Thus, this single instruction can search the entire memory for any 8-bit character.

The CPD (Compare and Decrement) and CPDR (Compare, Decrement and Repeat) are similar instructions, their only difference being that they decrement HL after every compare so that they search the memory in the opposite direction. (The search is started at the highest location in the memory block).

It should be emphasized again that these block transfer and compare instructions are extremely powerful in string manipulation applications.

ARITHMETIC AND LOGICAL

Table 5.3-6 lists all of the 8-bit arithmetic operations that can be performed with the accumulator, also listed are the increment (INC) and decrement (DEC) instructions. In all of these instructions, except INC and DEC, the specified 8-bit operation is performed between the data in the accumulator and the source data specified in the table. The result of the operation is placed in the accumulator with the exception of compare (CP) that leaves the accumulator unaffected. All of these operations affect the flag register as a result of the specified operation. (Section 6.0 provides all of the details on how the flags are affected by any instruction type). INC and DEC instructions specify a register or a memory location as both source and destination of the result. When the source operand is addressed using the index registers the displacement must follow directly. With immediate addressing the actual operand will follow directly. for example the instruction:

AND 07H

would appear as:

Address A	E6	OP Code
A+1	07	Operand

BLOCK SEARCH GROUP

SEARCH LOCATION	
REG. INDIR.	
(HL)	
ED A1	'CPI' Inc HL, Dec BC
ED B1	'CPIR', Inc HL, Dec BC repeat until BC = 0 or find match
ED A9	'CPD' Dec HL & BC
ED B9	'CPDR' Dec HL & BC Repeat until BC = 0 or find match

HL points to location in memory to be compared with accumulator contents
BC is byte counter

TABLE 5.3-5

Assuming that the accumulator contained the value F3H the result of 03H would be placed in the accumulator:

Acc before operation	1111 0011 = F3H
Operand	0000 0111 = 07H
Result to Acc	0000 0011 = 03H

The Add instruction (ADD) performs a binary add between the data in the source location and the data in the accumulator. The subtract (SUB) does a binary subtraction. When the add with carry is specified (ADC) or the subtract with carry (SBC), then the carry flag is also added or subtracted respectively. The flags and decimal adjust instruction (DAA) in the Z80 (fully described in section 6.0) allow arithmetic operations for:

multiprecision packed BCD numbers

multiprecision signed or unsigned binary numbers

multiprecision two's complement signed numbers

Other instructions in this group are logical and (AND), logical or (OR), exclusive or (XOR) and compare (CP).

There are five general purpose arithmetic instructions that operate on the accumulator or carry flag. These five are listed in Table 5.3-7. The decimal adjust instruction can adjust for subtraction as well as addition, thus making BCD arithmetic operations simple. Note that to allow for this operation the flag N is used. This flag is set if the last arithmetic operation was a subtract. The negate accumulator (NEG) instruction forms the two's complement of the number in the accumulator. Finally notice that a reset carry instruction is not included in the Z80 since this operation can be easily achieved through other instructions such as a logical AND of the accumulator with itself.

Table 5.3-8 lists all of the 16-bit arithmetic operations between 16-bit registers. There are five groups of instructions including add with carry and subtract with carry. ADC and SBC affect all of the flags. These two groups simplify address calculation operations or other 16-bit arithmetic operations.

8 BIT ARITHMETIC AND LOGIC

SOURCE

	REGISTER ADDRESSING							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

TABLE 5.3-6

GENERAL PURPOSE AF OPERATIONS

Decimal Adjust Acc, 'DAA'	27
Complement Acc, 'CPL'	2F
Negate Acc, 'NEG' (2's complement)	ED 44
Complement Carry Flag, 'CCF'	3F
Set Carry Flag, 'SCF'	37

TABLE 5.3-7

16 BIT ARITHMETIC

			SOURCE					
			BC	DE	HL	SP	IX	IY
DESTINATION	'ADD'	HL	09	19	29	39		
		IX	DD 09	DD 19		DD 39	DD 29	
		IY	FD 09	FD 19		FD 39		FD 29
	ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
	SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
	INCREMENT 'INC.		03	13	23	33	DD 23	FD 23
	DECREMENT 'DEC'		0B	1B	2B	3B	DD 2B	FD 2B

TABLE 5.3-8

ROTATE AND SHIFT

A major capability of the Z80 is its ability to rotate or shift data in the accumulator, any general purpose register, or any memory location. All of the rotate and shift OP codes are shown in Table 5.3-9. Also included in the Z80 are arithmetic and logical shift operations. These operations are useful in an extremely wide range of applications including integer multiplication and division. Two BCD digit rotate instructions (RRD and RLD) allow a digit in the accumulator to be rotated with the two digits in a memory location pointed to by register pair HL. (See Figure 5.3-9). These instructions allow for efficient BCD arithmetic.

BIT MANIPULATION

The ability to set, reset and test individual bits in a register or memory location is needed in almost every program. These bits may be flags in a general purpose software routine, indications of external control conditions or data packed into memory locations to make memory utilization more efficient.

The Z80 has the ability to set, reset or test any bit in the accumulator, any general purpose register or any memory location with a single instruction. Table 5.3-10 lists the 240 instructions that are available for this purpose. Register addressing can specify the accumulator or any general purpose register on which the operation is to be performed. Register indirect and indexed addressing are available to operate on external memory locations. Bit test operations set the zero flag (Z) if the tested bit is a zero. (Refer to section 6.0 for further explanation of flag operation).

JUMP, CALL AND RETURN

Figure 5.3-11 lists all of the jump, call and return instructions implemented in the Z80 CPU. A jump is a branch in a program where the program counter is loaded with the 16-bit value as specified by one of the three available addressing modes (Immediate Extended, Relative or Register Indirect). Notice that the jump group has several different conditions that can be specified to be met before the jump will be made. If these conditions are not met, the program merely continues with the next sequential instruction. The conditions are all dependent on the data in the flag register. (Refer to section 6.0 for details on the flag register). The immediate extended addressing is used to jump to any location in the memory. This instruction requires three bytes (two to specify the 16-bit address) with the low order address byte first followed by the high order address byte.

ROTATES AND SHIFTS

		Source and Destination									
		A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)
TYPE OF ROTATE OR SHIFT	'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
	'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
	'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
	'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
	'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
	'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
	'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
	'RLD'								ED 6F		
	'RRD'								ED 67		

	A
RLCA	07
RRCA	0F
RLA	17
RRA	1F

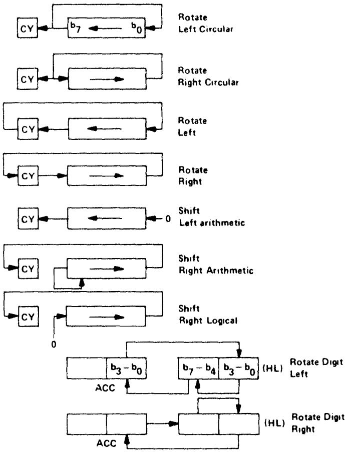


TABLE 5.3-9

For example an unconditional Jump to memory location 3E32H would be:

Address A	C3	OP Code
A+1	32	Low order address
A+2	3E	High order address

The relative jump instruction uses only two bytes, the second byte is a signed two's complement displacement from the existing PC. This displacement can be in the range of +129 to -126 and is measured from the address of the instruction OP code.

Three types of register indirect jumps are also included. These instructions are implemented by loading the register pair HL or one of the index registers IX or IY directly into the PC. This capability allows for program jumps to be a function of previous calculations.

A call is a special form of a jump where the address of the byte following the call instruction is pushed onto the stack before the jump is made. A return instruction is the reverse of a call because the data on the top of the stack is popped directly into the PC to form a jump address. The call and return instructions allow for simple subroutine and interrupt handling. Two special return instructions have been included in the Z80 family of components. The return from interrupt instruction (RETI) and the return from non-maskable interrupt (RETN) are treated in the CPU as an unconditional return identical to the OP code C9H. The difference is that (RETI) can be used at the end of an interrupt routine and all Z80 peripheral chips will recognize the execution of this instruction for proper control of nested priority interrupt handling. This instruction coupled with the Z80 peripheral devices implementation simplifies the normal return from nested interrupt. Without this feature the following software sequence would be necessary to inform the interrupting device that the interrupt routine is completed:

BIT MANIPULATION GROUP

		REGISTER ADDRESSING							REG. INDIR.	INDEXED	
	BIT	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
TEST 'BIT'	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	GB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RESET BIT 'RES'	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET BIT 'SET'	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

TABLE 5.3-10

Disable Interrupt	– prevent interrupt before routine is exited.
LD A, n OUT n, A	– notify peripheral that service routine is complete
Enable Interrupt	
Return	

This seven byte sequence can be replaced with the three byte EI RETI instruction sequence in the Z80. This is important since interrupt service time often must be minimized.

To facilitate program loop control the instruction DJNZ e can be used advantageously. This two byte, relative jump instruction decrements the B register and the jump occurs if the B register has not been decremented to zero. The relative displacement is expressed as a signed two's complement number. A simple example of its use might be:

Address	Instruction	Comments
N, N+1	LD B, 7	; set B register to count of 7
N + 2 to N + 9	(Perform a sequence of instructions)	; loop to be performed 7 times
N + 10, N + 11	DJNZ -10	; to jump from N + 12 to N + 2
N + 12	(Next Instruction)	

JUMP, CALL AND RETURN GROUP

			CONDITION									
			UN-COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG	SIGN POS	REG B≠0
JUMP 'JP'	IMMED. EXT.	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JR'	RELATIVE	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP 'JP'	REG. INDIR.	(HL)	E9									
JUMP 'JP'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
'CALL'	IMMED. EXT.	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DECREMENT B, JUMP IF NON ZERO 'DJNZ'	RELATIVE	PC+e										10 e-2
RETURN 'RET'	REGISTER INDIR.	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETURN FROM INT 'RETI'	REG. INDIR.	(SP) (SP+1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	REG. INDIR.	(SP) (SP+1)	ED 45									

TABLE 5.3-11

NOTE–CERTAIN FLAGS HAVE MORE THAN ONE PURPOSE. REFER TO SECTION 6.0 FOR DETAILS

Table 5.3-12 lists the eight OP codes for the restart instruction. This instruction is a single byte call to any of the eight addresses listed. The simple mnemonic for these eight calls is also shown. The value of this instruction is that frequently used routines can be called with this instruction to minimize memory usage.

RESTART GROUP

CALL ADDRESS	OP CODE	
0000_H	C7	'RST 0'
0008_H	CF	'RST 8'
0010_H	D7	'RST 16'
0018_H	DF	'RST 24'
0020_H	E7	'RST 32'
0028_H	EF	'RST 40'
0030_H	F7	'RST 48'
0038_H	FF	'RST 56'

TABLE 5.3-12

INPUT/OUTPUT

The Z80 has an extensive set of Input and Output instructions as shown in table 5.3-13 and table 5.3-14. The addressing of the input or output device can be either absolute or register indirect, using the C register. Notice that in the register indirect addressing mode data can be transferred between the I/O devices and any of the internal registers. In addition eight block transfer instructions have been implemented. These instructions are similar to the memory block transfers except that they use register pair HL for a pointer to the memory source (output commands) or destination (input commands) while register B is used as a byte counter. Register C holds the address of the port for which the input or output command is desired. Since register B is eight bits in length, the I/O block transfer command handles up to 256 bytes.

In the instructions IN A, n and OUT n, A an I/O device address n appears in the lower half of the address bus (A_0-A_7) while the accumulator content is transferred in the upper half of the address bus. In all register indirect input output instructions, including block I/O transfers the content of register C is transferred to the lower half of the address bus (device address) while the content of register B is transferred to the upper half of the address bus.

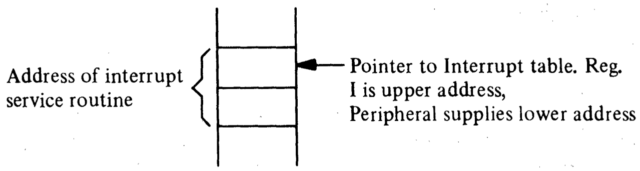
INPUT GROUP

				PORT ADDRESS		
				IMMED.	REG. INDIR.	
				n	(C)	
INPUT DESTINATION	INPUT 'IN'	REG ADDRESSING	A	DB	ED 78	
			B		ED 40	
			C		ED 48	
			D		ED 50	
			E		ED 58	
			H		ED 60	
			L		ED 68	
	'INI' – INPUT & Inc HL, Dec B	REG. INDIR	(HL)		ED A2	BLOCK INPUT COMMANDS
	'INIR'– INP, Inc HL, Dec B, REPEAT IF B≠0				ED B2	
	'IND'– INPUT & Dec HL, Dec B				ED AA	
	'INDR'– INPUT, Dec HL, Dec B, REPEAT IF B≠0				ED BA	

TABLE 5.3-13

CPU CONTROL GROUP

The final table, table 5.3-15 illustrates the six general purpose CPU control instructions. The NOP is a do-nothing instruction. The HALT instruction suspends CPU operation until a subsequent interrupt is received, while the DI and EI are used to lock out and enable interrupts. The three interrupt mode commands set the CPU into any of the three available interrupt response modes as follows. If mode zero is set the interrupting device can insert any instruction on the data bus and allow the CPU to execute it. Mode 1 is a simplified mode where the CPU automatically executes a restart (RST) to location 0038H so that no external hardware is required. (The old PC content is pushed onto the stack). Mode 2 is the most powerful in that it allows for an indirect call to any location in memory. With this mode the CPU forms a 16-bit memory address where the upper 8-bits are the content of register I and the lower 8-bits are supplied by the interrupting device. This address points to the first of two sequential bytes in a table where the address of the service routine is located. The CPU automatically obtains the starting address and performs a CALL to this address.


Address of interrupt service routine
Pointer to Interrupt table. Reg. I is upper address,
Peripheral supplies lower address

OUTPUT GROUP

			SOURCE							
			REGISTER							REG. IND.
			A	B	C	D	E	H	L	(HL)
'OUT'	IMMED.	n	D3 n							
	REG. IND.	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69	
'OUTI' – OUTPUT Inc HL, Dec b	REG. IND.	(C)								ED A3
'OTIR' – OUTPUT, Inc HL, Dec B, REPEAT IF B≠0	REG. IND.	(C)								ED B3
'OUTD' – OUTPUT Dec HL & B	REG. IND.	(C)								ED AB
'OTDR' – OUTPUT, Dec HL & B, REPEAT IF B≠0	REG. IND.	(C)								ED BB

BLOCK OUTPUT COMMANDS

PORT DESTINATION ADDRESS

TABLE 5.3-14

MISCELLANEOUS CPU CONTROL

'NOP'	00	
'HALT'	76	
DISABLE INT '(DI)'	F3	
ENABLE INT '(EI)'	FB	
SET INT MODE 0 'IM0'	ED 46	8080A MODE
SET INT MODE 1 'IM1'	ED 56	CALL TO LOCATION 0038_H
SET INT MODE 2 'IM2'	ED 5E	INDIRECT CALL USING REGISTER I AND 8 BITS FROM INTERRUPTING DEVICE AS A POINTER.

TABLE 5.3-15

Z80 Family

6.0 FLAGS

Each of the two Z80-CPU Flag registers contains six bits of information which are set or reset by various CPU operations. Four of these bits are testable; that is, they are used as conditions for jump, call or return instructions. For example a jump may be desired only if a specific bit in the flag register is set. The four testable flag bits are:

1) Carry Flag (C) – This flag is the carry from the highest order bit of the accumulator. For example, the carry flag will be set during an add instruction where a carry from the highest bit of the accumulator is generated. This flag is also set if a borrow is generated during a subtraction instruction. The shift and rotate instructions also affect this bit.

2) Zero Flag (Z) – This flag is set if the result of the operation loaded a zero into the accumulator. Otherwise it is reset.

3) Sign Flag(S) – This flag is intended to be used with signed numbers and it is set if the result of the operation was negative. Since bit 7 (MSB) represents the sign of the number (A negative number has a 1 in bit 7), this flag stores the state of bit 7 in the accumulator.

4) Parity/Overflow Flag(P/V) – This dual purpose flag indicates the parity of the result in the accumulator when logical operations are performed (such as AND A, B) and it represents overflow when signed two's complement arithmetic operations are performed. The Z80 overflow flag indicates that the two's complement number in the accumulator is in error since it has exceeded the maximum possible (+127) or is less than the minimum possible (–128) number that can be represented two's complement notation. For example consider adding:

```
+120 =   0111 1000
+105 =   0110 1001
  C = 0  1110 0001 = -95 (wrong) Overflow has occurred;
```

Here the result is incorrect. Overflow has occurred and yet there is no carry to indicate an error. For this case the overflow flag would be set. Also consider the addition of two negative numbers:

```
-5  =    1111 1011
-16 =    1111 0000
 C = 1   1110 1011 = -21 correct
```

Notice that the answer is correct but the carry is set so that this flag can not be used as an overflow indicator. In this case the overflow would not be set.

For logical operations (AND, OR, XOR) this flag is set if the parity of the result is even and it is reset if it is odd.

There are also two non-testable bits in the flag register. Both of these are used for BCD arithmetic. They are:

1) Half carry(H) – This is the BCD carry or borrow result from the least significant four bits of operation. When using the DAA (Decimal Adjust Instruction) this flag is used to correct the result of a previous packed decimal add or subtract.

2) Add/Subtract Flag (N) – Since the agorithim for correcting BCD operations is different for addition or subtraction, this flag is used to specify what type of instruction was executed last so that the DAA operation will be correct for either addition or subtraction.

The Flag register can be accessed by the programmer and its format is as follows:

D7							DØ
S	Z	X	H	X	P/V	N	C

X means flag is indeterminate.

Table 6.0-1 lists how each flag bit is affected by various CPU instructions. In this table a '•' indicates that the instruction does not change the flag, an 'X' means that the flag goes to an indeterminate state, an '0' means that it is reset, a '1' means that it is set and the symbol ↕ indicates that it is set or reset according to the previous discussion. Note that any instruction not appearing in this table does not affect any of the flags.

Table 6.0-1 includes a few special cases that must be described for clarity. Notice that the block search instruction sets the Z flag if the last compare operation indicated a match between the source and the accumulator data. Also, the parity flag is set if the byte counter (register pair BC) is not equal to zero. This same use of the parity flag is made with the block move instructions. Another special case is during block input or output instructions, here the Z flag is used to indicate the state of register B which is used as a byte counter. Notice that when the I/O block transfer is complete, the zero flag will be reset to a zero (i.e. B=0) while in the case of a block move command the parity flag is reset when the operation is complete. A final case is when the refresh or I register is loaded into the accumulator, the interrupt enable flip flop is loaded into the parity flag so that the complete state of the CPU can be saved at any time.

SUMMARY OF FLAG OPERATION

	D7							D0	
						P/			
Instruction	S	Z		H		V	N	C	Comments
ADD A,s; ADC A,s	↕	↕	X	↕	X	V	0	↕	8-bit add or add with carry
SUB,s; SBCA,s; CP,s; NEG	↕	↕	X	↕	X	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	↕	↕	X	1	X	P	0	0	Logical operations
OR s; XOR s	↕	↕	X	0	X	P	0	0	
INC s	↕	↕	X	↕	X	V	0	•	8-bit increment
DEC s	↕	↕	X	↕	X	V	1	•	8-bit decrement
ADD DD, SS	•	•	X	X	X	•	0	↕	16-bit add
ADC HL, SS	↕	↕	X	X	X	V	0	↕	16-bit add with carry
SBC HL, SS	↕	↕	X	X	X	V	1	↕	16-bit subtract with carry
RLA; RLCA; RRA; RRCA	•	•	X	0	X	•	0	↕	Rotate accumulator
RL s; RLC s; RR s; RRC s; SLA s; SRA s; SRL s	↕	↕	X	0	X	P	0	↕	Rotate and shift locations
RLD; RRD	↕	↕	X	0	X	P	0	•	Rotate digit left and right
DAA	↕	↕	X	↕	X	P	•	↕	Decimal adjust accumulator
CPL	•	•	X	1	X	•	1	•	Complement accumulator
SCF	•	•	X	0	X	•	0	1	Set carry
CCF	•	•	X	X	X	•	0	↕	Complement carry
IN r, (C)	↕	↕	X	0	X	P	0	•	Input register indirect
INI; IND; OUTI; OUTD	X	↕	X	X	X	X	1	X	Block input and output
INIR; INDR; OTIR; OTDR	X	1	X	X	X	X	1	X	Z = 0 if B ≠ 0 otherwise Z = 1
LDI; LDD	X	X	X	0	X	↕	0	•	Block transfer instructions
LDIR; LDDR	X	X	X	0	X	0	0	•	P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI; CPIR; CPD; CPDR	↕	↕	X	↕	X	↕	1	•	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	↕	↕	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	X	↕	X	1	X	X	0	•	The state of bit b of location s is copied into the Z flag

The following notation is used in this table:

SYMBOL	OPERATION
C	Carry/link flag. C=1 if the operation produced a carry from the MSB of the operand or result.
Z	Zero flag. Z=1 if the result of the operation is zero.
S	Sign flag. S=1 if the MSB of the result is one.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V=1 if the result of the operation is even, P/V=0 if result is odd. If P/V holds overflow, P/V=1 if the result of the operation produced an overflow.
H	Half-carry flag. H=1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.
N	Add/Subtract flag. N=1 if the previous operation was a subtract.
	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
	The flag is affected according to the result of the operation.
●	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care".
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range <0, 255>
nn	16-bit value in range <0, 65535>

TABLE 6.0-1

Z80
Family

7.0 SUMMARY OF OP CODES AND EXECUTION TIMES

The following section gives a summary of the Z80 instruction set. The instructions are logically arranged into groups as shown on Tables 7.0-1 through 7.0-11. Each table shows the assembly language mnemonic OP code, the actual OP code, the symbolic operation, the content of the flag register following the execution of each instruction, the number of bytes required for each instruction as well as the number of memory cycles and the total number of T states (external clock periods) required for the fetching and execution of each instruction. Care has been taken to make each table self-explanatory without requiring any cross reference with the text or other tables.

8-BIT LOAD GROUP

	Symbolic	Flags								Op-Code		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
LD r, s	r ← s	•	•	X	•	X	•	•	•	01 r s		1	1	4	r, s Reg.
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110		2	2	7	000 B
										← n →					001 C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010 D
LD r, (IX+d)	r ← (IX+d)	•	•	X	•	X	•	•	•	11 011 101	DD	3	5	19	011 E
										01 r 110					100 H
										← d →					101 L
LD r, (IY+d)	r ← (IY+d)	•	•	X	•	X	•	•	•	11 111 101	FD	3	5	19	111 A
										01 r 110					
										← d →					
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7	
LD (IX+d), r	(IX+d) ← r	•	•	X	•	X	•	•	•	11 011 101	DD	3	5	19	
										01 110 r					
										← d →					
LD (IY+d), r	(IY+d) ← r	•	•	X	•	X	•	•	•	11 111 101	FD	3	5	19	
										01 110 r					
										← d →					
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110	36	2	3	10	
										← n →					
LD (IX+d), n	(IX+d) ← n	•	•	X	•	X	•	•	•	11 011 101	DD	4	5	19	
										00 110 110	36				
										← d →					
										← n →					
LD (IY+d), n	(IY+d) ← n	•	•	X	•	X	•	•	•	11 111 101	FD	4	5	19	
										00 110 110	36				
										← d →					
										← n →					
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010	3A	3	4	13	
										← n →					
										← n →					
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010	32	3	4	13	
										← n →					
										← n →					
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101	ED	2	2	9	
										01 010 111	57				
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101	ED	2	2	9	
										01 011 111	5F				
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101	ED	2	2	9	
										01 000 111	47				
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101	ED	2	2	9	
										01 001 111	4F				

Notes: r, s means any of the registers A, B, C, D, E, H, L
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-1

16-BIT LOAD GROUP

	Symbolic	Flags								Op-Code		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00 dd0 001		3	3	10	dd Pair
										← n →					00 BC
										← n →					01 DE
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	11 011 101	DD	4	4	14	10 HL
										00 100 001	21				11 SP
										← n →					
										← n →					
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	11 111 101	FD	4	4	14	
										00 100 001	21				
										← n →					
										← n →					
LD HL, (nn)	H ← (nn+1)	•	•	X	•	X	•	•	•	00 101 010	2A	3	5	16	
	L ← (nn)									← n →					
										← n →					
LD dd, (nn)	dd_H ← (nn+1)	•	•	X	•	X	•	•	•	11 101 101	ED	4	6	20	
	dd_L ← (nn)									01 dd1 011					
										← n →					
										← n →					
LD IX, (nn)	IX_H ← (nn+1)	•	•	X	•	X	•	•	•	11 011 101	DD	4	6	20	
	IX_L ← (nn)									00 101 010	2A				
										← n →					
										← n →					
LD IY, (nn)	IY_H ← (nn+1)	•	•	X	•	X	•	•	•	11 111 101	FD	4	6	20	
	IY_L ← (nn)									00 101 010	2A				
										← n →					
										← n →					
LD (nn), HL	(nn+1) ← H	•	•	X	•	X	•	•	•	00 100 010	22	3	5	16	
	(nn) ← L									← n →					
										← n →					
LD (nn), dd	(nn+1) ← dd_H	•	•	X	•	X	•	•	•	11 101 101	ED	4	6	20	
	(nn) ← dd_L									01 dd0 011					
										← n →					
										← n →					
LD (nn), IX	(nn+1) ← IX_H	•	•	X	•	X	•	•	•	11 011 101	DD	4	6	20	
	(nn) ← IX_L									00 100 010	22				
										← n →					
										← n →					
LD (nn), IY	(nn+1) ← IY_H	•	•	X	•	X	•	•	•	11 111 101	FD	4	6	20	
	(nn) ← IY_L									00 100 010	22				
										← n →					
										← n →					
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	11 011 101	DD	2	2	10	
										11 111 001	F9				
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	11 111 101	FD	2	2	10	
										11 111 001	F9				qq Pair
PUSH qq	(SP-2) ← qq_L	•	•	X	•	X	•	•	•	11 qq0 101		1	3	11	00 BC
	(SP-1) ← qq_H														01 DE
PUSH IX	(SP-2) ← IX_L	•	•	X	•	X	•	•	•	11 011 101	DD	2	4	15	10 HL
	(SP-1) ← IX_H									11 100 101	E5				11 AF
PUSH IY	(SP-2) ← IY_L	•	•	X	•	X	•	•	•	11 111 101	FD	2	4	15	
	(SP-1) ← IY_H									11 100 101	E5				
POP qq	qq_H ← (SP+1)	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
	qq_L ← (SP)														
POP IX	IX_H ← (SP+1)	•	•	X	•	X	•	•	•	11 011 101	DD	2	4	14	
	IX_L ← (SP)									11 100 001	E1				
POP IY	IY_H ← (SP+1)	•	•	X	•	X	•	•	•	11 111 101	FD	2	4	14	
	IY_L ← (SP)									11 100 001	E1				

Notes: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively.
e.g. BC_L = C, AF_H = A

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ flag is affected according to the result of the operation.

Table 7.0-2

EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP

	Symbolic	Flags								Op-Code		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
EX DE, HL	DE↔HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF'	AF↔AF'	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	(BC↔BC'	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Register bank and
	DE↔DE'														auxiliary register
	HL↔HL')														bank exchange
EX (SP), HL	H ↔(SP+1)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
	L ↔(SP)														
EX (SP), IX	IX_H↔(SP+1)	•	•	X	•	X	•	•	•	11 011 101	DD	2	6	23	
	IX_L↔(SP)									11 100 011	E3				
EX (SP), IY	IY_H↔(SP+1)	•	•	X	•	X	•	•	•	11 111 101	FD	2	6	23	
	IY_L↔(SP)									11 100 011	E3				
LDI	(DE)←(HL)	•	•	X	0	X	① ↕	0	•	11 101 101	ED	2	4	16	Load (HL) into
	DE ← DE+1									10 100 000	A0				(DE), increment the
	HL ← HL+1														pointers and
	BC ← BC-1														decrement the byte
															counter (BC)
LDIR	(DE)←(HL)	•	•	X	0	X	0	0	•	11 101 101	ED	2	5	21	If BC ≠ 0
	DE ← DE+1									10 110 000	B0	2	4	16	If BC = 0
	HL ← HL+1														
	BC ← BC-1														
	Repeat until														
	BC = 0														
LDD	(DE)←(HL)	•	•	X	0	X	① ↕	0	•	11 101 101	ED	2	4	16	
	DE ← DE-1									10 101 000	A8				
	HL ← HL-1														
	BC ← BC-1														
LDDR	(DE)←(HL)	•	•	X	0	X	0	0	•	11 101 101	ED	2	5	21	If BC ≠ 0
	DE ← DE-1									10 111 000	B8	2	4	16	If BC = 0
	HL ← HL-1														
	BC ← BC-1														
	Repeat until														
	BC = 0														
CPI	A – (HL)	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101	ED	2	4	16	
	HL ← HL+1									10 100 001	A1				
	BC ← BC-1														
CPIR	A – (HL)	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101	ED	2	5	21	If BC ≠ 0 and A≠(HL)
	HL ← HL+1									10 110 001	B1	2	4	16	If BC = 0 or A = (HL)
	BC ← BC-1														
	Repeat until														
	A = (HL) or														
	BC = 0														
CPD	A – (HL)	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101	ED	2	4	16	
	HL ← HL-1									10 101 001	A9				
	BC ← BC-1														
CPDR	A – (HL)	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101	ED	2	5	21	If BC ≠ 0 and A ≠ (HL)
	HL ← HL-1									10 111 001	B9	2	4	16	If BC = 0 or A = (HL)
	BC ← BC-1														
	Repeat until														
	A = (HL) or														
	BC = 0														

Notes: ① P/V flag is 0 if the result of BC-1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-3

8-BIT ARITHMETIC AND LOGICAL GROUP

	Symbolic	Flags								Op-Code		No. of	No.of M	No.of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
ADD A, r	A ← A+r	↕	↕	X	↕	X	V	0	↕	10 $\boxed{000}$ r		1	1	4	r Reg.
ADD A, n	A ← A+n	↕	↕	X	↕	X	V	0	↕	11 $\boxed{000}$ 110		2	2	7	000 B
										← n →					001 C
															010 D
ADD A, (HL)	A ← A+(HL)	↕	↕	X	↕	X	V	0	↕	10 $\boxed{000}$ 110		1	2	7	011 E
ADD A, (IX+d)	A←A+(IX+d)	↕	↕	X	↕	X	V	0	↕	11 011 101	DD	3	5	19	100 H
										10 $\boxed{000}$ 110					101 L
										← d →					111 A
ADD A, (IY+d)	A←A+(IY+d)	↕	↕	X	↕	X	V	0	↕	11 111 101	FD	3	5	19	
										10 $\boxed{000}$ 110					
										← d →					
ADC A, s	A←A+s+CY	↕	↕	X	↕	X	V	0	↕	$\boxed{001}$					s is any of r, n,
SUB s	A←A - s	↕	↕	X	↕	X	V	1	↕	$\boxed{010}$					(HL), (IX+d),
SBC A, s	A←A - s - CY	↕	↕	X	↕	X	V	1	↕	$\boxed{011}$					(IY+d) as shown for
AND s	A←A ∧ s	↕	↕	X	1	X	P	0	0	$\boxed{100}$					ADD instruction.
OR s	A←A ∨ s	↕	↕	X	0	X	P	0	0	$\boxed{110}$					The indicated bits
XOR s	A←A ⊕ s	↕	↕	X	0	X	P	0	0	$\boxed{101}$					replace the $\boxed{000}$ in
CP s	A - s	↕	↕	X	↕	X	V	1	↕	$\boxed{111}$					the ADD set above.
INC r	r ← r+1	↕	↕	X	↕	X	V	0	●	00 r $\boxed{100}$		1	1	4	
INC (HL)	(HL)←(HL)+1	↕	↕	X	↕	X	V	0	●	00 110 $\boxed{100}$		1	3	11	
INC (IX+d)	(IX+d) ←	↕	↕	X	↕	X	V	0	●	11 011 101	DD	3	6	23	
	(IX+d)+1									00 110 $\boxed{100}$					
										← d →					
INC (IY+d)	(IY+d) ←	↕	↕	X	↕	X	V	0	●	11 111 101	FD	3	6	23	
	(IY+d)+1									00 110 $\boxed{100}$					
										← d →					
DEC s	s ← s - 1	↕	↕	X	↕	X	V	1	●	$\boxed{101}$					s is any of r, (HL), (IX+d), (IY+d) as shown for INC. DEC same format and states as INC. Replace $\boxed{100}$ with $\boxed{101}$ in OP Code.

Notes: The V symbol in the P/V flag column indicates that the P/V flag contains the overflow of the result of the operation. Similarly the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow, P = 1 means parity of the result is even, P = 0 means parity of the result is odd.

Flag Notation: ● = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Table 7.0-4

GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS

	Symbolic	Flags								Op-Code		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
DAA	Converts acc, content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	Decimal adjust accumulator
CPL	A ← $\overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Complement accumulator (One's complement)
NEG	A ← $\overline{A}$+1	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Negate acc, (two's complement)
CCF	CY ← $\overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Set carry flag
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI*	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI*	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

*Interrupts are not sampled at the end of EI or DI

Table 7.0-5

16-BIT ARITHMETIC GROUP

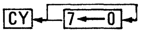
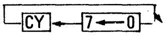
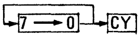
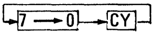
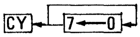
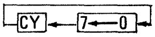
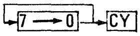
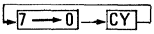
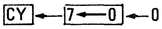
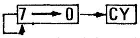
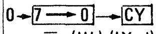
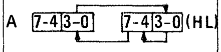
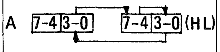
	Symbolic	Flags								Op-Code		No. of	No.of M	No.of T		
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments	
ADD HL, ss	HL ← HL+ss	●	●	X	X	X	●	0	↕	00 ss1 001		1	3	11	ss	Reg.
															00	BC
ADC HL, ss	HL ← HL+ss+CY	↕	↕	X	X	X	V	0	↕	11 101 101	ED	2	4	15	01	DE
										01 ss1 010					10	HL
															11	SP
SBC HL, ss	HL ← HL-ss-CY	↕	↕	X	X	X	V	1	↕	11 101 101	ED	2	4	15		
										01 ss0 010						
ADD IX, pp	IX ← IX + pp	●	●	X	X	X	●	0	↕	11 011 101	DD	2	4	15	pp	Reg.
										00 pp1 001					00	BC
															01	DE
															10	IX
															11	SP
ADD IY, rr	IY ← IY + rr	●	●	X	X	X	●	0	↕	11 111 101	FD	2	4	15	rr	Reg.
										00 rr1 001					00	BC
															01	DE
															10	IY
															11	SP
INC ss	ss ← ss + 1	●	●	X	●	X	●	●	●	00 ss0 011		1	1	6		
INC IX	IX ← IX + 1	●	●	X	●	X	●	●	●	11 011 101	DD	2	2	10		
										00 100 011	23					
INC IY	IY ← IY + 1	●	●	X	●	X	●	●	●	11 111 101	FD	2	2	10		
										00 100 011	23					
DEC ss	ss ← ss - 1	●	●	X	●	X	●	●	●	00 ss1 011		1	1	6		
DEC IX	IX ← IX - 1	●	●	X	●	X	●	●	●	11 011 101	DD	2	2	10		
										00 101 011	2B					
DEC IY	IY ← IY - 1	●	●	X	●	X	●	●	●	11 111 101	FD	2	2	10		
										00 101 011	2B					

Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: ● = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

Table 7.0-6

ROTATE AND SHIFT GROUP

Mnemonic	Symbolic Operation	Flags S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
RLCA		•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	Rotate left circular accumulator
RLA		•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	Rotate left accumulator
RRCA		•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	Rotate right circular accumulator
RRA		•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	Rotate right accumulator
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 00 $\boxed{000}$ r	CB	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 $\boxed{000}$ 110	CB	2	4	15	r Reg. 000 B
RLC (IX+d)	 r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 ← d → 00 $\boxed{000}$ 110	DD CB	4	6	23	001 C 010 D 011 E 100 H 101 L 111 A
RLC (IY+d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 ← d → 00 $\boxed{000}$ 110	FD CB	4	6	23	
RL s	 s≡r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{010}$					Instruction format and states are as shown for RLC's. To form new Op-Code replace $\boxed{000}$ of RLC's with shown code
RRC s	 s≡r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{001}$					
RR s	 s≡r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{011}$					
SLA s	 s≡r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{100}$					
SRA s	 s≡r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{101}$					
SRL s	 s≡r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	$\boxed{111}$					
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-7

BIT SET, RESET AND TEST GROUP

	Symbolic	Flags								Op-Code		No. of	No.of M	No.of T		
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments	
BIT b, r	$Z \leftarrow \overline{r}_b$	X	↕	X	1	X	X	0	●	11 001 011	CB	2	2	8	r	Reg.
										01 b r					000	B
BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	X	↕	X	1	X	X	0	●	11 001 011	CB	2	3	12	001	C
										01 b 110					010	D
BIT b, $(IX+d)_b$	$Z \leftarrow \overline{(IX+d)}_b$	X	↕	X	1	X	X	0	●	11 011 101	DD	4	5	20	011	E
										11 001 011	CB				100	H
										← d →					101	L
										01 b 110					111	A
															b	Bit Tested
BIT b, $(IY+d)_b$	$Z \leftarrow \overline{(IY+d)}_b$	X	↕	X	1	X	X	0	●	11 111 101	FD	4	5	20	000	0
										11 001 011	CB				001	1
										← d →					010	2
										01 b 110					011	3
															100	4
															101	5
															110	6
															111	7
SET b, r	$r_b \leftarrow 1$	●	●	X	●	X	●	●	●	11 001 011	CB	2	2	8		
										$\boxed{11}$ b r						
SET b, (HL)	$(HL)_b \leftarrow 1$	●	●	X	●	X	●	●	●	11 001 011	CB	2	4	15		
										$\boxed{11}$ b 110						
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	●	●	X	●	X	●	●	●	11 011 101	DD	4	6	23		
										11 001 011	CB					
										← d →						
										$\boxed{11}$ b 110						
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	●	●	X	●	X	●	●	●	11 111 101	FD	4	6	23		
										11 001 011	CB					
										← d →						
										$\boxed{11}$ b 110						
RES b, s	$s_b \leftarrow 0$ s ≡ r, (HL), (IX+d), (IY+d)	●	●	X	●	X	●	●	●	$\boxed{10}$					To form new Op-Code replace $\boxed{11}$ of SET b, s with $\boxed{10}$. Flags and time states for SET instruction	

Notes: The notation s_b indicates bit b (0 to 7) or location s.

Flag Notation: ● = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-8

JUMP GROUP

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Op-Code 76 543 210	Hex	No. of Bytes	No.of M Cycles	No.of T States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011	C3	3	3	10	
										← n →					
										← n →					cc | Condition
JP cc, nn	If condition cc	•	•	X	•	X	•	•	•	11 cc 010		3	3	10	000 | NZ non zero
	is true PC ← nn,									← n →					001 | Z zero
	otherwise									← n →					010 | NC non carry
	continue														011 | C carry
															100 | PO parity odd
															101 | PE parity even
															110 | P sign positive
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000	18	2	3	12	111 | M sign negative
										← e-2 →					
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 ← e-2 →	38	2	2	7	If condition not met
	If C = 1, PC ← PC+e											2	3	12	If condition is met
JR NC, e	If C = 1, continue	•	•	X	•	X	•	•	•	00 110 000 ← e-2 →	30	2	2	7	If condition not met
	If C = 0, PC ← PC+e											2	3	12	If condition is met
JR Z, e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 ← e-2 →	28	2	2	7	If condition not met
	If Z = 1, PC ← PC+e											2	3	12	If condition is met
JR NZ, e	If Z = 1, continue	•	•	X	•	X	•	•	•	00 100 000 ← e-2 →	20	2	2	7	If condition not met
	If Z = 0, PC ← PC+e											2	3	12	If condition is met
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	
JP (IY)	PC ← IY	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	B ← B-1 If B = 0, continue	•	•	X	•	X	•	•	•	00 010 000 ← e-2 →	10	2	2	8	If B = 0
	If B ≠ 0, PC ← PC+e											2	3	13	If B ≠ 0

Notes: e represents the extension in the relative addressing mode.

e is a signed two's complement number in the range <126, 129>

e-2 in the op-code provides an effective address of pc+e as PC is incremented by 2 prior to the addition of e.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-9

CALL AND RETURN GROUP

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	(SP-1) ← PC_H	•	•	X	•	X	•	•	•	11 001 101	CD	3	5	17	
	(SP-2) ← PC_L									← n →					
	PC ← nn									← n →					
CALL cc, nn	If condition	•	•	X	•	X	•	•	•	11 cc 100		3	3	10	If cc is false
	cc is false									← n →					
	continue,									← n →		3	5	17	If cc is true
	otherwise														
	same as														
	CALL nn														
RET	PC_L ← (SP)	•	•	X	•	X	•	•	•	11 001 001	C9	1	3	10	
	PC_H ← (SP+1)														
RET cc	If condition	•	•	X	•	X	•	•	•	11 cc 000		1	1	5	If cc is false
	cc is false														
	continue,											1	3	11	If cc is true
	otherwise														
	same as														
	RET														
RETI	Return from	•	•	X	•	X	•	•	•	11 101 101	ED	2	4	14	
	interrupt									01 001 101	4D				
RETN[1]	Return from	•	•	X	•	X	•	•	•	11 101 101	ED	2	4	14	
	non maskable									01 000 101	45				
	interrupt														
RST p	(SP-1) ← PC_H	•	•	X	•	X	•	•	•	11 t 111		1	3	11	
	(SP-2) ← PC_L														
	PC_H ← 0														
	PC_L ← p														

cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

[1] RETN loads IFF_2 → IFF_1

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-10

INPUT AND OUTPUT GROUP

Mnemonic	Symbolic Operation	Flags								Op-Code		No. of Bytes	No. of M Cycles	No. of T States	Comments
		S	Z		H		P/V	N	C	76 543 210	Hex				
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	•	11 011 011 ← n →	DB	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕	X	↕	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	X	① ↕	X	X	X	X	1	X	11 101 101 10 100 010	ED A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 110 010	ED B2	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	X	① ↕	X	X	X	X	1	X	11 101 101 10 101 010	ED AA	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 111 010	ED BA	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	•	11 010 011	D3	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	•	11 101 101 01 r 001	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	X	① ↕	X	X	X	X	1	X	11 101 101 10 100 011	ED A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 110 011	ED B3	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	X	① ↕	X	X	X	X	1	X	11 101 101 10 101 011	ED AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTDR	(C) ← (HL) B ← B - 1 HL ← HL - 1 Repeat until B = 0	X	1	X	X	X	X	1	X	11 101 101 10 111 011	ED BB	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

Notes: ① If the result of B - 1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
↕ = flag is affected according to the result of the operation.

Table 7.0-11

8.0 INTERRUPT RESPONSE

The prupose of an interrupt is to allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start a peripheral service routine. Usually this service routine is involved with the exchange of data, or status and control information, between the CPU and the peripheral. Once the service routine is completed, the CPU returns to the operation from which it was interrupted.

INTERRUPT ENABLE – DISABLE

The Z80-CPU has two interrupt inputs, a software maskable interrupt and a non-maskable interrupt. The non-maskable interrupt ($\overline{\text{NMI}}$) can not be disabled by the programmer and it will be accepted whenever a peripheral device requests it. This interrupt is generally reserved for very important functions that must be serviced whenever they occur, such as an impending power failure. The maskable interrupt ($\overline{\text{INT}}$) can be selectively enabled or disabled by the programmer. This allows the programmer to disable the interrupt during periods where his program has timing constraints that do not allow it to be interrupted. In the Z80-CPU there is an enable flip flop (called IFF) that is set or reset by the programmer using the Enable Interrupt (EI) and Disable Interrupt (DI) instructions. When the IFF is reset, an interrupt can not be accepted by the CPU.

Actually, for purposes that will be subsequently explained, there are two enable flip flops, called IFF$_1$ and IFF$_2$.

IFF$_1$	IFF$_2$
Actually disables interrupts from being accepted.	Temporary storage location for IFF$_1$.

The state of IFF$_1$ is used to actually inhibit interrupts while IFF$_2$ is used as a temporary storage location for IFF$_1$. The purpose of storing the IFF$_1$ will be subsequently explained.

A reset to the CPU will force both IFF$_1$ and IFF$_2$ to the reset state so that interrupts are disabled. They can then be enabled by an EI instruction at any time by the programmer. When an EI instruction is executed, any pending interrupt request will not be accepted until after the instruction following EI has been executed. This single instruction delay is necessary for cases when the following instruction is a return instruction and interrupts must not be allowed until the return has been completed. The EI instructions sets both IFF$_1$ and IFF$_2$ to the enable state. When an interrupt is accepted by the CPU, both IFF$_1$ and IFF$_2$ are automatically reset, inhibiting further interrupts until the programmer wishes to issue a new EI instruction. Note that for all of the previous cases, IFF$_1$ and IFF$_2$ are always equal.

The purpose of IFF$_2$ is to save the status of IFF$_1$ when a non-maskable interrupt occurs. When a non-maskable interrupt is accepted, IFF$_1$ is reset to prevent further interrupts until reenabled by the programmer. Thus, after a non-maskable interrupt has been accepted maskable interrupts are disabled but the previous state of IFF$_1$ has been saved so that the complete state of the CPU just prior to the non-maskable interrupt can be restored at any time. When a Load Register A with Register I (LD A, I) instruction or a Load Register A with Register R (LD A, R) instruction is executed, the state of IFF$_2$ is copied into the parity flag where it can be tested or stored.

A second method of restoring the status of IFF$_1$ is thru the execution of a Return From Non-Maskable Interrupt (RETN) instruction. Since this instruction indicates that the non maskable interrupt service routine is complete, the contents of IFF$_2$ are now copied back into IFF$_1$, so that the status of IFF$_1$ just prior to the acceptance of the non-maskable interrupt will be restored automatically.

Figure 8.0-1 is a summary of the effect of different instructions on the two enable flip flops.

INTERRUPT ENABLE/DISABLE FLIP FLOPS

Action	IFF$_1$	IFF$_2$	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	IFF$_2$ → Parity flag
LD A, R	•	•	IFF$_2$ → Parity flag
Accept NMI	0	•	
RETN	IFF$_2$	•	IFF$_2$ → IFF$_1$
Accept INT	0	0	
RETI	•	•	

"•" indicates no change

FIGURE 8.0-1

CPU RESPONSE

Non-Maskable

A non-maskable interrupt will be accepted at all times by the CPU. When this occurs, the CPU ignores the next instruction that it fetches and instead does a restart to location 0066H. Thus, it behaves exactly as if it had received a restart instruction but, it is to a location that is not one of the 8 software restart locations. A restart is merely a call to a specific address in page 0 memory.

Maskable

The CPU can be programmed to respond to the maskable interrupt in any one of three possible modes.

Mode 0

This mode is identical to the 8080A interrupt response mode. With this mode, the interrupting device can place any instruction on the data bus and the CPU will execute it. Thus, the interrupting device provides the next instruction to be executed instead of the memory. Often this will be a restart instruction since the interrupting device only need supply a single byte instruction. Alternatively, any other instruction such as a 3 byte call to any location in memory could be executed.

The number of clock cycles necessary to execute this instruction is 2 more than the normal number for the instruction. This occurs since the CPU automatically adds 2 wait states to an interrupt response cycle to allow sufficient time to implement an external daisy chain for priority control. Section 4.0 illustrates the detailed timing for an interrupt response. After the application of RESET the CPU will automatically enter interrupt Mode 0.

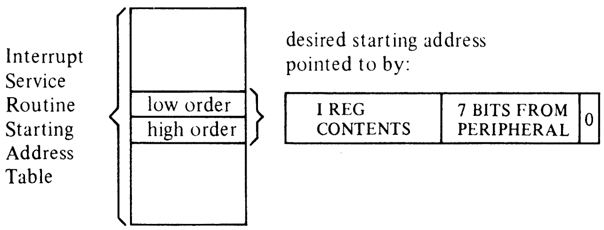
Mode 1

When this mode has been selected by the programmer, the CPU will respond to an interrupt by executing a restart to location 0038H. Thus the response is identical to that for a non maskable interrupt except that the call location is 0038H instead of 0066H. Another difference is that the number of cycles required to complete the restart instruction is 2 more than normal due to the two added wait states.

Mode 2

This mode is the most powerful interrupt response mode. With a single 8-bit byte from the user an indirect call can be made to any memory location.

With this mode the programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16 bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer is formed from the contents of the I register. The I register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that a CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Actually, only 7 bits are required from the interrupting device as the least bit must be a zero. This is required since the pointer is used to get two adjacent bytes to from a complete 16 bit service routine starting address and the addresses must always start in even locations.


Interrupt Service Routine Starting Address Table
low order
high order
desired starting address pointed to by:
I REG CONTENTS
7 BITS FROM PERIPHERAL
0

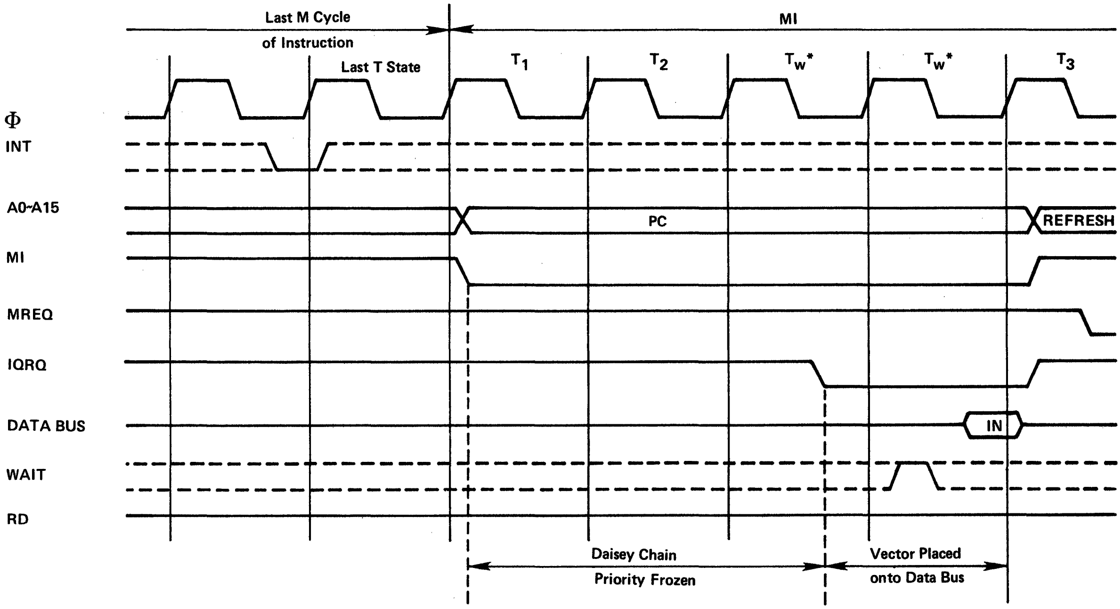
The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time by the programmer (if it is stored in Read/Write Memory) to allow different peripherals to be serviced by different service routines.

Once the interrupting device supplies the lower portion of the pointer, the CPU automatcally pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address.)

Note that the Z80 peripheral devices all include a daisy chain priority interrupt structure that automatically supplies the programmed vector to the CPU during interrupt acknowledge. Refer to the Z80-PIO, Z80-SIO and Z80-CTC manuals for details.

INTERRUPT REQUEST/ACKNOWLEDGE CYCLE



Z80 INTERRUPT ACKNOWLEDGE SUMMARY

1) PERIPHERAL DEVICE REQUESTS INTERRUPT. Any device requesting and interrupt can pull the wired-or line $\overline{\text{INT}}$ low.

2) CPU ACKNOWLEDGES INTERRUPT. Priority status is frozen when $\overline{\text{M1}}$ goes low during the Interrupt Acknowledge sequence. Propagation delays down the IEI/IEO daisy chain must be settled out when $\overline{\text{IORQ}}$ goes low. If IEI is HIGH, an active Peripheral Device will place its Interrupt Vector on the Data Bus when $\overline{\text{IORQ}}$ goes low. That Peripheral then releases its hold on $\overline{\text{INT}}$ allowing interrupts from a higher priority device. Lower priority devices are inhibited from placing their Vector on the Data Bus or Interrupting because IEO is low on the active device.

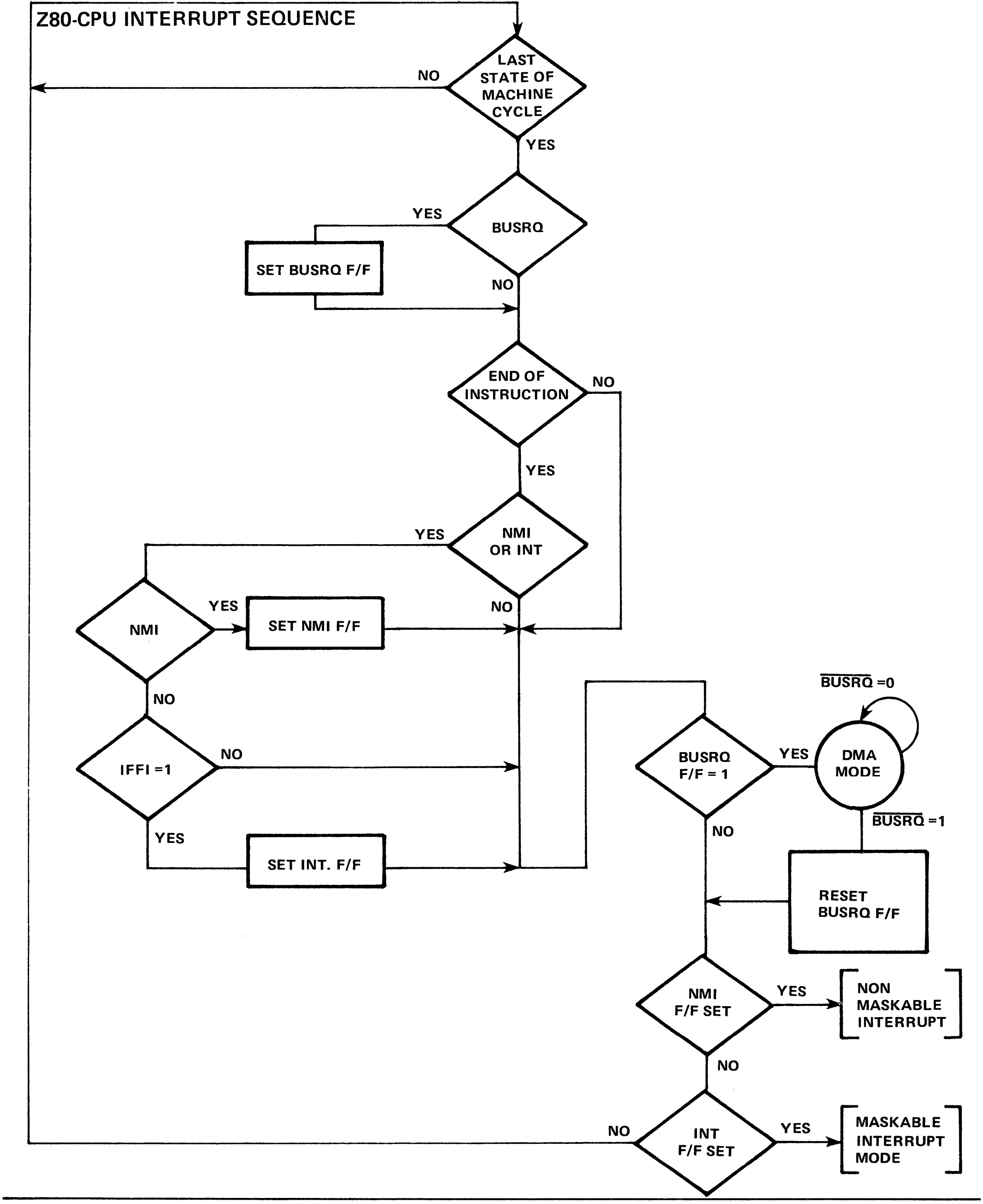
3) INTERRUPT IS CLEARED. An active Peripheral device (IEI=1, IEO=0) monitors OP Code fetches for an RETI (ED 4D) instruction which tells the peripheral that its Interrupt Service Routine is over. The peripheral device then re-activates its internal Interrupt structure as well as raising its IEO line to enable lower priority devices.

INTERRELATIONSHIP OF $\overline{\text{INT}}$, $\overline{\text{NMI}}$, AND $\overline{\text{BUSRQ}}$

The following flow chart details the relationship of three control inputs to the Z80-CPU. Note the following from the flow chart.

1. $\overline{\text{INT}}$ and $\overline{\text{NMI}}$ are always acted on at the end of an instruction.
2. $\overline{\text{BUSRQ}}$ is acted on at the end of a machine cycle.
3. While the CPU is in the DMA MODE, it will not respond to active inputs on $\overline{\text{INT}}$ or $\overline{\text{NMI}}$.
4. These three inputs are acted on in the following order of priority: a)$\overline{\text{BUSRQ}}$ b)$\overline{\text{NMI}}$ c)$\overline{\text{INT}}$

Z80-CPU INTERRUPT SEQUENCE



9.0 HARDWARE IMPLEMENTATION EXAMPLES

This chapter is intended to serve as a basic introduction to implementing systems with the Z80-CPU.

MINIMUM SYSTEM

Figure 9.0-1 is a diagram of a very simple Z80 system. Any Z80 system must include the following five elements:

1) Five volt power supply
2) Oscillator
3) Memory devices
4) I/O circuits
5) CPU

MINIMUM Z80 COMPUTER SYSTEM

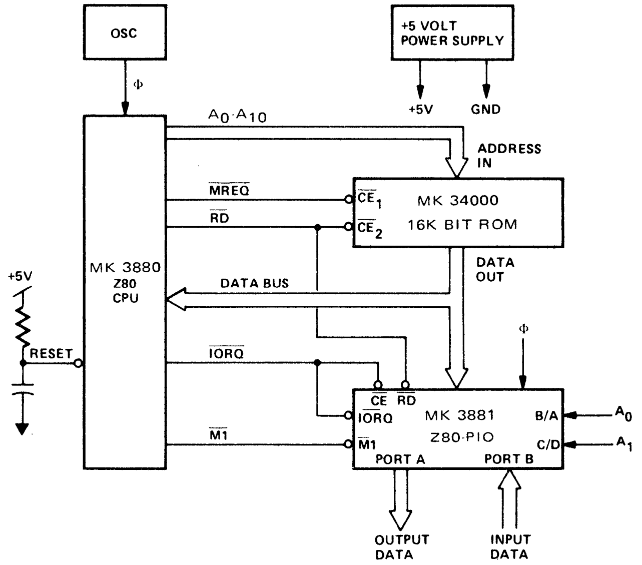


FIGURE 9.0-1

Since the Z80-CPU only requires a single 5 volt supply, most small systems can be implemented using only this single supply.

The oscillator can be very simple since the only requirement is that it be a 5 volt square wave. For systems not running at full speed, a simple RC oscillator can be used. When the CPU is operated near the highest possible frequency, a crystal oscillator is generally required because the system timing will not tolerate the drift or jitter that an RC network will generate. A crystal oscillator can be made from inverters and a few discrete components or monolithic circuits are widely available.

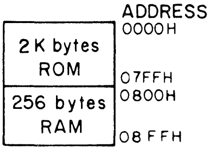
The external memory can be any mixture of standard RAM, ROM, or PROM. In this simple example we have shown a single 16K bit ROM (2K bytes) being utilized as the entire memory system. For this example we have assumed that the Z80 internal register configuration contains sufficient Read/Write storage so that external RAM memory is not required.

Every computer system requires I/O circuits to allow it to interface to the "real world." In this simple example it is assumed that the output is an 8 bit control vector and the input is an 8 bit status word. The input data could be gated onto the data bus using any standard tri-state driver while the output data could be latched with any type of standard TTL latch. For this example we have used a Z80-PIO for the I/O circuit. This single circuit attaches to the data bus as shown and provides the required 16 bits of TTL compatible I/O. (Refer to the Z80-PIO manual for details on the operation of this circuit.) Notice in this example that with only three LSI circuits, a simple oscillator and a single 5 volt power supply, a powerful computer has been implemented.

ADDING RAM

Most computer systems require some amount of external Read/Write memory for data storage and to implement a "stack". Figure 9.0-2 illustrates how 256 bytes of static memory can be added to the previous example. In this example the memory space is assumed to be organized as follows:

ROM & RAM IMPLEMENTATION EXAMPLE



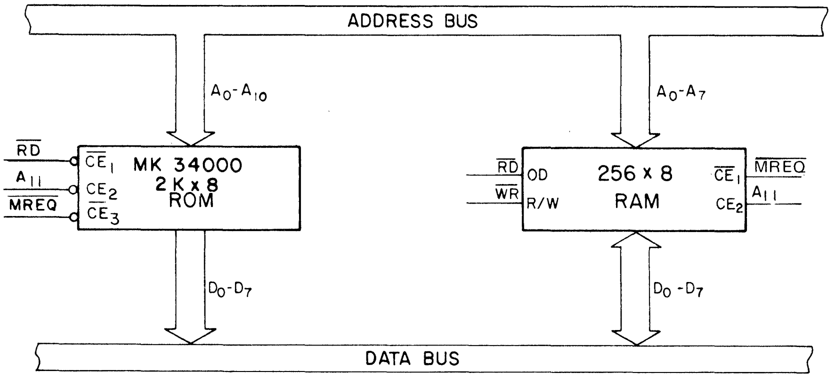


FIGURE 9.0-2

In this diagram the address space is described in hexidecimal notation. For this example, address bit A_{11} separates the ROM space from the RAM space so that it can be used for the chip select function. For larger amounts of external ROM or RAM, a simple TTL decoder will be required to form the chip selects.

MEMORY SPEED CONTROL

For many applications, it may be desirable to use slow memories to reduce costs. The WAIT line on the CPU allows the Z80 to operate with any speed memory. By referring back to section 4 you will notice that the memory access time requirements are most severe during the M1 cycle instruction fetch. All other memory accesses have an additional one half of a clock cycle to be completed. For this reason it may be desirable in some applications to add one wait state to the M1 cycle so that slower memories can be used. Figure 9.0-3 is an example of a simple circuit that will accomplish this task. This circuit can be changed to add a single wait state to any memory access as shown in F,gure 9.0-4.

ADDING ONE WAIT STATE TO AN M1 CYCLE

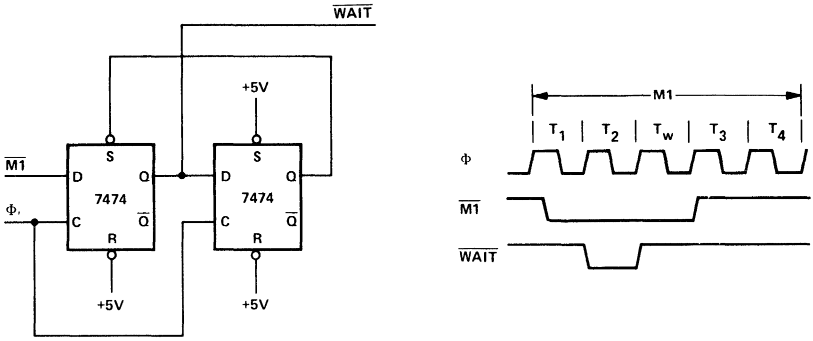


FIGURE 9.0-3

ADDING ONE WAIT STATE TO ANY MEMORY CYCLE

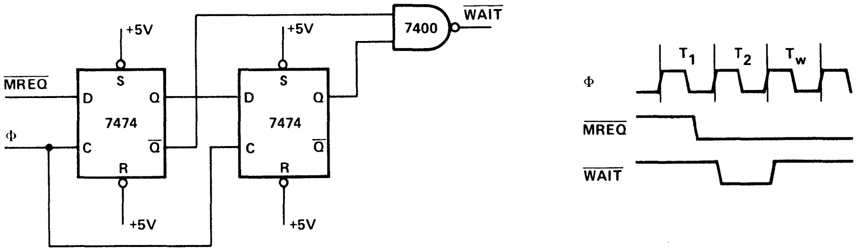


FIGURE 9.0-4

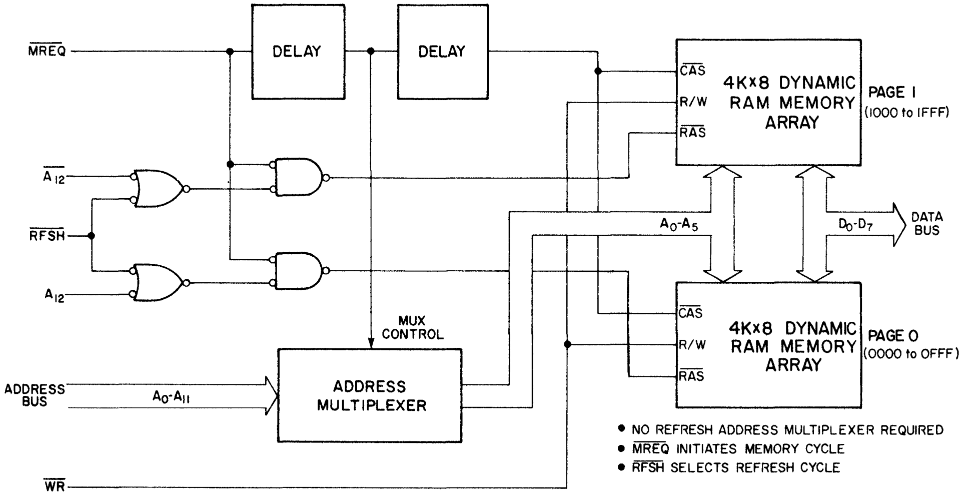
INTERFACING DYNAMIC MEMORIES

This section is intended only to serve as a brief introduction to interfacing dynamic memories. Each individual dynamic RAM has varying specifications that will require minor modifications to the description given here and no attempt will be made in this document to give details for any particular RAM.

Figure 9.0-5 illustrates the logic necessary to interface 8K bytes of dynamic RAM using 16-pin 4K dynamic memories. This Figure assumes that the RAM's are the only memory in the system so that A_{12} is used to select between the two pages of memory. During refresh time, all memories in the system must be read. The CPU provides the proper refresh address on lines A_0 through A_6. To add additional memory to the system it is necessary to only replace the two gates that operate on A_{12} with a decoder that operates on all required address bits. For larger systems, buffering for the address and data bus is also generally required.

An application note entitled "Z80 Interfacing Techniques for Dynamic RAM" is available from your MOSTEK representative which describes dynamic RAM design techniques.

INTERFACING DYNAMIC RAMS



- NO REFRESH ADDRESS MULTIPLEXER REQUIRED
- MREQ INITIATES MEMORY CYCLE
- RFSH SELECTS REFRESH CYCLE

FIGURE 9.0-5

Z80–CPU DESIGN CONSIDERATIONS: CLOCK CIRCUITRY

When using the Z80-CPU at less than its rated speed, the Clock Input (Φ) can be driven by a 7400 TTL gate with a resistor pull up (typically 330 ohms) to +5 Volts. Because of dynamic currents flowing into the Clock Input Pin, the rise time of the Clock Input waveform will be typically 60-80 nanoseconds. The resistor will eventually pull the clock input up to Vcc but with a slow rise time which will limit the maximum frequency of operation. Figure 9.0-6 shows a Clock Input driver which has an active pull-up and which will allow maximum frequency operation. The circuit is recommended for all but the most cost sensitive Z80 applications.

Z80 CPU CLOCK BUFFER CIRCUITRY

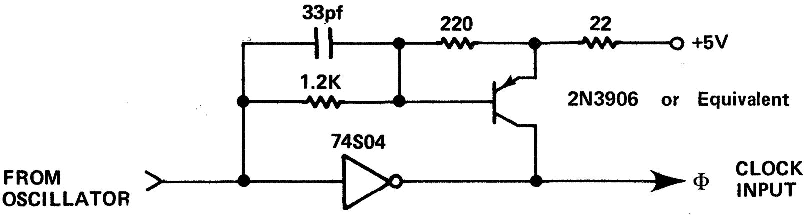


FIGURE 9.0-6

RESET CIRCUITRY

The Z80-CPU has the characteristic that if the $\overline{\text{RESET}}$ input goes low during T2 or T4 of a cycle that the MREQ signal will go to an indeterminate state for one T-State approximately 3 T-States later. If there are dynamic memories in the system this action could cause an aborted or short access of the dynamic RAM which could cause destruction of data within the RAM. If the contents of RAM are of no concern after RESET, then this characteristic is no problem as the CPU always resets properly. If RAM contents must be preserved, then the falling edge of the $\overline{\text{RESET}}$ input must be synchronized by the falling edge of $\overline{\text{M1}}$.

The circuitry of Figure 9.0-7 does this synchronization as well as providing a one-shot to limit the duration of the CPU $\overline{\text{RESET}}$ pulse. The CPU $\overline{\text{RESET}}$ signal must be a pulse even though the EXTERNAL RESET button is held closed to avoid suspending the CPU refresh of dynamic RAM for a time long enough to destroy data in the RAM.

MANUAL AND POWER—ON RESET CIRCUIT

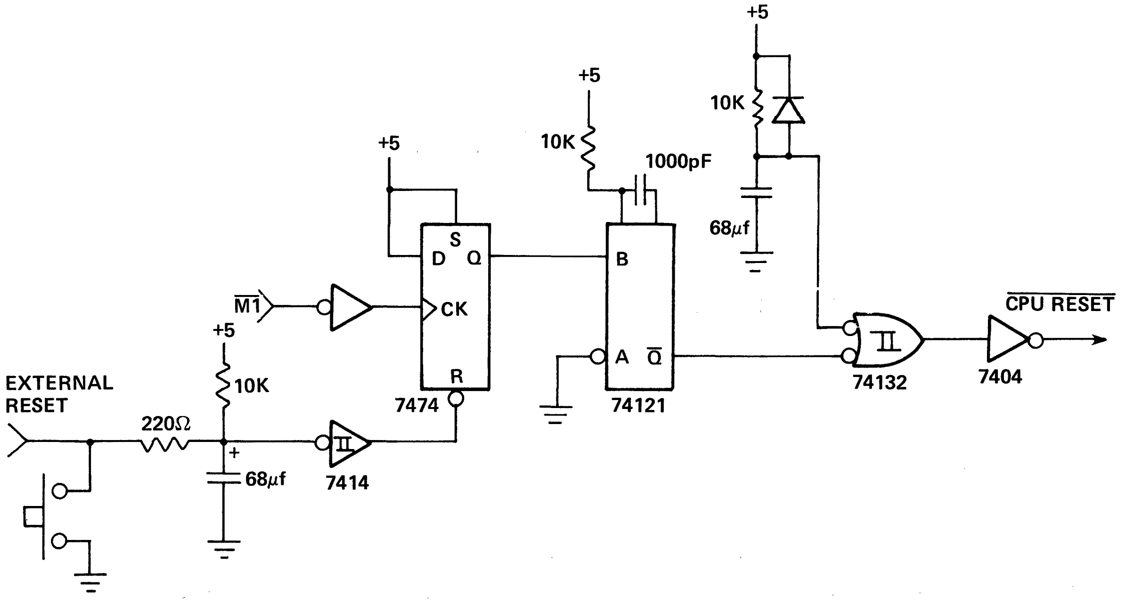


FIGURE 9.0-7

ADDRESS LATCHING

In order to guarantee proper operation of the Z80-CPU with dynamic RAMs the upper 4 bits of the address should be latched as shown in Figure 9.0-8. This action is required because the Z80-CPU does not guarantee that the Address Bus will hold valid before the rising edge of MREQ on an OP Code Fetch.

This action does not directly affect dynamic memories because they latch addresses internally. The problem comes from the address decoder which generates RAS. If the address lines which drive the decoder are allowed to change while $\overline{\text{MREQ}}$ is low, then a "glitch" can occur on the $\overline{\text{RAS}}$ line or lines, which may have the effect of destroying one row of data within the dynamic RAM.

ADDRESS LATCH

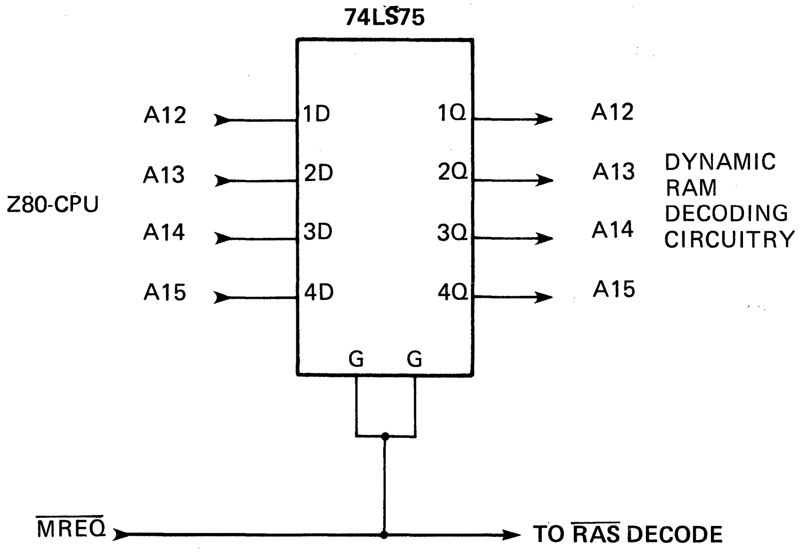


FIGURE 9.0-8

$\overline{\text{RAS}}$ TIMING WITH AND WITHOUT ADDRESS LATCH.

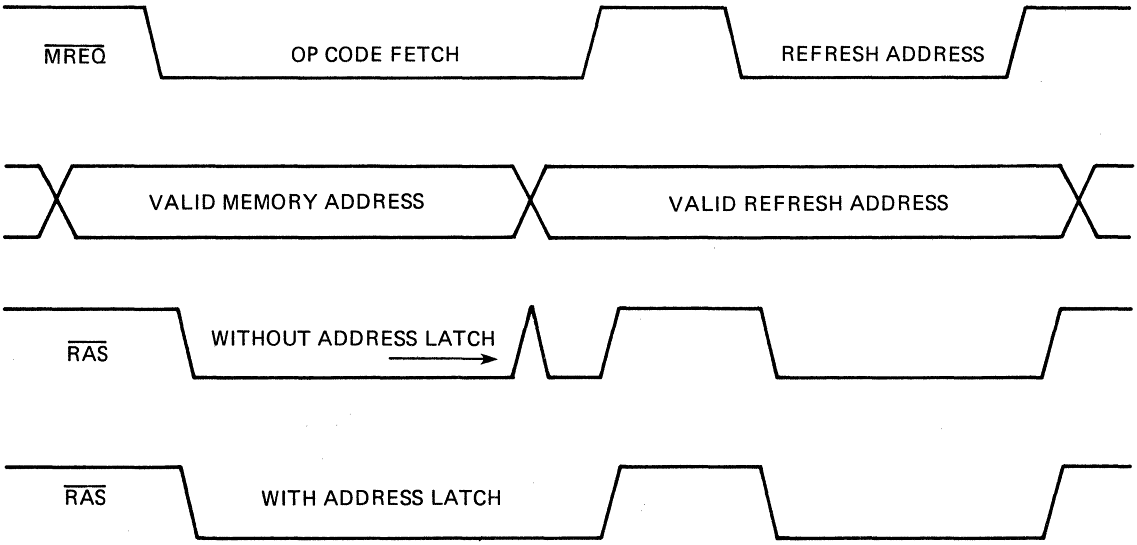


FIGURE 9.0-9

10.0 SOFTWARE IMPLEMENTATION EXAMPLES

10.1 Methods of Software Implementation

Several different approaches are possible in developing software for the Z80 (Figure 10.1) First of all, Assembly Language or a high level language may be used as the source language. These languages may then be translated into machine language on a commercial time sharing facility using a cross-assembler or cross-compiler or, in the case of assembly language, the translation can be accomplished on a Z80 Development System using a resident assembler. Finally, the resulting machine code can be debugged either on a time-sharing facility using a Z80 simulator or on a Z80 Development System which uses a Z80-CPU directly.

SOFTWARE GENERATION TECHNIQUES

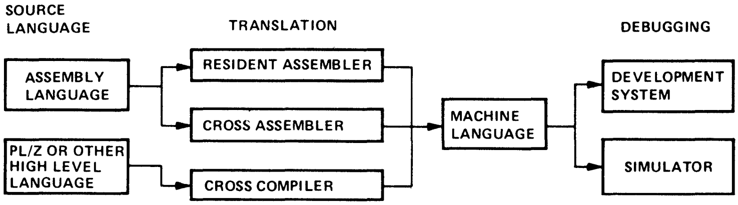


FIGURE 10.1

In selecting a source language, the primary factors to be considered are clarity and ease of programming vs. code efficiency. A high level language with its machine independent constraints is typically better for formulating and maintaining algorithms, but the resulting machine code is usually somewhat less efficient than what can be written directly in assembly language. These tradeoffs can often be balanced by combining high level language and assembly language routines, identifying those portions of a task which must be optimized and writing them as assembly language subroutines.

Deciding whether to use a resident or cross assembler is a matter of availability and short-term vs. long-term expense. While the initial expenditure for a development system is higher than that for a time-sharing terminal, the cost of an individual assembly using a resident assembler is negligible while the same operation on a time-sharing system is relatively expensive and in a short time this cost can equal the total cost of a development system.

Debugging on a development system vs. a simulator is also a matter of availability and expense combined with operational fidelity and flexibility. As with the assembly process, debugging is less expensive on a development system than on a simulator available through time-sharing. In addition, the fidelity of the operating environment is preserved through real-time execution on a Z80-CPU and by connecting the I/O and memory components which will actually be used in the production system. The only advantage to the use of a simulator is the range of criteria which may be selected for such debugging procedures as tracing and setting breakpoints. This flexibility exists because a software simulation can achieve any degree of complexity in its interpretation of machine instructions while development system procedures have hardware limitations such as the capacity of the real-time storage module, the number of breakpoint registers and the pin configuration of the CPU. Despite such hardware limitations, debugging on a development system is typically more productive than on a simulator because of the direct interaction that is possible between the programmer and the authentic execution of his program.

10.2 Software Features Offered by the Z80-CPU

The Z80 instruction set provides the user with a large and flexible repetoire of operations with which to formulate control of the Z80-CPU.

The primary, auxiliary and index registers can be used to hold the arguments of arithmetic and logical operations, or to form memory addresses, or as fast-access storage for frequently used data.

Information can be moved directly from register to register; from memory to memory; from memory to registers; or from registers to memory. In addition, register contents and register/memory contents can be exchanged without using temporary storage. In particular, the contents of primary and auxiliary registers can be completely exchanged by executing only two instructions. EX and EXX. This register exchange procedure can be used to separate the set of working registers between different logical procedures or to expand the set of available registers in a single procedure.

Storage and retrieval of data between pairs of registers and memory can be controlled on a last-in first-out basis through PUSH and POP instructions which utilize a special stack pointer register, SP. This stack register is available both to manipulate data and to automatically store and retrieve addresses for subroutine linkage. When a subroutine is called, for example, the address following the CALL instruction is placed on the top of the pushdown stack pointed to by SP. When a subroutine returns to the calling routine, the address on the top of the stack is used to set the program counter for the address of the next instruction. The stack pointer is adjusted automatically to reflect the current "top" stack position during PUSH, POP, CALL and RET instructions. This stack mechanism allows pushdown data stacks and subroutine calls to be nested to any practical depth because the stack area can potentially be as large as memory space.

The sequence of instruction execution can be controlled by six different flags (carry, zero, sign, parity/overflow, add-subtract, half-carry) which reflect the results of arithmetic, logical, shift and compare instructions. After the execution of an instruction which sets a flag, that flag can be used to control a conditional jump or return instruction. These instructions provide logical control following the manipulation of single bit, eight-bit byte (or) sixteen-bit data quantities.

A full set of logical operations, including AND, OR, XOR (exclusive –OR), CPL (NOR) and NEG (two's complement) are available for Boolean operations between the accumulator and 1) all other eight-bit registers, 2) memory locations or 3) immediate operands.

In addition, a full set of arithmetic and logical shifts in both directions are available which operate on the contents of all eight-bit primary registers or directly on any memory location. The carry flag can be included or simply set by these shift instructions to provide both the testing of shift results and to link register/register or register/memory shift operations.

10.3 Examples of Use of Special Z80 Instructions

A. Let us assume that a string of data in memory starting at location "DATA" is to be moved into another area of memory starting at location "BUFFER" and that the string length is 737 bytes. This operation can be accomplished as follows:

```
LD      HL, DATA      ;START ADDRESS OF DATA STRING
LD      DE, BUFFER    ;START ADDRESS OF TARGET BUFFER
LD      BC, 737       ;LENGTH OF DATA STRING
LDIR                  ;MOVE  STRING  –  TRANSFER  MEMORY
                      ;POINTED TO BY HL INTO MEMORY
                      ;LOCATION POINTED TO BY DE INCREMENT
                      ;HL  AND  DE,  DECREMENT  BC  PROCESS
                      ;UNTIL BC=0.
```

11 bytes are required for this operation and each byte of data is moved in 21 clock cycles.

B. Let's assume that a string in memory starting at location "DATA" is to be moved into another area of memory starting at location "BUFFER" until an ASCII $ character (used as string delimiter) is found. Let's also assume that the maximum string length is 132 characters. The operation can be performed as follows:

```
         LD      HL, DATA      ;STARTING ADDRESS OF DATA STRING
         LD      DE, BUFFER    ;STARTING ADDRESS OF TARGET BUFFER
         LD      BC, 132       ;MAXIMUM STRING LENGTH
         LD      A, '$'        ;STRING DELIMITER CODE
   LOOP: CP      (HL)          ;COMPARE MEMORY CONTENTS WITH DE-
                               ;LIMITER
         JR      Z, END-$      ;GO TO END IF CHARACTERS EQUAL
         LDI                   ;MOVE CHARACTER (HL) TO (DE)
                               ;INCREMENT HL AND DE, DECREMENT BC
         JP      PE,LOOP       ;GO TO "LOOP" IF MORE CHARACTERS
    END:                       ;OTHERWISE, FALL THROUGH
                               ;NOTE: P/V FLAG IS USED
                               ;TO INDICATE THAT REGISTER BC WAS
                               ;DECREMENTED TO ZERO.
```

19 bytes are required for this operation.

C. Let us assume that a 16-digit decimal number represented in packed BCD format (two BCD digits/byte) has to be shifted as shown in the Figure 10.2 in order to mechanize BCD multiplication or division. The operation can be accomplished as follows:

```
         LD      HL, DATA      ;ADDRESS OF FIRST BYTE
         LD      B, COUNT      ;SHIFT COUNT
         XOR     A             ;CLEAR ACCUMULATOR
  ROTAT: RLD                   ;ROTATE LEFT LOW ORDER DIGIT IN ACC
                               ;WITH DIGITS IN (HL)
         INC     HL            ;ADVANCE MEMORY POINTER
         DJNZ    ROTAT-$       ;DECREMENT B AND GO TO ROTAT IF
                               ;B IS NOT ZERO, OTHERWISE FALL THROUGH
```

BCD DATA SHIFTING

11 bytes are required for this operation.

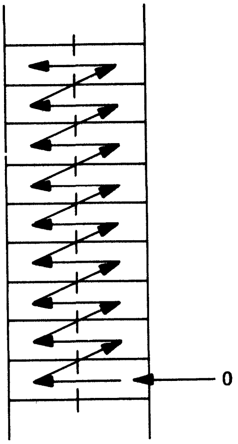


FIGURE 10.2

11 bytes are required for this operation.

D. Let us assume that one number is to be subtracted from another and a) that they are both in packed BCD format, b) that they are of equal but varying length, and c) that the result is to be stored in the location of the minuend. The operation can be accomplished as follows:

```
        LD      HL, ARG1      ;ADDRESS OF MINUEND
        LD      DE, ARG2      ;ADDRESS OF SUBTRAHEND
        LD      B, LENGTH     ;LENGTH OF TWO ARGUMENTS
        AND     A             ;CLEAR CARRY FLAG
SUBDEC: LD      A, (DE)       ;SUBTRAHEND TO ACC
        SBC     A, (HL)       ;SUBTRACT (HL) FROM ACC
        DAA                   ;ADJUST RESULT TO DECIMAL CODED VALUE
        LD      (HL), A       ;STORE RESULT
        INC     HL            ;ADVANCE MEMORY POINTERS
        INC     DE
        DJNZ    SUBDEC–$      ;DECREMENT B AND GO TO "SUBDEC" IF B
                              ;NOT ZERO, OTHERWISE FALL THROUGH
```

17 bytes are required for this operation.

10.4 Examples of Programming Tasks

A. The following program sorts an array of numbers each in the range <0,255> into ascending order using a standard exchange sorting algorithm.

```
01/22/76     11:14:37                 BUBBLE LISTING
LOC    OBJ CODE  STMT SOURCE STATEMENT

                  1    ; *** STANDARD EXCHANGE (BUBBLE) SORT ROUTINE***
                  2    ;
                  3    ; AT ENTRY: HL CONTAINS ADDRESS OF DATA
                  4    .           C CONTAINS NUMBER OF ELEMENTS TO BE SORTED
                  5    ;           (1<C<256)
                  6    ;
                  7    ; AT EXIT: DATA SORTED IN ASCENDING ORDER
                  8    ;
                  9    ; USE OF REGISTERS
                 10    ;
                 11    ; REGISTER   CONTENTS
                 12    ;
                 13    ; A          TEMPORARY STORAGE FOR CALCULATIONS
                 14    ; B          COUNTER FOR DATA ARRAY
                 15    ; C          LENGTH OF DATA ARRAY
                 16    ; D          FIRST ELEMENT IN COMPARISON
                 17    ; E          SECOND ELEMENT IN COMPARISON
                 18    ; H          FLAG TO INDICATE EXCHANGE
                 19    ; L          UNUSED
                 20    ; IX         POINTER INTO DATA ARRAY
                 21    ; IY         UNUSED
                 22    ;
```

```
01/22/76  11:14:37            BUBBLE LISTING (Cont'd.)

LOC    OBJ CODE  STMT SOURCE STATMENT
0000   222600    23   SORT:    LD     (DATA), HL    ;SAVE DATA ADDRESS
0003   CB84      24   LOOP:    RES    FLAG, H       ;INITIALIZE EXCHANGE FLAG
0005   41        25            LD     B,C           ;INITIALIZE LENGTH COUNTER
0006   05        26            DEC    B             ;ADJUST FOR TESTING
0007   DD2A2600  27            LD     IX, (DATA)    ;INITIALIZE ARRAY POINTER
000B   DD7E00    28   NEXT:    LD     A,(IX+0)      ;FIRST ELEMENT IN COMPARISON
000E   57        29            LD     D, A          ;TEMPORARY STORAGE FOR ELEMENT
000F   DD5E01    30            LD     E, (IX+1)     ;SECOND ELEMENT IN COMPARISON
0012   93        31            SUB    E             ;COMPARISON FIRST TO SECOND
0013   3008      32            JR     NC, NOEX-$    ;IF FIRST> SECOND, NO JUMP
0015   DD7300    33            LD     (IX), E       ;EXCHANGE ARRAY ELEMENTS
0018   DD7201    34            LD     (IX+1), D
001B   CBC4      35            SET    FLAG H        ;RECORD EXCHANGE OCCURRED
001D   DD23      36   NOEX:    INC    IX            ;POINT TO NEXT DATA ELEMENT
001F   10EA      37            DJNZ   NEXT-$        ;COUNT NUMBER OF COMPARISONS
                                                    ;REPEAT IF MORE DATA PAIRS
0021   CB44      39            BIT    FLAG, H       ;DETERMINE IF EXCHANGE OCCURRED
0023   20DE      40            JR     NZ, LOOP-$    ;CONTINUE IF DATA UNSORTED
0025   C9        41            RET                  ;OTHERWISE, EXIT
                 42   ;
0026             43   FLAG:    EQU    0             ;DESIGNATION OF FLAG BIT
0026             44   DATA:    DEFS   2             ;STORAGE FOR DATA ADDRESS
                 45            END
```

B. The following program multiplies two unsigned 16-bit integers and leaves the result in the HL register pair.

```
01/22/76     11:32:36               MULTIPLY LISTING
LOC    OBJ CODE    STMT   SOURCE STATEMENT

0000                 1    MULT:;   UNSIGNED SIXTEEN BIT INTEGER MULTIPLY.
                     2    ;        ON ENTRANCE: MULTIPLIER IN HL.
                     3    ;                     MULTIPLICAND IN DE.
                     4    ;
                     5    ;        ON EXIT: RESULT IN HL.
                     6    ;
                     7    ;        REGISTERS USES:
                     8    ;
                     9    ;
                    10    ;        H    HIGH ORDER PARTIAL RESULT
                    11    ;        L    LOW ORDER PARTIAL RESULT
                    12    ;        D    HIGH ORDER MULTIPLICAND
                    13    ;        E    LOW ORDER MULTIPLICAND
                    14    ;        B    COUNTER FOR NUMBER OF SHIFTS
                    15    ;        C    HIGH ORDER BITS OF MULTIPLIER
                    16    ;        A    LOW ORDER BITS OF MULTIPLIER
                    17    ;
0000   0610         18             LD   B, 16;         NUMBER OF BITS–INITIALIZE
0002   4A           19             LD   C,D;           MOVE MULTIPLIER
0003   7B           20             LD   A,E;
0004   EB           21             EX   DE,HL;         MOVE MULTIPLICAND
0005   210000       22             LD   HL,0;          CLEAR PARTIAL RESULT
0008   CB39         23    MLOOP:   SRL  C;             SHIFT MULTIPLIER RIGHT
000A   1F           24             RR   A;             LEAST SIGNIFICANT BIT IS
                                                       IN CARRY.
000B   3001         26             JR   NC, NOADD-$    IF NO CARRY' SKIP THE ADD.
```

```
01/22/76  11:32:36                    MULTIPLY LISTING (Cont'd.)

LOC    OBJ CODE   STMT  SOURCE STATMENT
000D   19         27              ADD HL, DE;         ELSE ADD MULTIPLICAND TO
                                                      PARTIAL RESULT.
000E   EB         29    NOADD:    EX   DE,HL;         SHIFT MULTIPLICANT LEFT
000F   29         30              ADD HL,HL;          BY MULTIPLYING IT BY TWO.
0010   EB         31              EX   DE,HL;
0011   10F5       32            DJNZ  MLOOP-$;        REPEAT UNTIL NO MORE BITS.
0013   C9         33              RET;
                  34              END;
```

11.0 ELECTRICAL SPECIFICATIONS
ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . Specified Operating Range

Storage Temperature . . . . –65° C to +150° C

Voltage on Any Pin with Respect to Ground . . . . –0.3V to +7V

Power Dissipation 1.5W

D.C. CHARACTERISTICS
T_A = 0° C to 70° C, V_{CC} = 5V ± 5% unless otherwise specified

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	–0.3		0.8	V	
V_{IHC}	Clock Input High Voltage	Vcc-.6		Vcc+.3	V	
V_{IL}	Input Low Voltage	–0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = –250 μA
I_{CC}	Power Supply Current			150*	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			–10	μA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leqslant V_{IN} \leqslant V_{CC}$

*200mA for -4, -10 or -20 devices

CAPACITANCE
T_A = 25° C, f = 1MHz unmeasured pins returned to ground

SYMBOL	PARAMETER	MAX.	UNIT
$C\Phi$	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

*Comment

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

A C CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITION
Φ	t_c	Clock Period	.4	[12]	μsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	180	(D)	nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
$A_{0\text{-}15}$	$t_{D(AD)}$	Address Output Delay		145	nsec	
	$t_{F(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to $\overline{\text{MREQ}}$ (Memory Cycle)	[1]		nsec	C_L = 50pF
	t_{aci}	Address Stable Prior to $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ or $\overline{\text{WR}}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable From $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{IORQ}}$ or $\overline{\text{MREQ}}$	[3]		nsec	Except T3-M1
	t_{caf}	Address Stable From $\overline{\text{RD}}$ or $\overline{\text{WR}}$ During Float	[4]		nsec	
$D_{0\text{-}7}$	$t_{D(D)}$	Data Output Delay		230	nsec	
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge at Clock During M2 to M5	60		nsec	C_L = 50pF
	t_{dcm}	Data Stable Prior to $\overline{\text{WR}}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{\text{WR}}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{\text{WR}}$	[7]		nsec	
	t_H	Input Hold Time	0		nsec	
$\overline{\text{MREQ}}$	$t_{DL\Phi(MR)}$	$\overline{\text{MREQ}}$ Delay From Falling Edge of Clock, $\overline{\text{MREQ}}$ Low		100	nsec	
	$t_{DH\Phi(MR)}$	$\overline{\text{MREQ}}$ Delay From Rising Edge of Clock, $\overline{\text{MREQ}}$ High		100	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{\text{MREQ}}$ Delay From Falling Edge of Clock, $\overline{\text{MREQ}}$ High		100	nsec	C_L = 50 pF
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{\text{MREQ}}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{\text{MREQ}}$ High	[9]		nsec	
$\overline{\text{IORQ}}$	$t_{DL\Phi(IR)}$	$\overline{\text{IORQ}}$ Delay From Rising Edge of Clock, $\overline{\text{IORQ}}$ Low		90	nsec	
	$t_{DL\overline{\Phi}(IR)}$	$\overline{\text{IORQ}}$ Delay From Falling Edge of Clock, $\overline{\text{IORQ}}$ Low		110	nsec	C_L = 50 pF
	$t_{DH\Phi(IR)}$	$\overline{\text{IORQ}}$ Delay From Rising Edge of Clock, $\overline{\text{IORQ}}$ High		100	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{\text{IORQ}}$ Delay From Falling Edge of Clock, $\overline{\text{IORQ}}$ High		110	nsec	
$\overline{\text{RD}}$	$t_{DL\Phi(RD)}$	$\overline{\text{RD}}$ Delay From Rising Edge of Clock, $\overline{\text{RD}}$ Low		100	nsec	
	$t_{DL\overline{\Phi}(RD)}$	$\overline{\text{RD}}$ Delay From Falling Edge of Clock, $\overline{\text{RD}}$ Low		130	nsec	C_L = 50pF
	$t_{DH\Phi(RD)}$	$\overline{\text{RD}}$ Delay From Rising Edge of Clock, $\overline{\text{RD}}$ High		100	nsec	
	$t_{DH\overline{\Phi}(BD)}$	$\overline{\text{RD}}$ Delay From Falling Edge of Clock, $\overline{\text{RD}}$ High		110	nsec	
$\overline{\text{WR}}$	$t_{DL\Phi(WR)}$	$\overline{\text{WR}}$ Delay From Rising Edge of Clock, $\overline{\text{WR}}$ Low		80	nsec	
	$t_{DL\overline{\Phi}(WR)}$	$\overline{\text{WR}}$ Delay From Falling Edge of Clock $\overline{\text{WR}}$ Low		90	nsec	C_L = 50pF
	$t_{DH\Phi(WR)}$	$\overline{\text{WR}}$ Delay From Falling Edge of Clock, $\overline{\text{WR}}$ High		100	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{\text{WR}}$ Low	[10]		nsec	

NOTES:

A Data should be enabled onto the CPU data bus when RD is active. During interrupt acknowledge data should be enabled when $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are both active.

B The $\overline{\text{RESET}}$ signal must be active for a minimum of 3 clock cycles.

cont'd on page 81

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
$\overline{\text{M1}}$	$t_{DL(M1)}$	$\overline{\text{M1}}$ Delay From Rising Edge of Clock $\overline{\text{M1}}$ Low		130	nsec	$C_L = 50pF$
	$t_{DH(M1)}$	$\overline{\text{M1}}$ Delay From Rising Edge of Clock $\overline{\text{M1}}$ High		130	nsec	
$\overline{\text{RFSH}}$	$t_{DL(RF)}$	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock, $\overline{\text{RFSH}}$ Low		180	nsec	$C_L = 30pF$
	$t_{DH(RF)}$	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock, $\overline{\text{RFSH}}$ High		150	nsec	
$\overline{\text{WAIT}}$	$t_{S(WT)}$	$\overline{\text{WAIT}}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{\text{HALT}}$	$t_{D(HT)}$	$\overline{\text{HALT}}$ Delay Time From Falling Edge of Clock		300	nsec	$C_L = 50pF$
$\overline{\text{INT}}$	$t_{s(IT)}$	$\overline{\text{INT}}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{\text{NMI}}$	$t_w(\overline{\text{NML}})$	Pulse Width, $\overline{\text{NMI}}$ Low	80		nsec	
$\overline{\text{BUSRQ}}$	$t_{s(BQ)}$	$\overline{\text{BUSRQ}}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{\text{BUSAK}}$	$t_{DL(BA)}$	$\overline{\text{BUSAK}}$ Delay From Rising Edge of Clock, $\overline{\text{BUSAK}}$ Low		120	nsec	$C_L = 50\ pF$
	$t_{DH(BA)}$	$\overline{\text{BUSAK}}$ Delay From Falling Edge of Clock, $\overline{\text{BUSAK}}$ High		110	nsec	
$\overline{\text{RESET}}$	$t_{s(RS)}$	$\overline{\text{RESET}}$ Setup Time to Rising Edge of Clock	90		nsec	
	$t_{F(C)}$	Delay to/from Float ($\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$)		100	nsec	
	t_{mr}	$\overline{\text{M1}}$ Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec	

[1] $t_{acm} = t_w\ (\Phi H) + t_f - 75$

[2] $t_{aci} = t_c - 80$

[3] $t_{ca} = t_w\ (\Phi L) + t_r - 40$

[4] $t_{caf} = t_w\ (\Phi L) + t_r - 60$

[5] $t_{dcm} = t_c - 210$

[6] $t_{dci} = t_w\ (\Phi L) + t_r - 210$

[7] $t_{cdf} = t_w\ (\Phi L) + t_r - 80$

[8] $t_w\ (\overline{MRL}) = t_c - 40$

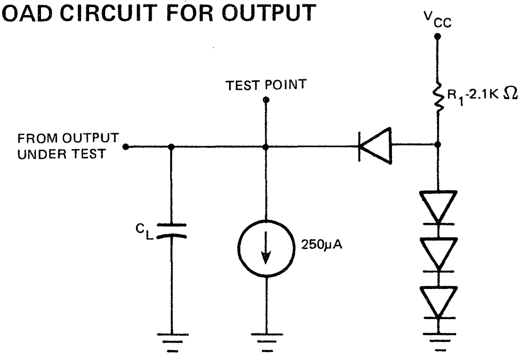
[9] $t_w\ (\overline{MRH}) = t_w\ (\Phi H) + t_f - 30$

[10] $t_w\ (\overline{WR}) = t_c - 40$

[11] $t_{mr} = 2\, t_c + t_w\ (\Phi H) + t_f - 80$

[12] $t_c = t_w\ (\Phi H) + t_w\ (\Phi L) + t_r + t_f$

LOAD CIRCUIT FOR OUTPUT



NOTES (Cont'd.)

C. Output Delay vs. Load Capacitance
$T_A = 70^{\circ}C\ V_{CC} = 5V \pm 5\%$
Add 10 nsec delay for each 50pF increase in load up to a maximum of 200pF for the data bus and 100pF for address and control lines.

D. Although static by design, testing guarantees $t_w\ (\Phi H)$ of 200 μsec maximum.

MK 3880-4 Z80A-CPU

A. C. CHARACTERISTICS T_A = 0°C to 70°C, Vcc = +5V ±5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITIONS
Φ	t_c	Clock Period	.25	[12]	μsec	
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	110	(D)	nsec	
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	110	2000	nsec	
	$t_{r,\,f}$	Clock Rise and Fall Time		30	nsec	
$A_{0\text{-}15}$	$t_{D(AD)}$	Address Output Delay		110	nsec	
	$t_{F(AD)}$	Delay to Float		90	nsec	
	t_{acm}	Address Stable Prior to $\overline{\text{MREQ}}$ (Memory Cycle)	[1]		nsec	C_L = 50pF
	t_{aci}	Address Stable Prior to $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ or $\overline{\text{WR}}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable From $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{IORQ}}$ or $\overline{\text{MREQ}}$	[3]		nsec	Except T3.M1
	t_{caf}	Address Stable From $\overline{\text{RD}}$ or $\overline{\text{WR}}$ During Float	[4]		nsec	
$D_{0\text{-}7}$	$t_{D(D)}$	Data Output Delay		150	nsec	
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	* $t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge at Clock During M2 to M5	60		nsec	C_L = 50pF
	t_{dcm}	Data Stable Prior to $\overline{\text{WR}}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{\text{WR}}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{\text{WR}}$	[7]		nsec	
	t_H	Input Hold Time	0		nsec	
$\overline{\text{MREQ}}$	* $t_{DL\overline{\Phi}(MR)}$	$\overline{\text{MREQ}}$ Delay From Falling Edge of Clock, $\overline{\text{MREQ}}$ Low	20	85	nsec	
	$t_{DH\Phi(MR)}$	$\overline{\text{MREQ}}$ Delay From Rising Edge of Clock, $\overline{\text{MREQ}}$ High		85	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{\text{MREQ}}$ Delay From Falling Edge of Clock, $\overline{\text{MREQ}}$ High		85	nsec	C_L = 50pF
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{\text{MREQ}}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{\text{MREQ}}$ High	[9]		nsec	
$\overline{\text{IORQ}}$	$t_{DL\Phi(IR)}$	$\overline{\text{IORQ}}$ Delay From Rising Edge of Clock, $\overline{\text{IORQ}}$ Low		75	nsec	
	$t_{DL\overline{\Phi}(IR)}$	$\overline{\text{IORQ}}$ Delay From Falling Edge of Clock, $\overline{\text{IORQ}}$ Low		85	nsec	C_L = 50pF
	$t_{DH\Phi(IR)}$	$\overline{\text{IORQ}}$ Delay From Rising Edge of Clock, $\overline{\text{IORQ}}$ High		85	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{\text{IORQ}}$ Delay From Falling Edge of Clock, $\overline{\text{IORQ}}$ High		85	nsec	
$\overline{\text{RD}}$	$t_{DL\Phi(RD)}$	$\overline{\text{RD}}$ Delay From Rising Edge of Clock, $\overline{\text{RD}}$ Low		85	nsec	
	$t_{DL\overline{\Phi}(RD)}$	$\overline{\text{RD}}$ Delay From Falling Edge of Clock, $\overline{\text{RD}}$ Low		95	nsec	C_L = 50pF
	$t_{DH\Phi(RD)}$	$\overline{\text{RD}}$ Delay From Rising Edge of Clock, $\overline{\text{RD}}$ High		85	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{\text{RD}}$ Delay From Falling Edge of Clock, $\overline{\text{RD}}$ High		85	nsec	
$\overline{\text{WR}}$	$t_{DL\Phi(WR)}$	$\overline{\text{WR}}$ Delay From Rising Edge of Clock, $\overline{\text{WR}}$ Low		65	nsec	
	$t_{DL\overline{\Phi}(WR)}$	$\overline{\text{WR}}$ Delay From Falling Edge of Clock, $\overline{\text{WR}}$ Low		80	nsec	C_L = 50pF
	$t_{DH\Phi(WR)}$	$\overline{\text{WR}}$ Delay From Falling Edge of Clock, $\overline{\text{WR}}$ High		80	nsec	
	$t_{w(\overline{WR}L)}$	Pulse Width, $\overline{\text{WR}}$ Low	[10]		nsec	

NOTES:

A Data should be enabled onto the CPU data bus when $\overline{\text{RD}}$ is active. During interrupt acknowledge data should be enabled when M1 and $\overline{\text{IORQ}}$ are both active.

B The $\overline{\text{RESET}}$ signal must be active for a minimum of 3 clock cycles.

(Cont'd. on page 83)

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNIT	TEST CONDITION
$\overline{\text{M1}}$	$t_{DL(M1)}$	$\overline{\text{M1}}$ Delay From Rising Edge of Clock $\overline{\text{M1}}$ Low		100	nsec	C_L = 50pF
	$t_{DH(M1)}$	$\overline{\text{M1}}$ Delay From Rising Edge of Clock, $\overline{\text{M1}}$ High		100	nsec	
$\overline{\text{RFSH}}$	$t_{DL(RF)}$	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock, $\overline{\text{RFSH}}$ Low		130	nsec	C_L = 50pF
	$t_{DH(RF)}$	$\overline{\text{RFSH}}$ Delay From Rising Edge of Clock $\overline{\text{RFSH}}$ High		120	nsec	
$\overline{\text{WAIT}}$	$t_{S(WT)}$	$\overline{\text{WAIT}}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{\text{HALT}}$	$t_{D(HT)}$	$\overline{\text{HALT}}$ Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
$\overline{\text{INT}}$	$t_{s(IT)}$	$\overline{\text{INT}}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{\text{NMI}}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{\text{NMI}}$ Low	80		nsec	
$\overline{\text{BUSRQ}}$	$t_{s(BQ)}$	$\overline{\text{BUSRQ}}$ Setup Time to Rising Edge of Clock	50		nsec	
$\overline{\text{BUSAK}}$	$t_{DL(BA)}$	$\overline{\text{BUSAK}}$ Delay From Rising Edge of Clock, $\overline{\text{BUSAK}}$ Low		100	nsec	C_L = 50pF
	$t_{DH(BA)}$	$\overline{\text{BUSAK}}$ Delay From Falling Edge of Clock, $\overline{\text{BUSAK}}$ High		100	nsec	
$\overline{\text{RESET}}$	$t_{s(RS)}$	$\overline{\text{RESET}}$ Setup Time to Rising Edge of Clock	60		nsec	
	$t_{F(C)}$	Delay to/From Float ($\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ and $\overline{\text{WR}}$)		80	nsec	
	t_{mr}	$\overline{\text{M1}}$ Stable Prior to $\overline{\text{IORQ}}$ (Interrupt Ack.)	[11]		nsec	

[1] $t_{acm} = t_w\ (\Phi H) + t_f - 65$

[2] $t_{aci} = t_c - 70$

[3] $t_{ca} = t_w\ (\Phi L) + t_r - 50$

[4] $t_{caf} = t_w\ (\Phi L) + t_r - 45$

[5] $t_{dcm} = t_c - 170$

[6] $t_{dci} = t_w\ (\Phi L) + t_r - 170$

[7] $t_{cdf} = t_w\ (\Phi L) + t_r - 70$

[8] $t_w\ (\overline{MRL}) = t_c - 30$

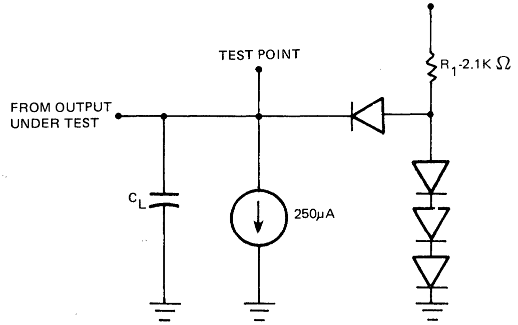
[9] $t_w\ (\overline{MRH}) = t_w\ (\Phi H) + t_f - 20$

[10] $t_w\ (\overline{WR}) = t_c - 30$

[11] $t_{mr} = 2t_c + t_w\ (\Phi H) + t_f - 65$

[12] $t_c = t_w\ (\Phi H) + t_w(\Phi L) + t_r + t_f$

LOAD CIRCUIT FOR OUTPUT



NOTES (Cont'd.)

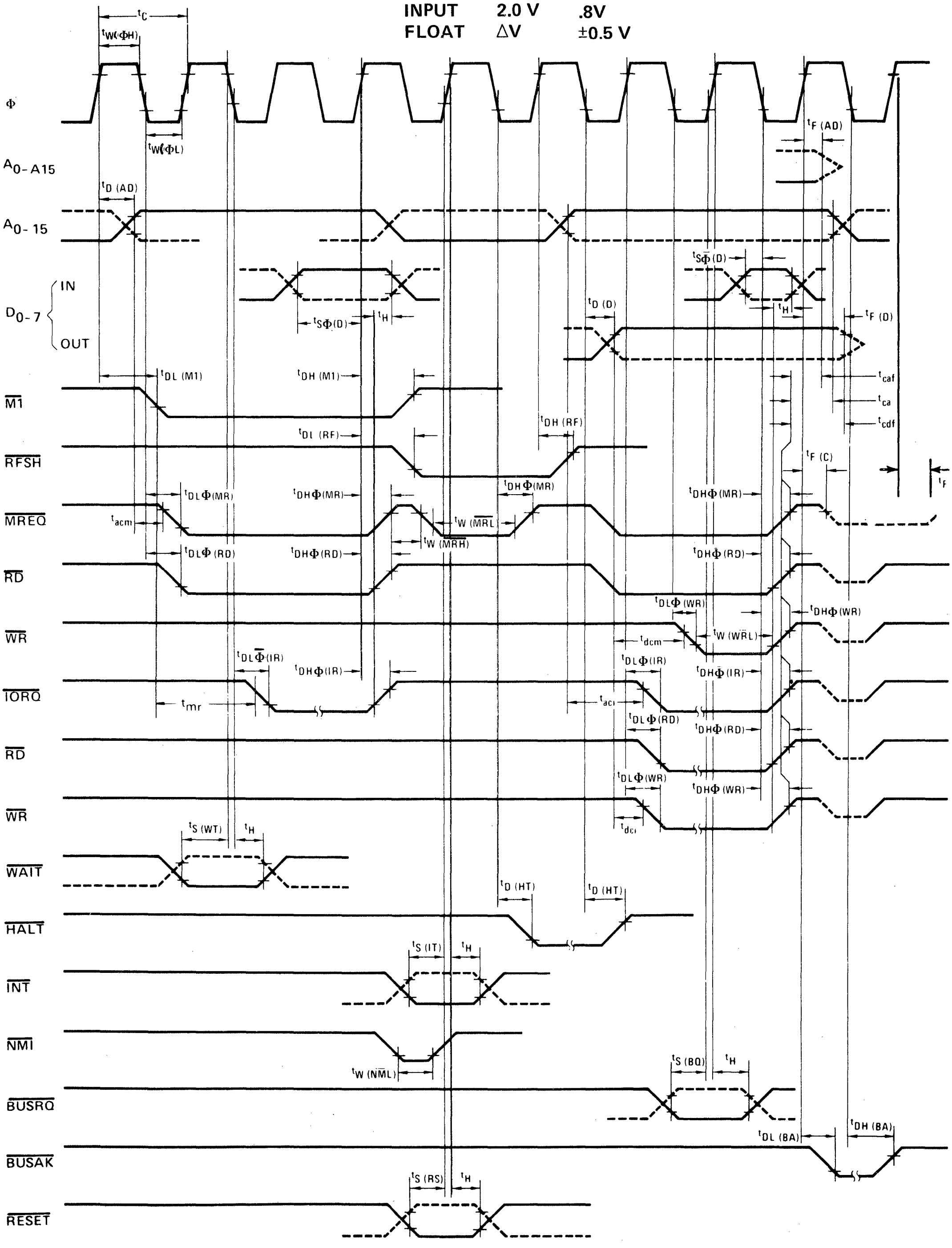
C. Output Delay vs. Load Capacitance
$T_A = 70^\circ C$ $V_{CC} = 5V \pm 5\%$
Add 10 nsec delay for each 50pF increase in load up to a maximum of 200pF for the data bus and 100pF for address and control lines

D. Although static by design, testing guarantees $t_W\ (\Phi H)$ of 200 μsec maximum.

A.C. TIMING DIAGRAM

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	$V_{CC}-.6$	.8V
OUTPUT	2.0 V	.8V
INPUT	2.0 V	.8V
FLOAT	ΔV	±0.5 V



12.0 Z80 INSTRUCTION BREAKDOWN BY MACHINE CYCLE

This section tabulates each Z80 instruction type and breaks each instruction down into its machine cycles and corresponding T States. The different standard machine cycles (OP Code Fetch, Memory Read, Port Read, etc.) are described in Section 4.0 of this manual. This chart will allow the system designer to predict what the Z80 will do on each clock cycle during the execution of a given instruction. The instruction types are listed together by functions and in the same order as the Tables in Section 7.

The best way to learn how to use these tables is to look at a few examples. The first example is to register exchange instructions (LD r, s) where r,s can be any of the following CPU Registers: B,C,D,E,H,L, or A. The instruction breakdown table shows this instruction to have one machine cycle (M1) four T-States long (number in parenthesis) which is an OP Code Fetch. Referring to Figure 4.0-1 one sees the standard form for an OP Code Fetch and the state of the CPU bus during these four T-States. Taking the next instruction shown (LD r, n) which loads one of the previous registers with data or immediate value "n" one finds the breakdown to be a four T-State OP Code Fetch followed by a three T-State Operand Data Read. An Operand Data Read takes the form of the Standard Memory Read shown in Figure 4.0-2.

After these two simple examples, a more complex one is in order. The LD r, (IX+d) is the first double byte OP Code shown and executes as follows: First there are two M1 cycles (and related memory refreshes) followed by an Operand Data Read of the displacement "d". Next M3 consists of a five T-State Internal Operation which is the calculation of the Indexed address (IX+d). The last machine cycle (M4) consists of a Memory Read of the data continued in address IX+d and the loading of register "r" with that data.

Z80 Family

The LD dd, (nn) instruction loads an internal 16-bit register pair with the contents of the memory location specified in the Operand Bytes of the instruction. This instruction is four bytes long (two bytes of OP Code + two bytes of Operand Address). As shown, there are two M1 cycles to fetch the OP Code and then two Machine Cycles to read the Operand Addresses, low order byte first. Machine cycle 4 is a read of memory to obtain the data for the low order register (e.g., C of BC, E of DE and L of HL) followed by a read of the data for the high order register.

The first instruction to use the Stack Register is the PUSH qq instruction which executes as follows: Machine cycle 1 is extended by one cycle and the Stack Pointer is decremented in the extra T-State to point to an empty location on the Stack. Machine cycle 2 is a write of the high byte of the referenced register to the address contained in the Stack Pointer. The Stack Pointer is again decremented and a write of the low byte of the referenced register is made to the Stack in Machine Cycle 3. Note that the Stack Pointer is left pointing to the last data referenced on the Stack. The block transfer instructions such as LDI and LDIR are very similar. LDI is 16 T-States long and is composed of a double byte OP Code Fetch (two memory refreshes) followed by a memory read and a memory write. The memory write is 5 T-States long to allow updating of the block length counter —BC. The repetitive form of this instruction (LDIR) has an additional Machine Cycle (M4) of 5 T-States to allow decrementing of the Program Counter by two (PC-2) which results in refetching of the OP Code (LDIR). Each movement of data by this instruction is 21 T-States long (except the last) and the refetching of the OP Codes results in memory refresh occurring as well as the sampling of interrupts and $\overline{\text{BUSRQ}}$.

The $\overline{\text{NMI}}$ Interrupt sequence is 11 T-States long with the first M1 being a dummy OP Code Fetch of 5 T-States long. The Program Counter is not advanced, the OP Code on the data bus is ignored and an internal Restart is done to address 66H. The following two Machine Cycles are a write of the Program Counter to the Stack.

The $\overline{\text{INT}}$ Mode 0 is the 8080A mode and requires the user to place an instruction on the data bus for the CPU to execute. If a RST instruction is used, the CPU stacks the Program Counter and begins execution at the Restart Address. If a CALL instruction is used, the CALL Op Code is placed on the data bus during the INTA cycle (M1). M2 and M3 are normal Memory Read cycles (not INTA cycles) of the CALL addresses (low byte first). Program Counter is stacked in M4 and M5.

Mode 2 is used by the Z80 System Peripherals and operates as follows: During the INTA cycle (M1) a Vector is sent in from the highest priority interrupting device. M2 and M3 are used to Stack the Program Counter. The Vector (low byte) and an internal Interrupt Register (I) from a pointer to a table containing the addresses of Interrupt Service Routines. During M4 and M5 the Service Routines address is read from this table into the CPU. The next M1 cycle will fetch an OP Code from the address received is M4 and M5.

LEGEND

IO	— Internal CPU Operation
MR	— Memory Read
MRH	— Memory Read of High Byte
MRL	— Memory Read of Low Byte
MW	— Memory Write
MWH	— Memory Write of High Byte
MWL	— Memory Write of Low Byte
OCF	— Op Code Fetch
ODH	— Operand Data Read of High Byte
ODL	— Operand Data Read of Low Byte
PR	— Port Read
PW	— Port Write
SRH	— Stack Read of High Byte
SRL	— Stack Read of Low Byte
SWH	— Stack Write of High Byte
SWL	— Stack Write of Low Byte
( )	— Number of T-States in that Machine Cycle

Z80 INSTRUCTION BREAKDOWN BY MACHINE CODE

INSTRUCTION TYPE	BYTES	MACHINE CYCLE M1	M2	M3	M4	M5
LD r, s	1	OCF (4)				
LD r, n	2	OCF (4)	OD (3)			
LD r, (HL)	1	OCF (4)	MR (3)			
LD (HL), r		OCF (4)	MW (3)			
LD r, (IX+d)	3	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (3)	
LD (IX+d), r		OCF (4)/OCF (4)	OD (3)	IO (5)	MW (3)	
LD (HL), n	2	OCF (4)	OD (3)	MW (3)		
LD A, ($\overset{BC}{DE}$)	1	OCF (4)	MR (3)			
LD ($^{BC}_{DE}$), A		OCF (4)	MW (3)			
LD A, (nn)	3	OCF (4)	ODL (3)	ODH (3)	MR (3)	
LD (nn) , A		OCF (4)	ODL (3)	ODH (3)	MW (3)	
LD A,$^{I}_{R}$	2	OCF (4)/OCF(5)				
LD$^{I}_{R}$, A						
LD dd, nn	3	OCF (4)	ODL (3)	ODH (3)		
LD IX, nn	4	OCF (4)/OCF (4)	ODL (3)	ODH (3)		
LD HL, (nn)	3	OCF (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)
LD (nn), HL		OCF (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)
LD dd, (nn)	4	OCF (4)/OCF (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)
LD (nn), dd		OCF (4)/OCF (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)
LD IX, (nn)		OCF (4)/OCF (4)	ODL (3)	ODH (3)	MRL (3)	MRH (3)
LD (nn), IX		OCF (4)/OCF (4)	ODL (3)	ODH (3)	MWL (3)	MWH (3)
LD SP, HL	1	OCF (6)				
LD SP, IX	2	OCF (4)/OCF (6)				
PUSH qq	1	OCF (5) SP-1 →	SWH (3) SP-1 →	SWL (3)		
PUSH IX	2	OCF (4)/OCF (5) SP-1 →	SWH (3) SP-1 →	SWL (3)		
POP qq	1	OCF (4)	SRH (3) SP+1 →	SRL (3)	SP+1 →	
POP IX	2	OCF (4)/OCF (4)	SRH (3) SP+1 →	SRL (3)	SP+1 →	
EX DE, HL	1	OCF (4)				
EX AF, AF′	1	OCF (4)				

INSTRUCTION TYPE	BYTES	M1	M2	M3	M4	M5
				MACHINE CYCLE		
EXX	1	OCF (4)				
EX (SP), HL	1	OCF (4)	SRL (3) SP+1 →	SRH (4)	SWH (3) SP-1 →	SWL (5)
EX (SP), IX	2	OCF (4)/OCF (4)	SRL (3) SP+1 →	SRH (4)	SWH (3) SP-1 →	SWL (5)
LDI LDD CPI CPD	2	OCF (4)/OCF (4)	MR (3)	MW (5)		
LDIR LDDR CPIR CPDR	2	OCF (4)/OCF (4)	MR (3)	MW (5)	IO (5)* *only if BC ≠ 0	
ALU A, r ADD ADC SUB SBC AND OR XOR CP	1	OCF (4)				
ALU A, n	2	OCF (4)	OD (3)			
ALU A, (HL)	1	OCF (4)	MR (3)			
ALU A, (IX+d)	3	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (3)	
DEC INC r	1	OCF (4)				
DEC INC (HL)	1	OCF (4)	MR (4)	MW (3)		
DEC INC (IX+D)	2	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	MW (3)
DAA CPL CCF SCF NOP HALT DI EI	1	OCF (4)				
NEG IMO IM1 IM2	2	OCF (4)/OCF (4)				

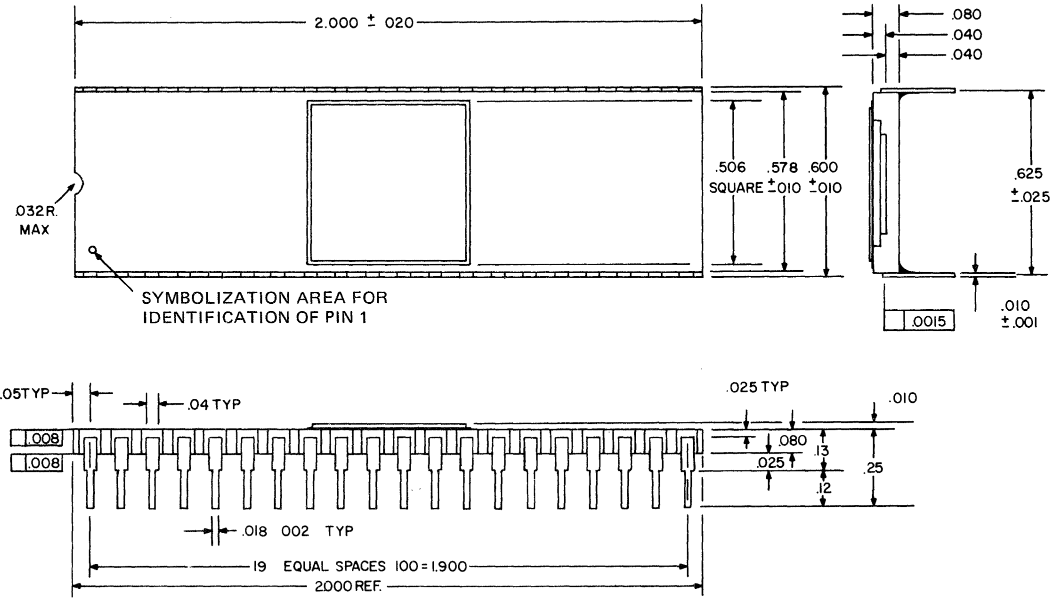
		MACHINE CYCLE				
INSTRUCTION TYPE	BYTES	M1	M2	M3	M4	M5
ADD HL, ss	1	OCF (4)	IO (4)	IO (3)		
ADC HL, ss SBC HL, ss ADD IX, pp	2	OCF (4)/OCF (4)	IO (4)	IO (3)		
INC ss DEC ss	1	OCF (6)				
DEC IX INC IX	2	OCF (4)/OCF (6)				
RLCA RLA RRCA RRA	1	OCF (4)				
RLC r RL RRC RR SLA SRA SRL	2	OCF (4)/OCF (4)				
RLC (HL) RL RRC RR SLA SRA SRL	2	OCF (4)/OCF (4)	MR (4)	MW (3)		
RLC (IX+d) RL RRC RR SLA SRA SRL	4	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	MW (3)
RLD RRD	2	OCF (4)/OCF (4)	MR (3)	IO (4)	MW (3)	
BIT b, r SET RES	2	OCF (4)/OCF (4)				

INSTRUCTION TYPE	BYTES	MACHINE CYCLE				
		M1	M2	M3	M4	M5
BIT b, (HL)	2	OCF (4)/OCF (4)	MR (4)			
SET b, (HL) RES	2	OCF (4)/OCF (4)	MR (4)	MW (3)		
BIT b, (IX+d)	4	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	
SET b, (IX+d) RES	4	OCF (4)/OCF (4)	OD (3)	IO (5)	MR (4)	MW (3)
JP nn JP cc, nn	3	OCF (4)	ODL (3)	ODH (3)		
JR e	2	OCF (4)	OD (3)	IO (5)		
JR C, e JR NC, e JR Z, e JR NZ, e	2	OCF (4)	OD (3)	IO (5)* * If condition is met		
JP (HL)	1	OCF (4)				
JP (IX)	2	OCF (4)/OCF (4)				
DJNZ, e	2	OCF (5)	OD (3)	IO (5)* * If B≠ 0		
CALL nn CALL cc, nn cc true	3	OCF (4)	ODL (3)	ODH (4) SP-1 →	SWH (3) SP-1 →	SWL (3)
CALL cc, nn cc false	3	OCF (4)	ODL (3)	ODH (3)		
RET	1	OCF (4)	SRL (3) SP+1 →	SRH (3)	SP+1 →	
RET cc	1	OCF (5)	SRL (3)* * If cc is true SP+1 →	SRH (3)*	SP+1 →	
RETI RETN	2	OCF (4)/OCF (4)	SRL (3) SP+1 →	SRH (3)	SP+1 →	
RST p	1	OCF (5) SP-1 →	SWH (3) SP-1 →	SWL (3)		

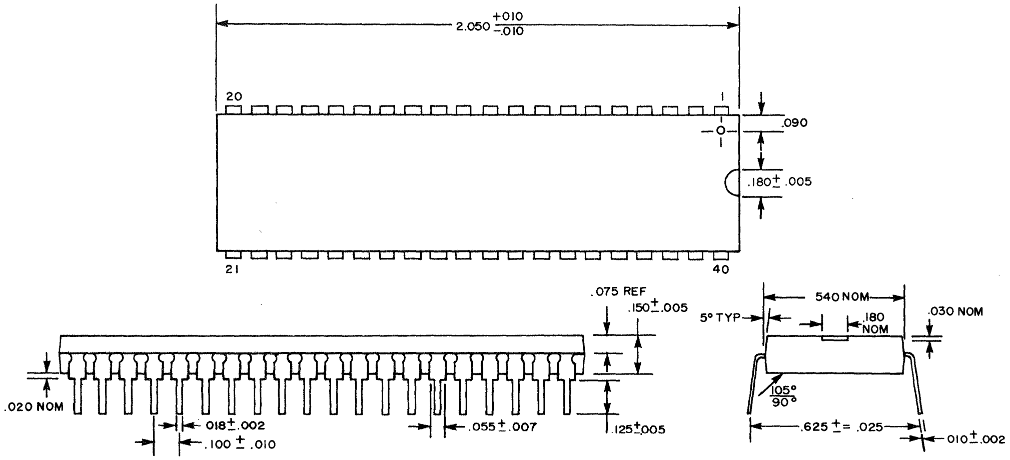
		MACHINE CYCLE				
INSTRUCTION TYPE	BYTES	M1	M2	M3	M4	M5
IN A, (n)	2	OCF (4)	OD (3)	PR (4)		
IN r, (c)	2	OCF (4)/OCF (4)	PR (4)			
INI IND	2	OCF (4)/OCF (5)	PR (4)	MW (3)		
INIR INDR	2	OCF (4)/OCF (5)	PR (4)	MW (3)	IO (5)	
OUT (n) , A	2	OCF (4)	OD (3)	PW (4)		
OUT (C), r	2	OCF (4)/OCF (4)	PW (4)			
OUTI OUTD	2	OCF (4)/OCF (5)	MR (3)	PW (4)		
OTIR OTDR	2	OCF (4)/OCF (5)	MR (3)	PW (4)	IO (5)	
<u>INTERRUPTS</u>						
NMI	–	OCF (5) * SP-1 →	SWH (3) SP-1 →	SWL (3)	*Op Code Ignored	
INT						
MODE 0	–	INTA (6) (CALL INSERTED)	ODL (3)	ODH (4) SP-1 →	SWH (3) SP-1 →	SWL (3)
	–	INTA (6) (RST INSERTED) SP-1 →	SWH (3) SP-1 →	SWL (3)		
MODE 1	–	INTA (7) (RST 38H INTERNAL) SP-1 →	SWH (3) SP-1 →	SWL (3)		
MODE 2	–	INTA (7) (VECTOR SUPPLIED) SP-1 →	SWH (3) SP-1 →	SWL (3)	MRL (3)	MRH (3)

13.0 PACKAGE DESCRIPTION AND ORDERING INFORMATION

PACKAGE DESCRIPTION – 40 Pin Dual-In-Line Ceramic Package



PACKAGE DESCRIPTION – 40-Pin Dual-In-Line Plastic Package



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	MAX CLOCK FREQUENCY	TEMPERATURE RANGE
MK3880N Z80-CPU	Plastic	2.5 MHz	0° to + 70° C
MK3880P Z80-CPU	Ceramic	2.5 MHz	
MK3880N-4 Z80-CPU	Plastic	4.0 MHz	
MK3880P-4 Z80A-CPU	Ceramic	4.0 MHz	
MK3880P-10 Z80-CPU	Ceramic	2.5 MHz	-40°C to +85°C
MK3880P-20 Z80-CPU	Ceramic	2.5 MHz	-55° C to +125° C

MK3881 PARALLEL I/O CONTROLLER

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

The Z80 Parallel I/O Circuit is a programmable, two port device which provides a TTL compatible interface between peripheral devices and the Z80-CPU. The CPU can configure the Z80-PIO to interface with a wide range of peripheral devices with no other external logic required. Typical peripheral devices that are fully compatible with the Z80-PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc. The Z80-PIO utilizes N channel silicon gate depletion load technology and is packaged in a 40 pin DIP. Major features of the Z80-PIO include:

- Two independent 8 bit bidirectional peripheral interface ports with 'handshake' data transfer control
- Interrupt driven 'handshake' for fast response
- Any one of four distinct modes of operation may be selected for a port including:

  Byte output
  Byte input
  Byte bidirectional bus (Available on Port A only)
  Bit control mode

  All with interrupt controlled handshake
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic
- Eight outputs are capable of driving Darlington transistors
- All inputs and outputs fully TTL compatible
- Single 5 volt supply and single phase clock required.

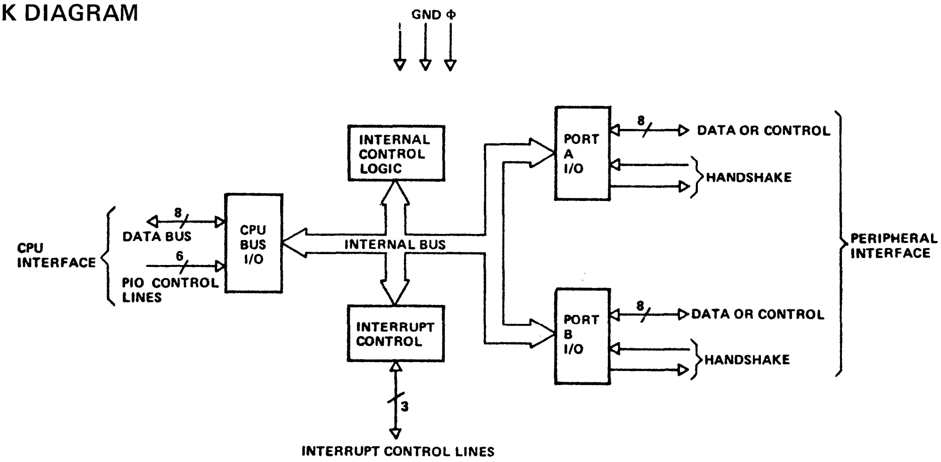
One of the unique features of the Z80-PIO that separates it from other interface controllers is that all data transfer between the peripheral device and the CPU is accomplished under total interrupt control. The interrupt logic of the PIO permits full usage of the efficient interrupt capabilities of the Z80-CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO so that additional circuits are not required. Another unique feature of the PIO is that it can be programmed to interrupt the CPU on the occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the amount of time that the processor must spend in polling peripheral status.

Z80
Family

2.0 PIO ARCHITECHTURE

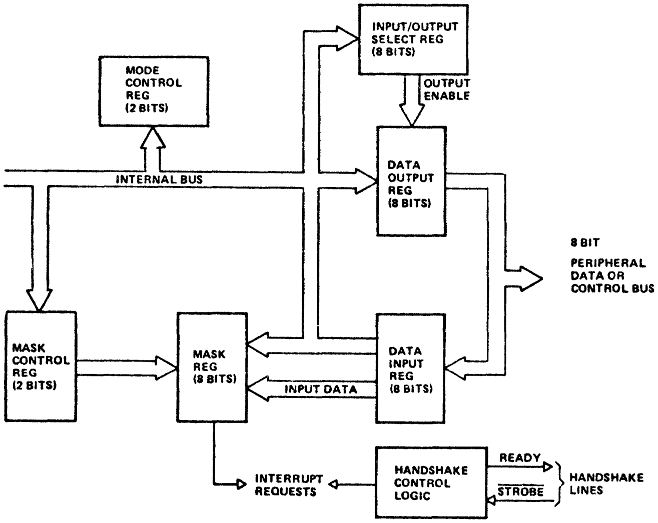
A block diagram of the Z80-PIO is shown in figure 2.0-1. The internal structure of the Z80-PIO consists of a Z80-CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic. The CPU bus interface logic allows the PIO to interface directly to the Z80-CPU with no other external logic. However, address decoders and/or line buffers may be required for large systems. The internal control logic synchronizes the CPU data bus to the peripheral device interfaces (Port A and Port B). The two I/O ports (A and B are virtually identical and are used to interface directly to peripheral devices.

PIO BLOCK DIAGRAM
Figure 2.0-1



The Port I/O logic is composed of 6 registers with "handshake" control logic as shown in figure 2.0-2. The registers include: an 8 bit data input register, an 8 bit data output register, a 2 bit mode control register, an 8 bit mask register, an 8 bit input/output select register, and a 2 bit mask control register.

PORT I/O BLOCK DIAGRAM
Figure 2.0-2



The 2-bit mode control register is loaded by the CPU to select the desired operating mode (byte output, byte input, byte bidirectional bus, or bit control mode). All data transfer between the peripheral device and the CPU is achieved through the data input and data output registers. Data may be written into the output register by the CPU or read back to the CPU from the input register at any time. The handshake lines associated with each port are used to control the data transfer between the PIO and the peripheral device.

The 8-bit mask register and the 8-bit input/output select register are used only in the bit control mode. In this mode any of the 8 peripheral data or control bus pins can be programmed to be an input or an output as specified by the select register. The mask register is used in this mode in conjunction with a special interrupt feature. This feature allows an interrupt to be generated when any or all of the unmasked pins reach a specified state (either high or low). The 2-bit mask control register specifies the active state desired (high or low) and if the interrupt should be generated when all unmasked pins are active (AND condition) or when any unmasked pin is active (OR condition). This feature reduces the requirement for CPU status checking of the peripheral by allowing an interrupt to be automatically generated on specific peripheral status conditions. For example, in a system with 3 alarm conditions, an interrupt may be generated if any one occurs or if all three occur.

The interrupt control logic section handles all CPU interrupt protocol for nested priority interrupt structures. The priority of any device is determined by its physical location in a daisy chain configuration. Two lines are provided in each PIO to form this daisy chain. The device closest to the CPU has the highest priority. Within a PIO, Port A interrupts have higher priority than those of Port B. In the byte input, byte output or bidirectional modes, an interrupt can be generated whenever a new byte transfer is requested by the peripheral. In the bit control mode an interrupt can be generated when the peripheral status matches a programmed value. The PIO provides for complete control of nested interrupts. That is, lower priority devices may not interrupt higher priority devices that have not had their interrupt service routine completed by the CPU. Higher priority devices may interrupt the servicing of lower priority devices.

When an interrupt is accepted by the CPU in mode 2, the interrupting device must provide an 8-bit interrupt vector for the CPU. This vector is used to form a pointer to a location in the computer memory where the address of the interrupt service routine is located. The 8-bit vector from the interrupting device forms the least significant 8 bits of the indirect pointer while the I Register in the CPU provides the most significant 8 bits of the pointer. Each port (A and B) has an independent interrupt vector. The least significant bit of the vector is automatically set to a 0 within the PIO since the pointer must point to two adjacent memory locations for a complete 16-bit address.

The PIO decodes the RETI (Return from interrupt) instruction directly from the CPU data bus so that each PIO in the system knows at all times whether it is being serviced by the CPU interrupt service routine without any other communication with the CPU.

3.0 PIN DESCRIPTION

A diagram of the Z80-PIO pin configuration is shown in figure 3.0-1. This section describes the function of each pin.

D_7-D_0

Z80-CPU Data Bus (bidirectional, tristate)
This bus is used to transfer all data and commands between the Z80-CPU and the Z80-PIO. D_0 is the least significant bit of the bus.

B/A Sel

Port B or A Select (input, active high)
This pin defines which port will be accessed during a data transfer between the Z80-CPU and the Z80-PIO. A low level on this pin selects Port A while a high level selects Port B. Often Address bit A_0 from the CPU will be used for this selection function.

C/D Sel

Control or Data Select (input, active high)
This pin defines the type of data transfer to be performed bwtween the CPU and the PIO. A high level on this pin during a CPU write to the PIO causes the Z80 data bus to be interpreted as a command for the port selected by the B/A Select line. A low level on this pin means that the Z80 data bus is being used to transfer data between the CPU and the PIO. Often Address bit A_1 from the CPU will be used for this function.

$\overline{CE}$

Chip Enable (input, active low)
A low level on this pin enables the PIO to accept command or data inputs from the CPU during a write cycle or to transmit data to the CPU during a read cycle. This signal is generally a decode of four I/O port numbers that encompass port A and B, data and control.

Φ

System Clock(input)
The Z80-PIO uses the standard Z80 system clock to synchronize certain signals internally. This is a single phase clock.

$\overline{M1}$

Machine Cycle One Signal from CPU (input, active low)
This signal from the CPU is used as a sync pulse to control several internal PIO operations. When M1 is active and the RD signal is active, the Z80-CPU is fetching an instruction from memory. Conversely, when M1 is active and IORQ is active, the CPU is acknowledging an interrupt. In addition, the M1 signal has two other functions within the Z80-PIO.

1. M1 synchronizes the PIO interrupt logic.
2. When M1 occurs without an active RD or IORQ signal the PIO logic enters a reset state.

$\overline{IORQ}$

Input/Output Request from Z80-CPU (input, active low)
The $\overline{IORQ}$ signal is used in conjunction with the B/A Select, C/D Select, $\overline{CE}$, and $\overline{RD}$ signals to transfer commands and data between the Z80-CPU and the Z80-PIO. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are active, the port addressed by B/A will transfer data to the CPU (a read operation). Conversely, when $\overline{CE}$ and $\overline{IORQ}$ are active but $\overline{RD}$ is not active, then the port addressed by B/A will be written into from the CPU with either data or control information as specified by the C/D Select signal. Also, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the interrupting port will automatically place its interrupt vector on the CPU data bus if it is the highest device requesting an interrupt.

$\overline{RD}$

Read Cycle Status from the Z80-CPU (input, active low)
If $\overline{RD}$ is active a MEMORY READ or I/O READ operation is in progress. The $\overline{RD}$ signal is used with B/A Select, C/D Select, $\overline{CE}$ and $\overline{IORQ}$ signals to transfer data from the Z80-PIO to the Z80-CPU.

IEI

Interrupt Enable In (input, active high)
This signal is used to form a priority interrupt daisy chain when more than one interrupt driven device is being used. A high level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.

IEO

Interrupt Enable Out (output, active high)
The IEO signal is the other signal required to form a daisy chain priority scheme. It is high only if IEI is high and the CPU is not servicing an interrupt from this PIO. Thus this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$

Interrupt Request (output, open drain, active low)
When INT is active the Z80-PIO is requesting an interrupt from the Z80-CPU.

A_0-A_7

Port A Bus (bidirectional, tri-state)
This 8 bit bus is used to transfer data and/or status or control information between Port A of the Z80-PIO and a peripheral device. A_0 is the least significant bit of the Port A data bus.

$\overline{A\ STB}$

Port A Strobe Pulse from Peripheral Device (input, active low)
The meaning of this signal depends on the mode of operation selected for Port A as follows:

1) Output mode: The positive edge of this strobe is issued by the peripheral to acknowledge the receipt of data made available by the PIO.

2) Input mode: The strobe is issued by the peripheral to load data from the peripheral into the Port A input register. Data is loaded into the PIO when this signal is active.

3) Bidirectional mode: When this signal is active, data from the Port A output register is gated onto Port A bidirectional data bus. The positive edge of the strobe acknowledges the receipt of the data.

4) Control mode: The strobe is inhibited internally.

A RDY

Register A Ready (output, active high)
The meaning of this signal depends on the mode of operation selected for Port A as follows:

1) Output mode: This signal goes active to indicate that the Port A output register has been loaded and the peripheral data bus is stable and ready for transfer to the peripheral device.

2) Input mode: This signal is active when the Port A input register is empty and is ready to accept data from the peripheral device.

3) Bidirectional mode: This signal is active when data is available in Port A output register for transfer to the peripheral device. In this mode data is not placed on the Port A data bus unless $\overline{A\ STB}$ is active.

4) Control mode: This signal is disabled and forced to a low state.

B_0-B_7

Port B Bus (bidirectional, tristate)
This 8 bit bus is used to transfer data and/or status or control information between Port B of the PIO and a peripheral device. The Port B data bus is capable of supplying 1.5ma@ 1.5V to drive Darlington transistors. B_0 is the least significant bit of the bus.

$\overline{\text{B STB}}$

Port B Strobe Pulse from Peripheral Device (input, active low)
The meaning of this signal is similar to that of $\overline{\text{A STB}}$ with the following exception:

In the Port A bidirectional mode this signal strobes data from the peripheral device into the Port A input register.

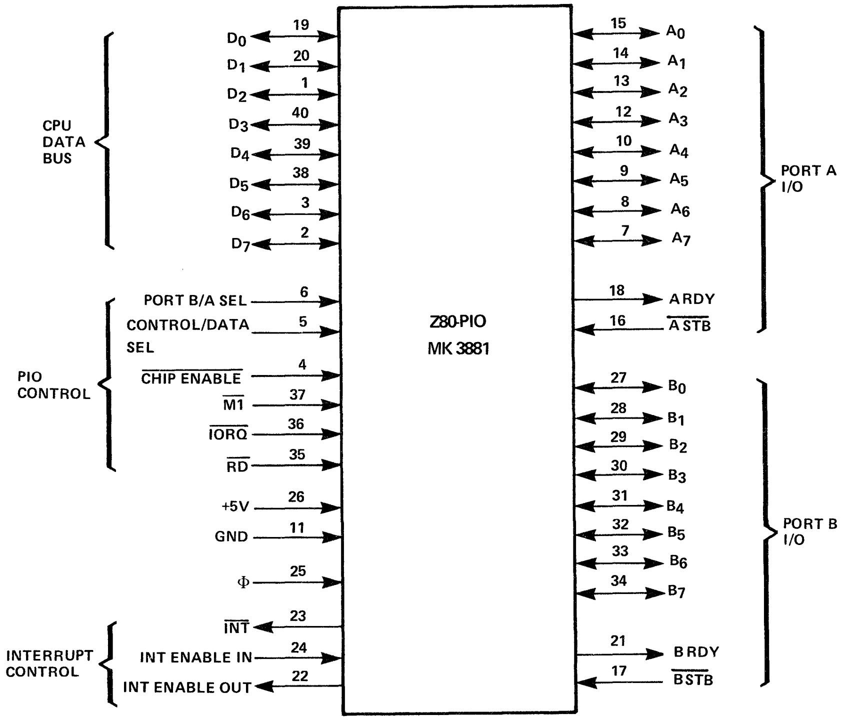
B RDY

Register B Ready (output, active high)
The meaning of this signal is similar to that of A Ready with the following exception:

In the Port A bidirectional mode this signal is high when the Port A input register is empty and ready to accept data from the peripheral device.

PIO PIN CONFIGURATION

Figure 3.0-1



4.0 PROGRAMMING THE PIO

4.1 RESET

The Z80-PIO automatically enters a reset state when power is applied. The reset state performs the following functions:

1) Both port mask registers are reset to inhibit all port data bits.

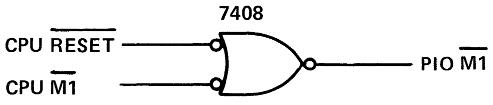
2) Port data bus lines are set to a high impedance state and the Ready "handshake" signals are inactive (low). Mode 1 is automatically selected.

3) The vector address registers are not reset.

4) Both port interrupt enable flip flops are reset.

5) Both port output registers are reset.

In addition to the automatic power on reset, the PIO can be reset by applying an $\overline{M1}$ signal without the presence of a $\overline{RD}$ or $\overline{IORQ}$ signal. If no $\overline{RD}$ or $\overline{IORQ}$ is detected during M1 the PIO will enter the reset state immediately after the $\overline{M1}$ signal goes inactive. The purpose of this reset is to allow a single external gate to generate a reset without a power down sequence. This approach was required due to the 40 pin packaging limitation. It is recommended that in breadboard systems and final systems with a "Reset" push button that a $\overline{M1}$ reset be implemented for the PIO.



A software RESET is possible as described in Section 4.4, however, use of this method during early system debug may not be desirable because of non-functional system hardware (bus buffers or memory for example).

Once the PIO has entered the internal reset state it is held there until the PIO receives a control word from the CPU.

4.2 LOADING THE INTERRUPT VECTOR

The PIO has been designed to operate with the Z80-CPU using the mode 2 interrupt response. This mode requires that an interrupt vector be supplied by the interrupting device. This vector is used by the CPU to form the address for the interrupt service routine of that port. This vector is placed on the Z80 data bus during an interrupt acknowledge cycle by the highest priority device requesting service at that time. (Refer to the Z80-CPU Technical Manual for details on how an interrupt is serviced by the CPU). The desired interrupt vector is loaded into the PIO by writing a control word to the desired port of the PIO with the following format:

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	0

signifies this control word is an interrupt vector

D0 is used in this case as a flag bit which when low causes V7 thru V1 to be loaded into the vector register. At interrupt acknowledge time, the vector of the interrupting port will appear on the Z80 data bus exactly as shown in the format above.

4.3 SELECTING AN OPERATING MODE

Port A of the PIO may be operated in any of four distinct modes: Mode 0 (output mode), Mode 1 (input mode), Mode 2 (bidirectional mode), and Mode 3 (control mode). Note that the mode numbers have been selected for mnemonic significance; i.e. 0=Out, 1=In, 2=Bidirectional. Port B can operate in any of these modes except Mode 2.

The mode of operation must be established by writing a control word to the PIO in the following format:

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

X=unused bit

mode word

signifies mode word to be set

Bits D7 and D6 from the binary code for the desired mode according to the following table:

D7	D6	MODE
0	0	0 (output)
0	1	1 (input)
1	0	2 (bidirectional)
1	1	3 (control)

Bits D5 and D4 are ignored. Bits D3-D0 must be set to 1111 to indicate "Set Mode".

Selecting Mode 0 enables any data written to the port output register by the CPU to be enabled onto the port data bus. The contents of the output register may be changed at any time by the CPU simply by writing a new data word to the port. Also the current contents of the output register may be read back to the Z80-CPU at any time through the execution of an input instruction.

With Mode 0 active, a data write from the CPU causes the Ready handshake line of that port to go high to notify the peripheral that data is available. This signal remains high until a strobe is received from the peripheral. The rising edge of the strobe generates an interrupt (if it has been enabled) and causes the Ready line to go inactive. This very simple handshake is similar to that used in many peripheral devices.

Selecting Mode 1 puts the port into the input mode. To start handshake operation, the CPU merely performs an input read operation from the port. This activates the Ready line to the peripheral to signify that data should be loaded into the empty input register. The peripheral device then strobes data into the port input register using the strobe line. Again, the rising edge of the strobe causes an interrupt request (if it has been enabled) and deactivates the Ready signal. Data may be strobed into the input register regardless of the state of the Ready signal if care is taken to prevent a data overrun condition.

Mode 2 is a bidirectional data transfer mode which uses all four handshake lines. Therefore only Port A may be used for Mode 2 operation. Mode 2 operation uses the Port A handshake signals for output control and the Port B handshake signals for input control. Thus, both A RDY and B RDY may be active simultaneously. The only operational difference between Mode 0 and the output portion of Mode 2 is that data from the Port A output register is allowed on to the port data bus only when $\overline{\text{A STB}}$ is active in order to achieve a bidirectional capability.

Mode 3 operation is intended for status and control applications and does not utilize the handshake signals. When Mode 3 is selected, the next control word sent to the PIO must define which of the port data bus lines are to be inputs and which are outputs. The format of the control word is shown below:

D7	D6	D5	D4	D3	D2	D1	D0
I/O_7	I/O_6	IO/5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

If any bit is set to a one, then the corresponding data bus line will be used as an input. Conversely, if the bit is reset, the line will be used as an output.

During Mode 3 operation the strobe signal is ignored and the Ready line is held low. Data may be written to a port or read from a port by the Z80-CPU at any time during Mode 3 operation. (An exception to this is when Port A is in Mode 2 and Port B is in Mode 3). When reading a port, the data returned to the CPU will be composed of input data from port data bus lines assigned as inputs plus port output register data from those lines assigned as outputs.

4.4 SETTING THE INTERRUPT CONTROL WORD

The interrupt control word for each port has the following format:

D7	D6	D5	D4	D3	D2	D1	D0
Enable Interrupt	AND/ OR	High/ Low	Masks follows	0	1	1	1

used in Mode 3 only (D6–D4) signifies interrupt control word (D3–D0)

If bit D7=1 the interrupt enable flip flop of the port is set and the port may generate an interrupt. If bit D7=0 the enable flag is reset and interrupts may not be generated. If an interrupt occurs while D7=0, it will be latched internally by the PIO and passed onto the CPU when PIO Interrupts are Re-Enabled (D7=1). Bits D6, D5 and D4 are used mainly with Mode 3 operation, however, setting bit D4 of the interrupt control word during any mode of operation will cause a pending interrupt to be reset. These three bits are used to allow for interrupt operation in Mode 3 when any group of the I/O lines go to certain defined states. Bit D6 (AND/OR) defines the logical operation to be performed in port monitoring. If bit D6=1, and AND function is specified and if D6=0, an OR function is specified. For example, if the AND function is specified, all bits must go to a specified state before an interrupt will be generated while the OR function will generate an interrupt if any specified bit goes to the active state.

Bit D5 defines the active polarity of the port data bus line to be monitored. If bit D5=1 the port data lines are monitored for a high state while if D5=0 they will be monitored for a low state.

If bit D4=1 the next control word sent to the PIO must define a mask as follows:

D7	D6	D5	D4	D3	D2	D1	D0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

Only those port lines whose mask bit is zero will be monitored for generating an interrupt.

The interrupt enable flip flop of a port may be set or reset without modifying the rest of the interrupt control word by using the following command:

Int Enable	X	X	X	0	0	1	1

If an external Asynchronous interrupt could occur while the processor is writing the disable word to the PIO (03H) then a system problem may occur. If interrupts are enabled in the processor it is possible that the Asynchronous interrupt will occur while the processor is writing the disable word to the PIO. The PIO will generate an INT and the CPU will acknowledge it, however, by this time, the PIO will have received the disable word and de-activated its interrupt structure. The result is that the PIO will not send in its interrupt vector during the interrupt acknowledge cycle because it is disabled and the CPU will fetch an erroneous vector resulting in a program fault. The cure for this problem is to disable interrupts within the CPU with the DI instruction just before the PIO is disabled and then re-enable interrupts with the EI instruction. This action causes the CPU to ignore any faulty interrupts produced by the PIO while it is being disabled. The code sequence would be:

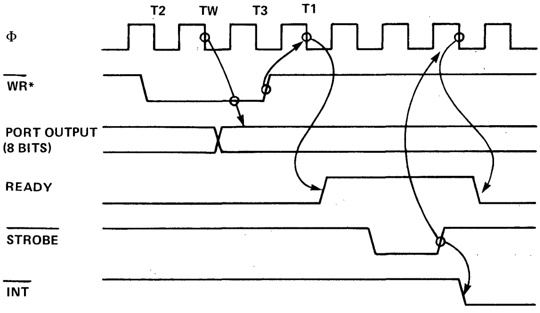
```
.
.
LD A,03H
DI              ; DISABLE CPU
OUT (PIO),A     ; DISABLE PIO
EI              ; ENABLE CPU
.
.
```

5.0 TIMING

5.1 OUTPUT MODE (MODE 0)

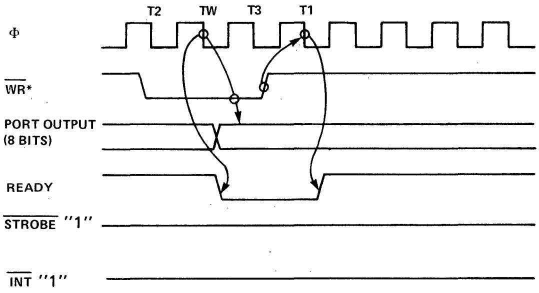
Figure 5.0-1a illustrates the timing associated with Mode 0 operation. An output cycle is always started by the execution of an output instruction by the CPU. A $\overline{\text{WR}}$* pulse is generated by the PIO during a CPU I/O write operation and is used to latch the data from the CPU data bus into addressed port's (A or B) output register. The rising edge of the $\overline{\text{WR}}$* pulse then raises the READY line after the next falling edge of Φ to indicate that data is available for the peripheral device. In most systems, the rising edge of the READY signal can be used as a latching signal in the peripheral device. The READY signal will remain active until a positive edge is received from the $\overline{\text{STROBE}}$ line indicating that the peripheral has taken the data shown in Figure 5.0-1a. If already active, READY will be forced low 1½ Φ cycles after the falling edge of $\overline{\text{IORQ}}$ if the port's output register is written into. READY will return high on the first falling edge of Φ after the rising edge of $\overline{\text{IORQ}}$ as shown in figure 5.0-1b. This action guarantees that READY is low while port data is changing and that a positive edge is generated on READY whenever an Output instruction is executed.

MODE 0 (OUTPUT)TIMING
Figure 5.0-1a



$\overline{\text{WR}}$* = RD · $\overline{\text{CE}}$ · $\overline{\text{C/D}}$ · $\overline{\text{IORQ}}$

MODE 0 (OUTPUT) TIMING
Figure 5.0-1b



$\overline{\text{WR}}$* = RD · $\overline{\text{CE}}$ · $\overline{\text{C/D}}$ · $\overline{\text{IORQ}}$

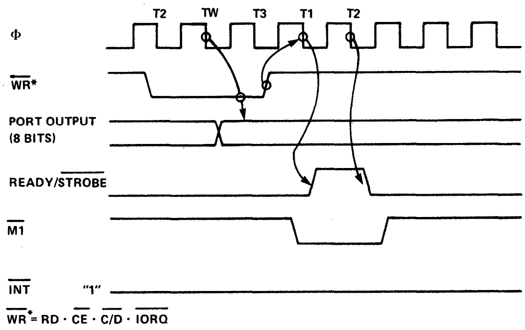
By connecting READY to $\overline{\text{STROBE}}$ a positive pulse with a duration of one clock period can be created as shown in Figure 5.0-1c. The positive edge of READY/$\overline{\text{STROBE}}$ will not generate an interrupt because the positive portion of $\overline{\text{STROBE}}$ is less than the width of $\overline{\text{M1}}$ and as such will not generate an interrupt due to the internal logic configuration of the PIO.

If the PIO is not in a reset status (i.e. a control mode has been selected), the output register may be loaded before Mode 0 is selected. This allows port output lines to become active in a user defined state. For example, assume the outputs are desired to become active in a logic one state, the following would be the initialization sequence:

a) PIO RESET
b) Load Interrupt Vector
c) Select Mode 1 (input) (automatic due ro RESET)
d) Write FF to Data Port
e) Select Mode 0 (Outputs go to "1's")
f) Enable Interrupt if desired

MODE 0 (OUTPUT) TIMING - READY TIED TO STROBE

Figure 5.0-1c



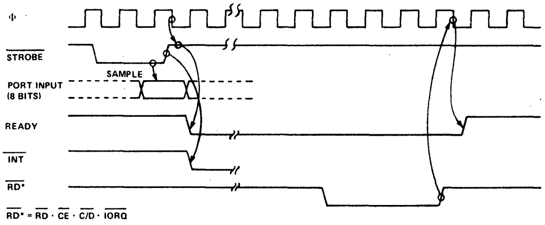
5.2 INPUT MODE (MODE 1)

Figure 5.0-2 illustrates the timing of an input cycle. The peripheral initiates this cycle using The STROBE line after the CPU has performed a data read. A low level on this line loads data into the port input register and the rising edge of the STROBE line activates the interrupt request line (INT) if the interrupt enable is set and this is the highest priority requesting device. The next falling edge of the clock line (Φ) will then reset the READY line to an inactive state signifying that the input register is full and further loading must be inhibited until the CPU reads the data. The CPU will in the course of its interrupt service routine, read the data from the interrupting port. When this occurs, the positive edge from the CPU RD signal will raise the READY line with the next low going transition of Φ, indicating that new data can be loaded into the PIO.

Since RESET causes READY to go low a dummy Input instruction may be needed in some systems to cause READY to go high the first time in order to start "handshaking".

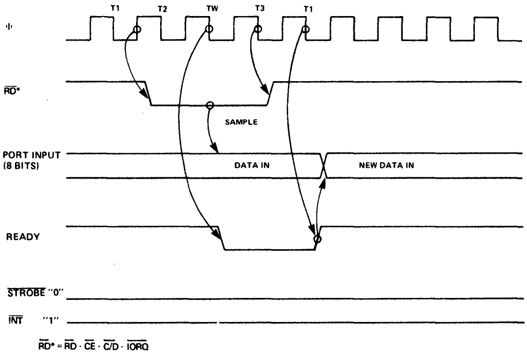
MODE 1 (INPUT) TIMING

Figure 5.0-2a



MODE 1 (INPUT) TIMING (NO STROBE INPUT)

Figure 5.0-2b



MODE 1 (INPUT) TIMING (NO STROBE INPUT)

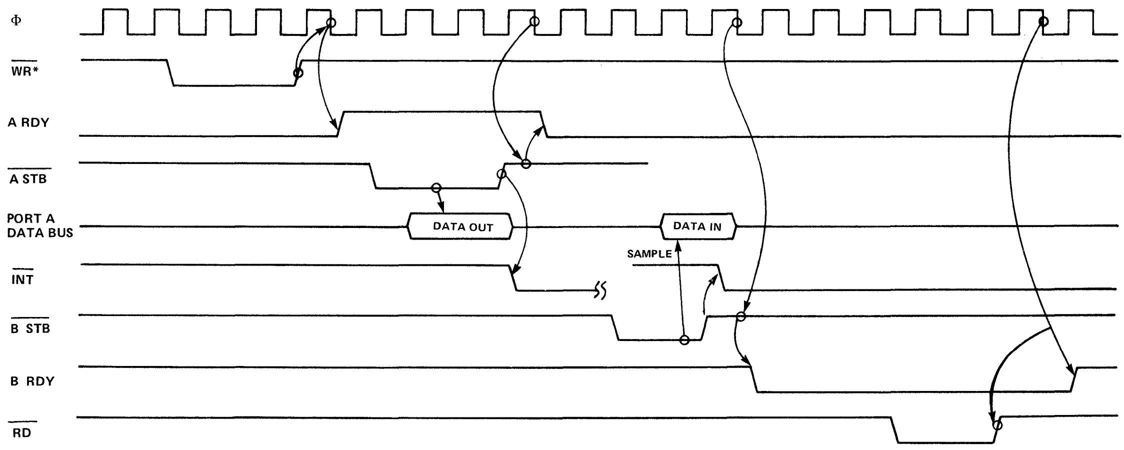
If already active, READY will be forced low one and one-half Φ periods following the falling edge of IORQ during a read of a PIO port as shown in Figure 5.0-2b. If the user strobes data into the PIO only when READY is high, the forced state of READY will prevent input register data from changing while the CPU is reading the PIO. Ready will go high again after the rising edge of the IORQ as previously described.

5.3 BIDIRECTIONAL MODE (MODE 2)

This mode is merely a combination of Mode 0 and Mode 1 using all four handshake lines. Since it requires all four lines, it is available only on Port A. When this mode is used on Port A, Port B must be set to the Bit Control Mode. The same interrupt vector will be returned for a Mode 3 interrupt on Port B and an input transfer interrupt during Mode 2 operation of Port A. Ambiguity is avoided if Port B is operated in a polled mode and the Port B mask register is set to inhibit all bits.

Figure 5.0-3 illustrates the timing for this mode. It is almost identical to that previously described for Mode 0 and Mode 1 with the Port A handshake lines used for output control and the Port B lines used for input control. The difference between the two modes is that, in Mode 2, data is allowed out onto the bus only when the A $\overline{\text{STROBE}}$ is low. The rising edge of this strobe can be used to latch the data into the peripheral since the data will remain stable until after this edge. The input portion of Mode 2 operates identically to Mode 1. Note that both Port A and Port B must have their interrupts enabled to achieve an interrupt driven bidirectional transfer.

PORT A, MODE 2 (BIDIRECTIONAL) TIMING
Figure 5.0-3



$\overline{WR}^* = RD \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$
$\overline{RD}^* = \overline{RD} \cdot \overline{CE} \cdot \overline{C/D} \cdot \overline{IORQ}$

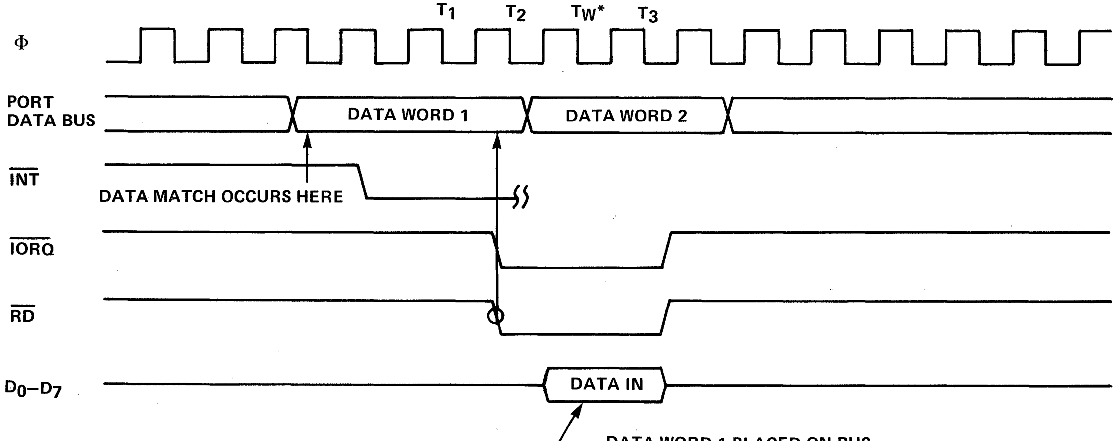
The peripheral must not gate data onto a port data bus while A STB is active. Bus contention is avoided if the peripheral uses B STB to gate input data onto the bus. The PIO uses the B STB low level to sample this data. The PIO has been designed with a zero hold time requirement for the data when latching in this mode so that this simple gating structure can be used by the peripheral. That is, the data can be disabled from the bus immediately after the strobe rising edge. Note that if A STB is low during a read operation of Port A (in response to a B STB interrupt) the data in the output register will be read by the CPU instead of the correct data in the data input register. The correct data is latched in the input register it just cannot be read by the CPU while A STB is low. If the A STB signal could go low during a CPU Read, it should be blocked from reaching the A STB input of the PIO while BRDY is low (the CPU read will occur while BRDY is low as the RD signal returns BRDY high).

5.4 CONTROL MODE (MODE 3)

The control mode does not utilize the handshake signals and a normal port write or port read can be executed at any time. When writing, the data will be latched into output registers with the same timing as Mode 0. A RDY will be forced low whenever Port A is operated in Mode 3. B RDY will be held low whenever Port B is operated in Mode 3 unless Port A is in Mode 2. In the latter case, the state of B RDY will not be affected.

When reading the PIO, the data returned to the CPU will be composed of output register data from those port data lines assigned as outputs and input register data from those port data lines assigned as inputs. The input register will contain data which was present immediately prior to the falling edge of $\overline{RD}$. See Figure 5.0-4.

MODE 3 TIMING
Figure 5.0-4a



*Timing Diagram Refers to Bit Mode Read

An interrupt will be generated if interrupts from the port are enabled and the data on the port data lines satisfies the logical equation defined by the 8-bit mask control registers. Another interrupt will not be generated until a change occurs in the status of the logical equation. A Mode 3 interrupt will be generated only if the result of a Mode 3 logical operation changes from false to true. For example, assume that the Mode 3 logical equation is an "OR" function. An unmasked port data line becomes active and an interrupt is requested. If a second unmasked port data line becomes active concurrently with the first, a new interrupt will not be requested since a change in the result of the Mode 3 logical operation has not occurred. Note that port pins defined as outputs can contribute to the logical equation if their bit positions are unmasked.

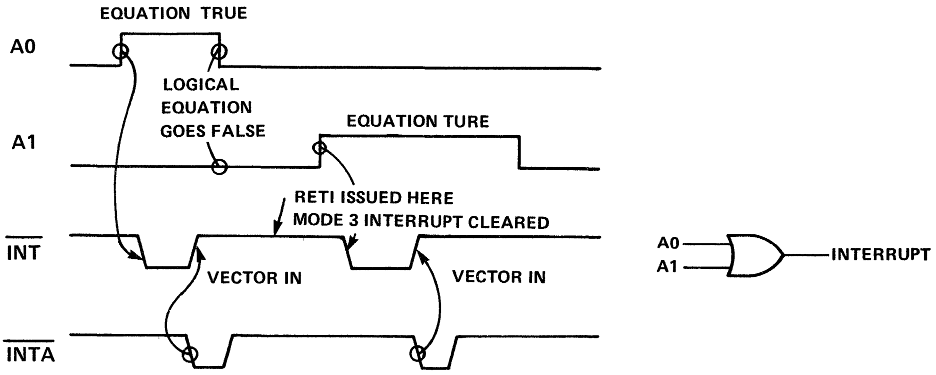
If the result of a logical operation becomes true immediately prior to or during $\overline{M1}$, an interrupt will be requested after the trailing edge of $\overline{M1}$, provided the logical equation remains true after $\overline{M1}$ returns high.

Figure 5.0-4b is an example of Mode 3 interrupts. The port has been placed in Mode 3 and OR logic selected and signals are defined to be high. All but bits A0 and A1 are masked out and are not monitored thereby creating a two input positive logic OR gate. In the timing diagram A0 is shown going high and creating an interrupt ($\overline{INT}$ goes low) and the CPU responds with an Interrupt Acknowledge cycle ($\overline{INTA}$). The PIO port with its interrupt pending sends in its Vector and the CPU goes off into the Interrupt Service Routine. A0 is shown going inactive either by itself or perhaps as a result of action taken in the Interrupt Service Routine (making the logical equation false). An arrow is shown at the point in time where the Service Routine issues the RETI instruction which clears the PIO interrupt structure. A1 is next shown going high making the logical equation-true and generating another interrupt. Two important points need to be made from this example:

1) A1 must not go high before A0 goes low or else the logical equation will not go false – a requirement fo' A1 to be able to generate an interrupt.

2) In order for A1 to generate an interrupt it must be high after the RETI issued by A0's Service Routine clears the PIO's Interrupt structure. In other words, if A1 were a positive pulse that occurred after A0 went low (to make the equation false) and went low before the RETI had cleared the Interrupt Structure it would have been missed. The logic equation must become false after the INTA for A0's service and then must be true or go true after RETI clears the previous interrupt for another interrupt to occur.

MODE 3 EXAMPLE
Figure 5.0-4b



Z80 Family

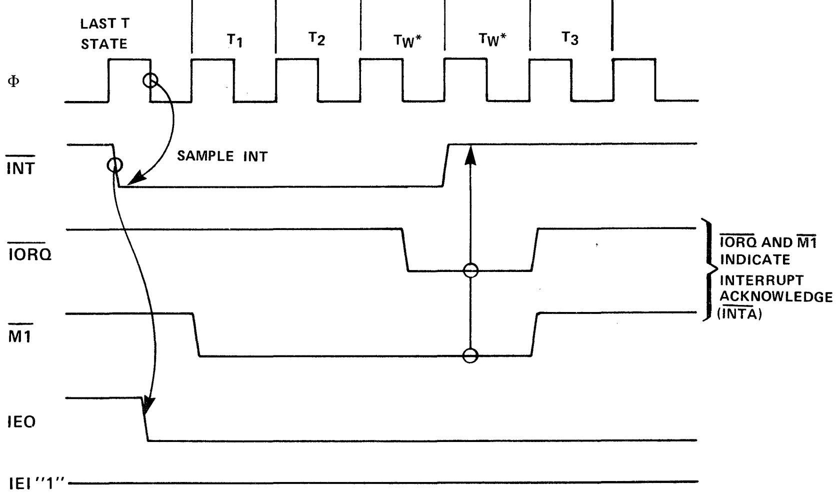
6.0 INTERRUPT SERVICING

Some time after an interrupt is requested by the PIO, the CPU will send out an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). During this time the interrupt logic of the PIO will determine the highest priority port which is requesting an interrupt. (This is simply the device with its Interrupt Enable Input high and its Interrupt Enable Output low). To insure that the daisy chain enable lines stabilize, devices are inhibited from changing their interrupt request status when $\overline{M1}$ is active. The highest priority device places the contents of its interrupt vector register onto the Z80 data bus during interrupt acknowledge.

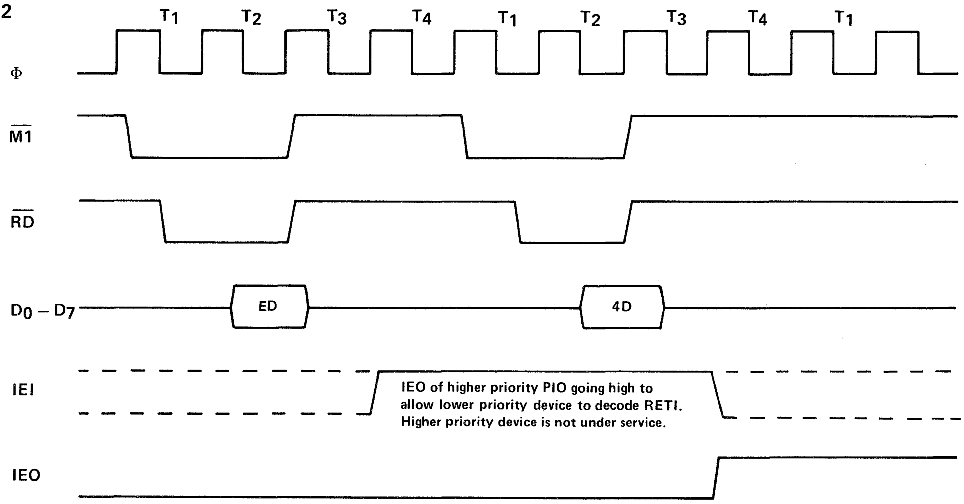
Figure 6.0-1 illustrates the timing associated with interrupt requests. During $\overline{M1}$ time, no new interrupt requests can be generated. This gives time for the Int Enable signals to ripple through up to four PIO circuits. The PIO with IEI high and IEO low during $\overline{INTA}$ will place the 8-bit interrupt vector of the appropriate port on the data bus at this time.

If an interrupt requested by the PIO is acknowledged, the requesting port is 'under service'. IEO of this port will remain low until a return from interrupt instruction (RETI) is executed while IEI of the port is high. If an interrupt request is not acknowledged, IEO will be forced high for one $\overline{M1}$ cycle after the PIO decodes the opcode 'ED'. This action guarantees that the two byte RETI instruction is decoded by the proper PIO port. See Figure 6.0-2.

INTERRUPT ACKNOWLEDGE TIMING
Figure 6.0-1

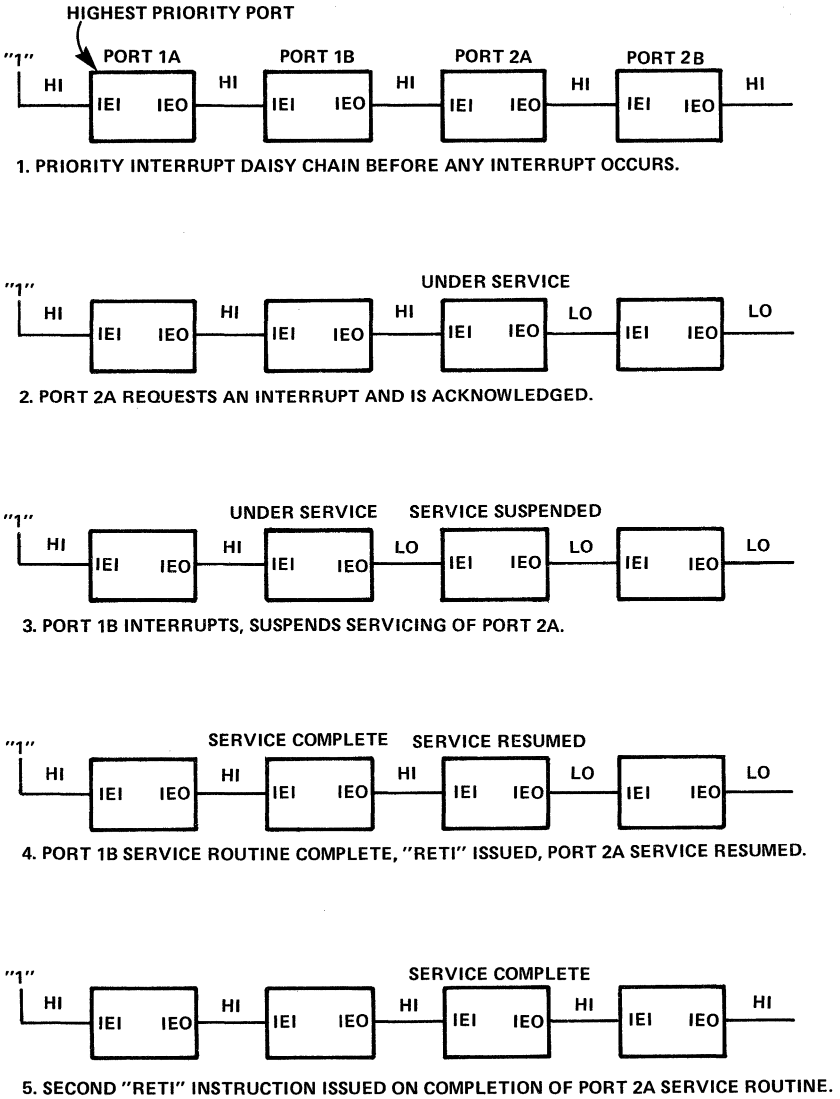


RETURN FROM INTERRUPT CYCLE
Figure 6.0-2


T_1
T_2
T_3
T_4
T_1
T_2
T_3
T_4
T_1
Φ
$\overline{M1}$
$\overline{RD}$
$D_0 - D_7$
ED
4D
IEI
IEO of higher priority PIO going high to allow lower priority device to decode RETI. Higher priority device is not under service.
IEO

DAISY CHAIN INTERRUPT SERVICING

Figure 6.0-3



1. PRIORITY INTERRUPT DAISY CHAIN BEFORE ANY INTERRUPT OCCURS.

2. PORT 2A REQUESTS AN INTERRUPT AND IS ACKNOWLEDGED.

3. PORT 1B INTERRUPTS, SUSPENDS SERVICING OF PORT 2A.

4. PORT 1B SERVICE ROUTINE COMPLETE, "RETI" ISSUED, PORT 2A SERVICE RESUMED.

5. SECOND "RETI" INSTRUCTION ISSUED ON COMPLETION OF PORT 2A SERVICE ROUTINE.

Figure 6.0-3 illustrates a typical nested interrupt sequence that could occur with four ports connected in the daisy chain. In this sequence Port 2A requests and is granted an interrupt. While this port is being serviced, a higher priority port (1B) requests and is granted an interrupt. The service routine for the higher priority port is completed and a RETI instruction is executed to indicate to the port that its routine is complete. At this time the service routine of the lower priority port is completed.

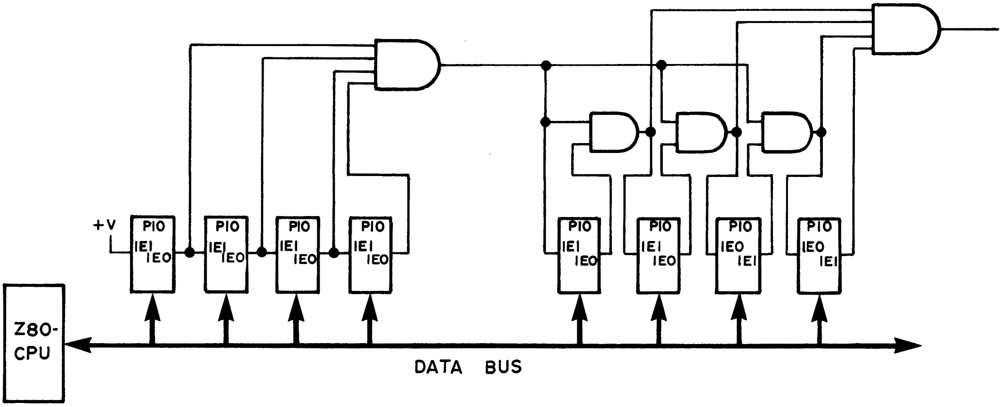
7.0 APPLICATIONS

7.1 EXTENDING THE INTERRUPT DAISY CHAIN

Without any external logic, a maximum of four Z80-PIO devices may be daisy chained into a priority interrupt structure. This limitation is required so that the interrupt enable status (IEO) ripples through the entire chain between the beginning of $\overline{M1}$, and the beginning of $\overline{IORQ}$ during an interrupt acknowledge cycle. Since the interrupt enable status cannot change during $\overline{M1}$, the vector address returned to the CPU is assured to be from the highest priority device which requested an interrupt.

If more than four PIO devices must be accommodated, a "look-ahead" structure may be used as shown in figure 7.0-1. With this technique more than thirty PIO's may be chained together using standard TTL logic.

A METHOD OF EXTENDING THE INTERRUPT PRIORITY DAISY CHAIN
Figure 7.0-1



7.2 I/O DEVICE INTERFACE

In this example, the Z80-PIO is connected to an I/O terminal device which communicates over an 8 bit parallel bidirectional data bus as illustrated in figure 7.0-2. Mode 2 operation (bidirectional) is selected by sending the following control word to Port A:

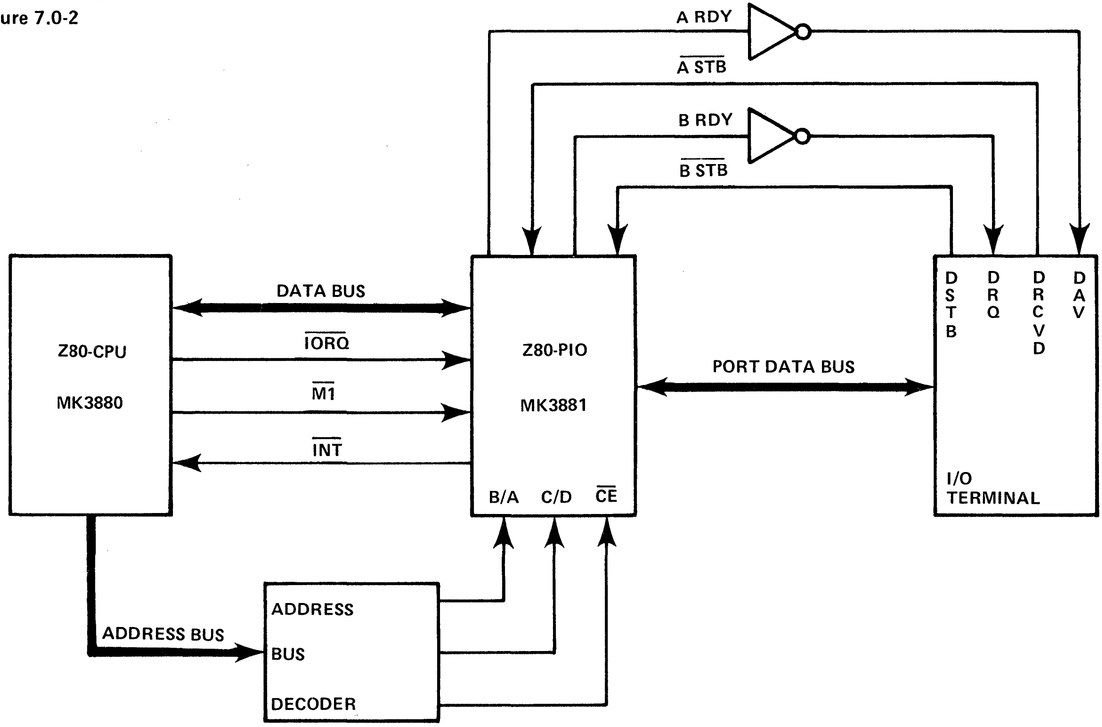
EXAMPLE I/O INTERFACE
Figure 7.0-2

D7	D6	D5	D4	D3	D2	D1	D0
1	0	X	X	1	1	1	1

MODE CONTROL

EXAMPLE I/O INTERFACE

Figure 7.0-2



Next, the proper interrupt vector is loaded (refer to CPU Manual for details on the operation of the interrupt).

V7	V6	V5	V4	V3	V2	V1	0

Interrupts are then enabled by the rising edge of the first $\overline{M1}$ after the interrupt mode word is set unless that $\overline{M1}$ defines an interrupt acknowledge cycle. If a mask follows the interrupt mode word, interrupts are enabled by the rising edge of the first $\overline{M1}$ following the setting of the mask.

Data can now be transferred between the peripheral and the CPU. The timing for this transfer is as described in Section 5.0.

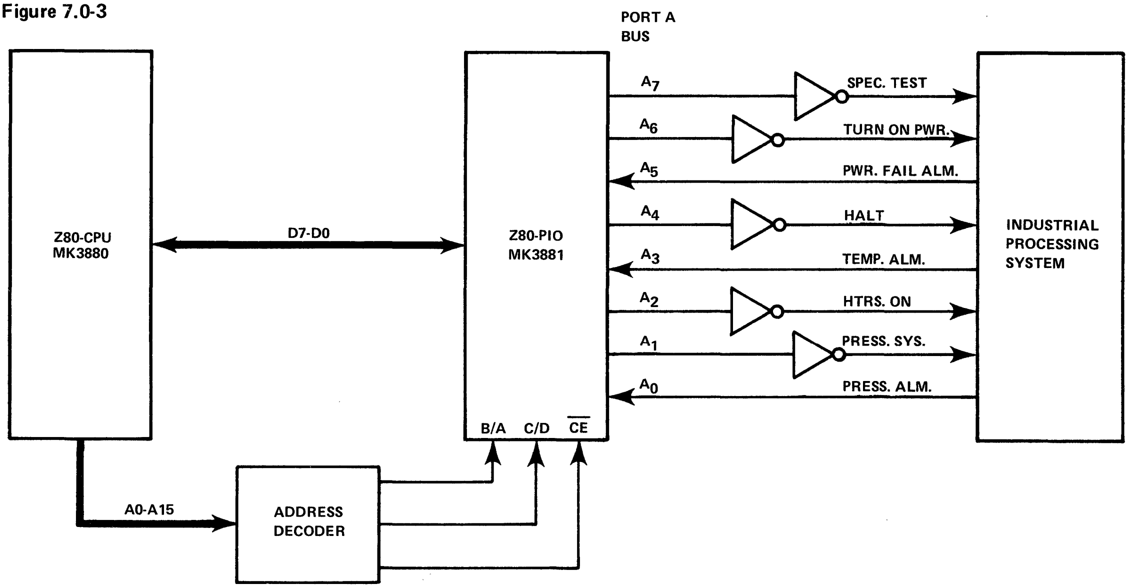
7.3 CONTROL INTERFACE

A typical control mode application is illustrated in figure 7.0-3. Suppose an industrial process is to be monitored. The occurrence of any abnormal operating condition is to be reported to a Z80-CPU based control system. The process control and status word has the following format:

D7	D6	D5	D4	D3	D2	D1	D0
Special Test	Turn On Power	Power Failure Alarm	Halt Processing	Temp. Alarm	Temp Heaters On	Pressurize System	Pressure Alarm

CONTROL MODE APPLICATION

Figure 7.0-3



The PIO may be used as follows. First Port A is set for Mode 3 operation by writing the following control word to Port A.

D7	D6	D5	D4	D3	D2	D1	D0
1	1	X	X	1	1	1	1

Whenever Mode 3 is selected, the next control word sent to the port must be an I/O select word. In this example we wish to select port data lines A5, A3, and A0 as inputs and so the following control word is written:

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	1	0	0	1

Next the desired interrupt vector must be loaded (refer to the CPU manual for details);

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	V0

An interrupt control word is next sent to the port:

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	1	0	1	1	1
Enable Interrupts	OR Logic	Active High	Mask Follows	Interrupt Control			

The mask word following the interrupt mode word is:

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	1	0	1	1	0

Selects A5, A3 and A0 to be monitored

Now, if a sensor puts a high level on line A5, A3, or A0, an interrupt request will be generated. The mask word may select any combination of inputs or outputs to cause an interrupt. For example, if the mask word above had been:

D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	1	0	1	1	0

then an interrupt request would also occur if bit A7 (special Test) of the output register was set.

Assume that the following port assignments are to be used:

$E0_H$=	Port A Data
$E1_H$=	Port B Data
$E2_H$=	Port A Control
$E3_H$=	Port B Control

All port numbers are in hexadecimal notation. This particular assignment of port numbers is convenient since A_0 of the address bus can be used as the Port B/A Select and A_1 of the address bus can be used as the Control/Data Select. The Chip Enable would be the decode of CPU address bits A_7 thru A_2 (111000). Note that if only a few peripheral devices are being used, a Chip Enable decode may not be required since a higher order address bit could be used directly.

8.0 PROGRAMMING SUMMARY

8.1 LOAD INTERRUPT VECTOR

V7	V6	V5	V4	V3	V2	V1	0

8.2 SET MODE

M1	M0	X	X	1	1	1	1

MODE NUMBER	M_1	M_0	MODE
0	0	0	Output
1	0	1	Input
2	1	0	Bidirectional
3	1	1	Bit Control

When selecting Mode 3, the next word to the PIO must set the I/O Register:

I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

I/O = 1 Sets bit to Input
I/O = 0 Sets bit to Output

8.3 SET INTERRUPT CONTROL

Int Enable	AND/ OR	High/ Low	Mask Follows	0	1	1	1

USED IN MODE 3 ONLY

If the "mask follows" bit is high, the next control word written to the PIO must be the mask:

MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

MB = 0, Monitor bit
MB = 1, Mask bit from being monitored

Also, the interrupt enable flip flop of a port may be set or reset without modifying the rest of the interrupt control word by using the following command:

Int Enable	X	X	X	0	0	1	1

9.0 ELECTRICAL SPECIFICATIONS

9.1 ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	Specified operating range.
Storage Temperature	–65° C to +150° C
Voltage On Any Pin With Respect To Ground	–0.3V to +7V
Power Dissipation	.6W

9.2 D. C. CHARACTERISTICS

Table 9.2-1

T_A = 0° C to 70° C, V_{CC} = 5 V ± 5% unless otherwise specified

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.0mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250μA
I_{CC}	Power Supply Current		70*	mA	
I_{LI}	Input Leakage Current		10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	μA	V_{OUT}=2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float		-10	μA	V_{OUT} = 0.4 V
I_{LD}	Data Bus Leakage Current in Input Mode		±10	μA	$0 \leq V_{IN} \leq V_{CC}$
I_{OHD}	Darlington Drive Current	-1.5		mA	V_{OH}=1.5V Port B Only

* 150mA for -4, -10, and -20 devices.

9.3 CAPACITANCE

Table 9.3-1

T_A = 25° C, f = 1 MHz

Symbol	Parameter	Max	Unit	Test Condition
C_{Φ}	Clock Capacitance	10	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

9.4A A.C. CHARACTERISTICS MK3880, MK3880-10, MK3880-20, Z80-PIO

Table 9.4-1A T_A= 0°C to 70°C, V_{CC} = +5V ± 5%, unless otherwise noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c	Clock Period	400	[1]	nsec	
	$t_W\ (\Phi H)$	Clock Pulse Width, Clock High	170	2000	nsec	
	$t_W\ (\Phi L)$	Clock Pulse Width, Clock Low	170	2000	nsec	
	t_r, t_f	Clock Rise and Fall Times		30	nsec	
	t_h	Any Hold Time for Specified Set-Up Time	0		nsec	
C/D SEL $\overline{\text{CE}}$ ETC.	$t_S\ \Phi(CS)$	Control Signal Set-up Time to Rising Edge of Φ During Read or Write Cycle	280		nsec	
D_0-D_7	$t_{DR(D)}$	Data Output Delay from Falling Edge of RD		430	nsec	[2]
	$t_{S\Phi(D)}$	Data Set-Up Time to Rising Edge of Φ During Write or $\overline{\text{M1}}$ Cycle	50		nsec	C_L = 50pF
	$t_{DI(D)}$	Data Output Delay from Falling Edge of IORQ During $\overline{\text{INTA}}$ Cycle		340	nsec	[3]
	$t_{F(D)}$	Delay to Floating Bus (Output Buffer Disable Time)		160	nsec	
IEI	$t_S\ (IEI)$	IEI Set-Up Time to Falling Edge of $\overline{\text{IORQ}}$ During $\overline{\text{INTA}}$ Cycle	140		nsec	
IEO	$t_{DH}\ (IO)$	IEO Delay Time from Rising Edge of IEI		210	nsec	[5]
	$t_{DL}\ (IO)$	IEO Delay Time from Falling Edge of IEI		190	nsec	[5] C_L = 50pF
	$t_{DM}\ (IO)$	IEO Delay from Falling Edge of M1 (Interrupt Occurring Just Prior to M1) See Note A.		300	nsec	[5]
$\overline{\text{IORQ}}$	$t_S\ \Phi(IR)$	$\overline{\text{IORQ}}$ Set-Up Time to Rising Edge of Φ During Read or Write Cycle	250		nsec	
$\overline{\text{M1}}$	$t_S\ \Phi\ (M1)$	$\overline{\text{M1}}$ Set-Up Time to Rising Edge of Φ During $\overline{\text{INTA}}$ or $\overline{\text{M1}}$ Cycle. See Note B.	210		nsec	
$\overline{\text{RD}}$	$t_S\ \Phi\ (RD)$	$\overline{\text{RD}}$ Set-Up Time to Rising Edge of Φ During Read or $\overline{\text{M1}}$ Cycle	240		nsec	
A_0-A_7 B_0-B_7	$t_S\ (PD)$	Port Data Set-Up Time to Rising Edge of $\overline{\text{STROBE}}$ (Mode 1)	260		nsec	
	$t_{DS}\ (PD)$	Port Data Output Delay from Falling Edge of $\overline{\text{STROBE}}$ (Mode 2)		230	nsec	[5]
	$t_F\ (PD)$	Delay to Floating Port Data Bus from Rising Edge of $\overline{\text{STROBE}}$ (Mode 2)		200	nsec	C_L = 50pF
	$t_{DI}\ (PD)$	Port Data Stable from Rising Edge of $\overline{\text{IORQ}}$ During $\overline{\text{WR}}$ Cycle (Mode 0)		200	nsec	[5]
$\overline{\text{ASTB}}$ $\overline{\text{BSTB}}$	$t_W\ (ST)$	Pulse Width, $\overline{\text{STROBE}}$	150 [4]		nsec nsec	
$\overline{\text{INT}}$	$t_D\ (IT)$	$\overline{\text{INT}}$ Delay Time from Rising Edge of $\overline{\text{STROBE}}$		490	nsec	
	$t_D\ (IT3)$	$\overline{\text{INT}}$ Delay Time from Data Match During Mode 3 Operation		650	nsec	
ARDY BRDY	$t_{DH}\ (RY)$	Ready Response Time from Rising Edge of $\overline{\text{IORQ}}$		t_c+ 460	nsec	[5] C_L = 50pF
	$t_{DL}\ (RY)$	Ready Response Time from Rising Edge of $\overline{\text{STROBE}}$		t_c+ 400	nsec	[5]

A. $2.5t_c > (N\text{-}2)t_{DL(IO)} + t_{DM\ (IO)} + t_{S(IEI)}$ + TTL Buffer Delay, if any

B. $\overline{\text{M1}}$ must be active for a minimum of 2 clock periods to reset the PIO.

[1] $t_c = t_{W\ (\Phi H)} + t_{W\ (\Phi L)} + t_r + t_f$

[2] Increase $t_{DR(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

[3] Increase $t_{DI\ (D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

[4] For Mode 2: $t_{W\ (ST)} > t_{S(PD)}$

[5] Increase these values by 2 nsec for each 10pF increase in loading up to 100pF max.

9.4B A.C. CHARACTERISTICS MK3881-4, Z80A-PIO

Table 9.4-1B T_A=0°C to 70°C, V_{CC} = +5V ± 5%, unless otherwise noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c	Clock Period	250	[1]	nsec	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	105	2000	nsec	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	105	2000	nsec	
	t_r, t_f	Clock Rise and Fall Times		30	nsec	
	t_h	Any Hold Time for Specified Set-Up Time	0		nsec	
C/D SEL CE ETC.	$t_{S\Phi}(CS)$	Control Signal Set-Up Time to Rising Edge of Φ During Read or Write Cycle	145		nsec	
D_0-D_7	$t_{DR}(D)$	Data Output Delay From Falling Edge of $\overline{RD}$		380	nsec	[2]
	$t_{S\Phi}(D)$	Data Set-Up Time to Rising Edge of Φ During Write or $\overline{M1}$ Cycle	50		nsec	C_L = 50pF
	$t_{DI}(D)$	Data Output Delay from Falling edge of $\overline{IORQ}$ During $\overline{INTA}$ Cycle		250	nsec	[3]
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		110	nsec	
IEI	$t_S(IEI)$	IEI Set-Up Time to Falling edge of $\overline{IORQ}$ during $\overline{INTA}$ Cycle	140		nsec	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		160	nsec	[5]
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		130	nsec	[5] C_L = 50pF
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$) See Note A.		190	nsec	[5]
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Set-Up Time to Rising Edge of Φ During Read or Write Cycle	115		nsec	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Set-Up Time to Rising Edge of Φ During $\overline{INTA}$ or $\overline{M1}$ Cycle. See Note B.	90		nsec	
$\overline{RD}$	$t_{S\Phi}(RD)$	$\overline{RD}$ Set-Up Time to Rising Edge of Φ During Read or $\overline{M1}$ Cycle	115		nsec	
A_0-A_7, B_0-B_7	$t_S(PD)$	Port Data Set-Up Time to Rising Edge of $\overline{STROBE}$ (MODE 1)	230		nsec	
	$t_{DS}(PD)$	Port Data Output Delay from Falling Edge of STROBE (Mode 2)		210	nsec	[5]
	$t_F(PD)$	Delay to Floating Port Data Bus from Rising Edge of STROBE (Mode 2)		180	nsec	C_L = 50pF
	$t_{DI}(PD)$	Port Data Stable from Rising Edge of $\overline{IORQ}$ During $\overline{WR}$ Cycle (Mode 0)		180	nsec	[5]
$\overline{ASTB}$ $\overline{BTSB}$	$t_W(ST)$	Pulse Width, $\overline{STROBE}$	150 [4]		nsec nsec	
$\overline{INT}$	$t_D(IT)$	$\overline{INT}$ Delay Time from Rising Edge of $\overline{STROBE}$		440	nsec	
	$t_D(IT3)$	$\overline{INT}$ Delay Time from Data Match During Mode 3 Operation		650	nsec	
ARBY, BRDY	$t_{DH}(RY)$	Ready Response Time from Rising Edge of $\overline{IORQ}$		t_c + 410	nsec	[5] C_L = 50pF
	$t_{DL}(RY)$	Ready Response Time from Rising Edge of $\overline{STROBE}$		t_c+ 360	nsec	[5]

A. $2.5t_c > (N-2)t_{DL}(IO) + t_{DM}(IO) + t_S(IEI)$ + TTL Buffer Delay, if any

B. M1 must be active for a minimum of 2 clock periods to reset the PIO.

[1] $t_c = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$

[2] Increase $t_{DR}(D)$ by 10 nsec for each 50pF increase in loading up to 200pF max.

[3] Increase $t_{DI}(D)$ by 10 nsec for each 50pF increase in loading up to 200pF max.

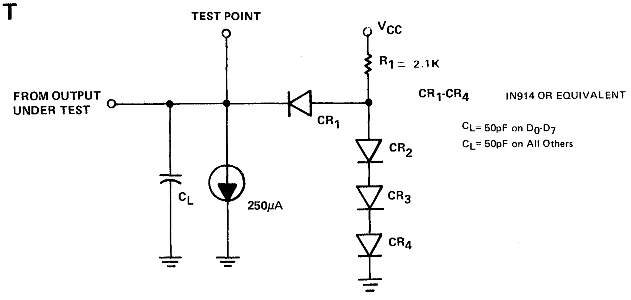
[4] For Mode 2: $t_W(ST) > t_S(PD)$

[5] Increase these values by 2 nsec for each 10pF increase in loading up to 100pF max.

Z80 Family

OUTPUT LOAD CIRCUIT

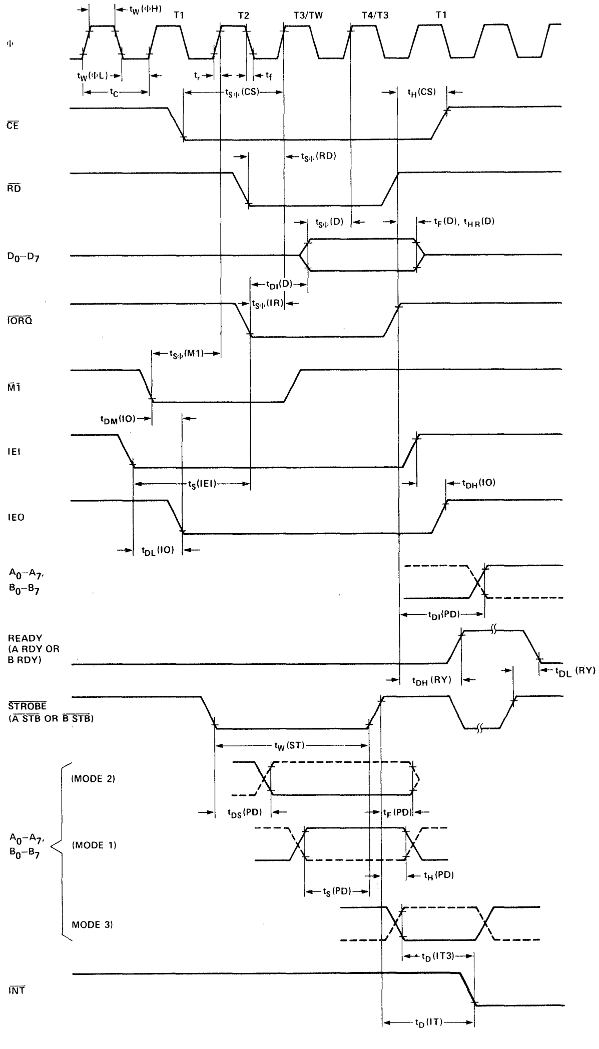
Figure 9.4-1



9.5 TIMING DIAGRAM

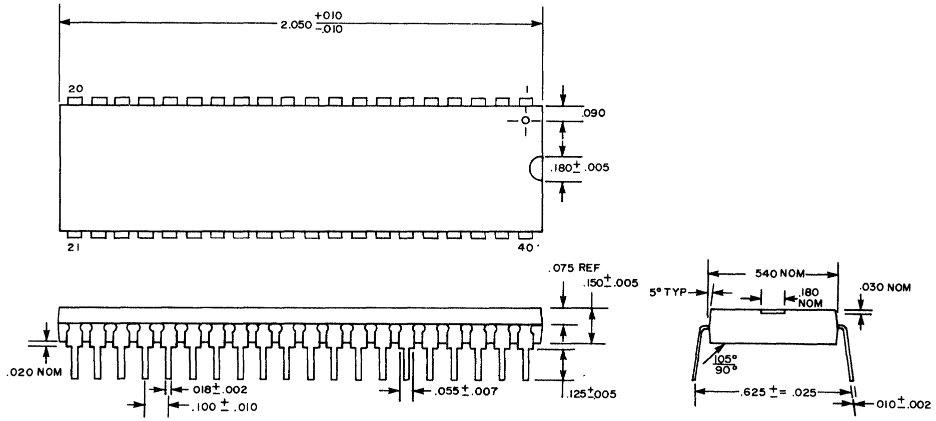
Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV	= +0.5V

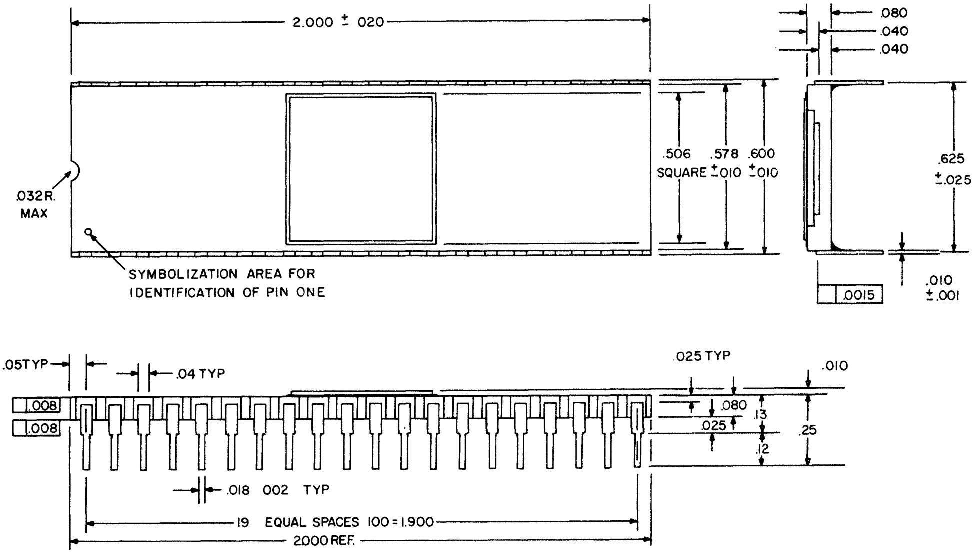


10.0 PACKAGE DESCRIPTION AND ORDERING INFORMATION

PACKAGE DESCRIPTION – 40 PIN DUAL IN-LINE PLASTIC PACKAGE



PACKAGE DESCRIPTION – 40 PIN DUAL IN-LINE CERAMIC PACKAGE



ORDERING INFORMATION

PART NO.	DESIGNATOR	PACKAGE TYPE	MAX CLOCK FREQUENCY	TEMPERATURE RANGE
MK3881N	Z80-PIO	Plastic	2.5 MHz	0° to 70°C
MK3881P	Z80-PIO	Ceramic	2.5 MHz	
MK3881N-4	Z80A-PIO	Plastic	4.0 MHz	
MK3881P-4	Z80A-PIO	Ceramic	4.0 MHz	
MK3881P-10	Z80-PIO	Ceramic	2.5 MHz	-40° to +85°C
MK3881P-20	Z80-PIO	Ceramic	2.5 MHz	-55° to +125°C

Z80 Family

MK3882 COUNTER TIMER CIRCUIT

Z80 Family

TABLE OF CONTENTS

1.0 INTRODUCTION

The Z80-Counter Timer Circuit (CTC) is a programmable component with four independent channels that provide counting and timing functions for microcomputer systems based on the Z80-CPU. The CPU can configure the CTC channels to operate under various modes and conditions as required to interface with a wide range of devices. In most applications, little or no external logic is required. The Z80-CTC utilizes N-channel silicon gate depletion load technology and is packaged in a 28-pin DIP. The Z80-CTC requires only a single 5 volt supply and a one-phase 5 volt clock. Major features of the Z80-CTC include:

- All inputs and outputs fully TTL compatible.
- Each channel may be selected to operate in either Counter Mode or Timer Mode.
- Used in either mode, a CPU-readable Down Counter indicates number of counts-to-go until zero.
- A Time Constant Register can automatically reload the Down Counter at Count Zero in Counter and Timer Mode.
- Selectable positive or negative trigger initiates time operation in Timer Mode. The same input is monitored for event counts in Counter Mode.
- Three channels have Zero Count/Timeout outputs capable of driving Darlington transistors.
- Interrupts may be programmed to occur on the zero count condition in any channel.
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic.

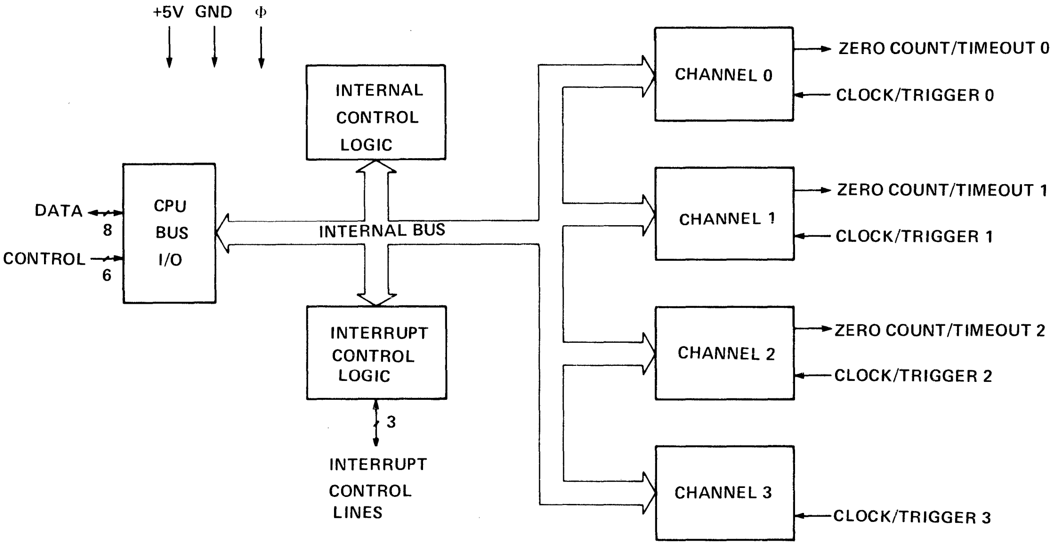
2.0 CTC ARCHITECTURE

2.1 OVERVIEW

A block diagram of the Z80-CTC is shown in Figure 2.0-1. The internal structure of the Z80-CTC consists of a Z80-CPU bus interface, Internal Control Logic, four sets of Counter/Timer Channel Logic, and Interrupt Control Logic. The four independent counter/timer channels are identified by sequential numbers from 0 to 3. The CTC has the capability of generating a unique interrupt vector for each separate channel (for automatic vectoring to an interrupt service routine). The 4 channels can be connected into four contiguous slots in the standard Z80 priority chain with channel number 0 having the highest priority.The CPU bus interface logic allows the CTC device to interface directly to the CPU with no other external logic. However, port address decoders and/or line buffers may be required for large systems.

Z80-CTC BLOCK DIAGRAM

Figure 2.0-1

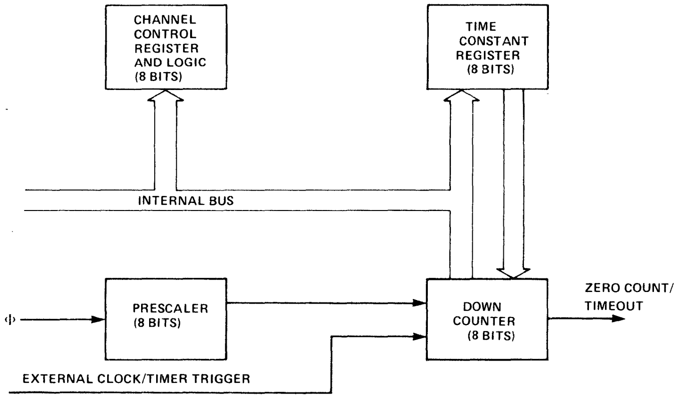


2.2 STRUCTURE OF CHANNEL LOGIC

The structure of one of the four sets of Counter/Timer Channel Logic is shown in Figure 2.0-2. This logic is composed of 2 registers, 2 counters and control logic. The registers are an 8-bit Time Constant Register and an 8-bit Channel Control Register. The counters are an 8-bit CPU-readable Down Counter and an 8-bit Prescaler.

CHANNEL BLOCK DIAGRAM

Figure 2.0-2



CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
Φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/ TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

2.2.1 THE CHANNEL CONTROL REGISTER AND LOGIC

The Channel Control Register (8-bit) and Logic is written to by the CPU to select the modes and parameters of the channel. Within the entire CTC device there are four such registers, corresponding to the four Counter/Timer Channels. Which of the four is being written to depends on the encoding of two channel select input pins: CS0 and CS1 (usually attached to A0 and A1 of the CPU address bus). This is illustrated in the truth table below:

	CS1	CS0
Ch0	0	0
Ch1	0	1
Ch2	1	0
Ch3	1	1

In the control word written to program each Channel Control Register, bit 0 is always set, and the other 7 bits are programmed to select alternatives on the channel's operating modes and parameters, as shown in the diagram below. (For a more complete discussion see section 4.0: "CTC Operating Modes" and section 5.0: "CTC Programming.")

CHANNEL CONTROL REGISTER

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT ENABLE	MODE	RANGE	SLOPE	TRIGGER	LOAD TIME CONSTANT	RESET	1

USED IN TIMER MODE ONLY

2.2.2 THE PRESCALER

Used in the Timer Mode only, the Prescaler is an 8-bit device which can be programmed by the CPU via the Channel Control Register to divide its input, the System Clock (Φ), by 16 or 256. The output of the Prescaler is then fed as an input to clock the Down Counter, which initially, and every time it clocks down to zero, is reloaded automatically with the contents of the Time Constant Register. In effect this again divides the System Clock by an additional factor of the time constant. Every time the Down Counter counts down to zero, its output, Zero Count/Timeout (ZC/TO), is pulsed high.

2.2.3 THE TIME CONSTANT REGISTER

The Time Constant Register is an 8-bit register, used in both Counter Mode and Timer Mode, programmed by the CPU just after the Channel Control Word with an integer time constant value of 1 through 256. This register loads the programmed value into the Down Counter when the CTC is first initialized and reloads the same value into the Down Counter automatically whenever it counts down thereafter to zero. If a new time constant is loaded into the Time Constant Register while a channel is counting or timing, the present down count will be completed before the new time constant is loaded into the Down Counter. (For details of how a time constant is written to a CTC channel, see section 5.0: "CTC Programming.")

2.2.4 THE DOWN COUNTER

The Down Counter is an 8-bit register used in both Counter Mode and Timer Mode loaded initially, and later when it counts down to zero, by the Time Constant Register. The Down Counter is decremented by each external clock edge in the Counter Mode, or in the Timer Mode, by the clock output of the Prescaler. At any time, by performing a simple I/O Read at the port address assigned to the selected CTC channel, the CPU can access the contents of this register and obtain the number of counts-to-zero. Any CTC channel may be programmed to generate an interrupt request sequence each time the zero count is reached.

In channels 0, 1, and 2, when the zero count condition is reached, a signal pulse appears at the corresponding ZC/TO pin. Due to package pin limitations, however, channel 3 does not have this pin and so may be used only in applications where this output pulse is not required.

2.3 INTERRUPT CONTROL LOGIC

The Interrupt Control Logic insures that the CTC acts in accordance with Z80 system interrupt protocol for nested priority interrupting and return from interrupt. The priority of any system device is determined by its physical location in a daisy chain configuration. Two signal lines (IEI and IEO) are provided in CTC devices to form this system daisy chain. The device closest to the CPU has the highest priority; within the CTC, interrupt priority is predetermined by channel number, with channel 0 having highest priority down to channel 3 which has the lowest priority. The purpose of a CTC-generated interrupt, as with any other peripheral device, is to force the CPU to execute an interrupt service routine. According to Z80 system interrupt protocol, lower priority devices or channels may not interrupt higher priority devices or channels that have already interrupted and have not had their interrupt service routines completed. However, high priority devices or channels may interrupt the servicing of lower priority devices or channels.

A CTC channel may be programmed to request an interrupt every time its Down Counter reaches a count of zero. (To utilize this feature requires that the CPU be programmed for interrupt mode 2.) Some time after the interrupt request, the CPU will send out an interrupt acknowledge, and the CTC's Interrupt Control Logic will determine the highest-priority channel which is requesting an interrupt within the CTC device. Then if the CTC's IEI input is active, indicating that it has priority within the system daisy chain, it will place an 8-bit Interrupt Vector on the system data bus. The high-order 5 bits of this vector will have been written to the CTC earlier as part of the CTC initial programming process; the next two bits will be provided by the CTC's Interrupt Control Logic as a binary code corresponding to the highest-priority channel requesting an interrupt; finally the low-order bit of the vector will always be zero according to a convention described below.

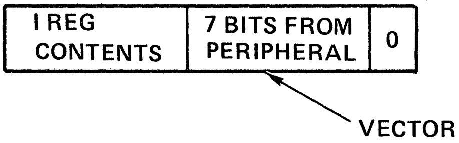
INTERRUPT VECTOR

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	X	X	0

D_2	D_1	
0	0	CHANNEL 0
0	1	CHANNEL 1
1	0	CHANNEL 2
1	1	CHANNEL 3

This interrupt vector is used to form a pointer to a location in memory where the address of the interrupt service routine is stored in a table. The vector represents the least significant 8 bits, while the CPU reads the contents of the I register to provide the most significant 8-bits of the 16-bit pointer. The address in memory pointed to will contain the low-order byte, and the next highest address will contain the high-order byte of an address which in turn contains the first opcode of the interrupt service routine. Thus in mode 2, a single 8-bit vector stored in an interrupting CTC can result in an indirect call to any memory location.

Z80 16-BIT POINTER (INTERRUPT STARTING ADDRESS)



2.3 INTERRUPT CONTROL LOGIC (Cont'd)

There is a Z80 system convention that all addresses in the interrupt service routine table should have their low-order byte in an even location in memory, and their high-order byte in the next highest location in memory, which will always be odd so that the least significant bit of any interrupt vector will always be even. Hence the least significant bit of any interrupt vector will always be zero.

The RETI instruction is used at the end of any interrupt service routine to initialize the daisy chain enable line IEO for proper control of nested priority interrupt handing. The CTC monitors the system data bus and decodes this instruction when it occurs. Thus the CTC channel control logic will know when the CPU has completed servicing an interrupt, without any further communication with the CPU being necessary.

3.0 CTC PIN DESCRIPTION

A diagram of the Z80-CTC pin configuration is shown in Figure 3.0-1. This section describes the function of each pin.

D7 - D0
Z80-CPU Data Bus (bi-directional, tri-state)

This bus is used to transfer all data and command words between the Z80-CPU and the Z80-CTC. There are 8 bits on this bus, of which D0 is the least significant.

CS1 - CS0
Channel Select (input, active high)

These pins form a 2-bit binary address code for selecting one of the four independent CTC channels for an I/O Write or Read (See truth table below.)

	CS1	CS0
Ch0	0	0
Ch1	0	1
Ch2	1	0
Ch3	1	1

$\overline{\text{CE}}$
Chip Enable (input, active low)

A low level on this pin enables the CTC to accept control words, Interrupt Vectors, or time constant data words from the Z80 Data Bus during an I/O Write cycle, or to transmit the contents of the Down Counter to the CPU during an I/O Read cycle. In most applications this signal is decoded from the 8 least significant bits of the address bus for any of the four I/O port addresses that are mapped to the four Counter/ Timer Channels.

Clock (Φ)
System Clock (input)

This single-phase clock is used by the CTC to synchronize certain signals internally.

$\overline{\text{M1}}$
Machine Cycle One Signal from CPU (input, active low)

When $\overline{\text{M1}}$ is active and the $\overline{\text{RD}}$ signal is active, the CPU is fetching an instruction from memory. When $\overline{\text{M1}}$ is active and the $\overline{\text{IORQ}}$ signal is active, the CPU is acknowledging an interrupt, alerting the CTC to place an Interrupt Vector on the Z80 Data Bus if it has daisy chain priority and one of its channels has requested an interrupt.

$\overline{\text{IORQ}}$
Input/Output Request from CPU (input, active low)

The $\overline{\text{IORQ}}$ signal is used in conjunction with the $\overline{\text{CE}}$ and $\overline{\text{RD}}$ signals to transfer data and Channel Control Words between the Z80-CPU and the CTC. During a CTC Write Cycle, $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ must be true and $\overline{\text{RD}}$ false. The CTC does not receive a specific write signal, instead generating its own internally from the inverse of a valid $\overline{\text{RD}}$ signal. In a CTC Read Cycle, $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ and $\overline{\text{RD}}$ must be active to place the contents of the Down Counter on the Z80 Data Bus. If $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are both true, the CPU is acknowledging an interrupt request, and the highest-priority interrupting channel will place its Interrupt Vector on the Z80 Data Bus.

3.0 CTC PIN DESCRIPTION (CONT'D)

$\overline{\text{RD}}$

Read Cycle Status from the CPU (input, active low)

The $\overline{\text{RD}}$ signal is used in conjunction with the $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ signals to transfer data and Channel Control Words between the Z80-CPU and the CTC. During a CTC Write Cycle, $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ must be true and $\overline{\text{RD}}$ false. The CTC does not receive a specific write signal, instead generating its own internally from the inverse of a valid $\overline{\text{RD}}$ signal. In a CTC Read Cycle, $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ and $\overline{\text{RD}}$ must be active to place the contents of the Down Counter on the Z80 Data Bus.

IEI

Interrupt Enable In (input, active high)

This signal is used to help form a system-wide interrupt daisy chain which establishes priorities when more than one peripheral device in the system has interrupting capability. A high level on this pin indicates that no other interrupting devices of higher priority in the daisy chain are being serviced by the Z80-CPU.

IEO

Interrupt Enable Out (output, active high)

The IEO signal, in conjunction with IEI, is used to form a system-wide interrupt priority daisy chain. IEO is high only if IEI is high and the CPU is not servicing an interrupt from any CTC channel. Thus this signal blocks lower priority devices from interrupting while a higher priority interrupting device is being serviced by the CPU.

$\overline{\text{INT}}$

Interrupt Request (output, open drain, active low)

This signal goes true when any CTC channel which has been programmed to enable interrupts has a zero-count condition in its Down Counter.

$\overline{\text{RESET}}$

Reset (input, active low)

This signal stops all channels from counting and resets channel interrupt enable bits in all control registers, thereby disabling CTC-generated interrupts. The ZC/TO and $\overline{\text{INT}}$ outputs go to their inactive states, IEO reflects IEI, and the CTC's data bus output drivers go to the high impedance state.

CLK/TRG3–CLK/TRG0

External Clock/Timer Trigger (input, user-selectable active high or low)

There are four CLK/TRG pins, corresponding to the four independent CTC channels. In the Counter Mode, every active edge on this pin decrements the Down Counter. In the Timer Mode, an active edge on this pin initiates the timing function. The user may select the active edge to be either rising or falling.

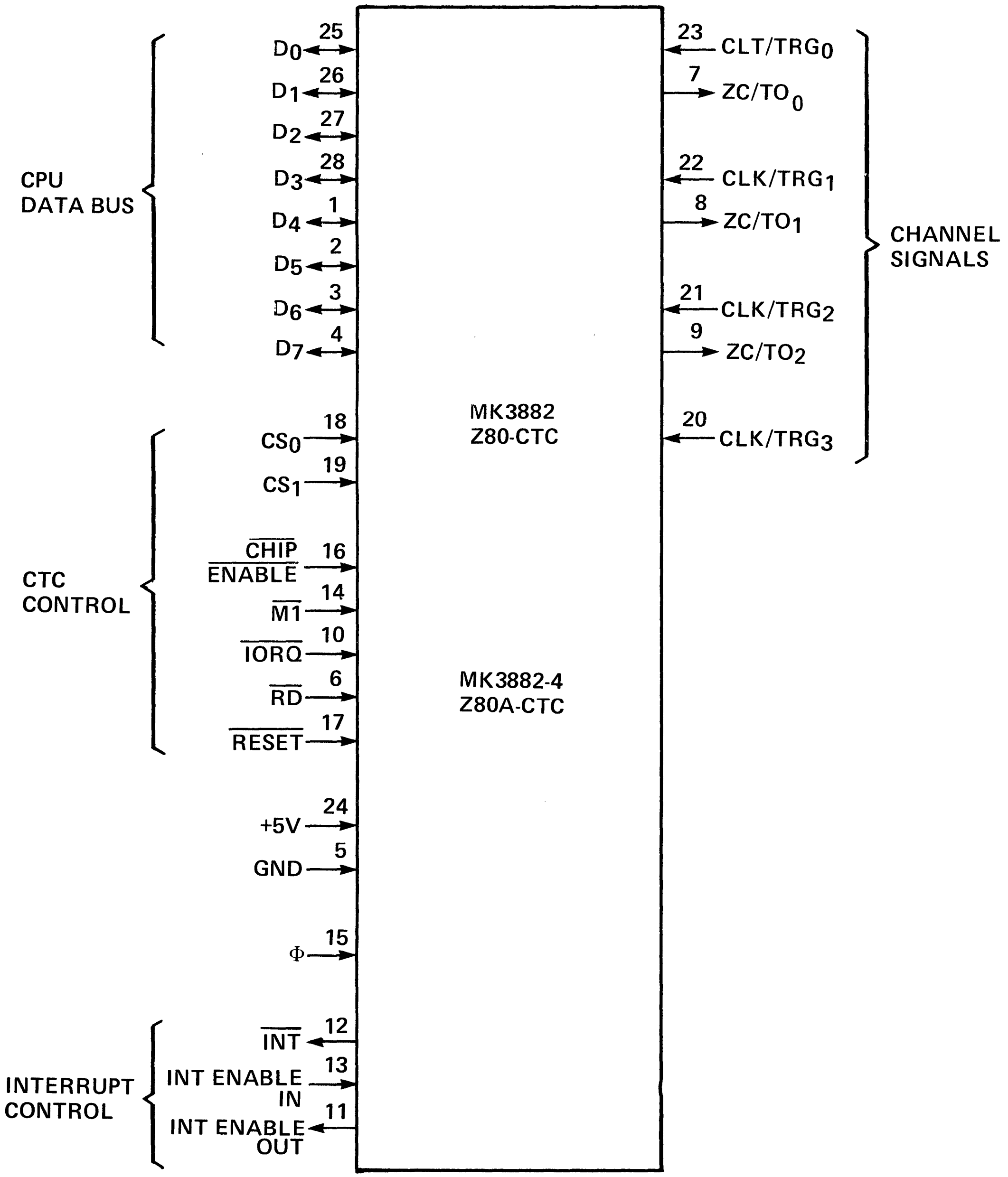
ZC/TO2–ZC/TO0

Zero Count/Timeout (output, active high)

There are three ZC/TO pins, corresponding to CTC channels 2 through 0. (Due to package pin limitations channel 3 has no ZC/TO pin.) In either Counter Mode or Timer Mode, when the Down Counter decrements to zero an active high going pulse appears at this pin.

Z80-CTC PIN CONFIGURATION

Figure 3.0-1



4.0 CTC OPERATING MODES

At power-on, the Z80-CTC state is undefined. Asserting $\overline{\text{RESET}}$ puts the CTC in a known state. Before any channel can begin counting or timing, a Channel Control Word and a time constant data word must be written to the appropriate registers of that channel. Further, if any channel has been programmed to enable interrupts, an Interrupt Vector word must be written to the CTC's Interrupt Control Logic. (For further details, refer to section 5.0: "CTC Programming.") When the CPU has written all of these words to the CTC, all active channels will be programmed for immediate operation in either the Counter Mode or the Timer Mode.

4.1 CTC COUNTER MODE

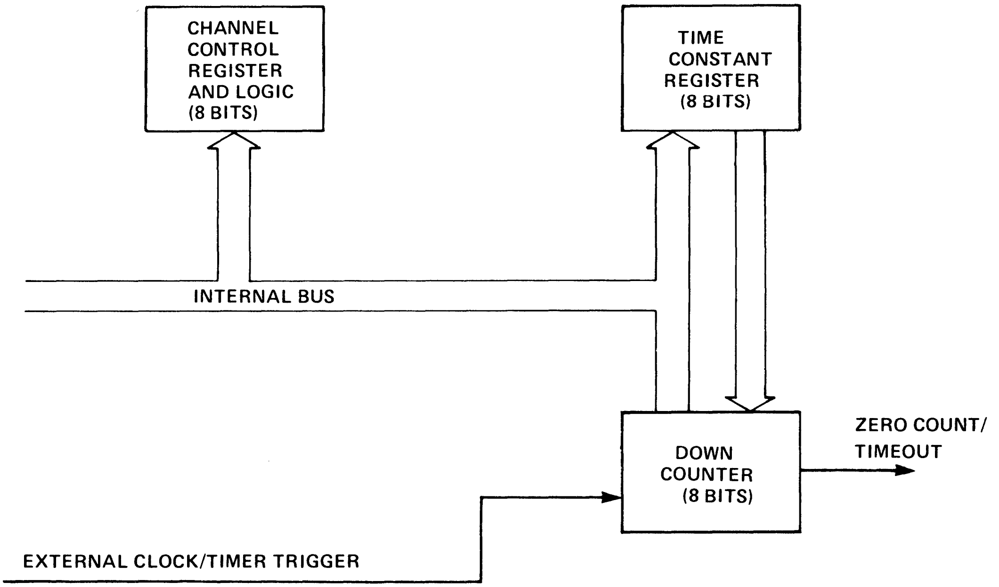
In this mode the CTC counts edges of the CLK/TRG input. The Counter Mode is programmed for a channel when its Channel Control Word is written with bit 6 set. The Channel's External Clock (CLK/TRG) input is monitored for a series of triggering edges; after each, in synchronization with the next rising edge of Φ (the System Clock), the Down Counter (which was initialized with the time constant data word at the start of any sequence of down-counting) is decremented. Although there is no set-up time requirement between the triggering edge of the External Clock and the rising edge of Φ, (Clock), the Down Counter will not be decremented until the following Φ pulse. (See the parameter ts(CK) in section 8.3: "A.C. Characteristics.") A channels's External Clock input is pre-programmed by bit 4 of the Channel Control Word to trigger the decrementing sequence with either a high or a low going edge.

In any of Channels 0, 1, or 2, when the Down Counter is successively decremented from the original time constant until finally it reaches zero, the Zero Count (ZC/TO) output pin for that channel will be pulsed active (high). (However, due to package pin limitations, channel 3 does not have this pin and so may only be used in applications where this output pulse is not required.) Further, if the channel has been so pre-programmed by bit 7 of the Channel Control Word, an interrupt request sequence will be generated. (For more details, see section 7.0: "CTC Interrupt Servicing.")

As the above sequence is proceeding, the zero count condition also results in the automatic reload of the Down Counter with the original time constant data word in the Time Constant Register. There is no interruption in the sequence of continued down-counting. If the Time Constant Register is written to with a new time constant data word while the Down Counter is decrementing, the present count will be completed before the new time constant will be loaded into the Down Counter.

CHANNEL - COUNTER MODE

Figure 4.1-0


CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
DOWN COUNTER (8 BITS)
ZERO COUNT/ TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

4.2 CTC TIMER MODE

In this mode the CTC generates timing intervals that are an integer value of the system clock period. The Timer Mode is programmed for a channel when its Channel Control Word is written with bit 6 reset. The channel then may be used to measure intervals of time based on the System Clock period. The System Clock is fed through two successive counters, the Prescaler and the Down Counter. Depending on the pre-programmed bit 5 in the Channel Control Word, the Prescaler divides the System Clock by a factor of either 16 or 256. The output of the Prescaler is then used as a clock to decrement the Down Counter, which may be pre-programmed with any time constant integer between 1 and 256. As in the Counter Mode, the time constant is automatically reloaded into the Down Counter at each zero-count condition, and counting continues. Also at zero-count, the channel's Time Out (ZC/TO) output (which is the output of the Down Counter) is pulsed, resulting in a uniform pulse train of precise period given by the product.

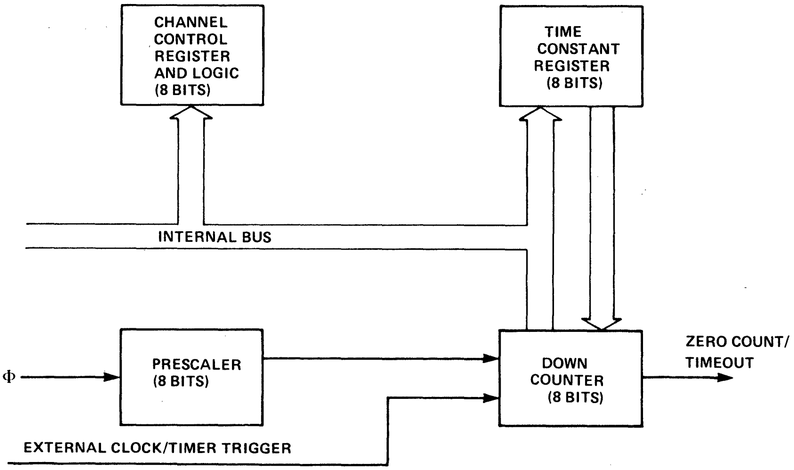
$$t_C * P * TC$$

where tc is the System Clock period, P is the Prescaler factor of 16 or 256 and TC is the pre-programmed time constant.

Bit 3 of the Channel Control Word is pre-programmed to select whether timing will be automatically initiated, or whether it will be initiated with a triggering edge at the channel's Timer Trigger (CLK/TRG) input. If bit 3 is reset the timer automatically begins operation at the start of the CPU cycle following the I/O Write machine cycle that loads the time constant data word to the channel. If bit 3 is set the timer begins operation on the second succeeding rising edge of Φ after the Timer Trigger edge following the loading of the time constant data word. If no time constant data word is to follow then the timer begins operation on the second succeeding rising edge of Φ after the Timer Trigger edge following the control word write cycle. Bit 4 of the Channel Control Word is pre-programmed to select whether the Timer Trigger will be sensitive to a rising or falling edge. Although there is no set-up requirement between the active edge of the Timer Trigger and the next rising edge of Φ. If the Timer Trigger edge occurs closer than a specified minimum set-up time to the rising edge of Φ, the Down Counter will not begin decrementing until the following rising edge of Φ. (See parameter ts(TR) in section 8.3: "A.C. Characteristics".)

If bit 7 in the Channel Control Word is set, the zero-count condition in the Down Counter, besides causing a pulse at the channel's Time Out pin, will be used to initiate an interrupt request sequence, (For more details, see section 7.0: "CTC Interrupt Servicing.")

CHANNEL - TIMER MODE
Figure 4.2-0


CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
Φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/ TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

5.0 CTC PROGRAMMING

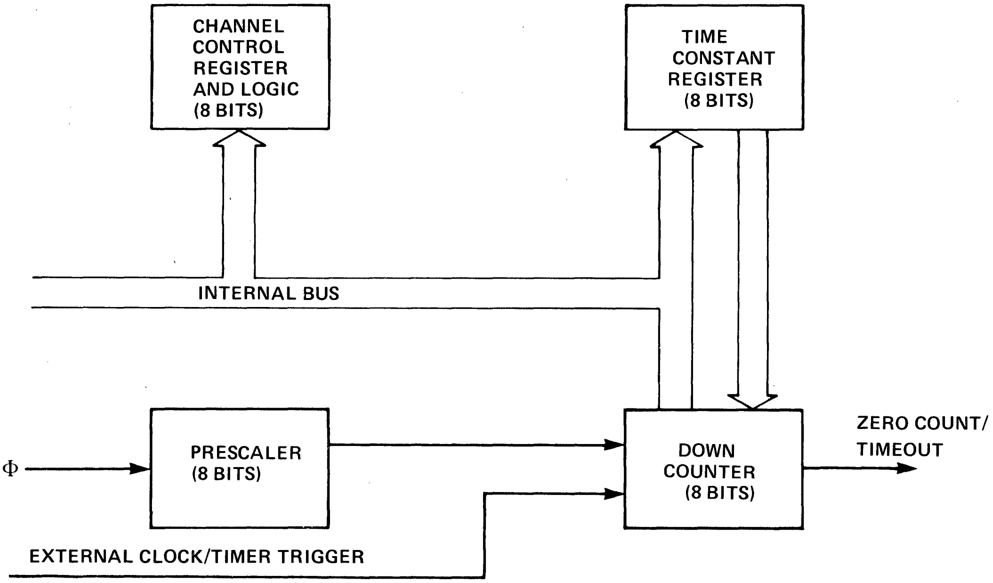
Before a Z80-CTC channel can begin counting or timing operations, a Channel Control Word and a Time Constant data word must be written to it by the CPU. These words will be stored in the Channel Control Register and the Time Constant Register of that channel. In addition, if any of the four channels have been programmed with bit 7 of their Channel Control Words to enable interrupts, an Interrupt Vector must be written to the appropriate register in the CTC. Due to automatic features in the Interrupt Control Logic, one pre-programmed Interrupt Vector suffices for all four channels.

5.1 LOADING THE CHANNEL CONTROL REGISTER

To load a Channel Control Word, the CPU performs a normal I/O Write sequence to the port address corresponding to the desired CTC channel. Two CTC input pins, namely CS0 and CS1, are used to form a 2-bit binary address to select one of four channels within the device. (For a truth table, see section 2.2.1: "The Channel Control Register and Logic".) In many system architectures, these two input pins are connected to Address Bus lines A0 and A1, respectively, so that the four channels in a CTC device will occupy contiguous I/O port addresses. A word written to a CTC channel will be interpreted as a Channel Control Word, and loaded into the Channel Control Register, its bit 0 is a logic 1. The other seven bits of this word select operating modes and conditions as indicated in the diagram below. Following the diagram the meaning of each bit will be discussed in detail.

CHANNEL BLOCK DIAGRAM

Figure 5.1-0


CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
Φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/ TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

CHANNEL CONTROL REGISTER

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT ENABLE	MODE	RANGE	SLOPE	TRIGGER	LOAD TIME CONSTANT	RESET	1

USED IN TIMER MODE ONLY

USED IN TIMER MODE ONLY

5.1 LOADING THE CHANNEL CONTROL REGISTER (CONT'D)

Bit 7 = 1

The channel is enabled to generate an interrupt request sequence every time the Down Counter reaches a zero-count condition. To set this bit to 1 in any of the four Channel Control Registers necessitates that an Interrupt Vector also be written to the CTC before operation begins. Channel interrupts may be programmed in either Counter Mode or Timer Mode. If an updated Channel Control Word is written to a channel already in operation, with bit 7 set, the interrupt enable selection will not be retroactive to a preceding zero-count condition.

Bit 7 = 0

Channel interrupts disabled. Any pending interrupt by that channel will be cleared.

Bit 6 = 1

Counter Mode selected. The Down Counter is decremented by each triggering edge of the External Clock (CLK/TRG) input. The Prescaler is not used.

Bit 6 = 0

Timer Mode selected. The Prescaler is clocked by the System Clock Φ, and the output of the Prescaler in turn clocks the Down Counter. The output of the Down Counter (the channel's ZC/TO output) is a uniform pulse train of period given by the product.

$$t_C * P * TC$$

where t_C is the period of System Clock Φ, P is the Prescaler factor of 16 or 256, and TC is the time constant data word.

Bit 5 = 1

(Defined for Timer Mode only.) Prescaler factor is 256.

Bit 5 = 0

(Defined for Timer Mode only.) Prescaler factor is 16.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
INTERRUPT ENABLE	MODE	RANGE	SLOPE	TRIGGER	LOAD TIME CONSTANT	RESET	1

USED IN TIMER MODE ONLY

USED IN TIMER MODE ONLY

Bit 4 = 1

TIMER MODE - positive edge trigger starts timer operation.
COUNTER MODE - positive edge decrements the down counter.

Bit 4 = 0

TIMER MODE - negative edge trigger starts timer operation.
COUNTER MODE - negative edge decrements the down counter.

5.1 LOADING THE CHANNEL CONTROL REGISTER (CONT'D)

Bit 3 = 1

Timer Mode Only - External trigger is valid for starting timer operation after rising edge of T_2 of the machine cycle following the one that loads the time constant. The Prescaler is decremented 2 clock cycles later if the setup time is met, otherwise 3 clock cycles. Once timer has been started it will free run at the rate determined by the Time Constant register.

Bit 3 = 0

Timer Mode Only - Timer begins operation on the rising edge of T_2 of the machine cycle following the one that loads the time constant.

Bit 2 = 1

The time constant data word for the Time Constant Register will be the next word written to this channel. If an updated Channel Control Word and time constant data word are written to a channel while it is already in operation, the Down Counter will continue decrementing to zero before the new time constant is loaded into it.

Bit 2 = 0

No time constant data word for the Time Constant Register should be expected to follow. To program bit 2 to this state implies that this Channel Control Word is intended to update the status of a channel already in operation, since a channel will not operate without a correctly programmed data word in the Time Constant Register, and a set bit 2 in this Channel Control Word provides the only way of writing to the Time Constant Register.

Bit 1 = 1

Reset channel. Channel stops counting or timing. This is not a stored condition. Upon writing into this bit a reset pulse discontinues current channel operation, however, none of the bits in the channel control register are changed. If both bit 2 = 1 and bit 1 = 1 the channel will resume operation upon loading a time constant.

Bit 1 = 0

Channel continues current operation.

5.2 DISABLING THE CTC'S INTERRUPT STRUCTURE

If an external Asynchronous interrupt could occur while the processor is writing the disable word to the CTC (01H); a system problem may occur. If interrupts are enabled in the processor it is possible that the Asynchronous interrupt will occur while the processor is writing the disable word to the CTC. The CTC will generate an INT and the CPU will acknowledge it, however, by this time, the CTC will have received the disable word and de-activated its interrupt structure. The result is that the CTC will not send in its interrupt vector during the interrupt acknowledge cycle because it is disabled and the CPU will fetch an erroneous vector resulting in a program fault. The cure for this problem is to disable interrupts within the CPU with the DI instruction just before the CTC is disabled and then re-enable interrupts with the EI instruction. This action causes the CPU to ignore any interrupts produced by the CTC while it is being disabled. The code sequence would be:

```
LD  A, 01H
DI              ; DISABLE CPU
OUT (CTC), A    ; DISABLE CTC
EI              ; ENABLE CPU
—
—
```

5.3 LOADING THE TIME CONSTANT REGISTER

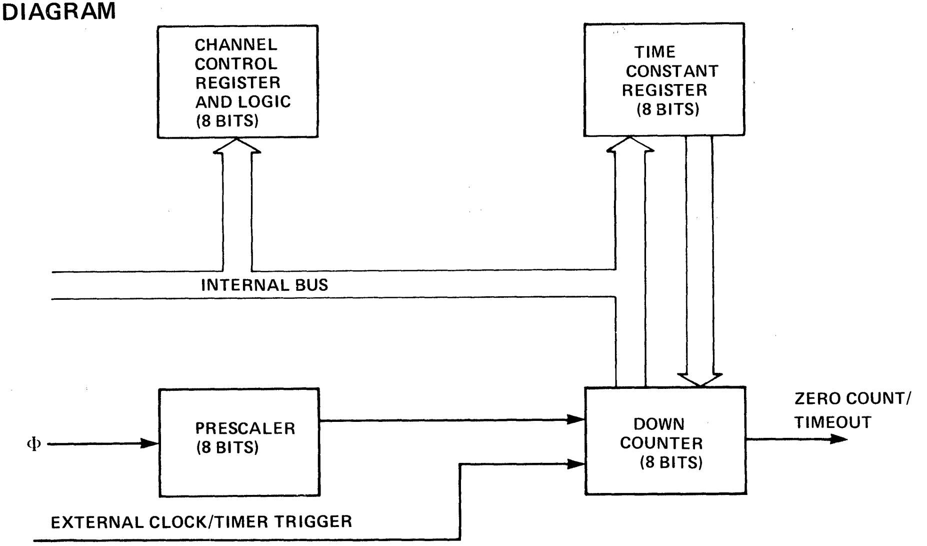
A channel may not begin operation in either Timer Mode or Counter Mode unless a time constant data word is written into the Time Constant Register by the CPU. This data word will be expected on the next I/O Write to this channel following the I/O Write of the Channel Control Word, provided that bit 2 of the Channel Control Word is set. The time constant data word may be an integer value in the range 1-256. If all eight bits in this word are zero, it is interpreted as 256. If a time constant data word is loaded to a channel already in operation, the Down Counter will continue decrementing to zero before the new time constant is loaded from the Time Constant Register to the Down Counter.

TIME CONSTANT REGISTER

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
TC_7	TC_6	TC_5	TC_4	TC_3	TC_2	TC_1	TC_0
MSB							LSB

CHANNEL BLOCK DIAGRAM

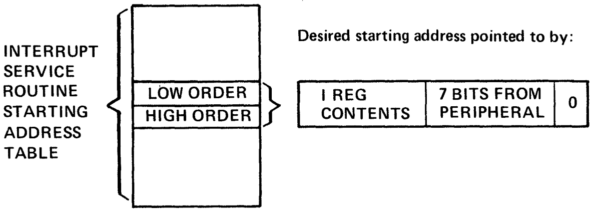
Figure 5.3-0


CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
PRESCALER (8 BITS)
Φ
DOWN COUNTER (8 BITS)
ZERO COUNT/ TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

5.4 LOADING THE INTERRUPT VECTOR REGISTER

The Z80-CTC has been designed to operate with the Z80-CPU programmed for mode 2 interrupt response. Under the requirements of this mode, when a CTC channel requests an interrupt and is acknowledged, a 16-bit pointer must be formed to obtain a corresponding interrupt service routine starting address from a table in memory. The upper 8 bits of this pointer are provided by the CPU's I register, and the lower 8 bits of the pointer are provided by the CTC in the form of an Interrupt Vector unique to the particular channel that requested the interrupt. (For further details, see section 7.0: "CTC Interrupt Servicing".)

MODE 2 INTERRUPT OPERATION


INTERRUPT SERVICE ROUTINE STARTING ADDRESS TABLE
LOW ORDER
HIGH ORDER
Desired starting address pointed to by:
I REG CONTENTS
7 BITS FROM PERIPHERAL
0

5.4 LOADING THE INTERRUPT VECTOR REGISTER (Cont'd)

The high order 5 bits of this Interrupt Vector must be written to the CTC in advance as part of the initial programming sequence. To do so, the CPU must write to the I/O port address corresponding to the CTC channel 0, just as it would if a Channel Control Word were being written to that channel, except that bit 0 of the word being written must contain a 0. (As explained above in section 5.1, if bit 0 of a word written to a channel were set to 1, the word would be interpreted as a Channel Control Word, so a 0 in bit 0 signals the CTC to load the incoming word into the Interrupt Vector Register.) Bits 1 and 2, however are not used when loading this vector. At the time when the interrupting channel must place the Interrupt Vector on the Z80 Data Bus, the Interrupt Control Logic of the CTC automatically supplies a binary code in bits 1 and 2 indentifying which of the four CTC channels is to be serviced.

INTERRUPT VECTOR REGISTER

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3	X	X	0

SUPPLIED BY USER

D_2	D_1		
0	0	CHANNEL 0	(Highest Priority)
0	1	CHANNEL 1	
1	0	CHANNEL 2	
1	1	CHANNEL 3	(Lowest Priority)

AUTOMATICALLY INSERTED BY Z80-CTC

6.0 CTC TIMING

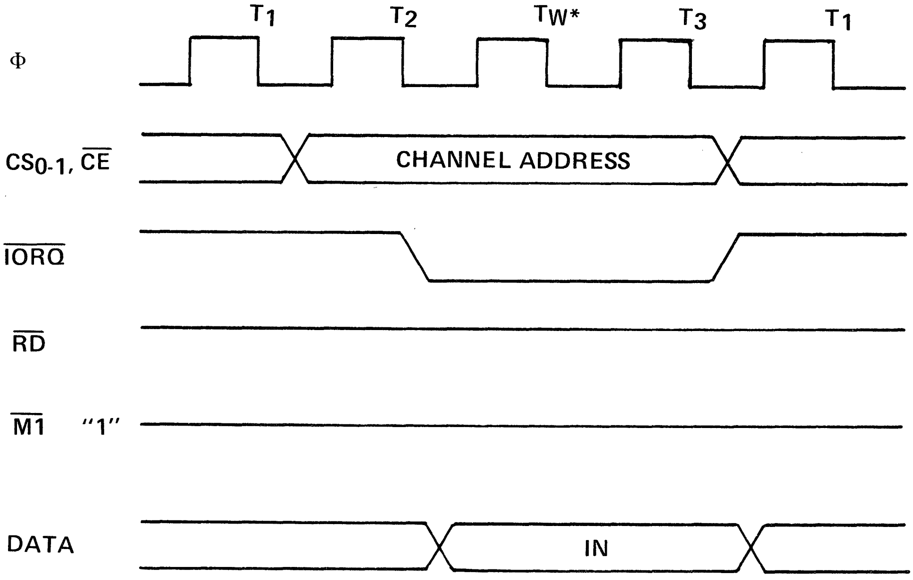
This section illustrates the timing relationships of the relevant CTC pins for the following types of operation: writing a word to the CTC, reading a word from the CTC, counting, and timing. Elsewhere in this manual may be found timing diagrams relating to interrupt servicing (section 7.0) and an A.C. Timing Diagram which quantitatively specifies the timing relationships (section 8.4).

6.1 CTC WRITE CYCLE

Figure 6.1-0 illustrates the timing associated with the CTC Write Cycle. This sequence is applicable to loading either a Channel Control Word, an Interrupt Vector, or a time constant data word.

In the sequence shown, during clock cycle T_1, the Z80-CPU prepares for the Write Cycle with a false (high) signal at CTC input pin $\overline{RD}$ (Read). Since the CTC has no separate Write signal input, it generates its own internally form the false $\overline{RD}$ input. Later, during clock cycle T_2, the Z80-CPU initiates the Write Cycle with true (low) signals at CTC input pins $\overline{IORQ}$ (I/O Request) and $\overline{CE}$ (Chip Enable). (Note: $\overline{M1}$ must be false to distinguish the cycle form an interrupt acknowledge.) Also at this time a 2-bit binary code appears at CTC inputs CS1 and CS0 (Channel Select 1 and 0), specifying which of the four CTC channels is being written to, and the word being written appears on the Z80 Data Bus. Now everything is ready for the word to be latched into the appropriate CTC internal register in synchronization with the rising edge beginning clock cycle T_3. No additional wait states are allowed.

CTC WRITE CYCLE
Figure 6.1-0

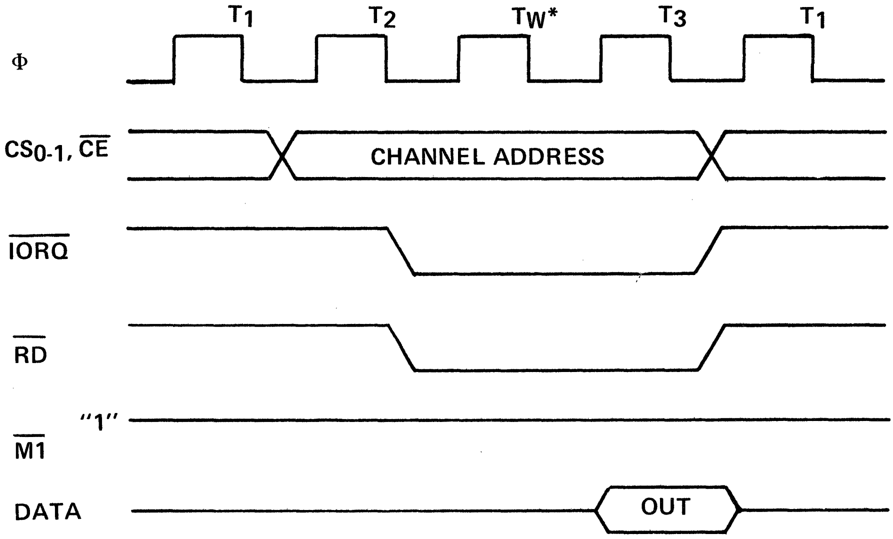


***AUTOMATICALLY INSERTED BY Z80-CPU**

6.2 CTC READ CYCLE

Figure 6.2-0 illustrates the timing associated with the CTC Read Cycle. This sequence is used any time the CPU reads the current contents of the Down Counter. During clock cycle T_2, the Z80-CPU initiates the Read Cycle with true signals at input pins $\overline{RD}$ (Read), $\overline{IORQ}$ (I/O Request), and $\overline{CE}$ (Chip Enable). also at this time a 2-bit binary code appears at CTC inputs CS1 and CS0 (Channel Select 1 and 0), specifying which of the four CTC channels is being read from. (Note: $\overline{M1}$ must be false to distinguish the cycle form an interrupt acknowledge.) On the rising edge of the cycle T_3 the valid contents of the Down Counter as of the rising edge of cycle T_2 will be available on the Z80 Data Bus. No additional wait states are allowed.

CTC READ CYCLE
Figure 6.2-0

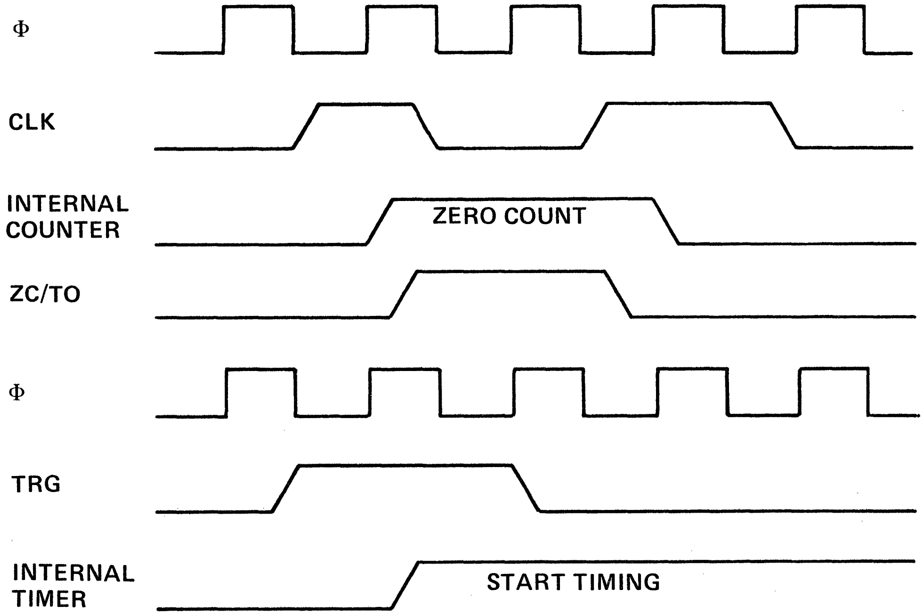


*AUTOMATICALLY INSERTED BY Z80-CPU

6.3 CTC COUNTING AND TIMING

Figure 6.3-0 illustrates the timing diagram for the CTC Counting and Timing Modes.

CTC COUNTING AND TIMING
Figure 6.3-0



6.3 CTC COUNTING AND TIMING (Cont'd)

In the Counter Mode, the edge (rising edge is active in this example) form the external hardware connected to pin CLK/TRG decrements the Down Counter in synchronization with the System Clock Φ. As specified in the A.C. Characteristics (Section 9.1) this CLK/TRG pulse must have a minimum width and the minimum period must not be less than twice the system clock period. Although there is no set-up requirement between the active edge of the CLK/TRG and the rising edge of Φ if the CLK/TRG edge occurs closer than a specified minimum time, the decrement of the Down Counter will be delayed one cycle of Φ. Immediately after the decrement of the Down Counter, 1 to 0, the ZC/TO output is pulsed true.

In the Timer Mode, a pulse trigger (user-selectable as either active high or active low) at the CLK/TRG pin enables timing function on the second succeeding rising edge of Φ. As in the Counter Mode, the triggering pulse is detected asynchronously and must have a minimum width. The timing function is initiated in syncronization with Φ, and a minimum set-up time is required between the active edge of the CLK/TRG and the next rising edge of Φ. If the CLK/TRG edge occurs closer than this, the initiation of the timer function will be delayed one cycle of Φ.

7.0 CTC INTERRUPT SERVICING

Each CTC channel may be individually programmed to request an interrupt every time its Down Counter reaches a count of zero. The purpose of a CTC-generated interrupt, as for any other peripheral device, is to force the CPU to execute an interrupt service routine. To utilize this feature the Z80-CPU must be programmed for mode 2 interrupt response. Under the requirements of this mode, when a CTC channel requests an interrupt and is acknowledged, a 16-bit pointer must be formed to obtain a corresponding interrupt service routine starting address from a table in memory. The lower 8 bits of the pointer are provided by the CTC in the form of an Interrupt Vector unique to the particular channel that requested the interrupt. (For further details, refer to Chapter 8.0 of the Z80-CPU Technical Manual.)

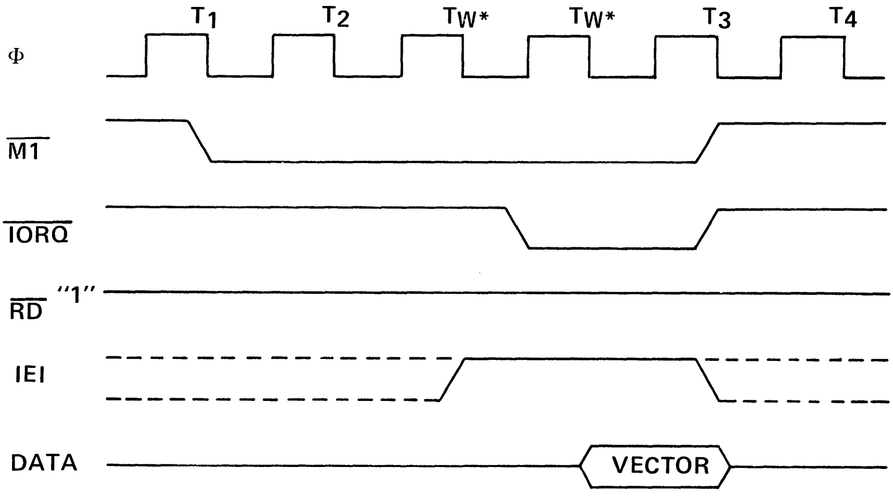
The CTC's Interrupt Control Logic insures that it acts in accordance with Z80 system interrupt protocol for nested priority interrupt and proper return from interrupt. The priority of any system device is determined by its physical location in a daisy chain configuration. Two signal lines (IEI and IEO) are provided in the CTC and all Z80 peripheral devices to form the system daisy chain. The device closest to the CPU has the highest priority; within the CTC, interrupt priority is predetermined by channel number, with channel 0 having highest priority. According to Z80 system interrupt protocol, low priority devices or channels may not interrupt higher priority devices or channels that have already interrupted and not had their interrupt service routines completed. However, high priority devices or channels may interrupt the servicing of lower priority devices or channels. (For further details, see section 2.3: "Interrupt Control Logic".)

Sections 7.1 and 7.2 below describe the nominal timing relationships of the relevant CTC pins for the Interrupt Acknowledge Cycle and the Return form Interrupt Cycle. Section 7.3 below discusses a typical example of daisy chain interrupt servicing.

7.1 INTERRUPT ACKNOWLEDGE CYCLE

Figure 7.1-0 illustrates the timing associated with the Interrupt Acknowledge Cycle. Some time after an interrupt is requested by the CTC, the CPU will send out an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). To insure that the daisy chain enable lines stabilize, channels are inhibited from changing their interrupt request status when $\overline{M1}$ is active. $\overline{M1}$ is active about two clock cycles earlier than $\overline{IORQ}$, and $\overline{RD}$ is false to distinguish the cycle from an instruction fetch. During this time the interrupt logic of the CTC will determine the highest priority interrupting channel within the CTC places its Interrupt Vector onto the Data Bus when $\overline{IORQ}$ goes active. Two wait states (T_W*) are automatically inserted at this time to allow the daisy chain to stablize. Additional wait states may be added.

INTERRUPT ACKNOWLEDGE CYCLE
Figure 7.1-0

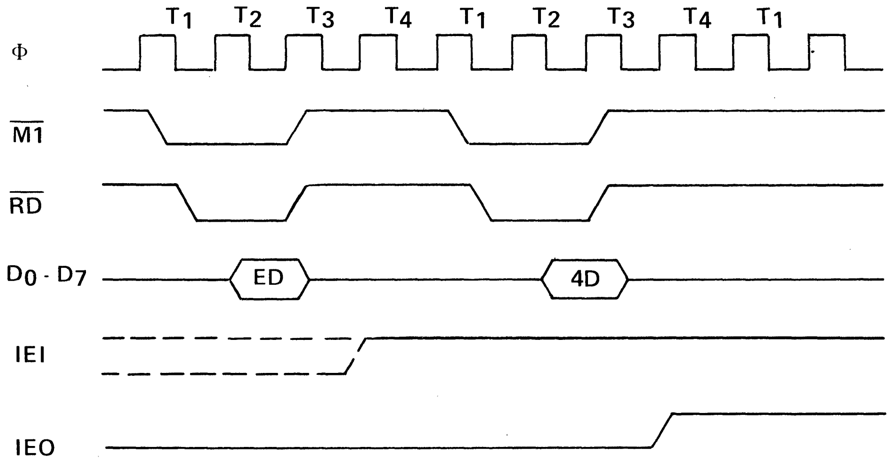


7.2 RETURN FROM INTERRUPT CYCLE

Figure 7.2-0 illustrates the timing associated with the RETI Instruction. This instruction is used at the end of an interrupt service routine to initialize the daisy chain enable lines for proper control of nested priority interrupt handling. The CTC decodes the two-byte RETI code internally and determines whether it is intended for a channel being serviced.

When several Z80 peripheral chips are in the daisy chain IEI will become active on the chip currently under service when an EDH opcode is decoded. If the following opcode is 4DH, the peripheral being serviced will be re-initialized and its IEO will become active. Additional wait states are allowed.

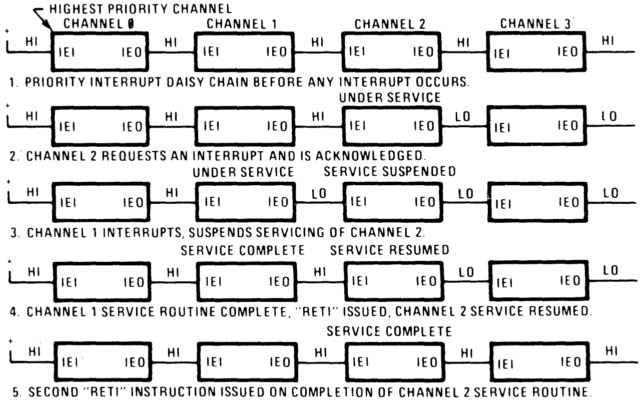
RETURN FROM INTERRUPT CYCLE
Figure 7.2-0



7.3 DAISY CHAIN INTERRUPT SERVICING

Figure 7.3-0 illustrates a typical nested interrupt sequence which may occur in the CTC. In this example, channel 2 interrupts and is granted service. While this channel is being serviced, higher priority channel 1 interrupts and is granted service. The service routine for the higher priority channel is completed, and a RETI instruction (see section 7.2 for further details) is executed to signal the channel that its routine is complete. At this time, the service routine of the lower priority channel 2 is resumed and completed.

DAISY CHAIN INTERRUPT SERVICING
Figure 7.3-0



HIGHEST PRIORITY CHANNEL
CHANNEL 0
CHANNEL 1
CHANNEL 2
CHANNEL 3
+
HI
IEI
IEO
1. PRIORITY INTERRUPT DAISY CHAIN BEFORE ANY INTERRUPT OCCURS.
UNDER SERVICE
LO
2. CHANNEL 2 REQUESTS AN INTERRUPT AND IS ACKNOWLEDGED.
UNDER SERVICE
SERVICE SUSPENDED
3. CHANNEL 1 INTERRUPTS, SUSPENDS SERVICING OF CHANNEL 2.
SERVICE COMPLETE
SERVICE RESUMED
4. CHANNEL 1 SERVICE ROUTINE COMPLETE, "RETI" ISSUED, CHANNEL 2 SERVICE RESUMED.
SERVICE COMPLETE
5. SECOND "RETI" INSTRUCTION ISSUED ON COMPLETION OF CHANNEL 2 SERVICE ROUTINE.

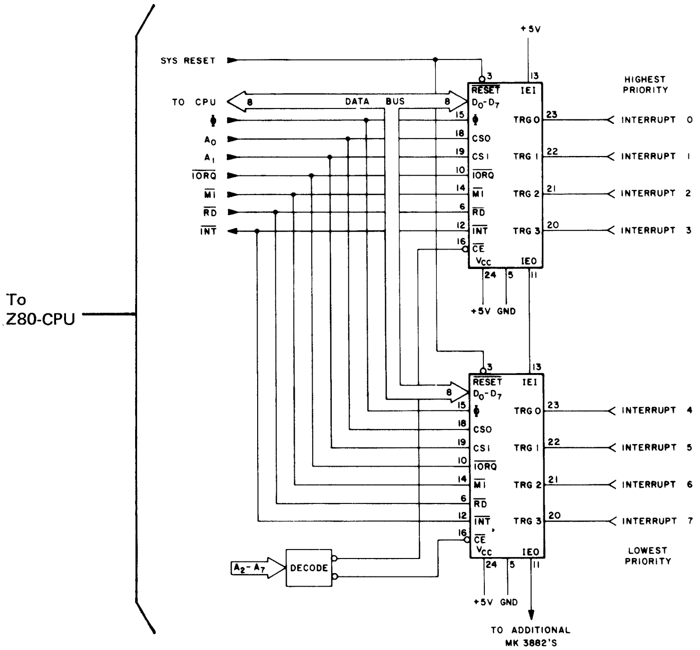
7.4 USING THE CTC AS AN INTERRUPT CONTROLLER

All of the Z80 family parts contain circuitry for prioritizing interrupts and supplying the vector to the CPU. However, in many Z80 based systems interrupts must be processed from devices which do not contain this interrupt circuitry. To handle this requirement the MK3882 CTC can be used, providing prioritized, independently vectored, maskable, edge selectable, count programmable external interrupt inputs. The MK3882 parts may be cascaded, expanding the system to as many as 256 interrupt inputs.

Each MK3882 contains 4 channels with counter inputs able to interrupt upon one or more (up to 256) edge transitions. The active transition may be programmed to be positive or negative. Each of the 4 channels has a programmable vector which is used in powerful Z80 mode 2 interrupt processing. When an interrupt is processed the vector is combined with the CPU I register to determine where the interrupt service routine start address is located. Additionally, priority resolution is handled within the MK3882 when more than one interrupt request is made simultaneously. When more than one MK3882 is used, the prioritizing is done, with the IEI/IEO chain resolving inter-chip priorities. Each channel can be independently "masked" by disabling that channel's local interrupt.

When programming the MK3882 to handle an input as a general purpose interrupt line, the channel is put in the counter mode, with the count set to 1, the active edge specified and the vector is loaded. When the programmed edge occurs a mode 2 interrupt will be generated by the CTC and the Z80-CPU can vector directly to the service routine for the non-Z80 peripheral device. Note that after the interrupt, the CTC down counter is automatically reloaded with a count of one and the CTC begins looking for another active edge. The second interrupt will not be passed on to the CPU until after the RETI of the first interrupts service routine.

CTC AS AN INTERRUPT CONTROLLER
Figure 7.4-0



8.0 ABSOLUTE MAXIMUM RATINGS

Temperature Under Bias	Specified Operating Range
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	0.8V

8.1 D. C. CHARACTERISTICS

TA = 0°C to 70°C, Vcc = 5V ± 5% unless otherwise specified

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITION
V_{ILC}	Clock Input Low Voltage	–0.3	.45	V	
V_{IHC}	Clock Input High Voltage (1)	V_{CC}–.6	V_{CC} +.3	V	
V_{IL}	Input Low Voltage	–0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = –250 μA
I_{CC}	Power Supply Current		120	mA	T_C = 400 nsec**
I_{LI}	Input Leakage Current		10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current In Float		10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current In Float		–10	μA	V_{OUT} = 0.4V
I_{OHD}	Darlington Drive Current	–1.5		mA	V_{OH} = 1.5V

**T_C = 250 nsec for MK 3882-4

8.2 CAPACITANCE

TA = 25°C, f = 1 MHz

SYMBOL	PARAMETER	MAX	UNIT	TEST CONDITION
C_{Φ}	Clock Capacitance	20	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

*COMMENT

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

8.3 A.C. CHARACTERISTICS MK 3882, MK 3882-10, Z80-CTC

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	t_C	Clock Period	400	(1)	ns	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	170	2000	ns	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	170	2000	ns	
	tr, tf	Clock Rise and Fall Times		30	ns	
	t_H	Any Hold Time for Specified Setup Time	0		ns	
CS, $\overline{\text{CE}}$, etc.	$t_S\Phi(CS)$	Control Signal Setup Time to Rising Edge of Φ During Read or Write Cycle	160		ns	
D_0-D_7	$t_{DR}(D)$	Data Output Delay from Rising Edge of $\overline{\text{RD}}$ During Read Cycle		480	ns	(2)
	$t_S\Phi(D)$	Data Setup Time to Rising Edge of Φ During Write or M1 Cycle	50		ns	
	$t_{DI}(D)$	Data Output Delay from Falling Edge of IORQ During INTA Cycle		340	ns	(2)
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		230	ns	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{\text{IORQ}}$ During INTA Cycle	200		ns	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		220	ns	(3)
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		190	ns	(3)
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{\text{M1}}$ (Interrupt Occurring just Prior to $\overline{\text{M1}}$)		300	ns	(3)
$\overline{\text{IORQ}}$	$t_S\Phi(IR)$	$\overline{\text{IORQ}}$ Setup Time to Rising Edge of Φ During Read or Write Cycle	250		ns	
$\overline{\text{M1}}$	$t_S\Phi(M1)$	$\overline{\text{M1}}$ Setup Time to Rising Edge of Φ During INTA or M1 Cycle	210		ns	
$\overline{\text{RD}}$	$t_S\Phi(RD)$	$\overline{\text{RD}}$ Setup Time to Rising Edge of Φ During Read or M1 Cycle	240		ns	
$\overline{\text{INT}}$	$t_{DCK}(IT)$	$\overline{\text{INT}}$ Delay Time from Rising Edge of CLK/TRG		$2t_C(\Phi) + 200$		Counter Mode
	$t_D\Phi(IT)$	$\overline{\text{INT}}$ Delay Time from Rising Edge of Φ		$t_C(\Phi) + 200$		Timer Mode
CLK/$TRG_{0\text{-}3}$	$t_C(CK)$	Clock Period	$2t_C(\Phi)$			Counter Mode
	t_r, tf	Clock and Trigger Rise and Fall Times		50	ns	
	$t_S(CK)$	Clock Setup Time to Rising Edge of Φ for Immediate Count	210		ns	Counter Mode
	$t_S(TR)$	Trigger Setup Time to Rising Edge of Φ for Enabling of Prescaler on Following Rising Edge of Φ	210		ns	Timer Mode
	$t_W(CTH)$	Clock and Trigger High Pulse Width	200		ns	Counter and Timer Modes
	$t_W(CTL)$	Clock and Trigger Low Pulse Width	200		ns	Counter and Timer Modes
$ZC/TO_{0\text{-}2}$	$t_{DH}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO High		190	ns	Counter and Timer Modes
	$t_{DL}(ZC)$	ZC/TO Delay Time from Falling Edge of Φ , ZC/TO Low		190	ns	Counter and Timer Modes

NOTES:

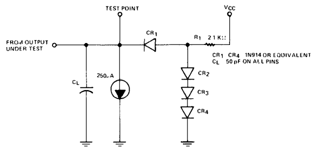
(1) $t_C = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$.

(2) Increase delay by 10 nsec for each 50 pF increase in loading 200pF maximum for data lines and 100pF for control lines.

(3) Increase delay by 2nsec for each 10pF increase in loading, 100pF maximum.

(4) $\overline{\text{RESET}}$ must be active for a minimum of 3 clock cycles.

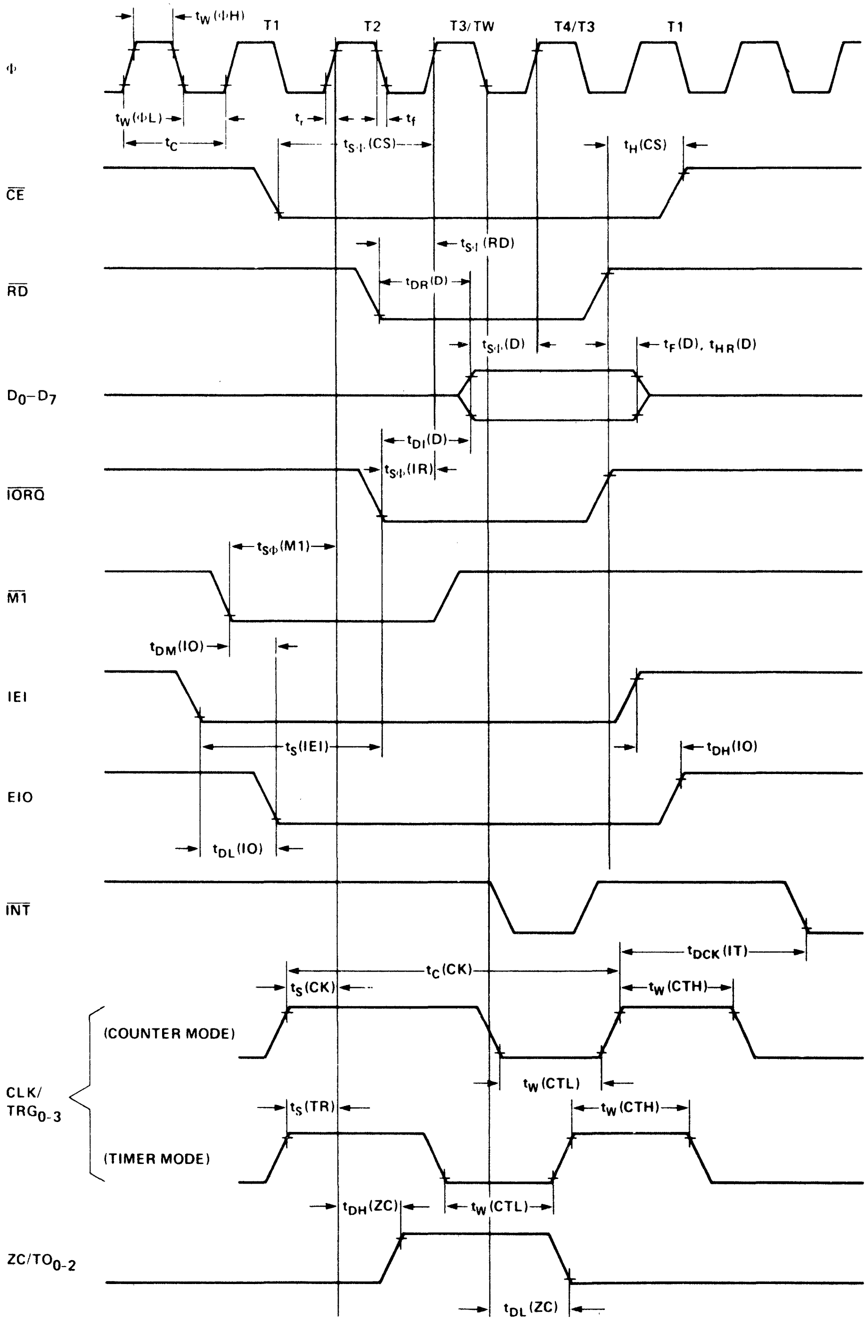
OUTPUT LOAD CIRCUIT



8.4 A. C. TIMING DIAGRAM

Timing measurements are made at the following voltages, unless otherwise specified

	"1"	"0"
CLOCK	V_{CC} - .6V	45V
OUTPUT	2.0V	.8V
INPUT	2.0V	.8V
FLOAT	ΔV	+0.5V



8.5 A.C. CHARACTERISTICS MK 3882-4, Z80A-CTC

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	t_C	Clock Period	250	(1)	ns	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	105	2000	ns	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	105	2000	ns	
	t_r, t_f	Clock Rise and Fall Times		30	ns	
	t_H	Any Hold Time for Specified Setup Time	0		ns	
CS, $\overline{CE}$, etc	$t_S\Phi(CS)$	Control Signal Setup Time to Rising Edge of Φ During Read or Write Cycle	145		ns	
D_0-D_7	$t_{DR}(D)$	Data Output Delay from Falling Edge of $\overline{RD}$ During Read Cycle		380	ns	(2)
	$t_S\Phi(D)$	Data Setup Time to Rising Edge of Φ During Write or M1 Cycle	50		ns	
	$t_{DI}(D)$	Data Output Delay form Falling Edge of IORG During INTA Cycle		160	ns	(2)
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		110	ns	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		ns	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		160	ns	(3)
	$t_{DL}(10)$	IEO Delay Time from Falling Edge of IEI		130	ns	(3)
	$t_{DM}(10)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring just Prior to $\overline{M1}$)		190	ns	(3)
$\overline{IORQ}$	$t_S\Phi(IR)$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Read or Write Cycle	115		ns	
$\overline{M1}$	$t_S\Phi(M1)$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or M1 Cycle	90		ns	
$\overline{RD}$	$t_S\Phi(RD)$	$\overline{RD}$ Setup Time to Rising Edge of Φ During Read or M1 Cycle	115		ns	
$\overline{INT}$	$t_{DCK}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of CLK/TRG		$2t_C(\Phi) + 140$		Counter Mode
	$t_D\Phi(IT)$	$\overline{INT}$ Delay Time from Rising Edge of Φ		$t_C(\Phi) + 140$		Timer Mode
CLK/TRG_{0-3}	$t_C(CK)$	Clock Period	$2t_C(\Phi)$		ns	Counter Mode
	t_r, t_f	Clock and Trigger Rise and Fall Times		30		
	$t_S(CK)$	Clock Setup Time to Rising Edge of Φ for Immediate Count	130		ns	Counter Mode
	$t_S(TR)$	Trigger Setup Time to Rising Edge of Φ for enabling of Prescaler on Following Rising Edge of Φ	130		ns	Timer Mode
	$t_W(CTH)$	Clock and Trigger High Pulse Width	120		ns	Counter and Timer Modes
	$t_W(CTL)$	Clock and Trigger Low Pulse Width	120		ns	Counter and Timer Modes
ZC/TO0-2	$t_{DH}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO High		120	ns	Counter and Timer Modes
ZC/TO_{0-2}	$t_{DL}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO Low		120	ns	Counter and Timer Modes

NOTES:

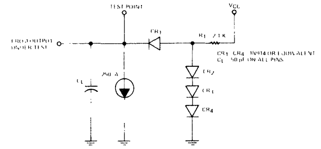
(1.) $t_C = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$.

(2.) Increase delay by 10 nsec for each 50 pF increase in loading, 200pF maximum for data lines and 100pF for control lines.

(3.) Increase delay by 2nsec for each 10pF increase in loading, 100pF maximum.

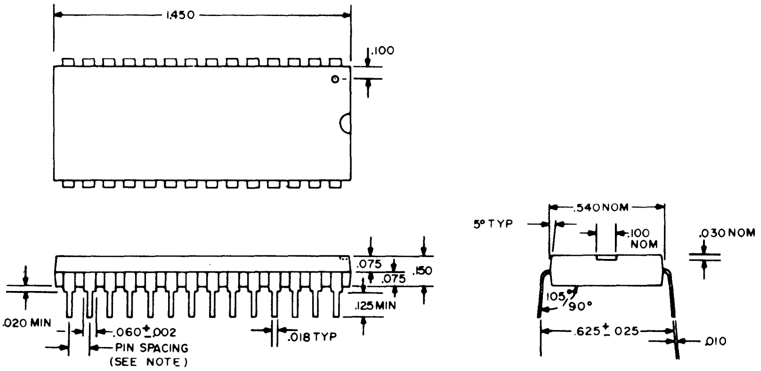
(4.) $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

OUTPUT LOAD CIRCUIT



8.6 PACKAGE DESCRIPTION AND ORDERING INFORMATION

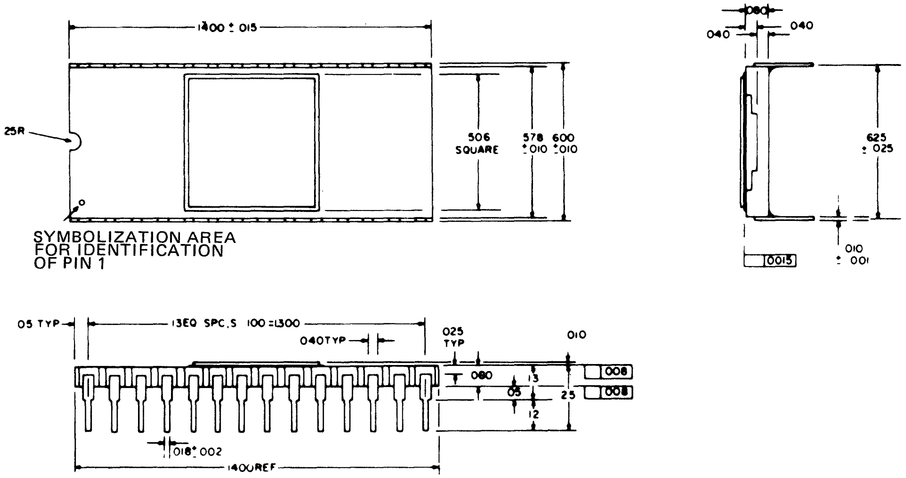
PACKAGE DESCRIPTION - 28 Pin Dual-In-Line Plastic Package



NOTE:

1. The true position pin spacing is 0.100 between center lines. Each pin centerline is located within ± .0100 of its true longitudinal position relative to pins 1 and 28.

PACKAGE DESCRIPTION - 28 Pin Dual-In-Line Ceramic Package



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	MAX CLOCK FREQUENCY	TEMPERATURE RANGE
MK3882N Z80 - CTC	Plastic	2.5 MHz	0° to + 70° C
MK3882P Z80 - CTC	Ceramic	2.5 MHz	
MK3882N - 4 Z80A - CTC	Plastic	4.0 MHz	
MK3882P - 4 Z80A - CTC	Ceramic	4.0 MHz	
MK3882N - 10 Z80 - CTC	Plastic	2.5 MHz	-40° C to +85° C
MK3882P - 10 Z80 - CTC	Ceramic	2.5 MHz	-40° C to +85° C

Z80
Family

PRELIMINARY

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Direct Memory Access MK3883

FEATURES

□ Three classes of operation:
—Transfer Only
—Search Only
—Search-Transfer

□ Address and Block Length Registers fully buffered. Values for next operation may be loaded without disturbing current values.

□ Dual addresses generated during a transfer (one for read port and one for write).

□ Programmable data transfers and searches, automatically incrementing or decrementing the port addresses from programmed starting addresses (they can also remain fixed).

□ Four modes of operation:
—Byte-at-a-time: One byte transferred per request
—Burst: Continues as long as ports are ready
—Continuous: Locks out CPU until operation complete
—Transparent: Steals refresh cycles

□ Timing may be programmed to match the speed of any port.

□ Interrupts on Match Found, End of Block, or Ready, may be programmed.

□ An entire previous operation may be repeated automatically or on command. (Auto restart or Load)

□ The DMA can signal when a specified number of bytes has been transferred, without halting transfer.

□ Multiple DMA's easily configured for rotating priority.

□ The channel may be enabled, disabled or reset under software control.

□ Complete channel status upon program (CPU) request.

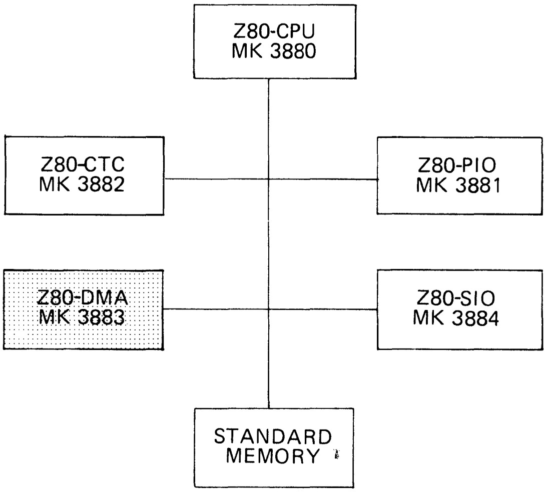
□ Up to 1.25 megabyte Search or Transfer Rate.

□ Diasy-chain priority interrupt and bus acknowledge included to provide automatic interrupt vectoring and bus request control, without need for additional external logic.

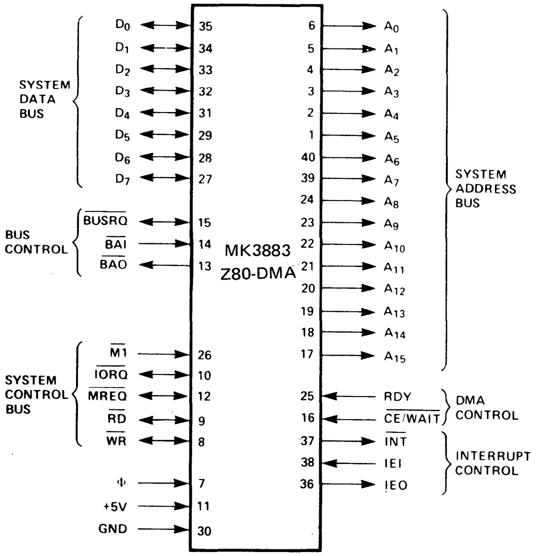
□ TTL compatible inputs and outputs

□ The CPU can read current Port counters, byte counters, or status. A mask word can be set which defines which registers can be accessed during read operations.

Z80 FAMILY



PIN DESCRIPTION



DESCRIPTION

Mostek's Z80 microcomputer product line includes a third generation LSI component set, development systems and support software. The component set includes all the logic circuits necessary for the user to build high performance microcomputer systems with virtually no external logic and a minimal number of standard low-cost memory components. The Z80-DMA (Direct Memory Access) circuit is a programable single-channel device which provides all address, timing and control signals to effect the transfer of blocks of data between two ports withing a Z80-CPU based system. These ports may be either system main memory or any system peripheral I/O device. The DMA can also search a block of data for a particular byte (bit maskable), with or without a simultaneous transfer.

STRUCTURE

- N-channel Silicon Gate Depletion Load Technology
- 40 Pin DIP
- Single 5 volt supply
- Single phase 5 volt clock
- Single channel, two port

DMA ARCHITECTURE

A block diagram of the Z80 DMA is shown in Figure 1. The internal structure consists of the following circuitry:

BUS INTERFACE: provides driver and receiver circuitry to interface to the Z80-CPU Bus.

CONTROL LOGIC AND REGISTERS: set the class, mode and other basic control parameters of the DMA.

ADDRESS, BYTE COUNT/AND PULSE CIRCUITRY: generates the proper port addresses for the read and write operations, with provisions for incrementing or decrementing the address. When zero bytes remain to be handled, the byte count circuitry sets a flag in the status register. Pulse circuitry generates a pulse each time the byte counter lower 8-bits equal the pulse reg.

TIMING CIRCUITRY: allows the user to completely specify the read/write timing for both of the channels' addressed ports.

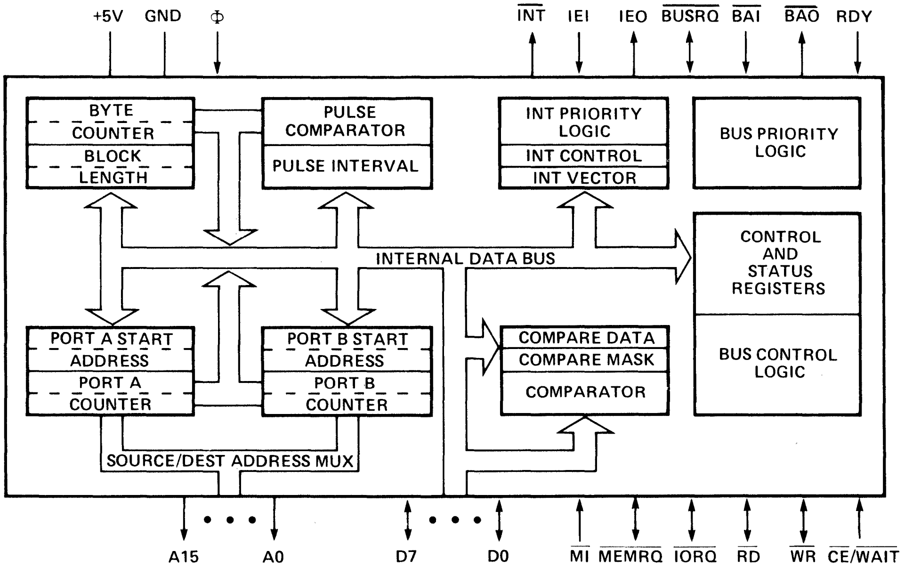
MATCH CIRCUITRY: holds the match byte and a mask byte which allows for the comparison of only certain bits within the byte. If a match is encountered during a Search or Transfer, this circuitry sets a flag in the status register.

INTERRUPT AND $\overline{\text{BUSRQ}}$ CIRCUITRY: includes a control register which specifies the conditions under which the DMA can generate an interrupt; priority encoding logic to select between the generation of an $\overline{\text{INT}}$ or $\overline{\text{BUSRQ}}$ output under these conditions; and an interrupt vector register for automatic vectoring to the interrupt service routine.

STATUS REGISTER: holds current status of DMA.

DMA INTERNAL BLOCK DIAGRAM

Figure 1



REGISTER DESCRIPTION

The following DMA-internal registers are available to the programmer:

CONTROL REGISTERS: Hold DMA control information : such as, when to initiate an interrupt or pulse, what mode or class of operation to perform, etc. (Write Only) (8 Bits)

TIMING REGISTERS: Hold read/write timing parameters for the two ports. (Write Only) (8 bits)

INTERRUPT VECTOR/REGISTER: Holds the 8-bit vector that the DMA will put onto the data bus after receiving an $\overline{\text{IORQ}}$ during an interrupt acknowledge sequence if it is the highest priority device requesting an interrupt. (This register is readable only during interrupt acknowledge cycles.) (Read/Write) (8 bits)

BLOCK LENGTH/REGISTER: Contains total block length of data to be searched and/or transferred. (Write Only) (16 bits)

BYTE COUNTER: Counts number of bytes transferred (or searched). On a Load or Continue the Byte Counter is reset to zero. Thereafter, each byte transfer operation increments it until it matches the contents of the Block Length Register, at which time End of Block is set in the status register and operation is suspended if programmed. Also is so programmed the DMA will generate an interrupt. (Read Only) (16 bits)

COMPARE REGISTER: Holds the byte for which a match is being sought in Search operations. (Write Only) (8 bits)

MASK REGISTER: Holds the 8 bit mask to determine which bits in the compare register are to be examined for a match. (Write Only) (8 bits)

STARTING ADDRESS REGISTERS/(PORT A AND PORT B): Holds the starting addresses (upper and lower 8 bits) for the two ports involved in Transfer operations. In Search Only operations, only one port address would have to be specified. Only memory starting addresses require both upper and lower 8-bits; I/O ports are generally addressed with only the lower 8-bits, and in this case the address contained in the register is a generally fixed address. (Write Only) (16 bits each)

ADDRESS COUNTERS (PORT A AND PORT B): These counters are loaded with the contents of the corresponding Starting Address Registers whenever Searches or Transfers are initiated with a Load or Continue. They are incremented, decremented or remain fixed, as programmed. (Read Only) (16 bits each)

PULSE CONTROL REGISTER: Holds program-supplied length (in bytes) of block after which the DMA will provide a signal pulse on the $\overline{\text{INT}}$ pin. (Since this occurs while both $\overline{\text{BUSRQ}}$ and $\overline{\text{BUSAK}}$ are active, the CPU will not interpret this as an interrupt request. Instead, the signal is used to communicate with a peripheral I/O device.) (Write Only) (16 bits each)

STATUS REGISTER: Match, End of Block, Ready Active, Interrupt Pending, and Write Address Valid bits indicate these functions when set. (Read Only) (8 bits)

MODES OF OPERATION

The DMA may be programmed for one of four modes of operation. (See Command Byte 2B).

BYTE AT A TIME: control is returned to the CPU after each one-byte cycle.

BURST: operation continues as long as the DMA's RDY input is active, indicating that the relevant port is ready. Control returns to the CPU when RDY is inactive or at end of block or a match if so programmed.

CONTINOUS: the entire Search and/or Transfer of a block of data is completed before control is returned to CPU.

TRANSPARENT: DMA operation occurs during a normal memory refresh times without visible loss of CPU time.

CLASSES OF OPERATION

The DMA has three classes of operation: Transfer only, Search Only and a combined Search-Transfer. (See Command Byte 1A.)

During a Transfer, data is read from one port and written to the other port, byte by byte. (The DMA's two ports are termed Port A and Port B.) The ports may be programmed to be either system main memory or peripheral I/O devices. Thus, a block of data might be written from one area in main memory to another; or from a peripheral to main memory.

During a Search, data is read only, and compared byte by byte against two DMA-internal registers, one of which contains a match byte and the other an optional mask byte which allows only certain bits to be compared. If any byte of searched data matches, a DMA-internal status bit is set; if programmed to do so, the DMA will then suspend operation and/or generate an interrupt.

CLASSES OF OPERATION (CONT'D)

The third class of operation is a combined Search-Transfer. In such an operation a block of data is transferred as described above until a match is found; then, as in a Search Only operation, the transfer may be suspended and/or an interrupt generated.

ADDRESSING

The DMA's addressing of ports is either fixed or sequential, incrementing or decrementing from a starting address. The length of the operation (number of bytes) is specified by the programmed contents of a block length register. The DMA can address block lengths of up to 64K bytes. During a transfer two separate port addresses are generated, one during the Read cycle and one during the Write cycle.

OPERATING SEQUENCE

Once the DMA has been programmed it may be "Enabled" (command byte 2D). In the enabled condition when Ready goes active the DMA will request the bus by bringing $\overline{\text{BUSRQ}}$ low. The CPU will acknowledge this with a $\overline{\text{BUS ACK}}$ which will normally be attached to $\overline{\text{BAI}}$. When the DMA receives $\overline{\text{BAI}}$ it will start its programmed operation releasing $\overline{\text{BUSRQ}}$ to a "high" state when it is through.

Z80-DMA PIN DESCRIPTION

A_0–A_{15} System Address Bus. All sixteen of these pins are used by the DMA to address system main memory or an I/O port (output)

D_0–D_7 System Data Bus. Commands from the CPU, DMA status and data from memory or peripherals are transferred on these tri-state pins (input/output)

+5V Power

GND Ground

Φ System clock (input)

$\overline{\text{M1}}$ Machine cycle One signal from CPU (input)

$\overline{\text{IORQ}}$ Input/Output Request to and from the System Bus (input/output)

$\overline{\text{MREQ}}$ Memory REQuest to the System Bus (input/output)

$\overline{\text{RD}}$ Read to and from the System Bus (input/output)

$\overline{\text{WR}}$ Write to and from the System Bus (input/output)

$\overline{\text{CE}}/\overline{\text{WAIT}}$ Chip Enable; may also be programmed to be $\overline{\text{WAIT}}$ during time when $\overline{\text{BAI}}$ is low (input)

$\overline{\text{BUSRQ}}$ BUS Request. Requests control of the CPU Address Bus, Data Bus and Status/Control Bus (input/output)

$\overline{\text{BAI}}$ Bus Acknowledge In. Signals that the system buses have been released for DMA control (input)

$\overline{\text{BAO}}$ Bus Acknowledge Out. $\overline{\text{BAI}}$ and $\overline{\text{BAO}}$ form a daisy-chain connection for system-wide priority bus control (output)

$\overline{\text{INT}}$ INTerrupt request (output)

IEI Interrupt Enable In (input)

IEO Interrupt Enable Out. IEI and IEO form a daisy-chain connection for system-wide priority interrupt control (output)

RDY ReaDY is monitored by the DMA to determine when a peripheral device associated with a DMA port is ready for a read or write operation (input, programmable as active high or low)

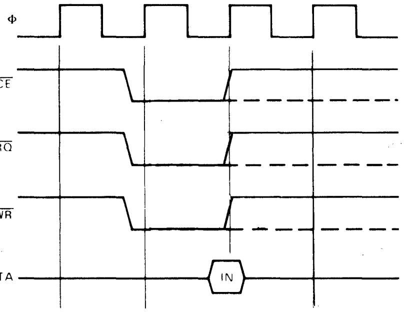
DMA TIMING WAVEFORMS

DMA COMMAND WRITE CYCLE

Illustrated here is the timing associated with a command byte or control byte being written to the DMA which is to be loaded into internal registers. Z80 Output instructions satisfy this timing.

COMMAND WRITE CYCLE

Figure 2

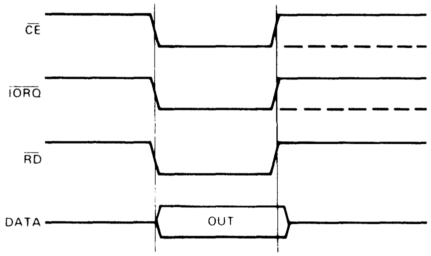


DMA REGISTER READ CYCLE

This timing is used when a read operation is performed on the DMA to access the contents of the Status Register, Address Counter or other readable registers. Z80 Input instructions satisfy this timing.

DMA TIMING WAVEFORMS (CONT'D)

REGISTER READ CYCLE
Figure 3

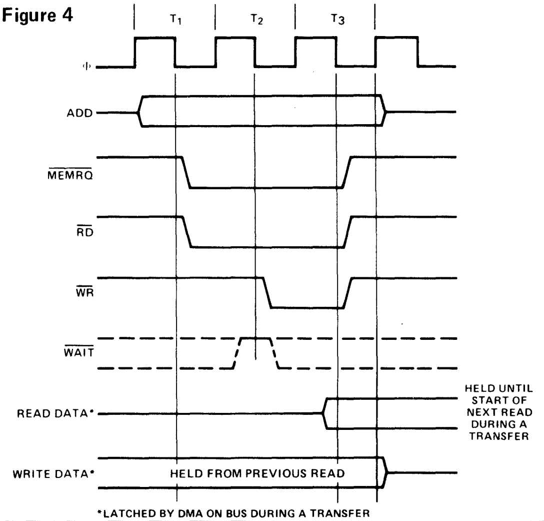


STD MEMORY TIMING

This timing is exactly the same as used by the Z80-CPU to access system main memory, either in a Read or Write operation. The DMA will default to this timing after a power-on reset, or when a Reset or Reset Timing command is written to it; and unless otherwise programmed, will used this timing during all Transfer or Search operations involving system main memory. During the memory Read portion of a transfer cycle, data is latched in the DMA on the negative edge of Φ during T_3 and held into the following Write cycle. During the memory Write portion of a transfer cycle, data is held form the previous Read cycle and released at the end of the present cycle.

NOTE: The DMA is normally programmed for a 3 T-cycle duration in memory transactions. But $\overline{\text{WAIT}}$ is sampled during the negative transistion of T_2, and if it is low, T_2 will be extended another T-cycle, after which $\overline{\text{WAIT}}$ will again be sampled. The duration of a memory transaction cycle may thus be indefinitely extended.

STD MEMORY TIMING
Figure 4



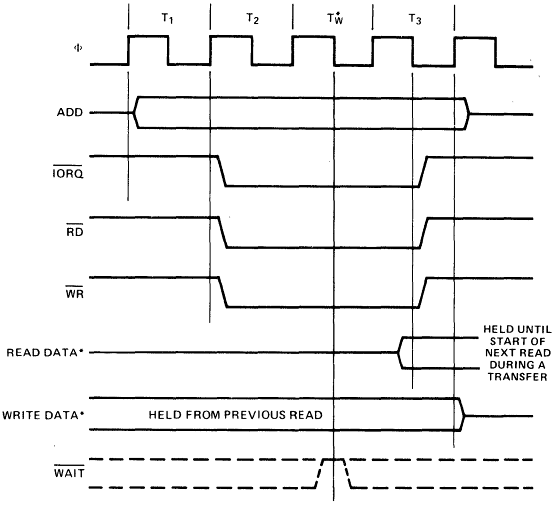
*LATCHED BY DMA ON BUS DURING A TRANSFER

STD PERIPHERAL TIMING

This timing is identical to the Z80-CPU's Read/Write timing to I/O peripheral devices. The DMA will default to this timing after a power-on reset, or when a Reset or Reset Timing command is written to it; and unless otherwise programmed, will use this timing during all Transfer or Search operations involving I/O peripherals. During the I/O Read of a transfer cycle, data is latched on the negative edge of Φ during T_3 and is then held into the Write cycle. During an I/O Write, data is held from the previous Read cycle until the end of the Write cycle.

NOTE: If $\overline{\text{WAIT}}$ is low during the negative transition of T_W*, then T_W* will be extended another T-cycle and $\overline{\text{WAIT}}$ will again be sampled. The duration of a peripheral transaction cycle may thus be indefinitely extended.

STD PERIPHERAL TIMING
Figure 5



*LATCHED BY DMA ON BUS DURING A TRANSFER

VARIABLE CYCLE

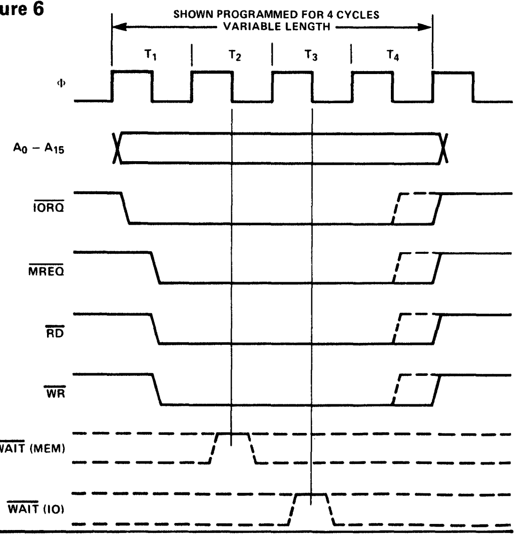
The Variable feature of the DMA allows the user to program the DMA's memory or peripheral transaction timing to values different than given above in the standard default diagrams. This permits the designer to tailor his timing to the particular requirements of his system components, and maximizes the data transfer rate while eliminating external signal conditioning logic. Cycle length can be one to four T-cycles (more if $\overline{\text{WAIT}}$ is used). Signal timing can be varied as shown. During a transfer, data will be latched by the DMA on the clock edge causing the rising edge of $\overline{\text{RD}}$ and will be held on the data lines until the end of the following Write cycle.

(See Timing Control Byte, page 9.)

DMA TIMING WAVEFORMS (CONT'D)

VARIABLE CYCLE

Figure 6



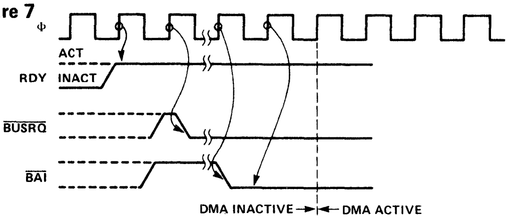
Z80 Family

DMA BUS REQUEST AND ACCEPTANCE FOR BYTE-AT-A-TIME, BURST, AND CONTINUOUS MODE

Ready is sampled on every rising edge of Φ. When it is found to be active, the following rising edge of Φ generates $\overline{\text{BUSRQ}}$. After receiving $\overline{\text{BUSRQ}}$ the CPU will grant a $\overline{\text{BUSAK}}$ which will be connected to $\overline{\text{BAI}}$ either directly or through the Bus Acknowledge Daisy Chain. When a low is detected on $\overline{\text{BAI}}$ (sampled on every rising edge of Φ), the next rising edge of Φ will start an active DMA cycle. See Figure 7.

BUS REQUEST AND ACCEPTANCE FOR BYTE-AT-A-TIME, BURST, AND CONTINOUS MODE

Figure 7

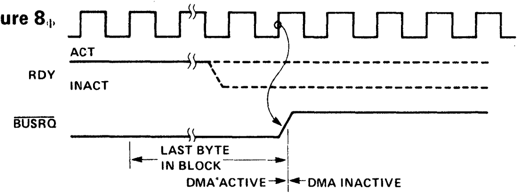


DMA BUS RELEASE AT END OF BLOCK FOR BURST OR CONTINUOUS MODE

Timing for End of Block and DMA not programmed for Auto-restart. See Figure 8.

BUS RELEASE AT END OF BLOCK

Figure 8

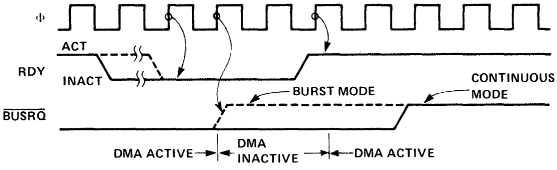


DMA BUS RELEASE WITH 'READY' FOR BURST AND CONTINUOUS MODE

The DMA will relinquish the bus after RDY has gone inactive (Burst mode) or after an End of Block or a Match is found (Continuous mode). With RDY inactive, the DMA in Continuous mode is inactive but maintains control of the bus ($\overline{\text{BUSRQ}}$ low) until the cycle is resumed when RDY goes active. See Figure 9.

BUS RELEASE WITH 'READY' FOR BURST AND CONTINOUS MODE

Figure 9

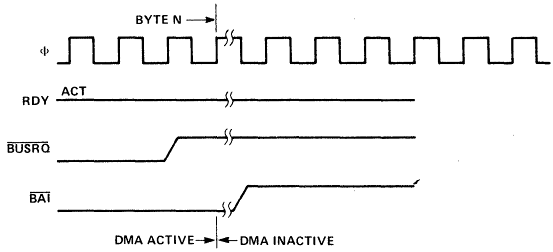


DMA BUS RELEASE FOR BYTE-AT-A-TIME MODE

In the Byte mode the DMA will release $\overline{\text{BUSRQ}}$ on the rising edge of Φ prior to the end of each Read cycle in Search Only or each Write cycle in a Transfer, regardless of the state of RDY. The next bus request will come after both $\overline{\text{BUSRQ}}$ and $\overline{\text{BAI}}$ have returned high. See Figure 10.

BUS RELEASE FOR BYTE-AT-A-TIME MODE

Figure 10

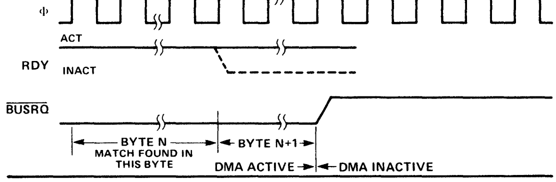


DMA BUS RELEASE WITH MATCH FOR BURST OR CONTINUOUS MODES

When a Match is found and the DMA is programmed to stop on Compare, the DMA performs an operation on the next byte and then releases bus. See Figure 11.

BUS RELEASE WITH MATCH FOR BURST OR CONTINOUS MODES

Figure 11



READING FROM THE DMA INTERNAL REGISTERS

Seven registers are available in the DMA for reading. They are: 8 bits of the status register, the upper and lower 8 bits of the block length register, and two port address registers.

These are available to be read sequentially: status, BLK Lower, BLK Upper, Port A Address lower, Port A Address Upper, Port B Address lower, Port B Address upper. An internal pointer points to each register in turn as each READ is accomplished. If a register is not to be read, it may be excluded by programming a 0 in the Read Byte. The internal pointer will skip any register not programmed with a 1 in the Read Byte. After a Reset or a Load, Reset RD must be given to set the internal pointer pointing to the first register programmed to be read by the Read Byte. After RD Status, the pointer will be pointing to the status register regardless of the mask and the next read will be from the status register. The following read will be from the register pointed to before RD Status.

PROGRAMMING THE DMA

Previous sections of this specification have indicated the various functions and modes of the DMA. The diagrams and charts below will show how the DMA is programmed to select among these functions and modes and to adapt itself to the requirements of the user system. More detailed programming information is available in the Z80-DMA Technical Manual.

The Z80-DMA chip may be in an "enable"state, in which it can gain control of the system buses and direct the transfer of data between its ports, or in a "disable" state, when it cannot gain control of the bus. Program commands can be written to it in either state, but writing a command to it automatically puts it in the disable state, which is maintained until an enable command is issued to the DMA. The CPU must program it in advance of any data search or transfer by addressing it as an I/O port and sending it a sequence of 8 bit command bytes via the system data bus using Output instructions. When the DMA is powered up or reset by any means, the DMA will automatically be placed into a disable state, in which it can initiate neither bus requests nor data transfers nor interrupts.

The command bytes contain information to be loaded into the DMA's control and other registers and/or information to alter the state of the chip, such as an Enable Interrupt command. The command structure is designed so that certain bits in some commands can be set to alert the DMA to expect the next byte written to it to be for a particular internal register.

The following diagrams and charts give the function of each bit in the six different command bytes. Two of these are defined as being from Group 1, and are termed command bytes 1A and 1B. These Group 1 commands contain the most basic DMA set-up information. The other four are categorized as Group 2, and are termed commands 2A, 2B, 2C, and 2D. Group 2 words specify more detailed set-up information.

COMMAND BYTE 1A

Figure 12

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	BLOCK LENGTH (UPPER) FOLLOWS	BLOCK LENGTH (LOWER) FOLLOWS	PORT A STARTING ADDRESS (UPPER) FOLLOWS	PORT A STARTING ADDRESS (LOWER) FOLLOWS	SOURCE PORT	CLASS CONTROL C_1	CLASS CONTROL C_0

Specifies Group 1

Byte 1A
cannot be 00

C_1	C_0	Function
0	0	Not allowed. (Command Byte 1B)
0	1	Transfer Only.
1	0	Search Only.
1	1	Search and Transfer.

$D_2 = 1$ Port A is read from, Port B is written to (unless the Search Only Mode has been selected, in which case Port B is never addressed).

$D_2 = 0$ Port B is read from, Port A is written to (unless the Search Only Mode has been selected, in which case Port A is never addressed).

COMMAND BYTE 1B

Figure 13

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	TIMING BYTE FOLLOWS	ADDRESS FIXED	ADDRESS INCREMENTS/ DECREMENTS	I/O OR MEMORY	PORT A OR B	0	0

Specifies Group 1

Specifies Byte 1B

$D_4 = 1$ Address for this port increments after each byte.

$D_4 = 0$ Address for this port decrements after each byte.

$D_3 = 1$ This port addresses an I/O peripheral.

$D_3 = 0$ This port addresses main memory.

$D_2 = 1$ This word programs Port A.

$D_2 = 0$ This word programs Port B.

COMMAND BYTE 2A

Figure 14

D7	D6	D5	D4	D3	D2	D1	D0
1	ENABLE CHIP	ENABLE INTERRUPT	MATCH BYTE FOLLOWS	MASK BYTE FOLLOWS	STOP ON COMPARE	0	0

Specifies Group 2 Specifies Byte 2A

COMMAND BYTE 2B

Figure 15

D7	D6	D5	D4	D3	D2	D1	D0
1	MODE M_1	MODE M_0	INTERRUPT CONTROL BYTE FOLLOWS	PORT B UPPER ADDRESS FOLLOWS	PORT B LOWER ADDRESS FOLLOWS	0	1

Specifies Group 2 Specifies Byte 2B

M1	M0	Mode
0	0	Byte
0	1	Continuous
1	0	Burst
1	1	Transparent

COMMAND BYTE 2C

Figure 16

D7	D6	D5	D4	D3	D2	D1	D0
1	NOT USED	AUTOMATIC RESTART	$\overline{\text{WAIT}}$ MULTIPLEXED	READY HIGH/LOW	NOT USED	1	0

Specifies Group 2 Specifies Byte 2C

$D_5 = 1$	Automatically repeats entire operation when end of block is reached.
$D_5 = 0$	No affect.
$D_4 = 1$	$\overline{\text{CE}}$ and $\overline{\text{WAIT}}$ multiplexed on same pin.
$D_4 = 0$	$\overline{\text{CE}}$ only.
$D_3 = 1$	Ready active high.
$D_3 = 0$	Ready active low.

COMMAND BYTE 2D

Figure 17

D7	D6	D5	D4	D3	D2	D1	D0
1	f_4	f_3	f_2	f_1	f_0	1	1

Specifies Group 2 Specifies Byte 2D

COMMAND BYTE 2D (CONT'D)

Hex	f_4	f_3	f_2	f_1	f_0	
C3	1	0	0	0	0	Reset
C7	1	0	0	0	1	Reset Port A Timing
CB	1	0	0	1	0	Reset Port B Timing
CF	1	0	0	1	1	Load
D3	1	0	1	0	0	Continue
AB	0	1	0	1	0	Enable Int
AF	0	1	0	1	1	Disable Int
A3	0	1	0	0	0	Reset Int
87	0	0	0	0	1	Enable DMA
83	0	0	0	0	0	Disable DMA
BB	0	1	1	1	0	Read Byte Follows
A7	0	1	0	0	1	Reset RD
BF	0	1	1	1	1	RD Status
B3	0	1	1	0	0	Force Ready
B7	0	1	1	0	1	Enable After RETI
8B	0	0	0	1	0	Reset Status

COMMAND BYTE 2D SUMMARY

Reset:	Resets all interrupt circuitry, disables interrupts and bus req. logic.
Reset Timing A or B:	Resets timing for Port A or B to standard Z80-CPU timing.
Load:	Zeros Byte Counter and loads Starting Address for both Ports.
Continue:	Resets byte counter only. Addresses continue from present location.
Enable Interrupt:	Permits interrupt to occur.
Disable Interrupt:	Inhibits interrupt from occurring.
Reset Interrupt:	Resets and disables all interrupt circuits (similar to RETI).
Enable DMA, Disable DMA:	Overall enable or disable for all operations except interrupts; does not reset any functions.
Read Byte Follows:	Next write to DMA will contain a mask to program which readable registers are to be read.
Reset RD:	Next read will be from status register.

COMMAND BYTE 2D SUMMARY (CONT'D)

RD Status:	Next read will be from status register.
Force Ready:	Ready will be considered active regardless of the state of external RDY pin. Used for Mem-Mem operations where no RDY signal is needed.
Enable after RETI:	DMA will not request bus until after it has received a RETI.
RST Status:	Resets Match and End of Block status bits.

READ BYTE

Figure 18

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
NOT USED	PORT B UPPER ADDR	PORT B LOWER ADDR	PORT A UPPER ADDR	PORT A LOWER ADDR	BYTE UPPER COUNT	BYTE LOWER COUNT	STATUS

A "1" in any bit position enables that register to be read.

INTERRUPT CONTROL BYTE

Figure 19

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
NO EFFECT	INTERRUPT BEFORE REQUESTING BUS	STATUS AFFECTS INTERRUPT VECTOR	INTERRUPT VECTOR FOLLOWS	PULSE COUNT FOLLOWS	PULSE GENERATED	INTERRUPT ON MATCH FOUND	INTERRUPT AT END OF BLOCK

A "1" in a bit position selects the option.

TIMING CONTROL BYTE

Figure 20

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{WR}$ END	$\overline{RD}$ END	NOT USED	NOT USED	$\overline{MREQ}$ END	$\overline{IORQ}$ END	T_1	T_0

T_1	T_0	Cycle Length
0	0	4
0	1	3
1	0	2
1	1	1

A "0" in D_2, D_3, D_6, or D_7 will cause the corresponding control signal to end ½ clock time before the end of the cycle. Note: the total operation (Read and Write in Transfer or Read in Search) must be at least 2 cycles long.

MASK BYTE

A zero in a given bit position will cause a compare to be performed between that bit position in the compare word register and the same bit position in the data being read.

MATCH BYTE

Up to an 8-bit word to be compared to D0 – D7 during a read. See MASK BYTE.

STATUS BYTE

Figure 21

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
NOT USED	NOT USED	END OF BLK	MATCH	INT. PENDING	NOT USED	READY ACTIVE	WRITE ADDRESS VALID

PULSE COUNT

This 8-bit word is loaded into a register. At the completion of each operation, the register is compared with the lower 8-bits of the byte counter. When it compares, the $\overline{INT}$ line is pulsed (but no interrupt is generated).

INTERRUPT VECTOR

This 8-bit byte is supplied to the CPU during Interrupt acknowledge if the DMA is the highest priority interrupting device.

If bit 5 of the Interrupt Control Byte (Fig. 19) has been set and the DMA has been programmed to interrupt on a given status condition then D_1 and D_2 of the vector will be modified as follows:

Vector Bits	D_2	D_1	
	0	0	INT on RDY
	0	1	Match
	1	0	End of Blk
	1	1	Match, End o Blk

DMA PROGRAMMING EXAMPLE

The following example will show how the DMA may be programmed to transfer data from a peripheral (Port A) to memory (Port B). The table of bytes may be stored in memory and transferred to the DMA with an output instruction such as an OTIR.

Port A	Data Flow	Port B
	⟶	Memory
Peripheral		Starting Address
Address 05H	Block Length	1050H
	1000 H Bytes	

READY from the peripheral is active high
Memory address increments on each write

DMA PROGRAMMING EXAMPLE (CONT'D)

	D7	D6	D5	D4	D3	D2	D1	D0	HEX
1 Command Byte 1A Sets the DMA to receive Block length and Port A address and sets direction of transfer	0 Group 1	1 Blk Length Upper Follows	1 Blk Length Lower Follows	0 No Port A Upper Addr Follows	1 Port A Lower Addr Follows	1 A→B	0 1A Transfer No Search	1	6D
2 Port A Address Lower 8-bits	0	0	0	0	0	1	0	1	01
3 Block Length Lower 8-bits	0	0	0	0	0	0	0	0	00
4 Block Length Upper 8-bits	0	0	0	1	0	0	0	0	10
5 Command Byte 1B Defines Port A as peripheral with fixed addresses	0 Group 1	0 No Timing Follows	1 Fixed Addresses	X	1 Port is IO	0 This is Port "A"	0 1B	0	
6 Command Byte 1B Defines Port B as a memory with incrementing addresses	0 Group 1	0 No Timing Follows	0 Address Changes	1 Address Increments	0 Port is Memory	1 This is Port "B"	0 1B	0	14
7 Command Byte 2B Sets mode to burst, sets DMA to expect Port B starting address	1 Group 2	1 Burst Mode	0	0 No Int Cont Byte Follows	1 Port B Upper Addr Follows	1 Port B Lower Addr Follows	0 2B	1	CD
8 Port B Address Lower 8-bits	0	1	0	1	0	0	0	0	50
9 Port B Address Upper 8-bits	0	0	0	1	0	0	0	0	10
10Command Byte 2C Sets Ready Active High	1 Group 2	X	0 No Auto Restart	0 No wait States	1 Rdy Active High	X	1 2C	0	
11Command Byte 2D loads starting addresses and resets block counter	1 Group 2	1 Load	0	0	1	1	1 2A	1	CF
12Command Byte 2D Enables DMA to start operation	1 Group 2	0 ENABLE DMA	0	0	0	1	1 2D	1	87

To reload the same addresses and block length for a subsequent operation, only two bytes are needed.

1. Command byte 2D 11001111
   Reloads port addresses Load
   and block length

2. Command byte 2D 10001011
   Enables DMA Enable DMA

ABSOLUTE MAXIMUM RATINGS

PRELIMINARY

Temperature Under Bias . . . . Specified operating range
Storage Temperature . . . . -65°C to +150°C
Voltage On Any Pin with Respect to Ground . . . . -0.3V to +7V
Power Dissipation 1.5W

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Z80-DMA D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6		V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	$I_{\emptyset L}$ = 2mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250 μA
V_{CC}	Power Supply Current			150	mA	t_C = 400 nsec
I_{LI}	Input Leakage Current			10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leqslant V_{IN} \leqslant V_{CC}$

Z80A-DMA D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ± 5% unless otherwise specified

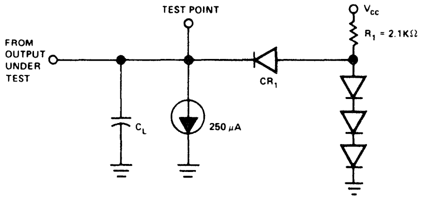
Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6		V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 2mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250μA
V_{CC}	Power Supply Current		90	200	mA	t_C = 250nsec
I_{LI}	Input Leakage Current			10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leqslant V_{IN} \leqslant V_{CC}$

CAPACITANCE

T_A = 25°C, f = 1 MHz

Symbol	Parameter	Max.	Unit	Test Condition
C_Φ	Clock Capacitance	35	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

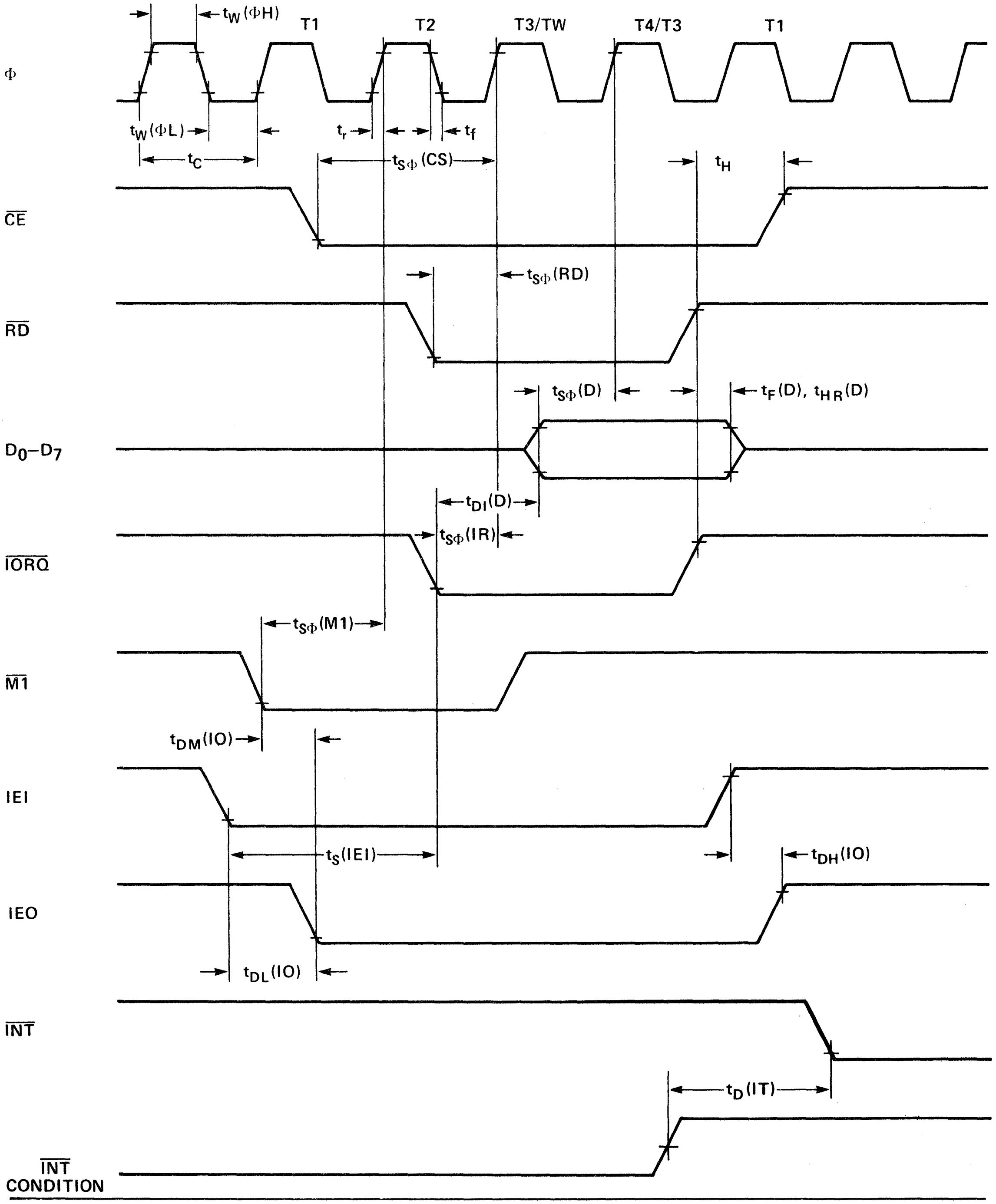
LOAD CIRCUIT FOR OUTPUT



Z80 and Z80A as a Peripheral Device (Inactive State)

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV	= +0.5V



Z80-DMA as a Peripheral Device (Inactive State).
T_A =0° C to 70° C, Vcc = +5V±5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_C	Clock Period	400	(1)	nsec	
	$t_{W(\Phi H)}$	Clock Pulse Width, Clock High	170	2000	nsec	
	$t_{W(\Phi L)}$	Clock Pulse Width, Clock Low	170	2000	nsec	
	$t_{r,\ f}$	Clock Rise and Fall Times		30	nsec	
	t_H	Any Hold Time for Specified Setup Time	0		nsec	
$\overline{CE}$	$t_{S\Phi(CS)}$	Control Signal Setup Time to Rising Edge of Φ During Write Cycle	280		nsec	
$D_{0\text{-}7}$	$t_{DR(D)}$	Data Output Delay from Falling Edge of $\overline{RD}$		430	nsec	(2)
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Φ During Write or $\overline{M1}$ Cycle	50		nsec	C_L = 50pF
	$t_{DI(D)}$	Data Output Delay from Falling Edge of $\overline{IORQ}$ During INTA Cycle		340	nsec	(3)
	$t_{F(D)}$	Delay to Floating Bus (Output Buffer Disable Time)		160	nsec	
IEI	$t_{S(IEI)}$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		nsec	
IEO	$t_{DH(IO)}$	IEO Delay Time from Rising Edge of IEI		210	nsec	
	$t_{DL(IO)}$	IEO Delay Time from Falling Edge of IEI		190	nsec	
	$t_{DM(IO)}$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$) See Note A.		300	nsec	C_L = 50pF
$\overline{IORQ}$	$t_{S\Phi(IR)}$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Write Cycle	250		nsec	
$\overline{M1}$	$t_{S\Phi(M1)}$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or $\overline{M1}$ Cycle. See Note B.	210		nsec	
$\overline{RD}$	$t_{S\Phi(RD)}$	$\overline{RD}$ Setup Time to Rising Edge of Φ During $\overline{M1}$ Cycle	240		nsec	
$\overline{INT}$	$t_{D(IT)}$	$\overline{INT}$ Delay Time from Condition Causing $\overline{INT}$. $\overline{INT}$ generated only when DMA is inactive.		500	nsec	
BAO	$t_{DH(BO)}$	BAO Delay from Rising Edge of BAI	150	200	nsec	
	$t_{DL(BO)}$	BAO Delay from Falling Edge of BAI	150	200	nsec	

NOTES:

A. 2.5 $t_c > (N\text{-}2)\ t_{DL(IO)} + t_{DM(IO)} + t_{S(IEI)}$ + TTL Buffer Delay, if any

(1) $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

(2) Increase $t_{DR(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

(3) Increase $t_{D(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

Z80A DMA as a Peripheral Device (Inactive State)
T_A = 0°C to 70°C, Vcc = +5V±5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_C	Clock Period	250	(1)	nsec	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	105	2000	nsec	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	105	2000	nsec	
	$t_{r,\ f}$	Clock Rise and Fall Times		30	nsec	
	t_H	Any Hold Time for Specified Setup Time	0		nsec	
$\overline{CE}$	$t_{S\Phi}(CS)$	Control Signal Setup Time to Rising Edge of Φ During Write Cycle	145		nsec	
D_{0-7}	$t_{DR}(D)$	Data Output Delay from Falling Edge of $\overline{RD}$		380	nsec	(2)
	$t_{S\Phi}(D)$	Data Setup Time to Rising Edge of Φ During Write or $\overline{M1}$ Cycle	50		nsec	C_L = 50pF
	$t_{DI}(D)$	Data Output Delay from Falling Edge of $\overline{IORQ}$ During INTA Cycle		250	nsec	(3)
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		110	nsec	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		nsec	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		160	nsec	
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		130	nsec	
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$) See Note A.		190	nsec	C_L = 50pF
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Write Cycle	115		nsec	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or $\overline{M1}$ Cycle. See Note B.	90		nsec	
$\overline{RD}$	$t_{S\Phi}(RD)$	$\overline{RD}$ Setup Time to Rising Edge of Φ During $\overline{M1}$ Cycle	115		nsec	
$\overline{INT}$	$t_D(IT)$	$\overline{INT}$ Delay Time from Condition Causing $\overline{INT}$. $\overline{INT}$ generated only when DMA is inactive.		500	nsec	
BAO	$t_{DH}(BO)$	BAO Delay from Rising Edge of BAI	150	200	nsec	
	$t_{DL}(BO)$	BAO Delay from Falling Edge of BAI	150	200	nsec	

NOTES:

A. $2.5t_C > (N\text{-}2)\ t_{DL(IO)} + t_{S(IEI)}$ + TTL Buffer Delay, if any

(1) $t_C = t_{W(\Phi H)} + t_{W(\Phi L)} + t_r + t_f$
(2) Increase $t_{DR(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.
(3) Increase $t_{D(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

A.C. CHARACTERISTICS MK3883 Z80-DMA

PRELIMINARY

Z80-DMA as a Bus Controller (Active State)
T_A =0°C to 70°C, Vcc = +5V±5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c	Clock Period	.4	(12)	μsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	180	2000	nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,\ f}$	Clock Rise and Fall Time		30	nsec	
$A_{0\text{-}15}$	$t_D(AD)$	Address Output Delay		145	nsec	
	$t_F(AD)$	Delay to Float		110	nsec	C_L =50pF
	t_{acm}	Address Stable Prior to $\overline{\text{MREQ}}$ (Memory Cycle)	(1)		nsec	D
	t_{aci}	Address Stable Prior to $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ or $\overline{\text{WR}}$ (I/O Cycle)	(2)		nsec	D
	t_{ca}	Address Stable from $\overline{\text{RD}}$ or $\overline{\text{WR}}$	(3)		nsec	D
	t_{caf}	Address Stable From $\overline{\text{RD}}$ or $\overline{\text{WR}}$ During Float	(4)		nsec	D
$D_{0\text{-}7}$	$t_D(D)$	Data Output Delay		260	nsec	
	$t_F(D)$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi}(D)$	Data Setup Time to Rising Edge of Clock	50		nsec	C_L =200pF
	$t_{S\overline{\Phi}}(D)$	Data Setup Time to Falling Edge of Clock During Read When Falling Edge Ends $\overline{\text{RD}}$	60		nsec	
	t_{dcm}	Data Stable Prior to $\overline{\text{WR}}$ (Memory Cycle)	(5)		nsec	D
	t_{dci}	Data Stable Prior to $\overline{\text{WR}}$ (I/O Cycle)	(6)		nsec	D
	t_{cdf}	Data Stable From $\overline{\text{WR}}$	(7)		nsec	D
	t_H	Any Hold Time for Setup Time	0		nsec	
$\overline{\text{MREQ}}$	$t_{DL\overline{\Phi}}(MR)$	$\overline{\text{MREQ}}$ Delay from Falling Edge of Clock, $\overline{\text{MREQ}}$ Low		100	nsec	
	$t_{DH\Phi}(MR)$	$\overline{\text{MREQ}}$ Delay from Rising Edge of Clock, $\overline{\text{MREQ}}$ High		100	nsec	
	$t_{DH\overline{\Phi}}(MR)$	$\overline{\text{MREQ}}$ Delay from Falling Edge of Clock, $\overline{\text{MREQ}}$ High		100	nsec	
	$t_{DL\Phi}(MR)$	$\overline{\text{MREQ}}$ Delay from Falling Edge of Clock, $\overline{\text{MREQ}}$ Low		100	nsec	C_L =50pF
	$t_w(\overline{MRL})$	Pulse Width, $\overline{\text{MREQ}}$ Low	(8)		nsec	D
	$t_w(MRH)$	Pulse Width, $\overline{\text{MREQ}}$ High	(9)		nsec	D
$\overline{\text{IORQ}}$	$t_{DL\Phi}(IR)$	$\overline{\text{IORQ}}$ Delay from Rising Edge of Clock, $\overline{\text{IORQ}}$ Low		90	nsec	
	$t_{DL\overline{\Phi}}(IR)$	$\overline{\text{IORQ}}$ Delay from Falling Edge of Clock, $\overline{\text{IORQ}}$ Low		110	nsec	
	$t_{DH\Phi}(IR)$	$\overline{\text{IORQ}}$ Delay from Rising Edge of Clock, $\overline{\text{IORQ}}$ High		100	nsec	C_L =50pF
	$t_{DH\overline{\Phi}}(IR)$	$\overline{\text{IORQ}}$ Delay from Falling Edge of Clock, $\overline{\text{IORQ}}$ High		110	nsec	
$\overline{\text{RD}}$	$t_{DL\Phi}(RD)$	$\overline{\text{RD}}$ Delay from Rising Edge of Clock, $\overline{\text{RD}}$ Low		100	nsec	
	$t_{DL\overline{\Phi}}(RD)$	$\overline{\text{RD}}$ Delay from Falling Edge of Clock, $\overline{\text{RD}}$ Low		130	nsec	
	$t_{DH\Phi}(RD)$	$\overline{\text{RD}}$ Delay from Rising Edge of Clock, $\overline{\text{RD}}$ High		100	nsec	C_L =50pF
	$t_{DH\Phi}(\overline{RD})$	$\overline{\text{RD}}$ Delay from Falling Edge of Clock, $\overline{\text{RD}}$ High		110	nsec	

A.C. CHARACTERISTICS MK3883 Z80-DMA (CONT'D) PRELIMINARY

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
$\overline{WR}$	$t_{DL\Phi}(WR)$	$\overline{WR}$ Delay from Rising Edge of Clock, $\overline{WR}$ Low		80	nsec	C_L =50pF
	$t_{DL\overline{\Phi}}(WR)$	$\overline{WR}$ Delay from Falling Edge of Clock, $\overline{WR}$ Low		90	nsec	
	$t_{DH\overline{\Phi}}(WR)$	$\overline{WR}$ Delay from Falling Edge of Clock, $\overline{WR}$ High		100	nsec	
	$t_{DH\Phi}(WR)$	$\overline{WR}$ Delay from Rising Edge of Clock, $\overline{WR}$ High		100	nsec	
	$t_w(\overline{WR}L)$	Pulse Width, $\overline{WR}$ Low	(10)		nsec	
$\overline{WAIT}$	$t_s(WT)$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{BUSRQ}$	$t_D(BQ)$	$\overline{BUSRQ}$ Delay Time from Rising Edge of Clock	100		nsec	
	$t_F(C)$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		100	nsec	

NOTES:

A. Data should be enable onto the DMA data bus when $\overline{RD}$ is active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C. Output Delay vs. Loaded Capacitance
$T_A = 70^\circ C$ $V_{cc} = +5V \pm 5\%$
(1) $\Delta C_L = +100pF$ ($A\phi$–A_{15} and Control Signals), add 30 nsec to timing shown.

D. During Standard CPU Timing.

1. $t_{acm} = t_w(\Phi_H) + t_f - 75$
2. $t_{aci} = t_c - 80$
3. $t_{ca} = t_w(\Phi_L) + t_r - 40$
4. $t_{caf} = t_w(\Phi_L) + t_r - 60$
5. $t_{dcm} = t_c - 180$
6. $t_{dci} = t_w(\Phi_L) + t_r - 180$
7. $t_{cdf} = t_w(\Phi_L) + t_r - 50$
8. $t_w\ (MRL) = t_c - 40$
9. $t_w\ (MRH) = t_c - 40$ Std. CPU Timing
$t_w\ (MRH) = t_w(\Phi_H) + t_f - 30$ Variable 1 Cycle.
10. $t_w(WR) = t_c - 40$Std. CPU Timing
$t_w(WR) = t_w(\Phi_H) + t_f - 30$ Variable 1 Cycle.
12. $t_c = t_w(\Phi_H) + t_w(\Phi_L) + t_r + t_f$

A.C. CHARACTERISTICS MK3883-4 Z80A-DMA

PRELIMINARY

Z80A-DMA as a Bus Controller (Active State)

T_A =0°C to 70°C, Vcc = +5V±5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c	Clock Period	.25	(12)	μsec	
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	110	2000	nsec	
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	110	2000	nsec	
	$t_{r,\ f}$	Clock Rise and Fall Time		30	nsec	
$A_{0\text{-}15}$	$t_{D(AD)}$	Address Output Delay		110	nsec	
	$t_{F(AD)}$	Delay to Float		90	nsec	C_L =50pF
	t_{acm}	Address Stable Prior to $\overline{\text{MREQ}}$ (Memory Cycle)	(1)		nsec	D
	t_{aci}	Address Stable Prior to $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ or $\overline{\text{WR}}$ (I/O Cycle)	(2)		nsec	D
	t_{ca}	Address Stable from $\overline{\text{RD}}$ or $\overline{\text{WR}}$	(3)		nsec	D
	t_{caf}	Address Stable From $\overline{\text{RD}}$ or $\overline{\text{WR}}$ During Float	(4)		nsec	D
$D_{0\text{-}7}$	$t_{D(D)}$	Data Output Delay		180	nsec	
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During Read When Rising Edge Ends RD	35		nsec	C_L = 200pF
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During Read When Falling Edge Ends $\overline{\text{RD}}$	50		nsec	
	t_{dcm}	Data Stable Prior to $\overline{\text{WR}}$ (Memory Cycle)	(5)		nsec	D
	t_{dci}	Data Stable Prior to $\overline{\text{WR}}$ (I/O Cycle)	(6)		nsec	D
	t_{cdf}	Data Stable From $\overline{\text{WR}}$	(7)		nsec	D
	t_H	Any Hold Time for Setup Time		0	nsec	
$\overline{\text{MREQ}}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{\text{MREQ}}$ Delay from Falling Edge of Clock, $\overline{\text{MREQ}}$ Low		75	nsec	
	$t_{DH\Phi(MR)}$	$\overline{\text{MREQ}}$ Delay from Rising Edge of Clock, $\overline{\text{MREQ}}$ High		75	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{\text{MREQ}}$ Delay from Falling Edge of Clock, $\overline{\text{MREQ}}$ High		75	nsec	
	$t_{DL\Phi(MR)}$	$\overline{\text{MREQ}}$ Delay from Falling Edge of Clock, $\overline{\text{MREQ}}$ Low		80	nsec	C_L =50pF
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{\text{MREQ}}$ Low	(8)		nsec	D
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{\text{MREQ}}$ High	(9)		nsec	D
$\overline{\text{IORQ}}$	$t_{DL\Phi(IR)}$	$\overline{\text{IORQ}}$ Delay from Rising Edge of Clock, $\overline{\text{IORQ}}$ Low		75	nsec	
	$t_{DL\overline{\Phi}(IR)}$	$\overline{\text{IORQ}}$ Delay from Falling Edge of Clock, $\overline{\text{IORQ}}$ Low		80	nsec	
	$t_{DH\Phi(IR)}$	$\overline{\text{IORQ}}$ Delay from Rising Edge of Clock, $\overline{\text{IORQ}}$ High		80	nsec	C_L =50pF
	$t_{DH\overline{\Phi}(IR)}$	$\overline{\text{IORQ}}$ Delay from Falling Edge of Clock, $\overline{\text{IORQ}}$ High		80	nsec	
$\overline{\text{RD}}$	$t_{DL\Phi(RD)}$	$\overline{\text{RD}}$ Delay from Rising Edge of Clock, $\overline{\text{RD}}$ Low		75	nsec	
	$t_{DL\overline{\Phi}(RD)}$	$\overline{\text{RD}}$ Delay from Falling Edge of Clock, $\overline{\text{RD}}$ Low		95	nsec	
	$t_{DH\Phi(RD)}$	$\overline{\text{RD}}$ Delay from Rising Edge of Clock, $\overline{\text{RD}}$ High		75	nsec	C_L =50pF
	$t_{DH\overline{\Phi}(RD)}$	$\overline{\text{RD}}$ Delay from Falling Edge of Clock, $\overline{\text{RD}}$ High		80	nsec	

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
$\overline{\text{WR}}$	$t_{DL\Phi}(WR)$	$\overline{\text{WR}}$ Delay from Rising Edge of Clock, $\overline{\text{WR}}$ Low		60	nsec	C_L =50pF
	$t_{DL\Phi}(WR)$	$\overline{\text{WR}}$ Delay from Falling Edge of Clock, $\overline{\text{WR}}$ Low		80	nsec	
	$t_{DH\Phi}(WR)$	$\overline{\text{WR}}$ Delay from Falling Edge of Clock, $\overline{\text{WR}}$ High		80	nsec	
	$t_{DH\Phi}(WR)$	$\overline{\text{WR}}$ Delay from Rising Edge of Clock, $\overline{\text{WR}}$ High		80	nsec	
	$t_w(WRL)$	Pulse Width, $\overline{\text{WR}}$ Low	(10)		nsec	
$\overline{\text{WAIT}}$	$t_s(WT)$	$\overline{\text{WAIT}}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{\text{BUSRQ}}$	$t_D(BQ)$	$\overline{\text{BUSRQ}}$ Delay Time from Rising Edge of Clock	100		nsec	
	$t_F(C)$	Delay to Float ($\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$ and WR)		80	nsec	

NOTES:

A. Data should be enable onto the DMA data bus when $\overline{\text{RD}}$ is active.
B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.
C. Output Delay vs. Loaded Capacitance
TA = 70°C Vcc = +5V ± 5%
(1) $\triangle C_L$ = +100pF ($A\phi$–A_{15} and Control Signals), add 30 nsec to timing shown.
D. During Standard CPU Timing.

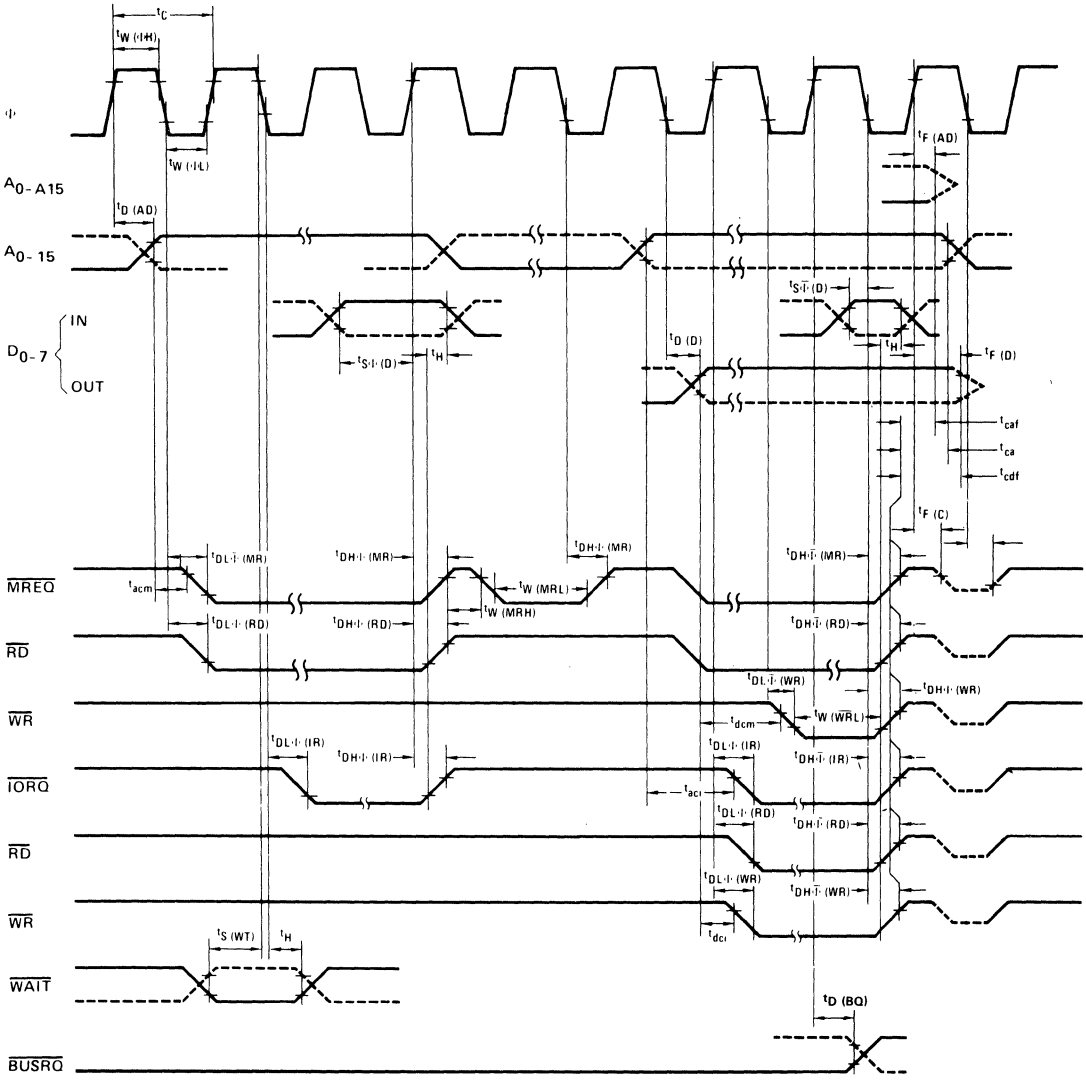
1. $t_{acm} = t_w(\Phi H) + t_f - 75$
2. $t_{aci} = t_c - 80$
3. $t_{ca} = t_w(\Phi L) + t_r - 40$
4. $t_{caf} = t_w(\Phi L) + t_r - 60$
5. $t_{dcm} = t_c - 180$
6. $t_{dci} = t_w(\Phi L) + t_r - 180$
7. $t_{cdf} = t_w(\Phi L) + t_r - 50$
8. $t_w\ (MRL) = t_c - 40$
9. $t_w\ (MRH) = t_c - 40$ Std. CPU Timing
$t_w\ (MRH) = t_w(\Phi H) + t_f - 30$ Variable 1 Cycle.
10. $t_w(WR) = t_c - 40$Std. CPU Timing
$t_w(WR) = t_w(\Phi H) + t_f - 30$ Variable 1 Cycle.
12. $t_c = t_w(\Phi H) + t_w(\Phi L) + t_r + t_f$

A.C. TIMING DIAGRAMS

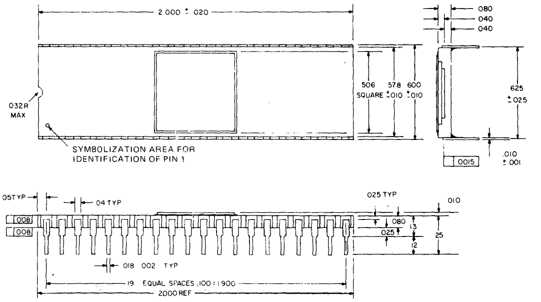
Z80 and Z80A as a Bus Controller (Active State)

Timing measurements are made at the following voltages, unless otherwise specified:

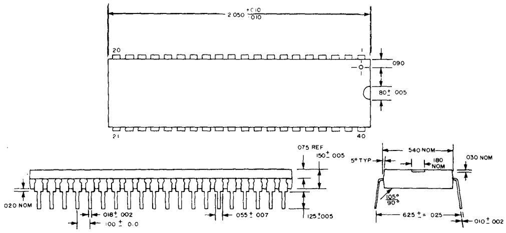
	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV	= +0.5V



PACKAGE DESCRIPTION
40 Pin Dual-In-Line Ceramic Package



PACKAGE DESCRIPTION
40-Pin Dual-In-Line Plastic Package



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	MAX CLOCK FREQ.	TEMPERATURE RANGE
MK3883N Z80-DMA	PLASTIC	2.5 MHz	0°C to 70°C
MK3883P Z80-DMA	CERAMIC	2.5 MHz	0°C to 70°C
MK3883N-4 Z80A-DMA	PLASTIC	4.0 MHz	0°C to 70°C
MK3883P-4 Z80A-DMA	CERAMIC	4.0 MHz	0°C to 70°C

PRELIMINARY

Z80 Family

MK3884/5/7 SERIAL I/O CONTROLLER

TABLE OF CONTENTS

LIST OF TABLES & FIGURES

1.0 INTRODUCTION

The MOSTEK Z80 product line is a complete set of microcomputer components, development systems and support software. The Z80 microcomputer component set includes all of the circuits necessary to build high-performance microcomputer systems with virtually no other logic and a minimum number of low cost standard memory elements.

The Z80-SIO (Serial Input/Output) circuit is a programmable, dual-channel device which provides formatting of data for serial data communication. It is capable of handling asynchronous, synchronous and synchronous bit oriented protocols such as IBM BiSync, HDLC, SDLC and virtually any other serial protocol. It can generate CRC codes in any synchronous mode and can be programmed by the CPU for any traditional asynchronous format.

1.1 STRUCTURE

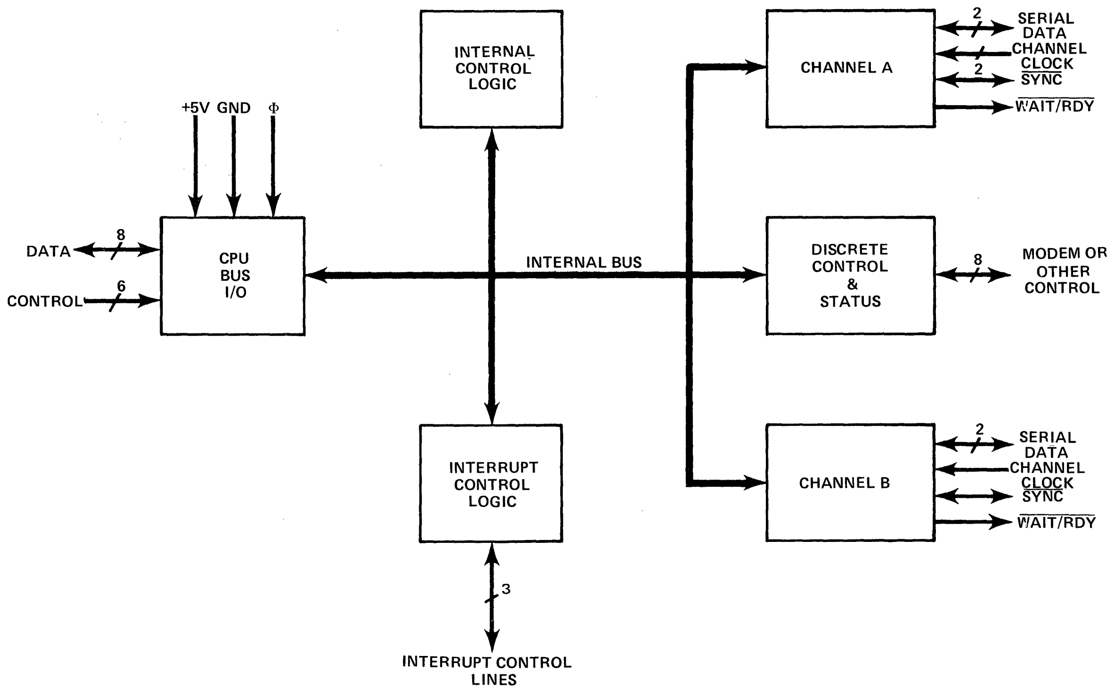
□ N-channel Silicon Gate Depletion Load Technology
□ Forty Pin DIP
□ Single 5 volt power supply
□ Single phase 5 volt clock
□ Two Full Duplex channels

1.2 FEATURES

□ Two independent full duplex channels
□ Data rates – 0 to 550K bits/second
□ Receiver data registers quadruply buffered; transmitter double buffered.
□ Asynchronous operation
  – 5, 6, 7 or 8 bits/character
  – 1, 1½ or 2 stop bits
  – Even, odd or no parity
  – x1, x16, x32 and x64 clock modes
  – Break generation and detection
  – Parity, Overrun and Framing error detection
□ Binary Synchronous operation
  – Internal or external character synchronization
  – One or two Sync characters in separate registers
  – Automatic Sync character insertion
  – CRC generation and checking
□ HDLC or IBM SDLC operation
  – Automatic Zero insertion and deletion
  – Automatic Flag insertion
  – Address field recognition
  – I-Field residue handling
  – Valid receive messages protected from overrun
  – CRC generation and checking
□ Eight modem control inputs and outputs
□ Both CRC-16 and CRC-CCITT are implemented
□ Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic.
□ All inputs and outputs fully TTL compatible.

SIO BLOCK DIAGRAM

Figure 1.0

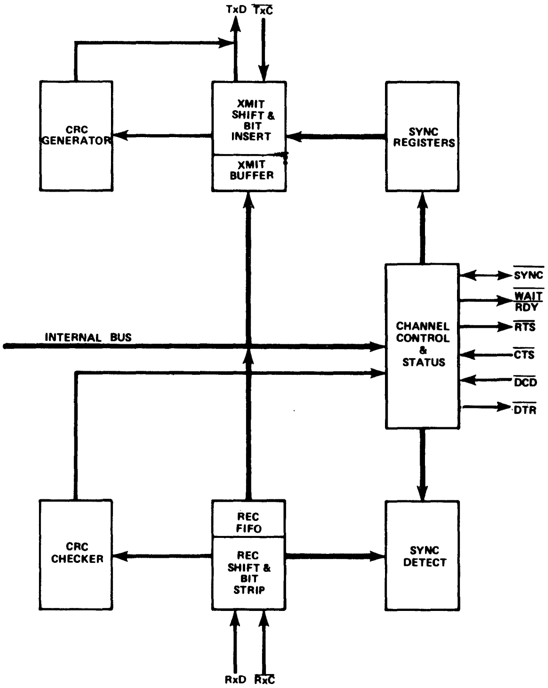


2.0 SIO ARCHITECTURE

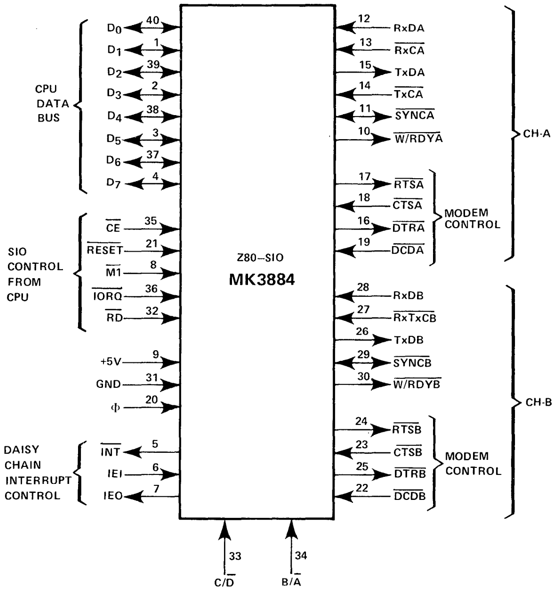
A block diagram of the SIO is shown in Figure 1. The internal structure includes a Z80-CPU bus interface, internal control and interrupt logic and two full duplex channels. The interrupt control logic determines which channel and which device within the channel is the highest priority for purposes of the automatic interrupt vectoring. Priority is fixed with Channel A assigned higher priority than Channel B and the Receiver, Transmitter and External/Status assigned priority in that order within each channel.

The channel logic is shown in block form in Figure 2. Each channel has five 8-bit control registers and three 8-bit status registers. The interrupt vector is written into an 8-bit register in Channel B and may also be read from that channel. The receiver has three 8-bit buffer registers in FIFO arrangement in addition to the 8-bit input shift register. The transmitter has one 8-bit buffer register in addition to the 8-bit output shift register. The CRC generator/checkers are 16-bit shift registers with appropriate internal feedback (programmable) for two different CRC codes.

CHANNEL BLOCK DIAGRAM *
Figure 2.0



SIO PIN OUT
Figure 2.1



*Configuration of Channel B will vary according to bonding option. See Section 2.2.

D_0-D_7	System Data Bus (bidirectional, tri-state)
$B/\overline{A}$	Channel B or A select (input high is Channel B)
$C/\overline{D}$	Control or Data select (input high is control)
$\overline{CE}$	Chip Enable (input, active low)
$\overline{M1}$	Machine Cycle One Signal from Z80-CPU (input, active low)
$\overline{IORQ}$	Input/Output request from Z80-CPU (input, active low)

$\overline{\text{RD}}$ Read Cycle Status from the Z80-CPU (input, active low)

Φ System Clock (input)

$\overline{\text{RESET}}$ Reset (input, active low) disables both receivers and transmitters. T x DA and T X DB are forced marking. Modem controls are forced high. Control registers must be rewritten after SIO is reset and before any data is transmitted or received. All interrupts are disabled.

IEI Interrupt Enable In (input, active high)

IEO Interrupt Enable Out (output, active high) IEI and IEO form a daisy-chain connection for priority interrupt control.

$\overline{\text{INT}}$ Interrupt Request (output, open drain, active low).

$\overline{\text{WAIT/READY A}}$
$\overline{\text{WAIT/READY B}}$ Two pins, one for each channel. They may be programmed to serve as ready lines for use with a DMA Controller or they may serve as wait lines to synchronize the Z80-CPU to the SIO data rate.

$\overline{\text{CTSA}}$, $\overline{\text{CTSB}}$ Clear to Send (2 pins, inputs, active low). When programmed as AUTO ENABLES, these inputs enable the transmitters of their respective channels. If these pins are not programmed as transmitter enables, they may be programmed as general-purpose input pins. These inputs are Schmitt-trigger buffered to allow slow-rise time inputs.

$\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$ Data Carrier Detect (2 pins, inputs, active low.) These pins are similar to the $\overline{\text{CTS}}$ inputs, except that they are usable as receiver enables rather than transmitter enables.

RxDA, RxDB Receive Data, (2 pins, inputs, active high.)

TxDA, TxDB Transmit Data. (2 pins, outputs, active high.)

$\overline{\text{RxCA}}$, $\overline{\text{RxCB}}$ Receiver clocks (inputs, active low.) (Two pads, one per channel. See note on Bonding Option.) Schmitt-trigger buffered.

$\overline{\text{TxCA}}$, $\overline{\text{TxCB}}$ Transmitter Clocks (inputs, active low.) (Two pads, one per channel. See note on Bonding Option.) Schmitt-trigger buffered.

$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$ Request to Send (2 pins, outputs, active low.) When the RTS bit is set, the $\overline{\text{RTS}}$ pin goes low. When the bit is reset in asynchronous mode, the pin goes high, but only after the transmitter is empty. In synchronous modes, $\overline{\text{RTS}}$ is a simple output which strictly follows the state of the $\overline{\text{RTS}}$ bit.

$\overline{\text{DTRA}}$, $\overline{\text{DTRB}}$ Data Terminal Ready (2 pins, output, active low.) Pin follows state programmed with DTR bit. (Two pads, one per channel. See note on Bonding Option.)

$\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$ External Character Synchronization (2 pins, input/output, active low.) If the External Synchronization mode is selected, assembly of characters will begin on the next rising edge of $\overline{\text{R x C}}$. If internal character sync modes are selected, the pins are outputs that are active during part of the clock cycles that a sync character is recognized. The sync condition is not latched, so this pin will be active every time a sync pattern is recognized, regardless of character boundaries. In asynchronous modes, these pins are simple inputs to the HUNT/SYNC bits in Status Register 0 and may be used for any input function desired. However, if EXTERNAL/STATUS interrupts are enabled in the asynchronus mode, then SYNC should not be left floating as this could cause spurious interrupts to occur.

NOTE: When used as an external synchronization pin, it must not become active for three system clock cycles after the previous rising edge of $\overline{\text{R x C}}$. This requirement normally can be met by allowing $\overline{\text{SYNC}}$ to change only on the falling edge of $\overline{\text{R x C}}$.

2.2 NOTE ON BONDING OPTION:

Due to package constraints, there are only five pins available for seven signals: $\overline{\text{DTRB}}$, T x T x DB, $\overline{\text{R x T x CB}}$, R x DB, $\overline{\text{SYNCB}}$, $\overline{\text{T x CB}}$ and $\overline{\text{R x CB}}$. Figure 2.2 outlines the pin out of all three SIO options. These options are designated by three different part numbers, the MK3884, MK3885 and MK3887. Since the parts differ by only the bonding option, all three parts are offered in the same packaging, frequency and temperature ranges.

SIO PIN	MK3884	MK3885	MK3887
25	$\overline{\text{DTRB}}$	T x DB	$\overline{\text{DTRB}}$
26	T x DB	$\overline{\text{T x CB}}$	T x DB
27	$\overline{\text{R x T x CB}}$	$\overline{\text{R x CB}}$	$\overline{\text{T x CB}}$
28	R x DB	R x DB	$\overline{\text{R x CB}}$
29	$\overline{\text{SYNCB}}$	$\overline{\text{SYNCB}}$	R x DB

3.0 OPERATION

Operation of the SIO is determined by the contents of the control registers. These must be programmed before any operations can be performed by the SIO. Some commands and modes may be changed during operation. The device status registers can be read at any time.

3.1 ASYNCHRONOUS MODES

The receiver ports are quadruply buffered, i.e. there are three storage registers in addition to the input shift register. This allows additional time for the CPU to service an interrupt at the beginning of a block of high-speed data transfer. The error flags are also quadruply buffered and are loaded at the same time as the character. The RECEIVER OVERRUN and PARITY ERROR flags are not reset unless an ERROR RESET Command (Command 6) is issued. END OF FRAME and CRC/FRAMING errors always reflect the state of the character currently in the buffer and are not reset by ERROR RESET. Thus, when the error status is read, it will reflect an error in the current word in the receive buffer in addition to any parity or overrun errors received since the last ERROR RESET Command. In order to keep correspondence between the stat of the error buffer and the contents of the receive registers, the status register should be read before the data (see exception). This is easily accomplished if the vectored interrupts are used since a special interrupt vector is generated for errors or end of frame.

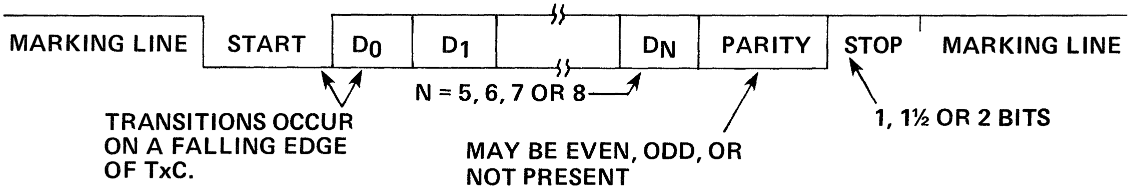
If the status is read after the data is read, the error data for the next data word will also be included if it has been stacked in the buffer. If operations are being performed rapidly enough so that the next character will not yet be received, then the status register will remain valid. The exception occurs when the RECEIVE INTERRUPT ON FIRST CHARACTER ONLY mode is selected. A special interrupt in this mode will hold error data and the character itself (even if read from the buffer) until the ERROR RESET, Command is issued. This prevents further data from becoming available in the receiver until the Reset is issued.

If the INTERRUPT ON EVERY CHARACTER mode is selected, the interrupt vector will be different if error states exist in the status register. If receiver overrun should occur despite the quadruple buffering, the most recent character received will be loaded. The character preceding it will be lost. When the character which has been written over other characters is read, the OVERFLOW bit will be set and the SPECIAL RECEIVE CONDITION vector returned if STATUS AFFECTS VECTOR is enabled.

It is possible to use the SIO in a polled environment. This requires monitoring of the RE CEIVE CHARACTER AVAILABLE bit to know when to read a character. This bit is reset automatically when the receive buffers are all empty. The TRANSMIT BUFFER EMPTY bit is high whenever the transmit buffer is empty. In polled operation, it should be checked before writing data into the transmitter to prevent overwriting of data.

ASYNCHRONOUS FORMAT

Figure 3.0



TRANSMISSION

A data character sent by the SIO will be assembled as follows in asynchronous modes:

Idle state (no characters being sent) is a marking line (high) unless a break has been programmed in the control register, in which case, the line will remain spacing until the SEND BREAK command has been removed or the chip is reset.

Transmission cannot begin unless the TRANSMIT ENABLE bit is set. If the AUTO ENABLES bit is selected, then $\overline{\text{CTS}}$ must be low as well. If the 5 bits/character mode is selected, then unused bits (D_5, D_6, and D_7) must be zero in each data byte written into the SIO.

RECEIVING

Asynchronous reception will begin when the RECEIVER ENABLE bit is set. If the AUTO ENABLES bit is selected, the $\overline{\text{DCD}}$ must be low as well: A low (spacing) condition on R x D indicates a start bit. If the low persists for ½ bit time, the start bit is assumed to be valid and the data input is then sampled at mid-bit time until the entire character is assembled. This method of detecting a start bit improves error rejection when noise spikes exist on an otherwise marking line. If the X1 clock mode is selected, bit synchronization must be accomplished externally.

3.2 SYNCHRONOUS MODES

The various synchronous modes all require a x1 clock for transmission and reception. Data is sampled on the rising edge of $\overline{\text{RxC}}$. Transmitter data transitions occur on the falling edge of $\overline{\text{TxC}}$.

In all cases, the receiver is in a hunt mode after a reset (internal or external). The hunt can begin only when the receiver is enabled. Only when character synchronization has been achieved can data transfer begin. If there is a loss of character synchronization, the hunt mode can be re-entered by writing a control word with the ENTER HUNT MODE bit set.

The differences in operation of the monosync, bisync and external sync modes are only in the manner in which initial synchronization is achieved. Note: The mode of operation must be selected before the sync characters are loaded, since the registers are used differently in the various modes.

MONOSYNC; (8-BIT SYNC MODE)

Matching of a single sync character, programmed into Write register 7, implies character synchronization, which enables data transfer.

BISYNC: (16-BIT SYNC MODE)

Matching of two adjacent sync characters programmed in Write Registers 6 and 7 implies character synchronization. In both monosync and bisync modes, the $\overline{\text{SYNC}}$ pin will be active (low) any time the sync character sequence is detected and will remain low for the clock cycle in which it is detected.

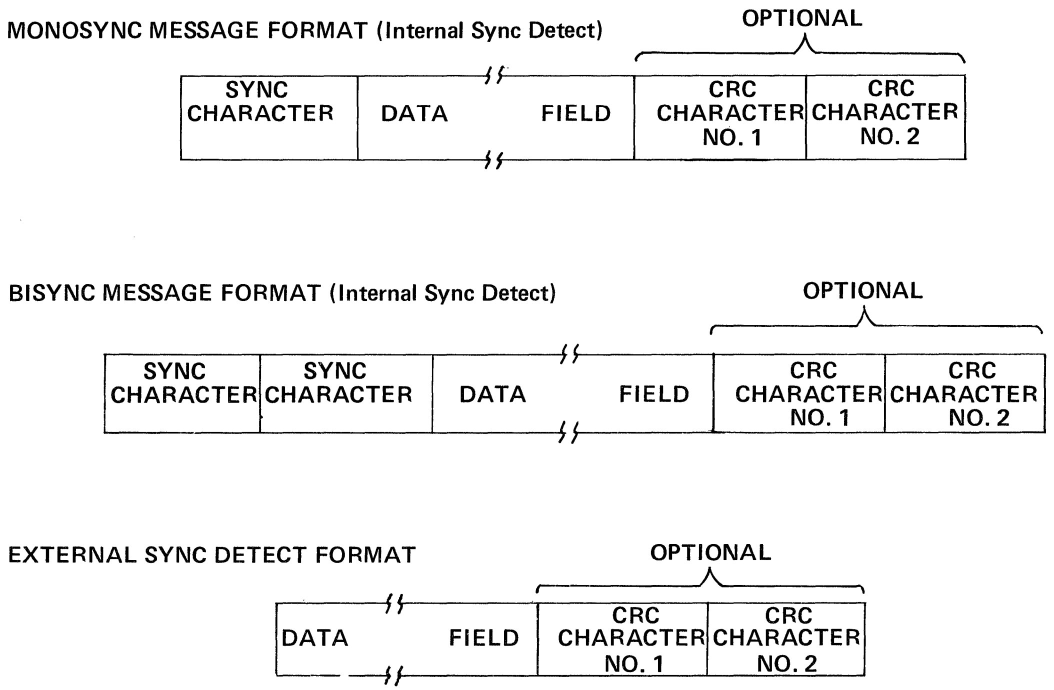
EXTERNAL SYNC MODE

In this mode, character assembly begins on the first rising edge of $\overline{\text{RxC}}$ after the $\overline{\text{SYNC}}$ pin becomes active (low). It should be held active for at least three complete clock cycles.

In Monosync, Bisync and External sync modes, assembly will continue until the SIO is reset (either internally or with the Reset pin) or until the receiver is disabled (by command or with the $\overline{\text{DCD}}$ pin in the AUTO ENABLES mode) or until the CPU sets the ENTER HUNT MODE bit.

After initial synchronization has been achieved, the Monosync, Bisync, and External Sync modes are very similar. Any differences will be noted in the following which is meant to apply to all three modes.

SYNCHRONOUS FORMATS
Figure 3.1


MONOSYNC MESSAGE FORMAT (Internal Sync Detect)

BISYNC MESSAGE FORMAT (Internal Sync Detect)

EXTERNAL SYNC DETECT FORMAT

Synchronous Modes (Except SDLC) Transmission:

A. Default state (after a Reset or transmitter not enabled) is a marking Line. Break may be programmed to generate a spacing line, which begins as soon as programmed, regardless of the contents of the send register. With the transmitter enabled, and after modes have been selected, default is continuous transmission of the 8 or 16 bit sync character.

B. Several Interrupt modes are possible:

1. Transmit interrupts enabled — every time that the transmit buffer becomes empty, an interrupt will be generated if the TRANSMIT INTERRUPT ENABLE bit is set. The interrupt may be satisfied by either writing another character into the transmitter or by resetting the TRANSMITTER INTERRUPT PENDING bit with the RESET TRANSMITTER INTERRUPT PENDING command (Command 5). If the interrupt is satisfied with this command and nothing more is written into the transmitter, there will be no further transmitter interrupts, as it is the buffer becoming empty that causes the interrupt. When another character is written, the the transmitter can again become empty and interrupt again.

2. External/Status interrupts enabled — If the EXTERNAL/STATUS INTERRUPT ENABLE bit is set, Transmitter conditions such as starting to send CRC characters, starting to send Sync characters, $\overline{\text{DCD}}$, $\overline{\text{SYNC}}$, and $\overline{\text{CTS}}$ changing state cause interrupts, which have a unique vector if STATUS AFFECTS VECTOR mode is selected.

3. All interrupts may be disabled for operation in a polled mode or to prevent interrupts at inappropriate times in a program's execution.

C. If CRC is not enabled, sync characters will automatically be inserted when the transmitter has no data to send. An interrupt is generated only after the first automatically inserted sync character has been loaded. If CRC is enabled, the first time the transmitter has no data to send, the 16-bit CRC is automatically sent, followed by sync characters. While sending CRC, the Tx UNDERRUN/EOM bit is set and the TRANSMIT BUFFER EMPTY bit indicates full. CRC is not calculated on the automatically inserted sync characters, but it will be calculated on any sync character sent as data unless the CRC generator is disabled when that character is loaded to the transmit shift register from the transmit buffer. When the CRC has been sent, the TRANSMIT BUFFER EMPTY bit goes high and an interrupt is generated to indicate that another message can begin. Control of the CRC generator may proceed as follows:

The CRC generator should be reset by issuing the RESET TRANSMIT CRC GENERATOR Command, before any data is loaded. After CRC and the entire transmitter is enabled, data may be loaded. Before CRC is to be sent (but after the first data has been loaded), the SNEDING CRC/SYNC bit must be reset with the RESET Tx UNDERRUN/EOM Command.

Because sending of the CRC is inhibited when the Tx UNDERRUN/EOM bit is set, the SIO can be used to automatically insert fill characters within messages instead of automatically sending the CRC. CRC is not calculated on syncs automatically inserted and when the end of the message is reached, the bit can be reset thus allowing the CRC to be sent.

D. If the transmitter is disabled while a character is being sent, that character (whether Data, CRC or SYNC) will be sent as normal but will be followed by a marking line rather than CRC or sync characters. A character in the buffer when the transmitter is disabled will remain in the buffer. However, a programmed break will be effective as soon as it is written into the control register. Characters being transmitted, if any, will be lost.

E. In all modes, characters are sent low-order bits first, i.e., D_0 before D_1, etc. for as many bits as are programmed. This requires right-hand justification of data to be transmitted if word length is less than 8 bits. If word length is 5 bits or less, the special technique described in the TRANSMIT BITS/CHAR section must be used for the data format.

Synchronous Modes (Except SDLC) Reception:

A. After programming the mode and sync characters (in that order), the receiver may be enabled. It will then be in the HUNT MODE and will stay in that mode until:

1. A match is made with a single sync character (monosync mode) or

2. A match is made with a dual sync character (BiSync mode) or

3. The external $\overline{\text{SYNC}}$ pin is forced low. In cases (1) and (2) the external $\overline{\text{SYNC}}$ pin is an output which indicates that character synchronization has been achieved. In case (3) it is an input.

B. Character assembly begins after sync has been achieved. Four interrupt modes are possible.

1. NO INTERRUPTS ENABLED – for a purely polled operation or for "off line" conditions.

2. INTERRUPT ON FIRST CHARACTER ONLY. This mode would normally be used to start a software polling loop or a block transfer instruction using the WAIT/READY output to synchronize the CPU to the incoming data rate. It could also be used with a DMA device. In this mode, the SIO will interrupt on the first character and thereafter will only interrupt if errors are detected. The mode is reset with the RESET RECEIVE INTERRUPT ON FIRST CHARACTER command (Command 4).
   The first character received after this command is issued will also cause an interrupt. If External/Status interrupts are enabled, they may interrupt at any time. Parity errors do not cause interrupts in this mode, but End-of-Frame (SDLC Mode) and receiver overrun do cause interrupts.

3. INTERRUPT ON EVERY CHARACTER – whenever the receiver buffer has a character an interrupt is generated. Error and special conditions generate a special vector if the STATUS AFFECTS VECTOR mode is selected. A parity error may optionally not generate the special vector.

C. CRC checking generation may be used in the synchronous modes.

1. Calculation of the CRC on a particular character begins 8 bit times after the word has been transferred to the receive buffer. If CRC is enabled before the next character is transferred to the receive buffer, CRC will be calculated on the character. If CRC is disabled before the time of the next transfer, calculation will proceed on the word in progress, but the word just transferred to the buffer will not be included. This allows starting and stopping CRC checking on the various characters employed in BiSync.

2. The CRC may be enabled and disabled as many times as necessary for a given calculation.

3. CRC Codes are selected during the mode selection process. Either the CRC-16 polynominal $X^{16} + X^{15} + X^2 + 1$ or the SDLC polynomial $X^{16} + X^{12} + X^5 + 1$ may be used. In all except SDLC mode, the CRC calculator and checker are reset to all 0's. Transmitter and receiver must use the same polynomial.

4. In Monosync, Bisync and External Sync modes, the CRC/FRAMING ERROR bit contains the result of the comparison of the CRC checker to "all zeros" 16 bit times after the character has been loaded from the receive shift register to the buffer. The comparison is made with each load and is valid only as long as the character remains in the buffer. If time increases down the page, then the following holds:

   Character "A" loaded into the buffer

   Character "B" loaded into the buffer . . .
   If CRC is disabled before "C" is in the buffer it will not be calculated on "B".

   Character "C" loaded into buffer . . .
   After "C" is loaded the CRC/FRAMING ERROR bit shows the result of the comparison thru Character "A"

   Character "D" loaded into buffer . . .
   After "D" is in buffer, the CRC/FRAMING ERROR bit shows the result of the comparison thru Character "B".

   Because of the serial operation of the CRC calculation, the receiver clock ($\overline{\text{RxC}}$) must go through 16 cycles after the CRC character has been loaded into the receive buffer (20 cycles after the last bit is at the SIO RxD pin) before the CRC calculation is complete.

TRANSMISSION SDLC/HDLC MESSAGE FORMAT

Figure 3.2

FLAG 01111110	ADDRESS 8 BITS	DATA FIELD "I"	CRC No. 1	CRC No. 2	FLAG 01111110

SDLC MODE TRANSMISSION:

A. Normally, the CRC generator should be reset (with the RESET TRANSMIT CRC GENERATOR command) before a data block is transmitted. Reset may occur any time after the CRC of the previous message has been sent. During the time that CRC is being sent the Tx UNDERRUN/EOM bit will be set, the TRANS BUFFER EMPTY bit will not be set. After the CRC has been sent the TRANS BUFFER EMPTY bit is set which will cause an interrupt signifying that the CRC has been sent, if transmit interrupts are enabled.

B. The idle device state (if the transmitter is enabled) is continuous flags being transmitted. If the transmitter is not enabled, a marking line is sent (idle line state).

C. An abort sequence may be sent by issuing the SEND ABORT command (Command 1). This causes at least 8 but less than 14 one's to be sent before the line reverts to continuous flags. Any data being transmitted and any data in the transmit buffer will be lost.

D. One to 8 bits per character may be sent. See the Register Description of Write Register 5, Transmit Bits/Char. for an explanation of how this is accomplished. Since the number of bits/character may be changed "on the fly", this feature may be used to fill a data field with any number of bits. When used in conjunction with the Receiver Residue Codes, the SIO may receive a message of any number of bits length and retransmit it exactly as received with no previous information about the character structure of the I-field (if any). A change in the number of bits/character will not affect the character in the process of being shifted out. Characters will be sent with the number of bits programmed at the time that the character is loaded from the buffer to the transmitter.

E. As in other synchronous modes, the two byte CRC sequence will be sent automatically when the transmitter has no more data to send, i.e. when there is no character in the transmit buffer and the transmit shift register is empty. When the CRC sending begins, the Tx UNDERRUN/EOM bit is set and a status change interrupt is generated if external/status interrupts are enabled. This may be used as a transmitter underrun indication. After the CRC has been sent, the line reverts to continuous flags, without shared zeros, i.e. . . .
0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 . . .

Control of the CRC generator may proceed as follows:

0. Set up necessary mode (only at initial power on)
1. Reset CRC generator
2. Write first 2 bytes of data (i.e. address and or control bytes)
3. Reset Tx UNDERRUN/EOM bit
4. Write rest of data
5. After data is complete, CRC & flags will be sent automatically, and this sequence can repeat from 1.

F. Extra zeros are automatically inserted in the data stream where required to fulfill the requirement of 5 ones maximum in a row, except for flags or aborts.

G. When SDLC mode is selected, Reset of the CRC generator is actually a preset to all 1's. The SDLC CRC code must be selected.

RECEPTION SDLC/HDLC MESSAGE FORMAT

Figure 3.3

FLAG 01111110	ADDRESS 8 BITS	DATA "I"	FIELD	CRC NO. 1	CRC NO. 2	FLAG 01111110

SDLC OPERATION, RECEIVER

A. Data transfer begins with the first non-flag character received after at least one flag (01111110) has been received if ADDRESS SEARCH MODE has not been enabled. If ADDRESS SEARCH MODE is enabled, then a flag followed by either the programmed address or the global address (1111111) is required before data transfer will begin.

1. If interrupts are disabled, the presence of characters in the receive buffer can be detected by observing the RECEIVE CHARACTER AVAILABLE bit in Read Register 0.
2. If the INTERRUPT ON FIRST CHARACTER ONLY mode has been selected, this would normally be used to initiate a block transfer. If the length of the message is unknown, the SPECIAL RECEIVE CONDITION (End of Frame) interrupt may be used to exit the instruction of software loop. The RESET INTERRUPT ON FIRST CHARACTER command (Command 4) must be issued before an interrupt for a following block's first character can be operated.
3. Flags are not transferred. The extra zeros inserted in transmission are automatically deleted.
4. Aborts are detected as 7 or more one's and cause a status interrupt (if enabled) with the BREAK/ABORT bit set in Read Register 0. After the RESET EXTERNAL/STATUS INTERRUPT command (Command 2) has been issued, a second interrupt will occur when the coninuous one's condition has been cleared.

B. In SDLC mode, control of the receive CRC checker is automatic. It is reset by the leading flag and CRC is calculated up to the final flag. The CRC/FRAMING ERROR bit indicates the result of the CRC check and is located in Read register 1. If the CRC/FRAMING ERROR bit is not set, then the CRC indicates a valid message. A special check sequence is used for the SDLC check because of the preset to all one's. The final check must be
0001110100001111.

C. Character length may be changed "on the fly." If address and control bytes are processed as 8-bit characters, the receiver may be switched to a smaller character length during the time that the first information character is being assembled. This change must be made quickly enough so that it is effective before the number of bits specified have been assembled, i.e., if the change is to be from the 8-bit control to a 7-bit information field character length, the change must be made before the first 7 bits of the I-field have been assembled.

D. If address search mode is not used, or if messages have multi-byte addresses, an unwanted message need not be completely read by the CPU. Once the determination has been made that the message is not needed, writing the ENTER HUNT MODE bit will suspend reception until another message headed by a flag has been received.

E. When the trailing flag is received, an interrupt with a special vector is generated (if enabled). This signals that the byte with the END OF FRAME bit set has been received. In addition to the results of the CRC check. Read Register 1 has 3 bits of Residue Code valid at this time. For those cases in which the number of bits in the I-field is not an integral multiple of the character length used, these bits indicate the boundary between the CRC check bits and the I-field bits. For a detailed description of the meaning of these bits, see the description of the Residue Codes in Read Register 1.

F. Parity checking may be used on data in the information field only if 5-7 bit characters are used and only if a half-duplex protocol is being used. (There are no separate controls for parity on the receiver and transmitter so parity cannot, for example, be simultaneously disabled for transmitting an 8-bit address and enabled for receiving a 5-bit I-field character).

4.0 SIO PROGRAMMING

4.1 GENERAL

The Z80-SIO is a multi-function peripheral component specifically designed to satisfy a wide variety of serial data communications requirements in microcomputer systems. Its basic role is that of a serial to parallel, parallel to serial converter/controller but within that role it is configured by systems software programming so that its function or "personality" can be optimized for a given serial data communications application.

To program the Z80-SIO the systems software issues a series of commands that initialize the basic mode of operation desired and other commands to qualify conditions within the mode selected i.e. Stop Bits, Bits/Char, Sync Char etc. The command structure of the Z80-SIO is designed to take advantage of the powerful Z80 BLOCK I/O instructions to simplify programming, minimize overhead and optimize CPU interaction activities.

Each of the two channels of the Z80-SIO contain command registers that must be programmed via system software prior to functional operation. The channel select input (B/$\overline{A}$) and the control/data input (C/$\overline{D}$) are the command structure addressing controls, normally controlled by the address bus of the Z80 CPU.

C/$\overline{D}$	B/$\overline{A}$	FUNCTION
0	0	**Channel A Data**
0	1	**Channel B Data**
1	0	**Channel A Commands/Status**
1	1	**Channel B Commands/Status**

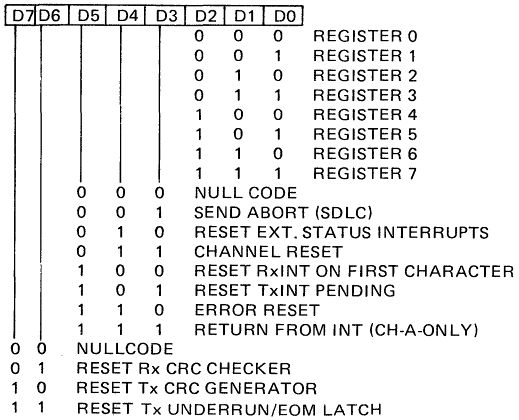
4.2 WRITE REGISTERS

The Z80-SIO contains eight (8) registers that are programmed (written into) by the system software to configure the functional personality of each channel. All Write Registers, with the exception of Write Register 0, require two bytes to be properly programmed. The first byte contains 3 bits which point to the selected register (D0-D2) the second byte is the actual control word that is being written that register to configure the SIO. WRITE Register 4 (WR4) parameters must be issued before WR1, WR3, and WR5 parameters or commands.

Write Register 0 is a special case. RESET (either internal command or external input) will initialize the SIO to Write Register 0. All basic commands (CMD0-CMD2) and CRC controls (CRC0, CRC1) can be accessed with a single byte using Write Register 0.

Contained in the first byte of any Write Register access are the basic commands (CMD0-CMD2) and the CRC controls (CRC0, CRC1) so that maximum system control and flexibility is maintained.

WRITE REGISTER 0



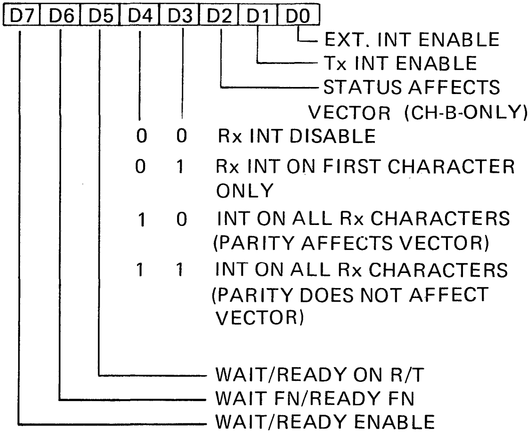
D7 D6 D5 D4 D3 D2 D1 D0

D2	D1	D0	
0	0	0	REGISTER 0
0	0	1	REGISTER 1
0	1	0	REGISTER 2
0	1	1	REGISTER 3
1	0	0	REGISTER 4
1	0	1	REGISTER 5
1	1	0	REGISTER 6
1	1	1	REGISTER 7

D5	D4	D3	
0	0	0	NULL CODE
0	0	1	SEND ABORT (SDLC)
0	1	0	RESET EXT. STATUS INTERRUPTS
0	1	1	CHANNEL RESET
1	0	0	RESET RxINT ON FIRST CHARACTER
1	0	1	RESET TxINT PENDING
1	1	0	ERROR RESET
1	1	1	RETURN FROM INT (CH-A-ONLY)

D7	D6	
0	0	NULLCODE
0	1	RESET Rx CRC CHECKER
1	0	RESET Tx CRC GENERATOR
1	1	RESET Tx UNDERRUN/EOM LATCH

WRITE REGISTER 1



D7 D6 D5 D4 D3 D2 D1 D0

D0: EXT. INT ENABLE

D1: Tx INT ENABLE

D2: STATUS AFFECTS VECTOR (CH-B-ONLY)

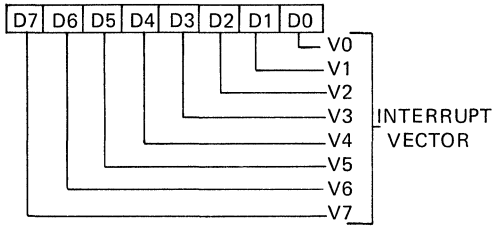
D4	D3	
0	0	Rx INT DISABLE
0	1	Rx INT ON FIRST CHARACTER ONLY
1	0	INT ON ALL Rx CHARACTERS (PARITY AFFECTS VECTOR)
1	1	INT ON ALL Rx CHARACTERS (PARITY DOES NOT AFFECT VECTOR)

D5: WAIT/READY ON R/T

D6: WAIT FN/READY FN

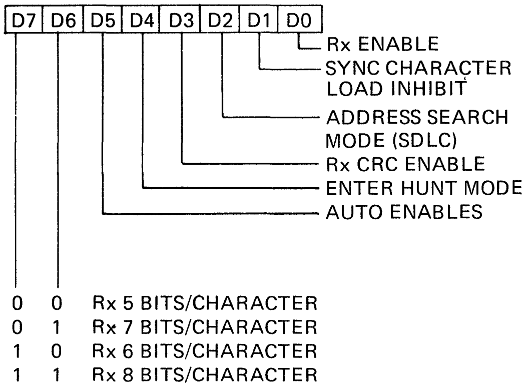
D7: WAIT/READY ENABLE

WRITE REGISTER 2 *

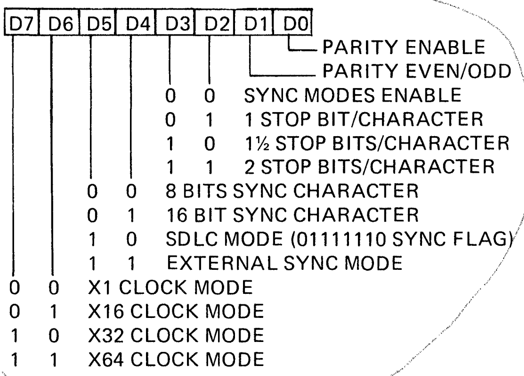


*CAN ONLY BE WRITTEN INTO CHANNEL B

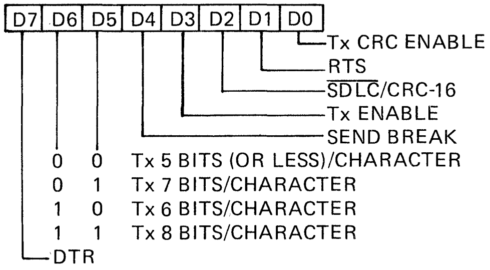
WRITE REGISTER 3



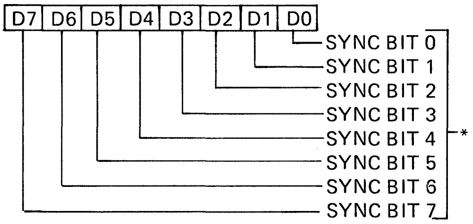
WRITE REGISTER 4



WRITE REGISTER 5

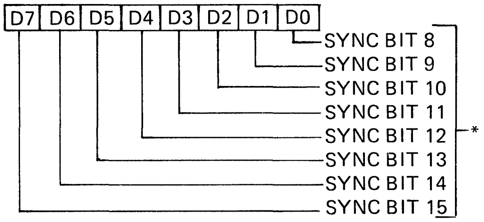


WRITE REGISTER 6



*ALSO SDLC ADDRESS FIELD

WRITE REGISTER 7



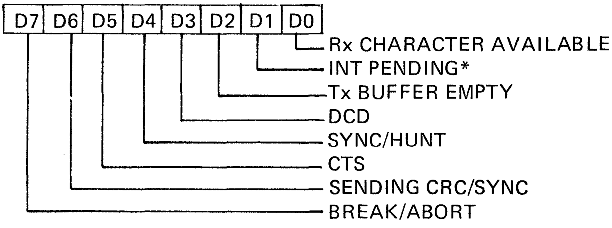
*FOR SDLC IT MUST BE PROGRAMMED TO "01111110" FOR FLAG RECOGNITION

4.3 READ REGISTERS

The Z80-SIO contains three (3) registers that can be read to obtain the status of each channel. Status information includes error conditions, interrupt vector, and standard communication interface protocol signals. To read the contents of a selected Read Register the system software must first write out to the SIO the byte containing pointer information (DO-D2) in exactly the same manner as a Write Register operation. Then by issuing a READ operation the contents of the addressed Read/Status Register can be read by the Z80-CPU.

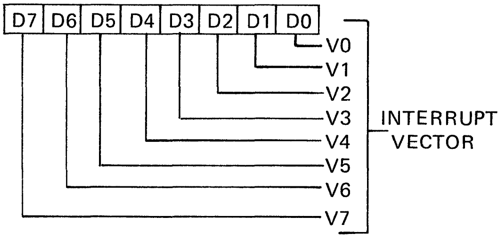
The real power in this type of command structure is that the programmer has complete freedom after pointing to the selected register of either Reading or Writing to initialize or test that register. By designing software to initialize the Z80-SIO in a modular, structured fashion, the programmer can use the powerful Z80 BLOCK I/O instructions to significantly simplify and speed his software development and debug.

READ REGISTER 0

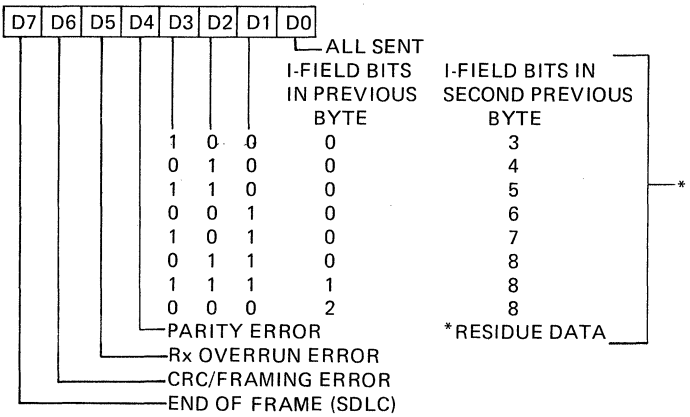


*CAN ONLY BE READ BY CHANNEL A

READ REGISTER 2 (Channel B Only)



READ REGISTER 1



D3	D2	D1	I-FIELD BITS IN PREVIOUS BYTE	I-FIELD BITS IN SECOND PREVIOUS BYTE
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

*RESIDUE DATA

4.4 REGISTER DESCRIPTION

Each channel contains the following control registers, addressed as commands (not data):

Write Register 0, a command register:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CRC Reset Code 1	CRC Reset Code 0	CMD 2	CMD 1	CMD 0	PNT 2	PNT 1	PNT 0

$PNT_0 - PNT_2$ (D_0-D_2)

These are pointer bits which tell the SIO into which register the following byte is to be written. The first byte written into each channel after a reset (either by command or with the external reset pin) will go to write register 0. The byte following a read or write to any register (not register 0) will be to register 0.

CMD_0 to CMD_2 (D_3-D_5)

These are commands:

Command	CMD_2	CMD_1	CMD_0	
0	0	0	0	Null Command (no affect)
1	0	0	1	Send Abort (SDLC Mode)
2	0	1	0	Reset External/Status Interrupts
3	0	1	1	Channel Reset
4	1	0	0	Reset Receive Interrupt on First Character
5	1	0	1	Reset Transmitter Interrupt Pending
6	1	1	0	Error Reset (latches)
7	1	1	1	Return from Interrupt (Channel A only)

COMMAND 0 (The NULL command) has no affect. It's normal use is to do nothing while setting the pointers for a following byte.

COMMAND 1 (SEND ABORT) is used only with the SDLC mode to generate a sequence of 8 to 13 ones.

COMMAND 2 (RESET EXTERNAL/STATUS INTERRUPTS). After an external or status interrupt (indicating a change on a modem line or a break condition, for example) the status bits of Read Register 0 are latched. This command reenables them and allows interrupts to occur. The latching allows capture of short pulses on the inputs until such time as the CPU can read the change.

COMMAND 3 (CHANNEL RESET). This command performs the same operation as an external reset, but only on a single channel. The Channel A Reset also resets the interrupt prioritization logic. All control registers must be rewritten after this command. After this command is written, four extra system (Φ) clock cycles should be allowed for the SIO reset time before any additional commands or controls are written into that channel of the SIO.

COMMAND 4 (RESET RECEIVE INTERRUPT ON FIRST RECEIVE CHARACTER.) If the INTERRUPT ONLY ON FIRST RECEIVE CHARACTER mode of operation is programmed, it needs to be reactivated after each complete message is received, in preparation for the next message.

COMMAND 5 (RESET TRANSMITTER INTERRUPT PENDING.) The transmitter will interrupt when it becomes empty if the ENABLE TRANSMIT INTERRUPT mode is selected. In those cases when there are no additional characters to be sent, issuing this command will prevent further transmitter interrupts (i.e. until after the next character has been loaded into the transmitter).

COMMAND 6 (ERROR RESET, LATCHES.) Parity and overrrun errors are latched in Read Register 1 until reset with this command. This allows errors occurring in block transfers to be examined only at the end of the block.

COMMAND 7 (RETURN FROM INTERRUPT.) This command (which must be issued in Channel A) is interpreted by the SIO in exactly the same way as it would interpret an RETI Command on the data bus, i.e. it would reset the Interrupt Under Service latch of the internal device (receiver, transmitter, etc.) under service and thus, by means of the daisy chain, allow lower priority devices to interrupt. The internal daisy chain may be used even in systems with no external daisy chain and no RETI Command by use of this command.

CRC RESET CODE 0 (D_6) and CRC RESET CODE 1 (D_7)

Together, these bits specify three reset modes.

CRC Reset Code 1	CRC Reset Code 0	
0	0	Null Code (no affect)
0	1	Reset Receive CRC Checker
1	0	Reset Transmit CRC Generator
1	1	Reset Transmit UNDERRUN/EOM latch

WRITE REGISTER 1 contains the control bits for the various interrupt and WAIT/READY modes.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Wait/ Ready Enable	ReadyFN/ WaitFN	W/Ready On R/T	Receive interrupt Mode 1	Receive Interrupt Mode 0	Status Affects Vector	Trans Interrupt Enable	Ext Interrupts Enable

EXT INT ENABLE (D_0)

External Interrupt Enable, allows interrupts to occur as a result of transitions on the $\overline{\text{DCD}}$, $\overline{\text{CTS}}$ or $\overline{\text{SYNC}}$ lines or as a result of a Break Condition or the beginning of sending CRC or sync characters. $\overline{\text{DCD}}$, $\overline{\text{CTS}}$, or $\overline{\text{SYNC}}$ if not used, should be pulled up to V_{CC} to prevent spurious interrupts from occuring.

TRANS INT ENABLE (D_1)

Transmitter Interrupt Enable. If enabled, interrupts will occur whenever the transmitter buffer becomes empty.

STATUS AFFECTS VECTOR (D_2) (Channel B only)

If this mode is selected, the vector returned from an interrupt acknowledge cycle will be variable according to the following:

	V_3	V_2	V_1	
Ch B	0	0	0	Ch B Transmit Buffer Empty
	0	0	1	Ch B External/Status Change
	0	1	0	Ch. B Receive Character Available
	0	1	1	Ch B Special Receive Condition
Ch A	1	0	0	Ch A Transmit Buffer Empty
	1	0	1	Ch A External/Status Change
	1	1	0	Ch A Receive Character Available
	1	1	1	Ch A Special Receive Condition

If this bit is 0, the fixed vector programmed in the vector register is returned.

REC INT MODE 0 (D_3), REC INT MODE 1 (D_4)

Receive Interrupt Mode 0 and Receive Interrupt Mode 1 together specify the various character available conditions:

MODE	D_4 REC INT MODE 1	D_3 REC INT MODE 0	
0	0	0	Receiver interrupts disabled
1	0	1	Receive interrupt on first character only
2	1	0	Interrupt on all Receive Characters-Parity affects Vector
3	1	1	Interrupt on all Receive Characters-Parity error does not affect Vector.

W/READY on R/T (D_5)

When the W/Ready line is enabled, this bit selects whether it will be active when the receiver is empty (bit=1) or when the transmit buffer is full (bit =0).

READY FN/WAIT FN (D_6)

When used with the CPU as a Wait line, this bit should be programmed "0". When used with a DMA as a Ready line, it must be programmed "1". The ready function can occur any time, regardless of whether the SIO is addressed or not. The Wait function is active only if the CPU attempts to read SIO data that has not yet been received, as would frequently occur if block transfer instructions are used with the SIO, or tries to write data while the transmit buffer is still full.

Also, as a Wait function, the output is open drain and occurs from the negative edge of Φ. As a Ready function, it is actively driven high and occurs from the positive edge of Φ.

WAIT/READY ENABL (D_7)

The Wait/Ready pin will remain high (Ready mode) or floating (Wait mode) until this bit is programmed to one.

WRITE REGISTER 2 (Channel B only)

Write Register 2 is the interrupt vector register and it exists only in Channel B. V_4-V_7 and V_0 are always returned exactly as written. V_1-V_3 are returned as written if the "Status Affects Vector", Control bit is "0".

WRITE REGISTER 3

Write register 3 contains control bits for some of the receiver logic.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
RCVR Bits/ Char 0	RCVR Bits/ Char 1	Auto Enables	Enter Hunt Mode	RECVR CRC Enable	Address Search Mode	Sync Char Load Inhibit	Receiver Enable

RECEIVER ENABLE (D_0)

A "1" programmed here allows receiver operations to begin.

SYNC CHAR LOAD INHIBIT (D_1)

Sync characters preceding a message will not be loaded into the receiver buffers if this option is selected. The CRC calculation is not stopped by the sync character being stripped.

ADDRESS SEARCH MODE (D_2)

If the SDLC mode is selected, this mode will cause messages with addresses not matching the programmed address or the global (11111111) address to be rejected, i.e., no interrupts occur unless an address match occurs if this mode is selected.

RECVR CRC ENABLE (D_3)

Receiver CRC Enable. If this bit is set, a calculation of CRC begins (or restarts) at the start of the last character transferred from the receive register to the buffer stack regardless of the number of characters in the stack.

ENTER HUNT MODE (D_4)

If character synchronization is lost for any reason, or if in SDLC mode, it is determined that the contents of an incoming message are not needed, Hunt mode may be reentered by writing a "1" to this bit.

AUTO ENABLES (D_5)

If this mode is selected, the $\overline{DCD}$ and $\overline{CTS}$ inputs are receiver and transmitter enables, respectively. If the mode is not selected, $\overline{DCD}$ and $\overline{CTS}$ are only inputs to their corresponding bits in Read Register 0.

RCVR BITS/CHAR 1 (D_6), RCVR BITS/CHAR 0 (D_7)

These bits together determine the number of serial receive bits that will be assembled to form a character.

These bits may be changed during the time that a character is being assembled, if it is done before the number of bits currently programmed is reached.

D7 Receiver Bits/Character 1	D_6 Receiver Bits/Character 0	Bits/Character
0	0	5
0	1	7
1	0	6
1	1	8

WRITE REGISTER 4

Write Register 4 contains control bits affecting both the receiver and transmitter.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Clock Rate 1	Clock Rate 0	Sync Modes 1	Sync Modes 0	Stop Bits 1	Stop Bits 0	Parity Even/ $\overline{\text{Odd}}$	Parity

PARITY (D_0)

If this bit is set, an additional bit position (in addition to those specified in the bits/character control) is added to transmitted data and is expected in receive data.

PARITY EVEN/$\overline{\text{ODD}}$ (D_1)

If parity is specified, this bit determines whether it is sent or checked as even or odd parity.

STOP BITS 0 (D_2), STOP BITS 1 (D_3)

These bits determine the number of stop bits added to each asynchronous character sent. The receiver always checks for one stop bit.

The special (00) mode is used to signify that a synchronous mode is to be selected.

D_3 Stop Bits 1	D_2 Stop Bits 0	
0	0	Sync Modes
0	1	1 Stop Bit Per Character
1	0	1½ Stop Bits Per Character
1	1	2 Stop Bits Per Character

SYNC MODES 0 (D_4), SYNC MODES (D_5)

These select the various options for character synchronization:

Sync Mode 1	Sync Mode 0	
0	0	8-bit programmed sync
0	1	16-bit programmed sync
1	0	SDLC Mode (01111110 sync pattern)
1	1	External Sync Mode

CLOCK RATE 0 (D_6), CLOCK RATE 1 (D_7)

Specifies the multiplier between clock and data rates. For synchronous modes X1 must be specified. Any rate may be specified for the asynchronous modes. The same multiplier is used for both the receiver and transmitter.

In all modes, the system clock (Φ) must be at least 5 X the data rate. If the X1 clock rate is selected, bit synchronization must be accomplished externally.

Clock Rate 1	Clock Rate 0	
0	0	Data Rate X 1 = Clock Rate
0	1	Data Rate X16 = Clock Rate
1	0	Data Rate X32 = Clock Rate
1	1	Data Rate X64 = Clock Rate

WRITE REGISTER 5

Write Register 5 contains mostly control bits affecting the transmitter.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DTR	Transmit Bits/ Char 0	Transmit Bits/ Char 1	Send Break	Transmit Enable	$\overline{\text{SDLC}}$/ CRC16	RTS	Transmit CRC Enable

TRANSMIT CRC ENABLE (D_0)

This bit determines whether CRC is to be calculated on any particular send character. If set at the time of loading the character from the transmit buffer to the transmit shift register, CRC will be calculated on the character. CRC will not be automatically sent unless this bit is set when the transmitter is completely empty.

RTS (D_1)

Request to Send is the control bit for the $\overline{\text{RTS}}$ pin. When the $\overline{\text{RTS}}$ bit is set, the $\overline{\text{RTS}}$ goes active (low). When the bit is reset (to 0), the $\overline{\text{RTS}}$ pin will go inactive (high) only after the transmitter is empty.

$\overline{\text{SDLC}}$/CRC-16 (D_2)

This bit selects the CRC code used by both the transmitter and the receiver. When reset, the SDLC polynomial $X^{16} + X^{12} + X^5 + 1$ is used. (In SDLC mode, the registers are preset to "all 1's" and a special check sequence is used.) When set, the CRC-16 polynomial $X^{16} + X^{15} + X^2 + 1$ is used, and the CRC registers are reset to "all 0's".

TRANSMIT ENABLE (D_3)

Data will not be transmitted and the TxD pin will be held marking (high) until this bit is set. Data or Sync characters in the process of being transmitted will be completely sent if the transmit enable bit is reset after transmission has started. CRC characters will not be completely sent if the transmitter is disabled during the sending of a CRC character.

SEND BREAK (D_4)

When set, this bit directly forces the TxD pin spacing, regardless of any data being transmitted. When reset, the TxD pin is released.

TRANSMIT BITS/CHAR 0 (D_5), TRANSMIT BITS/CHAR 1 (D_6)

These bits together control the number of bits that will be sent from each byte transferred to the transmit buffer.

D_6	D_5	
Transmit Bits/ Character 1	Transmit Bits/ Character 0	Bits/Character
0	0	5 or less
0	1	7
1	0	6
1	1	8

Bits to be sent are assumed to be right justified. Low order bits (D_0) are sent first. The "5 or less" mode allows transmission of 1 to 5 bits in a character.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
1	1	1	1	0	0	0	D	Sends one bit
1	1	1	0	0	0	D	D	Sends two bits
1	1	0	0	0	D	D	D	Sends three bits
1	0	0	0	D	D	D	D	Sends four bits
0	0	0	D	D	D	D	D	Sends five bits

D=DATA BIT

DTR (D_7)

Data Terminal Ready is the control bit for the $\overline{\text{DTR}}$ pin. When set, $\overline{\text{DTR}}$ is active (low). When reset (0) $\overline{\text{DTR}}$ is inactive (high).

WRITE REGISTER 6

This register contains the first 8 bits of a BiSync sequence. It must be programmed with the check address (if used) in SDLC mode, and must contain the sync character in the 8-bit sync mode. It is not used in the external sync mode.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SYN7	SYN6	SYN5	SYN4	SYN3	SYN2	SYN1	SYN0	MONO OR BI SYNC MODE
AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	SDLC MODE

WRITE REGISTER 7

This register contains the second byte of a 16-bit synchronization sequence, or the 8-bit sync character. For SDLC mode, it must be programmed to 01111110 . It is not used in the external sync mode.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
SYN15	SYN14	SYN13	SYN12	SYN11	SYN10	SYN9	SYN8	BI SYNC MODE
0	1	1	1	1	1	1	0	SDLC MODE

READ REGISTER 0

This is the register read if the register pointers are (000).

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Break/ Abort	Tx UNDERRUN/ EOM	CTS	Sync/ Hunt	DCD	Transmit Buffer Empty	Interrupt Pending	Receive Character Available

RECEIVE CHARACTER AVAILABLE (D_0)

This bit is set when at least one character is available in the receive buffers.

INTERRUPT PENDING (D_1) (Channel A only)

Any interrupt condition present in the entire SIO will cause this bit to be set, but it is present only in Channel A and is always 0 in Channel B.

TRANSMIT BUFFER EMPTY (D_2)

The Transmit Buffer Empty bit is set whenever the transmit buffer is empty, except when a CRC character is being sent in a synchronous mode.

DCD (D_3)

Shows the state of the $\overline{\text{DCD}}$ pin at the time of the last change of any of the five External/ Status bits. (DCD, CTS, SYNC/HUNT, BREAK/ABORT or Tx UNDERRUN/EOM.) To get the current state of the DCD pin, this bit must be read immediately following a RESET EXTERNAL/STATUS INTERRUPT command. (Command 2.)

SYNC/HUNT (D_4)

In asynchronous modes, this bit is similar to the $\overline{\text{DCD}}$ and the $\overline{\text{CTS}}$ bits, except that it shows the state of the $\overline{\text{SYNC}}$ pin. In synchronous modes, this bit is reset when character synchronization is achieved and is set by writing the ENTER HUNT MODE bit. Unlike the external pin, the bit remains reset until set by the ENTER HUNT MODE bit.

CTS (D_5)

This bit is similar to the $\overline{\text{DCD}}$ bit, except that it shows the state of the $\overline{\text{CTS}}$ pin.

BREAK/ABORT (D_7)

In asynchronous modes, this bit is set when a "break" is detected. After the inputs have been re-enabled (by the RESET EXTERNAL/STATUS INTERRUPTS command, Command 2), the bit will be reset when the break stops. If EXTERNAL STATUS interrupts are enabled, these changes of state cause interrupts. In SDLC mode, this bit is set by the detection of an abort sequence (7 or more 1's). It is not used in other synchronous modes.

TRANSMIT UNDERRUN/END OF MESSAGE (EOM)

In synchronous modes, CRC is automatically sent when the transmitter is empty for the first time in a message. Interrupts are generated (if enabled) when this bit is set, but not when reset. If this bit is set and the TRANSMIT BUFFER EMPTY bit is not set, then the CRC character is being sent. TRANSMIT BUFFER EMPTY and Tx UNDERRUN/EOM both set imply that SYNC characters are being sent.

READ REGISTER 1

This register is read when the register pointers are (001). The pointers automatically reset to (000) after a read from this register.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
End Of Frame (SDLC)	CRC/ Framing Error	Receiver Overrun Error	Parity Error	Residue Code 2	Residue Code 1	Residue Code 0	All Sent

ALL SENT (D_0)

In asynchronous modes, this bit is set when all characters have completely cleared the transmitter. Transitions of this bit do not cause interrupts. It is always set in synchronous modes.

RESIDUE CODE 0 (D_1) – RESIDUE CODE 2 (D_3)

These three bits indicate the length of the I-field in the SDLC mode in those cases where the I-field is not an integral multiple of the character length used. Only on the transfer on which the END OF FRAME (SDLC) bit is set do these codes have meaning.

For a receiver setting of eight bits per character, the codes signify the following:

Residue Code 2	Residue Code 1	Residue Code 0	I-Field In Previous Byte	I-Field In Second Previous Byte
1	0	0	0	3
0	1	0	0	4
1	1	0	0	5
0	0	1	0	6
1	0	1	0	7
0	1	1	0	8
1	1	1	1	8
0	0	0	2	8

I-field bits are right-justified in all cases.

If a receive character length different from eight bits is used for the I-field, a table similar to the above may be constructed for each different character length. For no residue, i.e., the last character boundary coincides with the boundary of the I-Field and CRC Field, the Residue Code will always be:

Residue Code 2	Residue Code 1	Residue Code 0
0	1	1

PARITY ERROR (D_4)

When parity is enabled, this bit is set for those characters whose parity does not match the sense programmed. The bit is latched so that once an error occurs, the bit remains set until the ERROR RESET COMMAND, Command 6, is given.

RECEIVER OVERRUN ERROR (D_5)

This indicates that more than four characters have been received without a read from the CPU. Only the character that has been written over is flagged with this error, but when this character is read, the error condition is latched until reset by the ERROR RESET COMMAND, Command 6. If STATUS AFFECTS VECTOR bit is enabled, the character that has been overrun will interrupt with the SPECIAL RECEIVE CONDITION vector.

CRC/FRAMING ERROR (D_6)

If a framing error occurs (in asynchronous modes), this bit is set (and not latched) only for the character on which it occurred. Detection of a framing error adds an additional ½ bit time to the character time so that the framing error will not also be interpreted as a new start bit. In synchronous modes, this bit indicates the result of comparing the CRC checker to the appropriate check value.

END OF FRAME (SDLC) (D_7)

In SDLC mode, this bit indicates that a valid ending flag has been received and that the CRC error and residue codes are valid.

READ REGISTER 2 (Channel B Only)

This register contains the interrupt vector as written into Write Register 2 if the STATUS AFFECTS VECTOR control bit is not set. If that control bit is set, it contains the interrupt vector as it would be returned were an interrupt from the SIO to be processed exactly at the time of the read. If no interrupts are pending, $V_3 = 0$, $V_2 = 1$, $V_1 = 1$ and other bits are as programmed. The register may be read only through Channel B.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
V_7	V_6	V_5	V_4	V_3 *	V_2 *	V_1 *	V_0

*V_1, V_2, and V_3 are varible if STATUS AFFECTS VECTOR mode is enabled

4.5 Z80-SIO COMMAND STRUCTURE

Reg.	Control			DATA BITS							
#	C/$\overline{\text{D}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	CRC 1	CRC 0	CMD 2	CMD 1	CMD 0	0	0	0
	1	1	0	CRC 1	CRC 0	CMD 2	CMD 1	CMD 0	0	0	0
	1	0	1	Break/Abort	Tx UNDERRUN/ EOM	CTS	SYNC/HUNT	DCD	TxBuffer EMPTY	INT Pending (CH A Only)	RxChar Avail
1	1	1	0	CRC 1	CRC 0	CMD 2	CMD 1	CMD 0	0	0	1
	1	1	0	Wait/RDY EN	WaitFN/RDYFN	Wait/RDYonR/T	RxINTMode 1	RxINTmode 0	Status Effects V (CH B Only)	TxINT EN	EXT INT EN
	1	0	1	End ofFrame SDLC	CRC FrameError	RxOVRN Error	Parity Error	Res. Code 2	Res. Code 1	Res. Code 0	All Sent

4.5 Z80-SIO COMMAND STRUCTURE (Cont'd.)

CHB-ONLY

2	1	1	0	CRC 1	CRC 0	CMD 2	CMD 1	CMD 0	0	1	0
	1	1	0	V7	V6	V5	V4	V3	V2	V1	V0
	1	0	1	V7	V6	V5	V4	V3	V2	V1	V0

3	1	1	0	CRC 1	CRC 0	CMD 2	CMD 1	CMD 0	0	1	1
	1	1	0	RxBits/Char 1	RxBits/Char 0	Auto Enables	EnterHuntMode	RxCRC EN	AddrssSearchMd	SyncChar LD INH	RxEN

4	1	1	0	CRC 1	CRC 0	CMD 2	CMD 1	CMD 0	1	0	0
	1	1	0	Clock Rate 1	Clock Rate 0	Sync Mode 1	Sync Mode 0	Stop Bits 1	Stop Bits 0	Parity Even/$\overline{\text{Odd}}$	Parity

5	1	1	0	CRC 1	CRC 0	CMD 2	CMD 1	CMD 0	1	0	1
	1	1	0	DTR	TxBits/Char 1	TxBits/Char 0	Send BREAK	TxEN	$\overline{\text{SDLC}}$/CRC 16	RTS	TxCRC EN

6	1	1	0	CRC 1	CRC 0	CMD 2	CMD 1	CMD 0	1	1	0
	1	1	0	SYNC/SDLC 7	SYNC/SDLC 6	SYNC/SDLC 5	SYNC/SDLC 4	SYNC/SDLC 3	SYNC/SDLC 2	SYNC/SDLC 1	SYNC/SDLC 0

7	1	1	0	CRC 1	CRC 0	CMD 2	CMD 1	CMD 0	1	1	1
	1	1	0	SYNC/SDLC 15	SYNC/SDLC 14	SYNC/SDLC 13	SYNC/SDLC 12	SYNC/SDLC 11	SYNC/SDLC 10	SYNC/SDLC 9	SYNC/SDLC8

4.6 PROGRAMMING EXAMPLE

A typical start-up routine following an internal or external reset, would be as follows:

B/$\overline{A}$	C/$\overline{D}$	$\overline{RD}$	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	COMMENTS
1	1	1	0	0	0	0	0	0	1	0	Pointer set to Register 2B
1	1	1	V_7	V_6	V_5	V_4	V_3	V_2	V_1	V_0	Interrupt Vector loaded
1	1	1	0	0	0	0	0	1	0	0	Pointer set to Write Register 4B
1	1	1	0	1	X	X	0	1	1	1	Even parity, 1 stop bit, X16 clock asynchronous mode selected
1	1	1	0	0	0	0	0	1	0	1	Pointer set to Write Register 5B
1	1	1	0	0	1	0	1	0	1	0	7 bits/transmit character, transmitter
1	1	1	0	0	0	0	0	0	1	1	Pointer set to Write Register 3B
1	1	1	0	1	1	0	0	0	0	1	7 bits/receive character, DCD and CTS enable Receiver and Transmitter, Receiver enabled
1	1	1	0	0	0	0	0	0	0	1	Pointer set to Register 1B
1	1	1	0	0	0	1	0	1	1	1	Interrupt on every character, status affects Vector external/status interrupts enabled

Channel B is now setup to send and receive asynchronous data.

Setup for Channel A follows:

0	1	1	0	0	0	0	0	1	0	0	Pointer set to Write Register 4A
0	1	1	0	0	1	0	0	0	0	0	SDLC mode and XI clock selected, no parity

Programming Example

B/$\overline{A}$	C/$\overline{D}$	$\overline{RD}$	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	COMMENTS
0	1	1	0	1	0	0	0	1	1	0	Pointer set to Write Register 6A, Reset Receive CRC Checker
0	1	1	AD_7	AD_6	AD_5	AD_4	AD_3	AD_2	AD_1	AD_0	SDLC message address entered
0	1	1	1	0	0	0	0	1	1	1	Pointer set to Write Register 7A, Reset mit CRC generator
0	1	1	0	1	1	1	1	1	1	0	SDLC Flag entered
0	1	1	0	0	0	0	0	0	0	1	Pointer set to Register 1A
0	1	1	0	0	0	1	0	1	1	1	Interrupt every character, status affects vector, external/status interrupts enabled
0	1	1	0	0	0	1	0	1	0	1	Pointer set, to Write Register 5A, Reset External/Status Interrupts
0	1	1	1	1	1	0	1	0	0	0	SDLC CRC Code selected, 8 bits/transmit character, CRC and transmitter enabled
0	1	1	0	0	0	0	0	0	1	1	Pointer set to Write Register 3A
0	1	1	1	1	1	0	1	1	0	1	8 bits/receive character, DCD and CTS enable receiver and transmitter, receiver is enabled, SIO searches for programmed address.

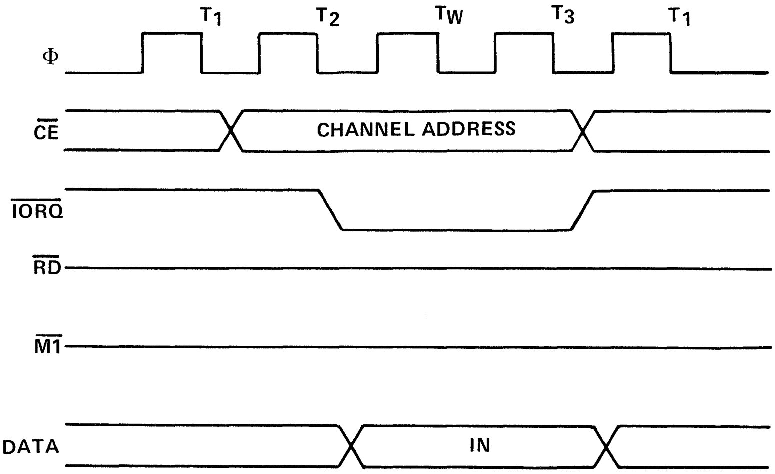
Channel A is now programmed for SDLC transfers.

0	0	1	D	D	D	D	D	D	D	D	Address byte to be sent by Ch. A
0	0	1	D	D	D	D	D	D	D	D	Address or control byte to be sent by Ch. A
0	1	1	1	1	1	0	0	0	0	0	Reset Tx UNDERRUN/EOM pointer to register 0, so CRC can be automatically sent at end of message.

5.0 TIMING WAVEFORMS

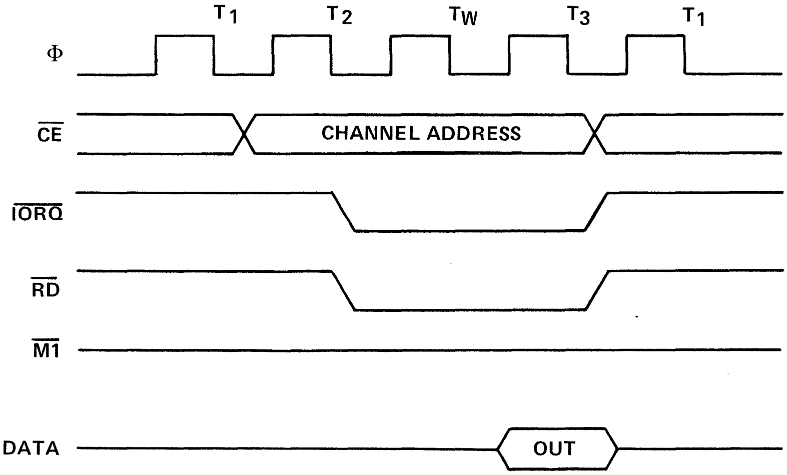
WRITE CYCLE

Illustrated here is the timing associated with a data or control byte being written into the SIO. Z80 output instructions satisfy this timing.



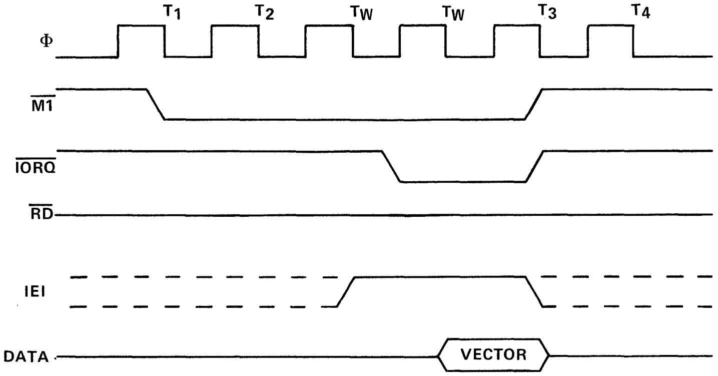
READ CYCLE

The timing associated with reading data or a status register within the SIO is illustrated here. Z80 Input instructions satisfy this timing.



INTERRUPT ACKNOWLEDGE CYCLE

Sometime after an interrupt is requested by the SIO, the CPU will send out an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$.) During this time, the interrupt logic of the SIO will determine the highest priority function which is requesting an interrupt. To insure that the daisy chain enable lines stabilize, channels are inhibited from changing their interrupt request status when $\overline{M1}$ is active (low). If the SIO is the highest priority device requesting an interrupt, the SIO will place the appropriate interrupt vector on the data bus when $\overline{IORQ}$ goes active.

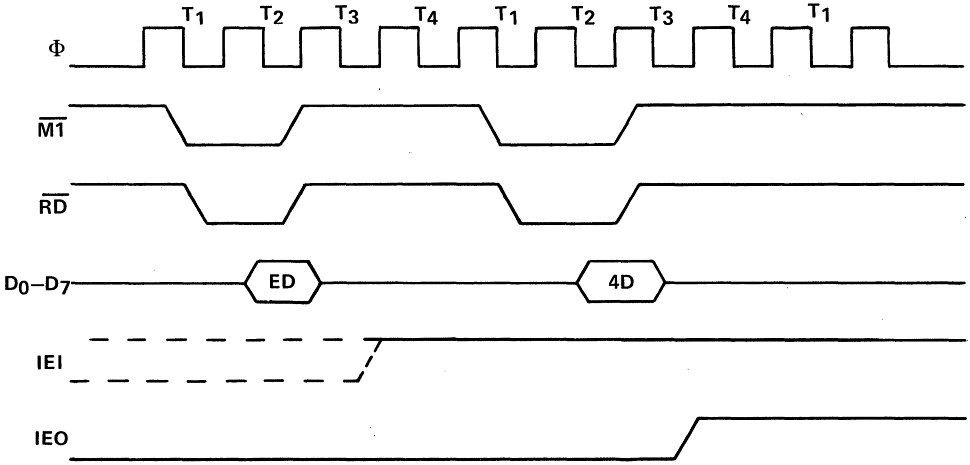


RETURN FROM INTERRUPT CYCLE

If a Z80 peripheral device has no interrupt pending and is not under service, then its IEO= IEI. If it has an interrupt under service (i.e. it has already interrupted and received an interrupt acknowledge) then its IEO is always low, inhibiting lower priority chips from interrupting. If it has an interrupt pending which has not yet been acknowledged, IEO will be low unless an "ED" is decoded as the first byte of a two byte opcode. In this case, IEO will go high until the next opcode byte is decoded, whereupon it will again go low. If the second byte of the opcode was a "4D" then the opcode was an RETI instruction.

After an "ED" opcode is decoded, only the peripheral device which has interrupted and is currently under service will have its IEI high and its IEO low. This device is the highest priority device in the daisy chain which has received an interrupt acknowledge. All other peripherals have IEI=IEO. If the next opcode byte decoded is "4D", this peripheral device will reset its " interrupt under service" condition.

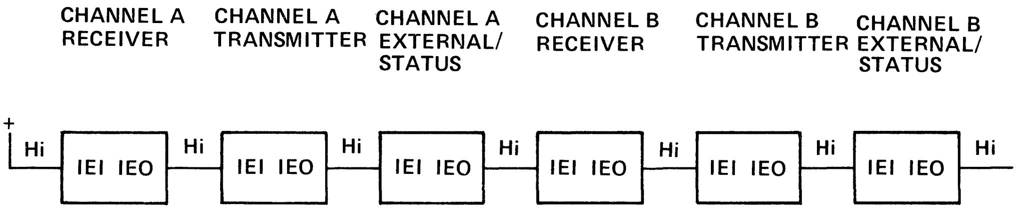
Wait cycles are allowed in the $\overline{M1}$ cycles.



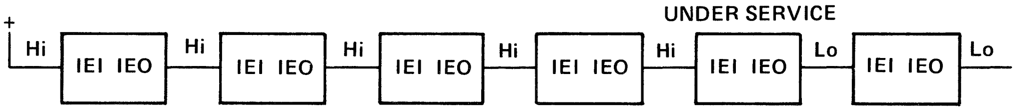
6.0 DAISY CHAIN INTERRUPT SERVICING

The following illustration is a typical nested interrupt sequence which may occur in the SIO. In a system with several peripheral chips, the other chips may be included in the daisy chain with either higher or lower priority than the SIO channels.

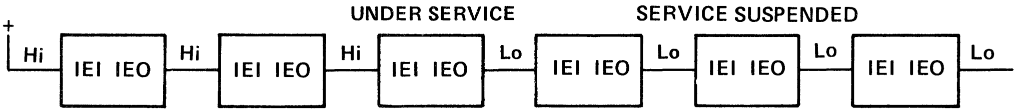
In this sequence, the transmitter of Channel B interrupts and is granted service. While it is being serviced, an external/status interrupt from Channel A occurs and is granted service. The service routine for the Channel A interrupt is completed and either the RETI instruction is executed or the RETI command is written into the SIO to indicated to Channel A that the external/status interrupt routine is complete. At this time, the service routine for the Channel B transmitter is resumed. When this routine is completed, another RETI instruction is executed to complete the service.



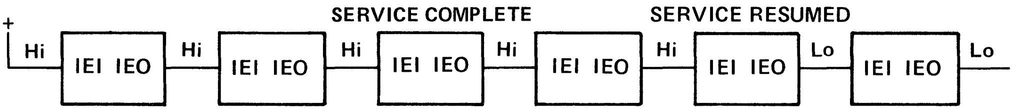
1. Priority Interrupt Daisy Chain before any interrupt occurs.



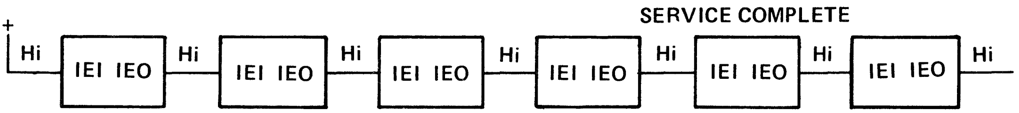
2. Channel B's transmitter interrupts and is acknowledged.



3. External/Status of Channel A interrupts suspending service of Channel B transmitter



4. Channel A External/Status routine complete. RETI issued, Channel B transmitter service resumed.



5. Channel B transmitter's service routine complete, second RETI issued.

Z80
Family

7.0 ABSOLUTE MAXIMUM RATINGS*

PRELIMINARY

Voltage on any pin relative to GND . . . . −0.3V to +7V
Operating Temperature (Ambient) T_A . . . . 0°C to 70°C
Storage Temperature - Ceramic (Ambient) . . . . −65°C to +150°C
Storage Temperature — Plastic (Ambient). . . . . −55°C to +125°C
Power Dissipation 1.5W

***Comment**

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

7.1 D.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = 5V ± 5% unless otherwise specified.

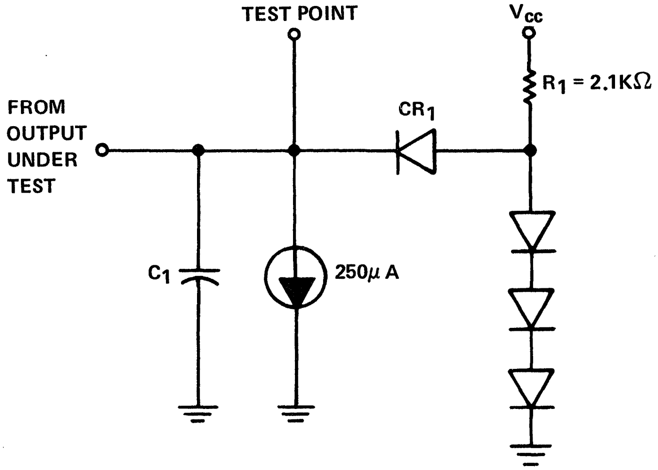
Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		.40	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.2		V_{CC}	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 1.8 mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = 250μA
V_{CC}	Power Supply Current			140	mA	t_c = 400 nsec
I_{LI}	Input Leakage Current			10	μA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{CC}$

7.2 CAPACITANCE

T_A = 25°C, f = 1 MHz

Symbol	Parameter	Max.	Unit	Test Condition
C_Φ	Clock Capacitance	35	pF	Unmeasured Pins
C_{IN}	Input Capacitance	5	pF	Returned to Ground
C_{OUT}	Output Capacitance	10	pF	

7.3 LOAD CIRCUIT FOR OUTPUT



Z80 Family

A. C. TIMING DIAGRAM

Figure 7.4

PRELIMINARY

Timing measurements are made at the following voltages, unless otherwise specified:
Only for timing measurements

CLOCK	4.2V		.8V
OUTPUT	2.0V		.8V
INPUT	2.0V		.8V
FLOAT	ΔV	=	±0.5V

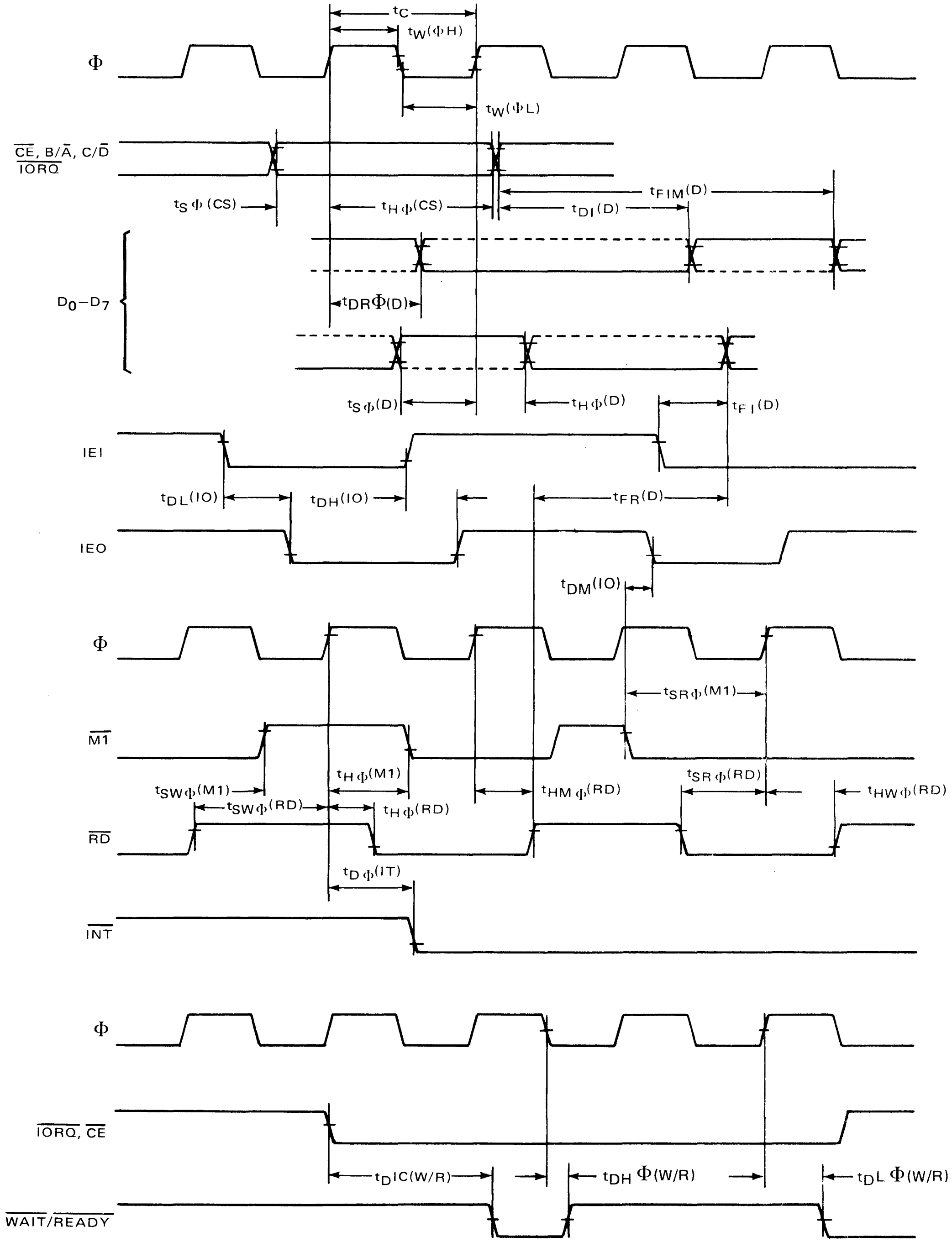
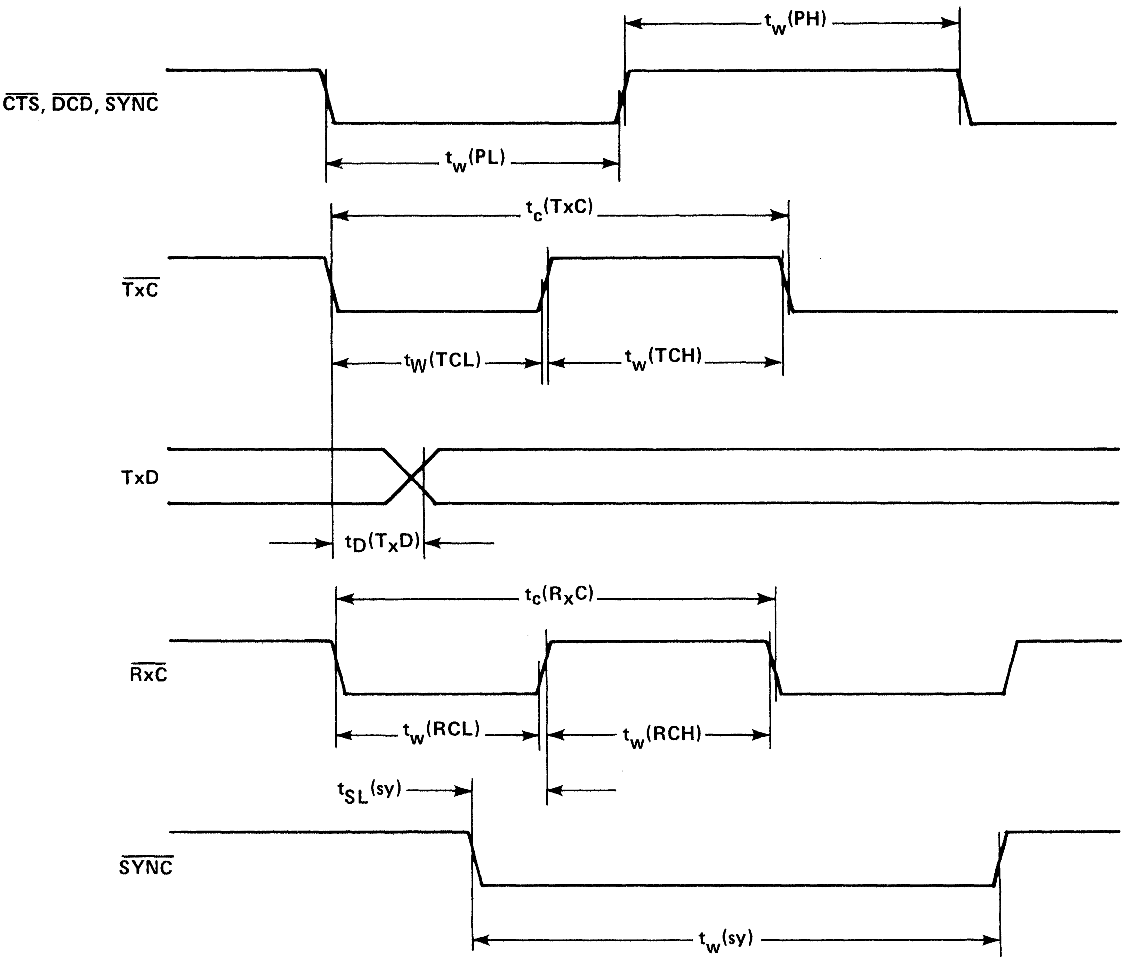


Figure 7.4


t_w(PH)
$\overline{CTS}$, $\overline{DCD}$, $\overline{SYNC}$
t_w(PL)
t_c(TxC)
$\overline{TxC}$
t_W(TCL)
t_w(TCH)
TxD
$t_D(T_xD)$
$t_c(R_xC)$
$\overline{RxC}$
t_w(RCL)
t_w(RCH)
t_{SL}(sy)
$\overline{SYNC}$
t_w(sy)

A. C. CHARACTERISTICS

PRELIMINARY

Figure 7.5

T_a = 0°C to 70 T_A = 0°C to 70°C, V_{CC} = +5V ±5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	$t_c(\Phi)$	Clock Period	400		nsec	
	$t_W(\Phi H\ )$	Clock Pulse Width, Clock High	170	2000	nsec	
	$t_W(\Phi L\ )$	Clock Pulse Width, Clock Low	170	2000	nsec	
	t_r,t_f	Clock Rise and Fall Times	-0-	30	nsec	
$\overline{CE}$,B/$\overline{A}$	$t_H\ \Phi(CS)$	Control signal hold time from Rising Edge of Φ	-0-		nsec	NOTE 1
C/$\overline{D}$, $\overline{IORQ}$	$t_S\ \Phi(CS)$	Control Signal setup time from Rising Edge of Φ	160		nsec	
	$t_{DR}\ \Phi(D)$	Data Output Delay from Rising Edge of Φ during Read Cycle		480	nsec	
	$t_s\Phi(D)$	Data Setup Time to Rising Edge of Φduring Write Cycle or M1 Cycle	50		nsec	
	$t_H\ \Phi(D)$	Data Hold Time from Rising Edge of Φ during Write Cycle or M1 Cycle	-0-		nsec	
D_0-D_7	$t_{DI(D)}$	Data Output Delay from Falling Edge of IORQ during INTA Cycle		340	nsec	
	$t_{FIM(D)}$	Delay to Floating Bus from Rising Edge of IORQ during INTA Cycle		230	nsec	
	$t_{FR(D)}$	Delay to Floating Bus from Rising Edge of RD during Read Cycle		230	nsec	
	$t_{FI(D)}$	Delay to Floating Bus from Falling Edge of IEI during INTA Cycle		230	nsec	
IEO	$t_{DL(IO)}$	IEO Delay Time from Falling Edge of IEI		200	nsec	
	$t_{DH(IO)}$	IEO Delay Time from Rising Edge of IEI		200	nsec	
	t_D M (I[illegible])	IEO Delay Time from Falling Edge of M1 (when interrupt occurs just prior to M1)		300	nsec	
	$t_{SW}\ \Phi(M1)$	M1 Setup Time to Rising Edge of Φ during Read or Write Cycle	210		nsec	
$\overline{M1}$	$t_{SR}\ \Phi(M1)$	M1 Setup Time to Rising Edge of Φ during INTA or M1 Cycle	210		nsec	
	$t_H\ \Phi(M1)$	M1 Hold Time From Rising Edge of Φ	-0-		nsec	
$\overline{RD}$	$t_{SW}\Phi(RD)$	RD Setup Time to Rising Edge of Φduring Write or INTA Cycle	240		nsec	
	$t_H\Phi(RD)$	RD Hold Time from Rising Edge of Φ during INTA Cycle	-0-		nsec	
	$t_{SR}\Phi(RD)$	RD Setup Time to Rising Edge of Φ during Read or M1 Cycle	240		nsec	
	$t_{HW}\Phi(RD)$	RD Hold Time from Rising Edge of Φ during Write Cycle	-0-		nsec	
	$t_{HM}\Phi(RD)$	RD Hold Time from Rising Edge of Φ during M1 Cycle	-0-		nsec	
$\overline{INT}$	$t_{DRx(IT)}$	INT Delay Time from center of Receive Data Bit	10	13	ΦPeriods	
	$t_{DTx(IT)}$	INT Delay Time from center of Transmit Data Bit	5	9	Φ Periods	
	$t_{D\Phi(IT)}$	INT Delay Time from Rising Edge of Φ		200	nsec	
$\overline{WAIT}$/	$t_DIC(W/R)$	WAIT/READY Delay Time from IORQ or CE in WAIT Mode		300	nsec	
$\overline{READY}$	$t_DH\Phi(W/R)$	WAIT/READY Delay Time from Falling Edge of Φ, WAIT/READY (high) WAIT Mode		210	nsec	
	$t_DRx(W/R)$	WAIT/READY Delay Time from center of Receive Data Bit, Ready Mode	10	13	Φ Periods	
	$t_DTx(W/R)$	WAIT/READY Delay Time from Center of Transmit Data Bit, Ready Mode	5	9	Φ Periods	
	$t_DL\Phi(W/R)$	WAIT/READY Delay from Rising Edge of Φ, WAIT/READY, (Low) Ready Mode		120	nsec	
$\overline{CTSA}$, $\overline{CTSB}$ $\overline{DCDA}$, $\overline{DCDB}$, $\overline{SYNCA}$, $\overline{SYNCB}$	$t_W(PH)$	Minimum High Pulse Width for latching states into register and generating interrupt	200		nsec	
	$t_W(PL)$	Minimum Low Pulse Width for latching state into register and generating interrupt	200		nsec	
$\overline{SYNCA}$,$\overline{SYNCB}$	$t_{DL}(SY)$	Sync Pulse Delay Time from Center of Receive Data Bit, Output	4	7	Φ Periods	
	$t_{SL(SY)}$	Sync Pulse Setup Time to Rising Edge of RxC, External Sync	100		nsec	
	$t_{W(SY)}$	Sync Pulse Width to Start Character Assembly	1		$\overline{RxC}$ Period	
	$t_c(TxC)$	Transmit Clock Period	400	∞	nsec	
$\overline{TxCA}$,$\overline{TxCB}$	$t_W(TCH)$	Transmit Clock Pulse Width, Clock High	180	∞	nsec	NOTE 2
	$t_W(TCL)$	Transmit Clock Pulse Width, Clock Low	180	∞	nsec	
TxDA,TxDB	$t_D(TxD)$	TxD Output Delay from Falling Edge of $\overline{TxC}$ (1x Clock Mode)		400	nsec	
$\overline{RxCA}$,$\overline{RxCB}$	$t_c(RxC)$	Receive Clock Period	400	∞	nsec	
	$t_W(RCH)$	Receive Clock Pulse Width, Clock High	180	∞	nsec	NOTE 3
	$t_W(RCL)$	Receive Clock Pulse Width, Clock Low	180	∞	nsec	

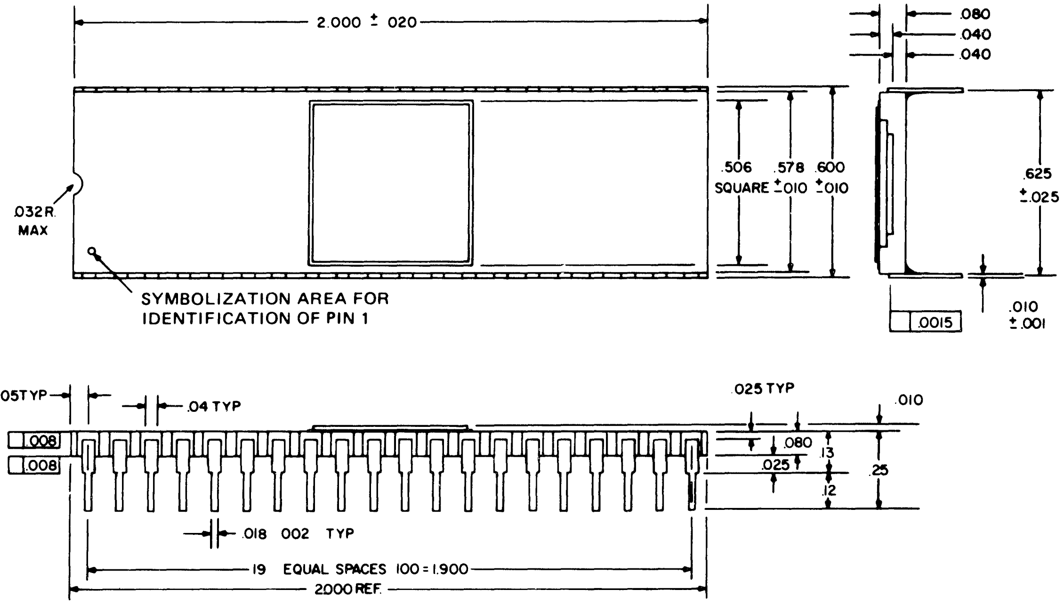
NOTE 1: If WAIT is to be used. $\overline{CE}$, $\overline{IORQ}$, C/$\overline{D}$ and $\overline{M1}$ must be valid for as long as WAIT condition is to persist.

NOTE 2: In all modes, maximum data rate must be less than $\frac{1}{5.0}$ of system clock (Φrate.

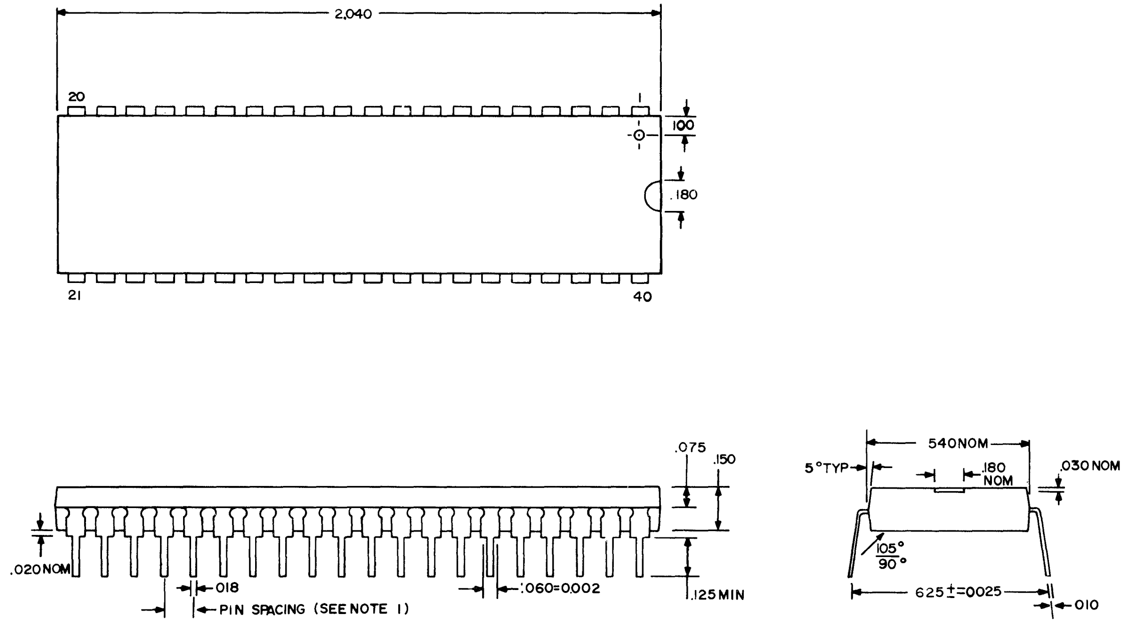
NOTE 3: The RESET signal must be active a minimum of one complete Φ cycle.

8.0 PACKAGE CONFIGURATION

PACKAGE DESCRIPTION – 40 Pin Dual-In-Line Ceramic Package



PACKAGE DESCRIPTION – 40-Pin Dual-In-Line Plac Plastic Package



NOTES:

1. The true-position pin spacing is 0.100 between centerlines. Each pin centerline is located within ± 0.010 of its true longitudinal position relative to pins 1 and 40.

9.0 ORDERING INFORMATION

PART NO.		PACKAGE TYPE	MAX CLOCK FREQUENCY	TEMPERATURE RANGE
MK3884N	Z80 - SIO/0	Plastic	2.5MHz	0°C to + 70°C
MK3884P	Z80 - SIO/0	Ceramic	2.5MHz	0°C to + 70°C
MK3884N-10	Z80 - SIO/0	Plastic	2.5MHz	-40°C to +85°C
MK3884P-10	Z80 - SIO/0	Ceramic	2.5MHz	-40°C to +85°C
MK3884N-4	Z80A - SIO/0	Plastic	4MHz	0°C to +70°C
MK3884P-4	Z80A - SIO/0	Ceramic	4MHz	0°C to +70°C
MK3885N	Z80 - SIO/1	Plastic	2.5MHz	0°C to +70°C
MK3885P	Z80 - SIO/1	Ceramic	2.5MHz	0°C to +70°C
MK3885N-10	Z80 - SIO/1	Plastic	2.5MHz	-40°C to +85°C
MK3885P-10	Z80 - SIO/1	Ceramic	2.5MHz	-40°C to +85°C
MK3885N-4	Z80A - SIO/1	Plastic	4MHz	0° C to +70°C
MK3885P-4	Z80A - SIO/1	Ceramic	4MHz	0°C to +70°C
MK3887N	Z80-SIO/2	Plastic	2.5MHz	0°C to +70°C
MK3887P	Z80-SIO/2	Ceramic	2.5MHz	0°C to +70°C
MK3887N-10	Z80-SIO/2	Plastic	2.5MHz	-40°C to +85°C
MK3887P-10	Z80-SIO/2	Ceramic	2.5MHz	-40°C to +85°C
MK3887N-4	Z80-SIO/2	Plastic	4MHz	0°C to +70°C
MK3887P-4	Z80-SIO/2	Ceramic	4MHz	0°C to +70°C

NOTE: See Section 2.2 for explanation of the differences between the MK3884, MK3885, and MK3887.

PRELIMINARY

MOSTEK®

MK3886 Z80 COMBO CHIP

Product Brief

FEATURES

- 256 x 8 Static RAM

  Low power standby mode for 64 bytes

- Serial I/O Port

  One 16 bit shift register available to the CPU as two 8 bit ports

  Data is shifted into or out of the register in serial form

  One of the Programmable Timers can be used as the shift clock

  Synchronous or Asynchronous operation with programmable end of word interrupt

- Two Programmable Timers
- Four External Interrupt Channels with a Programmable Vector for each Channel
- Z80 Compatible Daisy Chain Interrupt Structure
- Compatible with 6800 and 8080 CPU's
- Single +5 Volt Supply
- 40 Pin DIP

Z80 Family

MOSTEK®

Z80 INTERFACING TECHNIQUES FOR DYNAMIC RAM

By JERRY WINFIELD

Application Note

INTRODUCTION

Since the introduction of second generation microprocessors, there has been a steady increase in the need for larger RAM memory for microcomputer systems. This need for larger RAM memory is due in part to the availability of higher level languages such as PL/M, PL/Z, FORTRAN, BASIC and COBOL. Until now, when faced with the need to add memory to a microcomputer system, most designers have chosen static memories such as the 2102 1Kx1 or possibly one of the new 4Kx1 static memories. However, as most mini or mainframe memory designers have learned, 16-pin dynamic memories are often the best overall choice for reliability, low power, performance, and board density. This same philosophy is true for a microcomputer system. Why then have microcomputer designers been reluctant to use dynamic memory in their system? The most important reason is that second generation microprocessors such as the 8080 and 6800 do not provide the necessary signals to easily interface dynamic memories into a microcomputer system.

Today, with the introduction of the Z80, a true third generation microprocessor, not only can a microcomputer designer increase system throughput by the use of more powerful instructions, but he can also easily interface either static or dynamic memories into the microcomputer system. This application note provides specific examples of how to interface 16-pin dynamic memories to the Z80.

OPERATION OF 16-PIN DYNAMIC MEMORIES

The 16-pin dynamic memory concept, pioneered by MOSTEK, uses a unique address multiplexing technique which allows memories as large as 16, 384 bits x 1 to be packaged in a 16-pin package. For example the MK4027 (4,096x1 dynamic MOS RAM) and the MK4116 (16,384x1 dynamic MOS RAM) both use address multiplexing to load the address bits into memory. The MK4027 needs 12 address bits to select 1 out of 4,096 locations, while the MK4116 requires 14 bits to select 1 out of 16,384. The internal memories of the MK4027 and MK4116 can be thought of as a matrix. The MK4027 matrix can be thought of as 64x64, and the MK4116 as 128x128. To select a particular location, a row and column address is supplied to the memory. For the MK4027, address bits A_0-A_5 are the row address, and bits A_6-A_{11} are the column addresses. For the MK4116, address bits A_0-A_6 are the row address, and A_7-A_{13} are the column address. The row and column addresses are strobed into the memory by two negative going clocks called Row Address Strobe ($\overline{RAS}$) and Column Address Strobe ($\overline{CAS}$). By the use of $\overline{RAS}$ and $\overline{CAS}$, the address bits are latched into the memory for access to the desired memory location.

Dynamic memories store their data in the form of a charge on a small capacitor. In order for the dynamic memory to retain valid data, this charge must be periodically restored. The process by which data is restored in a dynamic memory is known as refreshing. A refresh cycle is performed on a row of data each time a read or write cycle is performed on any bit within the given row. A row consists of 64 locations for the MK4027 and 128 locations for the MK4116. The refresh period for the MK4027 and the MK4116 is 2ms which means that the memory will retain a row of data for 2ms without a refresh. Therefore, to refresh all rows within 2ms, a refresh cycle must be executed every 32μs (2ms÷64) for the MK4027 and 16μs (2ms÷128) for the MK4116.

To ensure that every row within a given memory is refreshed within the specified time, a refresh row address counter must be implemented either in external hardware or as an internal CPU function as in the Z80. (Discussed in more detail under Z80 Refresh Control and Timing.) The refresh row address counter should be incremented each time that a refresh cycle is executed. When a refresh is performed, all RAMs in the system should be loaded with the refresh row address. For the MK4027 and the MK4116, a refresh cycle consists of loading the refresh row address on the address lines and then generating a $\overline{RAS}$ for all RAMs in the system. This is known as a $\overline{RAS}$ only refresh. The row that was addressed will be refreshed in each memory. The $\overline{RAS}$ only refresh prevents a conflict between the outputs of all the RAMs by disabling the output on the MK4116, and maintaining the output state from the previous memory cycle on the MK4027.

Z80 TIMING AND MEMORY CONTROL SIGNALS

The Z80 was designed to make the job of interfacing

Z80 Family

to dynamic memories easier. One of the reasons the Z80 makes dynamic memory interfacing easier is because of the number of memory control signals that are available to the designer. The Z80 control signals associated with memory operations are:

MEMORY REQUEST ($\overline{\text{MREQ}}$) - Memory request signal indicating that the address bus holds a valid memory address for a memory read, memory write, or memory refresh cycle.

READ ($\overline{\text{RD}}$) - Read signal indicating that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

WRITE ($\overline{\text{WR}}$) - Write signal indicating that the CPU data bus hold valid data to be stored in the addressed memory or I/O device.

REFRESH ($\overline{\text{RFSH}}$) - Refresh signal indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current MREQ signal should be used to generate a refresh cycle for all dynamic memories in the system.

Figures 1a, 1b, and 1c show the timing relationships of the control signals, address bus, data bus and system clock Φ. By using these timing diagrams, a set of equations can be derived to show the worst case access times needed for dynamic memories with the Z80 operating at 2.5MHz.

The access time needed for the op code fetch cycle and the memory read cycle can be computed by equations 1 and 2.

(1) $t_{\text{ACCESS OP CODE}} = 3(t_c/2) - t_{DL\overline{\Phi}(MR)} - t_{S\Phi(D)}$

where: t_c = Clock period

$t_{DL\overline{\Phi}(MR)}$ = $\overline{\text{MREQ}}$ delay from falling edge of clock.

$t_{s\Phi(D)}$ = Data setup time to rising edge of clock during op code fetch cycle.

let: t_C = 400ns; $t_{DL\overline{\Phi}(MR)}$ = 100ns; $t_{s\Phi}$ = 50ns

then: $\underline{t_{\text{ACCESS OP CODE}} = 450\text{ns}}$

(2) $t_{\text{ACCESS MEMORY READ}} = 4(t_c/2) - t_{DL\overline{\Phi}(MR)} - t_{S\overline{\Phi}(D)}$

where: t_C = Clock period

$t_{DL\overline{\Phi}(MR)}$ = $\overline{\text{MREQ}}$ delay from falling edge of clock

$t_{S\overline{\Phi}(D)}$ = Data Setup time to falling edge of clock

let: t_C = 400ns; $t_{DL\ (MR)}$ = 100ns; $t_{S\ (D)}\ \overline{\Phi}$ = 60ns

then: $\underline{t_{\text{ACCESS MEMORY READ}} = 640\text{ns}}$

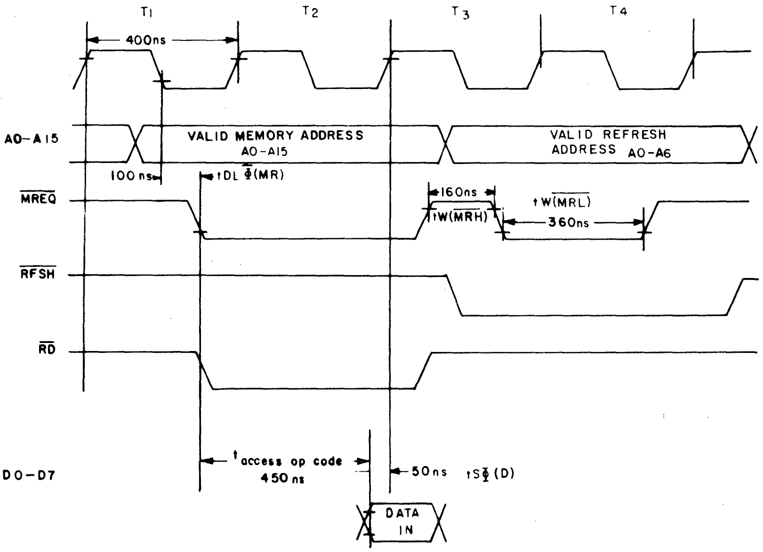
The access times computed in equations 1 and 2 are overall worst case access times required by the CPU. The overall access times must include all TTL buffer delays and the access time for the memory device. For example, a typical dynamic memory design would have the following characteristics, (see Figure 2).

The example in Figure 2 shows an overall access time of 336ns. This would more than satisfy the 450ns required for the op code fetch and the 640ns required for a memory read.

CPU $\overline{\text{MREQ}}$ buffer delay.	12ns (8T97)
Memory gating and timing delays	40ns
Memory device access time	250ns (MK4027/4116-4)
Memory data bus buffer delay	17ns (8T28)
CPU data bus buffer delay.	17ns (8T28)
	336ns

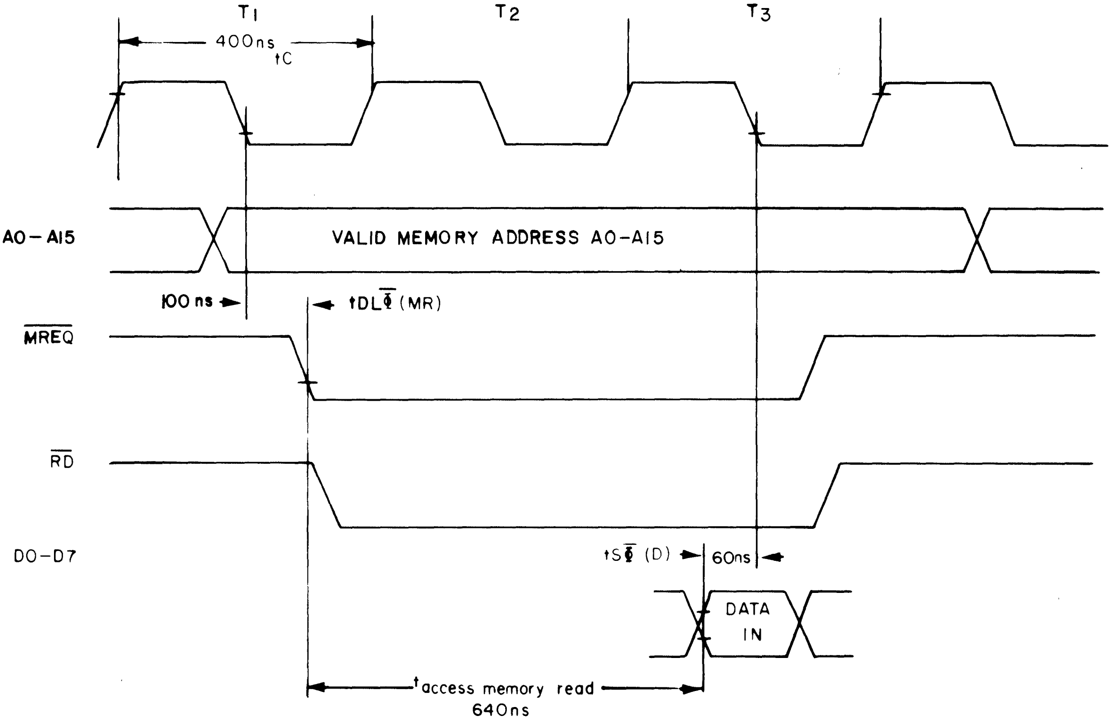
OP CODE FETCH TIMING

Figure 1a.



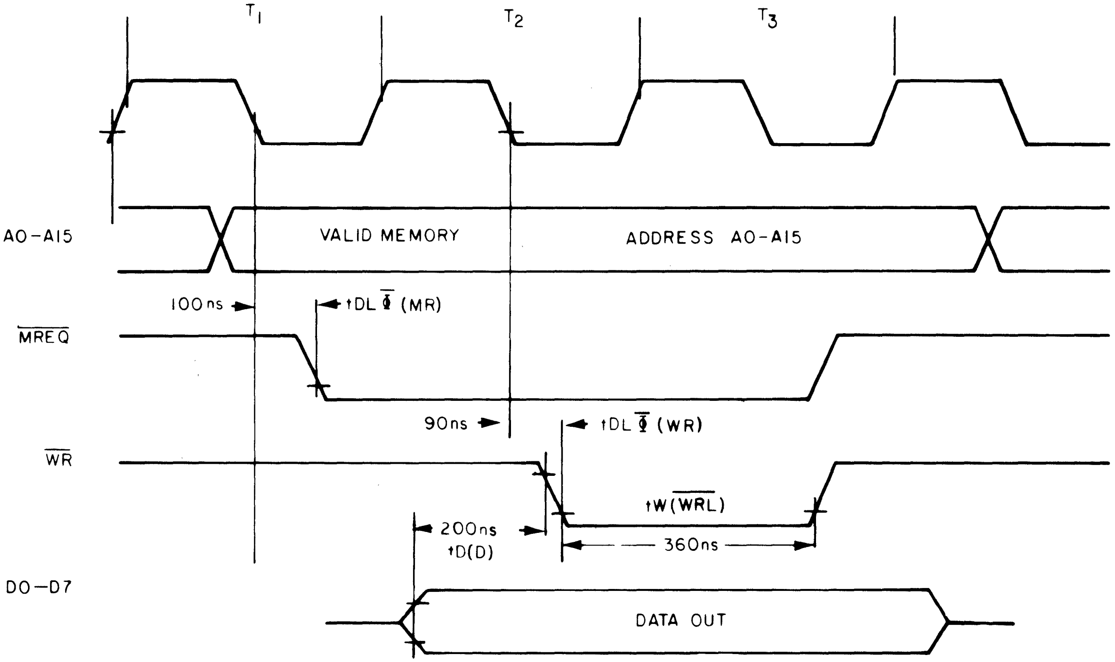
MEMORY READ TIMING

Figure 1b.



MEMORY WRITE TIMING

Figure 1c.



Z80 Family

Z80 REFRESH CONTROL AND TIMING

One of the most important features provided by the Z80 for interfacing to dynamic memories is the execution of a refresh cycle every time an op code fetch cycle is performed. By placing the refresh cycle in the op code fetch, the Z80 does not have to allocate time in the form of "wait states" or by "stretching" the clock to perform the refresh cycle. In other words, the refresh cycle is "totally transparent" to the CPU and does not decrease the system throughput (see Figure 1a). The refresh cycle is transparent to the CPU because, once the op code has been fetched from memory during states T_1 and T_2, the memory would normally be idle during states T_3 and T_4.

Therefore, by placing the refresh in the T_3 and T_4 states of the op code fetch, no time is lost for refreshing dynamic memory. The critical timing parameters involving the Z80 and dynamic memories during the refresh cycle are: $t_{W(MRH)}$ and $t_{W(MRL)}$. The parameter known as $t_{W(MRH)}$ refers to the time that $\overline{MREQ}$ is high during the op code fetch between the fetch of the op code and the refresh cycle. This time is known as "precharge" for dynamic memories and is necessary to allow certain internal nodes of the RAM to be charged-up for another memory cycle. The equation for the minimum $t_{W(MRH)}$ time period is:

(3) $t_{W(MRH)} = t_{W(\Phi H)} + t_f - 30$
where: $t_{W(\Phi H)}$ is clock pulse width high
t_f is clock fall time
let: $t_{W(\Phi H)} = 180\text{ns}; t_f = 10\text{ns}$
then: $\underline{t_{W(MRH)} = 160\text{ns (min)}}$

A $t_{W(MRH)}$ of 160ns is more than adequate to meet the worst case precharge times for most dynamic RAMs. For example, the MK4027-4 and the MK4116-4 require a 120ns precharge.The other refresh cycle parameter of importance to dynamic RAMs is $t_{W(MRL)}$, (the time that $\overline{MREQ}$ is low during the refresh cycle). This time is important because $\overline{MREQ}$ is used to directly generate $\overline{RAS}$. The equation for the minimum time period is:

(4) $t_{W(MRL)} = t_c - 40$
where: t_c is the clock period
let: $t_c = 400\text{ns}$
then: $\underline{t_{W(MRL)} = 360\text{ns}}$

A 360ns $t_{W(MRL)}$ exceeds the 250ns min $\overline{RAS}$ time required for the MK4027-4 and the MK4116-4.

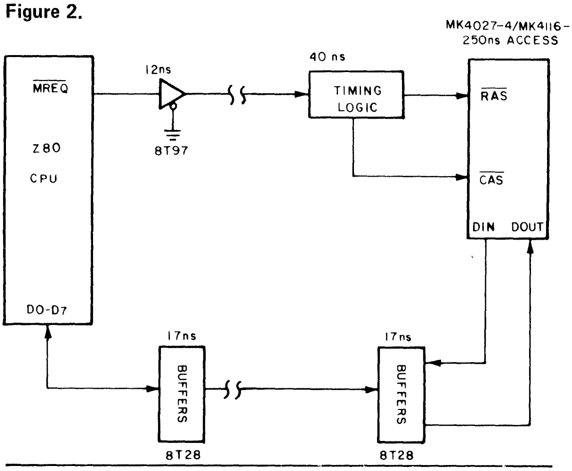
By controlling the refresh internally with the Z80, the designer must be aware of one limitation. The limitation is that to refresh memory properly, the Z80 CPU must be able to execute op codes since the refresh cycle occurs during the op code fetch. The following conditions cause the execution of op codes to be inhibited, and will destroy the contents of dynamic memory.

(1) Prolonged reset > 1ms
(2) Prolonged wait state operation > 1ms
(3) Prolonged bus acknowledge (DMA) > 1ms
(4) Φ clock of < 1.216 MHz for 16K RAMs
< .608 MHz for 4K RAMs

The clocks rate in number 4 are based on the Z80 continually executing the worst case instruction which is an EX (SP), HL that executes in 19 T states. Therefore, by operating the Z80 at or above these clocks frequencies, the user is ensured that the dynamic memories in the system will be refreshed properly.

Remember to refresh memory properly, the Z80 must be able to execute op codes!

DELAY FOR A TYPICAL MEMORY SYSTEM
Figure 2.



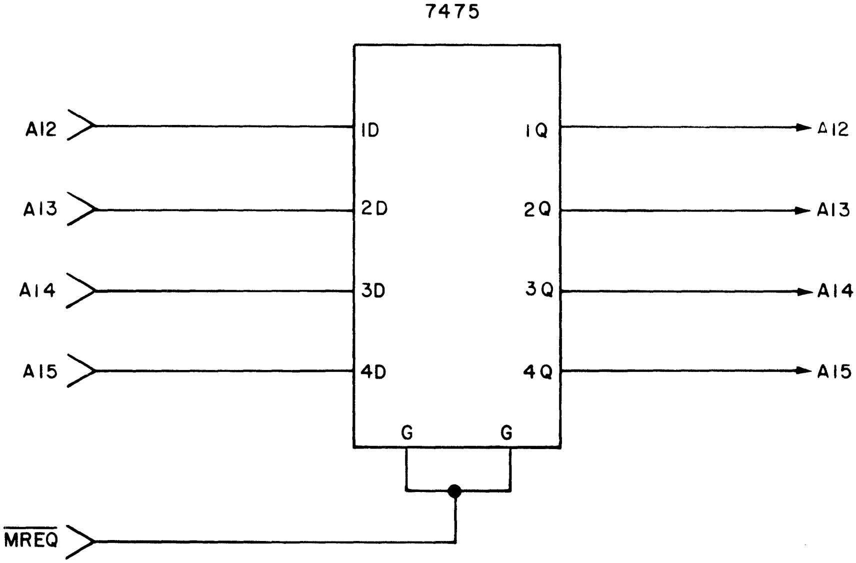
SUPPORT CIRCUITS FOR DYNAMIC MEMORY INTERFACE

Two support circuits are necessary to ensure reliable operation of dynamic memory with the Z80.

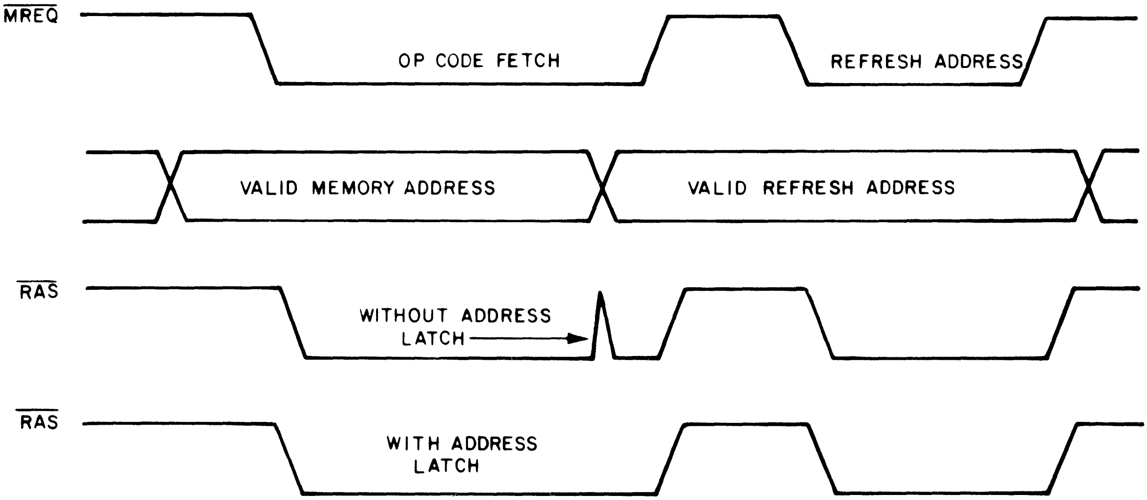
The first of these circuits is an address latch shown in Figure 3. The latch is used to hold addresses A_{12}-A_{15} while $\overline{MREQ}$ is active. This action is necessary because the Z80 does not ensure the validity of the address bus at the end of the op code fetch (see Figure 4). This action does not directly affect dynamic memories because they latch addresses internally. The problem comes from the address decoder which generates $\overline{RAS}$. If the address lines which drive the decoder are allowed to change while $\overline{MREQ}$ is low, then a "glitch" can occur on the $\overline{RAS}$ line or lines (if more than one row of RAMs are used) which may have the effect of destroying one row of data.

The second support circuit is used to generate a power on and short manual reset pulse. Recall from the discussion under Z80 Timing and Memory Con-

ADDRESS LATCH
Figure 3.



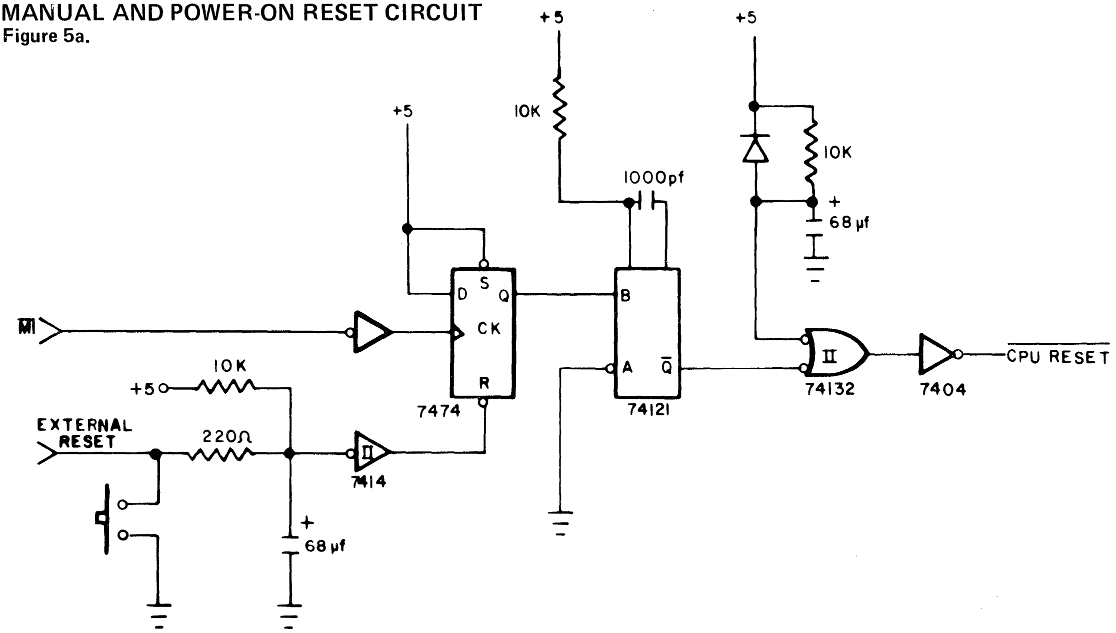
$\overline{\text{RAS}}$ TIMING WITH AND WITHOUT ADDRESS LATCH
Figure 4.



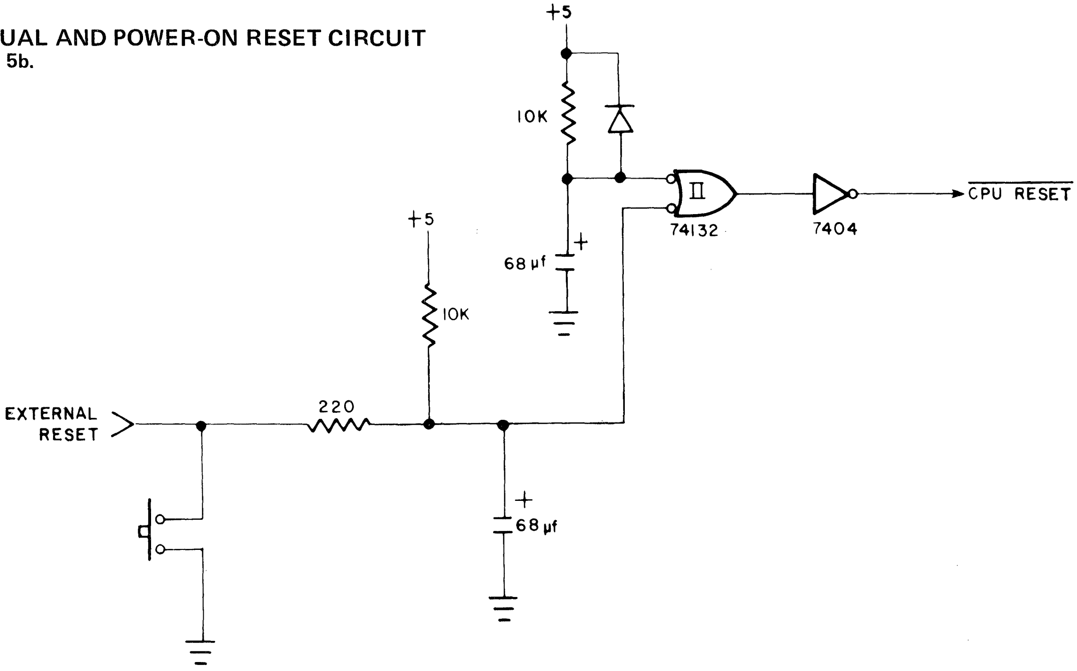
trol Signals that one of the conditions that will cause dynamic memory to be destroyed is a reset pulse of duration greater than 1ms. The circuit shown in Figure 5a can be used to generate a short reset pulse from either a push button or an external source. Additionally the manual reset is synchronized to the start of an M1 cycle so that the reset will not fall during the middle of a memory cycle. Along with the manual reset, the circuit will also generate a power on reset.

If it is not necessary that the contents of the dynamic memory be preserved, then the reset circuit shown in Figure 5b may be used to generate a manual or power on reset.

MANUAL AND POWER-ON RESET CIRCUIT
Figure 5a.

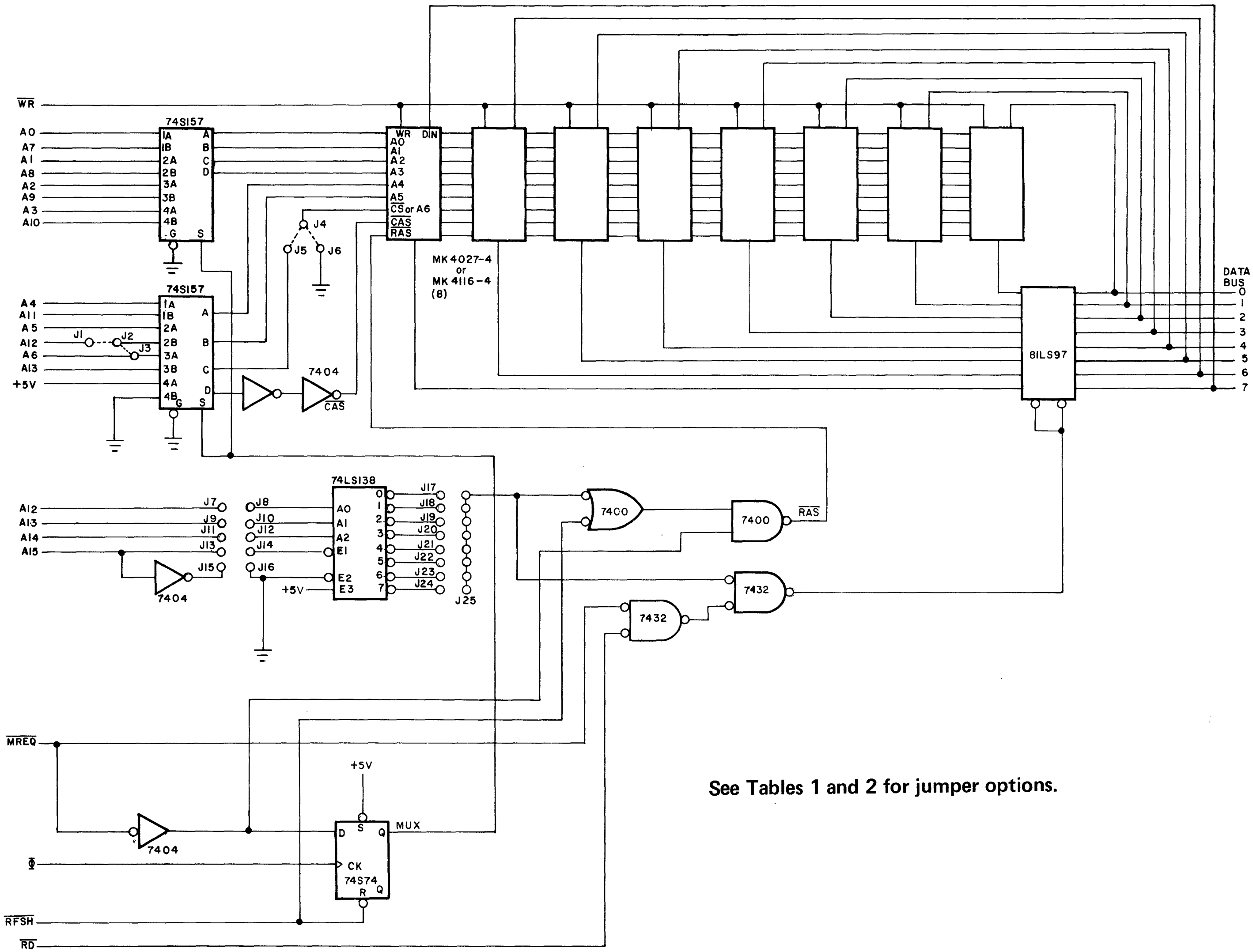


MANUAL AND POWER-ON RESET CIRCUIT
Figure 5b.



DESIGN EXAMPLE NO. 1 SCHEMATIC DIAGRAM

Figure 6.



See Tables 1 and 2 for jumper options.

DESIGN EXAMPLES FOR INTERFACING THE Z80 TO DYNAMIC MEMORY

To illustrate the interface between the Z80 and dynamic memory, two design examples are presented. Example number 1 is for a 4K/16Kx8 memory and the example number 2 is a 16K/64Kx8 memory.

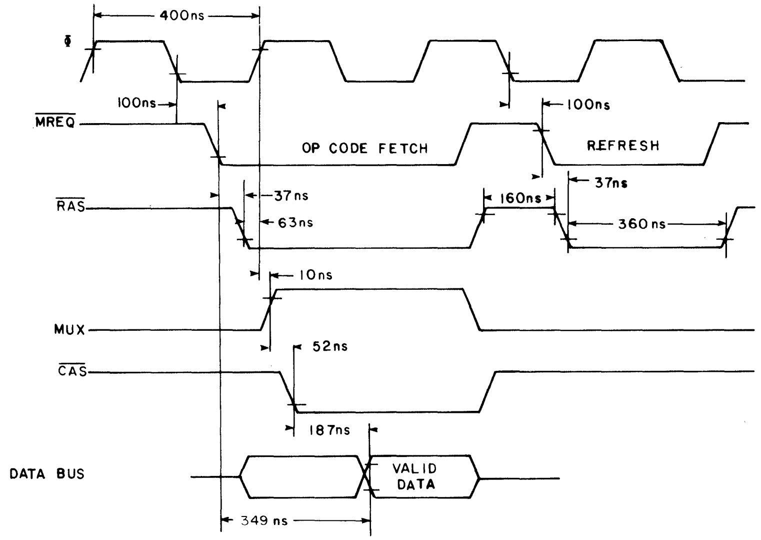
Design Example Number 1: 4K/16Kx8 Memory

This design example describes a 4K/16Kx8 memory that is best suited for a small single board Z80 based microcomputer system. The memory devices used in the example are the MK4027 (4,096x1 MOS Dynamic RAM) and the MK4116 (16,384x1 MOS Dynamic RAM). A very important feature of this design is the ease in which the memory can be expanded from a 4Kx8 to a 16Kx8 memory. This is made possible by the use of jumper options which configure the memory for either the MK4027 or the MK4116. See Table 1 and 2 for jumper options.

Figure 6 shows the schematic diagram for the 4K/16Kx8 memory. A timing diagram for the Z80 control signals and memory control signals is shown in Figure 7. The operation of the circuit may be described as follows: $\overline{RAS}$ is generated by NANDing MREQ with RFSH + ADDRESS DECODE. $\overline{RFSH}$ is generated directly from the Z80 while address decode comes from the 74LS138 decoder. Address decode indicates that the address on the bus falls within the memory boundaries of the memory. If an op code fetch or memory read is being executed the 81LS97 output buffer will be enabled at approximately the same time as $\overline{RAS}$ is generated for the memory array. The output buffer is enabled only during an op code fetch or memory read when $\overline{ADDRESS\ DECODE}$, $\overline{MREQ}$, and $\overline{RD}$ are all low. The switch multiplexer signal (MUX) is generated on the rising edge of Φ after $\overline{MREQ}$ has gone low during an op code fetch, memory read or memory write. After MUX is generated and the address multiplexers switch from the row address to column address, $\overline{CAS}$ will be generated. $\overline{CAS}$ comes from one of the outputs of the multiplexer and is delayed by two gate delays to ensure that the proper column address set-up time will be achieved. Once $\overline{RAS}$ and $\overline{CAS}$ have been generated for the memory array, the memory will then access the desired location for a read or write operation.

7404	22ns	Generate $\overline{RAS}$ from $\overline{MREQ}$
7400	15ns	
	63ns	$\overline{RAS}$ to rising edge of Φ
74S74	10ns	Φ to MUX
74S157	15ns	Generate $\overline{CAS}$ from MUX
7404	22ns	
7404	15ns	
t_{CAC}	165ns	$\overline{CAS}$ access time
81LS97	22ns	Output buffer delay
	349ns	Worst case access

DESIGN EXAMPLE NO. 1 MEMORY TIMING
Figure 7.



The worst case access time required by the CPU for the op code fetch is 450ns (from equation 1); therefore, the circuit exceeds the required access time by 101ns (worst case).

The circuit shown in Figure 6 provides excellent performance when used as a small on board memory. The memory size should be held at eight devices because there is not sufficient timing margin to allow the interface circuit to drive a larger memory array.

Design Example Number 2: 16Kx8 Memory

This design example describes a 16K/64Kx8 memory which is best suited for a Z80 based microcomputer system where a large amount of RAM is desired. The memory devices used in this example are the same as for the first example, the MK4027 and the MK4116. Again as with the first example, the memory may be expanded from a 16Kx8 to a 64Kx8 by reconfiguring jumpers. See Table 3 and 4 for jumper options.

Figure 8 shows the schematic diagram for the 16K/64K memory. A timing diagram is shown in Figure 9. The operation of the circuit can be described as follows: $\overline{RAS}$ is generated by NANDing $\overline{MREQ}$ with ADDRESS DECODE (from the two 74LS138s) + RFSH. Only one row of RAMs will receive a $\overline{RAS}$ during an op code fetch, memory read or memory write. However, a $\overline{RAS}$ will be generated for all rows within the array during a refresh cycle. $\overline{MREQ}$ is inverted and fed into a TTL compatible delay line to generate MUX and $\overline{CAS}$. (This particular approach differs from the method used in example number 1 in that all memory timing is referenced to $\overline{MREQ}$, whereas the circuit in example number 1 bases its memory timing from both $\overline{MREQ}$ and the clock. Both methods offer good results, however, the TTL delay line approach offers the best control over the memory timing.) MUX is generated 65ns later and is used to switch the 74157 multiplexers from the row to the column address. The 65ns delay was chosen to allow adequate margin for the row address hold time t_{RAH}. At 110ns, $\overline{CAS}$ is generated from the delay line and NANDed with $\overline{RFSH}$, which inhibits a $\overline{CAS}$ during refresh cycle. After $\overline{CAS}$ is applied to the memory, the desired location is then accessed. A worst case access timing analysis for the circuit shown in Figure 8 can be computed as follows:

74LS14	22ns	Generate $\overline{RAS}$ from $\overline{MREQ}$
74LS00	15ns	
delay line	50ns	MUX from $\overline{RAS}$
delay line	45ns	$\overline{CAS}$ delay from MUX
7400	20ns	
t_{CAC}	165ns	Access time from $\overline{CAS}$
8833	30ns	Output buffer delay
	347ns	

The required access time from the CPU is 450ns (from equation 1). This leaves 103ns of margin for additional CPU buffers on the control and address lines.This particular circuit offers excellent results for an application which requires a large amount of RAM memory. As mentioned earlier, the memory timing used in this example offers the best control over the memory timing and would be ideally suited for an application which required direct memory access (DMA).

4K x 8 CONFIGURATION(MK4027) JUMPER
Table 1

Connect: J2 to J3
J4 to J6
J7 to J8
J9 to J10
J11 to J12

CONNECT: J13 to J14

ADDRESS	CONNECT
0000-0FFF	J17 to J25
1000-1FFF	J18 to J25
2000-2FFF	J19 to J25
3000-3FFF	J20 to J25
4000-4FFF	J21 to J25
5000-5FFF	J22 to J25
6000-6FFF	J23 to J25
7000-7FFF	J24 to J25

CONNECT: J14 to J15

ADDRESS	CONNECT
8000-8FFF	J17 to J25
9000-9FFF	J18 to J25
A000-AFFF	J19 to J25
B000-BFFF	J20 to J25
C000-CFFF	J21 to J25
D000-DFFF	J22 to J25
E000-EFFF	J23 to J25
F000-FFFF	J24 to J25

16K x 8 CONFIGURATION (MK4116) JUMPER CONNECTIONS
Table 2

CONNECT:	J1 to J2	ADDRESS	CONNECT
	J4 to J5		
	J8 to J11	0-3FFF	J17 to J25
	J10 to J13	4000-7FFF	J18 to J25
	J12 to J16	8000-BFFF	J19 to J25
	J14 to J16	C000-FFFF	J20 to J25

16K x 8 CONFIGURATION (MK4027)

Table 3

CONNECT:	J1 to J3
	J5 to J6
	J7 to J8
	J9 to J10
	J11 to J12
	J13 to J14

ADDRESS:	0-3FFF	ADDRESS:	4000-7FFF	ADDRESS:	8000-BFFF	ADDRESS:	C000-FFFF
CONNECT:	J24 to J25	CONNECT:	J16 to J17	CONNECT:	J40 to J41	CONNECT:	J32 to J33
	J26 to J27		J18 to J19		J42 to J43		J34 to J35
	J28 to J29		J20 to J21		J44 to J43		J36 to J37
	J30 to J31		J22 to J23		J46 to J47		J38 to J39

64K x 8 CONFIGURATION(MK4116)

Table 4

CONNECT:	J1 to J2	ADDRESS:	0-FFFF
	J4 to J5	CONNECT:	J32 to J33
	J8 to J11		J34 to J35
	J10 to J13		J36 to J37
	J12 to J15		J38 to J39
	J14 to J15		

SYSTEM PERFORMANCE CHARACTERISTICS

Table 5

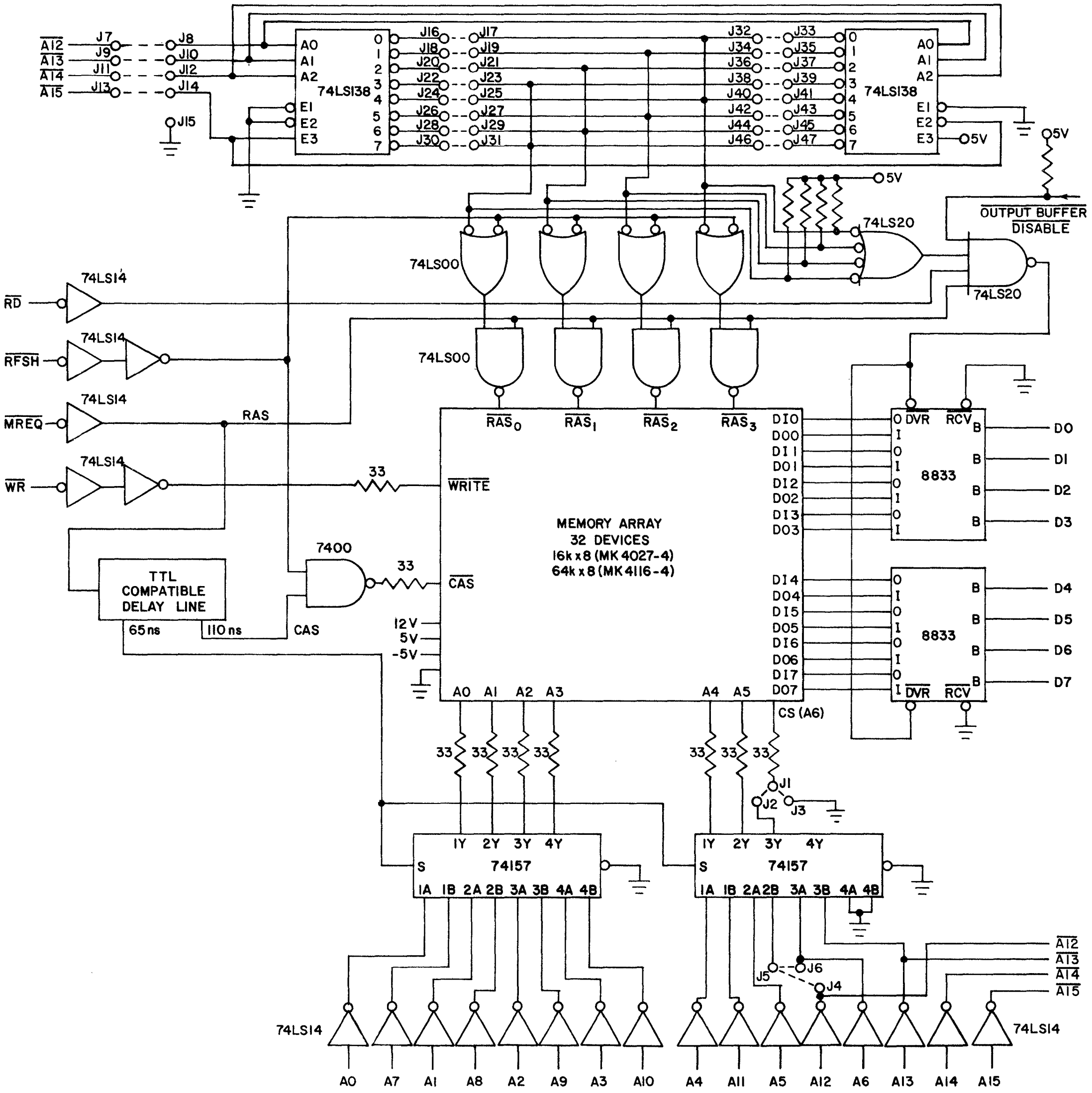
The system characteristics for the preceeding design examples are shown in Table 5.

EXAMPLE #	MEMORY CAPACITY	MEMORY ACCESS	POWER REQUIREMENTS
1	4K/16Kx8	349ns max.	+12V @ 0.0250 A max. +5V @ 0.422 A max.* -5V @ 0.030 A max.
2	16K/64Kx8	347ns max.	+12V @ 0.600 A max. +5V @ 0.550 A max. * -5V @ 0.030 A max.

*All power requirements are max.; operating temperature 0°C to 70°C ambient, max +12V current computed with Z80 executing continuous op code fetch cycles from RAM at 1.6 μ s intervals.

DESIGN EXAMPLE NO. 2 SCHEMATIC DIAGRAM

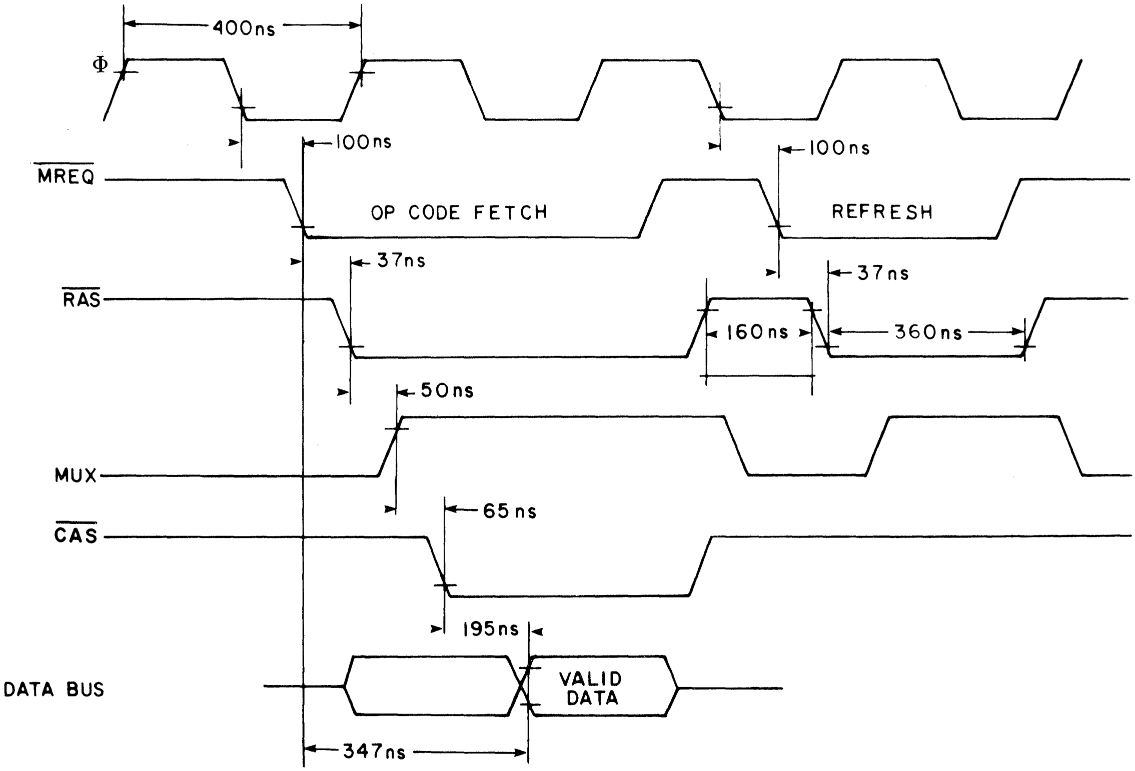
Figure 8.



FOR JUMPER OPTIONS SEE TABLES 3 AND 4

DESIGN EXAMPLE NO. 2 MEMORY TIMING

Figure 9.



PRINTED CIRCUIT LAYOUT

One of the most important parts of a dynamic memory design is the printed circuit layout. Figure 10 illustrates a recommended layout for 32 devices. A very important factor in the P.C. layout is the power distribution. Proper power distribution on the V_{DD} and V_{BB} supply lines is necessary because of the transient current characteristics which dynamic memories exhibit. To achieve proper power distribution, V_{DD}, V_{BB}, V_{CC} and ground should be laid out in a grid to help minimize the power distribution impedance. Along with good power distribution, adequate capacitive bypassing for each device in the memory array is necessary. In addition to the individual by-passing capacitors, it is recommended that each supply (V_{BB}, V_{CC} and V_{DD}) be bypassed with an electrolytic capacitor 20μF.

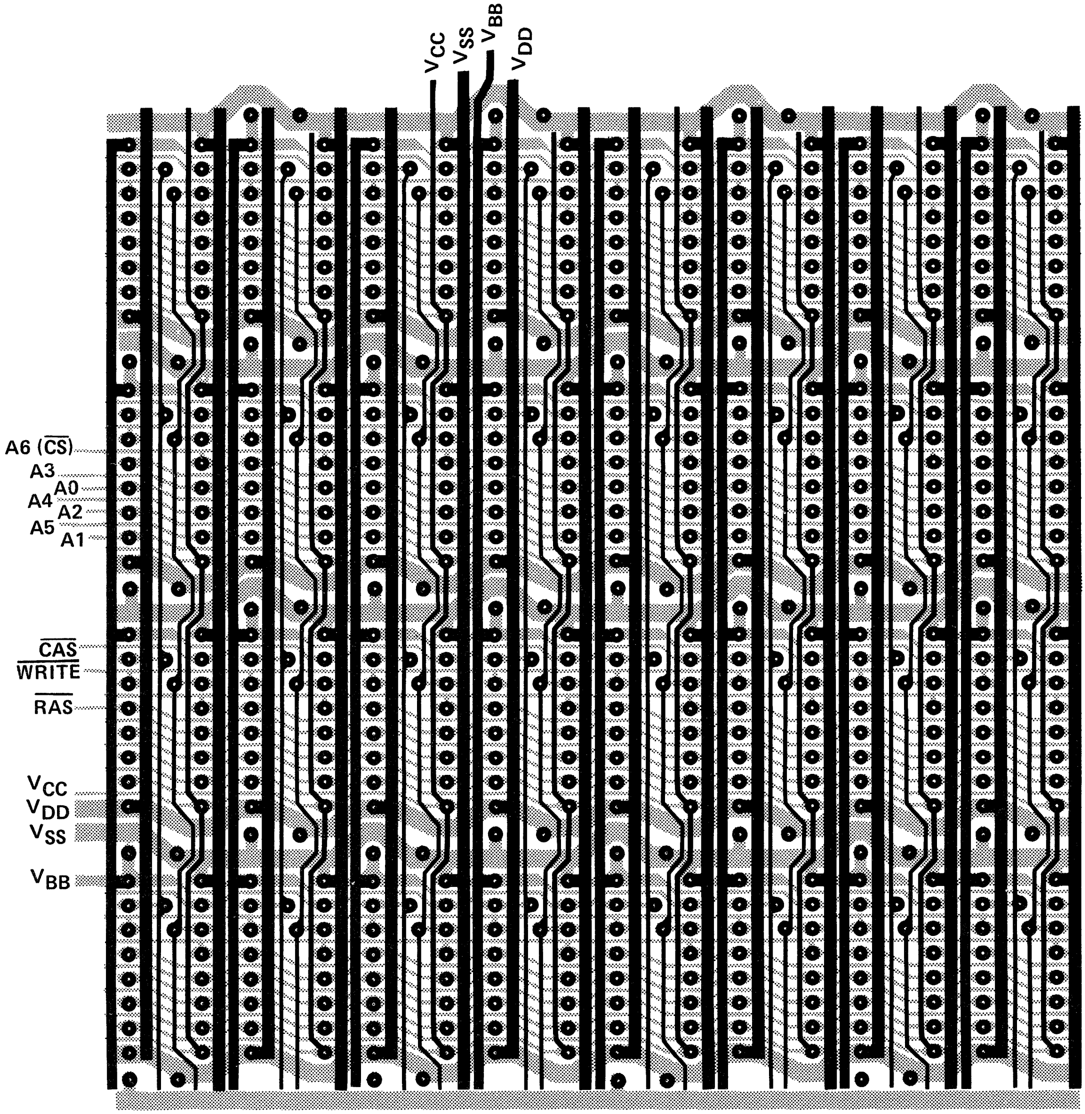
By using good power distribution techniques and using the recommended number of bypassing capacitors, the designer can minimize the amount of noise in the memory array. Other layout considerations are the placement of signal lines. Lines such as address, chip select, column address strobe, and write should be bussed together as rows; then, bus all rows together at one end of the array. Interconnection between rows should be avoided. Row address strobe lines should be bussed together as a row, then connected to the appropriate $\overline{\text{RAS}}$ driver. TTL drivers for the memory array signals should be located as close as possible to the array to help minimize signal noise.

For a large memory array such as the one shown in design example number 2, series terminating resistors should be used to minimize the amount of negative undershoot. These resistors should be used on the address lines, $\overline{\text{CAS}}$ and $\overline{\text{WRITE}}$, and have values between 20 Ω to a 33 Ω .

The layout for a 32 device array can be put in a 5" x 5" area on a two sided printed circuit board.

SUGGESTED P. C. LAYOUT FOR MK4027 or MK4116

Figure 10.



4MHz Z80 DYNAMIC MEMORY INTERFACE CONSIDERATIONS

A 4MHz Z80 is available for the microcomputer designer who needs higher system throughput. Considerations which must be faced by the designer when interfacing the 4MHz Z80 to dynamic memory are the need for memories with faster access times and for providing minimum RAM precharge time. The access times required for dynamic memory interfaced to a 4MHz Z80 can be computed from equations 1 and 2 under Z80 Timing and Memory Control Signals.

Access time for op code fetch for 4MHz Z80,

let: $t_C = 250\text{ns};\ t_{DL\overline{\Phi}(MR)} = 75\text{ns};\ t_{s\Phi\ (D)} = 35\text{ns}$

then: $t_{\text{ACCESS OP CODE}} = 265\text{ns}$

Access time for memory read for 4MHz Z80,

let: $t_C = 250\text{ns};\ t_{DL\overline{\Phi}(MR)} = 75\text{ns};\ t_{S\overline{\Phi}\ (D)} = 50\text{ns}$

then: $\underline{t_{\text{ACCESS MEMORY READ}} = 375\text{ns}}$

The problem of faster access times can be solved by using 200ns memories such as the MK4027-3 or MK4116-3. Depending on the number of buffer delays in the system, the designer may have to use 150ns memories such as the MK4027-2 or MK4116-2. The most critical problem that exists when interfacing dynamic memory to the 4MHz Z80 is the RAM precharge time (trp). This parameter is called $t_{W(MRH)}$ on the Z80 and can be computed by the following equation.

(4) $t_{W(RH)} = t_{W(\Phi H)} + t_f\text{-}20\text{ns}$

let: $t_{W(\Phi H)} = 110\text{ns};\ t_f = 5\text{ns}$

then: $\underline{t_{W(MRH)} = 95\text{ns}}$

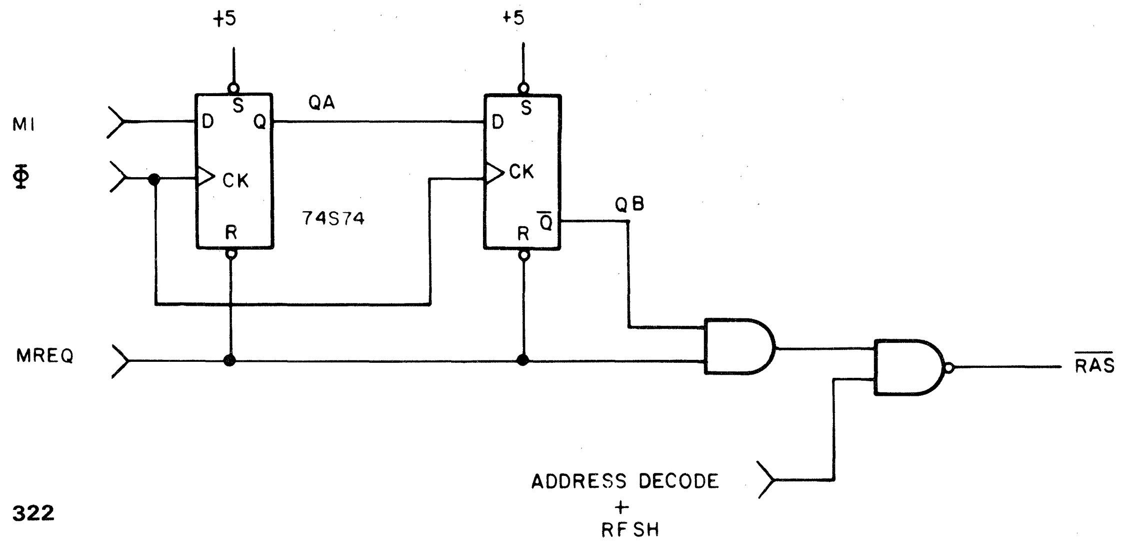
A $t_{W(MRH)}$ of 95ns will not meet the minimum precharge time of the MK4027-2 or MK4116-2 which is 100ns. The MK4027-3 and MK4116-3 require a 120ns precharge. Figure 11 shows a circuit that will lengthen the $t_{W(MRH)}$ pulse from 95ns to a minimum of 126ns while only inserting one gate delay into the access timing chain. Figure 12 shows the timing for the circuit of Figure 11. The operation of the circuit in Figure 11 can be explained as follows: The D flip flops are held in a reset condition until MREQ goes to its active state. After MREQ goes active, on the next positive clock edge, the D input of U1 and U2 will be transferred to the outputs of the flip flops. Output QA will go high if M1 was high when Φ clocked U1. Output QB will go low on the next positive going clock edge, which will cause the output of U3 to go low and force the output of U4, which is $\overline{\text{RAS}}$, high. The flip flops will be reset when MREQ goes inactive.

The circuit shown in Figure 11 will give a minimum of 126ns precharge for dynamic memories, with the Z80 operating at 4MHz. The 126ns $t_{W(MRH)}$ is computed as follows.

110ns	$t_{W(\Phi H)}$ - clock pulse width high (min)
5ns	t_F - clock full time (min)/
20ns	$t_{DL\overline{\Phi}(MR)}$ - MREQ delay (min)
-9ns	74S74 delay (min)
126ns	$t_{W(MRH)}$ modified (min)

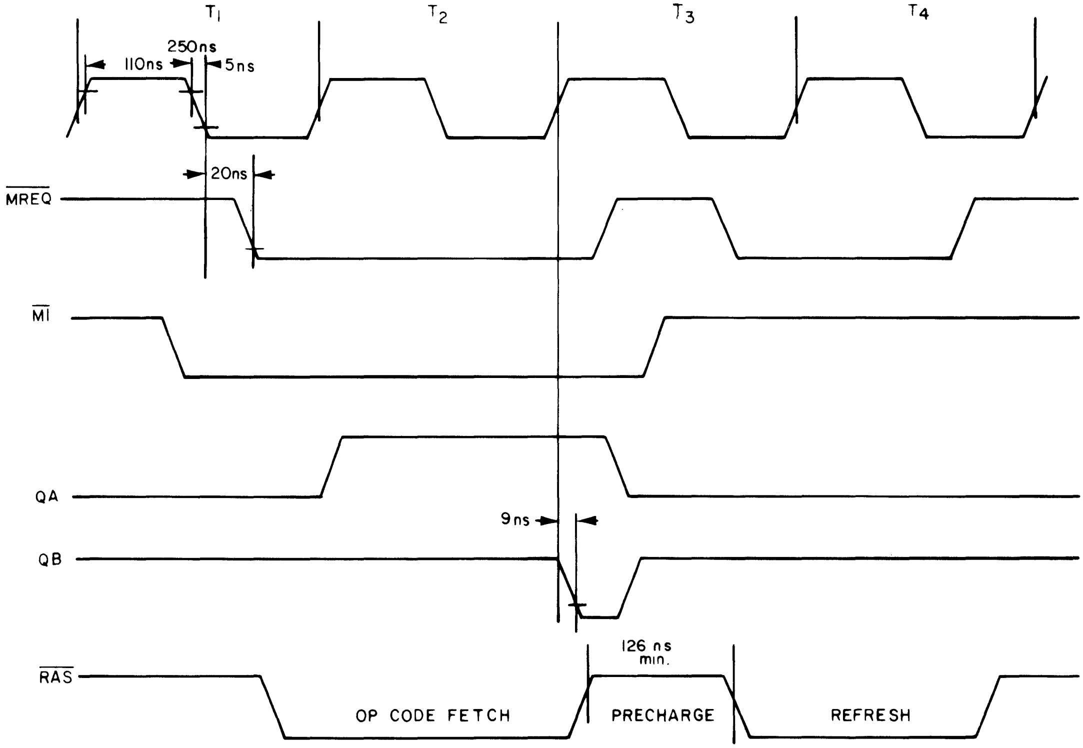
4MHz Z80 PRECHARGE EXTENDER FOR DYNAMIC MEMORIES

Figure 11.



TIMING DIAGRAM FOR 4MHz Z80 PRECHARGE EXTENDER

Figure 12



APPENDIX

MEMORY TEST ROUTINE

This section is intended to give the microcomputer designer a memory diagnostic suitable for testing memory systems such as the ones shown in Section VI.

The routine is a modified address storage test with an incrementing pattern. A complete test requires 256_{10} passes, which will execute in less than 4 minutes for a 16Kx8 memory. If an error occurs, the program will store the pattern in location '2C'H and the address of the error at locations '2D'H and '2E'H.

The program is set up to test memory starting at location '2F'H up to the end of the block of memory defined by the bytes located at 'OC'H and 'OD'H. The test may be set up to start at any location by modifying locations '03'H - '04'H and '11'H - '12'H with the starting address that is desired.

```
                             MXRTS  LISTING                    PAGE   0001
  LOC    OBJ CODE    STMT SOURCE STATEMENT

                     0001 ;TRANSLATED FROM DEC 1976 INTERFACE MAGAZINE
                     0002 ;
                     0003 ;THIS IS A MODIFIED ADDRESS STORAGE TEST WITH AN
                     0004 ;INCREMENTING PATTERN
                     0005 ;
                     0006 ;256 PASSES MUST BE EXECUTED BEFORE THE MEMORY IS
                     0007 ;COMPLETELY TESTED.
                     0008 ;
                     0009 ;IF AN ERROR OCCURS, THE PATTERN WILL BE STORED
                     0010 ;AT LOCATION '002C'H AND THE ADDRESS OF THE
                     0011 ;ERROR LOCATION WILL BE STORED AT '002D'H AND
                     0012 ;'002E'H.
                     0013 ;
```

```
                        ;THE CONTENTS OF LOCATIONS '000C'H AND '001D'H
                        ;SHOULD BE SELECTED ACCORDING TO THE FOLLOWING
                        ;MEMORY SIZE TO BE TESTED
                        ;
                        ;TOP OF MEMORY TO
                        ;BE TESTED                        VALUE OF EPAGE
                        ;
                        ;      4K                                  '10'H
                        ;      8K                                  '20'H
                        ;     16K                                  '40'H
                        ;     32K                                  '80'H
                        ;     48K                                  'C0'H
                        ;     64K                                  'FF'H
                        ;
                        ;THE PROGRAM IS SET UP TO START TESTING AT
                        ;LOCATION '002F'H. THE STARTING ADDRESS FOR THE
                        ;TEST CAN BE MODIFIED BY CHANGING LOCATIONS
                        ;'0003-0004'H AND '0011-0012'H.
                        ;
                        ;TEST TIME FOR A 16K X 8 MEMORY IS APPROX. 4 MIN
                        ;
0000                               ORG    0000H
0000     0600                      LD     B,0         ;CLEAR B PATRN MODIFIER
                        ;LOAD UP MEMORY
0002     212F00         LOOP:      LD     HL,START    ;GET STARTING ADDR
0005     7D             FILL:      LD     A,L         ;LOW BYTE TO ACCM
0006     AC                        XOR    H           ;XOR WITH HIGH BYTE
0007     A8                        XOR    B           ;XOR WITH PATTERN
0008     77                        LD     (HL),A      ;STORE IN ADDR
0009     23                        INC    HL          ;INCREMENT ADDR
000A     7C                        LD     A,H         ;LOAD HIGH BYTE OF ADDR
000B     FE10                      CP     EPAGE       ;COMPARE WITH STOP ADDR
000D     C20500                    JP     NZ,FILL     ;NOT DONE,GO BACK
                        ;READ AND CHECK TEST DATE
0010     212F00                    LD     HL,START    ;GET STARTING ADDR
0013     7D             TEST:      LD     A,L         ;LOAD LOW BYTE
0014     AC                        XOR    H           ;XOR WITH HIGH BYTE
0015     A8                        XOR    B           ;XOR WITH MODIFIER
0016     BE                        CP     (HL)        ;COMPARE WITH MEMORY LOC
0017     C22500                    JP     NZ,FXIT     ;ERROR EXIT
001A     23                        INC    HL          ;UPDATE MEMORY ADDRESS
001B     7C                        LD     A,H         ;LOAD HIGH BYTE
001C     FE10                      CP     EPAGE       ;COMPARE WITH STOP ADDR
001E     C21300                    JP     NZ,TEST     ;LOOP BACK
0021     04                        INC    B           ;UPDATE MODIFIER



                                   MXRTS  LISTING                 PAGE   0002
 LOC     OBJ CODE       STMT SOURCE STATEMENT

0022     C30200                    JP     LOOP        ;RST WITH NEW MODIFIER
                        ;ERROR EXIT
0025     222D00         FXIT:      LD     (BYTE),HL   ;SAVE ERROR ADDRESS
0028     322C00                    LD     (PATRN),A   ;SAVE BAD PATTERN
002B     76                        HALT               ;FLAG OPERATOR
002C                    PATRN:     DEFS   1
002D                    BYTE:      DEFS   2
002F     2F00           START:     DEFW   $
                        EPAGE:     EQU    10H         ;SET UP FOR 4K TEST
                                   END
```

1979 MICROCOMPUTER DATA BOOK

MOSTEK®

F8 MICROCOMPUTER DEVICES

Single-Chip Microcomputer MK3870

FEATURES

- □ Software compatible with 3870/F8 family
- □ 2048 X 8 mask programmable ROM
- □ 64 byte scratchpad RAM
- □ 32 bits (4 ports) TTL compatible I/O
- □ Programmable binary timer
  - Interval timer mode
  - Pulse width measurement mode
  - Event counter mode
- □ External interrupt
- □ Crystal, LC, RC, or external time base
- □ Low power (275 mW typ.)
- □ Single +5 volt ± 10% power supply
- □ Pinout compatible with 3870 family

GENERAL DESCRIPTION

The MK3870 is a complete 8-bit microcomputer on a single MOS integrated circuit. The 3870 can execute the F8 instruction set of more than 70 commands, allowing expansion into multi-chip configurations with software compatibility. The device features 2048 bytes of ROM, 64 bytes of scratchpad RAM, a programmable binary timer, 32 bits of I/O, and a single +5 volt power supply requirement.

Utilizing ion-implanted, N-channel silicon-gate technology and advanced circuit design techniques, the single-chip 3870 offers maximum cost effectiveness in a wide range of control and logic replacement applications.

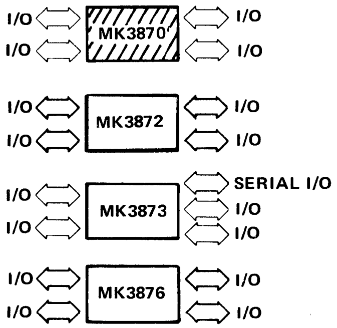
FUNCTIONAL PIN DESCRIPTION

$\overline{\text{P0-0}}$—$\overline{\text{P0-7}}$, $\overline{\text{P1-0}}$—$\overline{\text{P1-7}}$, $\overline{\text{P4-0}}$—$\overline{\text{P4-7}}$, and $\overline{\text{P5-0}}$—$\overline{\text{P5-7}}$ are 32 lines which can be individually used as either TTL compatible inputs or as latch outputs.

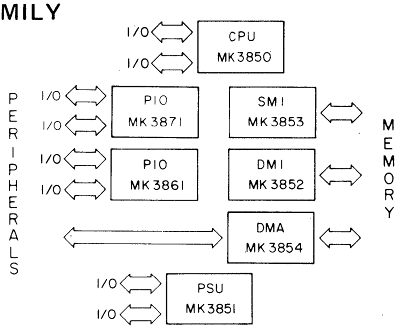
$\overline{\text{STROBE}}$ is a ready strobe associated with I/O Port 4. This pin which is normally high provides a single low pulse after valid data is present on the P4-0—P4-7 pins during an output instruction.

$\overline{\text{RESET}}$ may be used to externally reset the 3870. When pulled low the 3870 will reset. When then

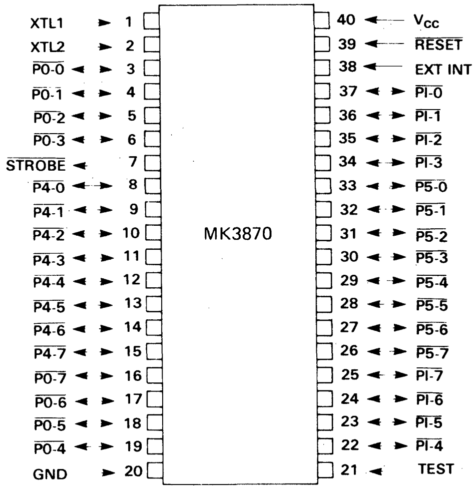
SINGLE CHIP 3870 MICROCOMPUTER FAMILY



F8 FAMILY



PIN CONNECTIONS


XTL1 1
XTL2 2
$\overline{\text{P0-0}}$ 3
$\overline{\text{P0-1}}$ 4
$\overline{\text{P0-2}}$ 5
$\overline{\text{P0-3}}$ 6
STROBE 7
$\overline{\text{P4-0}}$ 8
$\overline{\text{P4-1}}$ 9
$\overline{\text{P4-2}}$ 10
$\overline{\text{P4-3}}$ 11
$\overline{\text{P4-4}}$ 12
$\overline{\text{P4-5}}$ 13
$\overline{\text{P4-6}}$ 14
$\overline{\text{P4-7}}$ 15
$\overline{\text{P0-7}}$ 16
$\overline{\text{P0-6}}$ 17
$\overline{\text{P0-5}}$ 18
$\overline{\text{P0-4}}$ 19
GND 20
40 V_{CC}
39 $\overline{\text{RESET}}$
38 EXT INT
37 $\overline{\text{PI-0}}$
36 $\overline{\text{PI-1}}$
35 $\overline{\text{PI-2}}$
34 $\overline{\text{PI-3}}$
33 $\overline{\text{P5-0}}$
32 $\overline{\text{P5-1}}$
31 $\overline{\text{P5-2}}$
30 $\overline{\text{P5-3}}$
29 $\overline{\text{P5-4}}$
28 $\overline{\text{P5-5}}$
27 $\overline{\text{P5-6}}$
26 $\overline{\text{P5-7}}$
25 $\overline{\text{PI-7}}$
24 $\overline{\text{PI-6}}$
23 $\overline{\text{PI-5}}$
22 $\overline{\text{PI-4}}$
21 TEST
MK3870

PIN NAME	DESCRIPTION	TYPE
$\overline{\text{P0-0}}$ – $\overline{\text{P0-7}}$	I/O Port 0	Bidirectional
$\overline{\text{P1-0}}$ – $\overline{\text{P1-7}}$	I/O Port 1	Bidirectional
$\overline{\text{P4-0}}$ – $\overline{\text{P4-7}}$	I/O Port 4	Bidirectional
$\overline{\text{P5-0}}$ – $\overline{\text{P5-7}}$	I/O Port 5	Bidirectional
$\overline{\text{STROBE}}$	Ready Strobe	Output
EXT INT	External Interrupt	Input
$\overline{\text{RESET}}$	External Reset	Input
TEST	Test Line	Input
XTL 1, XTL 2	Time Base	Input
V_{CC}, GND	Power Supply Lines	Input

allowed to go high the 3870 will begin program execution at program location H '000'.

EXT INT is the external interrupt input. Its active state is software programmable. This input is also used in conjunction with the timer for pulse width measurement and event counting.

XTL 1 and XTL 2 are the time base inputs to which a crystal (1 to 4 MHz), LC network, RC network, or an external single-phase clock may be connected.

TEST is an input, used only in testing the 3870. For normal circuit functionality this pin is left unconnected or may be grounded.

VCC is the power supply input (+5V ± 10%).

3870 ARCHITECTURE

This section describes the basic functional elements of the 3870 as shown in the block diagram of Figure 1. A programming model is shown in Figure 2.

Main Control Logic

The Instruction Register (IR) receives the operation code (OP code) of the instruction to be executed from the program ROM via the data bus. During all OP code fetches eight bits are latched into the IR. Some instructions are completely specified by the upper 4 bits of the OP code. In those instructions the lower 4 bits are an immediate register address or an immediate 4 bit operand. Once latched into the IR the main control logic decodes the instruction and provides the necessary control gating signals to all circuit elements.

ROM Address Registers

There are four 11 bit registers associated with the 2K x 8 ROM. These are the Program Counter (P0), the Stack Register (P), the Data Counter (DC) and the Auxiliary Data Counter (DC1). The Program Counter is used to address instructions or immediate operands. P is used to save the contents of P0 during an interrupt or subroutine call. Thus P contains the return address at which processing is to resume upon completion of the subroutine or the interrupt routine.

The Data Counter (DC) is used to address data tables. This register is auto-incrementing. Of the two data counters only DC can access the ROM. However, the XDC instruction allows DC and DC1 to be exchanged.

Associated with the address registers is an 11 bit Adder/Incrementer. This logic element is used to increment P0 or DC when required and is also used to add displacements to P0 on relative branches or to add the data bus contents to DC in the ADC (Add Data Counter) instruction.

2048 X 8 ROM

The microcomputer program and data constants are stored in the program ROM. When a ROM access is required, the appropriate address register (P0 or DC) is gated onto the ROM address bus and the ROM output is gated onto the main data bus. The first byte in the ROM is location zero.

Scratchpad and IS

The scratchpad provides 64 8-bit registers which may be used as general purpose RAM memory. The Indirect Scratchpad Address Register (IS) is a 6 bit register used to address the 64 registers. All 64 registers may be accessed using IS. In addition the lower order 12 registers may also be directly addressed.

IS can be visualized as holding two octal digits. This division of IS is important since a number of instructions increment or decrement only the least significant 3 bits of IS when referencing scratchpad bytes via IS. This makes it easy to reference a buffer consisting of contiguous scratchpad bytes. For example, when the low order octal digit is incremented or decremented IS is incremented from octal 27 (0 '27') to 0 '20) or is decremented from 0 '20' to 0 '27'. This feature of the IS is very useful in many program sequences. All six bits of IS may be loaded at one time or either half may be loaded independently.

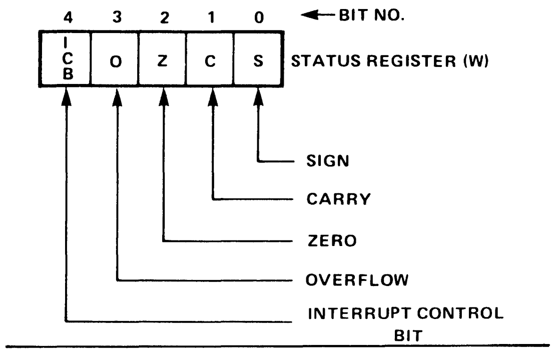
Scratchpad registers 9 through 15 (decimal) are given mnemonic names (J, H, K, and Q) because of special linkages between these registers and other registers such as the Stack Register. These special linkages facilitate the implementation of multi-level interrupts and subroutine nesting. For example, the instruction LR K,P stores the lower eight bits of the Stack Register into register 13 (K lower or KL) and stores the upper three bits of P into register 12 (K upper or KU).

Arithmetic and Logic Unit (ALU)

After receiving commands from the main control logic, the ALU performs the required arithmetic or logic operations (using the data presented on the two input busses) and provides the result on the result bus. The arithmetic operations that can be performed in the ALU are binary add, decimal adjust, add with carry, decrement, and increment. The logic operations that can be performed are AND, OR, EXCLUSIVE OR, 1's complement, shift right, and shift left. Besides providing the result on the result bus, the ALU also provides four signals representing the status of the result. These signals, stored in the Status Register (W), represent CARRY, OVERFLOW, SIGN, and ZERO condition of the result of the operation.

Accumulator(A)

The Accumulator (A) is the principal register for data manipulation within the 3870. The A serves as one input to the ALU for arithmetic or logical operations. The result of ALU operations are stored in the A.



Summary of Status Bits

$$\text{OVERFLOW} = \text{CARRY}_7 \oplus \text{CARRY}_6$$

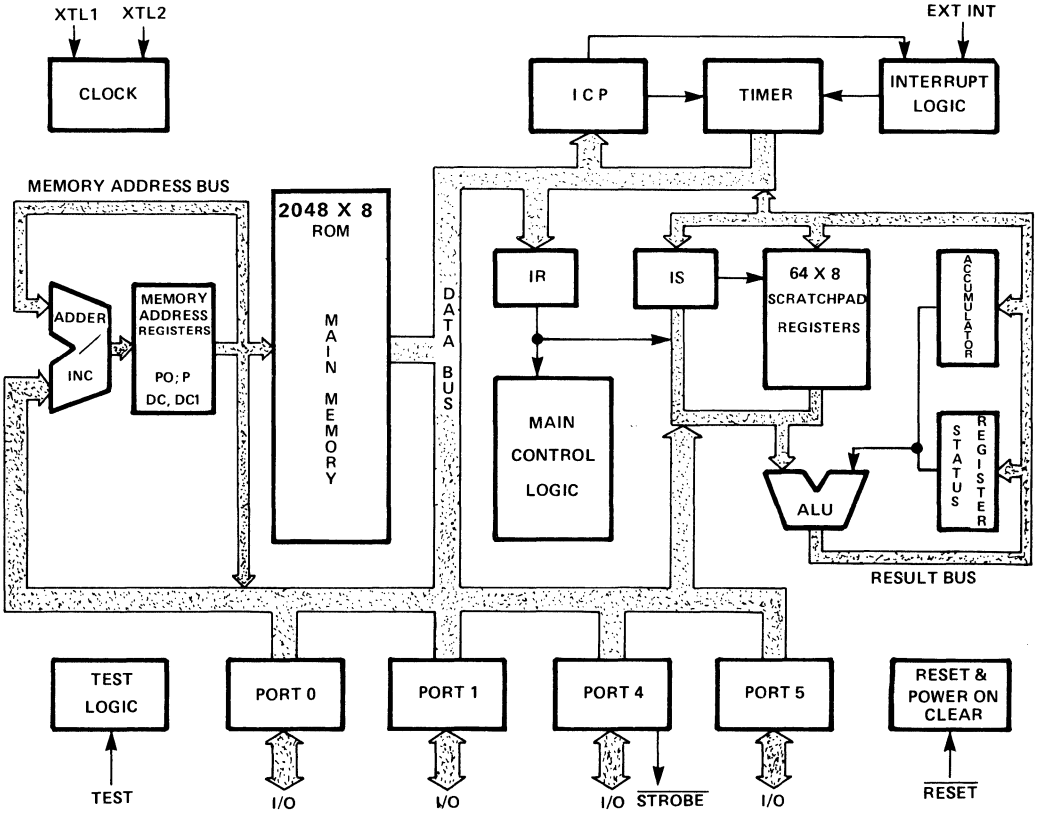
$$\text{ZERO} = \overline{\text{ALU}_7} \wedge \overline{\text{ALU}_6} \wedge \overline{\text{ALU}_5} \wedge \overline{\text{ALU}_4} \wedge \overline{\text{ALU}_3} \wedge \overline{\text{ALU}_2} \wedge \overline{\text{ALU}_1} \wedge \overline{\text{ALU}_0}$$

$$\text{CARRY} = \text{CARRY}_7$$

$$\text{SIGN} = \overline{\text{ALU}_7}$$

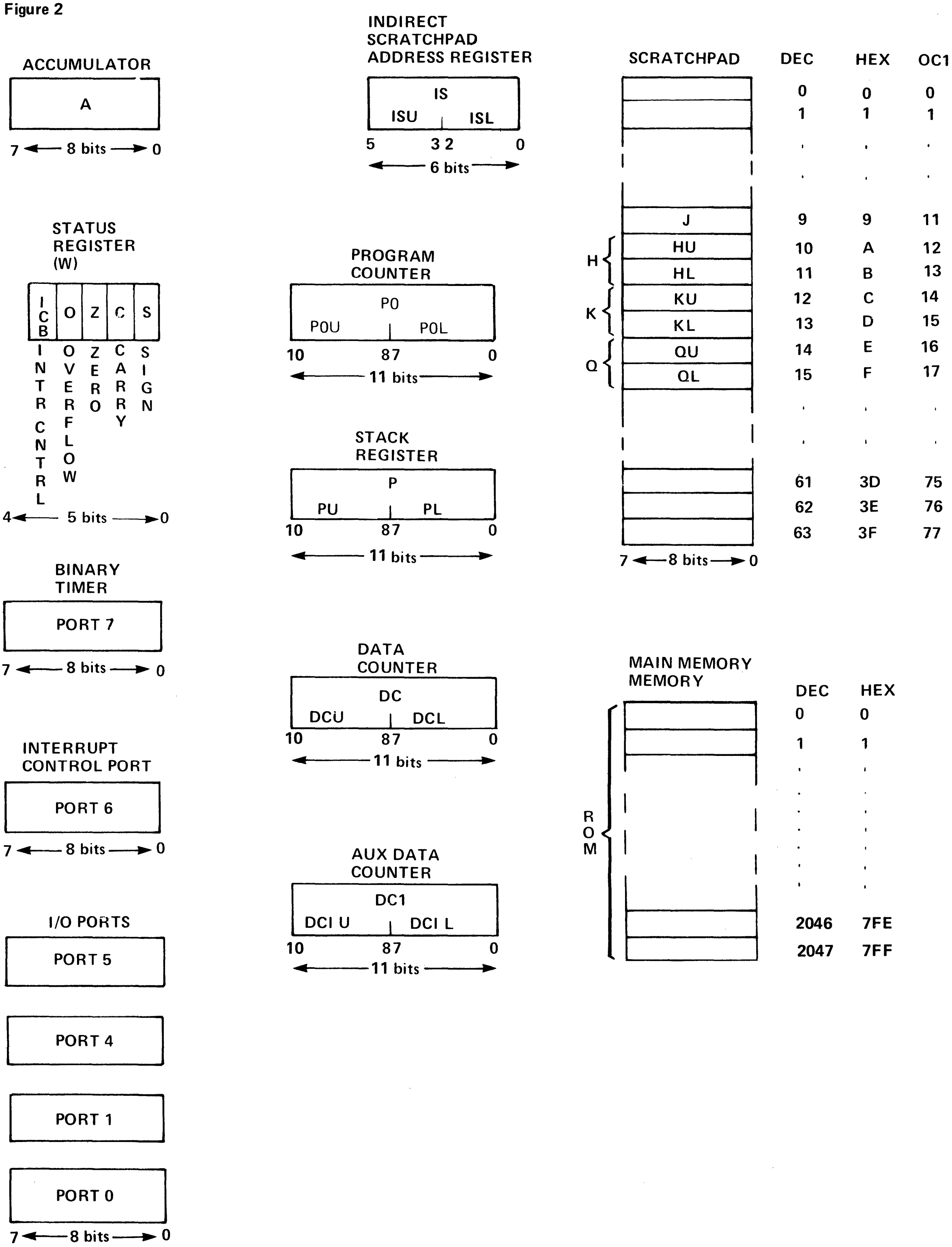
MK3870 BLOCK DIAGRAM

Figure 1



3870 PROGRAMMABLE REGISTERS, PORTS AND MEMORY MAP

Figure 2



The Status Register(W)

The Status Register (also called the W register) holds five status flags as follows:

Interrupt Control Bit (ICB)

The ICB may be used to allow or disallow interrupts in the 3870. This bit is not the same as the two interrupt enable bits in the Interrupt Control Port (ICP). If the ICB is set and the 3870 interrupt logic communicates an interrupt request to the CPU section, the interrupt will be acknowledged and processed upon completion of the first non-privileged instruction. If the ICB is cleared an interrupt request will not be acknowledged or processed until the ICB is set.

I/O Ports

The 3870 provides four complete bidirectional Input/Output ports. These are ports 0, 1, 4, and 5. In addition, the Interrupt Control Port is addressed as port 6 and the binary timer is addressed as port 7. An output instruction (OUT or OUTS) causes the contents of A to be latched into the addressed port. An input instruction (IN or INS) transfers the contents of the port to A (port 6 is an exception which is described later). The schematic of an I/O pin and available output drive options are shown in Figure 3.

An output ready strobe is associated with port 4. This flag may be used to signal a peripheral device that the 3870 has just completed an output of new data to port 4. The strobe provides a single low pulse shortly after the output operation is completely finished, so either edge may be used to signal the peripheral. $\overline{\text{STROBE}}$ may also be used as an input strobe simply by doing a dummy output of H '00' strobe to port 4 after completing the input operation.

Timer and Interrupt Control Port

The Timer is an 8-bit binary down counter which is software programmable to operate in one of three modes: the Interval Timer Mode, the Pulse Width Measurement Mode, or the Event Counter Mode. As shown in Figure 4, associated with the Timer are an 8-bit register called the Interrupt Control Port, a programmable prescaler, and an 8-bit modulo-N register. A functional logic diagram is shown in Figure 5.

The desired timer mode, prescale value, starting and stopping the timer, active level of the EXT INT pin, and local enabling or disabling of interrupts are selected by outputting the proper bit configuration from the Accumulator to the Interrupt Control Port (port 6) with an OUT or OUTS instruction. Bits within the Interrupt Control Port are defined as follows:

Interrupt Control Port (Port 6)

Bit 0 - External Interrupt Enable
Bit 1 - Timer Interrupt Enable
Bit 2 - EXT INT Active Level
Bit 3 - Start/Stop Timer
Bit 4 - Pulse Width/Interval Timer
Bit 5 - ÷ 2 Prescale
Bit 6 - ÷ 5 Prescale
Bit 7 - ÷ 20 Prescale

A special situation exists when reading the Interrupt Control Port (with an IN or INS instruction). The Accumulator is <u>not</u> loaded with the content of the ICP; instead, Accumulator bits 0 through 6 are loaded with 0's while bit 7 is loaded with the logic level being applied to the EXT INT pin, thus allowing the status of EXT INT to be determined without the necessity of servicing an external interrupt request. When reading the Interrupt Control Port (Port 6) bit 7 of the Accumulator is loaded with the actual logic level being applied to the EXT INT pin, regardless of the status of ICP bit 2 (the EXT INT Active Level bit); that is, if EXT INT is at +5V bit 7 of the Accumulator is set to a logic 1, but if EXT INT is at GND then Accumulator bit 7 is reset to logic 0. This capability is useful in establishing a high speed polled handshake procedure or for using EXT INT as an extra input pin if external interrupts are not required and the Timer is used only in the Interval Timer Mode. However, if it is desirable to read the contents of the ICP then one of the 64 scratchpad registers or one byte of RAM may be used to save a copy of whatever is written to the ICP.

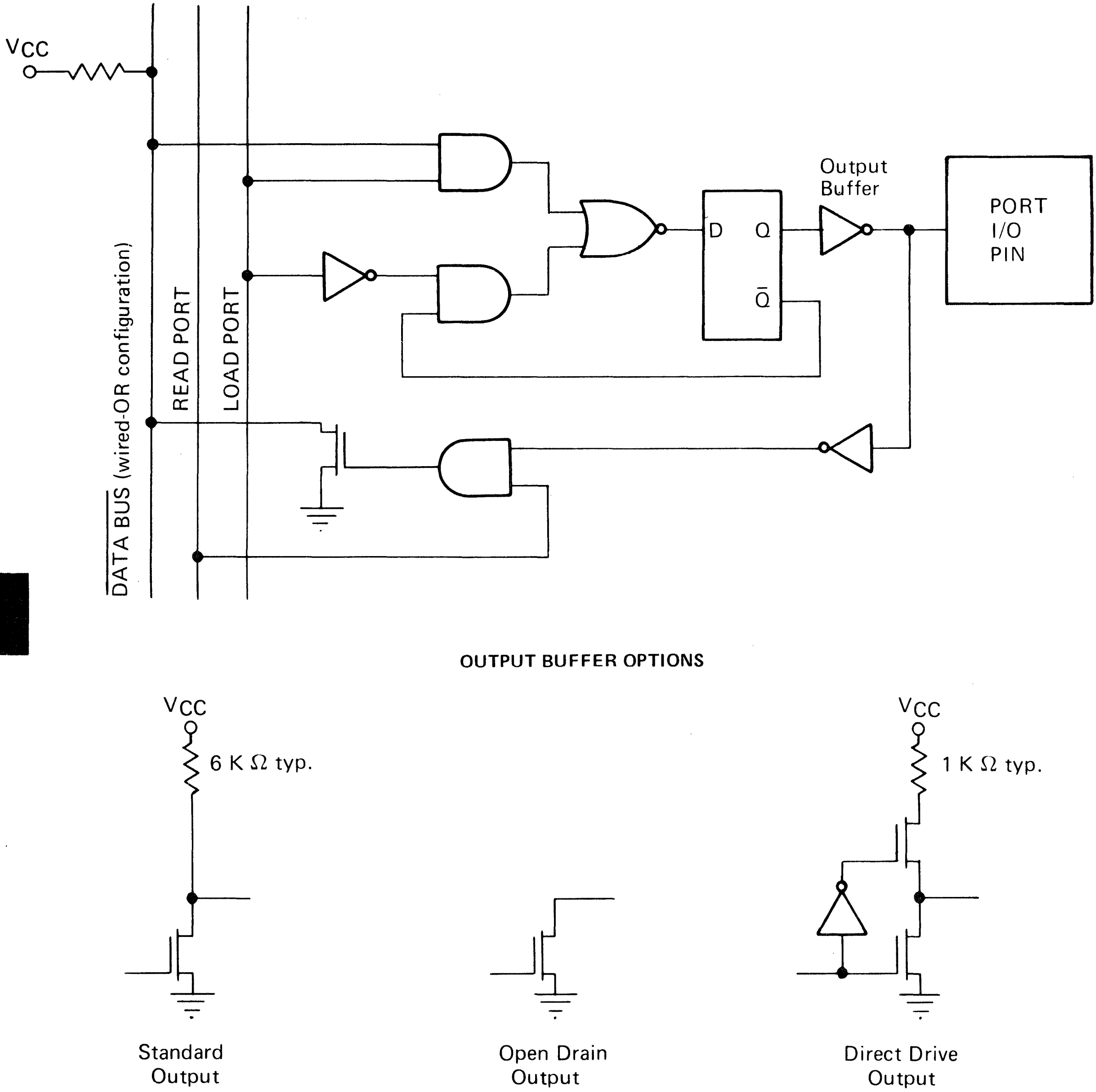
The rate at which the timer is clocked in the Interval Timer Mode is determined by the frequency of an internal Φ clock and by the division value selected for the prescaler. (The internal Φ clock operates at one-half the external time base frequency). If ICP bit 5 is set and bits 6 and 7 are cleared, the prescaler divides Φ by 2. Likewise, if bit 6 or 7 is individually set the prescaler divides Φ by 5 or 20 respectively. Combinations of bits 5, 6 and 7 may also be selected. For example, if bits 5 and 7 are set while 6 is cleared the prescaler will divide by 40. Thus possible prescaler values are ÷2, ÷5, ÷10, ÷20, ÷40, ÷100, and ÷200.

Any of three conditions will cause the prescaler to be reset: whenever the timer is stopped by clearing ICP bit 3, execution of an output instruction to Port 7, (the timer is assigned port address 7), or on the trailing edge transition of the EXT INT pin when in the Pulse Width Measurement Mode. These last two conditions are explained in more detail below.

An OUT or OUTS instruction to Port 7 will load the content of the Accumulator to both the Timer and the 8-bit modulo-N register, reset the prescaler, and

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS

Figure 3



Ports 0 and 1 are Standard Output type only.

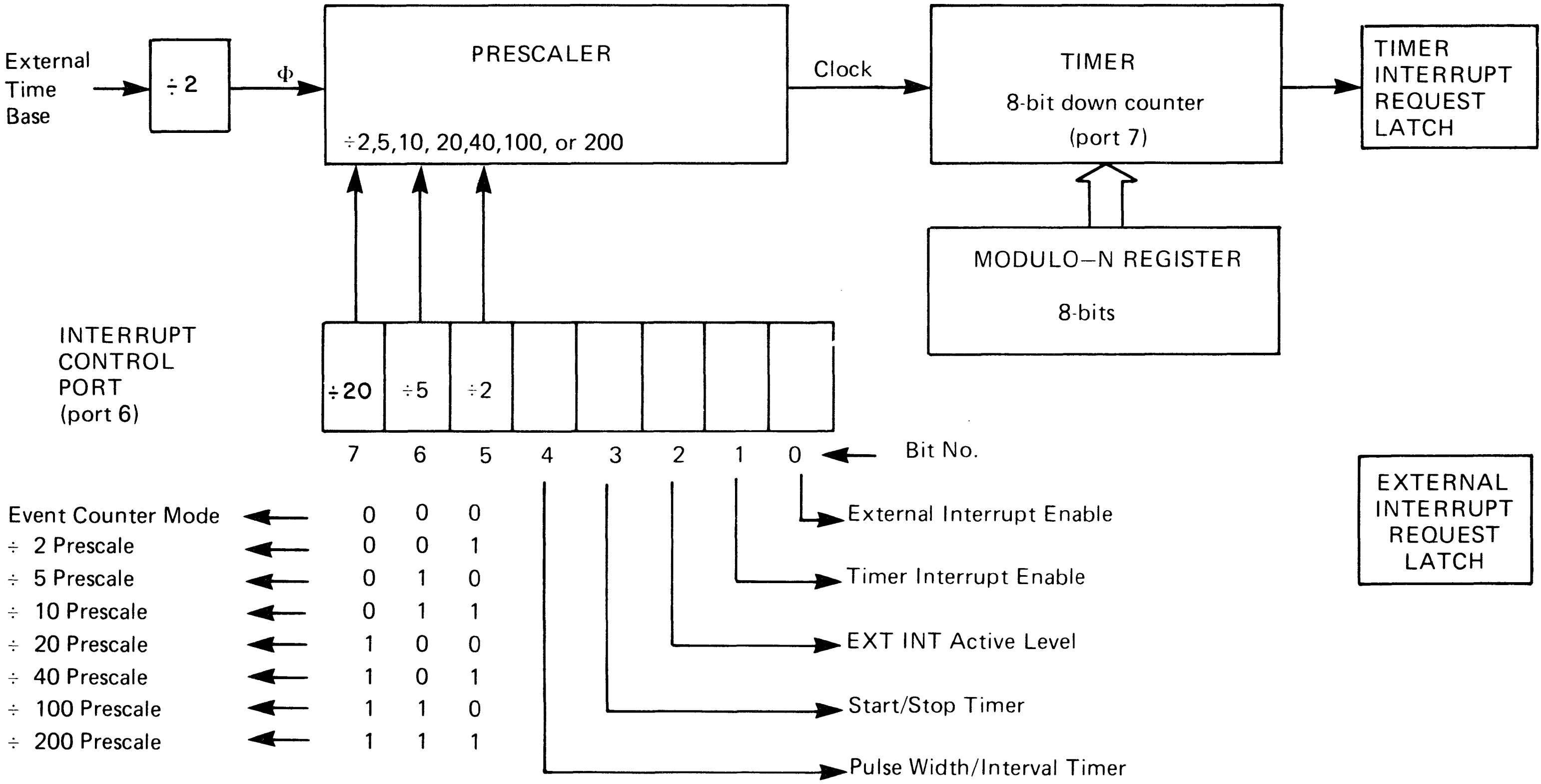
Ports 4 and 5 may both be any of the three output options (programmable bit by bit).

The $\overline{\text{STROBE}}$ output is always configured similar to a Direct Drive Output except that it is capable of driving 3 TTL loads.

$\overline{\text{RESET}}$ and EXT INT may have standard 6KΩ (typical) pull-up or may have no pull-up.
These two inputs have Schmidt trigger inputs with a minimum of 0.2 volts of hysteresis.

TIMER & CONTROL PORT BLOCK DIAGRAM

Figure 4

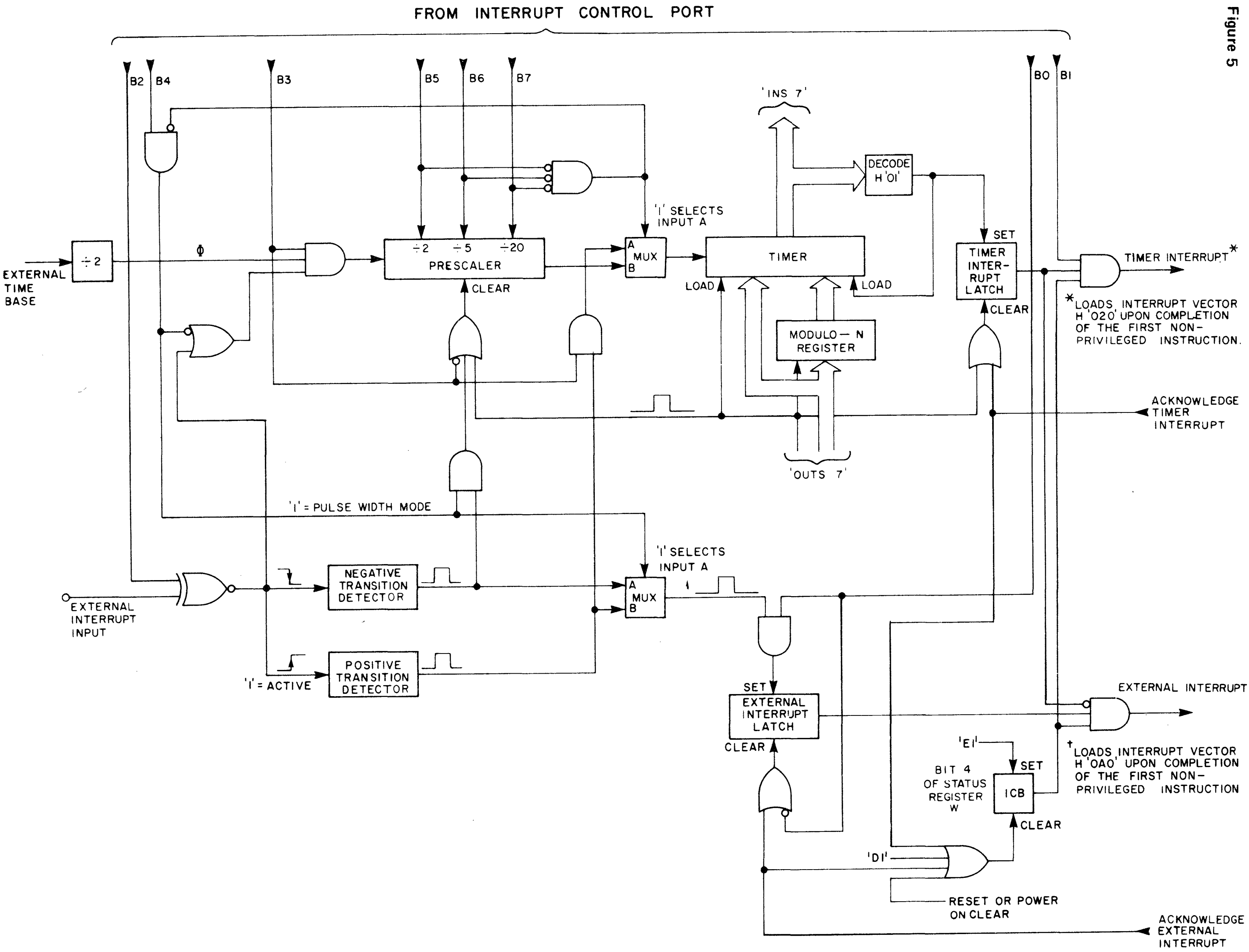


	7	6	5
Event Counter Mode	0	0	0
÷ 2 Prescale	0	0	1
÷ 5 Prescale	0	1	0
÷ 10 Prescale	0	1	1
÷ 20 Prescale	1	0	0
÷ 40 Prescale	1	0	1
÷ 100 Prescale	1	1	0
÷ 200 Prescale	1	1	1

Note: See Fiqure 5 for a more detailed functional diagram.

MK3870 TIMER/INTERRUPT FUNCTIONAL DIAGRAM

Figure 5



*LOADS INTERRUPT VECTOR H'020' UPON COMPLETION OF THE FIRST NON-PRIVILEGED INSTRUCTION.

†LOADS INTERRUPT VECTOR H'0A0' UPON COMPLETION OF THE FIRST NON-PRIVILEGED INSTRUCTION

clear any previously stored timer interrupt request. As previously noted, the Timer is an 8-bit down counter which is clocked by the prescaler in the Interval Timer Mode and in the Pulse Width Measurement Mode. The prescaler is not used in the Event Counter Mode. The Modulo-N register is a buffer whose function is to save the value which was most recently outputted to Port 7. The modulo-N register is used in all three timer modes.

Interval Timer Mode

When ICP bit 4 is cleared (logic 0) and at least one prescale bit is set the Timer operates in the Interval Timer Mode. When bit 3 of the ICP is set the Timer will start counting down from the modulo-N value. After counting down to H'01', the Timer returns to the modulo-N value at the next count. On the transition from H'01' to H 'N' the Timer sets a timer interrupt request latch. Note that the interrupt request latch is set by the transition to H 'N' and not be the presence of H 'N' in the Timer, thus allowing a full 256 counts if the modulo-N register is preset to H '00'. If bit 1 of the ICP is set, the interrupt request is passed on to the CPU section of the 3870. However, if bit 1 of the ICP is a logic 0 the interrupt request is not passed on to the CPU section but the interrupt request latch remains set. If ICP bit 1 is subsequently set, the interrupt request will then be passed on to the CPU section. (Recall from the discussion of the Status Register's Interrupt Control Bit that the interrupt request will be acknowledged by the CPU section only if ICB is set). Only two events can reset the timer interrupt request latch; when the timer interrupt request latch is acknowledged by the CPU section, or when a new load of the modulo-N register is performed.

Consider an example in which the modulo-N register is loaded with H '64' (decimal 100). The timer interrupt request latch will be set at the 100th count following the timer start and the timer interrupt request latch will repeatedly be set on precise 100 count intervals. If the prescaler is set at ÷40 the timer interrupt request latch will be set every 4000 Φ clock periods. For a 2MHz Φ clock (4MHz time base frequency) this will produce 2 millisecond intervals.

The range of possible intervals is from 2 to 51,200 Φ clock periods (1μs to 25.6ms for a 2MHzΦ clock). However, approximately 50 Φ periods is a practical minimum because the time between setting the interrupt request latch and the execution of the first instruction of the interrupt service routine is at least 29 Φ periods (the response time is dependent upon how many privileged instructions are encountered when the request occurs); 29 is based on the timer interrupt occuring at the beginning of a non-priviledged short instruction. To establish time intervals greater than 51,200 Φ clock periods is a simple matter of using the timer interrupt service routine to count the number of interrupts, saving the result in one or more of the scratchpad registers until the desired interval is achieved. With this technique virtually any time interval, or several time intervals, may be generated.

The Timer may be read at any time and in any mode using an input instruction (IN 7 or INS 7) and may take place "on the fly" without interfering with normal timer operation. Also, the Timer may be stopped at any time by clearing bit 3 of the ICP. The Timer will hold its current contents indefinitely and will resume counting when bit 3 is again set. Recall however that the prescaler is reset whenever the Timer is stopped; thus a series of starting and stopping will result in a cumulative truncation error.

A summary of other timer errors is given in the timing section of this specification. For a free running timer in the Interval Timer Mode the time interval between any two interrupt requests may be in error by ± 6 Φ clock periods although the cumulative error over many intervals is zero. The prescaler and Timer generate precise intervals for setting the timer interrupt request latch but the time out may occur at any time withi n a machine cycle. (There are two types of machine cycles; short cycles which consist of 4 Φ clock periods and long cycles which consist of 6 Φ clock periods. In the multi-chip F8 family there is a signal called the WRITE clock which corresponds to a machine cycle). Interrupt requests are synchronized with the internal WRITE clock thus giving rise to the possible ± 6 Φ error. Additional errors may arise due to the interrupt request occuring while a privileged instruction or multicycle instruction is being executed. Nevertheless, for most applications all of the above errors are negligible, especially if the desired time interval is greater than 1 ms.

Pulse Width Measurement Mode

When ICP bit 4 is set (logic 1) and at least one prescale bit is set the Timer operates in the Pulse Width Measurement Mode. This mode is used for accurately measuring the duration of a pulse applied to the EXT INT pin. The Timer is stopped and the prescaler is reset whenever EXT INT is at its inactive level. The active level of EXT INT is defined by ICP bit 2; if cleared, EXT INT is active low; if set, EXT INT is active high. If ICP bit 3 is set, the prescaler and Timer will start counting when EXT INT transitions to the active level. When EXT INT returns to the inactive level the Timer then stops, the prescaler resets, and _if ICP bit 0_ is set an external interrupt request latch is set. (Unlike timer interrupts, external interrupts are not latched if the ICP Interrupt Enable bit is not set).

As in the Interval Timer Mode, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, the prescaler and ICP bit 1 function as previously described, and the Timer still functions as an 8-bit binary down counter with the timer interrupt request latch being set on the Timer's transition from H '01' to H 'N'. Note that the EXT INT pin has nothing to do with loading the Timer; its action is that of automatically starting and stopping the Timer and of generating external interrupts. Pulse widths longer than the prescale value times the modulo-N value are easily measured by using the timer interrupt service routine to store the number of timer interrupts in one or more scratchpad registers.

As for accuracy, the actual pulse duration is typically slightly longer than the measured value because the status of the prescaler is not readable and is reset when the Timer is stopped. Thus for maximum accuracy it is advisable to use a small division setting for the prescaler.

Event Counter Mode

When ICP bit 4 is cleared and all prescale bits (ICP bits 5, 6, and 7) are cleared the Timer operates in the Event Counter Mode. This mode is used for counting pulses applied to the EXT INT pin. If ICP bit 3 is set the Timer will decrement on each transition from the inactive level to the active level of the EXT INT pin. The prescaler is not used in this mode; but as in the other two timer modes, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, ICP bit 1 functions previously described, and the timer interrupt request latch is set on the Timer's transition from H '01' to H 'N'.

Normally ICP bit 0 should be kept cleared in the Event Counter Mode; otherwise, external interrupts will be generated on the transition from the inactive level to the active level of the EXT INT pin.

For the Event Counter Mode the minimum pulse width required on EXT INT is 2 Φ clock periods and the minimum inactive time is 2 Φ clock periods; therefore, the maximum repetition rate is 500 KHz.

Timer Emulation

For total software compatibility when expanding into a multi-chip configuration the MK3871 Peripheral Input/Output circuit should be used rather than the older MK3861 PIO. The MK3871 has the same improved Timer (binary count, readable, and three modes of operation rather than one) and ready strobe output as are on the MK3870.

External Interrupts

When the timer is in the Interval Timer Mode the EXT INT pin is available for non-timer related interrupts. If ICP bit 0 is set an external interrupt request latch is set when there is a transition from the inactive level to the active level of EXT INT. (EXT INT is an edge-triggered input). The interrupt request is latched until either acknowledged by the CPU section or until ICP bit 0 is cleared (unlike timer interrupt requests which remain latched even when ICP bit 1 is cleared). External interrupts are handled in the same fashion when the Timer is in the Pulse Width Measurement Mode or in the Event Counter Mode, except that only in the Pulse Width Measurement Mode the external interrupt request latch is set on the trailing edge of EXT INT; that is, on the transition from the active level to the inactive level.

Interrupt Handling

When either a timer or an external interrupt request is communicated to the CPU section of the 3870, it will be acknowledged and processed at the completion of the first non-privileged instruction if the Interrupt Control Bit of the Status Register is set. If the Interrupt Control Bit is not set, the interrupt request will continue until either the Interrupt Control Bit is set and the CPU section acknowledges the interrupt or until the interrupt request is cleared as previously described.

If there is both a timer interrupt request and an external interrupt request when the CPU section starts to process the requests, the timer interrupt is handled first.

When an interrupt is allowed the CPU section will request that the interrupting element pass its interrupt vector address to the Program Counter via the data bus. The vector address for a timer interrupt is H '020'. The vector address for external interrupts is H '0A0'. After the vector address is passed to the Program Counter, the CPU section sends an acknowledge signal to the appropriate interrupt request latch which clears that latch. The exection of the interrupt service routine will then commence. The return address of the original program is automatically saved in the Stack Register, P.

The Interrupt Control Bit of W (Status Register) is automatically reset when an interrupt request is acknowledged. It is then the programmer's responsibility to determine when ICB will again be set (by executing an EI instruction). This action prevents an interrupt service routine from being interrupted unless the programmer so desires.

Figure 6 details the interrupt sequence which occurs whether the interrupt request is from an external source via EXT INT or from the 3870's internal timer. Events are labeled with the letters A through G and are described below.

Event A

An interrupt request must satisfy a hold time requirement as specified in the AC Characteristics in order to guarantee that it is valid on the rising edge of the WRITE clock.

Event B

Event B represents the instruction being executed when the interrupt occurs. The last cycle of B is normally the instruction fetch for the next cycle. However, if B is not a privileged instruction and the CPU's Interrupt Control Bit is set, then the last cycle becomes a "freeze" cycle rather than a fetch. At the end of the freeze cycle the interrupt request latches are inhibited from altering the interrupt daisy-chain so that sufficient time will be allowed for the daisy-chain to settle. (If B is a privileged instruciton, the instruction fetch is not replaced by a freeze cycle; instead, the fetch is performed and the next instruction is executed. Although unlikely to be encountered, a series of privileged instructions will be sequentially executed without interrupt. One more instruction, called a 'protected' instruction, will always be executed after the last privileged instruction. The last cycle of the protected instruction then performs the freeze.)

The dashed lines on EXT INT illustrate the last opportunity for EXT INT to cause the last cycle of a non-protected instruction to become a freeze cycle.

The freeze cycle is a short cycle (4 Φ clock periods) in all cases except where B is the Decrement Scratchpad instruction, in which case the freeze cycle is a long cycle (6 Φ clock periods).

INT REQ goes low on the next negative edge of WRITE if both PRI IN is low and the appropriate interrupt enable bit of the Interrupt Control Part is set. Both INT REQ and WRITE are internal signals.

Event C

A NO-OP long cycle to allow time for the internal priority chain to settle.

Event D

The program counter (PO) is pushed to the stack register (P) in order to save the return address. The interrupt circuitry places the lower 8 bits of the interrupt vector address onto the data bus. This is always a long cycle.

Event E

A long cycle in which the interrupt circuitry places the upper 8 bits of the interrupt vector address onto the data bus.

Event E

A long cycle in which the interrupt circuitry places the upper 8 bits of the interrupt vector address onto the data bus.

Event F

A short cycle in which the interrupting interrupt request latch is cleared. Also, the CPU's Interrupt Control Bit is cleared, thus disabling interrupts until an EI instruction is performed. The fetch of the next instruction from the interrupt address.

Event G

Begin execution of the first instruction of the interrupt service routine.

Summary Of Interrupt Sequence

For the MK3870 the interrupt response time is defined as the time elapsed between the occurence of EXT INT going active (or the Timer transitioning to H'N') and the beginning of execution of the first instruction of the interrupt service routine. The interrupt response time is a variable depedent upon what the microprocessor is doing when the interrupt request occurs. As shown in Figure 5, the minimum interrupt response time is 3 long cycles plus 2 short cycles plus one WRITE clock pulse width plus a setup time of EXT INT prior to the leading edge of the WRITE pulse – a total of 27 Φ clock periods plus the setup time. At a 2 MHz Φ this is 14.25 μs. Although the maximum could theoretically be infinite, a practical maximum is 35 μs (based on the interrupt request occurring near the beginning of a PI and LR K, P sequence).

Power-On Clear

The intent of the Power-On-Reset circuitry on the 3870 is to automatically reset the device following a typical power-up situation, thus saving external reset circuitry in many applications. This circuitry is not guaranteed to sense a "Brown Out" (low voltage) condition nor is it guaranteed to operate under all possible power-on situations.

Three conditions are required before the 3870 will leave the reset state and begin operation. Refer to Figure 7 as an aid to the following descriptions. The On-Chip Vcc detector senses a minimum value of Vcc before it will allow the 3870 to operate. The threshold of this detector is set by analog circuitry because a stable voltage reference is not available with n-channel MOS processing. Processing variations will cause this threshold to vary from a low of 3.0 volts to a high of 4.3 volts with 3.5 volts being typical.

The 3870 uses a substrate bias as a technique to provide improved performance verses power consumption relative to conventional grounded substrate approaches. This bias generator may start operating as low as Vcc = 3 volts on some devices while others may require Vcc = 4 volts in order to get adequate substrate bias. Until the substrate reaches the proper bias, the 3870 will not be released from the reset state. The final condition required is that the clocks of the 3870 must be functioning. Typically the clocks will start to function at Vcc equal to 3 to 3.5 volts but since the part is tested at 4.5 volts MOSTEK cannot guarantee any operation below 4.5 volts. The output of the delay circuit in Figure 7 will stay low until the clocks start to function. If the input to the delay circuit is high, typically after 100 cycles of the WRITE clock (800 cycles of the external clock) the output of the delay circuit will go high allowing the 3870 to begin execution.

If Vcc falls to ground for at least a few hundred nanoseconds the output of the delay circuit will go low immediately and the 3870 will reset.

The internal logic may detect a valid Vcc, bias and clocks at Vcc = 3.5 volts and allow the 3870 to start executing after the time delay. With a slowly rising power supply the part may start running before Vcc is above 4.5 volts which is below the guaranteed voltage range. When power-on-clear is required with a slowly rising power supply, an external capacitor must be used on the $\overline{\text{RESET}}$ pin to hold it below 0.8 volts until Vcc is stable above 4.5 volts. (Note: The option to disconnect the internal pull-up resistor on $\overline{\text{RESET}}$ is available which allows the use of a larger external pull-up resistor and a small capacitor on $\overline{\text{RESET}}$.)

In many applications, it is desirable if the unit does an automatic power-on-clear, but not mandatory. The unit will have a RESET push button and if the unit does not power-up correctly or malfuctions because of some disturbance on the Vcc line, the operator will simply press RESET and restore normal operation. It is for these applications that the internal power-on-clear circuitry was designed.

In some applications it is required that the microcomputer continue to run properly without operator intervention after brown-outs, power line disturbances, electrical noise, computer malfunction due to a programming bug or any other disturbance except a catastrophic failure of some component.

Once concept used to keep computers running is that of the "WATCHDOG TIMER". The computer is programmed to periodically reset the watchdog timer during the normal execution of its program (this is easily done in the 3870 as its normal application is in some control function which is typically periodic). As long as the computer continues to execute its program the watchdog timer is continually reset and never times out. Should the computer stop executing its program for whatever reason, the watchdog timer will time out producing a RESET pulse to the CPU re-starting execution. This is a very positive way to assure that the computer is doing its job, i.e., executing the program. It is important that the software driving the watchdog timer test as many functional blocks (timer, ALU, scratchpad RAM, and Ports) of the 3870 as possible before reseting the watchdog timer. This is because operation of the 3870 with an out of spec power supply may allow some of the functions to operate correctly while other functions are not operable.

MOSTEK can guarantee correct operation of the 3870 only while the Vcc voltage remains within its specified limits. If proper operation of the 3870 must be guaranteed after a disturbance on the Vcc line, then an external circuit must be used to monitor the Vcc line and produce a $\overline{\text{RESET}}$ to the 3870 whenever Vcc is out of the specified limits.

A related characteristic to power-on-clear is the Startup time of the basic timing element. The LC, and RC, oscillators begin to function almost immediately once Vcc is high enough to allow the on-board oscillator to operate (Vcc = 3.5). Operation with a crystal is partly mechanical and some start time is required to get the mass of the crystal into vibrational motion. This time is basically dependent on the frequency (mass) of the crystal. 4 MHz crystals typically require about 2-3 mSec to start while 1 MHz crystals require 60-70 mSec to start oscillating. Of course, this time may vary greatly from crystal to crystal and is also a function of the power supply rise time characteristic, however, the high frequency crystals start faster and are definately recommended (i.e., 3-4 MHz).

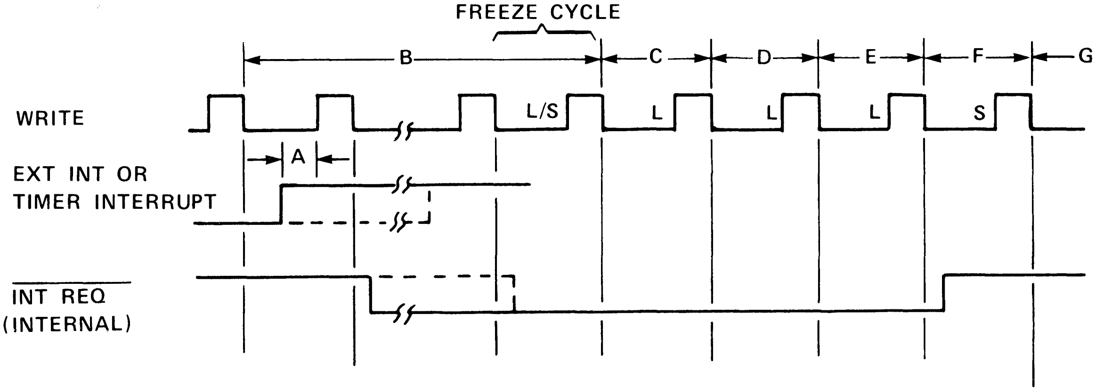
The condition of the port pins during the power-on-clear sequence is often asked. The port pins or the STROBE line cannot be specified until Vcc reaches 4.5V and the 3870 enters the RESET state. Before this, the port pins may stay at Vss, may track Vcc as it rises, or they may track Vcc part way up then return to Vss (Ports 4 & 5 will go to Vcc once the clocks are running and the 3870 has sufficient Vcc to properly operate the internal control logic and I/O ports).

External Reset

When $\overline{\text{RESET}}$ is taken low the content of the Program Counter is pushed to the Stack Register and then the Program Counter and the ICB bit of the W Status Register are cleared. The original Stack Register content is lost. Ports 4, 5, 6 and 7 are loaded

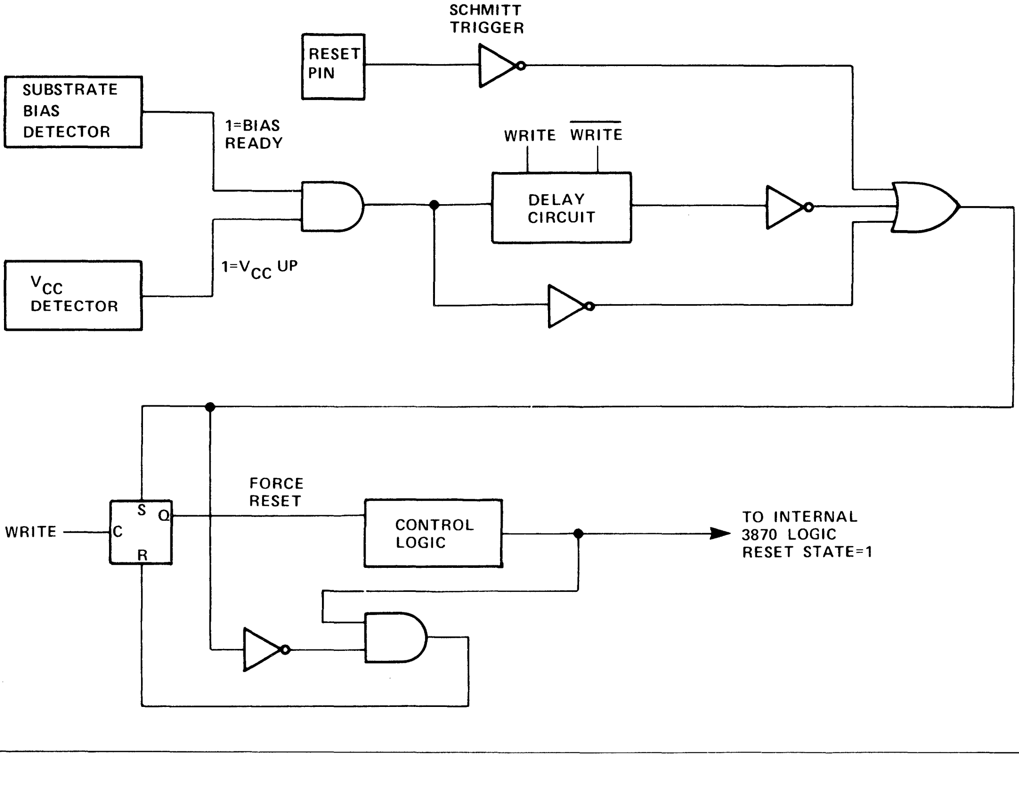
INTERRUPT SEQUENCE

Figure 6



POWER ON CLEAR BLOCK DIAGRAM

Figure 7



with H '00'. The contents of all other registers and ports are unchanged or undefined. When RESET is taken high the first program instruction is fetched from ROM location H'000'. When an external reset of the 3870 occurs, P0 is pushed into P and the old contents of P are lost. It must be noted that an external reset is recognized at the start of a machine cycle and not necessarily at the end of an instruction. Thus if the 3870 is executing a multicycle instruction, that instruction is not completed and the contents of P upon reset may not necessarily be the address of the instruction that would have been executed next. It may, for example, point to an immediate operand if the reset occurred during the second cycle of a LI or CI instruction. Additionally, several instructions (JMP, PI, PK, LR P0, Q) as well as the interrupt acknowledge sequence modify P0 in parts. That is, they alter P0 by first loading one part then the other and the entire operation takes more than one cycle. Should reset occur during this modification process the value pushed into P will be part of the old P0 (the as yet unmodified part) and part of the new P0 (already modified part). Thus care should be taken (perhaps by external gating) to insure that reset does not occur at an undesirable time if any significance is to be given to the contents of P after a reset occurs.

Vcc Decoupling

The 3870 family devices have dynamic circuitry internally which requires a good high frequency decoupling capacitor to surpress noise on the Vcc line. A .01 μF or .1 μF ceramic capacitor should be placed between Vcc and ground, located physically close to the 3870 device. This will reduce noise generated by the 3870 to about 70-100mVolts on the Vcc line.

Test Logic

Special test logic is implemented to allow access to the internal main data bus for test purposes.

In normal operation the TEST pin is unconnected or is connected to GND. When TEST is placed at a TTL level (2.0V to 2.6V) port 4 becomes an output of the internal data bus and port 5 becomes a wired-OR input to the internal data bus. The data appearing on the port 4 pins is logically true whereas input data forced on port 5 must be logically false. When TEST is placed at a high level (6.0V to 7.0V), the ports act as above and additionally the 2K x 8 program ROM is prevented from driving the data bus. In this mode operands and instructions may be forced externally through port 5 instead of being accessed from the program ROM. When TEST is in either the TTL state or the high state, $\overline{\text{STROBE}}$ ceases its normal function and becomes a machine cycle clock (identical to the F8 multi-chip system WRITE clock except inverted).

Timing complexities render the capabilities associated with the TEST pin impractical for use in a user's application, but these capabilities are thoroughly sufficient to provide a rapid method for thoroughly testing the 3870.

3870 Clocks

The time base for the 3870 may originate from one of four sources.

The four configurations are shown in Figure 8. There is an internal 26pF capacitor between XTL 1 and GND and an internal 26pF capacitor between XTL 2 and GND. Thus external capacitors are not neccesarily required. In all external clock modes the external time base frequently is divided by two to form the internal Φ clock.

Crystal Selection

The use of a crystal as the time base is highly recommended as the frequency stability and reproducability from system to system is unsurpassed. The 3870 has an internal divide by two to allow the user of inexpensive and widely available TV Color Burst Crystals (3.58MHz). The following crystal parameters and vendors are suggested for 3870 applications:

Parameters

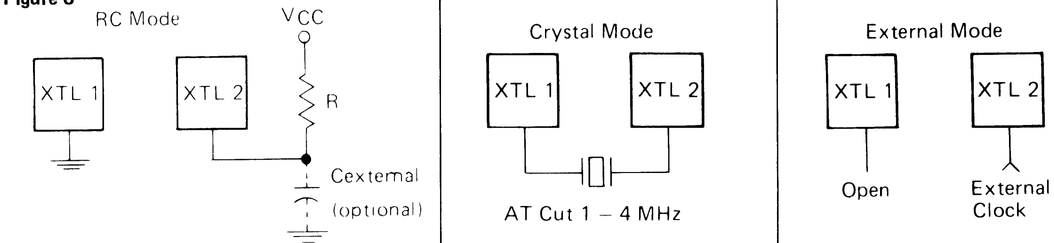
a) Parallel Resonance, Fundamental Mode AT-CUT
b) Frequency Tolerance measured with 18pF load (0.1% accuracy). Drive level 10mW.
c) Shunt Capacitance (Co) = 7pF max.
d) Series Resistance (Rs)

		Holder
f = 1MHz	Rs = 550 ohms max.	HC-6
f = 2MHz	Rs = 300 ohms max.	HC-33
f = 3MHz	Rs = 150 ohms max.*	HC-6
f = 3.58MHz	Rs = 150 ohms max.	HC-18
f = 4MHz	Rs = 150 ohms max.	HC-25 HC-33

*HC-18 or HC-25 may not be available at 3MHz.

CLOCK CONFIGURATION

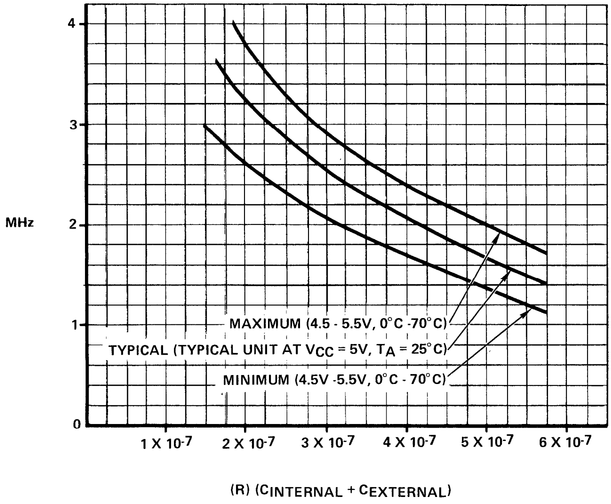
Figure 8



Minimum R = 4K Ω

C = 26.5 pF ± 2.6pF + Cexternal

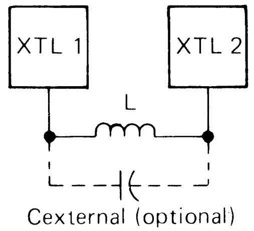
FREQUENCY VRS RC



UNIT TO UNIT VARIATION = ± 12%
VARIATION FROM 4.5 to 5.5V
REFERENCED TO 5V = +7% -4%
VARIATION FROM 0°C TO 70°C
REFERENCED TO 25°C = +6% -9%

TOTAL VARIATION NOT CONSIDERING
VARIATION IN EXTERNAL COMPONENTS = ± 25%

LC Mode



Minimum L = 0.1 mH
Minimum Q = 40

Maximum Cexternal = 30pF

C = 13pF ± 1.3pF + Cexternal

$$f \quad \frac{1}{2\pi\sqrt{LC}}$$

Suggested Crystal Vendors

a) Electro-Dynamics
5625 Foxridge Drive
Mission, Kansas 66201
913-262-2500

b) CRYSTEK
1000 Crystal Drive
Ft. Myers, Florida 33901
813-936-2109

c) W.T. Liggett Corp.
1500 Worcester Rd.
Section 30
Framingham, MA 01701
617-620-1150

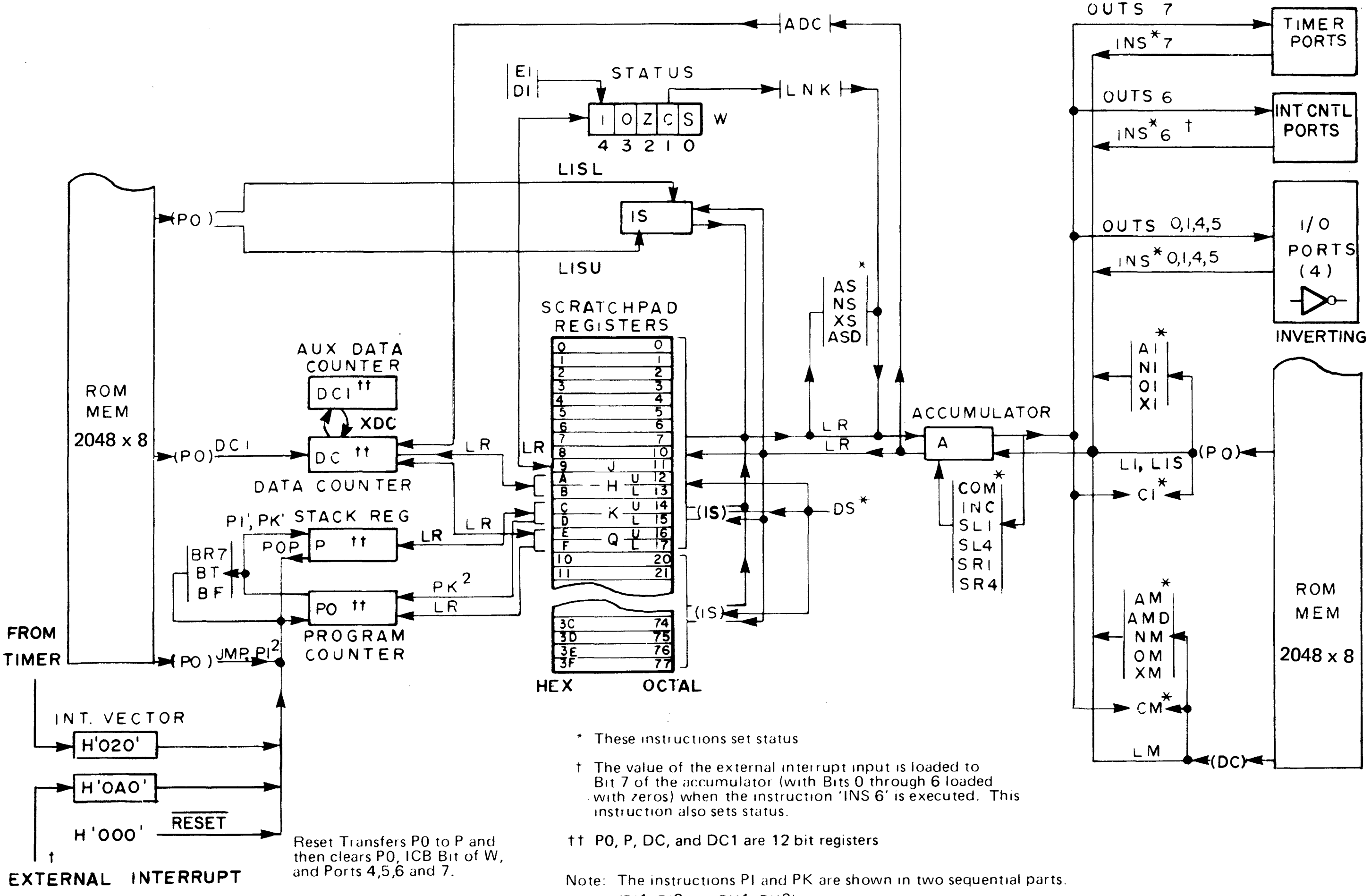
d) Erie Frequency Control
453 Lincoln Street
Carlisle, Penn 17013
717-249-2232

e) Electronic Crystals Corp.
1153 Southwest Blvd.
Kansas City, Kansas 66103
913-262-1274

f) M-TRON Industries
P.O. Box 630
100 Douglas Avenue
Yankton, South Dakota
605-665-9321

MK3870 PROGRAMMING MODEL

Figure 9



Reset Transfers P0 to P and then clears P0, ICB Bit of W, and Ports 4,5,6 and 7.

* These instructions set status

† The value of the external interrupt input is loaded to Bit 7 of the accumulator (with Bits 0 through 6 loaded with zeros) when the instruction 'INS 6' is executed. This instruction also sets status.

†† P0, P, DC, and DC1 are 12 bit registers

Note: The instructions PI and PK are shown in two sequential parts. (PI1, PI2 and PK1, PK2).

INSTRUCTION EXECUTION

This section details the timing and execution of the 3870 instruction set. The 3870 executes the entire F8 instruction set with exact F8 timing. Refer to Figure 11 for a 3870 Programming Model.

F8 INSTRUCTION SET

ACCUMULATOR GROUP INSTRUCTIONS

	MNEMONIC			MACHINE		CYCLES		μS		STATUS BITS		
OPERATION	OP CODE	OPERAND	FUNCTION	CODE	BYTES	SHORT	LONG	(2MHzΦ)	OVR	ZERO	CRY	SIGN
Add Carry	LNK		A←(A) + CRY	19	1	1		2	1/0	1/0	1/0	1/0
Add Immediate	AI	ii	A←(A) + H 'ii'	24ii	2	1	1	5	1/0	1/0	1/0	1/0
And Immediate	NI	ii	A←(A)∧H 'ii'	21ii	2	1	1	5	0	1/0	0	1/0
Clear	CLR		A←H'00'	70	1	1		2	–	–	–	–
Compare Immediate	CI	ii	H'ii'+ (A) + 1	25ii	2	1	1	5	1/0	1/0	1/0	1/0
Complement	COM		A←(A) + H'FF'	18	1	1		2	0	1/0	0	1/0
Exclusive or Immediate	XI	ii	A←(A) + H'ii'	23ii	2	1	1	5	0	1/0	0	1/0
Increment	INC		A←(A) + 1	1F	1	1		2	1/0	1/0	1/0	1/0
Load Immediate	LI	ii	A←H'ii'	20ii	2	1	1	5	–	–	–	–
Load Immediate Short	LIS	i	A←H' 0i'	7i	1	1		2				
OR Immediate	OI	ii	A←(A) V H 'ii'	22ii	2	1	1	5	0	1/0	0	1/0
Shift Left One	SL	1	Shift Left 1	13	1	1		2	0	1/0	0	1/0
Shift Left Four	SL	4	Shift Left 4	15	1	1		2	0	1/0	0	1/0
Shift Right One	SR	1	Shift Right 1	12	1	1		2	0	1/0	0	1
Shift Right Four	SR	4	Shift Right 4	14	1	1		2	0	1/0	0	1

BRANCH INSTRUCTIONS In all conditional branches P0←(P0) + 2 if the test condition is not met. Execution is complete in 3 short cycles.

	MNEMONIC			MACHINE		CYCLES		μS		STATUS BITS		
OPERATION	OP CODE	OPERAND	FUNCTION	CODE	BYTES	SHORT	LONG	(2MHzΦ)	OVR	ZERO	CRY	SIGN
Branch on Carry	BC	aa	P0←(P0) +1+ H'aa' if CRY 1	82aa	2	2	1	7	—	—	—	—
Branch on Positive	BP	aa	P0←(P0) + 1 + H'aa ' if SIGN 1	81aa	2	2	1	7	—	—	—	—
Branch on Zero	BZ	aa	P0←(P0) + 1 + H'aa' if Zero 1	84aa	2	2	1	7	—	—	—	—
Branch on True	BT t TEST CONDITION 2^2 2 2^0 ZERO CRY SIGN	taa	P0←(P0) + 1 + H'aa ' if any test is true	8taa	2	2	1	7	—	—	—	—
Branch If Negative	8M	aa	P0←(P0) +1+ H'aa ' if SIGN 0	91aa	2	2	1	7	—	—	—	—
Branch if No Carry	BNC	aa	P0←(P0) +1+ H'aa ' if CARRY 0	92aa	2	2	1	7	—	—	—	—
Branch if No Overflow	BNO	aa	P0←(P0) +1+ H'aa ' if OVR 0	98aa	2	2	1	7	—	—	—	—
Branch if Not Zero	BNZ	aa	P0←(P0) +1+ H'aa ' if ZERO 0	94aa	2	2	1	7	—	—	—	—
Branch if False Test	BF t TEST CONDITION 2^3 2^2 2^1 2^0 OVF ZERO CRY SIGN	taa	P0←(P0) +1+ H'aa ' if all false test bits	9taa	2	2	1	7	—	—	—	—
Branch if ISAR (Lower) ≠7	BR7	aa	P0←(P0)+1+ H'aa ' if ISARL ≠7	8Faa	2	1	1	5	—	—	—	—
			P0←(P0)+2 if ISARL =		2	2		4	—	—	—	—
Branch Relative	BR	aa	P0←(P0)+1+ H'aa '	90aa	2	2	1	7	—	—	—	—
Jump	JMP	aaaa	P0←H'aaaa'	29aaaa	3	1	3	11	—	—	—	—

*Privileged instruction, Accumulator contents altered during execution JMP

MEMORY REFERENCE INSTRUCTIONS In all Memory Reference Instructions, the Data Counter is incremented DC ←(DC)+1

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	 LONG	µS (2MHzΦ)	 OVR	STATUS BITS ZERO	 CRY	 SIGN
Add Binary	AM		A←(A) + [(DC)]	88	1	1	1	5	1/0	1/0	1/0	1/0
Add Decimal	AMD		A←(A) + [(DC)] • BCD Adjust	89	1	1	1	5	?	?	1/0	?
AND	NM		A←(A) ∧ [(DC)]	8A	1	1	1	5	0	1/0	0	1/0
Compare	CM		[(DC)] + $(\overline{A})$ + 1	8D	1	1	1	5	1/0	1/0	1/0	1/0
Exclusive OR	XM		A←(A)⊕[(DC)]	8C	1	1	1	5	0	1/0	0	1/0
Load	LM		A←[(DC)]	16	1	1	1	5	—	—	—	—
Logical OR	OM		A← (A) V '(DC)]	8B	1	1	1	5	0	1/0	0	1/0
Store	ST		A→[(DC)]	17	1	1	1	5	—	—	—	—

ADDRESS REGISTER GROUP INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	 LONG	µS (2MHzΦ)	 OVR	STATUS BITS ZERO	 CRY	 SIGN
Add to Data Counter	ADC		DC←(DC) + (A)	8E	1	1	1	5	–	–	–	–
Call to Subroutine*	PK		P0U←(r12); P0L←(r13), P←(P0)	0C	1	1	2	8	—	—	—	—
Call to Subroutine Immediate*	PI	aaaa	P← (P0), P0←H'aaaa	28aaaa	3	2	3	13	–	—	—	—
Exchange DC	XDC		(DC)⇄(DC1)	2C	1	2		4	–	–	–	–
Load Data Counter	LR	DC,Q	DCU←(r14), DCL←(r15)	0F	1	1	2	8	—	—	—	—
Load Data Counter	LR	DC'H	DCU←(r10), DCL←(r11)	10	1	1	2	8	–	–	–	–
Load DC Immediate	DCI	aaaa	DC H'aaaa'	2Aaaaa	3	3	2	12	—	—	—	—
Load Program Counter	LR	P0,Q	P0U←(r14), P0L←(r15)	0D	1	1	2	8	–	—	—	—
Load Stack Register	LR	P,K	PU←(r12); PL←(r13)	09	1	1	2	8	–	–	–	—
Return from Subroutine*	POP		P0⇄(P)	1C	1	2		4	–	–	–	—
Store Data Counter	LR	Q,DC	r14←(DCU), r15←(DCL)	0E	1	1	2	8	–	–	–	—
Store Data Counter	LR	H,DC	r10←DCU), r11←(DCL)	11	1	1	2	8	—	–	—	—
Store Stack Register	LR	K,P	r12←(PU); r13←(PL)	08	1	1	2	8	—	—	—	—

SCRATCHPAD REGISTER INSTRUCTIONS (Refer to Scratchpad Addressing Modes)

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	 LONG	µS (2MHzΦ)	 OVR	STATUS BITS ZERO	 CRY	 SIGN
Add Binary	AS	r	A←(A) + (r)	Cr	1	1		2	1/0	1/0	1/0	1/0
Add Decimal	ASD	r	A←(A) + (r)	Dr	1	2	?	4	?	?	1/0	?
Decrement	DS	r	r←(r) + H'FF'	3r	1		1	3	1/0	1/0	1/0	1/0
Load	LR	A,r	A←(r)	4r	1	1		2	–	–	–	–
Load	LR	A, KU	A←(r12)	00	1	1		2	–	–	–	–
Load	LR	A, KL	A←(r13)	01	1	1		2	–	–	–	–
Load	LR	A, QU	A←(r14)	02	1	1		2	–	–	–	–
Load	LR	A, QL	A←(r15)	03	1	1		2	–	–	–	–
Load	LR	r, A	r←(A)	5r	1	1		2	–	–	–	–
Load	LR	KU, A	r12←(A)	04	1	1		2	–	–	–	–
Load	LR	KL, A	r13←(A)	05	1	1		2	–	–	–	–
Load	LR	QU, A	r14←(A)	06	1	1		2	–	–	–	–
Load	LR	QL,A	r15←(A)	07	1	1		2	–	–	–	–
And	NS	r	A← (A) ∧ (r)	Fr	1	1		2	0	1/0	0	1/0
Exclusive Or	XS	r	A←(A) + (r)	Er	1	1		2	0	1/0	0	1/0

*Privileged instruction, Accumulator contents altered during execution of PI instruction.

MISCELLANEOUS INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	µS (2MHzΦ)	OVR	STATUS BITS ZERO	CRY	SIGN
Disable Interrupt	DI		RESET ICB	1A	1	1		2	–	–	–	–
Enable Interrupt *	EI		SET ICB	1B	1	1		2	–	–	–	–
Input	IN	04,05,06,07	A←(Input Port aa)	26aa	2	1	2	8	0	1/0	0	1/0
Input Short	INS	0, 1	A←(Input Port 0 or 1)	A0,A1	1	2		4	0	1/0	0	1/0
Input Short	INS	4,5,6,7	A←(Input Port a)	Aa	1	1	2	8	0	1/0	0	1/0
Load ISAR	LR	IS,A	IS←(A)	0B	1	1		2	–	–	–	–
Load ISAR Lower	LISL	bbb	ISL←bbb	6(1bbb)**	1	1		2	–	–	–	–
Load ISAR Upper	LISU	bbb	ISU←bbb	6(0bbb)**	1	1		2	–	–	–	–
Load Status Register *	LR	W,J	W←(r9)	1D	1	2		4	1/0	1/0	1/0	1/0
No Operation	NOP		P0←(P0) + 1	2B	1	1		2	–	–	–	–
Output *	OUT	04,05,06,07	Output Port aa←(A)	27aa	2	1	2	8	–	–	–	–
Output Short	OUTS	0, 1	Output Port 0 or 1←(A)	B0, B1	1	2		4	–	–	–	–
Output Short	OUTS	4,5,6,7	Output Port a←(A)	Ba	1	1	2	8	–	–	–	–
Store ISAR	LR	A,IS	A←(IS)	0A	1	1		2	–	–	–	–
Store Status Reg	LR	J,W	r9←(W)	1E	1	1		2	–	–	–	–

*Privileged instruction

**b = 1 bit immediate operand

NOTES.

Lower case denotes variables specified by programmer

Function Definitions

←	is replaced by
(_)	the contents of
( )	Binary "1's" complement of
+	Arithmetic Add (Binary or Decimal)
⊕	Logical "OR" exclusive
Λ	Logical "AND"
V	Logical "OR" inclusive
H' '	Hexadecimal digit
[( )]	Contents of memory specified by ( )
a	Address Variable (four bits)
A	Accumulator
b	One bit immediate operand
DC	Data Counter (Indirect Address Register)
DC1	Data Counter 1 (Auxiliary Data Counter)
DCL	Least significant 8 bits of Data Counter Addressed
DCU	Most significant 8 bits of Data Counter Addressed
H	Scratchpad Register 10 and 11
i	Immediate operand (four bits)
ICB	Interrupt Control Bit
IS	Indirect Scratchpad Address Register
ISL	Least Significant 3 bits of ISAR
ISU	Most Significant 3 bits of ISAR
J	Scratchpad Register 9
K	Registers 12 and 13

KL	Register 13
KU	Register 12
P0	Program Counter
P0L	Least Significant 8 bits of Program Counter
P0U	Most Significant 8 bits of Program Counter
P	Stack Register
PL	Least Significant 8 bits of Program Counter
PU	Most Significant 8 bits of Active Stack Register
Q	Registers 14 and 15
QL	Register 15
QU	Register 14
r	Scratchpad Register (any address 0 thru B) (See Below)
W	Status Register

Scratchpad Addressing Modes Using IS. (r ≠ 0 thru B)

r=H'C'	Register Addressed by IS is (Unmodified)
r=H'D'	Register Addressed by IS is Incremented
r=H'E'	Register Addressed by IS is Decremented
r=H'F'	Illegal OP Code.

Status Register

?	Status flag has no meaning
—	No change in condition
1/0	is set to "1" or"0" depending on conditions
CRY	Carry Flag
OVR	Overflow Flag
SIGN	Sign of Result Flag
ZERO	Zero Flag

ELECTRICAL SPECIFICATIONS
ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	0°C to 70°C
Storage Temperature	–65°C to +150°C
Voltage on Any Pin With Respect To Grouns (except open drain pins)	–1.0V to +7V
Voltage On Open Drain Pins	–1.0V to +13.5V
Power Dissipation	1.5W
Power Dissipated by any one I/O pin[4]	.60mW
Power Dissipated by all I/O pins[4]	.600mW

A.C. CHARACTERISTICS – See Figure 12 and 13 for Timing Diagrams

T_A = 0°C to 70°C, V_{CC} = 5V ± 10%, I/O POWER DISSIPATION ≤ 100mW

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
	t_O(INT)	Time Base Period, internal oscillator	250	1000	ns	4MHz - 1.0MHz
XTL 1	t_o(EX)	Time base period, all external modes	250	1000	ns	4MHz-1MHz
XTL 2	t_{EX}(H)	External Clock Pulse Width High	90	700	ns	
	t_{EX}(L)	External Clock Pulse Width Low	100	700	ns	
Φ	$t_Φ$	Internal Φ Clock Period	$2t_0$			
WRITE	tw	Internal WRITE Clock Period	4tΦ 6tΦ			Short Cycle Long Cycle
I/O	t_dI/O	Output delay from internal WRITE Clock	0	1000	ns	50pF plus one TTL load
	t_sI/O	Input Setup time to WRITE Clock	1000		ns	
$\overline{\text{STROBE}}$	$t_{I/O\text{-}s}$	Output valid to STROBE Delay	3tΦ -1000	3tΦ +250		I/O load = 50pF + 1 TTL STROBE Load= 50pF + 3 TTL
	tsl	$\overline{\text{STROBE}}$ Low Time	8tΦ -250	12tΦ +250	ns	
$\overline{\text{RESET}}$	t_{RH}	$\overline{\text{RESET}}$ Hold Time, Low	6tΦ +750		ns	
EXT INT	t_{EH}	EXT INT Hold Time,	6tΦ + 750		ns	To trigger interrupt
		Active and Inactive State	2tΦ			To trigger timer

CAPACITANCE

$T_A = 25°C$, f=2MHz

SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input Capacitance: I/O Ports, $\overline{\text{RESET}}$, $\overline{\text{EXTINT}}$, $\overline{\text{RAMPRT}}$, TEST		7	pF	Unmeasured Pins Grounded
C_{XTL}	Input Capacitance: XTL1, XTL2	20.5	32.5	pF	

DC CHARACTERISTICS - See Figures 12-17 for typical curves.

$T_A = 0°C$ to $70°C$, $V_{CC} = +5V \pm 10\%$, I/O POWER DISSIPATION $\leq$100mW

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
I_{CC}	Power Supply Current		85	mA	Outputs Open
P_D	Power Dissipation		400	mW	Outputs Open
V_{IHEX}	External Clock Input High Level	2.4	5.8	V	
V_{ILHEX}	External Clock Input Low Current	-0.3	0.6	V	
I_{IHEX}	External Clock Input High Current		100	μA	$V_{IHEX} = V_{CC}$
I_{ILEX}	External Clock Input Low Current		-100	μA	$V_{ILEX} = V_{SS}$
V_{IH}	Input High Level Ports,$\overline{\text{RESET}}$[1], EXT INT[1]	2.0	5.8	V	
V_{IHOD}	Open Drain Input High Level	2.0	13.2	V	
V_{IL}	Input Low Level Ports, $\overline{\text{RESET}}$[1], EXT INT[1]	--0.3	0.8	V	
I_{IL}	Input Low Current Ports, $\overline{\text{RESET}}$[2], EXT INT[2]		−1.6	mA	V_{IL}=0.4V
I_L	Leakage Current Open drain ports, $\overline{\text{RAMPRT}}$ $\overline{\text{RESET}}$[3], EXT INT[3]		+10 −5	μA	V_{IN}=13.2V V_{IN}=0.0V
I_{OH}	Output High Current Standard ports, $\overline{\text{RESET}}$[2] EXT INT[2]	−100 −30		μA μA	V_{OH}=2.4V V_{OH}=3.9V
I_{OHDD}	OUTPUT High Current Direct Drive Ports	-0.1		mA	V_{OH} = 2.4V
		−1.5		mA	V_{OH}=1.5V
			−8.5	mA	V_{OH}=.7V
I_{OL}	Output Low Current IO ports	1.8		mA	V_{OL}=0.4V
I_{OHS}	$\overline{\text{STROBE}}$ Output High Current	-300		μA	V_{OH}=2.4V
I_{OLS}	$\overline{\text{STROBE}}$ Output Low Current	5.0		mA	V_{OL} = 0.4V

* Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

1. $\overline{\text{RESET}}$ and EXT INT have internal Schmit triggers giving minimum .2V hysteresis.
2. $\overline{\text{RESET}}$ or EXT INT programmed with standard pull-up
3. $\overline{\text{RESET}}$ or EXT INT programmed without standard pull-up
4. Power dissipation for I/O pins is calculated by$\Sigma(V_{cc} - V_{IL})\ (|I_{IL}|) + \Sigma(V_{CC} - V_{OH})\ (|I_{OH}|) + \Sigma(V_{OL})\ (I_{OL})$

TIMER AC CHARACTERISTICS

Definitions:

Error = Indicated time value - actual time value

tpsc = t Φ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3)	±6tΦ
Cumulative interval error, free running (Note 3)	0
Error between two Timer reads (Note 2)	±(tpsc + tΦ)
Start Timer to stop Timer error (Notes 1,4)	+tΦ to –(tpsc +tΦ)
Start Timer to read Timer error (Notes 1,2)	–5tΦ to –(tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1,3)	–2tΦ to –8tΦ
Load Timer to stop Timer error (Note 1)	+tΦ to –(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1,2)	–5tΦ to –(tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1,3)	–2t Φ to –9tΦ

Pulse Width Measurement Mode:

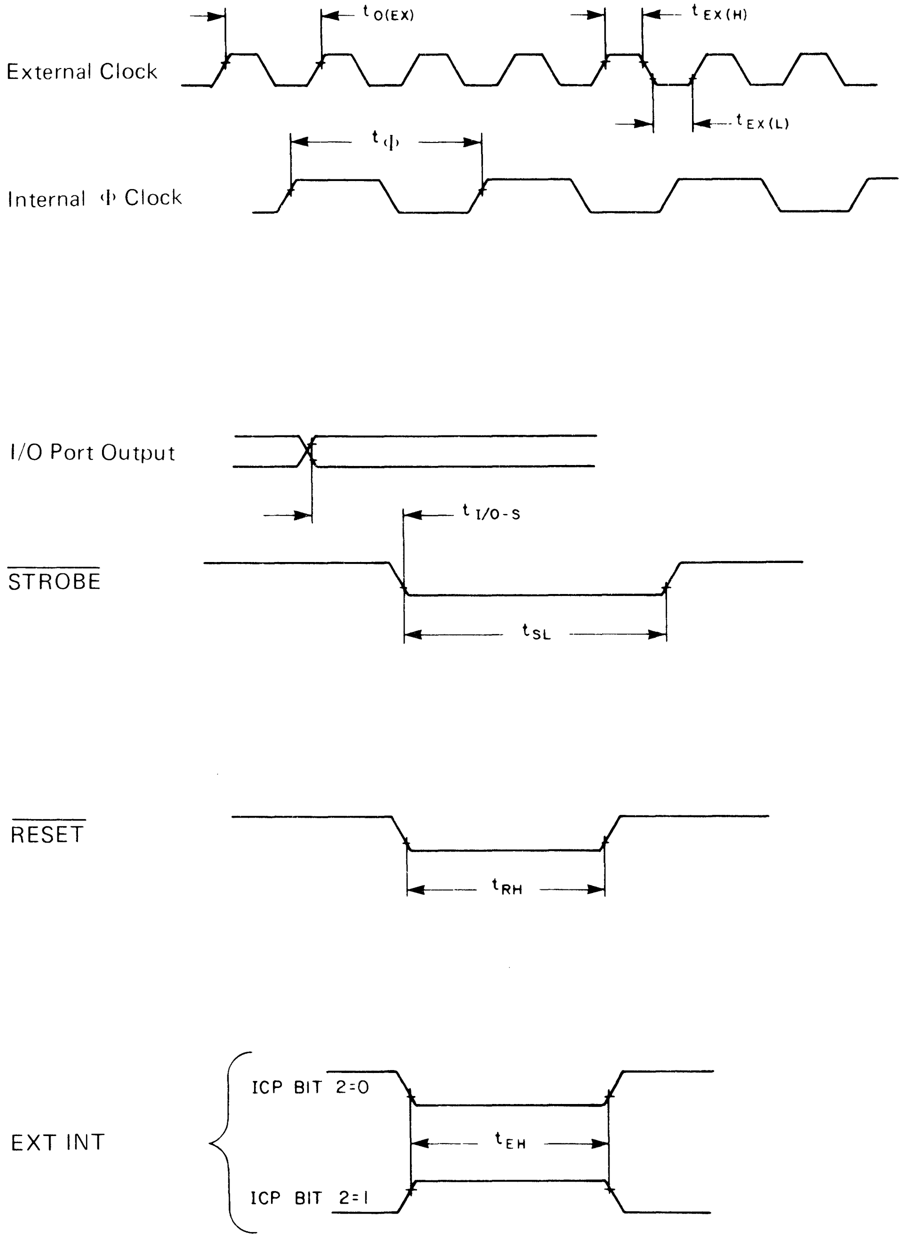
Measurement accuracy (Note 4)	+t Φto –(tpsc +2t Φ)
Minimum pulse width of EXT INT pin	2t Φ

Event Counter Mode:

Minimum active time of EXT INT pin	2tΦ
Minimum inactive time of EXT INT pin	2tΦ

Notes:

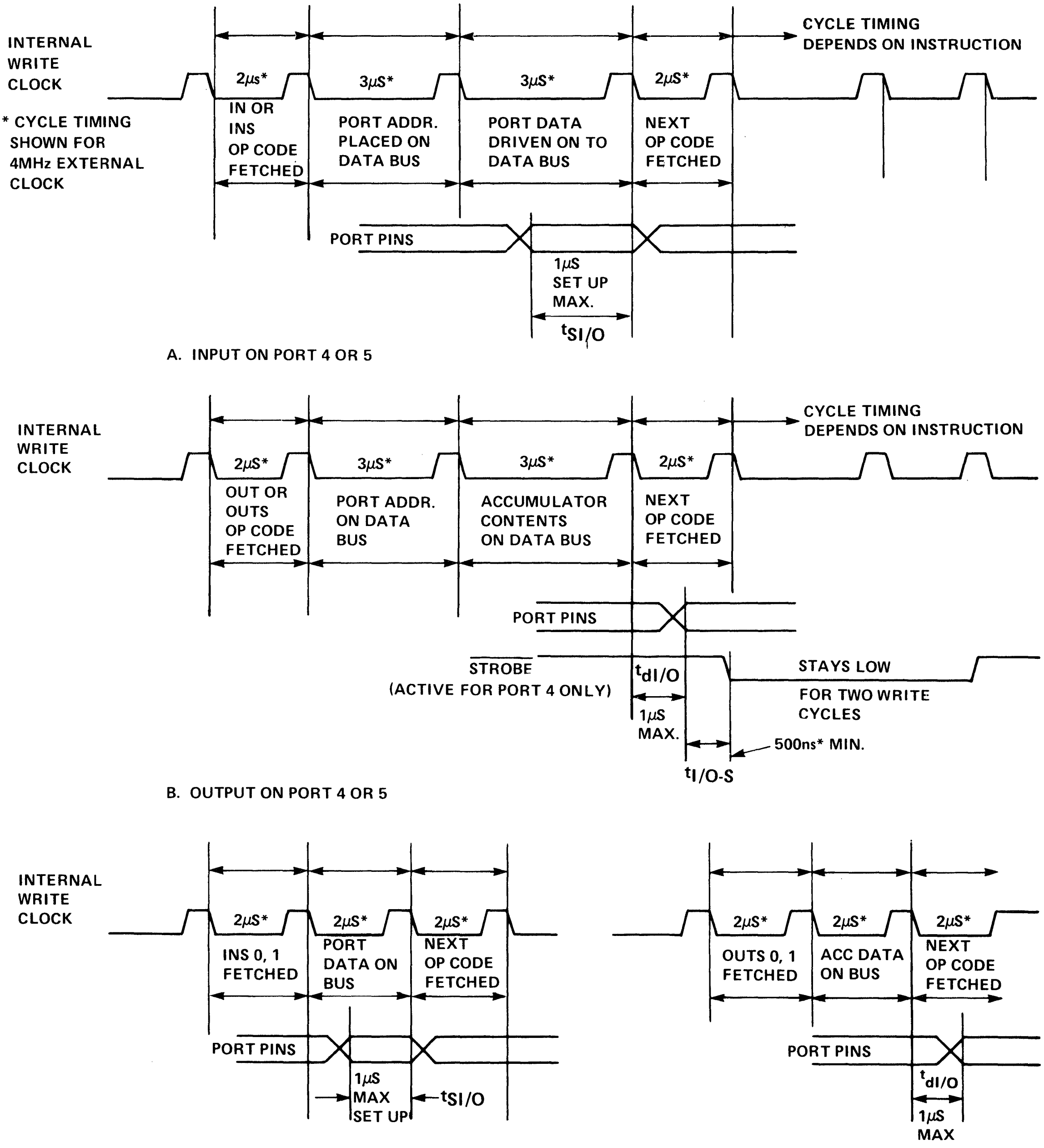
1. All times which entail loading, starting, or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.



Note: All measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} max., or V_{OH} min.

INPUT/OUTPUT AC TIMING

Figure 11



A. INPUT ON PORT 4 OR 5

B. OUTPUT ON PORT 4 OR 5

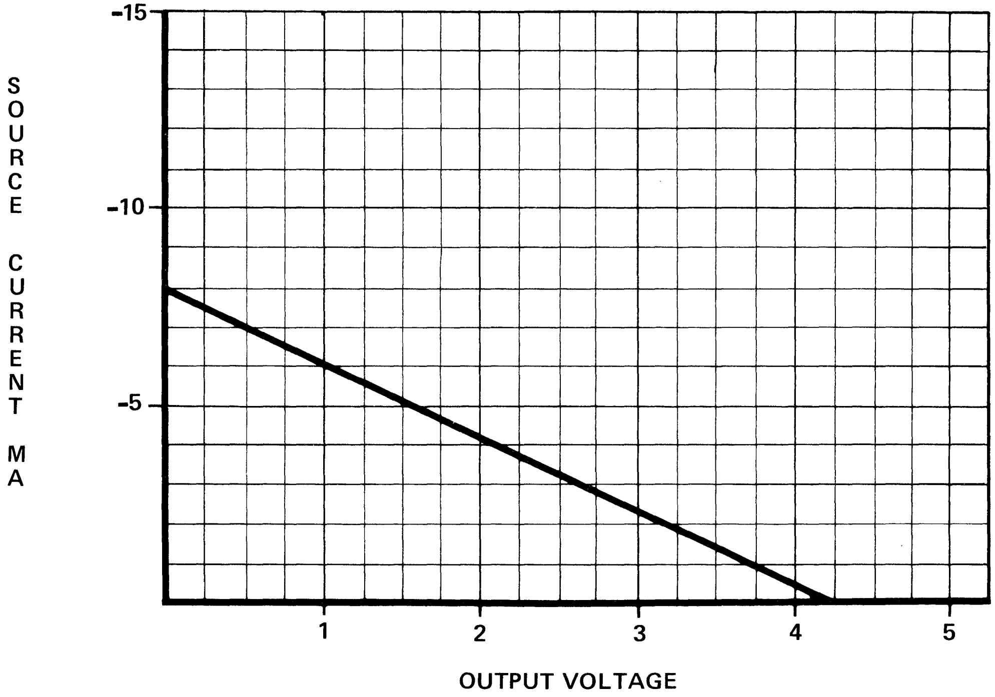
C. INPUT ON PORT 0 OR 1

D. OUTPUT ON PORT 0, 1

3870 Family

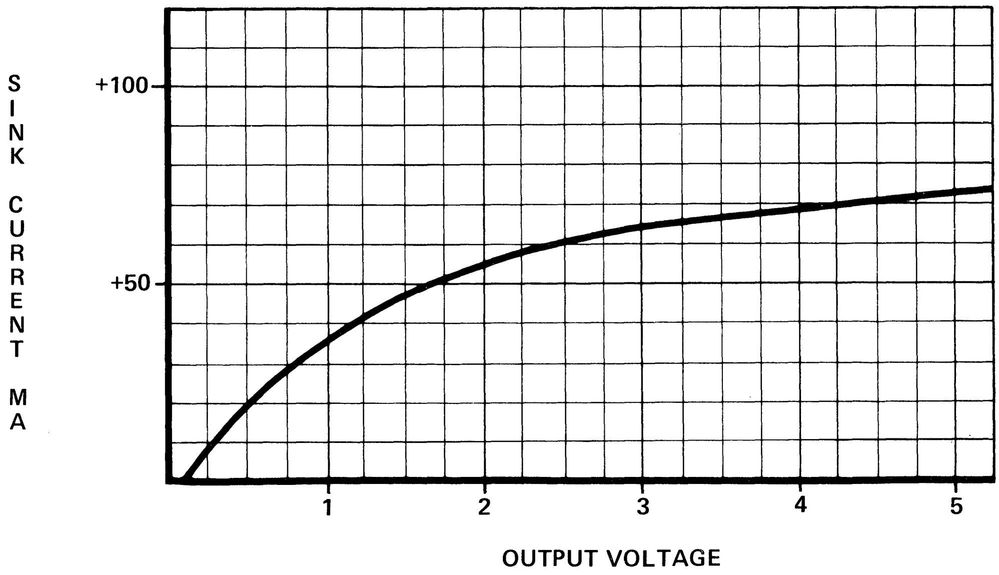
$\overline{\text{STROBE}}$ SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25° C)

Figure 12



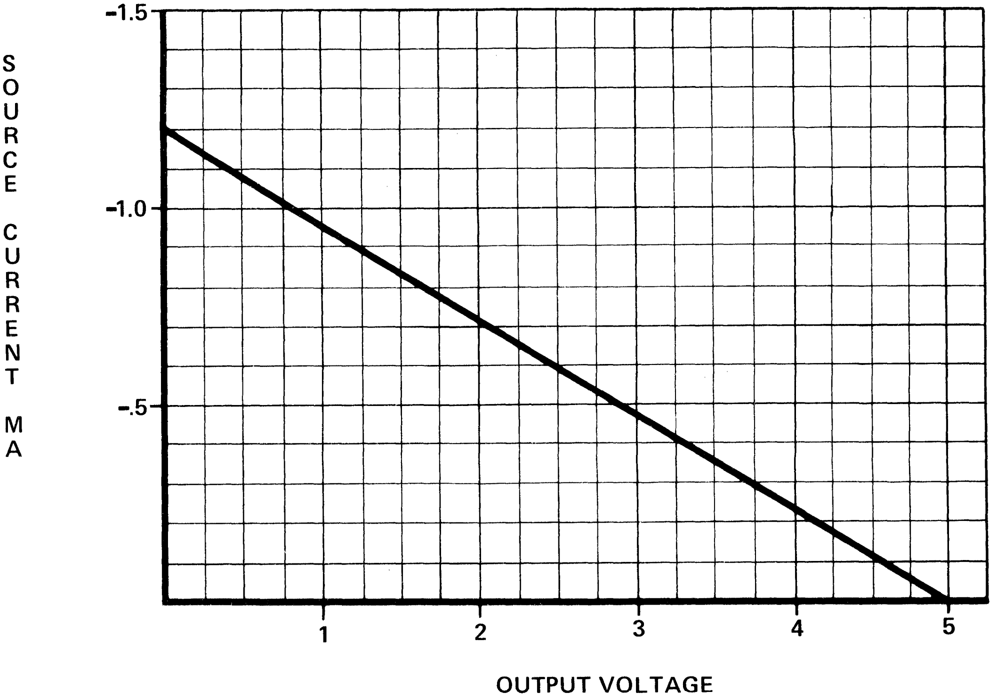
$\overline{\text{STROBE}}$ SINK CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25° C)

Figure 13



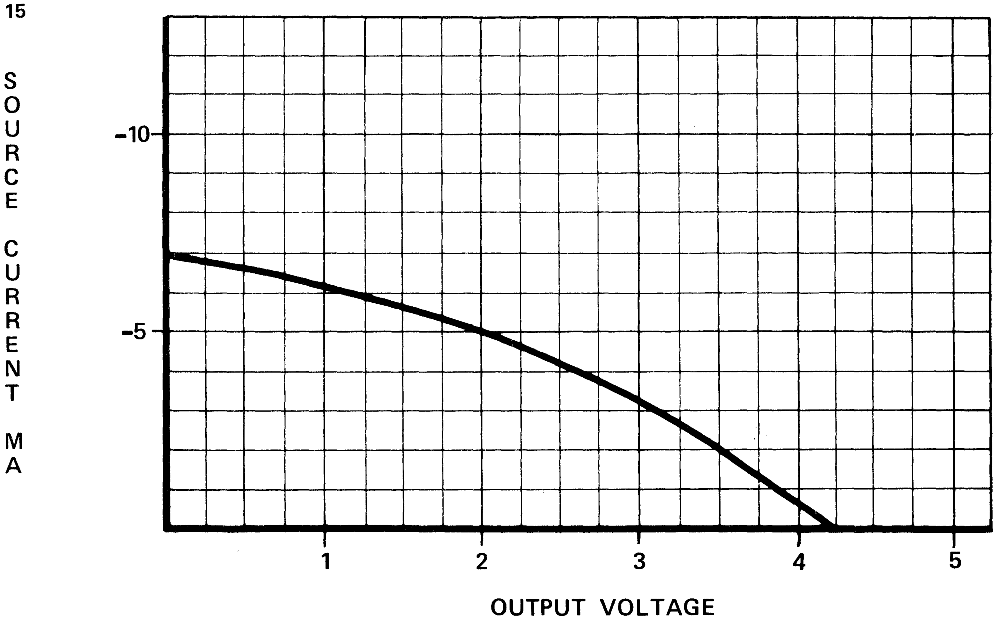
STANDARD I/O PORT SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 14



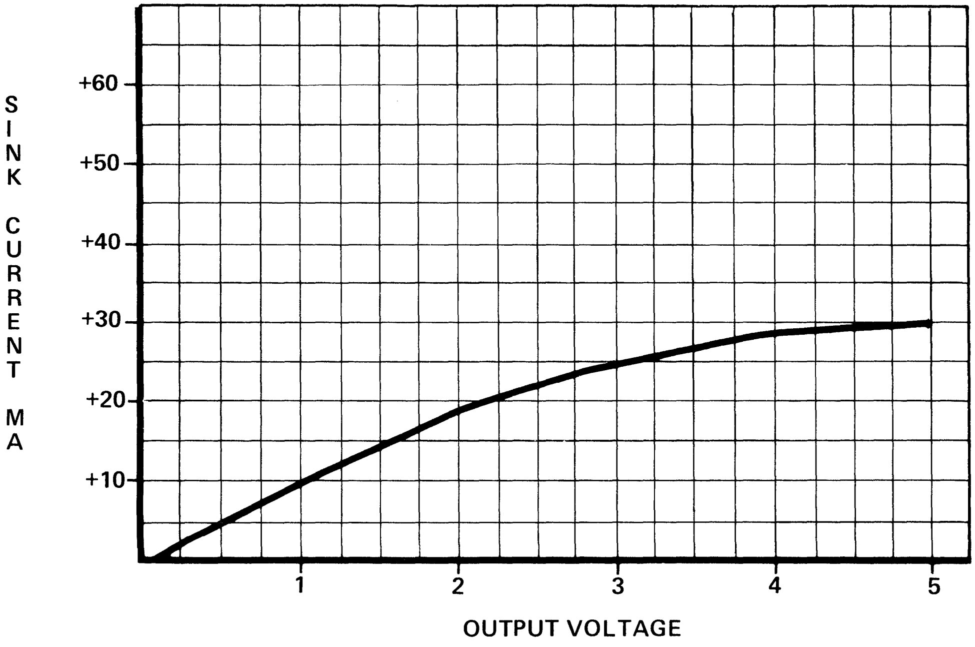
DIRECT DRIVE I/O PORT SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 15



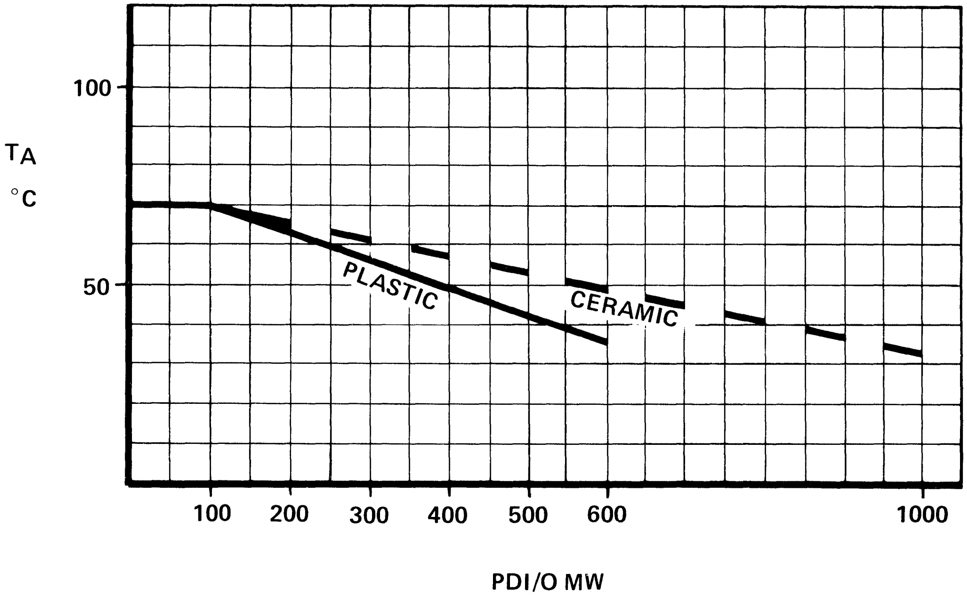
I/O PORT SINK CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 16

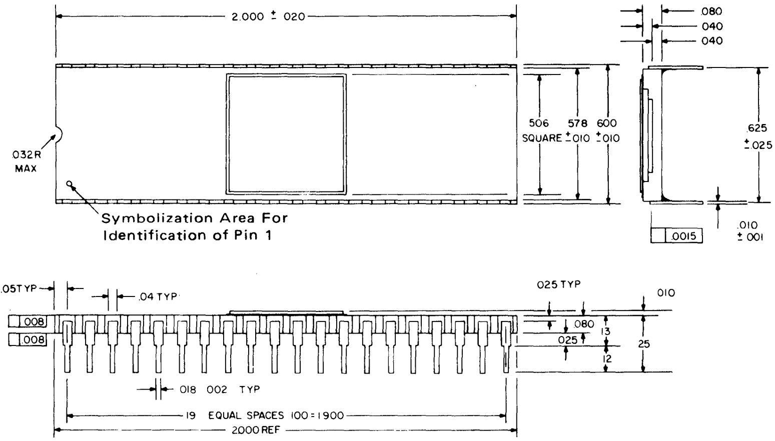


MAXIMUM OPERATING TEMPERATURE VS. I/O POWER DISIPATION

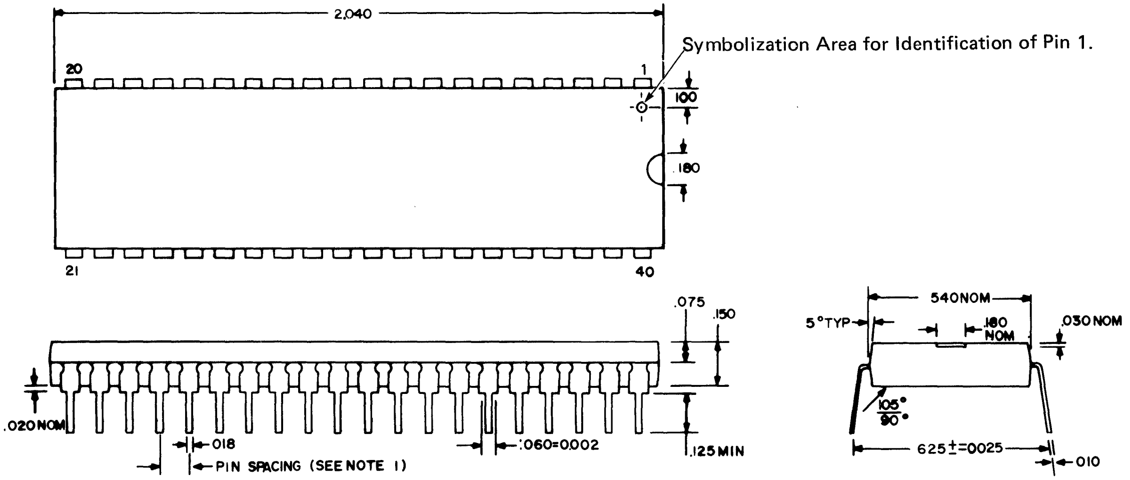
Figure 17



PACKAGE DESCRIPTION: 40-Pin Dual In-Line Ceramic Package



PACKAGE DESCRIPTION 40-Pin Dual-in-Line Plastic Package



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	TEMPERATURE RANGE
MK3870N/14XXX	Plastic	0°C to +70°C
MK3870P/14XXX	Ceramic	0°C to +70°C

APPENDIX A

ORDERING INFORMATION

Custom MK3870 Option Specifications

The custom MK3870 program may be transmitted to Mostek in any of the following media, listed in order of preference:

1) PROMs from the EMU-70
2) Punched paper tape
3) AID-80F Flexible Disk
4) Card Deck (IBM 80 column cards)

The program may be specified in the following forms:

PROMS with correct object code in each location

OBJECT CODE produced by one of Mostek's assemblers.

XFOR-50/70 Fortran IV Cross Assembler, SDB-50/70 resident assembler (ASMB-50/70), AID-80F F8 Cross-Assembler (FZCASM)

OBJECT CODE produced by the dump command from any of Mostek's F8 development hardware (SDB-50/70, AID-80F).

DATA DECK FORMAT as described in the Data Deck section

A completed cover letter (See Fig. A-1) must be attached. The information should be properly packed and mailed prepaid and insured to:

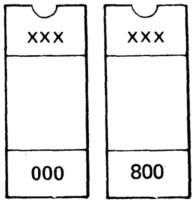
MOSTEK Corporation
Microcomputer Product Marketing
1215 West Crosby Road
Carrollton, Texas 75006

A second copy of the cover letter should be mailed separately to the above address.

PROMS

2708 type PROMs, programmed with the customer program (positive logic sense for addresses and data) may be submitted. The PROMs must be clearly marked to indicate which PROM corresponds to address space 000 7FF and which PROM corresponds to address space 800 FFF. See Fig. A-2 for marking. Include a three-letter customer ID on each PROM. After the PROMs are removed from the EMU-70, they must be placed in a conductive IC carriers and securely packed.

Figure A-2



XXX = Customer ID

Paper Tape

Punched paper tapes (1" wide, 8 level ASCII) will be accepted. The tape must contain the absolute object output from the above mentioned F8 assemblers. Paper object tapes in absolute format generated by the "D" (dump) command of DDT-2 or the dump command of the AID-80F (F8 debug option) are also acceptable if the entire memory space is dumped continuously. Tapes may also be punched using the DATA DECK FORMAT. They must contain 80 characters per record with a CR (carriage return) and LF (line feed) separating each record. The tape must be clearly labeled with customer name, and format used. Fan fold tape is preferred. Tape transparency should be limited to 60% transmissivity (40% opaque). Specifically, thin yellow or white tape is error prone on photo-electric readers and must not be used.

FLEXIBLE DISKS

FLEXIBLE DISKS (Floppy Disks) produced on the Mostek AID-80F development station may be submitted. The format must be the absolute object output from the assemblers, or an object dump using the memory dump command (F8 Debug Option). The disk must be clearly labeled with the format of the data (object, or object dump) and the customer's name.

Punched Card Deck

Standard 80 column punched cards must be used. They must be punched in IBM 029 code. The deck must contain two type of cards:

COMMENT CARDS
DATA CARDS

Comment Cards

Comment Cards must have an asterisk (*) in column 1. The remaining 79 columns may be any character. Comment Cards may be placed anywhere throughout the data deck.

Data Cards

These cards specify the actural ROM data. All fields are right justified.

COLUMN 1:	C (the letter C)
COLUMN 2-9:	ADDR
COLUMN 10-12:	BYTE
COLUMN 14-16:	DATA 1
COLUMN 17-19:	DATA 2
COLUMN 20-22:	DATA 3

3870 ORDERING INFORMATION

DATE ____________________ CUSTOMER PO NUMBER ____________________

CUSTOMER NAME ____________________

ADDRESS ____________________

CITY ____________ STATE ____________ ZIP ____________

COUNTRY ____________________

PHONE ____________________ EXTENSION ____________________

CONTACT ____________________

CUSTOMER PART NUMBER ____________________

OPTIONS:

EXTERNAL INTERRUPT:	Pull-Up □	No Pull-Up □
RESET:	Pull-Up □	No Pull-Up □

PORT OPTIONS:

	STANDARD TTL	OPEN DRAIN	DRIVER PULL-UP
P4-0	□	□	□
P4-1	□	□	□
P4-2	□	□	□
P4-3	□	□	□
P4-4	□	□	□
P4-5	□	□	□
P4-6	□	□	□
P4-7	□	□	□
P5-0	□	□	□
P5-1	□	□	□
P5-2	□	□	□
P5-3	□	□	□
P5-4	□	□	□
P5-5	□	□	□
P5-6	□	□	□
P5-7	□	□	□

PATTERN MEDIA

□ PROMS
(Customer can send in two extra PROM's, MOSTEK will program the customer's code on these PROM's for code verification in the Emulator-70.)

□ PAPER TAPE (DATA DECK)

□ PAPER TAPE (OBJECT)

□ CARD DECK (DATA DECK)

□ DISKETTE (OBJECT)

THESE ITEMS MAY AFFECT COST

BRANDING REQUIREMENT (If any, 10 Alpha-numeric digits allowed)

PROTOTYPE QUANTITY (10 pieces at no charge - higher quantity extra charge)

WAIVE PROTOTYPES (Customer accepts liability for all work in process)

Yes ____________________ No ____________________

SIGNATURE ________________________________

TITLE ________________________________

COLUMN 76-78:	DATA 21
COLUMN 77-79:	DATA 22 or SEQUENCE NUMBER

ADDR is the address of the first byte of data (DATA 1) contained on that card. Successive data bytes read from that card will be placed in successively greater address locations. BYTE is the number of data bytes to be read from that card (1 to 22). If sequence numbers are used, the maximum number of bytes per card is 21. The base for ADDR and BYTE may be either decimal or hex but both must be the same. Data may be either in decimal or hex regardless of the base used for ADDR and BYTE. The base for sequence numbers (if they are used) is always decimal. The bases must be consistent throughout the deck. Data cards need not occur in order of increasing or decreasing addresses. Any unspecified address will be filled with zero. Any unpunched field will be read as a zero. If two data cards specify data for the same address, the one encountered second in the deck will override the first.

A portion of an example deck is shown.

```
*    3870 DATA DECK

*    MOSTEK CORP, EXAMPLE APPLICATION

*    ADDR/BYTE ARE IN DECIMAL

*    DATA IS IN HEX

C  0  8  20  FF  OB  54  34  56  71  B6

C  8  8  1B  28  03  F3  4C  25  2E  94

C  16  8  04  29  01  00

*    START OF SUBROUTINE ALPHA

C  1096  4  20  32  7C  53

C  1100  4  52  47  29  06

C  1104  1  07
```

Verification Media

All original pattern media (PROMs, paper tape, etc.) are filed for contractural purposes and are not returned. Two copies of computer listings printed during the creation of the custom mask pattern are returned. One copy may be kept by the customer. The other copy should be checked thoroughly, signed, and returned to Mostek. The signed listing constitutes the contractual agreement for creation of the customer mask. Though the computer listing serves as the actual verification media, Mostek will program 2708 PROMs programmed from the data file used to create the custom mask to aid in the verification process. If programmed PROMs are desired, two blank 2708 type PROMs must be provided by the customer.

PRELIMINARY

MOSTEK®

F8 MICROCOMPUTER DEVICES

Single-Chip Microcomputer MK3872

FEATURES

□ Software compatible with F8 family

□ 4032 x 8 mask programmable ROM

□ 64 byte scratchpad RAM

□ 64 additional bytes of executable RAM addressable by program counter or data counter

□ Standby option for executable RAM including:
—Low standby power, less than 8.2mW
—Minimum 2.2V standby supply voltage
—No external components required to trickle charge battery

□ 32 bits (4 ports) TTL Compatible I/O

□ Programmable binary timer
—Internal timer mode
—Pulse width measurement mode
—Event counter mode

□ External interrupt

□ Crystal, LC, RC, external, or internal time base

□ Low power (285mW typ.)

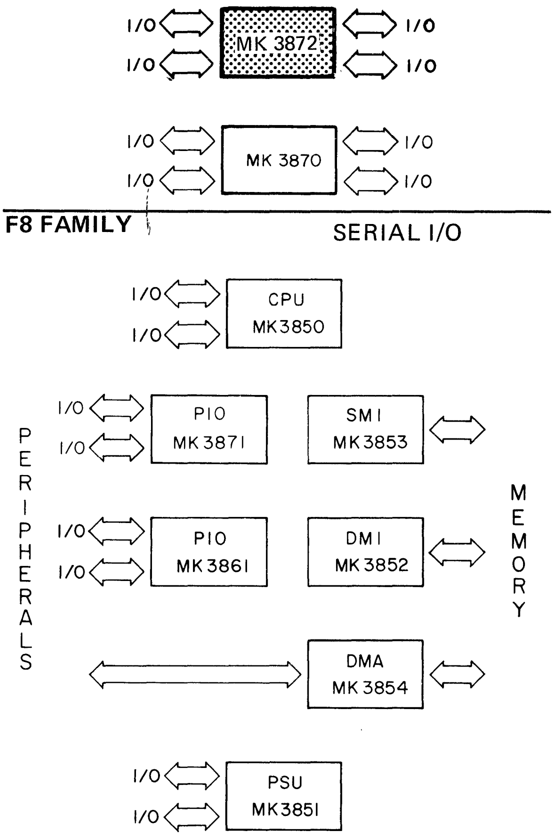
□ Single +5 volt ± 10% power supply

□ Same pinout as MK3870

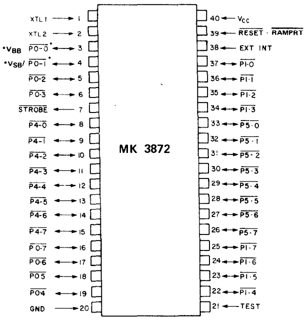
GENERAL DESCRIPTION

The MK3872 is a complete 8-bit microcomputer on a single MOS integrated circuit. The 3872 can execute the F8 instruction set of more than 70 commands, allowing expansion into multi-chip configurations with software compatibility. The device features 4032 bytes of ROM, 64 bytes of scratchpad RAM, 64 bytes of executable RAM, a programmable binary timer, 32 bits of I/O, and a single +5 volt power supply requirement. Utilizing ion-implanted, N-channel silicon gate technology and advanced circuit design techniques the single-chip 3872 offers maximum cost-effectiveness in a wide range of control and logic replacement applications. The 3872 is an expanded memory version of the 3870 single chip microcomputer. The 3872 is identical to the 3870 in the following areas: instruction set, architecture, AC and DC characteristics, and pinout. The only change is in the memory expansion along with the appropriate memory address registers.

SINGLE CHIP 3870 MICROCOMPUTER FAMILY



PIN CONNECTIONS



*PROGRAMMABLE PIN FUNCTION DEPENDS ON DEVICE OPTION (STANDBY MODE DEVICE OR STANDARD DEVICE).

PIN NAME	DESCRIPTION	TYPE
P0-0 – P0-7	I/O Port 0	Bidirectional
P1-0 – P1-7	I/O Port 1	Bidirectional
P4-0 – P4-7	I/O Port 4	Bidirectional
P5-0 – P5-7	I/O Port 5	Bidirectional
STROBE	Ready Strobe	Output
EXT INT	External Interrupt	Input
RESET – RAMPRT	External Reset, RAM Protect	Input
TEST	Test Line	Input
XTL 1, XTL 2	Time Base	Input
V_{CC}, GND	Power Supply Lines	Input
V_{SB}	Standby Power	Input
V_{BB}	Substrate Decoupling	Input

FUNCTIONAL PIN DESCRIPTION

P0-0–P0-7, P1-0–P1-7, P4-0–P4-7, and P5-0–P5-7 are 32 lines which can be individually used as either TTL compatible inputs or as latched outputs.

STROBE is a ready strobe associated with I/O Port 4. This pin which is normally high provides a single low pulse after valid data is present on the P4-0–P4-7 pins during an output instruction.

RESET - RAMPRT may be used to externally reset the 3872. When pulled low the 3872 will reset. When allowed to go high the 3872 will begin program execution at program location H '000'. Additionally when RESET - RAMPRT is brought low all accesses of the executable RAM are prevented and the RAM is placed in a protected state for powering down V_{CC} without loss of data when the STANDBY option is selected.

EXT INT is the external interrupt input. Its active state is software programmable. This input is also used in conjunction with the timer for pulse width measurement and event counting.

XTL 1 and XTL 2 are the time base inputs to which a crystal (1 to 4MHz), LC network, RC network, or an external single-phase clock may be connected.

TEST is an input, used only in testing the 3872. For normal circuit functionality this pin is left unconnected or may be grounded.

V_{CC} is the power supply input (+5V± 10%).

V_{SB} is the RAM standby power supply input if the standby option is selected (+5.5V to +2.2V).

V_{BB} is the substrate decoupling pin. A .01 microFarad capacitor is required to provide substrate decoupling. It is only used when standby option is selected.

3870 ARCHITECTURE

This section describes the basic functional elements of the 3872 as shown in the block diagram of Figure 1. A programming model is shown in Figure 2.

Main Control Logic

The Instruction Register (IR) receives the operation code (OP code) of the instruction to be executed from the program ROM via the data bus. During all OP code fetches eight bits are latched into the IR. Some instructions are completely specified by the upper 4 bits of the OP code. In those instructions the lower 4 bits are an immediate register address or an immediate 4 bit operand. Once latched into the IR the main control logic decodes the instruction and provides the necessary control gating signals to all circuit elements.

ROM Address Registers

There are four 12 bit registers associated with the 4K x 8 ROM and 64 x 8 RAM. These are the Program Counter (P0), the Stack Register (P), the Data Counter (DC) and the Auxiliary Data Counter (DC1). The Program Counter is used to address instructions or immediate operands. P is used to save the contents of P0 during an interrupt or subroutine call. Thus, P contains the return address at which processing is to resume upon completion of the subroutine or the interrupt routine.

The Data Counter (DC) is used to address data tables. This register is auto-incrementing. Of the two data counters only DC can access the memory. However, the XDC instruction allows DC and DC1 to be exchanged.

Associated with the address registers is a 12 bit Adder/Incrementer. This logic element is used to increment P0 or DC when required and is also used to add displacements to P0 on relative branches or to add the data bus contents to DC in the ADC (add data counter) instruction.

4032 x 8 ROM

The microcomputer program and data constants are stored in the program ROM. When a ROM access is required, the appropriate address register (P0 or DC) is gated onto the ROM address bus and the ROM output is gated onto the main data bus. The first byte in ROM is location zero.

64 x 8 Executable RAM

The upper 64 bytes of the total 4096 byte memory of the 3872 is RAM memory. The first byte is at address 4032 decimal (FCO hex). As with the ROM memory the RAM memory may be accessed by the PO and DC address registers. It may be written via the STORE (ST)' instruction. It may be read via the LOAD (LM) instruction. Additionally instructions may be executed from the RAM. A mask programmable standby power option is available whereby the 64x8 RAM remains powered and protected so that its contents are saved during a loss of the normal circuit power supply.

Scratchpad and IS

The scratchpad provides 64 8-bit registers which may be used as general purpose RAM memory. The Indirect Scratchpad Address Register (IS) is a 6 bit register used to address the 64 registers. All 64 registers may be accessed using IS. In addition the lower order 12 registers may also be directly addressed.

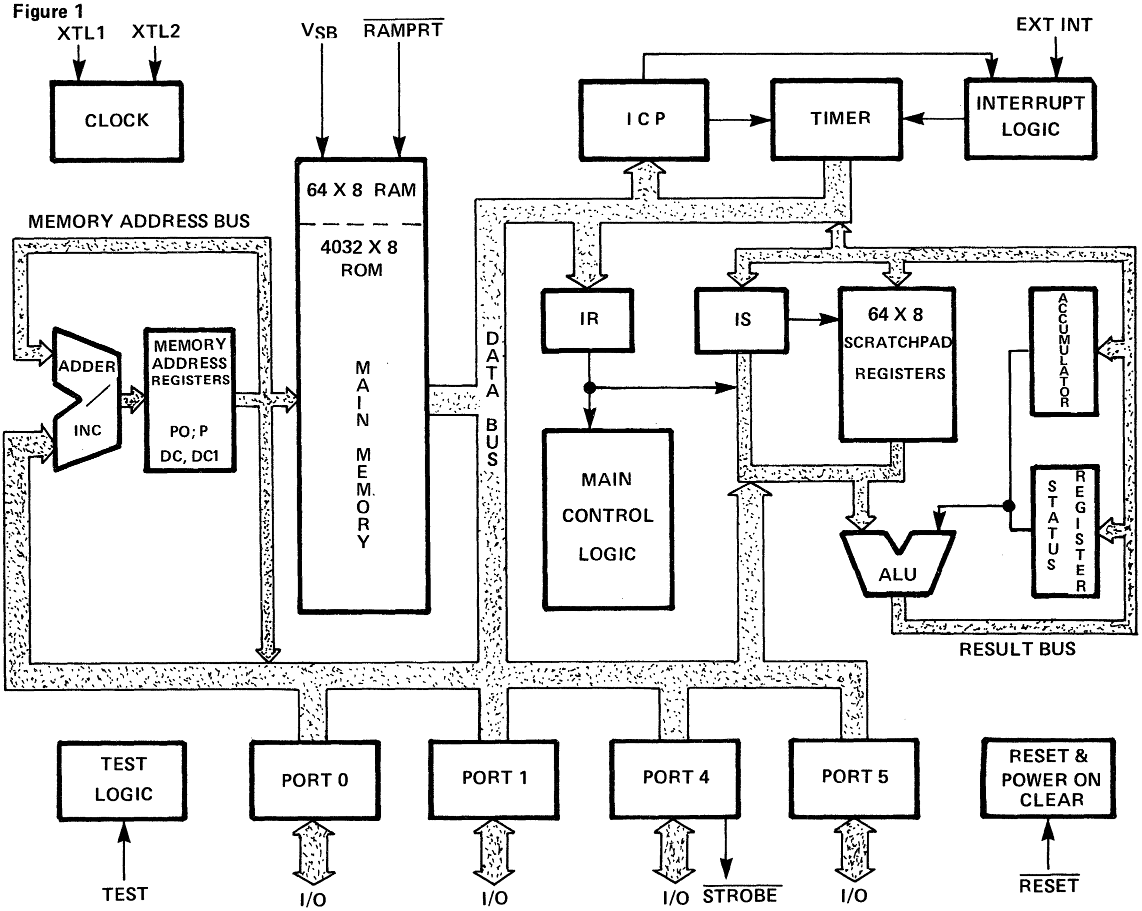
IS can be visualized as holding two octal digits. This division of IS is important since a number of instructions increment or decrement only the least significant 3 bits of IS when referencing scratchpad bytes via IS. This makes it easy to reference a buffer consisting of contiguous scratchpad bytes. For example, When the low order octal digit is incremented or decremented IS is incremented from octal 27 (0 '27') to 0 '20' or is decremented from 0 '20' to 0 '27'. This feature of the IS is very useful in many program sequences. All six bits of IS may be loaded at one time or either half may be loaded independently.

Scratchpad registers 9 through 15 (decimal) are given mnemonic names (J, H, K, and Q) because of special linkages between these registers and other registers such as the Stack Register. These special linkages facilitate the implementation of multi-level interrupts and subroutine nesting. For example, the instruction LR K, P stores the lower eight bits of the Stack Register into register 13 (K lower or KL) and stores the upper three bits of P into register 12 (K upper or KU) The scratchpad is not protected with the standby power option.

Arithmetic and Logic Unit (ALU)

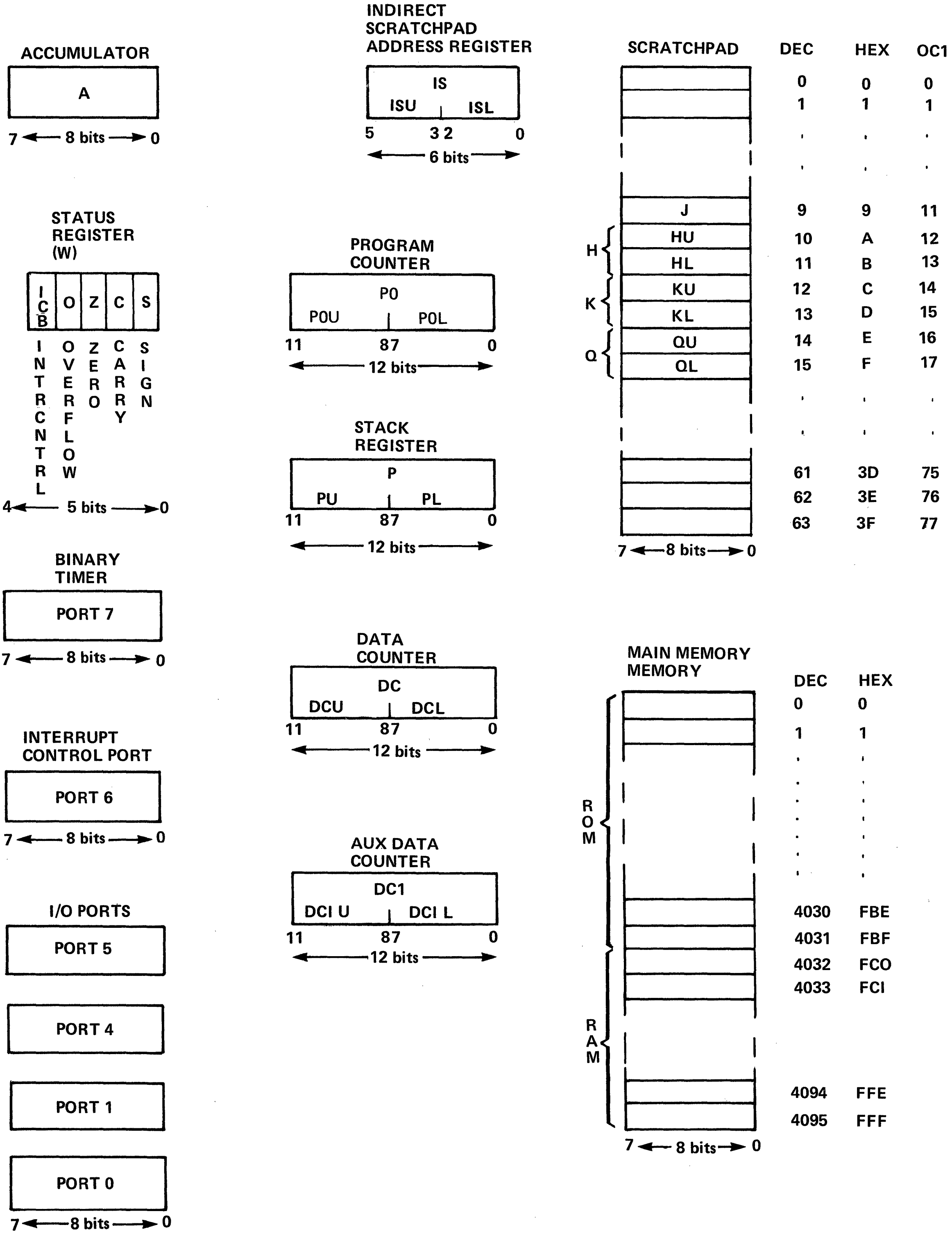
After receiving commands from the main control

MK 3872 BLOCK DIAGRAM
Figure 1



3872 PROGRAMMABLE REGISTERS, PORTS AND MEMORY MAP

Figure 2



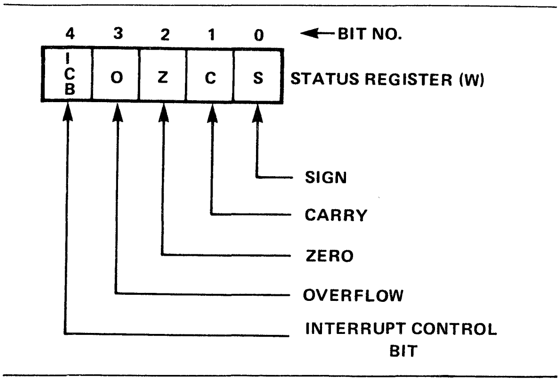
logic, the ALU performs the required arithmetic or logic operations (using the data presented on the two input busses) and provides the result on the result bus. The arithmetic operations that can be performed in the ALU are binary add, decimal adjust, add with carry, decrement, and increment. The logic operations that can be performed are AND, OR, EXCLUSIVE OR, 1's complement, shift right, and shift left. Besides providing the result on the result bus, the ALU also provides four signals representing the status of the result. These signals, stored in the Status Register (W), represent CARRY, OVERFLOW, SIGN, and ZERO condition of the result of the operation.

Accumulator (A)

The Accumulator (A) is the prinicpal register for data manipulation within the 3872. A serves as one input to the ALU for arithmetic or logical operations. The result of ALU operations are stored in A.

The Status Register (W)

The Status Register (also called the W register) holds five status flags as follows:



Summary of Status Bits

$$\text{OVERFLOW} = \text{CARRY}_7 \oplus \text{CARRY}_6$$

$$\text{ZERO} = \overline{\text{ALU}_7} \wedge \overline{\text{ALU}_6} \wedge \overline{\text{ALU}_5} \wedge \overline{\text{ALU}_4} \wedge \overline{\text{ALU}_3} \wedge \overline{\text{ALU}_2} \wedge \overline{\text{ALU}_1} \wedge \overline{\text{ALU}_0}$$

$$\text{CARRY} = \text{CARRY}_7$$

$$\text{SIGN} = \overline{\text{ALU}_7}$$

Interrupt Control Bit (ICB)

The ICB may be used to allow or disallow interrupts in the 3872. This bit is not the same as the two interrupt enable bits in the Interrupt Control Port (ICP). If the ICB is set and the 3872 interrupt logic communicates an interrupt request to the CPU section, the interrupt will be acknowledged and processed upon completion of the first non-privileged instruction. If the ICB is cleared an interrupt request will not be acknowledged or processed until the ICB is set.

I/O Ports

The 3872 provides four complete bidirectional Input/Output ports. (When standby option is used, Port 0, bit 0 and 1 are not available). These are Ports 0, 1, 4, and 5. In addition, the Interrupt Control Port is addressed as Port 6 and the binary timer is addressed as Port 7. An output instruction (OUT or OUTS) causes the contents of A to be latched into the addressed port. An input instruction (IN or INS) transfers the contents of the port to A (port 6 is an exception which is described later). The I/O pins on the 3872 are logically inverted. The schematic of an I/O pin and available output drive options are shown in Figure 3.

An output ready strobe is associated with Port 4. This flag may be used to signal a peripheral device that the 3872 has just completed an output of new data to Port 4. The strobe provides a single low pulse shortly after the output operation is completely finished, so either edge may be used to signal the peripheral. $\overline{\text{STROBE}}$ may also be used as an input strobe simply by doing a dummy output of H '00' to Port 4 after completing the input operation.

Timer and Interrupt Control Port

The Timer is an 8-bit binary down counter which is software programmable to operate in one of three modes: the Interval Timer Mode, the Pulse Width Measurement Mode, or the Event Counter Mode. As shown in Figure 4, associated with the Timer are an 8-bit register called the Interrupt Control Port, a programmable prescaler, and an 8-bit modulo-N register. A functional logic diagram is shown in Figure 5.

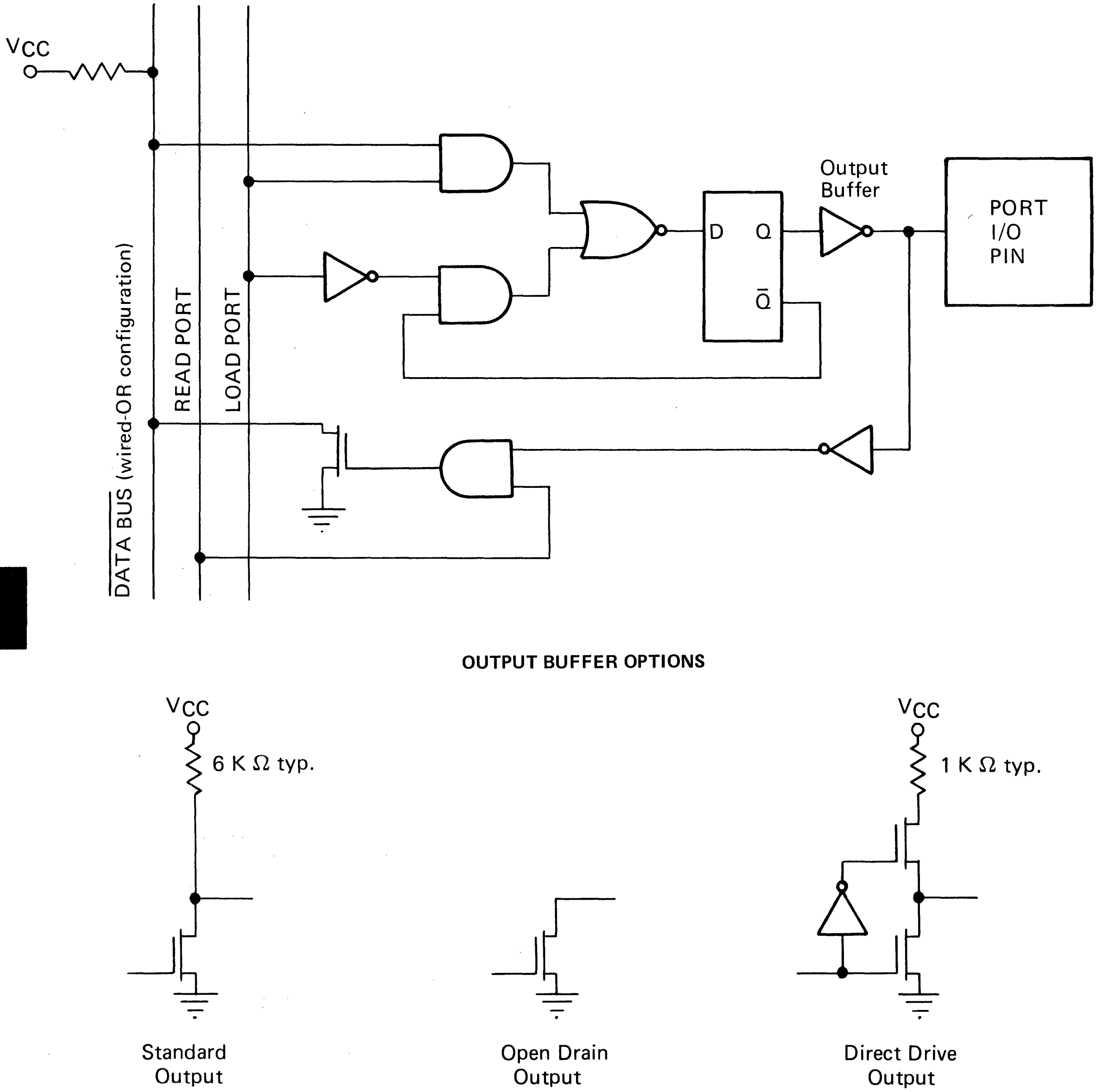
The desired timer mode, prescale value, starting and stopping the timer, active level of the EXT INT pin, and local enabling or disabling of interrupts are selected by outputting the proper bit configuration from the Accumulator to the Interrupt Control Port (Port 6) with an OUT or OUTS instruction. Bits within the Interrupt Control Port are defined as follows:

Interrupt Control Port (Port 6)

Bit 0 - External Interrupt Enable
Bit 1 - Timer Interrupt Enable
Bit 2 - EXT INT Active Level
Bit 3 - Start/Stop Timer
Bit 4 - Pulse Width/Interval Timer

Bit 5 - ÷ 2 Prescale
Bit 6 - ÷ 5 Prescale
Bit 7 - ÷ 20 Prescale

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS
Figure 3



Ports 0 and 1 are Standard Output type only.

Ports 4 and 5 may both be any of the three output options (programmable bit by bit).

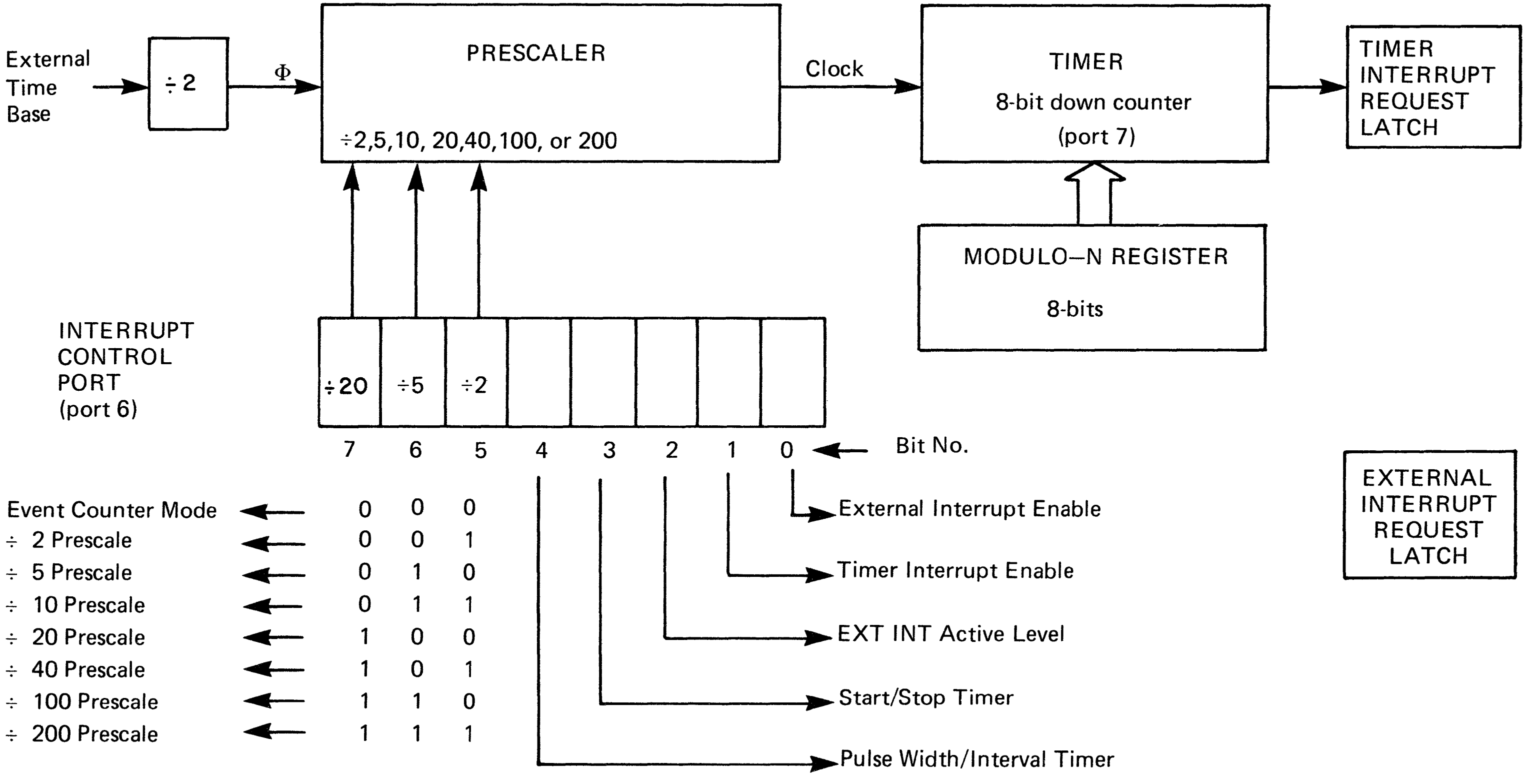
The $\overline{\text{STROBE}}$ output is always configured similar to a Direct Drive Output except that it is capable of driving 3 TTL loads.

$\overline{\text{RESET}}$ and EXT INT may have standard 6KΩ (typical) pull-up or may have no pull-up.

TIMER & CONTROL PORT BLOCK DIAGRAM

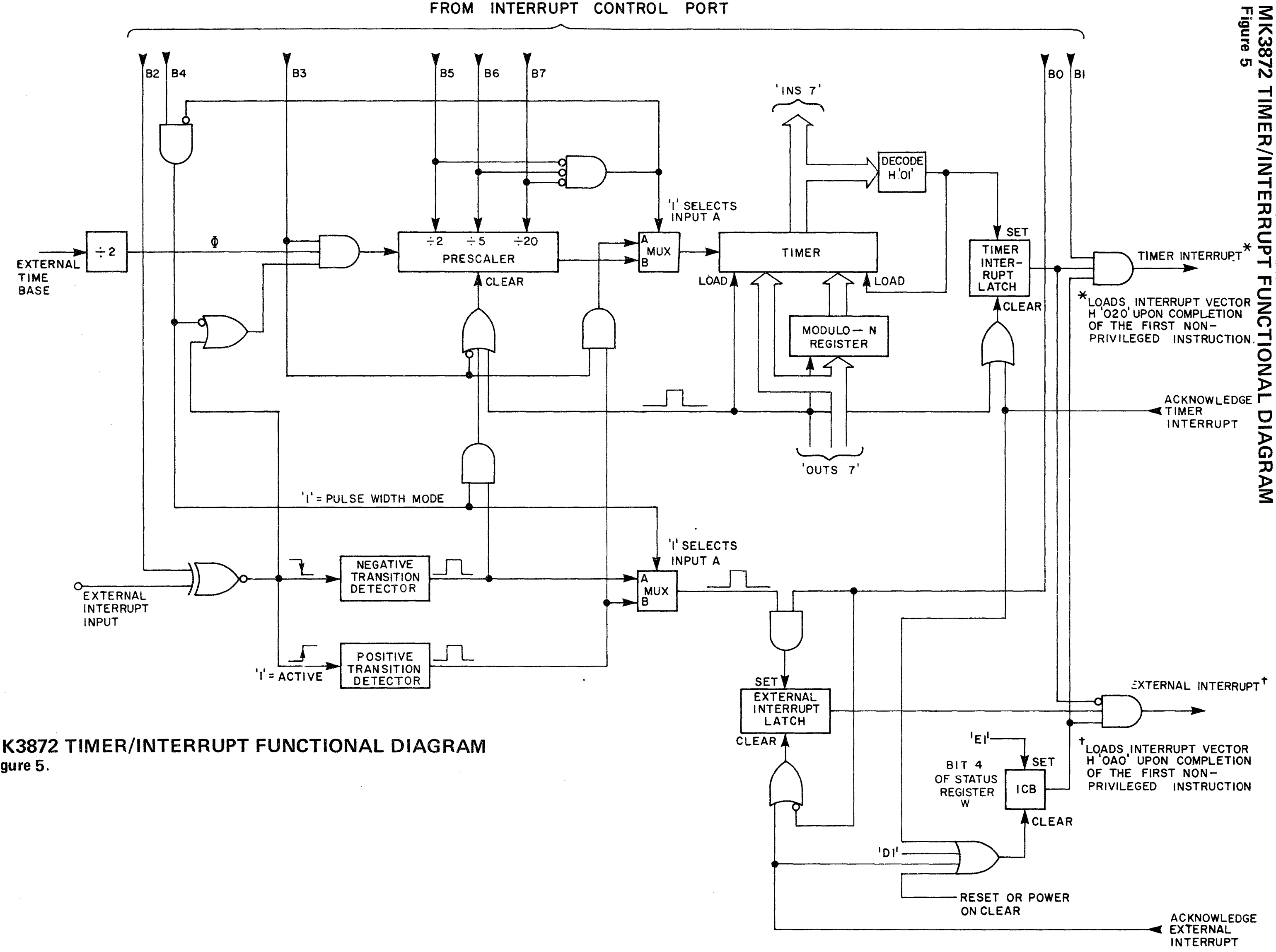
Figure 4

TIMER & CONTROL PORT BLOCK DIAGRAM
Figure 4



	7	6	5
Event Counter Mode	0	0	0
÷ 2 Prescale	0	0	1
÷ 5 Prescale	0	1	0
÷ 10 Prescale	0	1	1
÷ 20 Prescale	1	0	0
÷ 40 Prescale	1	0	1
÷ 100 Prescale	1	1	0
÷ 200 Prescale	1	1	1

Note: See Figure 5 for a more detailed functional diagram.



MK3872 TIMER/INTERRUPT FUNCTIONAL DIAGRAM
Figure 5.

A special situation exists when reading the Interrupt Control Port (with an IN or INS instruction). The Accumulator is not loaded with the content of the ICP; instead, Accumulator bits 0 through 6 are loaded with 0's while bit 7 is loaded with the logic level being applied to the EXT INT pin, thus allowing the status of EXT INT to be determined without the necessity of servicing an external interrupt request. When reading the Interrupt Control Port (Port 6) bit 7 of the Accumulator is loaded with the actual logic level being applied to the EXT INT pin, regardless of the status of ICP bit 2 (the EXT INT Active Level bit); that is, if EXT INT is at +5V bit 7 of the Accumulator is set to a logic 1, but if EXT INT is at GND then Accumulator bit 7 is reset to logic 0. This capability is useful in establishing a high speed polled handshake procedure or for using EXT INT as an extra input pin if external interrupts are not required and the Timer is used only in the Interval Timer Mode. However, if it is desirable to read the contents of the ICP then one of the 64 scratchpad registers or one byte of RAM may be used to save a copy of whatever is written to the ICP.

The rate at which the timer is clocked in the Interval Timer Mode is determined by the frequency of an internal Φ clock and by the division value selected for the prescaler. (The internal Φ clock operates at one-half the external time base frequency). If ICP bit 5 is set and bits 6 and 7 are cleared, the prescaler divides Φ by 2. Likewise, if bit 6 or 7 is individually set the prescaler divides Φ by 5 or 20 respectively. Combinations of bits 5, 6 and 7 may also be selected. For example, if bits 5 and 7 are set while 6 is cleared the prescaler will divide by 40. Thus possible prescaler values are ÷2, ÷5. ÷10, ÷20, ÷40, ÷100, and ÷200.

Any of three conditions will cause the prescaler to be reset: whenever the timer is stopped by clearing ICP bit 3, execution of an output instruction to Port 7, (the timer is assigned port address 7), or on the trailing edge transition of the EXT INT pin when in the Pulse Width Measurement Mode. These last two conditions are explained in more detail below.

An OUT or OUTS instruction to Port 7 will load the content of the Accumulator to both the Timer and the 8-bit modulo-N register, reset the prescaler, and clear any previously stored timer interrupt request. As previously noted, the Timer is an 8-bit down counter which is clocked by the prescaler in the Interval Timer Mode and in the Pulse Width Measurement Mode. The prescaler is not used in the Event Counter Mode. The Modulo-N register is a buffer whose function is to save the value which was most recently outputted to Port 7. The modulo-N register is used in all three timer modes.

Interval Timer Mode

When ICP bit 4 is cleared (logic 0) and at least one prescale bit is set the Timer operates in the Interval Timer Mode. When bit 3 of the ICP is set the Timer will start counting down from the modulo-N value. After counting down to H'01', the Timer returns to the modulo-N value at the next count. On the transition from H'01' to H 'N' the Timer sets a timer interrupt request latch. Note that the interrupt request latch is set by the transition to H 'N' and not be the presence of H 'N' in the Timer, thus allowing a full 256 counts if the modulo-N register is preset to H '00'. If bit 1 of the ICP is set, the interrupt request is passed on to the CPU section of the 3872. However, if bit 1 of the ICP is a logic 0 the interrupt request is not passed on to the CPU section but the interrupt request latch remains set. If ICP bit 1 is subsequently set, the interrupt request will then be passed on to the CPU section. (Recall from the discussion of the Status Register's Interrupt Control Bit that the interrupt request will be acknowledged by the CPU section only if ICB is set). Only two events can reset the timer interrupt request latch; when the timer interrupt request latch is acknowledged by the CPU section, or when a new load of the modulo-N register is performed.

Consider an example in which the modulo-N registe is loaded with H '64' (decimal 100). The time interrupt request latch will be set at the 100th count following the timer start and the timer interrupt request latch will repeatedly be set on precise 100 count intervals. If the prescaler is set at ÷40 the timer interrupt request latch will be set every 4000 Φ clock periods. For a 2MHz Φ clock (4MHz time base frequency) this will produce 2 millisecond intervals.

The range of possible intervals is from 2 to 51,200 Φ clock periods (1μs to 25.6ms for a 2MHzΦ clock). However, approximately 50 Φ periods is a practical minimum because the time between setting the interrupt request latch and the execution of the first instruction of the interrupt service routine is at least 29 Φ periods (the response time is dependent upon how many privileged instructions are encountered when the request occurs). To establish time intervals greater than 51,200 Φ clock periods is a simple matter of using the timer interrupt service routine to count the number of interrupts, saving the result in one or more of the scratchpad registers until the desired interval is achieved. With this technique virtually any time interval, or several time intervals, may be generated.

The Timer may be read at any time and in any mode using an input instruction (IN 7 or INS 7) and may take place "on the fly" without interferring with normal timer operation. Also, the Timer may be stopped at any time by clearing bit 3 of the ICP. The timer will hold its current contents indefinitely and will resume counting when bit 3 is again set. Recall however that the prescaler is reset whenever the Timer is stopped; thus a series of starting and stopping will result in a cumulative truncation error.

A summary of other timer errors is given in the timing section of this specification. For a free running timer in the Interval Timer Mode the time interval between any two interrupt requests may be in error by ± 6 Φ clock periods although the cumulative error over many intervals is zero. The prescaler and Timer generate precise intervals for setting the timer interrupt request latch but the time out may occur at any time within a machine cycle. (There are two types of machine cycles: short cycles which consist of 4 Φ clock periods and long cycles which consist of 6 Φ clock periods. In the multi-chip F8 family there is a signal called the WRITE clock which corresponds to a machine cycle). Interrupt requests are synchronized with the internal WRITE clock thus giving rise to the possible ± 6 Φ error. Additional errors may arise due to the interrupt request occurring while a privileged instruction or multicycle instruction is being executed. Nevertheless, for most applications all of the above errors are negligible, especially if the desired time intervall is greater than 1ms.

Pulse Width Measurement Mode

When ICP bit 4 is set (logic 1) and at least one prescale bit is set the Timer operates in the Pulse Width Measurement Mode. This mode is used for accurately measuring the duration of a pulse applied to the EXT INT pin. The Timer is stopped and the prescaler is reset whenever EXT INT is at its inactive level. The active level of EXT INT is defined by ICP bit 2; if cleared, EXT INT is active low; if set, EXT INT is active high. If ICP bit 3 is set, the prescaler and Timer will start counting when EXT INT transitions to the active level. When EXT INT returns to the inactive level the Timer then stops, the prescaler resets, and if ICP bit 0 is set an external interrupt request latch is set. (Unlike timer interrupts, external interrupts are not latched if the ICP Interrupt Enable bit is not set).

As in the Interval Timer Mode, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, the prescaler and ICP bit 1 function as previously described, and the Timer still functions as an 8-bit binary down counter with the timer interrupt request latch being set on the Timer's transition from H '01' to H 'N'. Note that the EXT INT pin has nothing to do with loading the Timer; its action is that of automatically starting and stopping the Timer and of generating external interrupts. Pulse widths longer than the prescale value times the modulo-N value are easily measured by using the timer interrupt service routine to store the number of timer interrupts in one or more scratchpad registers.

As for accuracy, the actual pulse duration is typically slightly longer than the measured value because the status of the prescaler is not readable and is reset when the Timer is stopped. Thus for maximum accuracy it is advisable to use a small division setting for the prescaler.

Event Counter Mode

When ICP bit 4 is cleared and all prescale bits (ICP bits 5, 6, and 7) are cleared the Timer operates in the Event Counter Mode. This mode is used for counting pulses applied to the EXT INT pin. If ICP bit 3 is set the Timer will decrement on each transition from the inactive level to the active level or the EXT INT pin. The prescaler is not used in this mode, but as in the other two timer modes, the timer may be read at any time, may be stopped at any time by clearing ICP bit 3, ICP bit 1 functions as previously described, and the timer interrupt request latch is set on the Timer's transition from H '01' to H 'N'.

Normally ICP bit 0 should be kept cleared in the Event Counter Mode; otherwise, external interrupts will be generated on the transition from the inactive level to the active level of the EXT INT pin.

For the Event Counter Mode the minimum pulse width required on EXT INT is 2 Φ clock periods and the minimum inactive time is 2 Φ clock periods; therefore, the maximum repetition rate is 500KHz.

Timer Emulation

For total software compatibility when expanding into a multi-chip configuration the MK3871 Peripheral Input/Output circuit should be used rather than the older MK3861 PIO. The MK3871 has the same improved Timer (binary count, readable, and three modes of operation rather than one) and ready strobe output as are on the MK3872.

External Interrupts

When the timer is in the Interval Timer Mode the EXT INT pin is available for non-timer related interrupts. If ICP bit 0 is set an external interrupt request latch is set when there is a transition from the inactive level to the active level of EXT INT. (EXT INT is an edge-triggered input). The interrupt request is latched until either acknowledged by the CPU section or until ICP bit 0 is cleared (unlike timer interrupt requests which remain latched even when ICP bit 1 is cleared). External interrupts are handled in the same fashion when the Timer is in the Pulse Width Measurement Mode or in the Event Counter Mode, except that only in the Pulse Width Measurement Mode the external interrupt request latch is set on the trailing edge of EXT INT, that is, on the transition from the active level to the inactive level.

Interrupt Handling

When either a timer or an external interrupt request is communicated to the CPU section of the 3872, it will be acknowledged and processed at the completion of the first non-privileged instruction if the Interrupt Control Bit of the Status Register is set. If the Interrupt Control Bit is not set, the interrupt request will continue until either the Interrupt Control Bit is set and the CPU section acknowledges the interrupt or until the interrupt request is cleared as previously described.

If there is both a timer interrupt request and an external interrupt request when the CPU section starts to process the requests, the timer interrupt is handled first.

When an interrupt is allowed the CPU section will request that the interrupting element pass its interrupt vector address to the Program Counter via the data bus. The vector address for a timer interrupt is H '020'. The vector address for external interrupts is H '0A0'. After the vector address is passed to the Program Counter, the CPU section sends an acknowledge signal to the appropriate interrupt request latch which clears that latch. The execution of the interrupt service routine will then commence. The return address of the original program is automatically saved in the Stack Register, P.

The Interrupt Control Bit of W (Status Register) is automatically reset when an interrupt request is acknowledged. It is then the programmer's responsibility to determine when ICB will again be set (by executing an EI instruction). This action prevents an interrupt service routine from being interrupted unless the programmer so desires.

External Reset

When $\overline{\text{RESET}}$ is taken low the content of the Program Counter is pushed to the Stack Register and then the Program Counter and the ICB bit of the W Status Register are cleared. The original Stack Register content is lost. Ports 4, 5, 6 and 7 are loaded with H '00'. The contents of all other registers and ports are unchanged. When power is first applied all ports and registers are undefined until a reset is performed. When RESET is taken high the first program instruction is fetched from ROM location H '000'.

When an external reset of the 3872 occurs, P0 is pushed into P and the old contents of P are lost. It must be noted that an external reset is recognized at the start of a machine cycle and not necessarily at the end of an instruction. Thus if the 3872 is executing a multi-cycle instruction, that instruction is not completed and the contents of P upon reset may not necessarily be the address of the instruction that would have been executed next. It may, for example, point to an immediate operand if the reset occurred during the second cycle of a LI or CI instruction. Additionally, several instructions (JMP, PI, PK, LR P0, Q) as well as the interrupt acknowledge sequence modify P0 in parts. That is, they alter P0 by first loading one part then the other and the entire operation takes more than one cycle. Should reset occur during this modification process the value pushed into P will be part of the old P0 (the as yet unmodified part) and part of the new P0 (already modified part). Thus care should be taken (perhaps by external gating) to insure that reset does not occur at an undesirable time if any significance is to be given to the contents of P after a reset occurs.

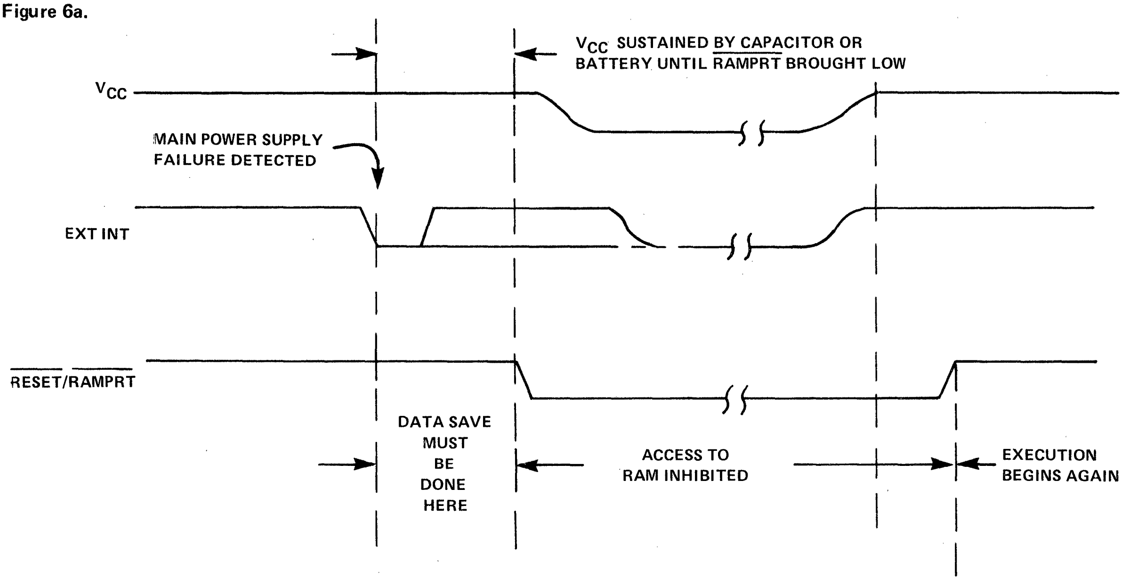
Test Logic

Special test logic is implemented to allow access to the internal main data bus for test purposes.

In normal operation the TEST pin is unconnected or is connected to GND. When TEST is placed at a TTL level (2.0V to 2.6V) Port 4 becomes an output of the internal data bus and Port 5 becomes a wired-OR input to the internal data bus. The data appearing on the Port 4 pins is logically true whereas input data forced on Port 5 must be logically false. When TEST is placed at high level (6.0V to 7.0V), the ports act as above and additionally the 2K x 8 program ROM is prevented from driving the data bus. In this mode operands and instructions may be forced externally through Port 5 instead of being accessed from the program ROM. When TEST is in either the TTL state or the high state, $\overline{\text{STROBE}}$ ceases its normal function and becomes a machine cycle clock (identical to the F8 multi-chip system WRITE clock except inverted).

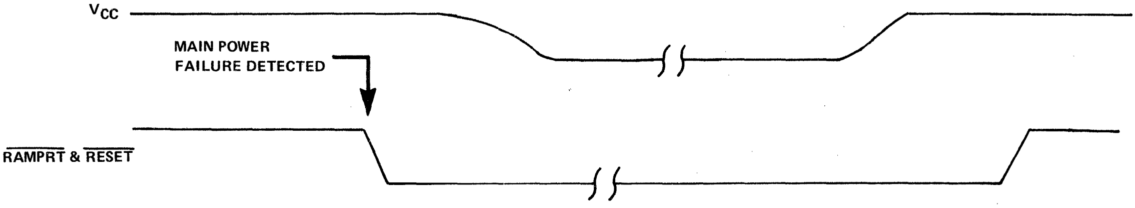
SAVE ROUTINE REQUIRED, $V_{SB} \geqslant 2.2$ VOLTS

Figure 6a.


V_{CC} SUSTAINED BY CAPACITOR OR BATTERY UNTIL $\overline{RAMPRT}$ BROUGHT LOW
V_{CC}
MAIN POWER SUPPLY FAILURE DETECTED
EXT INT
$\overline{RESET}$/$\overline{RAMPRT}$
DATA SAVE MUST BE DONE HERE
ACCESS TO RAM INHIBITED
EXECUTION BEGINS AGAIN

NO SAVE ROUTINE REQUIRED, $V_{SB} \geqslant 2.2$ VOLTS

Figure 6b.


V_{CC}
MAIN POWER FAILURE DETECTED
$\overline{RAMPRT}$ & $\overline{RESET}$

Timing complexities render the capabilities associated with the TEST pin impractical for use in a user's application, but these capabilities are thoroughly sufficient to enable a rapid method for thoroughly testing the 3872.

STANDBY POWER OPTION

If the standby power option has not been selected Port 0-bit 0 and 1 are readable and writeable.

If the standby power option is selected Port 0-Bit 1 is readable only. Port 0-Bit 0 remains readable and writable although it is not connected to a package pin. The standby power source (V_{SB}) is connected to Pin 4.

A .01 μF capacitor must be connected to Pin 3. The purpose of the capacitor is to decouple noise coupled to the substrate of the circuit when V_{CC} is switched off and on. It is recommended that Nickel-Cadmium batteries (typical voltage of 3 series cells = 3.6V) be used for standby power, since the MK3872 can automatically trickle charge the three Ni-Cad's. If more than three cells in series are used, the charging circuit must be provided outside the MK3872. Whenever $\overline{\text{RESET-RAMPRT}}$ is brought low, the standby RAM (64x8 bit words in PO/DC address space, 4032 to 4095_{10} or FCO to FFF_{16}) is placed in a protected state. Also the RAM itself is switched from V_{CC} power to the V_{SB} power. Two modes of powering down are recommended. In the first mode, the processor must be interrupted early enough to save all necessary data before the V_{CC} falls below the minimum level. After the save is done, $\overline{\text{RESET}}$ can fall. This prevents any further access of the RAM; V_{CC} may now fall. As the power comes up, the $\overline{\text{RESET/RAMPRT}}$ signal should be held low until V_{CC} is above the minimum level.

The second mode may be used if a special save data routine is not needed. The $\overline{\text{EXT INTERRUPT}}$ need not be used and the only requirement to save the RAM data is that $\overline{\text{RESET-RAMPRT}}$ be low before V_{CC} drops below 4.5V. For example if a few key variables are to be stored in RAM and it is desired that these be saved during a loss of power, two copies of each variable are kept with an associated flag, thus no interrupt and save routine is necessary. The method of updating a variable is as follows:

- Clear Flag Word 1
- Update Variable (Copy 1)
- Set Flag Word 1
- Clear Flag Word 2
- Update Variable (Copy 2)
- Set Flag Word 2

Now execution may terminate at any time, even during the update of a variable or flag word, causing that byte in RAM to be bad data. There is always a good data byte which contains either the most recent or next most recent value of the variable. Any copy of the variable where the flag word is "set" is a good data byte. While this method significantly encumbers the data storage process, it eliminates the need for a power fail interrupt which both reduces external circuitry and leaves the external interrupt pin completely free for other use.

3872 Clocks

The time base for the 3872 may originate from one of five sources. There are four external modes and one internal mode.

If both XTL 1 and XTL 2 are grounded, the 3872 will activate its internal oscillator.

The four external configurations are shown in Figure 7. There is an internal 20pF capacitor between XTL 1 and GND and an internal 20pF capacitor between XTL 2 and GND. Thus external capacitors are not neccessarily required. In all external clock modes the external time base frequently is divided by two to form the internal Φ clock.

Crystal Selection

The use of a crystal as the time base is highly recommended as the frequency stability and reproducability from system to system is unsurpassed. The 3872 has an internal divide by two to allow the user of inexpensive and widely available TV Color Burst Cyrstals (3.58MHz). The following crystal parameters and vendors are suggested for 3872 applications:

Parameters

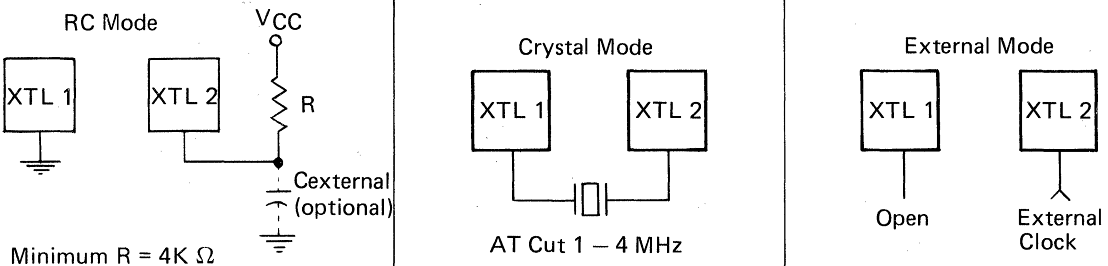
a) Parallel Resonance, Fundamental Mode AT-Cut
b) Frequency Tolerance measured with 18pF load (0.1% accuracy). Drive level 10mW.
c) Shunt Capacitance (Co) = 7pF max.
d) Series Resistance (Rs)

		Holder
f = 1MHz	Rs = 550 ohms max.	HC-6
f = 2MHz	Rs = 300 ohms max.	HC-33
f = 3MHz	Rs = 150 ohms max.*	HC-6
f = 3.58MHz	Rs = 150 ohms max.	HC-18
f = 4MHz	Rs = 150 ohms max.	HC-25 HC-33

*HC-18 or HC-25 holder may not be available at 3MHz.

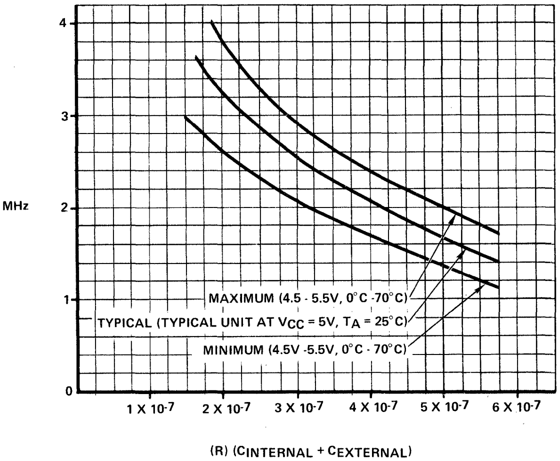
CLOCK CONFIGURATIONS

Figure 7



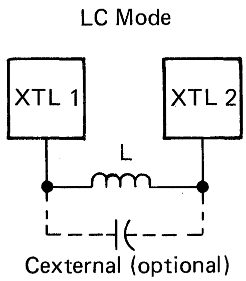
C = 26.5 pF ±2.6pF + Cexternal

FREQUENCY VRS RC



UNIT TO UNIT VARIATION = ± 12%
VARIATION FROM 4.5 to 5.5V
REFERENCED TO 5V = +7% -4%
VARIATION FROM 0°C TO 70°C
REFERENCED TO 25°C = +6% -9%

TOTAL VARIATION NOT CONSIDERING
VARIATION IN EXTERNAL COMPONENTS = ± 25%



Minimum L = 0.1 mH
Minimum Q = 40

Maximum Cexternal = 30pF

C = 10pF ± 1.3pF + Cexternal

$$f \cong \frac{1}{2\pi\sqrt{LC}}$$

NOTE: The stray capacitance across the inductor must be included as Cexternal in all calculations.

Suggested Crystal Vendors

a) Electro-Dynamics
5625 Foxridge Drive
Mission, Kansas 66201
913-262-2500

b) CRYSTEK
1000 Crystal Drive
Ft. Myers, Florida 33901
813-936-2109

c) W.T. Liggett Corp.
1500 Worcester Rd.
Section 30
Framingham, MA 01701
617-620-1150

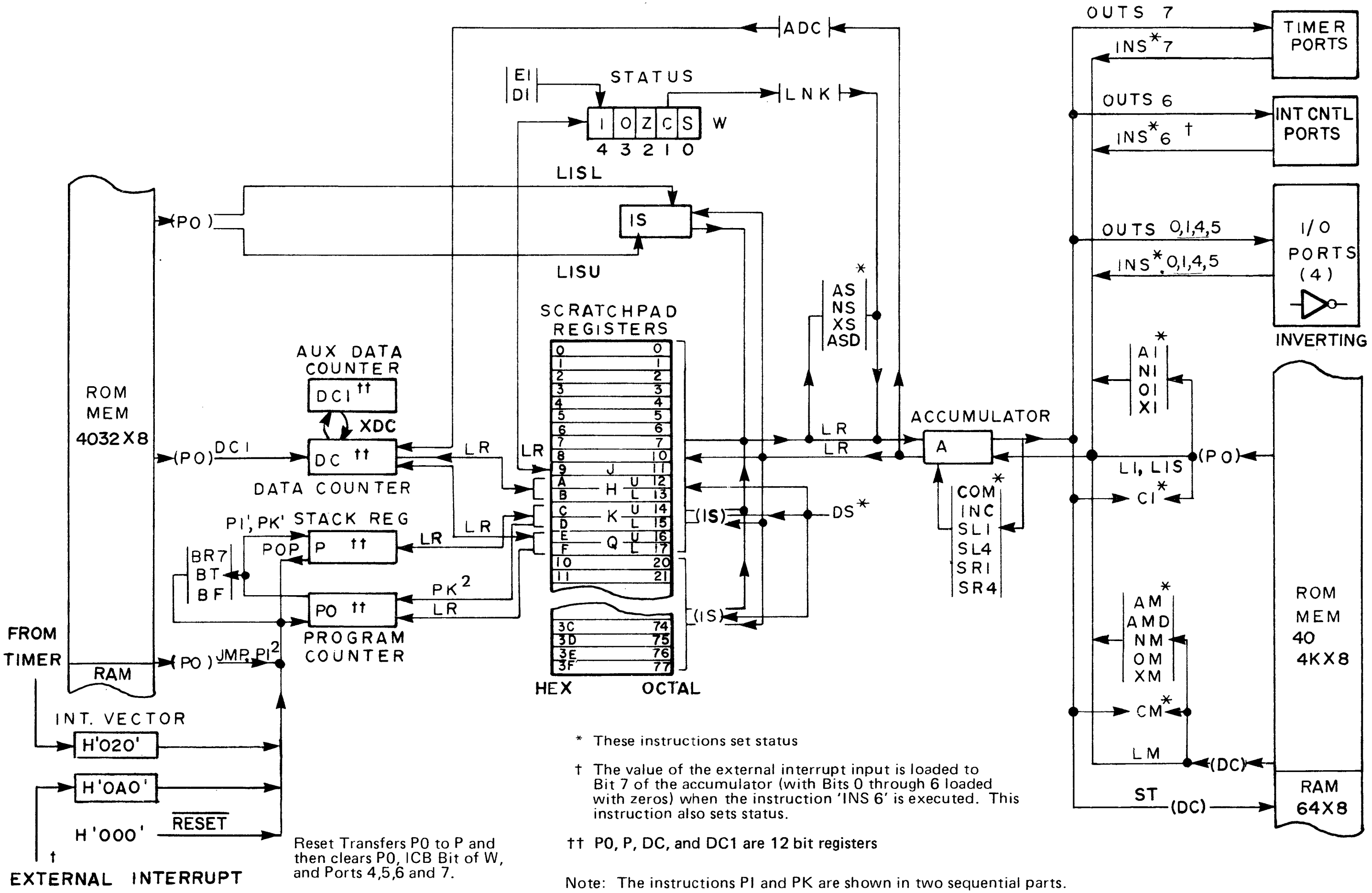
d) Erie Frequency Control
453 Lincoln Street
Carlisle, Penn 17013
717-249-2232

e) Electronic Crystals Corp.
1153 Southwest Blvd.
Kansas City, Kansas 66103
913-262-1274

f) M-TRON Industries
P.O. Box 630
100 Douglas Avenue
Yankton, South Dakota
605-665-9321

MK3872 PROGRAMMING MODEL
Figure 8

MK3872 PROGRAMMING MODEL
Figure 8



Reset Transfers PO to P and then clears PO, ICB Bit of W, and Ports 4,5,6 and 7.

* These instructions set status

† The value of the external interrupt input is loaded to Bit 7 of the accumulator (with Bits 0 through 6 loaded with zeros) when the instruction 'INS 6' is executed. This instruction also sets status.

†† PO, P, DC, and DC1 are 12 bit registers

Note: The instructions PI and PK are shown in two sequential parts. (PI1, PI2 and PK1, PK2).

INSTRUCTION EXECUTION

This section details the timing and execution of the 3872 instruction set. The 3872 executes the entire F8 instruction set with exact F8 timing.

F8 INSTRUCTION SET

ACCUMULATOR GROUP INSTRUCTIONS

	MNEMONIC			MACHINE		CYCLES		µS		STATUS BITS		
OPERATION	OP CODE	OPERAND	FUNCTION	CODE	BYTES	SHORT	LONG	(2MHzΦ)	OVR	ZERO	CRY	SIGN
Add Carry	LNK		A←(A) + CRY	19	1	1		2	1/0	1/0	1/0	1/0
Add Immediate	AI	ii	A←(A) + H 'ii'	24ii	2	1	1	5	1/0	1/0	1/0	1/0
And Immediate	NI	ii	A←(A)∧H 'ii'	21ii	2	1	1	5	0	1/0	0	1/0
Clear	CLR		A←H'00'	70	1	1		2	–	–	–	–
Compare Immediate	CI	ii	H'ii'+ (A) + 1	25ii	2	1	1	5	1/0	1/0	1/0	1/0
Complement	COM		A←(A)+H'FF'	18	1	1		2	0	1/0	0	1/0
Exclusive or Immediate	XI	ii	A←(A)+H'ii'	23ii	2	1	1	5	0	1/0	0	1/0
Increment	INC		A←(A) + 1	1F	1	1		2	1/0	1/0	1/0	1/0
Load Immediate	LI	ii	A←H'ii'	20ii	2	1	1	5	–	–	–	–
Load Immediate Short	LIS	i	A←H' 0i'	7i	1	1		2				
OR Immediate	OI	ii	A←(A) V H 'ii'	22ii	2	1	1	5	0	1/0	0	1/0
Shift Left One	SL	1	Shift Left 1	13	1	1		2	0	1/0	0	1/0
Shift Left Four	SL	4	Shift Left 4	15	1	1		2	0	1/0	0	1/0
Shift Right One	SR	1	Shift Right 1	12	1	1		2	0	1/0	0	1
Shift Right Four	SR	4	Shift Right 4	14	1	1		2	0	1/0	0	1

BRANCH INSTRUCTIONS In all conditional branches P0←(P0) + 2 if the test condition is not met. Execution is complete in 3 short cycles.

	MNEMONIC			MACHINE		CYCLES		µS		STATUS BITS		
OPERATION	OP CODE	OPERAND	FUNCTION	CODE	BYTES	SHORT	LONG	(2MHzΦ)	OVR	ZERO	CRY	SIGN
Branch on Carry	BC	aa	P0←(P0) +1+ H'aa' if CRY = 1	82aa	2	2	1	7	—	—	—	—
Branch on Positive	BP	aa	P0←(P0) + 1 + H'aa ' if SIGN = 1	81aa	2	2	1	7	—	—	—	—
Branch on Zero	BZ	aa	P0←(P0) + 1 + H'aa' if Zero = 1	84aa	2	2	1	7	—	—	—	—
Branch on True	BT t=TEST CONDITION 2^2 ZERO, 2 CRY, 2^0 SIGN	taa	P0←(P0) + 1 + H'aa ' if any test is true	8taa	2	2	1	7	—	—	—	—
Branch If Negative	8M	aa	P0←(P0)+1+ H'aa ' if SIGN =0	91aa	2	2	1	7	—	—	—	—
Branch if No Carry	BNC	aa	P0←(P0) +1+ H'aa ' if CARRY = 0	92aa	2	2	1	7	—	—	—	—
Branch if No Overflow	BNO	aa	P0←(P0) +1+H'aa ' if OVR = 0	98aa	2	2	1	7	—	—	—	—
Branch if Not Zero	BNZ	aa	P0←(P0)+1+ H'aa ' if ZERO = 0	94aa	2	2	1	7	—	—	—	—
Branch if False Test	BF t=TEST CONDITION 2^3 OVF, 2^2 ZERO, 2^1 CRY, 2^0 SIGN	taa	P0←(P0) +1+ H'aa ' if all false test bits	9taa	2	2	1	7	—	—	—	—
Branch if ISAR (Lower) ≠7	BR7	aa	P0←(P0)+1+ H'aa ' if ISARL ≠7	87aa	2	1	1	5	—	—	—	—
			P0←(P0)+2 if ISARL =		2	2		4	—	—	—	—
Branch Relative	BR	aa	P0← (P0)+1+ H'aa '	90aa	2	2	1	7	—	—	—	—
Jump*	JMP	aaaa	P0←H'aaaa'	29aaaa	3	1	3	11	—	—	—	—

*Privileged instruction, Accumulator contents altered during execution JMP instruction.

MEMORY REFERENCE INSTRUCTIONS In all Memory Reference Instructions, the Data Counter is incremented

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	µS (2MHzΦ)	OVR	STATUS BITS ZERO	STATUS BITS CRY	SIGN
Add Binary	AM		A←(A) + [(DC)]	88	1	1	1	5	1/0	1/0	1/0	1/0
Add Decimal	AMD		A←(A) + [(DC)] • BCD Adjust	89	1	1	1	5	?	?	1/0	?
AND	NM		A←(A) ∧ [(DC)]	8A	1	1	1	5	0	1/0	0	1/0
Compare	CM		[(DC)] + $\overline{\text{(A)}}$ + 1	8D	1	1	1	5	1/0	1/0	1/0	1/0
Exclusive OR	XM		A←(A)⊕[(DC)]	8C	1	1	1	5	0	1/0	0	1/0
Load	LM		A←[(DC)]	16	1	1	1	5	—	—	—	—
Logical OR	OM		A← (A) V·ᶦ(DC)]	8B	1	1	1	5	0	1/0	0	1/0
Store	ST		A→[(DC)]	17	1	1	1	5	—	—	—	—

ADDRESS REGISTER GROUP INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	µS (2MHzΦ)	OVR	STATUS BITS ZERO	STATUS BITS CRY	SIGN
Add to Data Counter	ADC		DC←(DC) + (A)	8E	1	1	1	5	—	—	—	—
Call to Subroutine*	PK		P0U←(r12); P0L←(r13), P←(P0)	OC	1	1	2	8	—	—	—	—
Call to Subroutine Immediate*	PI	aaaa	P← (P0), P0←H'aaaa	28aaaa	3	2	3	13	—	—	—	—
Exchange DC	XDC		(DC)⇄(DC1)	2C	1	2		4	—	—	—	—
Load Data Counter	LR	DC,Q	DCU←(r14); DCL←(r15)	OF	1	1	2	8	—	—	—	—
Load Data Counter	LR	DC'H	DCU←(r10); DCL←(r11)	10	1	1	2	8	—	—	—	—
Load DC Immediate	DCI	aaaa	DC H'aaaa'	2Aaaaa	3	3	2	12	—	—	—	—
Load Program Counter	LR	P0,Q	P0U←(r14); P0L←(r15)	OD	1	1	2	8	—	—	—	—
Load Stack Register	LR	P,K	PU←(r12); PL←(r13)	09	1	1	2	8	—	—	—	—
Return from Subroutine*	P0P		P0←(P)	1C	1	2		4	—	—	—	—
Store Data Counter	LR	Q,DC	r14←(DCU); r15←(DCL)	OE	1	1	2	8	—	—	—	—
Store Data Counter	LR	H,DC	r10←(DCU); r11←(DCL)	11	1	1	2	8	—	—	—	—
Store Stack Register	LR	K,P	r12←(PU); r13←(PL)	08	1	1	2	8	—	—	—	—

SCRATCHPAD REGISTER INSTRUCTIONS (Refer to Scratchpad Addressing Modes)

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	µS (2MHzΦ)	OVR	STATUS BITS ZERO	STATUS BITS CRY	SIGN
Add Binary	AS	r	A←(A)+ (r)	Cr	1	1		2	1/0	1/0	1/0	1/0
Add Decimal	ASD	r	A←(A) + (r)	Dr	1	2		4	?	?	1/0	?
Decrement	DS	r	r←(r) + H'FF'	3r	1		1	3	1/0	1/0	1/0	1/0
Load	LR	A,r	A←(r)	4r	1	1		2	—	—	—	—
Load	LR	A, KU	A←(r12)	00	1	1		2	—	—	—	—
Load	LR	A, KL	A←(r13)	01	1	1		2	—	—	—	—
Load	LR	A, QU	A←(r14)	02	1	1		2	—	—	—	—
Load	LR	A, QL	A←(r15)	03	1	1		2	—	—	—	—
Load	LR	r, A	r←(A)	5r	1	1		2	—	—	—	—
Load	LR	KU, A	r12←(A)	04	1	1		2	—	—	—	—
Load	LR	KL, A	r13←(A)	05	1	1		2	—	—	—	—
Load	LR	QU, A	r14←(A)	06	1	1		2	—	—	—	—
Load	LR	QL,A	r15←(A)	07	1	1		2	—	—	—	—
And	NS	r	A← (A) ∧ (r)	Fr	1	1		2	0	1/0	0	1/0
Exclusive Or	XS	r	A←(A) + (r)	Er	1	1		2	0	1/0	0	1/0

*Privileged instruction, Accumulator contents altered during execution of PI instruction.

MISCELLANEOUS INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	 LONG	µS (2MHzΦ)	 OVR	STATUS BITS ZERO	 CRY	 SIGN
Disable Interrupt	DI		RESET ICB	1A	1	1		2	—	—	—	—
Enable Interrupt *	EI		SET ICB	1B	1	1		2	—	—	—	—
Input	IN	04,05,06,07	A←(Input Port aa)	26aa	2	1	2	8	0	1/0	0	1/0
Input Short	INS	0, 1	A←(Input Port 0 or 1)	A0,A1	1	2		4	0	1/0	0	1/0
Input Short	INS	4,5,6,7	A←(Input Port a)	Aa	1	1	2	8	0	1/0	0	1/0
Load ISAR	LR	IS,A	IS←(A)	0B	1	1		2	—	—	—	—
Load ISAR Lower	LISL	bbb	ISL←bbb	6(1bbb)**	1	1		2	—	—	—	—
Load ISAR Upper	LISU	bbb	ISU←bbb	6(0bbb)**	1	1		2	—	—	—	—
Load Status Register*	LR	W,J	W←(r9)	1D	1	2		4	1/0	1/0	1/0	1/0
No Operation	NOP		P0←(P0) + 1	2B	1	1		2	—	—	—	—
Output*	OUT	04,05,06,07	Output Port aa←(A)	27aa	2	1	2	8	—	—	—	—
Output Short	OUTS	0, 1	Output Port 0 or 1←(A)	B0, B1	1	2		4	—	—	—	—
Output Short*	OUTS	4,5,6,7	Output Port a←(A)	Ba	1	1	2	8	—	—	—	—
Store ISAR	LR	A,IS	A←(IS)	0A	1	1		2	—	—	—	—
Store Status Reg	LR	J,W	r9←(W)	1E	1	1		2	—	—	—	—

*Privileged instruction

**b = 1 bit immediate operand

NOTES.

Lower case denotes variables specified by programmer

Function Definitions

←	is replaced by
( )	the contents of
$\overline{(\ )}$	Binary "1's" complement of
+	Arithmetic Add (Binary or Decimal)
⊕	Logical "OR" exclusive
∧	Logical "AND"
∨	Logical "OR" inclusive
H' '	Hexadecimal digit
[( )]	Contents of memory specified by ( )
a	Address Variable (four bits)
A	Accumulator
b	One bit immediate operand
DC	Data Counter (Indirect Address Register)
DC1	Data Counter 1 (Auxiliary Data Counter)
DCL	Least significant 8 bits of Data Counter Addressed
DCU	Most significant 8 bits of Data Counter Addressed
H	Scratchpad Register 10 and 11
i	Immediate operand (four bits)
ICB	Interrupt Control Bit
IS	Indirect Scratchpad Address Register
ISL	Least Significant 3 bits of ISAR
ISU	Most Significant 3 bits of ISAR
J	Scratchpad Register 9
K	Registers 12 and 13
KL	Register 13
KU	Register 12
P0	Program Counter
P0L	Least Significant 8 bits of Program Counter
P0U	Most Significant 8 bits of Program Counter
P	Stack Register
PL	Least Significant 8 bits of Program Counter
PU	Most Significant 8 bits of Active Stack Register
Q	Registers 14 and 15
QL	Register 15
QU	Register 14
r	Scratchpad Register (any address 0 thru B) (See Below)
W	Status Register

Scratchpad Addressing Modes Using IS. (r ≠ 0 thru B)

r=H'C'	Register Addressed by IS is (Unmodified)
r=H'D'	Register Addressed by IS is Incremented
r=H'E'	Register Addressed by IS is Decremented
r=H'F'	Illegal OP Code.

Status Register

—	No change in condition
1/0	is set to "1" or"0" depending on conditions
CRY	Carry Flag
OVR	Overflow Flag
SIGN	Sign of Result Flag
ZERO	Zero Flag

ELECTRICAL SPECIFICATIONS
ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	0°C to 70°C
Storage Temperature	–65°C to +150°C
Voltage on Any Pin With Respect To Grouns (except open drain pins)	–1.0V to +7V
Voltage On Open Drain Pins	–1.0V to +13.5V
Power Dissipation	1.5W

A.C. CHARACTERISTICS – See Figure 9 and 10 for Timing Diagrams

T_A = 0°C to 70°C, V_{CC} = 5V ±10%, I/O POWER DISSIPATION ≤100mW

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
	t_O(INT)	Time Base Period, internal oscillator	250	1000	ns	4MHz - 1.0MHz
XTL 1	t_O(EX)	Time base period, all external modes	250	1000	ns	4MHz-1MHz
XTL 2	$t_{EX(H)}$	External Clock Pulse Width High	90	700	ns	
	$t_{EX(L)}$	External Clock Pulse Width Low	100	700	ns	
Φ	t_Φ	Internal Φ Clock Period	$2t_O$			
WRITE	tw	Internal WRITE Clock Period	$4t_\Phi$ $6t_\Phi$			Short Cycle Long Cycle
I/O	$t_{dI/O}$	Output delay from internal WRITE Clock	0	1000	ns	50pF plus one TTL load
	$t_{sI/O}$	Input Setup time to WRITE Clock	1000		ns	
$\overline{\text{STROBE}}$	$t_{I/O\text{-}s}$	Output valid to STROBE Delay	3tΦ -1000	3tΦ +250		I/O load = 50pF + 1 TTL STROBE Load= 50pF + 3 TTL
	tsl	$\overline{\text{STROBE}}$ Low Time	8tΦ -250	12tΦ +250	ns	
$\overline{\text{RESET}}$	t_{RH}	$\overline{\text{RESET}}$ Hold Time, Low	6tΦ +750		ns	
EXT INT	t_{EH}	EXT INT Hold Time,	6tΦ + 750		ns	To trigger interrupt
		Active and Inactive State	2tΦ			To trigger timer

CAPACITANCE

T_A = 25° C, f=2MHz

SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input Capacitance: I/O Ports, $\overline{\text{RESET - RAMPRT}}$, EXTINT, TEST		7	pF	Unmeasured Pins Grounded
C_{XTL}	Input Capacitance: XTL1, XTL2	23.5	29.5	pF	

DC CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 10%, I/O POWER DISSIPATION ≤100mW

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
I_{CC}	Power Supply Current		100	mA	Outputs Open
P_D	Power Dissipation		500	mW	Outputs Open
V_{IHEX}	External Clock Input High Level	2.4	5.8	V	
V_{ILHEX}	External Clock Input Low Current	−0.3	0.6	V	
I_{IHEX}	External Clock Input High Current		100	μA	$V_{IHEX} = V_{CC}$
I_{ILEX}	External Clock Input Low Current		-100	μA	$V_{ILEX} = V_{SS}$
V_{IH}	Input High Level Ports,$\overline{\text{RESET}}$[1], EXT INT[1]	2.0	5.8	V	
V_{IHOD}	Open Drain Input High Level	2.0	13.2	V	
V_{IL}	Input Low Level Ports, $\overline{\text{RESET}}$[1], EXT INT[1]	-0.3	0.8		
I_{IL}	Input Low Current Ports, $\overline{\text{RESET}}$[2], EXT INT[2]	-1.6		mA	V_{IL}=0.4V
I_L	Leakage Current Open drain ports, $\overline{\text{RAMPRT-RESET}}$[3], EXT INT[3]		+10 −5	μA	V_{IN}=13.2V V_{IN}=0.0V
I_{OH}	Output High Current Standard ports, $\overline{\text{RESET}}$[2] EXT INT[2]	−100 −30		μA μA	V_{OH}=2.4V V_{OH}=3.9V
I_{OHDD}	OUTPUT High Current Direct Drive Ports	−0.1		mA	V_{OH} = 2.4V
		−1.5		mA	V_{OH}=1.5V
			−8.5	mA	V_{OH}=.7V
I_{OL}	Output Low Current IO ports	1.8		mA	V_{OL}=0.4V
I_{OHS}	$\overline{\text{STROBE}}$ Output High Current	−300		μA	V_{OH}=2.4V
I_{OLS}	$\overline{\text{STROBE}}$ Output Low Current	5.0		mA	V_{OL} = 0.4V
V_{IHRPR}	Input High Level For RAM Protect Function To be effective.	1.9	5.8	V	Guaranteed .1V less than V_{IH} for $\overline{\text{RESET}}$
V_{ILRPR}	Input High Level For RAM Protect Function To be effective.	−0.3	0.4	V	Guaranteed .1V less than V_{IL} for $\overline{\text{RESET}}$

DC CHARACTERISTICS (Cont'd)

SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
V_{SB}	Standby V_{CC} for RAM	3.2	5.5	V	
I_{SB}	Standby current		6 3.7	mA mA	V_{SB}= 5.5 V_{SB}= 3.2
I_{CHARGE}	Trickle charge available on V_{SB} with V_{CC}=4.5 to 5.5	-.8	 -15	mA mA	V_{SB}=3.8V V_{SB}=3.2V
P_{DIO}	Power dissipated by I/O Pins[4]		600 60	mW mW	All Pins any one pin

* Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

1. $\overline{\text{RESET}}$ and EXT INT have internal Schmit triggers giving minimum .2V hysteresis.
2. $\overline{\text{RESET}}$ or EXT INT programmed with standard pull-up
3. $\overline{\text{RESET}}$ or EXT INT programmed without standard pull-up
4. Power dissipation for I/O pins is calculated by$\Sigma(V_{cc} - V_{IL})\,(|I_{IL}|) + \Sigma(V_{CC} - V_{OH})\,(|I_{OH}|) + \Sigma(V_{OL})\,(I_{OL})$

TIMER AC CHARACTERISTICS

Definitions:

Error = Indicated time value - actual time value

tpsc = tΦ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3)	±6tΦ
Cumulative interval error, free running (Note 3)	0
Error between two Timer reads (Note 2)	±(tpsc + tΦ)
Start Timer to stop Timer error (Notes 1,4)	+tΦ to –(tpsc +tΦ)
Start Timer to read Timer error (Notes 1,2)	–5tΦ to –(tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1,3)	–2tΦ to –8tΦ
Load Timer to stop Timer error (Note 1)	+tΦ to –(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1,2)	–5tΦ to –(tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1,3)	–2t Φ to –9tΦ

Pulse Width Measurement Mode:

Measurement accuracy (Note 4)	+t Φto –(tpsc +2t Φ)
Minimum pulse width of EXT INT pin	2tΦ

Event Counter Mode:

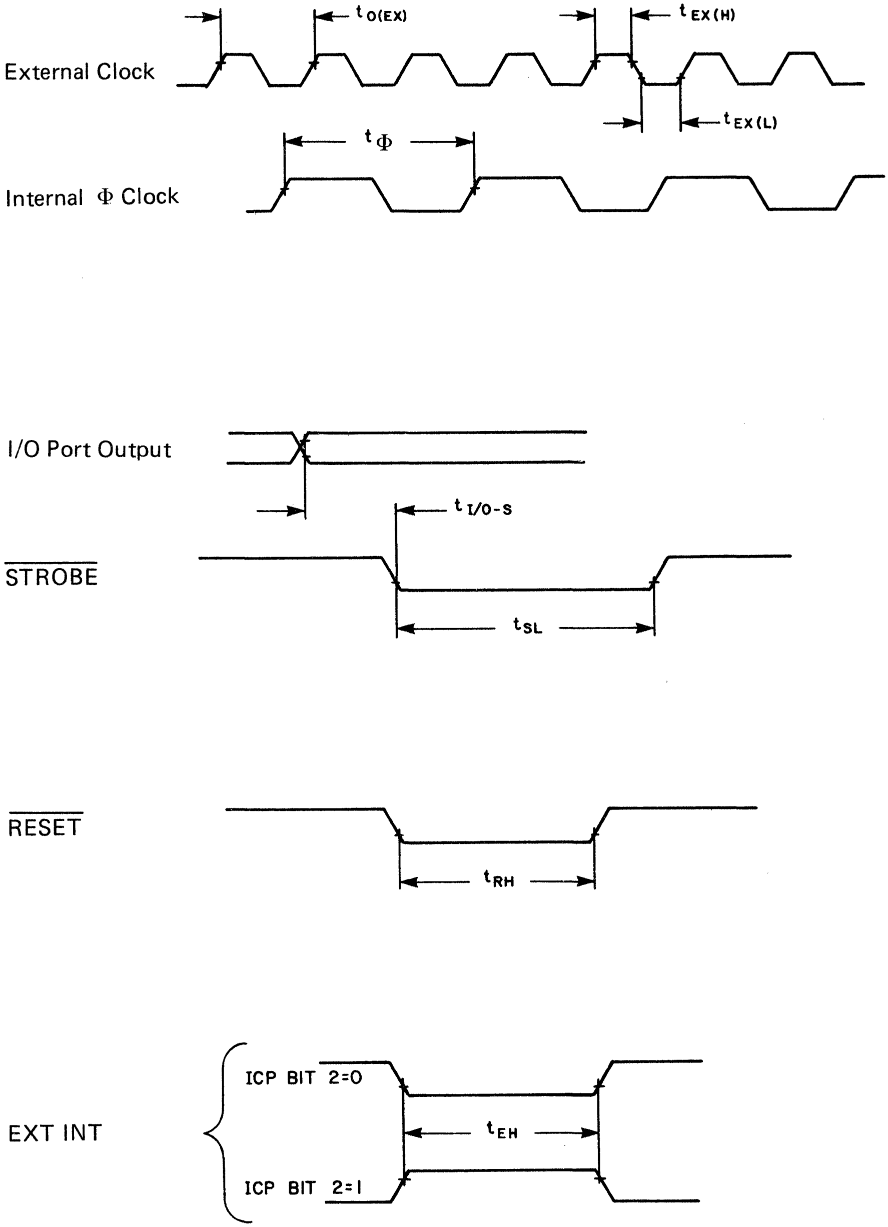
Minimum active time of EXT INT pin	2tΦ
Minimum inactive time of EXT INT pin	2tΦ

Notes:

1. All times which entail loading, starting, or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.

AC TIMING DIAGRAM

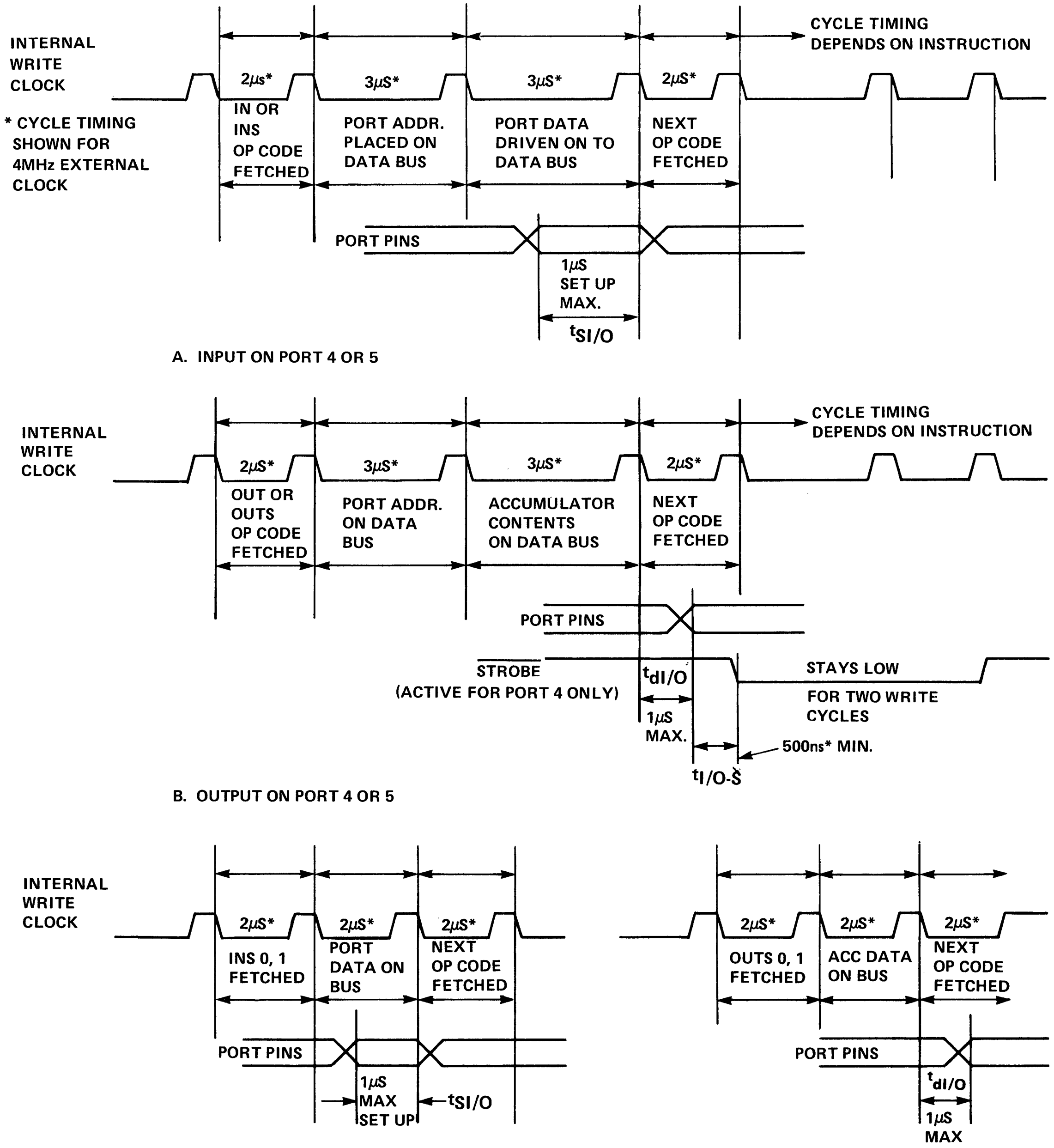
Figure 9



Note: All measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} max., or V_{OH} min.

INPUT/OUTPUT AC TIMING

Figure 10



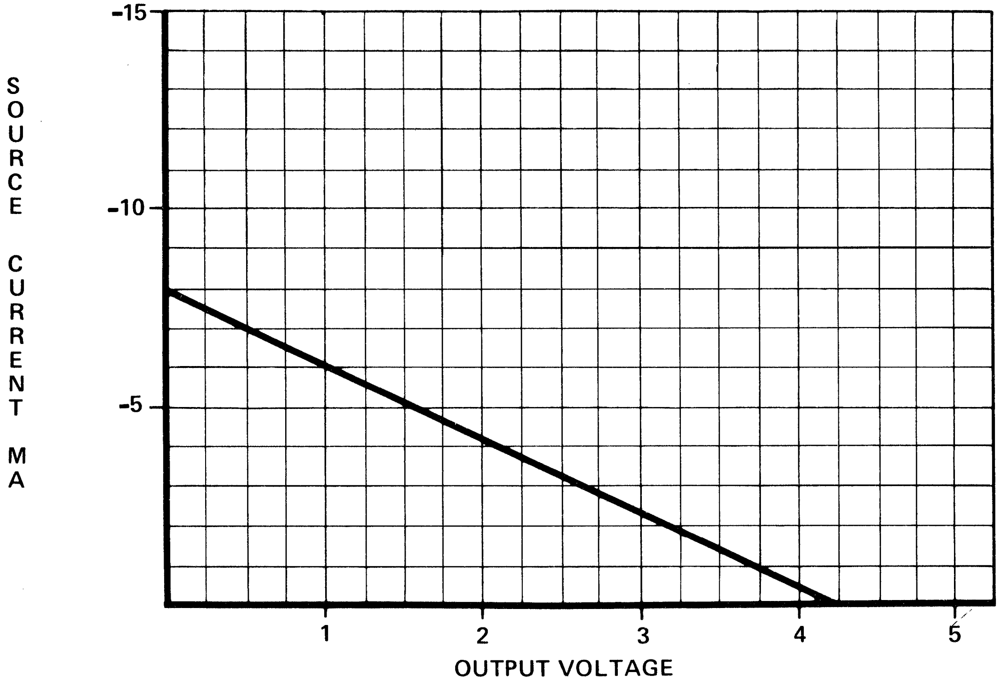
A. INPUT ON PORT 4 OR 5

B. OUTPUT ON PORT 4 OR 5

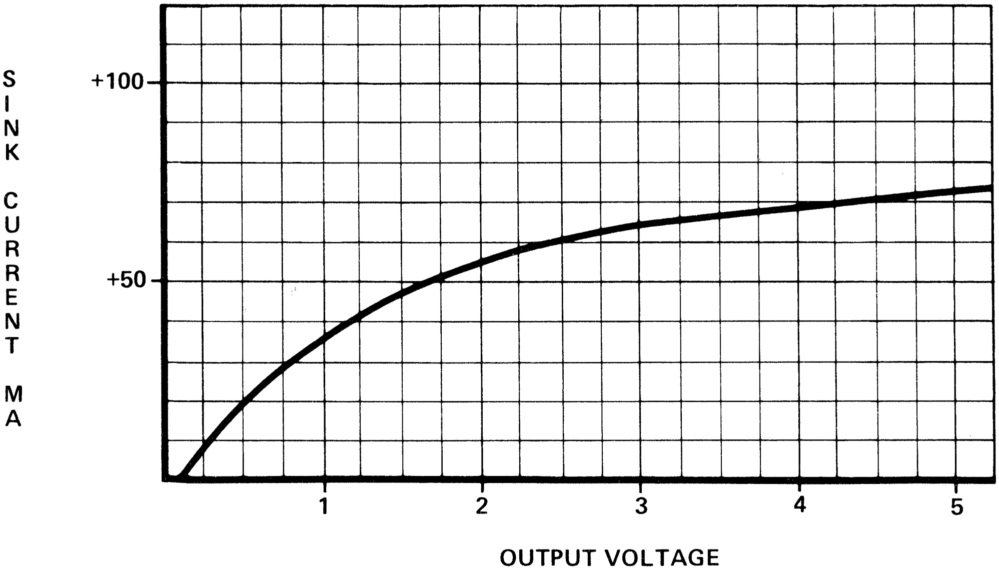
C. INPUT ON PORT 0 OR 1

D. OUTPUT ON PORT 0, 1

STROBE SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25° C)
Figure 11

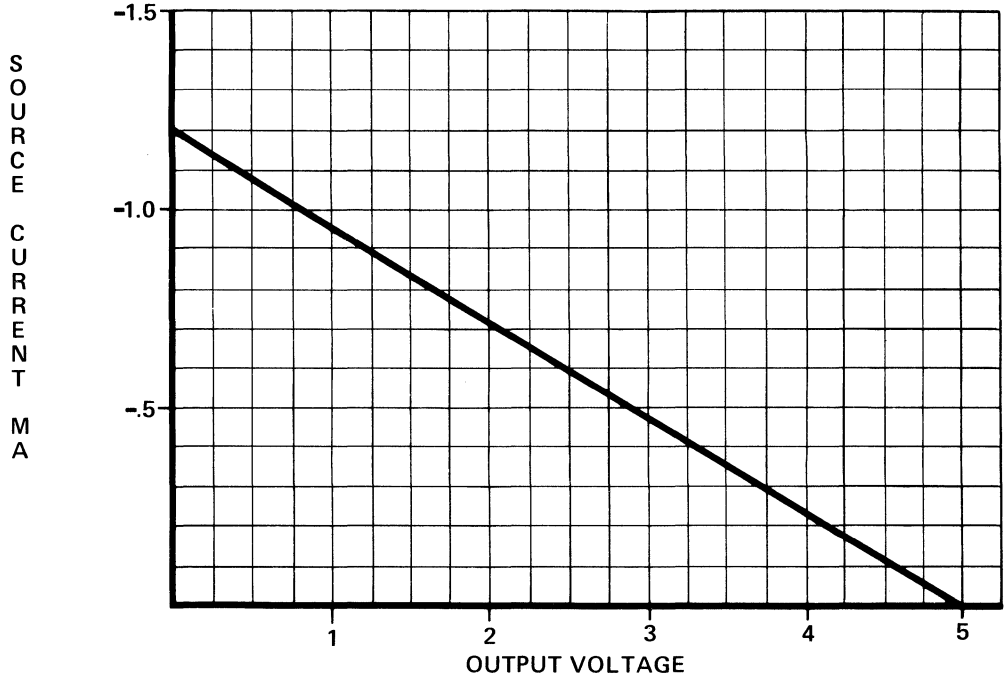


STROBE SINK CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25° C)
Figure 12



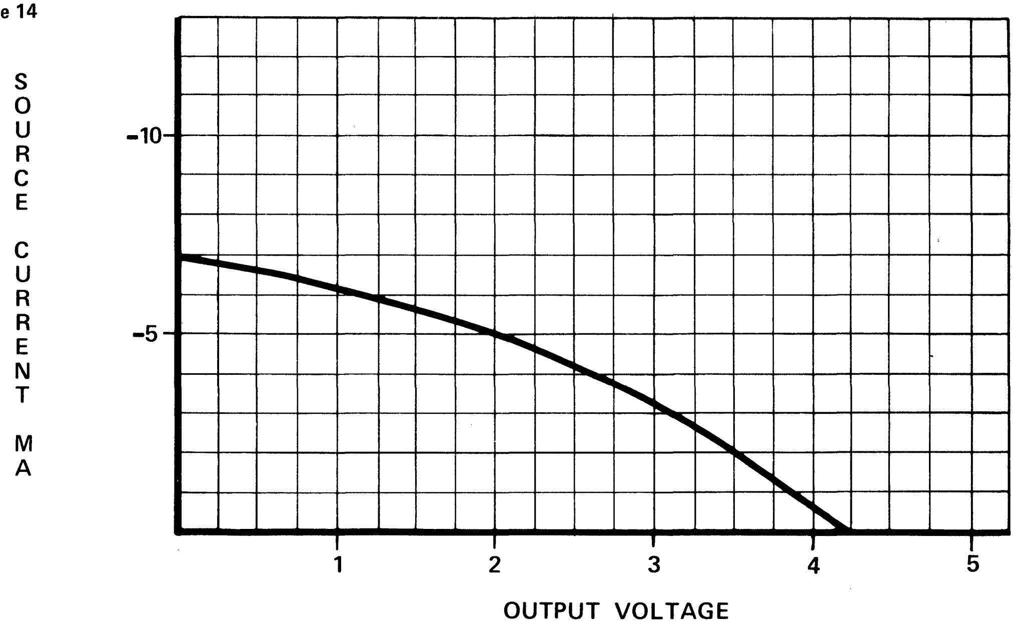
STANDARD I/O PORT SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 13



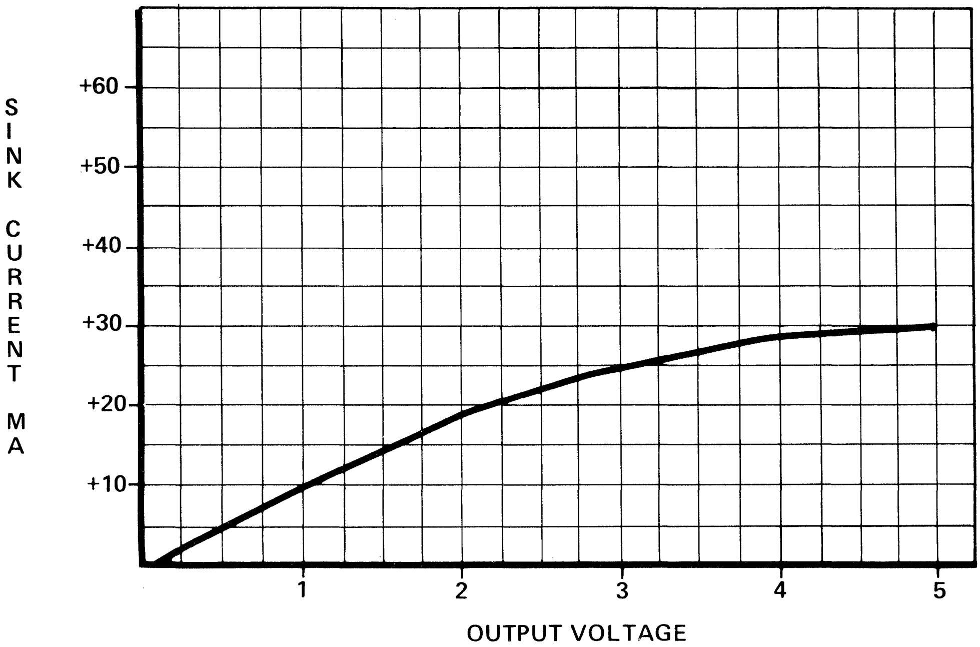
DIRECT DRIVE I/O PORT SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 14



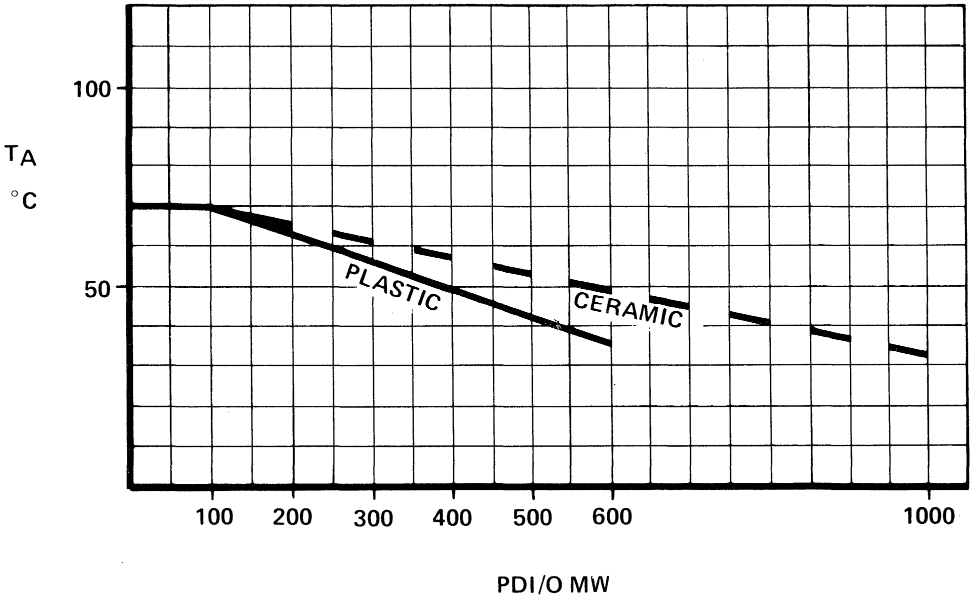
I/O PORT SINK CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 15



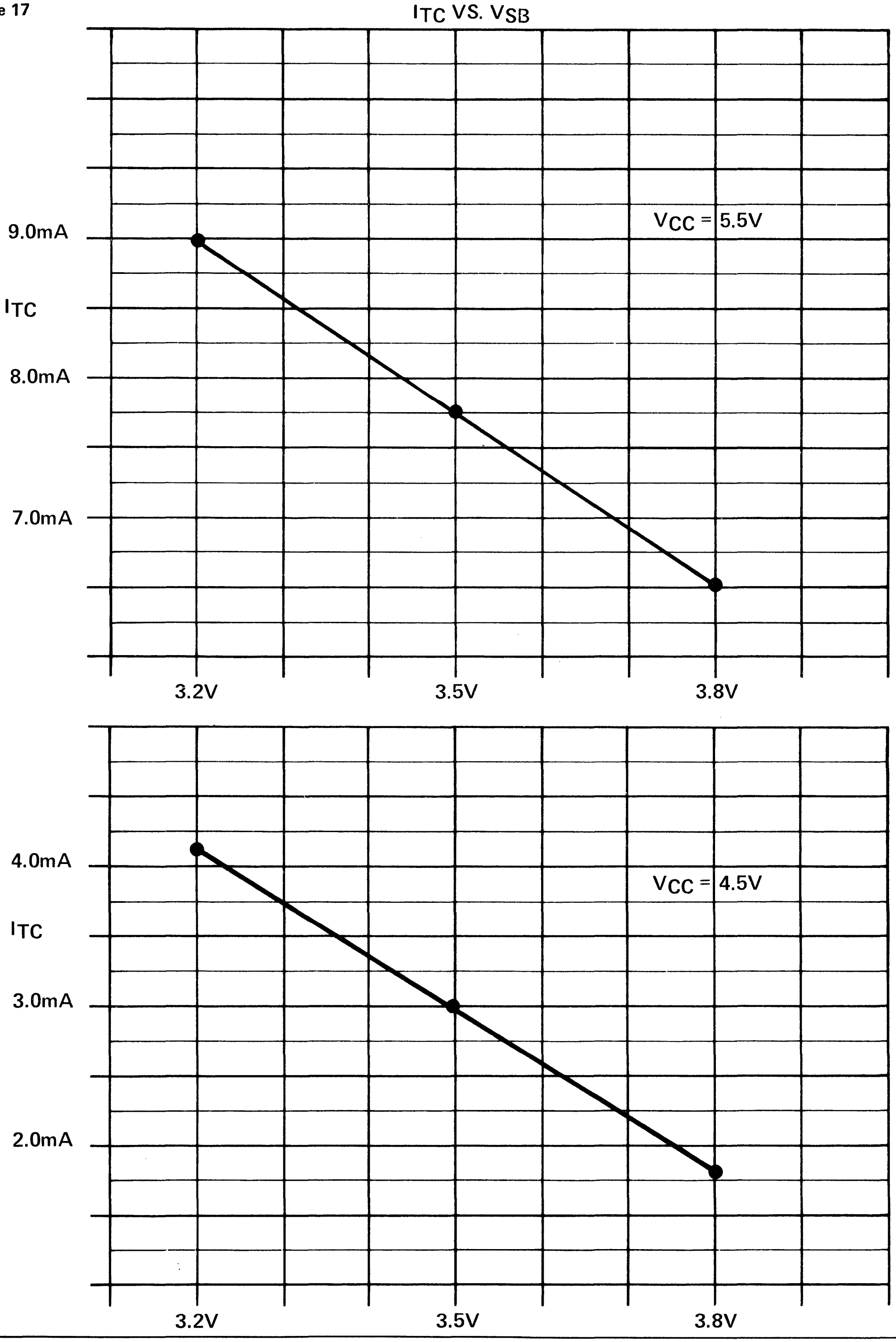
MAXIMUM OPERATING TEMPERATURE VS. I/O POWER DISIPATION

Figure 16

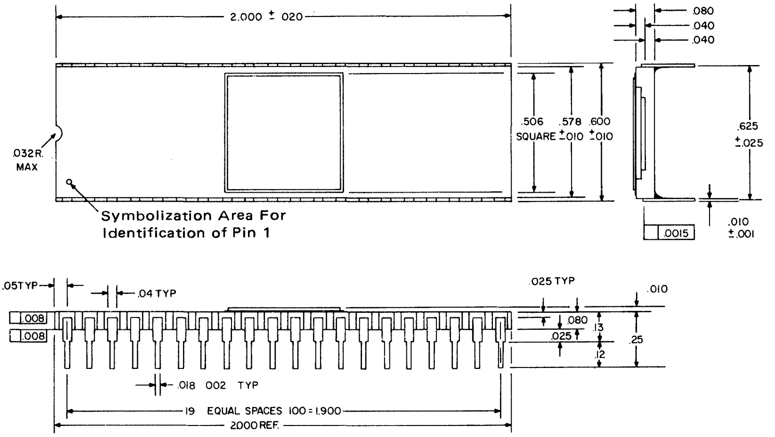


TRICKLE CHARGE CURRENT

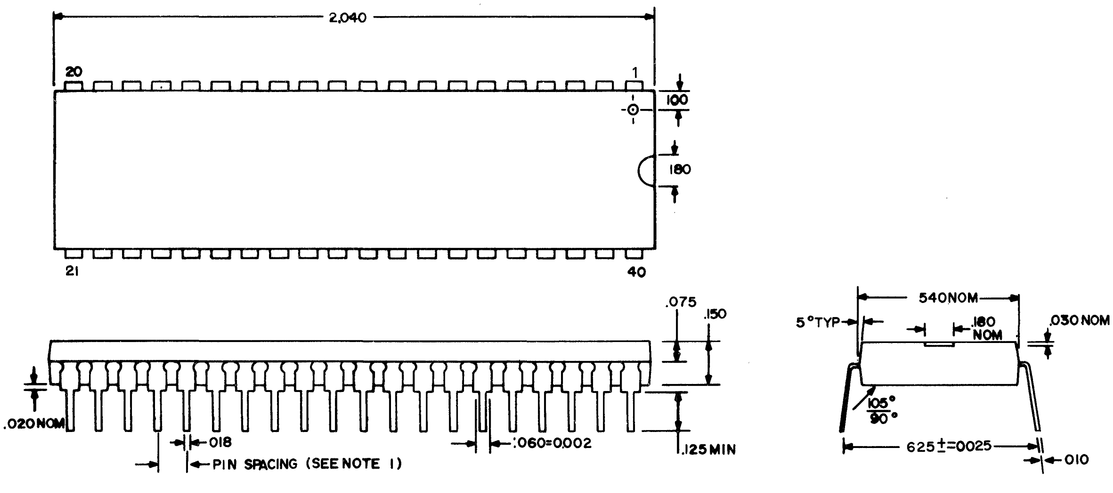
Figure 17



PACKAGE DESCRIPTION: 40-Pin Dual In-Line Ceramic Package



PACKAGE DESCRIPTION 40-Pin Dual-in-Line Plastic Package



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	TEMPERATURE RANGE
*MK3872(N)/16XXX	Plastic	0° C to +70° C
*MK3872(P)/16XXX	Ceramic	0° C to +70° C
+MK3872(N)/17XXX	Plastic	0° C to +70° C
+MK3872(P)/17XXX	Ceramic	0° C to +70° C

*Non Standby Device
+Standby Device

APPENDIX A

ORDERING INFORMATION

CUSTOM MK3872 OPTION SPECIFICATIONS

The custom MK3872 program may be transmitted to MOSTEK in any of the following media, listed in order of preference:

1) PROMs from the EMU-72
2) Punched paper tape
3) AID-80F Flexible Disk
4) Silent 700 cassette
5) Card Deck (IBM 80 column cards)

The program may be specified in the following forms:

PROMS with correct object code in each location

OBJECT CODE produced by one of MOSTEK's assemblers:

XFOR-50/70 Fortran IV Cross Assembler, SDB-50/70 resident assembler (ASMB-50/70), AID-80F F8 Cross-Assembler (FZCASM)

OBJECT CODE produced by the dump command from any of MOSTEK's F8 development hardware (SDB-50/70, AID-80F).

DATA DECK FORMAT as described in the Data Deck section

3870
Family

A completed cover letter (See Fig. A-1) must be attached. The information should be properly packed and mailed prepaid and insured to:

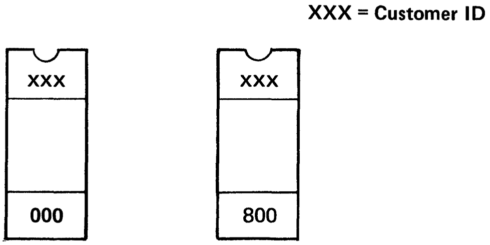
MOSTEK Corporation
Microcomputer Product Marketing
1215 West Crosby Road
Carrollton, Texas 75006

A second copy of the cover letter should be mailed separately to the above address.

PROMS

2716 type PROMs, programmed with the customer program (positive logic sense for addresses and data) may be submitted. The PROMs must be clearly marked to indicate which PROM corresponds to address space 000-7FF and which PROM corresponds to address space 800-FFF. See Fig. A-2 for marking. Include a three-letter customer ID on each PROM. After PROMs are removed from the EMU-72, they must be placed in conductive IC carriers and securely packed.

Figure A-2



PAPER TAPE

Punched paper tapes (1" wide, 8 level ASCII) will be accepted. The tape must contain the absolute object output from the above mentioned F8 assemblers. Paper object tapes in absolute format generated by the "D" (dump) command of DDT-2 or the dump command of the AID-80F (F8 debug option) are also acceptable if the entire memory space is dumped continuously. Tapes may also be punched using the DATA DECK FORMAT. They must contain 80 characters per record with a CR (carriage return) and LF (line feed) separating each record. The tape must be clearly labeled with customer name, and format used. Fan fold tape is preferred. Tape transparency should be limited to 60% transmissivity (40% opaque). Specifically, thin yellow or white tape is error prone on photo-electric readers and must not be used.

FLEXIBLE DISKS

FLEXIBLE DISKS (Floppy Disks) produced on the MOSTEK AID-80F development station may be submitted. The format must be the absolute object output from the assembler or an object dump using the memory dump command (F8 Debug Option). The disk must be clearly labeled with the format of the data (object, or object dump) and the customer's name.

MK3872 ORDER FORM
Figure A-1

DATE ______________ CUSTOMER PO NUMBER ______________

CUSTOMER NAME ____

ADDRESS ____

CITY ______________ STATE ______________ ZIP ______________

COUNTRY ______________

PHONE ______________________________ EXTENSION ______________

CONTACT MR. MS. ____

CUSTOMER PART # ____

OPTIONS:

EXTERNAL INTERRUPT: Pull-Up ______________ No Pull-Up ______________

RESET: Pull-Up ______________ No Pull-Up ______________

STANDBY POWER OPTION: Yes: ______________ No ______________

PORT OPTIONS:

	STANDARD TTL	OPEN DRAIN	DRIVER PULL-UP
P4-0 - - - - -	________	________	________
P4-1 - - - - -	________	________	________
P4-2 - - - - -	________	________	________
P4-3 - - - - -	________	________	________
P4-4 - - - - -	________	________	________
P4-5 - - - - -	________	________	________
P4-6 - - - - -	________	________	________
P4-7 - - - - -	________	________	________
P5-0 - - - - -	________	________	________
P5-1 - - - - -	________	________	________
P5-2 - - - - -	________	________	________
P5-3 - - - - -	________	________	________
P5-4 - - - - -	________	________	________
P5-5 - - - - -	________	________	________
P5-6 - - - - -	________	________	________
P5-7 - - - - -	________	________	________

Figure A-1 (Cont'd)

PATTERN MEDIA:

PROMS ______________________ PAPER TAPE (OBJECT) ______________________

SILENT 700 CASSETTE (OBJECT) ______________________

PAPER TAPE (DATA DECK) ______________________

CARD DECK (DATA DECK) ____________ DISKETTE (OBJECT) ____________

PREFERRED BASE ON VERIFICATION LISTING: HEX ________ DECIMAL ________

THESE ITEMS MAY AFFECT COST

BRANDING REQUIREMENT (If Any, 10 Alpha-numeric digits allowed)

____

____

PROTOTYPE QUANTITY (10 pieces normal - higher quantity extra charge)

____

WAIVE PROTOTYPES (Customer accepts liability for all work in progress)

Yes _____ No _____

____

Customer Signature

SIGNATURE ______________________

TITLE ______________________

3870 Family

PUNCHED CARD DECK

Standard 80 column punched cards must be used. They must be punched in IBM 029 code. The deck must contain two types of cards:

COMMENT CARDS
DATA CARDS

COMMENT CARDS

Comment Cards must have an asterisk (*) in column 1. The remaining 79 columns may be any character. Comment Cards may be placed anywhere throughout the data deck.

DATA CARDS

These cards specify the actual ROM data. All fields are right justified.

COLUMN 1:	C (the letter C)
COLUMN 2-9:	ADDR
COLUMN 10-12:	BYTE
COLUMN 14-16:	DATA 1
COLUMN 17-19:	DATA 2
COLUMN 20-22:	DATA 3
.	
.	
.	
COLUMN 76-78:	DATA 21
COLUMN 77-79:	DATA 22 or SEQUENCE NUMBER

ADDR is the address of the first byte of data (DATA 1) contained on that card. Successive data bytes read from that card will be placed in successively greater address locations. BYTE is the number of data bytes to be read from that card (1 to 22). If sequence numbers are used, the maximum number of bytes per card is 21. The base for ADDR and BYTE may be either decimal or hex but both must be the same. Data may be either in decimal or hex regardless of the base used for ADDR and BYTE. The base for sequence numbers (if they are used) is always decimal. The bases must be consistent throughout the deck. Data cards need not occur in order of increasing or decreasing addresses. Any unspecified address will be filled with zero. Any unpunched field will be read as a zero. If two data cards specify data for the same address, the one encountered second in the deck will override the first.

A portion of an example deck is shown.

```
*   3872 DATA DECK
*   MOSTEK CORP, EXAMPLE DECK
*   ADDR/BYTE ARE IN DECIMAL
*   DATA IS IN HEX
C  0  8  20  FF  OB  54  34  56  71  B6
C  8  8  1B  28  03  F3  4C  25  2E  94
C 16  8  04  29  01  00

*   START OF SUBROUTINE ALPHA

C  1096   4  20  32  7C  53
C  1100   4  52  47  29  06
C  1104   1  07
```

VERIFICATION MEDIA

All original pattern media (PROMs, paper tape, etc.) are filed for contractural purposes and are not returned. Two copies of computer listings printed during the creation of the custom mask pattern are returned. One copy may be kept by the customer. The other copy should be checked thoroughly, signed, and returned to MOSTEK. The signed listing constitutes the contractual agreement for creation of the custom mask. Though the computer listing serves as the actual verification media, MOSTEK will program 2716 PROMs programmed from the data file used to create the custom mask to aid in the verification process. If programmed PROMs are desired, two blank 2716 type PROMs must be provided by the customer.

3870 Family

PRELIMINARY

MOSTEK®

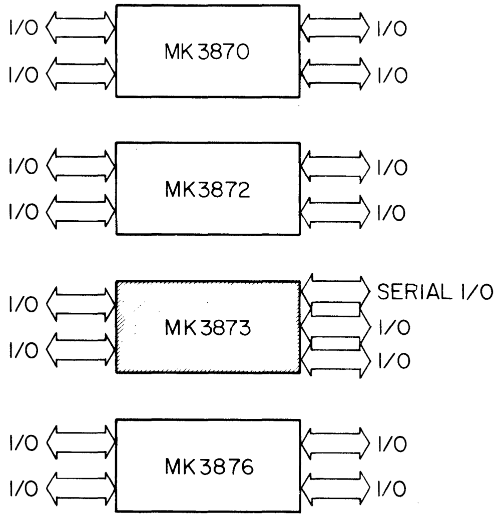
SINGLE-CHIP MICROCOMPUTER

MK3873

FEATURES

- □ Software Compatible with F8 family
- □ 2048 x 8 mask programmable ROM
- □ 64 byte scratchpad RAM
- □ 29 bits (4 ports) TTL compatible parallel I/O
- □ Serial Input/Output port
  - • External or Internal Serial Port Clock
  - • Internal Baud rate generator
  - • Data rates to 9600 bits per second
  - • Synchronous or Asynchronous I/O
  - • Vectored interrupts
  - • 3 I/O pins dedicated as Serial In, Serial Out, and Serial Clock
  - • Serial In/Serial Clock Schmitt Trigger Inputs T^{2}L compatible
- □ Programmable binary timer
  - • Internal timer mode
  - • Pulse width measurement mode
  - • Event counter mode
- □ External Interrupt
- □ Crystal, LC, RC or external time base
- □ Low Power (325 mW typ.)
- □ Single +5 volt ±10% power supply
- □ Pinout compatible with 3870 family

SINGLE CHIP 3870 MICROCOMPUTER FAMILY

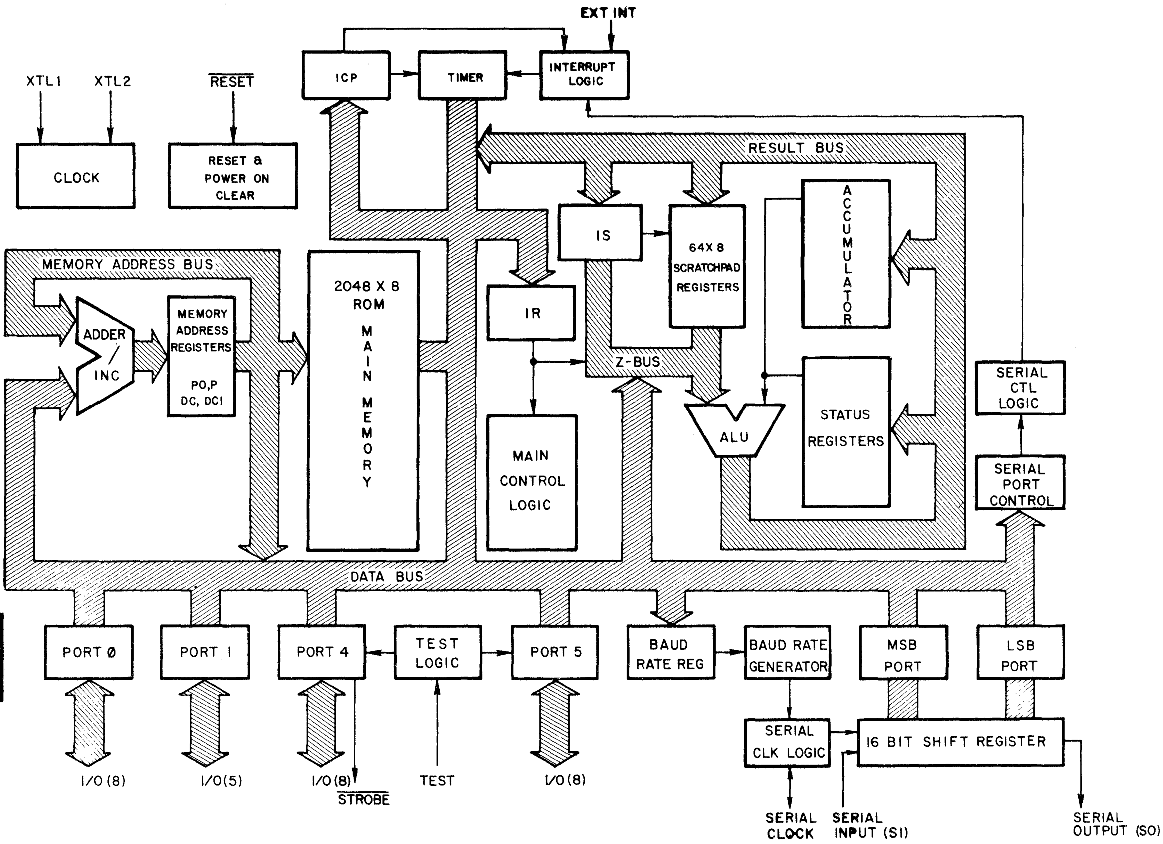


INTRODUCTION

The new MK3873 single chip microcomputer introduces a major addition to the MK3870 microcomputer family, a serial input/output port. This input/output port is capable of either synchronous or asynchronous serial data transfers. The heart of the serial port is a 16-bit shift register that can be read from or written to by the CPU while data is being shifted in or out of the shift register. The shift rate is determined by either an internal baud rate generator or an external clock. An end-of-word vectored interrupt is generated in either transmit or receive mode so that the CPU overhead is only at the word rate and not at the serial bit rate. This serial channel can be used to provide a low-cost data channel for communicating between 3873 Microcomputers or with a host computer. The serial port is also flexible in design so that it could be used for other purposes such as interface to external serial logic or serial memory devices.

The 3873 retains commonality with the 3870 family of single chip microcomputers. It has the central processor, oscillator, and clock circuits, 2,048 bytes of mask read-only memory for program storage and 64 bytes of scratch pad random-access memory. Also the sophisticated programmable binary timer is included which provides for system flexibility by operating in 3 different modes. The 3873 has a large number of parallel I/O lines available to the user. Twenty nine pins of the 3873 are I/O. In addition three pins are dedicated to the serial I/O port. These pins provide input, output and clock for the serial port. The serial clock pin can be driven externally or programmed to provide 50% duty cycle TTL compatible serial clock. No additional CPU instructions are necessary for use with the serial port.

MK3873 BLOCK DIAGRAM



MOSTEK®

PRODUCT BRIEF

P-PROM™ Microcomputer MK3874

FEATURES

□ EPROM version of the MK3870, MK3872 and MK3876

□ Accepts 24-pin, industry-standard EPROMs or bipolar PROMs

□ PROM capacity: 1K, 2K, 4K bytes

□ Completely pin compatible with 3870 family of single-chip microcomputers

□ Software compatibility with 3870

□ Use as prototyping tool or for low volume production

□ 64 x 8 scratchpad RAM

□ 64 x 8 of executable RAM addressable by program or data counter

□ Standby power mode option for executable RAM which includes

- Low standby power, less than 8.2 mW
- Minimum 2.2V standby supply voltage
- No external components required to trickle charge battery

□ 32 bits (4 eight-bit ports) TTL compatible I/O (30 with Standby Option)

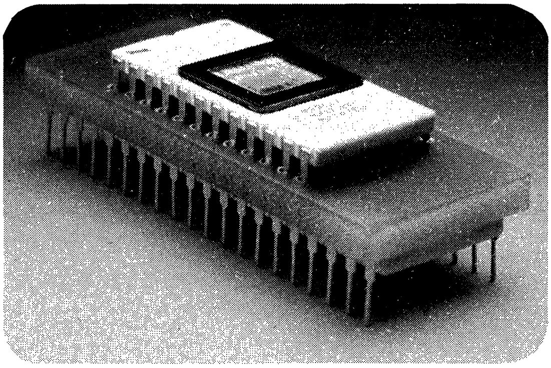
□ Programmable binary timer which includes:

- Interval timer mode
- Pulse width measurement mode
- Event counter mode

□ External interrupt

□ Crystal, LC, RC, or external time base

□ Single +5 volt supply

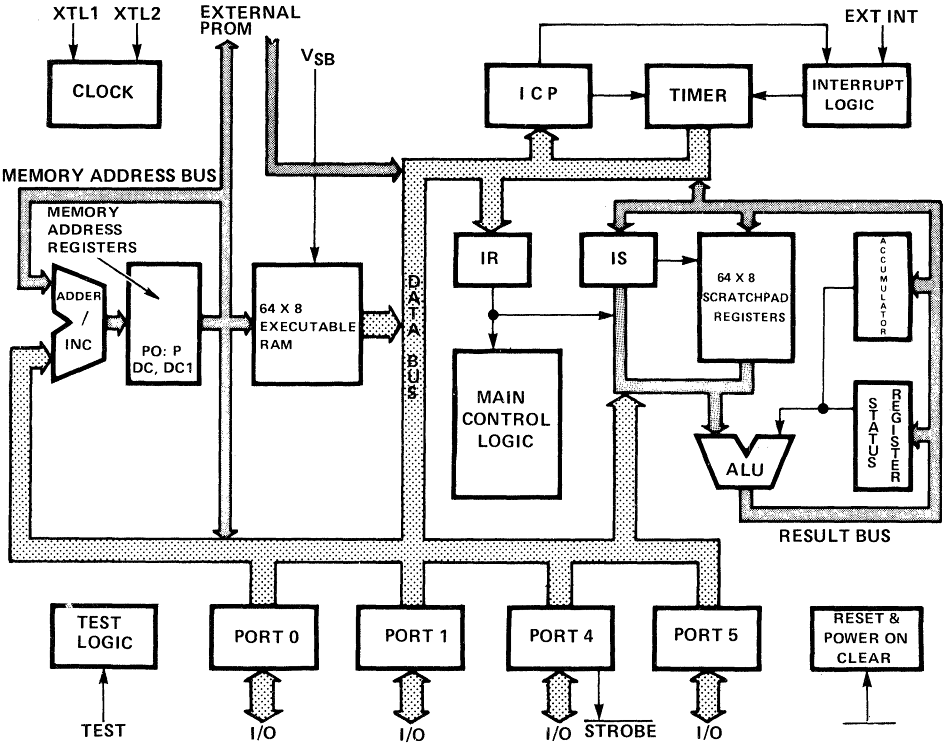


INTRODUCTION

The new MK3874 microprocessor is the PROM based version of the industry-standard 3870 family of single-chip microprocessors. The MK3874 is called the Piggyback PROM (P-PROM)TM because of a new Double-DipTM packaging concept. This concept allows a standard 24-pin PROM to be mounted directly on top of the microprocessor. This allows a standard EPROM to be easily removed for reprogramming and then reinserted as many times as desired. The MK3874 retains exactly the same pinout and architectural features as other members of the 3870 family. There are 32 lines (or 30 with the standby power RAM option) of bi-directional input/output, a sophisticated timer, vectored interrupts, executable and scratchpad RAM and an 8-bit CPU. Thus the 3874 P-PROMTM has the same functional capability and pinout as its 3870 masked-ROM counterpart while being able to support a standard PROM plugged into the top of the package.

Industry standard 24-pin, 5 volt PROMs are used with the MK3874. Presently six PROMs are compatible with the MK3874. They are the 2716 (2K x 8) 5 volt only, 2516 (2K x 8) 5 volt only, 2758 (1K x 8), 2532 (4K x 8), 2732 (4K x 8) and 82S2708 (1K x 8). The 2716 EPROM with its 2K of storage will allow the 3874 to emulate the 3870 and 3876 while a 2732 or 2532 EPROM containing 4K bytes of memory will allow emulation of the 3872. The 1K x 8 PROMs can be used for developing shorter programs. The standby power option is also available with the MK3874.

MK3874 BLOCK DIAGRAM



Supporting the 3874 is a complete line of development equipment including the low-cost Software Development Board (SDB-50/70) and an Application Interface Module (AIM-72). A fully integrated 3870/F8 development capability is provided by the AID-80F Disk Based Development System. Coupled with the AIM-72 and F8 Cross Assembler, it provides software generation and in-circuit emulation capabilities for the 3870 family of microcomputers.

PROM programming capability is provided through the use of the PPG 8/16 programming module available for either of the above systems.

Six different versions of the MK3874 are available and are designated MK974XX. The available versions of the MK3874 and their relevant features are listed in the table below. The different versions are provided so as to offer an option for low-power standby mode for the executable RAM and for different PROM pinouts. The MK97401 and MK97404 are supplied with MK2716 EPROMs. Otherwise the MK97401 is identical to the MK97400 and the MK97404 is identical to the MK97403.

All MK3874 versions have no internal pull-up resistor for the external interrupt and reset inputs. All are configured with the standard TTL port option for Ports 4 and 5. An open-drain and direct-drive version will be available in the second quarter of 1979.

ORDERING INFORMATION

MK3874 VERSION	PROM INCLUDED	STANDBY POWER OPTION	COMPATIBLE 5 - VOLT PROM's	3870 FAMILY DEVICE EMULATED
MK97400	NO	NO	2758 (1K x 8) 82S2708 (1K x 8) 2516 (2K x 8) 2716 (2K x 8) 2532 (4K x 8)	Partial 3870 Partial 3870 3870, 3876 3870, 3876 3872
MK97401	Yes (MK 2716)	NO	Same as MK97400	Same as MK 97400
MK97402	NO	NO	82S2708 (1K x 8) 2732 (4K x 8)	Partial 3870 3872
MK97403	NO	YES	2758 (1K x 8) 82S2708 (1K x 8) 2716 (2K x 8) 2516 (2K x 8) 2532 (4K x 8)	Partial 3870 Partial 3870 3876 3876 3872
MK97404	YES (MK 2716)	YES	Same as MK97403	Same as MK97403
MK97405	NO	YES	82S2708 (1K x 8) 2732 (4K x 8)	Partial 3870 3872

PRELIMINARY

MOSTEK®

F8 MICROCOMPUTER DEVICES

Single-Chip Microcomputer MK3876

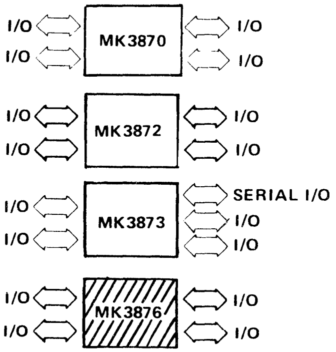
FEATURES

- □ Software compatible with F8 family
- □ 2048X 8 mask programmable ROM
- □ 64 byte scratchpad RAM
- □ 64 additional bytes of executable RAM addressable by program counter or data counter
- □ Standby option for executable RAM including:
  - —Low standby power, less than 8.2mW
  - —Minimum 2.2V standby supply voltage
  - —No external components required to trickle charge battery
- □ 32 bits (4 ports) TTL Compatible I/O
- □ Programmable binary timer
  - —Internal timer mode
  - —Pulse width measurement mode
  - —Event counter mode
- □ External interrupt
- □ Crystal, LC, RC, or external time base
- □ Low power (285mW typ.)
- □ Single +5 volt ± 10% power supply
- □ Same pinout as MK3870

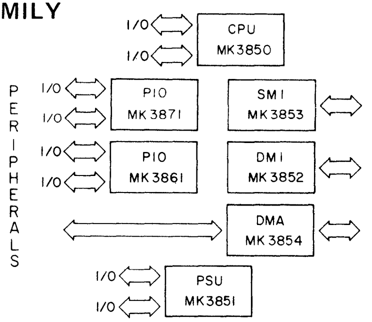
GENERAL DESCRIPTION

The MK3876 is a complete 8-bit microcomputer on a single MOS integrated circuit. The 3876 can execute the F8 instruction set of more than 70 commands, allowing expansion into multi-chip configurations with software compatibility. The device features 2048 bytes of ROM, 64 bytes of scratchpad RAM, 64 bytes of executable RAM, a programmable binary timer, 32 bits of I/O, and a single +5 volt power supply requirement. Utilizing ion-implanted, N-channel silicon gate technology and advanced circuit design techniques the single-chip 3876 offers maximum cost-effectiveness in a wide range of control and logic replacement applications. The 3876 is an expanded memory version of the 3870 single chip microcomputer. The 3876 is identical to the 3870 in the following areas: instruction set, architecture, AC and DC characteristics, and pinout. The only change is in the memory expansion along with the appropriate memory address registers.

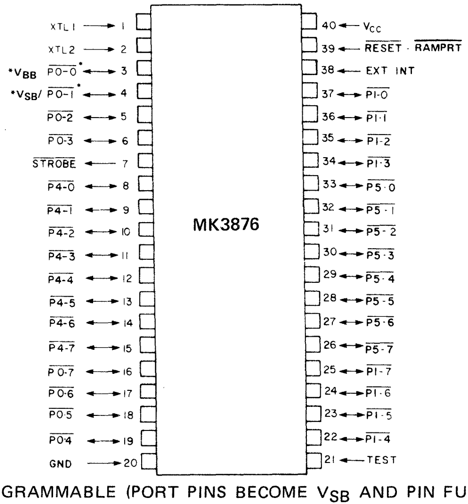
SINGLE CHIP 3870 MICROCOMPUTER FAMILY



F8 FAMILY



PIN CONNECTIONS



*PROGRAMMABLE (PORT PINS BECOME V_{SB} AND PIN FUNCTION DEPENDS ON DEVICE OPTION (STANDBY DEVICE OR STANDARD DEVICE)

PIN NAME	DESCRIPTION	TYPE
$\overline{\text{P0-0}}$ – $\overline{\text{P0-7}}$	I/O Port 0	Bidirectional
$\overline{\text{P1-0}}$ – $\overline{\text{P1-7}}$	I/O Port 1	Bidirectional
$\overline{\text{P4-0}}$ – $\overline{\text{P4-7}}$	I/O Port 4	Bidirectional
$\overline{\text{P5-0}}$ – $\overline{\text{P5-7}}$	I/O Port 5	Bidirectional
$\overline{\text{STROBE}}$	Ready Strobe	Output
EXT INT	External Interrupt	Input
$\overline{\text{RESET}}$ – $\overline{\text{RAMPRT}}$	External Reset, RAM Protect	Input
TEST	Test Line	Input
XTL 1, XTL 2	Time Base	Input
V_{CC}, GND	Power Supply Lines	Input
V_{SB}	Standby Power	Input
V_{BB}	Substrate Decoupling	Input

FUNCTIONAL PIN DESCRIPTION

$\overline{\text{P0-0}}$–$\overline{\text{P0-7}}$, $\overline{\text{P1-0}}$–$\overline{\text{P1-7}}$, $\overline{\text{P4-0}}$–$\overline{\text{P4-7}}$, and $\overline{\text{P5-0}}$–$\overline{\text{P5-7}}$ are 32 lines which can be individually used as either TTL compatible inputs or as latched outputs.

$\overline{\text{STROBE}}$ is a ready strobe associated with I/O Port 4. This pin which is normally high provides a single low pulse after valid data is present on the P4-0–P4-7 pins during an output instruction.

$\overline{\text{RESET}}$ - $\overline{\text{RAMPRT}}$ may be used to externally reset the 3876. When pulled low the 3876 will reset. When allowed to go high the 3876 will begin program execution at program location H '000'. Additionally when $\overline{\text{RESET}}$ - $\overline{\text{RAMPRT}}$ is brought low all accesses of the executable RAM are prevented and the RAM is placed in a protected state for powering down V_{CC} without loss of data when the STANDBY option is selected.

EXT INT is the external interrupt input. Its active state is software programmable. This input is also used in conjunction with the timer for pulse width measurement and event counting.

XTL 1 and XTL 2 are the time base inputs to which a crystal (1 to 4MHz), LC network, RC network, or an external single-phase clock may be connected.

TEST is an input, used only in testing the 3876. For normal circuit functionality this pin is left unconnected or may be grounded.

V_{CC} is the power supply input (+5V± 10%).

V_{SB} is the RAM standby power supply input if the standby option is selected (+5.5V to +2.2V).

V_{BB} is the substrate decoupling pin. A .01 micro-Farad capacitor is required to provide substrate decoupling. It is only used when standby option is selected.

3870 ARCHITECTURE

This section describes the basic functional elements of the 3876 as shown in the block diagram of Figure 1. A programming model is shown in Figure 2.

Main Control Logic

The Instruction Register (IR) receives the operation code (OP code) of the instruction to be executed from the program ROM via the data bus. During all OP code fetches eight bits are latched into the IR. Some instructions are completely specified by the upper 4 bits of the OP code. In those instructions the lower 4 bits are an immediate register address or an immediate 4 bit operand. Once latched into the IR the main control logic decodes the instruction and provides the necessary control gating signals to all circuit elements.

ROM Address Registers

There are four 12 bit registers associated with the 4K x 8 ROM and 64 x 8 RAM. These are the Program Counter (P0), the Stack Register (P), the Data Counter (DC) and the Auxiliary Data Counter (DC1). The Program Counter is used to address instructions or immediate operands. P is used to save the contents of P0 during an interrupt or subroutine call. Thus, P contains the return address at which processing is to resume upon completion of the subroutine or the interrupt routine.

The Data Counter (DC) is used to address data tables. This register is auto-incrementing. Of the two data counters only DC can access the memory. However, the XDC instruction allows DC and DC1 to be exchanged.

Associated with the address registers is a 12 bit Adder/Incrementer. This logic element is used to increment P0 or DC when required and is also used to add displacements to P0 on relative branches or to add the data bus contents to DC in the ADC (add data counter) instruction.

4032 x 8 ROM

The microcomputer program and data constants are stored in the program ROM. When a ROM access is required, the appropriate address register (P0 or DC) is gated onto the ROM address bus and the ROM output is gated onto the main data bus. The first byte in ROM is location zero.

64 x 8 Executable RAM

The upper 64 bytes of the total 4096 byte memory of the 3876 is RAM memory. The first byte is at address 4032 decimal (FCO hex). As with the ROM memory the RAM memory may be accessed by the PO and DC address registers. It may be written via the STORE (ST) instruction. It may be read via the LOAD (LM) instruction. Additionally instructions may be executed from the RAM. A mask programmable standby power option is available whereby the 64x8 RAM remains powered and protected so that its contents are saved during a loss of the normal circuit power supply.

Scratchpad and IS

The scratchpad provides 64 8-bit registers which may be used as general purpose RAM memory. The Indirect Scratchpad Address Register (IS) is a 6 bit register used to address the 64 registers. All 64 registers may be accessed using IS. In addition the lower order 12 registers may also be directly addressed.

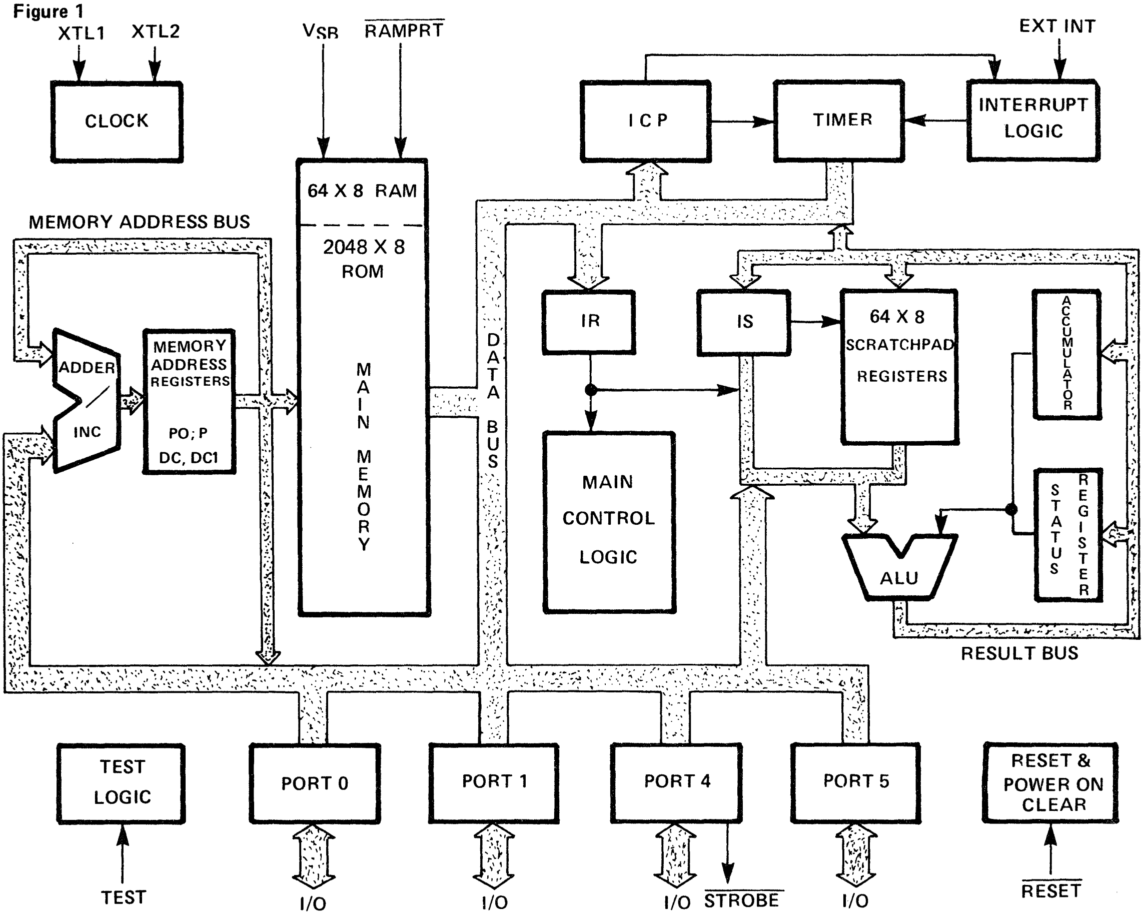
IS can be visualized as holding two octal digits. This division of IS is important since a number of instructions increment or decrement only the least significant 3 bits of IS when referencing scratchpad bytes via IS. This makes it easy to reference a buffer consisting of contiguous scratchpad bytes. For example, When the low order octal digit is incremented or decremented IS is incremented from octal 27 (0 '27') to 0 '20' or is decremented from 0 '20' to 0 '27'. This feature of the IS is very useful in many program sequences. All six bits of IS may be loaded at one time or either half may be loaded independently.

Scratchpad registers 9 through 15 (decimal) are given mnemonic names (J, H, K, and Q) because of special linkages between these registers and other registers such as the Stack Register. These special linkages facilitate the implementation of multi-level interrupts and subroutine nesting. For example, the instruction LR K, P stores the lower eight bits of the Stack Register into register 13 (K lower or KL) and stores the upper three bits of P into register 12 (K upper or KU) The scratchpad is not protected with the standby power option.

Arithmetic and Logic Unit (ALU)

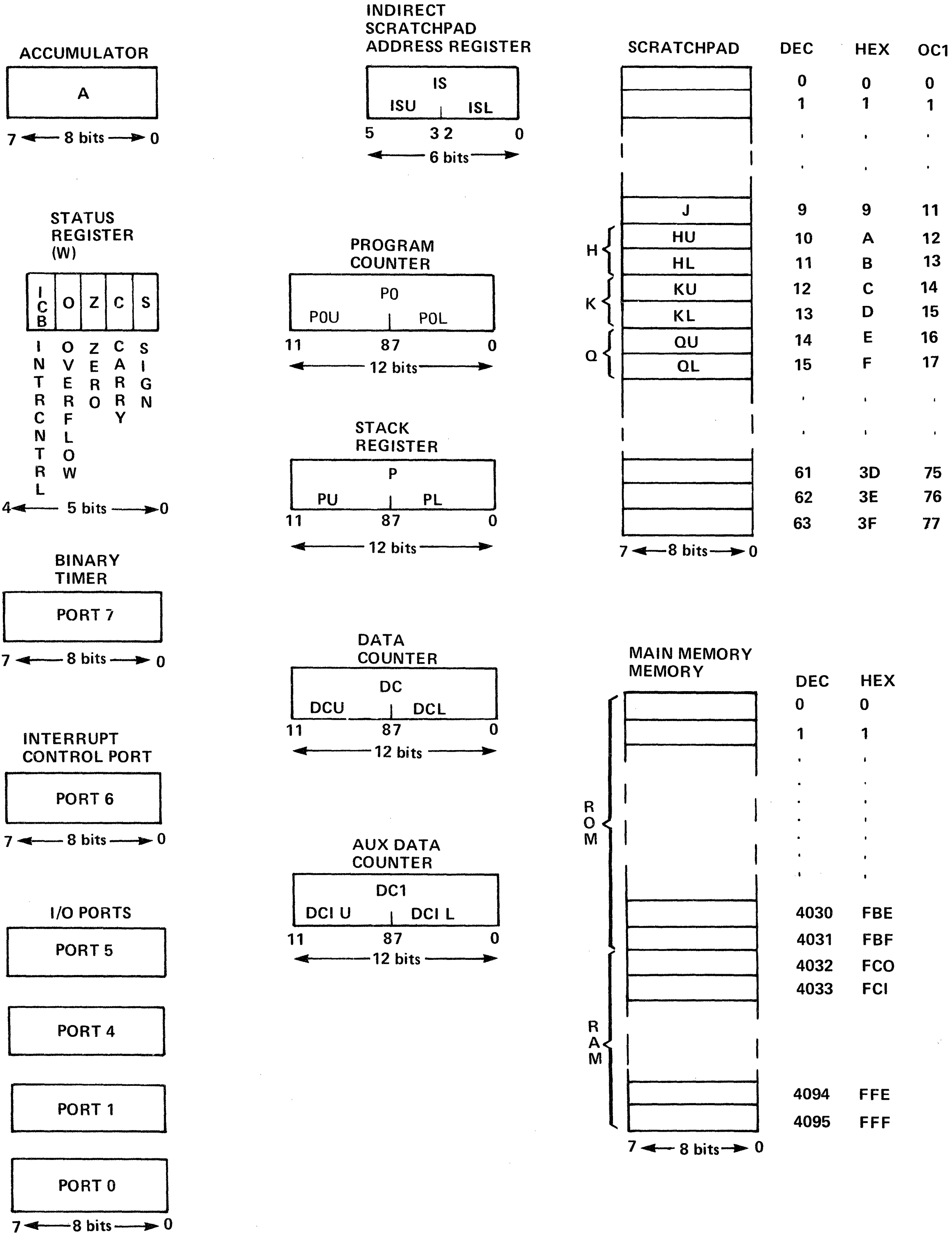
After receiving commands from the main control

MK 3876 BLOCK DIAGRAM
Figure 1



3876 PROGRAMMABLE REGISTERS, PORTS AND MEMORY MAP

Figure 2



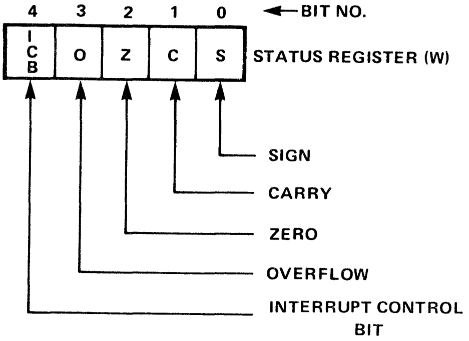
logic, the ALU performs the required arithmetic or logic operations (using the data presented on the two input busses) and provides the result on the result bus. The arithmetic operations that can be performed in the ALU are binary add, decimal adjust, add with carry, decrement, and increment. The logic operations that can be performed are AND, OR, EXCLUSIVE OR, 1's complement, shift right, and shift left. Besides providing the result on the result bus, the ALU also provides four signals representing the status of the result. These signals, stored in the Status Register (W), represent CARRY, OVERFLOW, SIGN, and ZERO condition of the result of the operation.

Accumulator (A)

The Accumulator (A) is the prinicpal register for data manipulation within the 3876. A serves as one input to the ALU for arithmetic or logical operations. The result of ALU operations are stored in A.

The Status Register (W)

The Status Register (also called the W register) holds five status flags as follows:



Summary of Status Bits

$$\text{OVERFLOW} = \text{CARRY}_7 \oplus \text{CARRY}_6$$

$$\text{ZERO} = \overline{\text{ALU}_7} \wedge \overline{\text{ALU}_6} \wedge \overline{\text{ALU}_5} \wedge \overline{\text{ALU}_4} \wedge \overline{\text{ALU}_3} \wedge \overline{\text{ALU}_2} \wedge \overline{\text{ALU}_1} \wedge \overline{\text{ALU}_0}$$

$$\text{CARRY} = \text{CARRY}_7$$

$$\text{SIGN} = \overline{\text{ALU}_7}$$

Interrupt Control Bit (ICB)

The ICB may be used to allow or disallow interrupts in the 3876. This bit is not the same as the two interrupt enable bits in the Interrupt Control Port (ICP). If the ICB is set and the 3876 interrupt logic communicates an interrupt request to the CPU section, the interrupt will be acknowledged and processed upon completion of the first non-privileged instruction. If the ICB is cleared an interrupt request will not be acknowledged or processed until the ICB is set.

I/O Ports

The 3876 provides four complete bidirectional Input/Output ports. (When standby option is used, Port 0, bit 0 and 1 are not available). These are Ports 0, 1, 4, and 5. In addition, the Interrupt Control Port is addressed as Port 6 and the binary timer is addressed as Port 7. An output instruction (OUT or OUTS) causes the contents of A to be latched into the addressed port. An input instruction (IN or INS) transfers the contents of the port to A (port 6 is an exception which is described later). The I/O pins on the 3876 are logically inverted. The schematic of an I/O pin and available output drive options are shown in Figure 3.

An output ready strobe is associated with Port 4. This flag may be used to signal a peripheral device that the 3876 has just completed an output of new data to Port 4. The strobe provides a single low pulse shortly after the output operation is completely finished, so either edge may be used to signal the peripheral. $\overline{\text{STROBE}}$ may also be used as an input strobe simply by doing a dummy output of H '00' to Port 4 after completing the input operation.

Timer and Interrupt Control Port

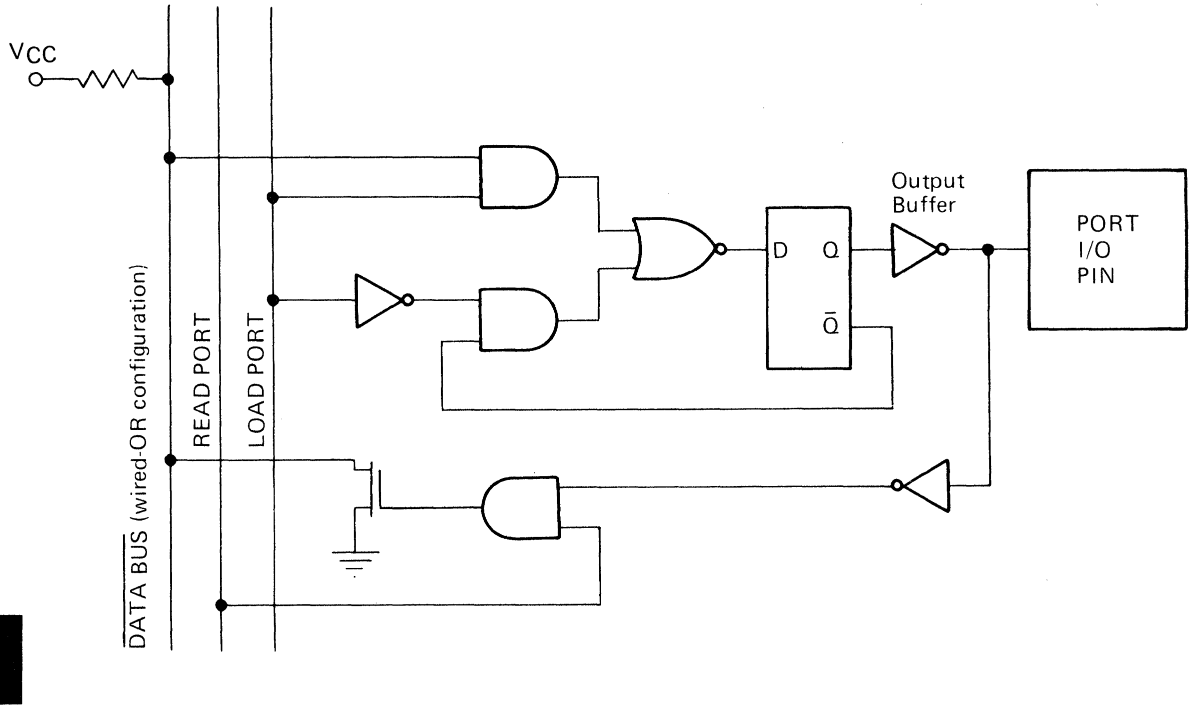
The Timer is an 8-bit binary down counter which is software programmable to operate in one of three modes: the Interval Timer Mode, the Pulse Width Measurement Mode, or the Event Counter Mode. As shown in Figure 4, associated with the Timer are an 8-bit register called the Interrupt Control Port, a programmable prescaler, and an 8-bit modulo-N register. A functional logic diagram is shown in Figure 5.

The desired timer mode, prescale value, starting and stopping the timer, active level of the EXT INT pin, and local enabling or disabling of interrupts are selected by outputting the proper bit configuration from the Accumulator to the Interrupt Control Port (Port 6) with an OUT or OUTS instruction. Bits within the Interrupt Control Port are defined as follows:

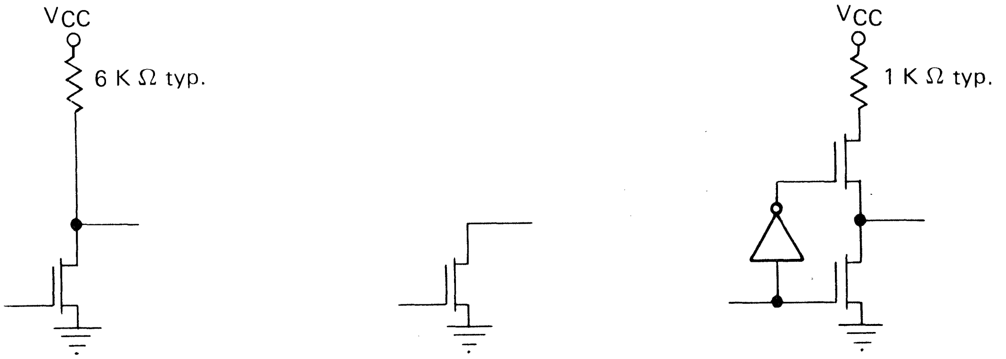
Interrupt Control Port (Port 6)

Bit 0 - External Interrupt Enable
Bit 1 - Timer Interrupt Enable
Bit 2 - EXT INT Active Level
Bit 3 - Start/Stop Timer
Bit 4 - Pulse Width/Interval Timer
Bit 5 - ÷ 2 Prescale
Bit 6 - ÷ 5 Prescale
Bit 7 - ÷ 20 Prescale

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS
Figure 3



OUTPUT BUFFER OPTIONS



Standard Output

Open Drain Output

Direct Drive Output

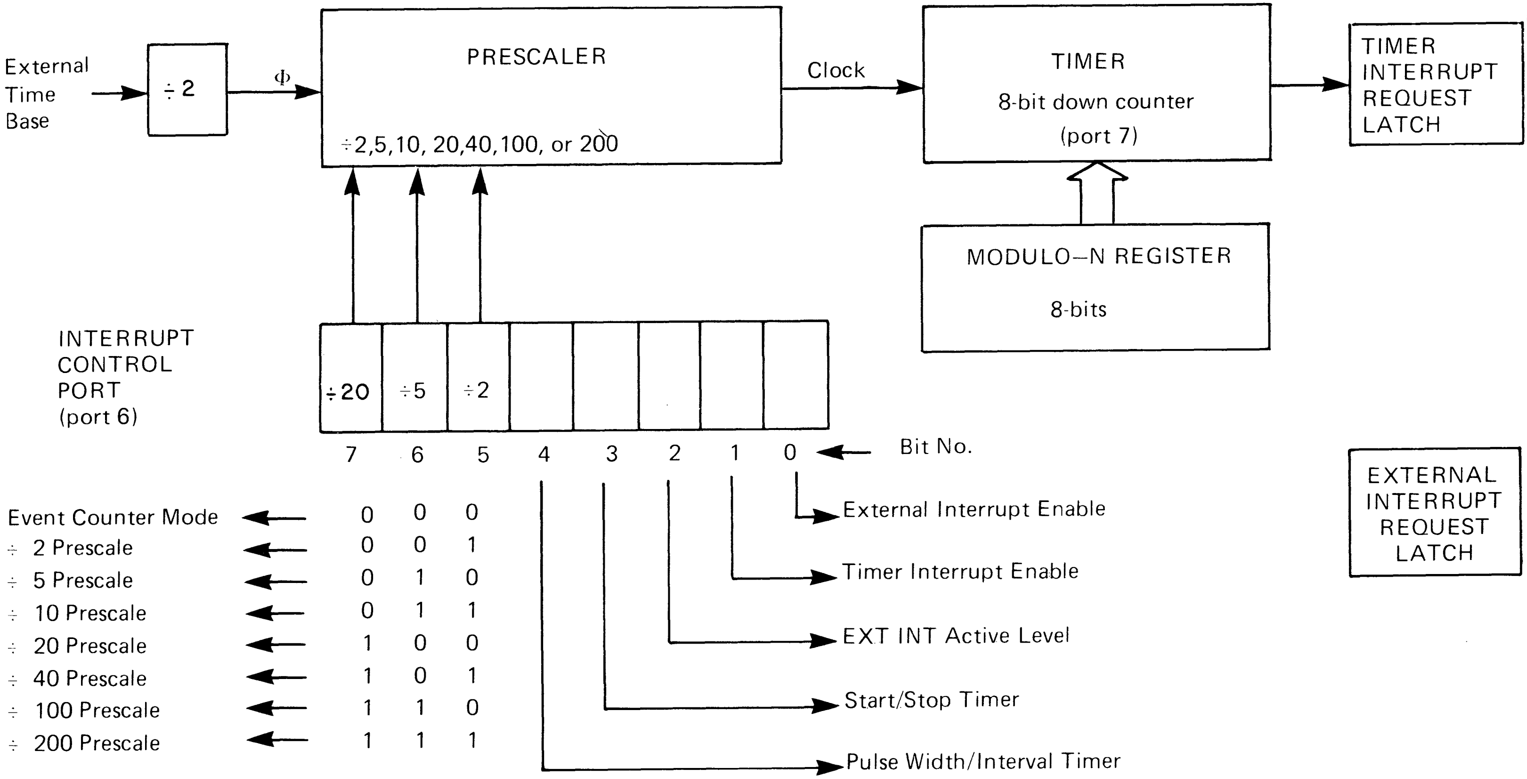
Ports 0 and 1 are Standard Output type only.

Ports 4 and 5 may both be any of the three output options (programmable bit by bit).

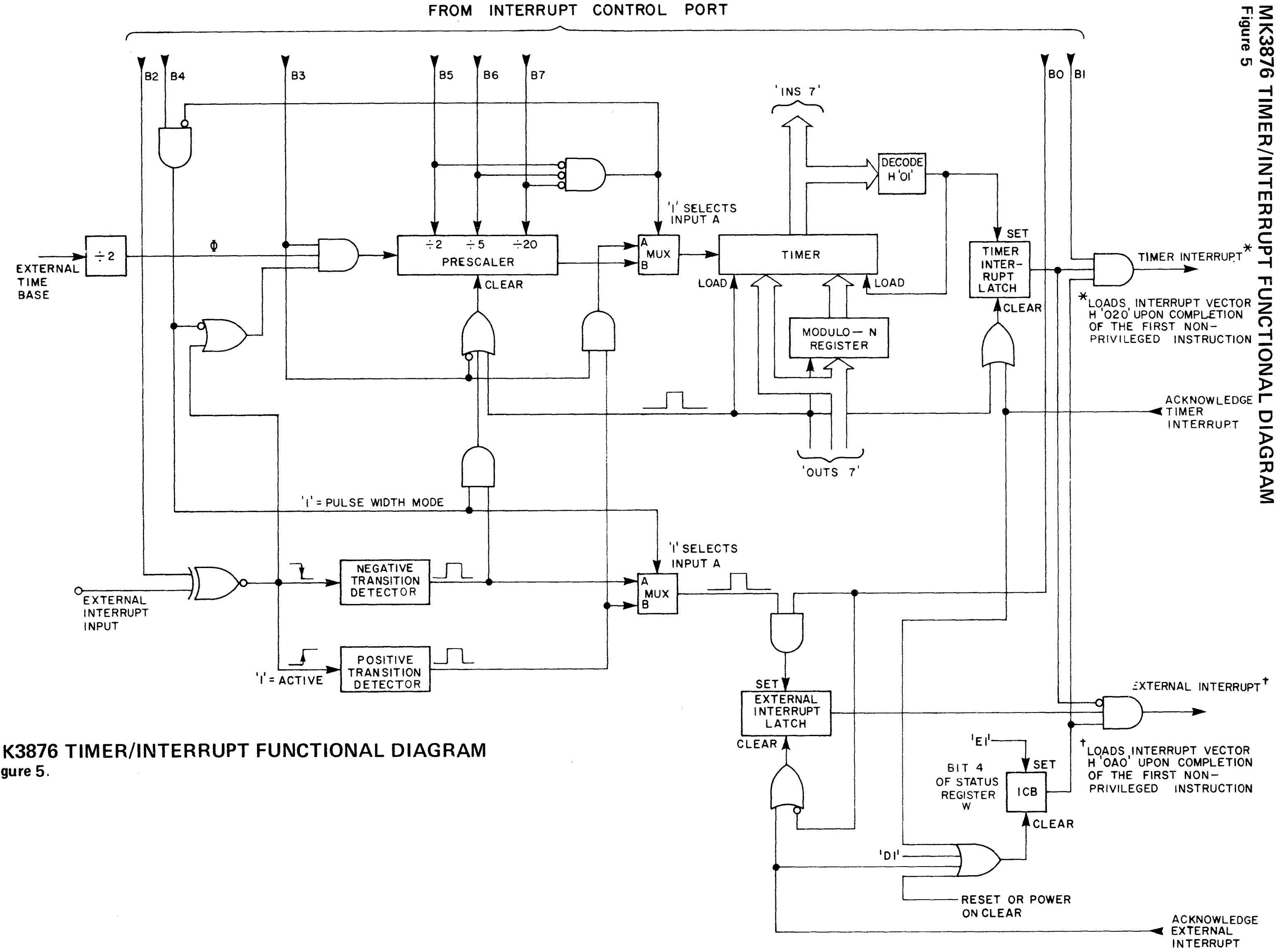
The $\overline{\text{STROBE}}$ output is always configured similar to a Direct Drive Output except that it is capable of driving 3 TTL loads.

$\overline{\text{RESET}}$ and EXT INT may have standard 6KΩ (typical) pull-up or may have no pull-up.

TIMER & CONTROL PORT BLOCK DIAGRAM
Figure 4



Note: See Figure 5 for a more detailed functional diagram.



MK3876 TIMER/INTERRUPT FUNCTIONAL DIAGRAM
Figure 5.

MK3876 TIMER/INTERRUPT FUNCTIONAL DIAGRAM
Figure 5

A special situation exists when reading the Interrupt Control Port (with an IN or INS instruction). The Accumulator is <u>not</u> loaded with the content of the ICP; instead, Accumulator bits 0 through 6 are loaded with 0's while bit 7 is loaded with the logic level being applied to the EXT INT pin, thus allowing the status of EXT INT to be determined without the necessity of servicing an external interrupt request. When reading the Interrupt Control Port (Port 6) bit 7 of the Accumulator is loaded with the actual logic level being applied to the EXT INT pin, regardless of the status of ICP bit 2 (the EXT INT Active Level bit); that is, if EXT INT is at +5V bit 7 of the Accumulator is set to a logic 1, but if EXT INT is at GND then Accumulator bit 7 is reset to logic 0. This capability is useful in establishing a high speed polled handshake procedure or for using EXT INT as an extra input pin if external interrupts are not required and the Timer is used only in the Interval Timer Mode. However, if it is desirable to read the contents of the ICP then one of the 64 scratchpad registers or one byte of RAM may be used to save a copy of whatever is written to the ICP.

The rate at which the timer is clocked in the Interval Timer Mode is determined by the frequency of an internal Φ clock and by the division value selected for the prescaler. (The internal Φ clock operates at one-half the external time base frequency). If ICP bit 5 is set and bits 6 and 7 are cleared, the prescaler divides Φ by 2. Likewise, if bit 6 or 7 is individually set the prescaler divides Φ by 5 or 20 respectively. Combinations of bits 5, 6 and 7 may also be selected. For example, if bits 5 and 7 are set while 6 is cleared the prescaler will divide by 40. Thus possible prescaler values are ÷2, ÷5. ÷10, ÷20, ÷40, ÷100, and ÷200.

Any of three conditions will cause the prescaler to be reset: whenever the timer is stopped by clearing ICP bit 3, execution of an output instruction to Port 7, (the timer is assigned port address 7), or on the trailing edge transition of the EXT INT pin when in the Pulse Width Measurement Mode. These last two conditions are explained in more detail below.

An OUT or OUTS instruction to Port 7 will load the content of the Accumulator to both the Timer and the 8-bit modulo-N register, reset the prescaler, and clear any previously stored timer interrupt request. As previously noted, the Timer is an 8-bit down counter which is clocked by the prescaler in the Interval Timer Mode and in the Pulse Width Measurement Mode. The prescaler is not used in the Event Counter Mode. The Modulo-N register is a buffer whose function is to save the value which was most recently outputted to Port 7. The modulo-N register is used in all three timer modes.

Interval Timer Mode

When ICP bit 4 is cleared (logic 0) and at least one prescale bit is set the Timer operates in the Interval Timer Mode. When bit 3 of the ICP is set the Timer will start counting down from the modulo-N value. After counting down to H'01', the Timer returns to the modulo-N value at the next count. On the transition from H'01' to H 'N' the Timer sets a timer interrupt request latch. Note that the interrupt request latch is set by the transition to H 'N' and not be the presence of H 'N' in the Timer, thus allowing a full 256 counts if the modulo-N register is preset to H '00'. If bit 1 of the ICP is set, the interrupt request is passed on to the CPU section of the 3876. However, if bit 1 of the ICP is a logic 0 the interrupt request is not passed on to the CPU section but the interrupt request latch remains set. If ICP bit 1 is subsequently set, the interrupt request will then be passed on to the CPU section. (Recall from the discussion of the Status Register's Interrupt Control Bit that the interrupt request will be acknowledged by the CPU section only if ICB is set). Only two events can reset the timer interrupt request latch; when the timer interrupt request latch is acknowledged by the CPU section, or when a new load of the modulo-N register is performed.

Consider an example in which the modulo-N register is loaded with H '64' (decimal 100). The timer interrupt request latch will be set at the 100th count following the timer start and the timer interrupt request latch will repeatedly be set on precise 100 count intervals. If the prescaler is set at ÷40 the timer interrupt request latch will be set every 4000 Φ clock periods. For a 2MHz Φ clock (4MHz time base frequency) this will produce 2 millisecond intervals.

The range of possible intervals is from 2 to 51,200 Φ clock periods (1μs to 25.6ms for a 2MHzΦ clock). However, approximately 50 Φ periods is a practical minimum because the time between setting the interrupt request latch and the execution of the first instruction of the interrupt service routine is at least 29 Φ periods (the response time is dependent upon how many privileged instructions are encountered when the request occurs). To establish time intervals greater than 51,200 Φ clock periods is a simple matter of using the timer interrupt service routine to count the number of interrupts, saving the result in one or more of the scratchpad registers until the desired interval is achieved. With this technique virtually any time interval, or several time intervals, may be generated.

The Timer may be read at any time and in any mode using an input instruction (IN 7 or INS 7) and may take place "on the fly" without interferring with normal timer operation. Also, the Timer may be stopped at any time by clearing bit 3 of the ICP. The timer will hold its current contents indefinitely and will resume counting when bit 3 is again set. Recall however that the prescaler is reset whenever the Timer is stopped; thus a series of starting and stopping will result in a cumulative truncation error.

A summary of other timer errors is given in the timing section of this specification. For a free running timer in the Interval Timer Mode the time interval between any two interrupt requests may be in error by ± 6 Φ clock periods although the cumulative error over many intervals is zero. The prescaler and Timer generate precise intervals for setting the timer interrupt request latch but the time out may occur at any time within a machine cycle. (There are two types of machine cycles: short cycles which consist of 4 Φ clock periods and long cycles which consist of 6 Φ clock periods. In the multi-chip F8 family there is a signal called the WRITE clock which corresponds to a machine cycle). Interrupt requests are synchronized with the internal WRITE clock thus giving rise to the possible ± 6 Φ error. Additional errors may arise due to the interrupt request occurring while a privileged instruction or multicycle instruction is being executed. Nevertheless, for most applications all of the above errors are negligible, especially if the desired time intervall is greater than 1ms.

Pulse Width Measurement Mode

When ICP bit 4 is set (logic 1) and at least one prescale bit is set the Timer operates in the Pulse Width Measurement Mode. This mode is used for accurately measuring the duration of a pulse applied to the EXT INT pin. The Timer is stopped and the prescaler is reset whenever EXT INT is at its inactive level. The active level of EXT INT is defined by ICP bit 2; if cleared, EXT INT is active low; if set, EXT INT is active high. If ICP bit 3 is set, the prescaler and Timer will start counting when EXT INT transitions to the active level. When EXT INT returns to the inactive level the Timer then stops, the prescaler resets, and <u>if ICP bit 0 is set</u> an external interrupt request latch is set. (Unlike timer interrupts, external interrupts are not latched if the ICP Interrupt Enable bit is not set).

As in the Interval Timer Mode, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, the prescaler and ICP bit 1 function as previously described, and the Timer still functions as an 8-bit binary down counter with the timer interrupt request latch being set on the Timer's transition from H '01' to H 'N'. Note that the EXT INT pin has nothing to do with loading the Timer; its action is that of automatically starting and stopping the Timer and of generating external interrupts. Pulse widths longer than the prescale value times the modulo-N value are easily measured by using the timer interrupt service routine to store the number of timer interrupts in one or more scratchpad registers.

As for accuracy, the actual pulse duration is typically slightly longer than the measured value because the status of the prescaler is not readable and is reset when the Timer is stopped. Thus for maximum accuracy it is advisable to use a small division setting for the prescaler.

Event Counter Mode

When ICP bit 4 is cleared and all prescale bits (ICP bits 5, 6, and 7) are cleared the Timer operates in the Event Counter Mode. This mode is used for counting pulses applied to the EXT INT pin. If ICP bit 3 is set the Timer will decrement on each transition from the inactive level to the active level or the EXT INT pin. The prescaler is not used in this mode, but as in the other two timer modes, the timer may be read at any time, may be stopped at any time by clearing ICP bit 3, ICP bit 1 functions as previously described, and the timer interrupt request latch is set on the Timer's transition from H '01' to H 'N'.

Normally ICP bit 0 should be kept cleared in the Event Counter Mode; otherwise, external interrupts will be generated on the transition from the inactive level to the active level of the EXT INT pin.

For the Event Counter Mode the minimum pulse width required on EXT INT is 2 Φ clock periods and the minimum inactive time is 2 Φ clock periods; therefore, the maximum repetition rate is 500KHz.

Timer Emulation

For total software compatibility when expanding into a multi-chip configuration the MK3871 Peripheral Input/Output circuit should be used rather than the older MK3861 PIO. The MK3871 has the same improved Timer (binary count, readable, and three modes of operation rather than one) and ready strobe output as are on the MK3876.

External Interrupts

When the timer is in the Interval Timer Mode the EXT INT pin is available for non-timer related interrupts. If ICP bit 0 is set an external interrupt request latch is set when there is a transition from the inactive level to the active level of EXT INT. (EXT INT is an edge-triggered input). The interrupt request is latched until either acknowledged by the CPU section or until ICP bit 0 is cleared (unlike timer interrupt requests which remain latched even when ICP bit 1 is cleared). External interrupts are handled in the same fashion when the Timer is in the Pulse Width Measurement Mode or in the Event Counter Mode, except that only in the Pulse Width Measurement Mode the external interrupt request latch is set on the trailing edge of EXT INT, that is, on the transition from the active level to the inactive level.

Interrupt Handling

When either a timer or an external interrupt request is communicated to the CPU section of the 3876, it will be acknowledged and processed at the completion of the first non-privileged instruction if the Interrupt Control Bit of the Status Register is set. If the Interrupt Control Bit is not set, the interrupt request will continue until either the Interrupt Control Bit is set and the CPU section acknowledges the interrupt or until the interrupt request is cleared as previously described.

If there is both a timer interrupt request and an external interrupt request when the CPU section starts to process the requests, the timer interrupt is handled first.

When an interrupt is allowed the CPU section will request that the interrupting element pass its interrupt vector address to the Program Counter via the data bus. The vector address for a timer interrupt is H '020'. The vector address for external interrupts is H '0A0'. After the vector address is passed to the Program Counter, the CPU section sends an acknowledge signal to the appropriate interrupt request latch which clears that latch. The execution of the interrupt service routine will then commence. The return address of the original program is automatically saved in the Stack Register, P.

The Interrupt Control Bit of W (Status Register) is automatically reset when an interrupt request is acknowledged. It is then the programmer's responsibility to determine when ICB will again be set (by executing an EI instruction). This action prevents an interrupt service routine from being interrupted unless the programmer so desires.

Figure 6 details the interrupt sequence which occurs whether the interrupt request is from an external source via EXT INT or from the 3876's internal timer. Events are labeled with the letters A through G and are described below.

Event A

An interrupt request must satisfy a hold time requirement as specified in the AC Characteristics in order to guarantee that it is valid on the rising edge of the WRITE clock.

Event B

Event B represents the instruction being executed when the interrupt occurs. The last cycle of B is normally the instruction fetch for the next cycle. However, if B is not a privileged instruction and the CPU's Interrupt Control Bit is set, then the last cycle becomes a "freeze" cycle rather than a fetch. At the end of the freeze cycle the interrupt request latches are inhibited from altering the interrupt daisy-chain so that sufficient time will be allowed for the daisy-chain to settle. (If B is a privileged instruciton, the instruction fetch is not replaced by a freeze cycle; instead, the fetch is performed and the next instruction is executed. Although unlikely to be encountered, a series of privileged instructions will be sequentially executed without interrupt. One more instruction, called a 'protected' instruction, will always be executed after the last privileged instruction. The last cycle of the protected instruction then performs the freeze.)

The dashed lines on EXT INT illustrate the last opportunity for EXT INT to cause the last cycle of a non-protected instruction to become a freeze cycle.

The freeze cycle is a short cycle (4 Φ clock periods) in all cases except where B is the Decrement Scratchpad instruction, in which case the freeze cycle is a long cycle (6 Φ clock periods).

INT REQ goes low on the next negative edge of WRITE if the appropriate interrupt enable bit of the Interrupt Control Port is set. Both INT REQ and WRITE are internal signals.

Event C

A NO-OP long cycle to allow time for the internal priority chain to settle.

Event D

The program counter (P0) is pushed to the stack register (P) in order to save the return address. The interrupt circuitry places the lower 8 bits of the interrupt vector address onto the data bus. This is always a long cycle.

Event E

A long cycle in which the interrupt circuitry places the upper 8 bits of the interrupt vector address onto the data bus.

Event F

A short cycle in which the interrupting interrupt request latch is cleared. Also, the CPU's Interrupt Control Bit is cleared, thus disabling interrupts until an EI

Event F (Cont'd)

instruction is performed. The fetch of the next instruction from the interrupt address.

Event G

Begin execution of the first instruction of the interrupt service routine.

Summary Of Interrupt Sequence

For the MK3876 the interrupt response time is defined as the time elapsed between the occurence of EXT INT going active (or the Timer transitioning to H'N') and the beginning of execution of the first instruction of the interrupt service routine. The interrupt response time is a variable depedent upon what the microprocessor is doing when the interrupt request occurs. As shown in Figure 5, the minimum interrupt response time is 3 long cycles plus 2 short cycles plus one WRITE clock pulse width plus a setup time of EXT INT prior to the leading edge of the WRITE pulse – a total of 27 Φ clock periods plus the setup time. At a 2 MHz Φ this is 14.25 μs. Although the maximum could theoretically be infinite, a practical maximum is 35 μs (based on the interrupt request occurring near the beginning of a PI and LR K, P sequence).

Power-On Clear

The intent of the Power-On-Reset circuitry on the 3876 is to automatically reset the device following a typical power-up situation, thus saving external reset circuitry in many applications. This circuitry is not guaranteed to sense a "Brown Out" (low voltage) condition nor is it guaranteed to operate under all possible power-on situations.

Three conditions are required before the 3876 will leave the reset state and begin operation. Refer to Figure 7 as an aid to the following descriptions. The On-Chip Vcc detector senses a minimum value of Vcc before it will allow the 3870 to operate. The threshold of this detector is set by analog circuitry because a stable voltage reference is not available with n-channel MOS processing. Processing variations will cause this threshold to vary from a low of 3.0 volts to a high of 4.3 volts with 3.5 volts being typical.

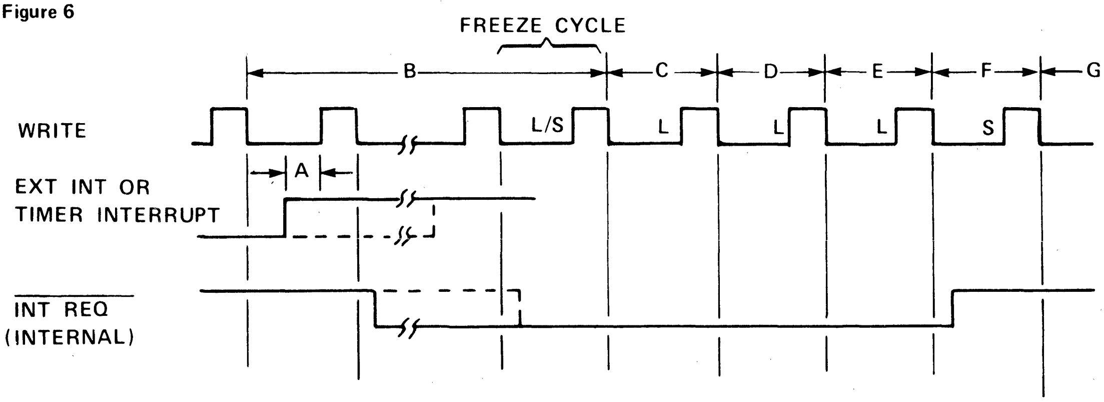
The 3876 uses a substrate bias as a technique to provide improved performance verses power consumption relative to conventional grounded substrate approaches. This bias generator may start operating as low as Vcc = 3 volts on some devices while others may require Vcc = 4 volts in order to get adequate substrate bias. Until the substrate reaches the proper bias, the 3876 will not be released from the reset state. The final condition required is that the clocks of the 3876 must be functioning. Typically the clocks will start to function at Vcc equal to 3 to 3.5 volts but since the part is tested at 4.5 volts MOSTEK cannot guarantee any operation below 4.5 volts. The output of the delay circuit in Figure 7 will stay low until the clocks start to function. If the input to the delay circuit is high, typically after 100 cycles of the WRITE clock (800 cycles of the external clock) the output of the delay circuit will go high allowing the 3876 to begin execution.

If Vcc falls to ground for at least a few hundred nanoseconds the output of the delay circuit will go low immediately and the 3876 will reset.

The internal logic may detect a valid Vcc, bias and clocks at Vcc = 3.5 volts and allow the 3876 to start executing after the time delay. With a slowly rising power supply the part may start running before Vcc is above 4.5 volts which is below the guaranteed voltage range. When power-on-clear is required with a slowly rising power supply, an external capacitor must be used on the $\overline{\text{RESET}}$ pin to hold it below 0.8

INTERRUPT SEQUENCE

Figure 6



Power-On Clear (Cont'd)

volts until Vcc is stable above 4.5 volts. (Note: The option to disconnect the internal pull-up resistor on RESET is available which allows the use of a larger external pull-up resistor and a small capacitor on RESET.)

In many applications, it is desirable if the unit does an automatic power-on-clear, but not mandatory. The unit will have a RESET push button and if the unit does not power-up correctly or malfunctions because of some disturbance on the Vcc line, the operator will simply press RESET and restore normal operation. It is for these applications that the internal power-on-clear circuitry was designed.

In some applications it is required that the microcomputer continue to run properly without operator intervention after brown-outs, power line disturbances, electrical noise, computer malfunction due to a programming bug or any other disturbance except a catastrophic failure of some component.

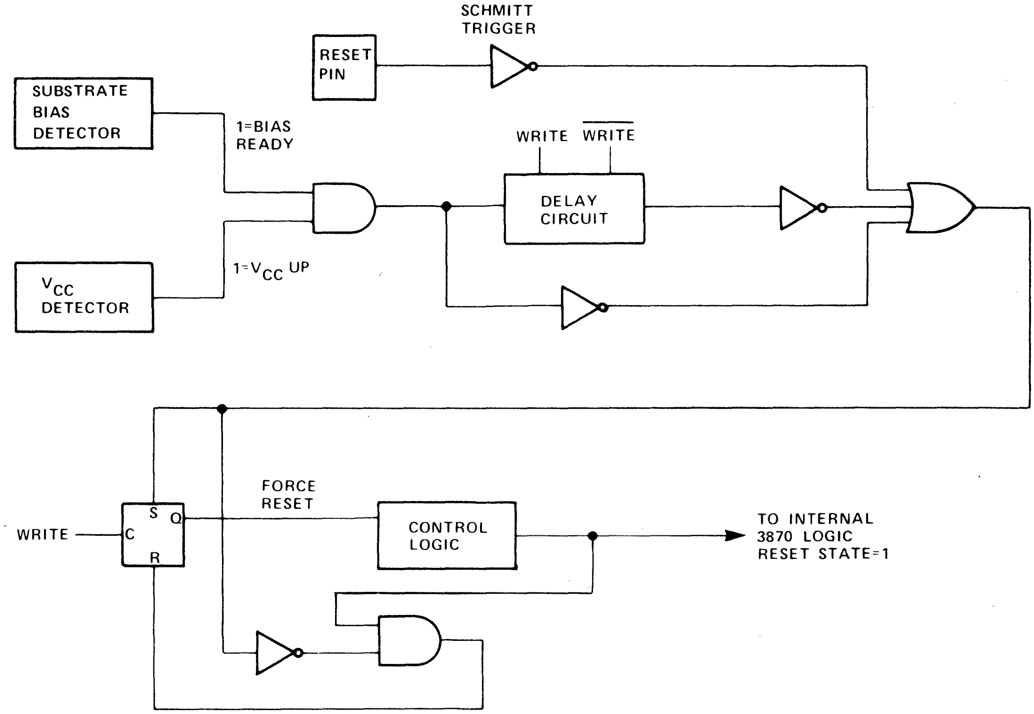
One concept used to keep computers running is that of the "WATCHDOG TIMER". The computer is programmed to periodically reset the watchdog timer during the normal execution of its program (this is easily done in the 3876 as its normal application is in some control function which is typically periodic). As long as the computer continues to execute its program the watchdog timer is continually reset and never times out. Should the computer stop executing its program for whatever reason, the watchdog timer will time out producing a RESET pulse to the CPU restarting execution. This is a very positive way to assure that the computer is doing its job, i.e., executing the program. It is important that the software driving the watchdog timer test as many functional blocks (timer, ALU, scratchpad RAM, and Ports) of the 3876 as possible before reseting the watchdog timer. This is because operation of the 3876 with an out of spec power supply may allow some of the funcitons to operate correctly while other functions are not operable.

MOSTEK can guarantee correct operation of the 3876 only while the Vcc voltage remains within its specified limits. If proper operation of the 3876 must be guaranteed after a disturbance on the Vcc line, then an external circuit must be used to monitor the Vcc line and produce a RESET to the 3876 whenever Vcc is out of the specified limits.

A related characteristic to power-on-clear is the Start-up time of the basic timing element. The LC and RC oscillators begin to function almost immediately once Vcc is high enough to allow the on-board

POWER ON CLEAR BLOCK DIAGRAM

Figure 7



Power-On Clear (Cont'd)

oscillator to operate (Vcc = 3.5V). Operation with a crystal is partly mechanical and some start time is required to get the mass of the crystal into vibrational motion. This time is basically dependent on the frequency (mass) of the crystal. 4 MHz crystals typically require about 2-3 mSec to start while 1 MHz crystals require 60-70 mSect to start oscillating. Of course, this time may vary greatly from crystal to crystal and is also a function of the power supply rise time characteristic, however, the higher frequency crystals start faster and are definitely recommended (i.e., 3-4 MHz).

The condition of the port pins during the power-on-clear sequence is often asked, The port pins or the STROBE line cannot be specified until Vcc reaches 4.5V and the 3876 enters the RESET state. Before this, the port pins may stay at Vss, may track Vcc as it rises, or they may track Vcc part way up then return to Vss (Ports 4 and 5 will go to Vcc once the clocks are running and the 3870 has sufficient Vcc to properly operate the internal control logic and I/O ports, Ports 0 and 1 must be controlled by the program).

External Reset

When $\overline{\text{RESET}}$ is taken low the content of the Program Counter is pushed to the Stack Register and then the Program Counter and the ICB bit of the W Status Register are cleared. The original Stack Register content is lost. Ports 4, 5, 6, and 7 are loaded with H '00'. The contents of all other registers and ports are unchanged. When power is first applied all ports and registers are undefined until a reset is performed. When $\overline{\text{RESET}}$ is taken high the first program instruction is fetched from ROM location H '000'. When an external reset of the 3876 occurs, P0 is pushed into P and the old contents of P are lost. It must be noted that an external reset is recognized at the start of a machine cycle and not necessarily at the end of an instruction. Thus if the 3876 is executing a multi-cycle instruction, that instruction is not completed and the contents of P upon reset may not necessarily be the address of the instruction that would have been executed next. It may, for example, point to an immediate operand if the reset occurred during the second cycle of a LI or CI instruction. Additionally, several instructions (JMP, PI, PK, LR P0, Q) as well as the interrupt acknowledge sequence modify P0 in parts. That is, they alter P0 by first loading one part then the other and the entire operation takes more than one cycle. Should reset occur during this modification process the value pushed into P will be part of the old P0 (the as yet unmodified part) and part of the new P0 (already modified part). Thus care should be taken (perhaps by external gating) to insure that reset does not occur at an undesirable time if any significance is to be given to the contents of P after a reset occurs.

V_{CC} Decoupling

The 3870 family devices have dynamic circuitry internally which requires a good high frequency decoupling capacitor to surpress noise on the Vcc line. A .01 μF or .1 μF ceramic capacitor should be placed between Vcc and ground, located physically close to the 3870 device. This will reduce noise generated by the 3870 to about 70-100 mVolts on the Vcc line.

Test Logic

Special test logic is implemented to allow access to the internal main data bus for test purposes.

In normal operation the TEST pin is unconnected or is connected to GND. When TEST is placed at a TTL level (2.0V to 2.6V) Port 4 becomes an output of the internal data bus and Port 5 becomes a wired-OR input to the internal data bus. The data appearing on the Port 4 pins is logically true whereas input data forced on Port 5 must be logically false. When TEST is placed at high level (6.0V to 7.0V), the ports act as above and additionally the 2K x 8 program ROM is prevented from driving the data bus. In this mode operands and instructions may be forced externally through Port 5 instead of being accessed from the program ROM. When TEST is in either the TTL state or the high state, $\overline{\text{STROBE}}$ ceases its normal function and becomes a machine cycle clock (identical to the F8 multi-chip system WRITE clock except inverted).

Timing complexities render the capabilities associated with the TEST pin impractical for use in a user's application, but these capabilities are thoroughly sufficient to provide a rapid method for thoroughly testing the 3876.

STANDBY POWER OPTION

If the standby power option has not been selected Port 0-bit 0 and 1 are readable and writeable.

If the standby power option is selected Port 0-Bit 1 is readable only. Port 0-Bit 0 remains readable and writable although it is not connected to a package pin. The standby power source (V_{SB}) is connected to Pin 4.

A .01μF capacitor must be connected to Pin 3. The purpose of the capacitor is to decouple noise coupled to the substrate of the circuit when V_{CC} is switched off and on. It is recommended that Nickel-Cadmium batteries (typical voltage of 3 series cells = 3.6V) be used for standby power, since the MK3876 can automatically trickle charge the three Ni-Cad's. If more than three cells in series are used, the charging circuit must be provided outside the MK3876. Whenever $\overline{\text{RESET-RAMPRT}}$ is brought low, the standby RAM

STANBY POWER OPTION (Cont'd)

(64x8 bit words in PO/DC address space, 4032 to 4095_{10} or FCO to FFF_{16}) is placed in a protected state. Also the RAM itself is switched from V_{CC} power to the V_{SB} power. Two modes of powering down are recommended. In the first mode, the processor must be interrupted early enough to save all necessary data before the V_{CC} falls below the minimum level. After the save is done, $\overline{\text{RESET}}$ can fall. This prevents any further access of the RAM; V_{CC} may now fall. As the power comes up, the $\overline{\text{RESET}}$/ $\overline{\text{RAMPRT}}$ signal should be held low until V_{CC} is above the minimum level.

The second mode may be used if a special save data routine is not needed. The $\overline{\text{EXT}}$ $\overline{\text{INTERRUPT}}$ need not be used and the only requirement to save the RAM data is that $\overline{\text{RESET-RAMPRT}}$ be low before V_{CC} drops below 4.5V. For example if a few key variables are to be stored in RAM and it is desired that these be saved during a loss of power, two copies of each variable are kept with an associated flag, thus no interrupt and save routine is necessary. The method of updating a variable is as follows:

- Clear Flag Word 1
- Update Variable (Copy 1)
- Set Flag Word 1
- Clear Flag Word 2
- Update Variable (Copy 2)
- Set Flag Word 2

Now execution may terminate at any time, even during the update of a variable or flag word, causing that byte in RAM to be bad data. There is always a good data byte which contains either the most recent or next most recent value of the variable. Any copy of the variable where the flag word is "set" is a good data byte. While this method significantly encumbers the data storage process, it eliminates the need for a power fail interrupt which both reduces external circuitry and leaves the external interrupt pin completely free for other use.

Figure 9 represents the internal circuitry which can be connected to pins 3, 4, and 39 to provide this Standby Mode. If the Standby Mode is selected, switches A1 and A2 are masked in the position shown, thereby disconnecting the normal port circuitry from pins 3 and 4. Switches B1 and B2 are masked in the position shown to allow pin 39 to become the control (RAMPRT) and pin 4 the power (V_{SB}) for the Standby Mode. If the Standby Mode is not selected all switches are masked opposite of the positions shown and pins 3 and 4 become normal 3870 type ports.

$\overline{\text{RAMPRT}}$ is an input signal used to control access to the Standby RAM. If $\overline{\text{RAMPRT}}$ is high, access to the 64 Byte Standby RAM is permitted by the CPU via the Program Counter (PO) of the Data Counter (DC). The Standby RAM current is supplied by the series pass transistor and a 4 to 12 mA current can be supplied out of pin 4 (Vsb) to trickle charge two Ni-Cad cells (nominal 2.5 volts). The resistors shown simulate device impedances that limit the current available at pin 4 so that the battery is not overcharged. If $\overline{\text{RAMPRT}}$ is low, the Control Logic turns off the pass transistor and the Standby RAM is maintained by a current supplied by the battery connected to pin 4. When $\overline{\text{RAMPRT}}$ is low, the CPU cannot access the Standby RAM thereby protecting its contents as Vcc fails.

The Standby RAM can be maintained by a capacitor, however, a resistor and a diode will be required in order to charge the capacitor to Vcc. Internal voltage drops will not allow Vsb to go above 3 volts (typically) without this external resistor.

3876 Clocks

The time base for the 3876 may originate from one of four sources.

The four configurations are shown in Figure 10. There is an internal 26pF capacitor between XTL 1 and GND and an internal 26pF capacitor between XTL 2 and GND, thus external capacitors are not neccessarily required. In all external clock modes the external time base frequently is divided by two to form the internal Φ clock.

Crystal Selection

The use of a crystal as the time base is highly recommended as the frequency stability and reproducability from system to system is unsurpassed. The 3876 has an internal divide by two to allow the user of inexpensive and widely available TV Color Burst Cyrstals (3.58MHz). The following crystal parameters and vendors are suggested for 3876 applications:

Parameters

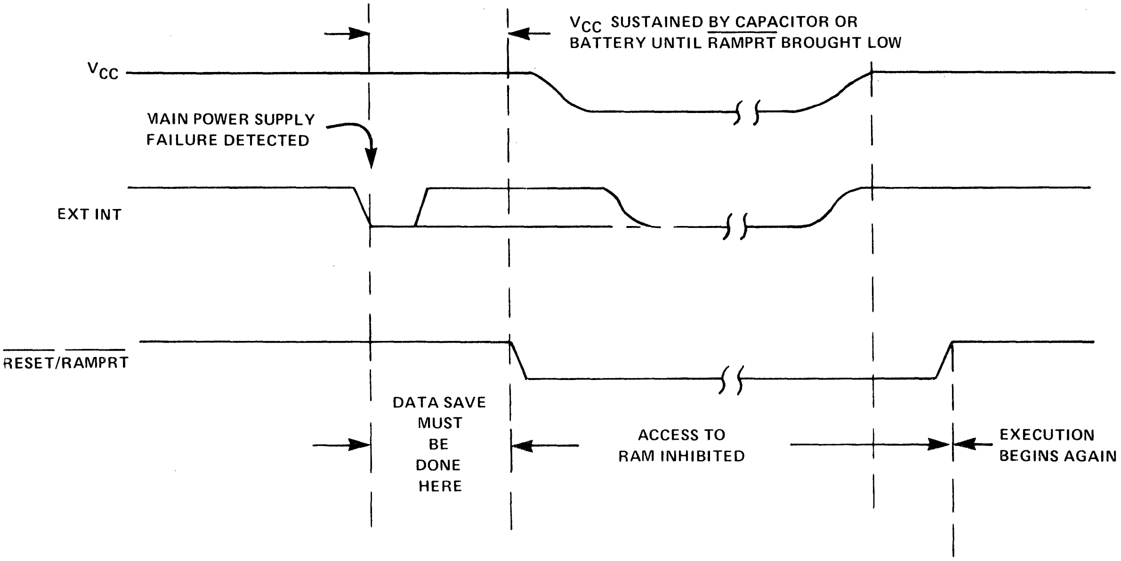
a) Parallel Resonance, Fundamental Mode AT-Cut
b) Frequency Tolerance measured with 18pF load (0.1% accuracy). Drive level 10mW.
c) Shunt Capacitance (Co) = 7pF max.
d) Series Resistance (Rs)

		Holder
f = 1MHz	Rs = 550 ohms max.	HC-6
f = 2MHz	Rs = 300 ohms max.	HC-33
f = 3MHz	Rs = 150 ohms max.*	HC-6
f = 3.58MHz	Rs = 150 ohms max.	HC-18
f = 4MHz	Rs = 150 ohms max.	HC-25 HC-33

*HC-18 or HC-25 holder may not be available at 3MHz.

SAVE ROUTINE REQUIRED, $V_{SB} \geqslant 2.2$ VOLTS

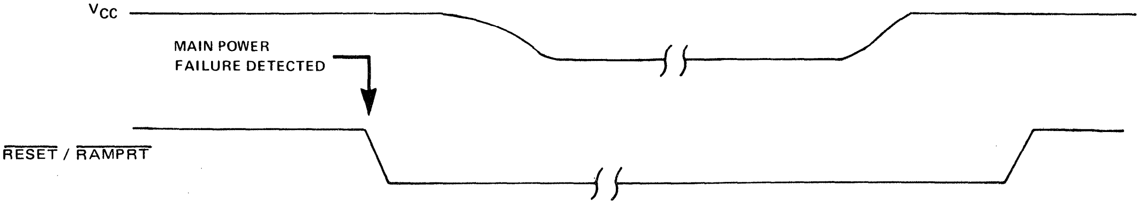
Figure 8a



V_{CC} SUSTAINED BY CAPACITOR OR BATTERY UNTIL $\overline{RAMPRT}$ BROUGHT LOW
V_{CC}
MAIN POWER SUPPLY FAILURE DETECTED
EXT INT
$\overline{RESET}/\overline{RAMPRT}$
DATA SAVE MUST BE DONE HERE
ACCESS TO RAM INHIBITED
EXECUTION BEGINS AGAIN

NO SAVE ROUTINE REQUIRED, $V_{SB} \geqslant 2.2$ VOLTS

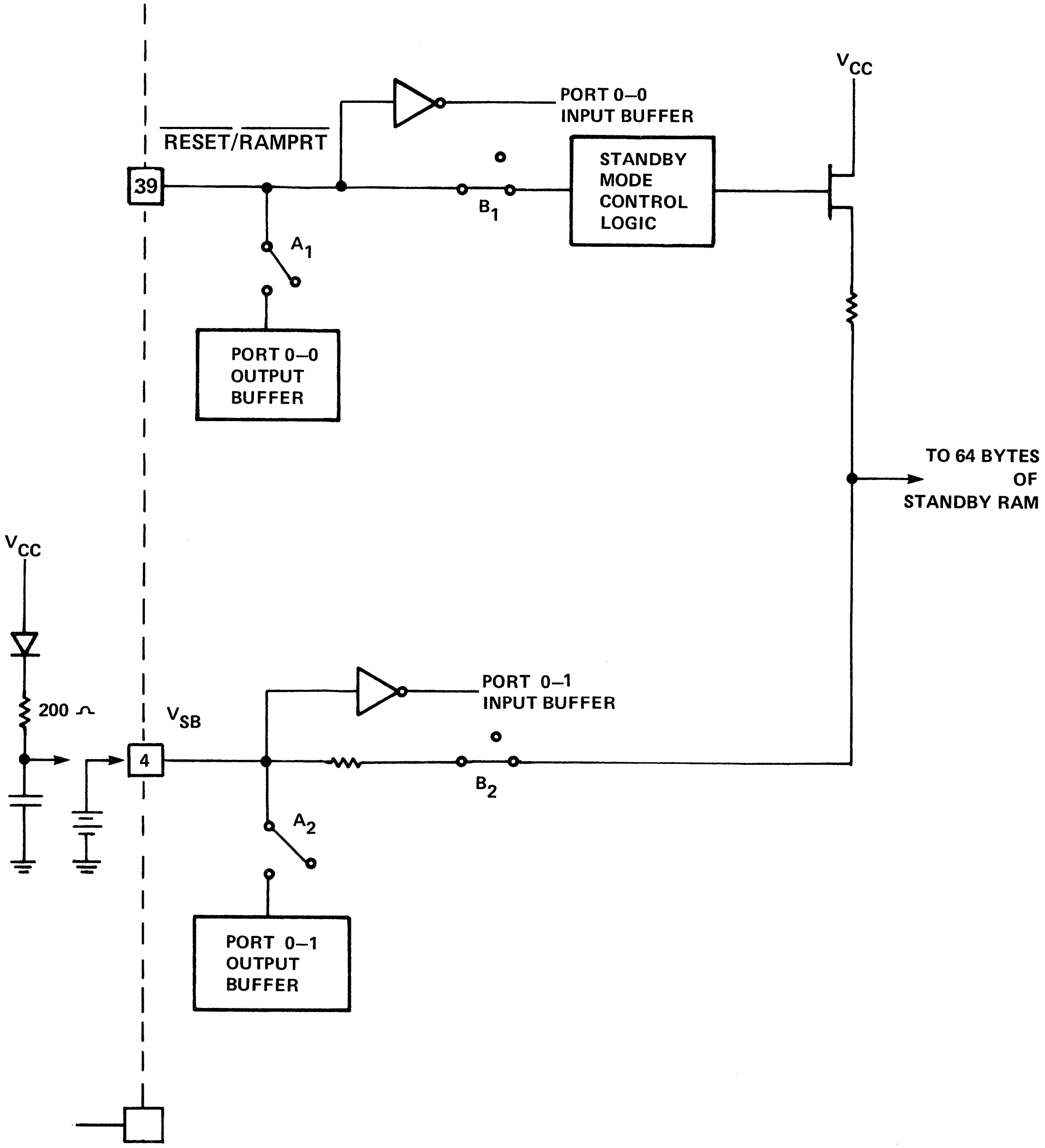
Figure 8b



V_{CC}
MAIN POWER FAILURE DETECTED
$\overline{RESET}$ / $\overline{RAMPRT}$

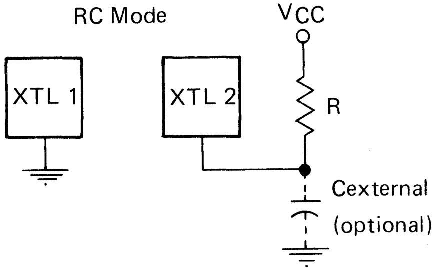
MK3876 STANDBY MODE CIRCUITRY

Figure 9

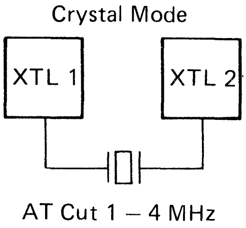


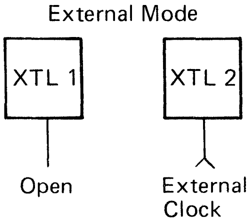
CLOCK CONFIGURATIONS

Figure 10

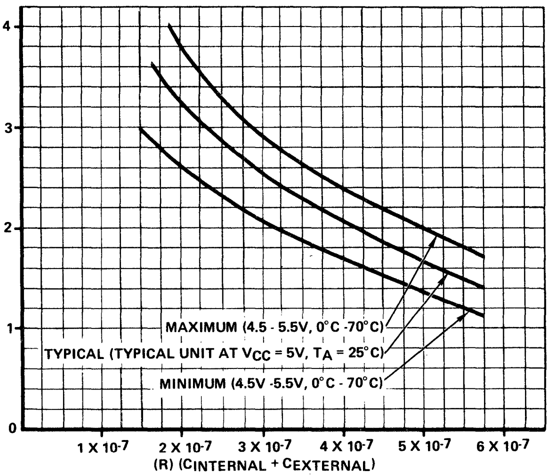


Minimum R = 4KΩ
C = 26.5pF ± 2.6pF + $C_{EXTERNAL}$



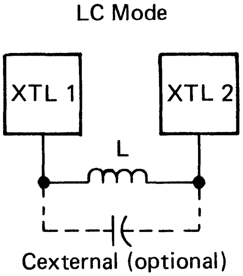


FREQUENCY VRS RC



UNIT TO UNIT VARIATION = ± 12%
VARIATION FROM 4.5 to 5.5V
REFERENCED TO 5V = +7% -4%
VARIATION FROM 0°C TO 70°C
REFERENCED TO 25°C = +6% -9%

TOTAL VARIATION NOT CONSIDERING
VARIATION IN EXTERNAL COMPONENTS = ± 25%



Minimum L = 0.1 mH
Minimum Q = 40

Maximum Cexternal = 30pF

C = 13pF ± 1.3pF + Cexternal

$$f \cong \frac{1}{2\pi\sqrt{LC}}$$

NOTE: The stray capacitance across the inductor must be included as Cexternal in all calculations.

Suggested Crystal Vendors

a) Electro-Dynamics
5625 Foxridge Drive
Mission, Kansas 66201
913-262-2500

b) CRYSTEK
1000 Crystal Drive
Ft. Myers, Florida 33901
813-936-2109

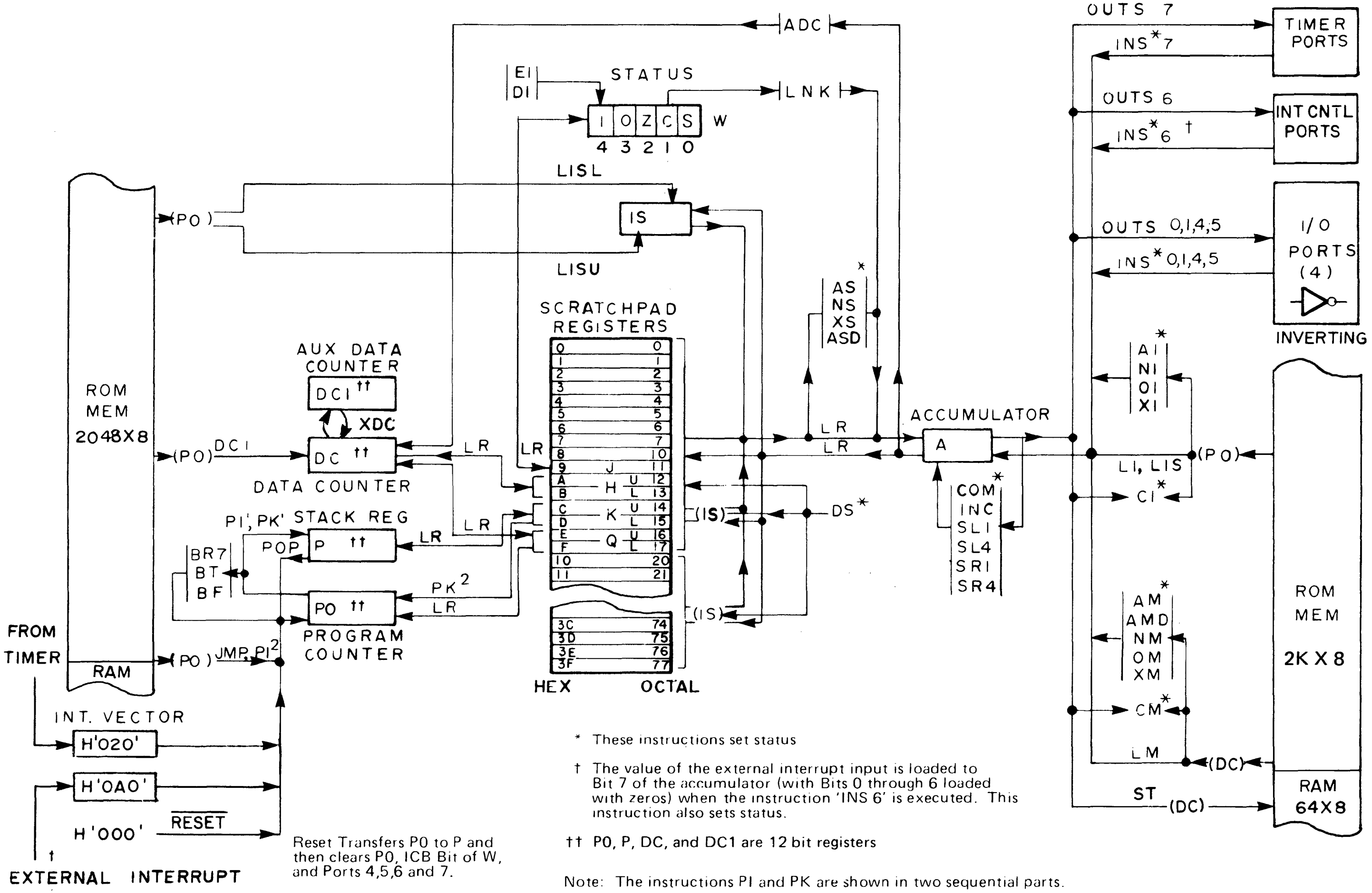
c) W.T. Liggett Corp.
1500 Worcester Rd.
Section 30
Framingham, MA 01701
617-620-1150

d) Erie Frequency Control
453 Lincoln Street
Carlisle, Penn 17013
717-249-2232

e) Electronic Crystals Corp.
1153 Southwest Blvd.
Kansas City, Kansas 66103
913-262-1274

f) M-TRON Industries
P.O. Box 630
100 Douglas Avenue
Yankton, South Dakota
605-665-9321

MK3876 PROGRAMMING MODEL
Figure 11



Reset Transfers P0 to P and then clears P0, ICB Bit of W, and Ports 4,5,6 and 7.

* These instructions set status

† The value of the external interrupt input is loaded to Bit 7 of the accumulator (with Bits 0 through 6 loaded with zeros) when the instruction 'INS 6' is executed. This instruction also sets status.

†† P0, P, DC, and DC1 are 12 bit registers

Note: The instructions PI and PK are shown in two sequential parts. (PI1, PI2 and PK1, PK2).

INSTRUCTION EXECUTION

This section details the timing and execution of the 3876 instruciton set. The 3876 executes the entire F8 instruction set with exact F8 timing. Refer to Figure 11 for a 3876 Programming Model.

F8 INSTRUCTION SET

ACCUMULATOR GROUP INSTRUCTIONS

	MNEMONIC			MACHINE		CYCLES		μS		STATUS BITS		
OPERATION	OP CODE	OPERAND	FUNCTION	CODE	BYTES	SHORT	LONG	(2MHzΦ)	OVR	ZERO	CRY	SIGN
Add Carry	LNK		A←(A) + CRY	19	1	1		2	1/0	1/0	1/0	1/0
Add Immediate	AI	ii	A←(A) + H 'ii'	24ii	2	1	1	5	1/0	1/0	1/0	1/0
And Immediate	NI	ii	A←(A)∧H 'ii'	21ii	2	1	1	5	0	1/0	0	1/0
Clear	CLR		A←H'00'	70	1	1		2	–	–	–	–
Compare Immediate	CI	ii	H'ii'+ (A) + 1	25ii	2	1	1	5	1/0	1/0	1/0	1/0
Complement	COM		A←(A) + H'FF'	18	1	1		2	0	1/0	0	1/0
Exclusive or Immediate	XI	ii	A←(A) + H'ii'	23ii	2	1	1	5	0	1/0	0	1/0
Increment	INC		A←(A) + 1	1F	1	1		2	1/0	1/0	1/0	1/0
Load Immediate	LI	ii	A←H'ii'	20ii	2	1	1	5	–	–	–	–
Load Immediate Short	LIS	i	A←H' 0i'	7i	1	1		2				
OR Immediate	OI	ii	A←(A) V H 'ii'	22ii	2	1	1	5	0	1/0	0	1/0
Shift Left One	SL	1	Shift Left 1	13	1	1		2	0	1/0	0	1/0
Shift Left Four	SL	4	Shift Left 4	15	1	1		2	0	1/0	0	1/0
Shift Right One	SR	1	Shift Right 1	12	1	1		2	0	1/0	0	1
Shift Right Four	SR	4	Shift Right 4	14	1	1		2	0	1/0	0	1

BRANCH INSTRUCTIONS In all conditional branches P0←(P0) + 2 if the test condition is not met. Execution is complete in 3 short cycles.

	MNEMONIC			MACHINE		CYCLES		μS		STATUS BITS		
OPERATION	OP CODE	OPERAND	FUNCTION	CODE	BYTES	SHORT	LONG	(2MHzΦ)	OVR	ZERO	CRY	SIGN
Branch on Carry	BC	aa	P0←(P0) +1+ H'aa' if CRY 1	82aa	2	2	1	7	—	—	—	—
Branch on Positive	BP	aa	P0←(P0) + 1 + H'aa ' if SIGN = 1	81aa	2	2	1	7	—	—	—	—
Branch on Zero	BZ	aa	P0←(P0) + 1 + H'aa' if Zero = 1	84aa	2	2	1	7	—	—	—	—
Branch on True	BT t-TEST CONDITION 2^2 | 2 | 2^0 ZERO | CRY | SIGN	taa	P0←(P0) + 1 + H'aa ' if any test is true	8taa	2	2	1	7	—	—	—	—
Branch If Negative	8M	aa	P0←(P0) +1+ H'aa ' if SIGN =0	91aa	2	2	1	7	—	—	—	—
Branch if No Carry	BNC	aa	P0←(P0) +1+ H'aa ' if CARRY = 0	92aa	2	2	1	7	—	—	—	—
Branch if No Overflow	BNO	aa	P0←(P0) +1+ H'aa ' if OVR = 0	98aa	2	2	1	7	—	—	—	—
Branch if Not Zero	BNZ	aa	P0←(P0)+1+ H'aa ' if ZERO = 0	94aa	2	2	1	7	—	—	—	—
Branch if False Test	BF t TEST CONDITION 2^3 | 2^2 | 2^1 | 2^0 OVF | ZERO | CRY | SIGN	taa	P0←(P0) +1+ H'aa ' if all false test bits	9taa	2	2	1	7	—	—	—	—
Branch if ISAR (Lower) ≠7	BR7	aa	P0←(P0)+1+ H'aa ' if ISARL ≠7	87aa	2	1	1	5	—	—	—	—
			P0←(P0)+2 if ISARL =		2	2		4	—	—	—	—
Branch Relative	BR	aa	P0←(P0)+1+ H'aa '	90aa	2	2	1	7	—	—	—	—
Jump*	JMP	aaaa	P0←H'aaaa'	29aaaa	3	1	3	11	—	—	—	—

*Privileged instruction, Accumulator contents altered during execution JMP instruction.

MEMORY REFERENCE INSTRUCTIONS In all Memory Reference Instructions, the Data Counter is incremented DC (DC)+1

	MNEMONIC			MACHINE		CYCLES		µS		STATUS BITS		
OPERATION	OP CODE	OPERAND	FUNCTION	CODE	BYTES	SHORT	LONG	(2MHzΦ)	OVR	ZERO	CRY	SIGN
Add Binary	AM		A←(A) + [(DC)]	88	1	1	1	5	1/0	1/0	1/0	1/0
Add Decimal	AMD		A←(A) + [(DC)] • BCD Adjust	89	1	1	1	5	?	?	1/0	?
AND	NM		A←(A) ∧ [(DC)]	8A	1	1	1	5	0	1/0	0	1/0
Compare	CM		[(DC)] + ($\overline{A}$) + 1	8D	1	1	1	5	1/0	1/0	1/0	1/0
Exclusive OR	XM		A←(A)⊕[(DC)]	8C	1	1	1	5	0	1/0	0	1/0
Load	LM		A←[(DC)]	16	1	1	1	5	—	—	—	—
Logical OR	OM		A← (A) V '(DC)]	8B	1	1	1	5	0	1/0	0	1/0
Store	ST		A→[(DC)]	17	1	1	1	5	—	—	—	—

ADDRESS REGISTER GROUP INSTRUCTIONS

	MNEMONIC			MACHINE		CYCLES		µS		STATUS BITS		
OPERATION	OP CODE	OPERAND	FUNCTION	CODE	BYTES	SHORT	LONG	(2MHzΦ)	OVR	ZERO	CRY	SIGN
Add to Data Counter	ADC		DC←(DC) + (A)	8E	1	1	1	5	—	—	—	—
Call to Subroutine*	PK		P0U←(r12); P0L←(r13), P←(P0)	0C	1	1	2	8	—	—	—	—
Call to Subroutine Immediate*	PI	aaaa	P← (P0), P0←H'aaaa	28aaaa	3	2	3	13	—	—	—	—
Exchange DC	XDC		(DC)⇄(DC1)	2C	1	2		4	—	—	—	—
Load Data Counter	LR	DC,Q	DCU←(r14); DCL←(r15)	0F	1	1	2	8	—	—	—	—
Load Data Counter	LR	DC'H	DCU←(r10); DCL←(r11)	10	1	1	2	8	—	—	—	—
Load DC Immediate	DCI	aaaa	DC H'aaaa'	2Aaaaa	3	3	2	12	—	—	—	—
Load Program Counter	LR	P0,Q	P0U←(r14); P0L←(r15)	0D	1	1	2	8	—	—	—	—
Load Stack Register	LR	P,K	PU←(r12); PL←(r13)	09	1	1	2	8	—	—	—	—
Return from Subroutine*	POP		P0←(P)	1C	1	2		4	—	—	—	—
Store Data Counter	LR	Q,DC	r14←(DCU); r15←(DCL)	0E	1	1	2	8	—	—	—	—
Store Data Counter	LR	H,DC	r10←(DCU); r11←(DCL)	11	1	1	2	8	—	—	—	—
Store Stack Register	LR	K,P	r12←(PU); r13←(PL)	08	1	1	2	8	—	—	—	—

SCRATCHPAD REGISTER INSTRUCTIONS (Refer to Scratchpad Addressing Modes)

	MNEMONIC			MACHINE		CYCLES		µS		STATUS BITS		
OPERATION	OP CODE	OPERAND	FUNCTION	CODE	BYTES	SHORT	LONG	(2MHzΦ)	OVR	ZERO	CRY	SIGN
Add Binary	AS	r	A←(A)+ (r)	Cr	1	1		2	1/0	1/0	1/0	1/0
Add Decimal	ASD	r	A←(A) + (r)	Dr	1	2		4	?	?	1/0	?
Decrement	DS	r	r←(r) + H'FF'	3r	1		1	3	1/0	1/0	1/0	1/0
Load	LR	A,r	A←(r)	4r	1	1		2	—	—	—	—
Load	LR	A, KU	A←(r12)	00	1	1		2	—	—	—	—
Load	LR	A, KL	A←(r13)	01	1	1		2	—	—	—	—
Load	LR	A, QU	A←(r14)	02	1	1		2	—	—	—	—
Load	LR	A, QL	A←(r15)	03	1	1		2	—	—	—	—
Load	LR	r, A	r←(A)	5r	1	1		2	—	—	—	—
Load	LR	KU, A	r12←(A)	04	1	1		2	—	—	—	—
Load	LR	KL, A	r13←(A)	05	1	1		2	—	—	—	—
Load	LR	QU, A	r14←(A)	06	1	1		2	—	—	—	—
Load	LR	QL,A	r15←(A)	07	1	1		2	—	—	—	—
And	NS	r	A ← (A) ∧ (r)	Fr	1	1		2	0	1/0	0	1/0
Exclusive Or	XS	r	A ←(A) + (r)	Er	1	1		2	0	1/0	0	1/0

*Privileged instruction, Accumulator contents altered during execution of PI instruction.

MISCELLANEOUS INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	µS (2MHzΦ)	OVR	STATUS BITS ZERO	CRY	SIGN
Disable Interrupt	DI		RESET ICB	1A	1	1		2	—	—	—	—
Enable Interrupt *	EI		SET ICB	1B	1	1		2	—	—	—	—
Input	IN	04,05,06,07	A←(Input Port aa)	26aa	2	1	2	8	0	1/0	0	1/0
Input Short	INS	0, 1	A←(Input Port 0 or 1)	A0,A1	1	2		4	0	1/0	0	1/0
Input Short	INS	4,5,6,7	A←(Input Port a)	Aa	1	1	2	8	0	1/0	0	1/0
Load ISAR	LR	IS,A	IS←(A)	0B	1	1		2	—	—	—	—
Load ISAR Lower	LISL	bbb	ISL←bbb	6(1bbb)**	1	1		2	—	—	—	—
Load ISAR Upper	LISU	bbb	ISU←bbb	6(0bbb)**	1	1		2	—	—	—	—
Load Status Register *	LR	W,J	W←(r9)	1D	1	2		4	1/0	1/0	1/0	1/0
No Operation	NOP		P0←(P0) + 1	2B	1	1		2	—	—	—	—
Output *	OUT	04,05,06,07	Output Port aa←(A)	27aa	2	1	2	8	—	—	—	—
Output Short	OUTS	0, 1	Output Port 0 or 1←(A)	B0, B1	1	2		4	—	—	—	—
Output Short*	OUTS	4,5,6,7	Output Port a←(A)	Ba	1	1	2	8	—	—	—	—
Store ISAR	LR	A,IS	A←(IS)	0A	1	1		2	—	—	—	—
Store Status Reg	LR	J,W	r9←(W)	1E	1	1		2	—	—	—	—

*Privileged instruction

**b = 1 bit immediate operand

NOTES.

Lower case denotes variables specified by programmer

Function Definitions

←	is replaced by
( )	the contents of
(‾)	Binary "1's" complement of
+	Arithmetic Add (Binary or Decimal)
⊕	Logical "OR" exclusive
Λ	Logical "AND"
V	Logical "OR" inclusive
H' '	Hexadecimal digit
[( )]	Contents of memory specified by ( )
a	Address Variable (four bits)
A	Accumulator
b	One bit immediate operand
DC	Data Counter (Indirect Address Register)
DC1	Data Counter 1 (Auxiliary Data Counter)
DCL	Least significant 8 bits of Data Counter Addressed
DCU	Most significant 8 bits of Data Counter Addressed
H	Scratchpad Register 10 and 11
i	Immediate operand (four bits)
ICB	Interrupt Control Bit
IS	Indirect Scratchpad Address Register
ISL	Least Significant 3 bits of ISAR
ISU	Most Significant 3 bits of ISAR
J	Scratchpad Register 9
K	Registers 12 and 13
KL	Register 13
KU	Register 12
P0	Program Counter
P0L	Least Significant 8 bits of Program Counter
P0U	Most Significant 8 bits of Program Counter
P	Stack Register
PL	Least Significant 8 bits of Program Counter
PU	Most Significant 8 bits of Active Stack Register
Q	Registers 14 and 15
QL	Register 15
QU	Register 14
r	Scratchpad Register (any address 0 thru B) (See Below)
W	Status Register

Scratchpad Addressing Modes Using IS. (r ≠ 0 thru B)

r=H'C'	Register Addressed by IS is (Unmodified)
r=H'D'	Register Addressed by IS is Incremented
r=H'E'	Register Addressed by IS is Decremented
r=H'F'	Illegal OP Code.

Status Register

—	No change in condition
1/0	is set to "1" or"0" depending on conditions
CRY	Carry Flag
OVR	Overflow Flag
SIGN	Sign of Result Flag
ZERO	Zero Flag

ELECTRICAL SPECIFICATIONS
ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	0°C to 70°C
Storage Temperature	–65°C to +150°C
Voltage on Any Pin With Respect To Ground (except open drain pins)	–1.0V to +7V
Voltage On Open Drain Pins	–1.0V to +13.5V
Power Dissipation	1.5W
Power Dissipated by any one I/O pin[4]	60mW
Power Dissipated by all I/O pins[4]	600mW

A.C. CHARACTERISTICS – See Figure 12 and 13 for Timing Diagrams

T_A = 0°C to 70°C, V_{CC} = 5V ± 10%, I/O POWER DISSIPATION ≤ 100mW

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
XTL 1 XTL 2	$t_o(EX)$	Time base period, all external modes	250	1000	ns	4MHz-1MHz
	$t_{EX(H)}$	External Clock Pulse Width High	90	700	ns	
	$t_{EX(L)}$	External Clock Pulse Width Low	100	700	ns	
Φ	$t_Φ$	Internal Φ Clock Period	$2t_0$			
WRITE	tw	Internal WRITE Clock Period	4tΦ 6tΦ			Short Cycle Long Cycle
I/O	$t_{dI/O}$	Output delay from internal WRITE Clock	0	1000	ns	50pF plus one TTL load
	$t_{sI/O}$	Input Setup time to WRITE Clock	1000		ns	
$\overline{\text{STROBE}}$	$t_{I/O\text{-}s}$	Output valid to STROBE Delay	3tΦ -1000	3tΦ +250		I/O load = 50pF + 1 TTL STROBE Load= 50pF + 3 TTL
	tsl	$\overline{\text{STROBE}}$ Low Time	8tΦ -250	12tΦ +250	ns	
$\overline{\text{RESET}}$	t_{RH}	$\overline{\text{RESET}}$ Hold Time, Low	6tΦ +750		ns	
EXT INT	t_{EH}	EXT INT Hold Time, Active and Inactive State	6tΦ + 750		ns	To trigger interrupt
			2tΦ			To trigger timer

TIMER AC CHARACTERISTICS

Definitions:

Error = Indicated time value - actual time value

tpsc = tΦ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3)	±6tΦ
Cumulative interval error, free running (Note 3)	0
Error between two Timer reads (Note 2)	±(tpsc + tΦ)
Start Timer to stop Timer error (Notes 1,4)	+tΦ to –(tpsc +tΦ)
Start Timer to read Timer error (Notes 1,2)	–5tΦ to –(tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1,3)	–2tΦ to –8tΦ
Load Timer to stop Timer error (Note 1)	+tΦ to –(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1,2)	–5tΦ to –(tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1,3)	–2t Φ to –9tΦ

Pulse Width Measurement Mode:

Measurement accuracy (Note 4)	+t Φto –(tpsc +2t Φ)
Minimum pulse width of EXT INT pin	2t Φ

Event Counter Mode:

Minimum active time of EXT INT pin	2tΦ
Minimum inactive time of EXT INT pin	2tΦ

Notes:

1. All times which entail loading, starting, or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.

CAPACITANCE

T_A = 25°C, f=2MHz

SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input Capacitance: I/O Ports, $\overline{\text{RESET}}$/ $\overline{\text{RAMPRT}}$, $\overline{\text{EXTINT}}$, TEST		7	pF	Unmeasured Pins Grounded
C_{XTL}	Input Capacitance: XTL1, XTL2	20.5	32.5	pF	

DC CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 10%, I/O POWER DISSIPATION $\leqslant$100mW

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
I_{CC}	Power Supply Current		93	mA	Outputs Open
P_D	Power Dissipation		440	mW	Outputs Open

DC CHARACTERISTICS (Cont'd)

SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
V_{IHEX}	External Clock Input High Level	2.4	5.8	V	
V_{ILHEX}	External Clock Input Low Current	–0.3	0.6	V	
I_{IHEX}	External Clock Input High Current		100	µA	$V_{IHEX} = V_{CC}$
I_{ILEX}	External Clock Input Low Current		-100	µA	$V_{ILEX} = V_{SS}$
V_{IH}	Input High Level Ports, $\overline{\text{RESET}}^1$, EXT INT^1	2.0	5.8	V	
V_{IHOD}	Open Drain Input High Level	2.0	13.2	V	
V_{IL}	Input Low Level Ports, $\overline{\text{RESET}}^1$, EXT INT^1	–0.3	0.8		
I_{IL}	Input Low Current Ports, $\overline{\text{RESET}}^2$, EXT INT^2		–1.6	mA	V_{IL}=0.4V
I_L	Leakage Current Open drain ports, $\overline{\text{RESET}}^3$, EXT INT^3		+10 –5	µA	V_{IN}=13.2V V_{IN}=0.0V
I_{OH}	Output High Current Standard ports, $\overline{\text{RESET}}^2$ EXT INT^2	–100 –30		µA µA	V_{OH}=2.4V V_{OH}=3.9V
I_{OHDD}	OUTPUT High Current Direct Drive Ports	–0.1		mA	V_{OH} = 2.4V
		–1.5		mA	V_{OH}=1.5V
			–8.5	mA	V_{OH}=.7V
I_{OL}	Output Low Current IO ports	1.8		mA	V_{OL}=0.4V
I_{OHS}	$\overline{\text{STROBE}}$ Output High Current	–300		µA	V_{OH}=2.4V
I_{OLS}	$\overline{\text{STROBE}}$ Output Low Current	5.0		mA	V_{OL} = 0.4V
V_{IHRPR}	Input High Level For RAM Protect Function To be effective.	1.9	5.8	V	Guaranteed .1V less than V_{IH} for $\overline{\text{RESET}}$
V_{ILRPR}	Input High Level For RAM Protect Function To be effective.	–0.3	0.4	V	Guaranteed .1V less than V_{IL} for $\overline{\text{RESET}}$
V_{SB}	Standby V_{CC} for RAM	3.2	5.5	V	
I_{SB}	Standby current		6 3.7	mA mA	V_{SB} = 5.5V V_{SB} = 2.2V
I_{CHARGE}	Trickle charge available on V_{SB} with V_{CC}=4.5 to 5.5	-.8 -4.5	-12 -15	mA mA	V_{SB}= 3.8V$\overline{\text{RESET}}$/ $\overline{\text{RAMPRT}}$ high V_{SB}= 3.2V

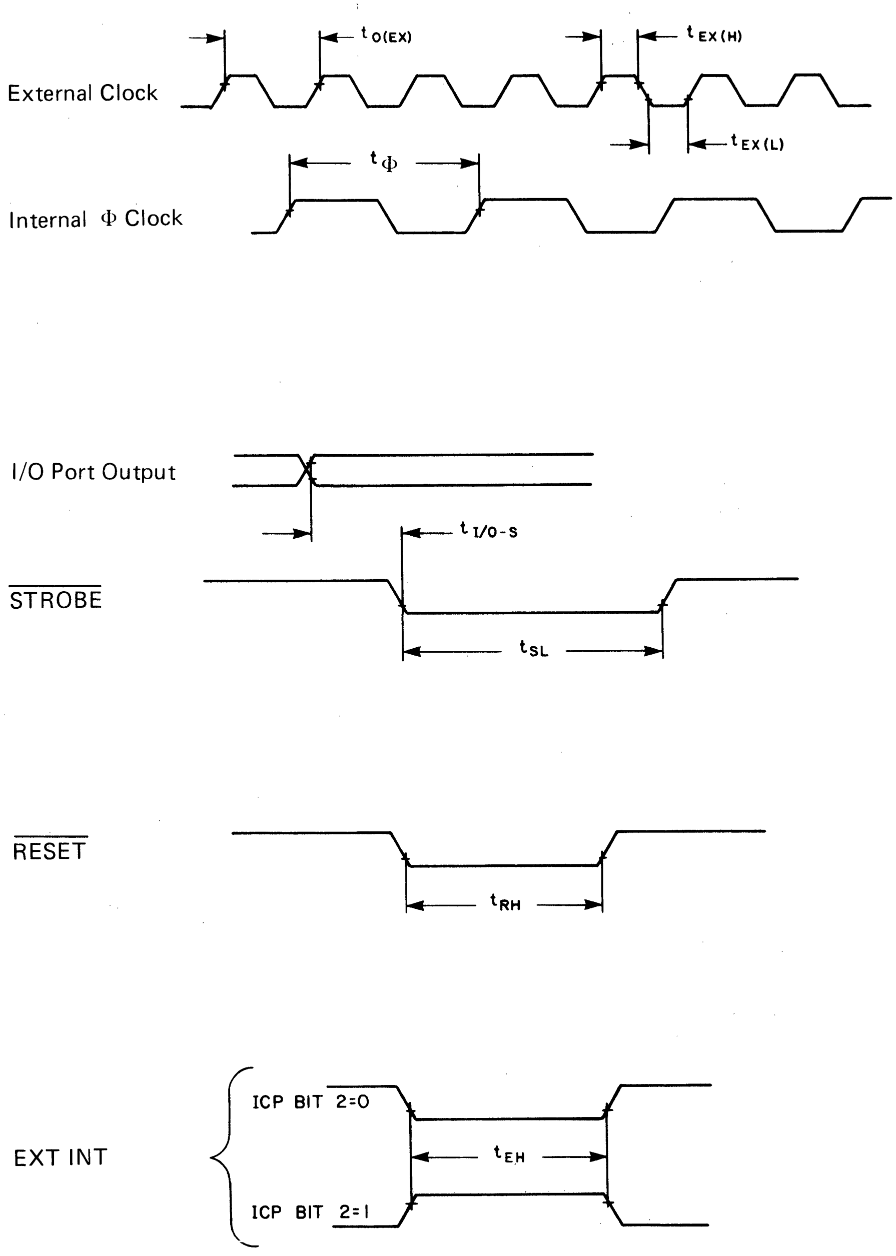
* Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

1. $\overline{\text{RESET}}$ and EXT INT have internal Schmit triggers giving minimum .2V hysteresis.
2. $\overline{\text{RESET}}$ or EXT INT programmed with standard pull-up
3. $\overline{\text{RESET}}$ or EXT INT programmed without standard pull-up
4. Power dissipation for I/O pins is calculated by$\Sigma(V_{cc} - V_{IL})\,(|I_{IL}|) + \Sigma(V_{CC} - V_{OH})\,(|I_{OH}|) + \Sigma(V_{OL})\,(I_{OL})$

3870 Family

AC TIMING DIAGRAM

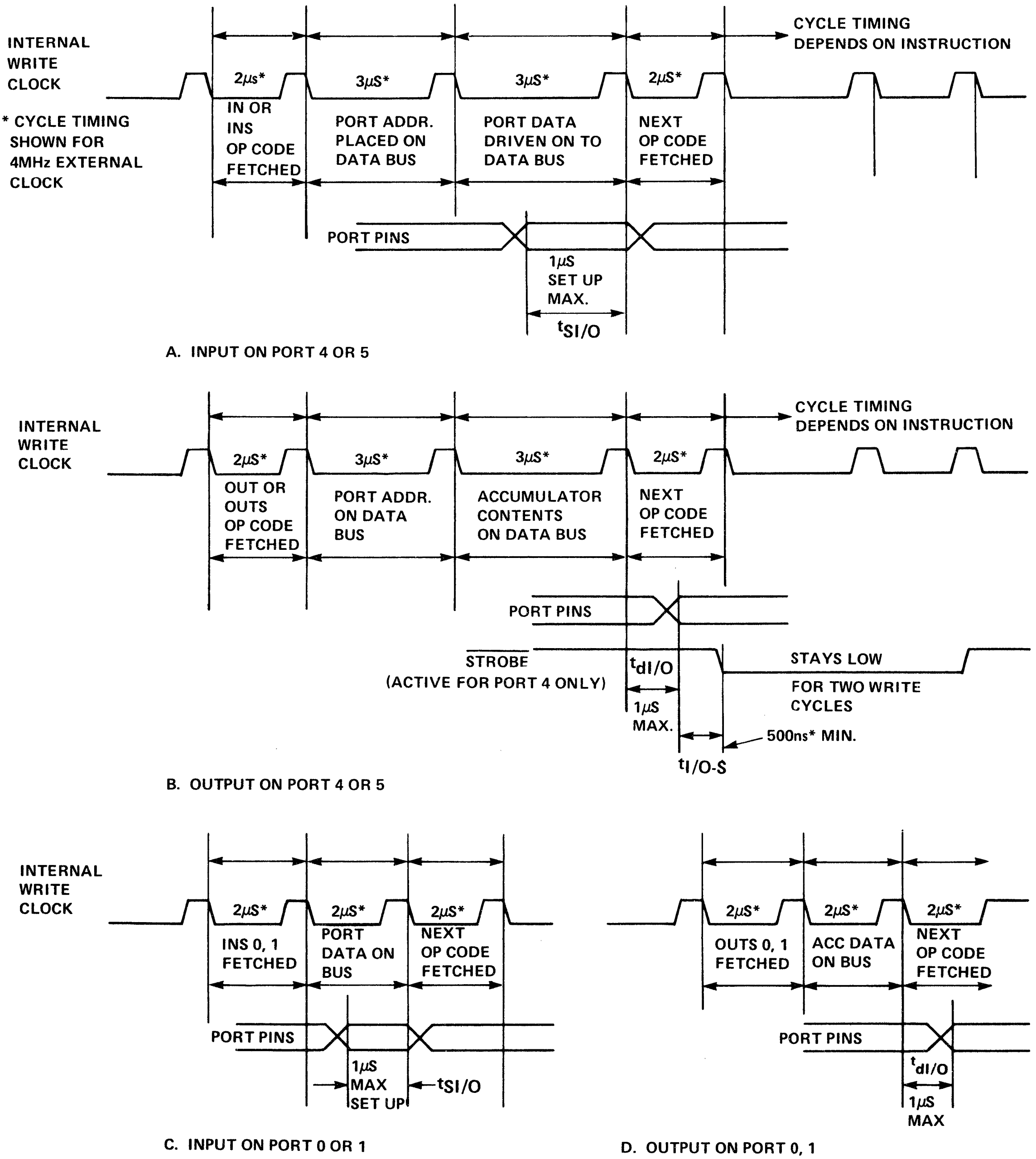
Figure 12



Note: All measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} max., or V_{OH} min.

INPUT/OUTPUT AC TIMING

Figure 13



A. INPUT ON PORT 4 OR 5

B. OUTPUT ON PORT 4 OR 5

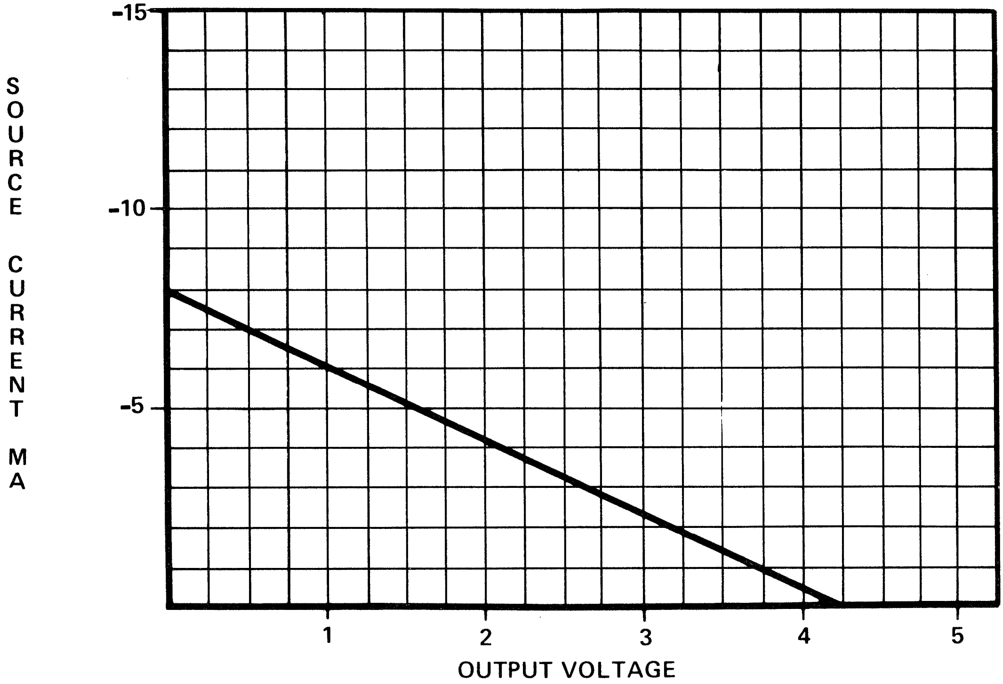
C. INPUT ON PORT 0 OR 1

D. OUTPUT ON PORT 0, 1

3870 Family

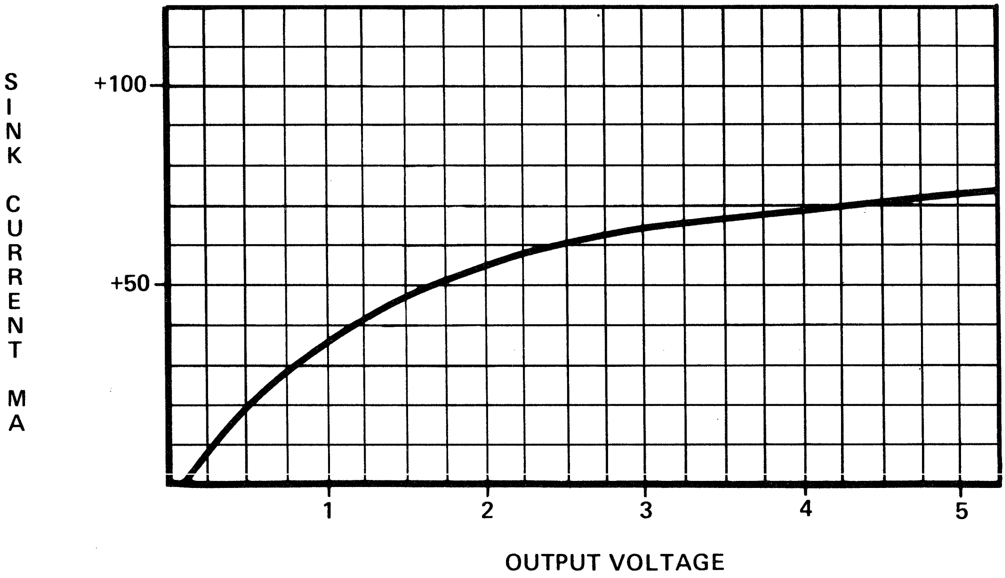
$\overline{\text{STROBE}}$ SOURCE CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 14

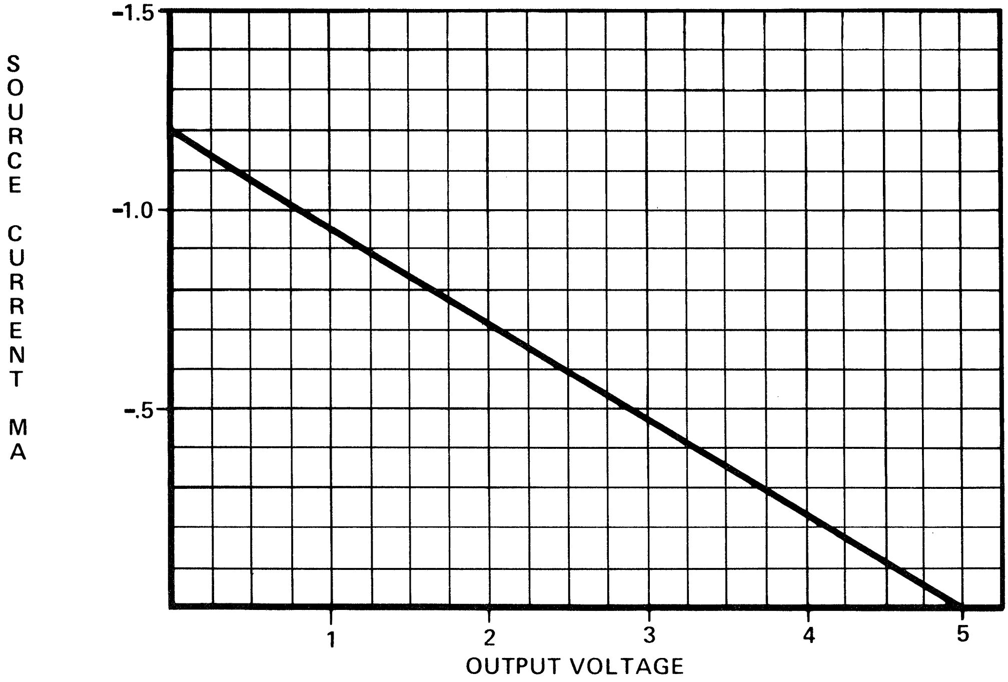


$\overline{\text{STROBE}}$ SINK CAPABILITY (TYPICAL AT V_{CC} = 5V, T_A = 25°C)

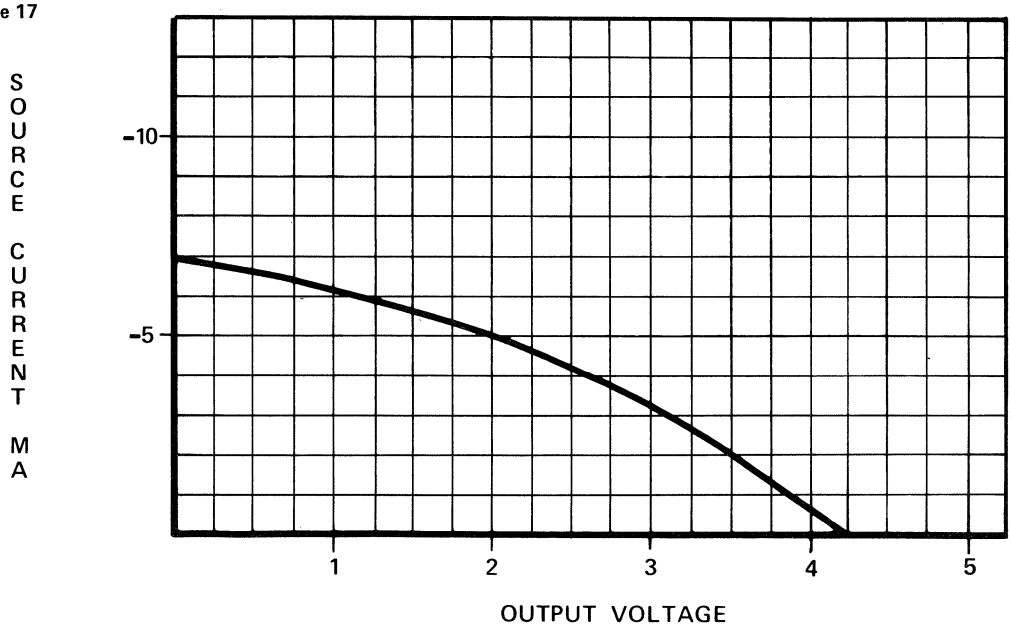
Figure 15



STANDARD I/O PORT SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)
Figure 16

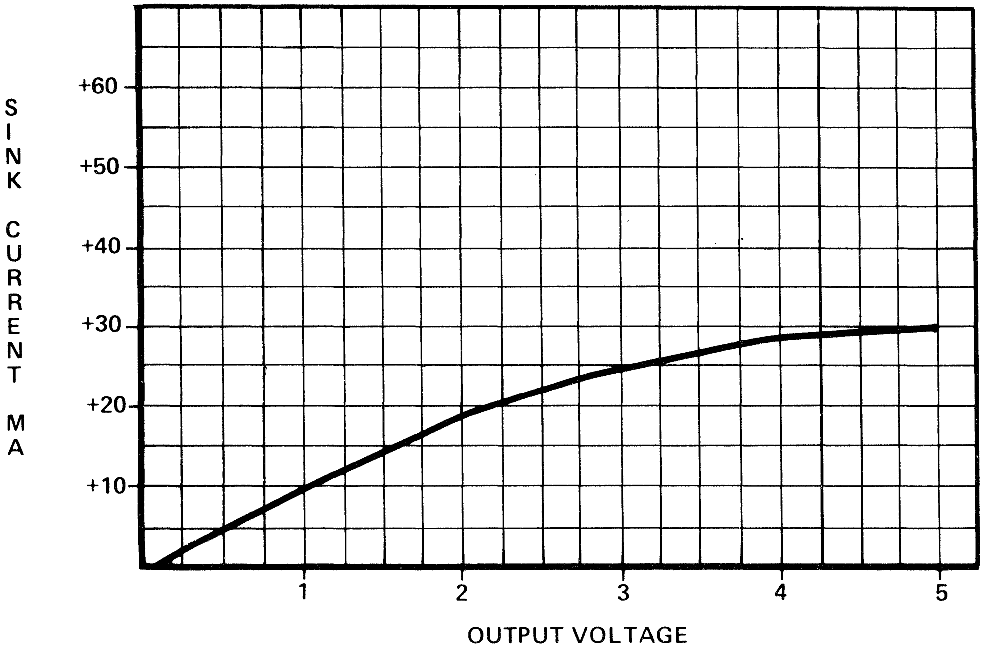


DIRECT DRIVE I/O PORT SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)
Figure 17



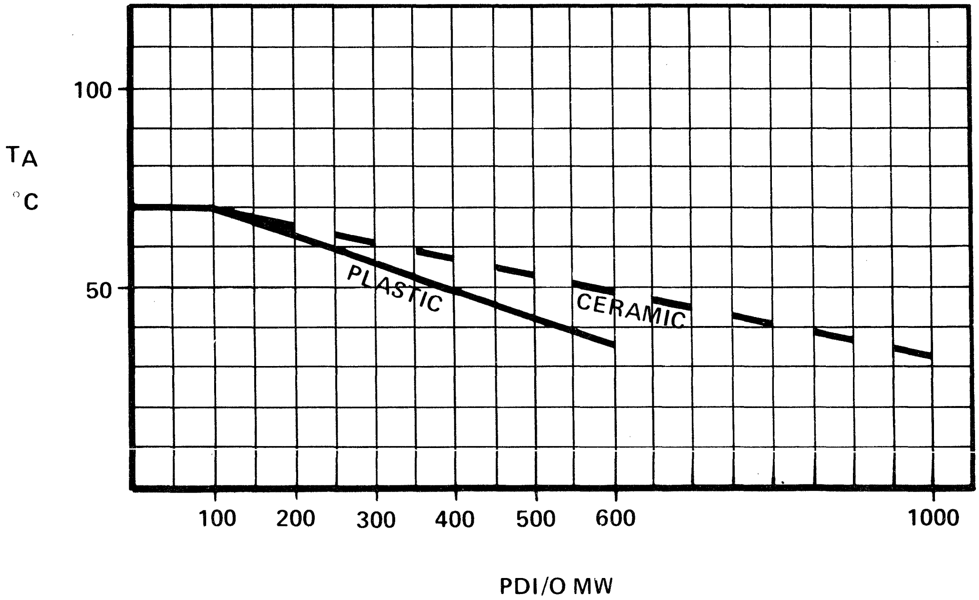
I/O PORT SINK CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 18



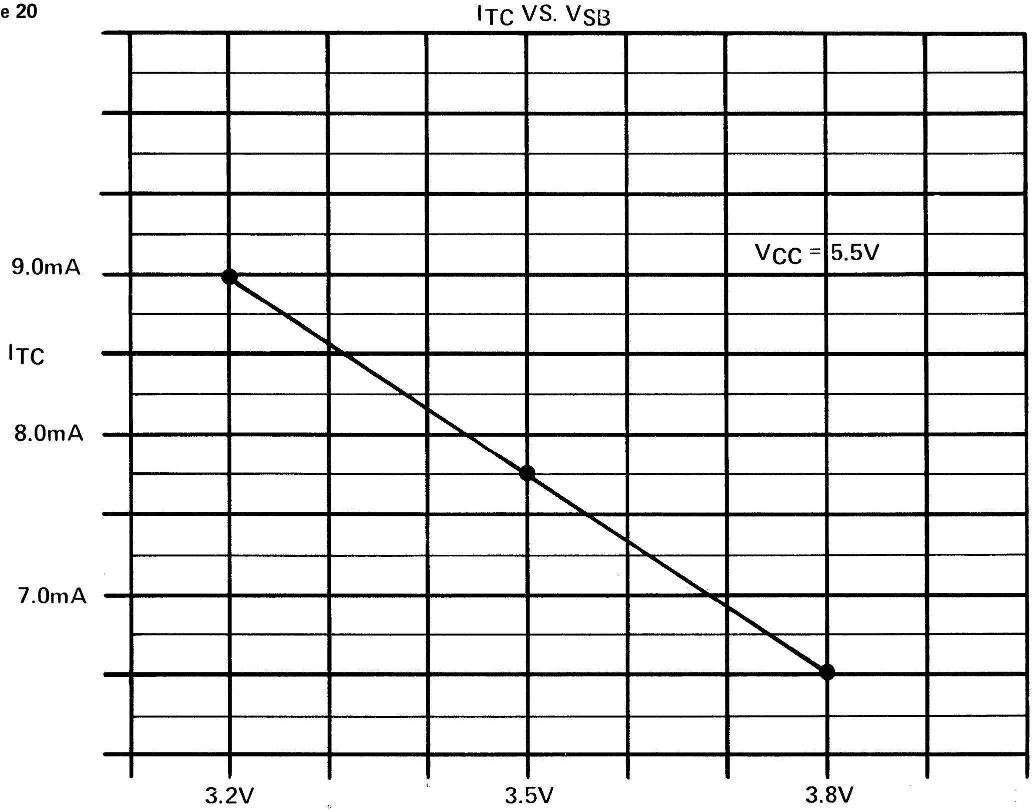
MAXIMUM OPERATING TEMPERATURE VS. I/O POWER DISIPATION

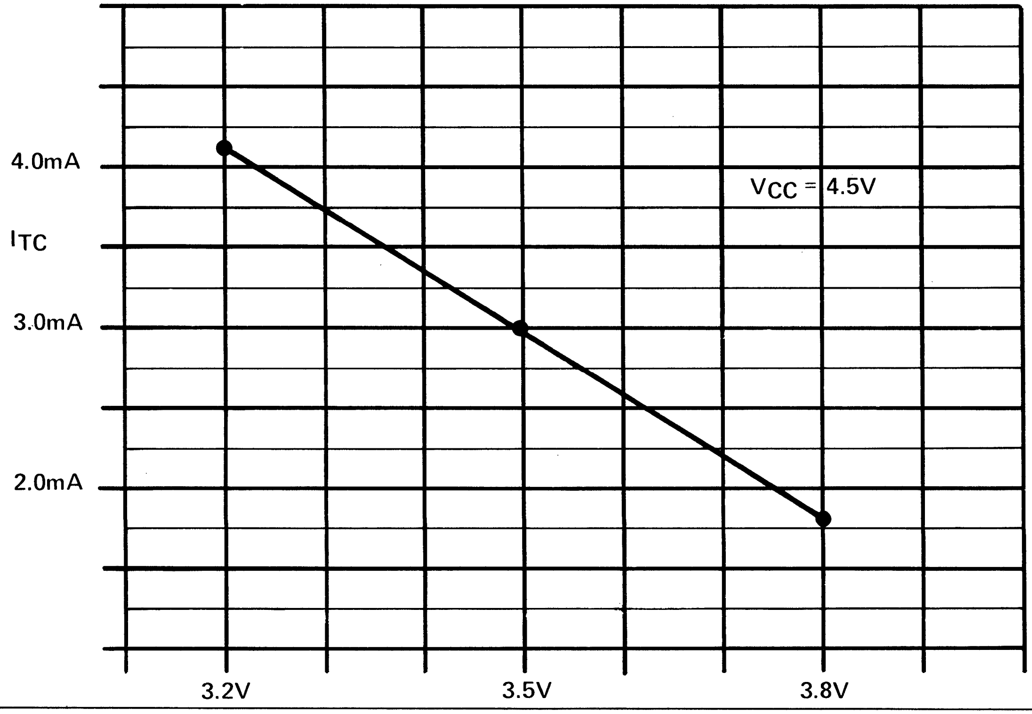
Figure 19



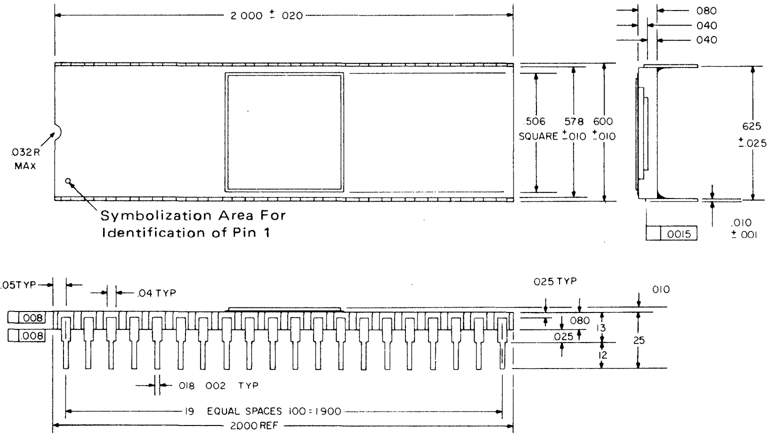
TRICKLE CHARGE CURRENT

Figure 20

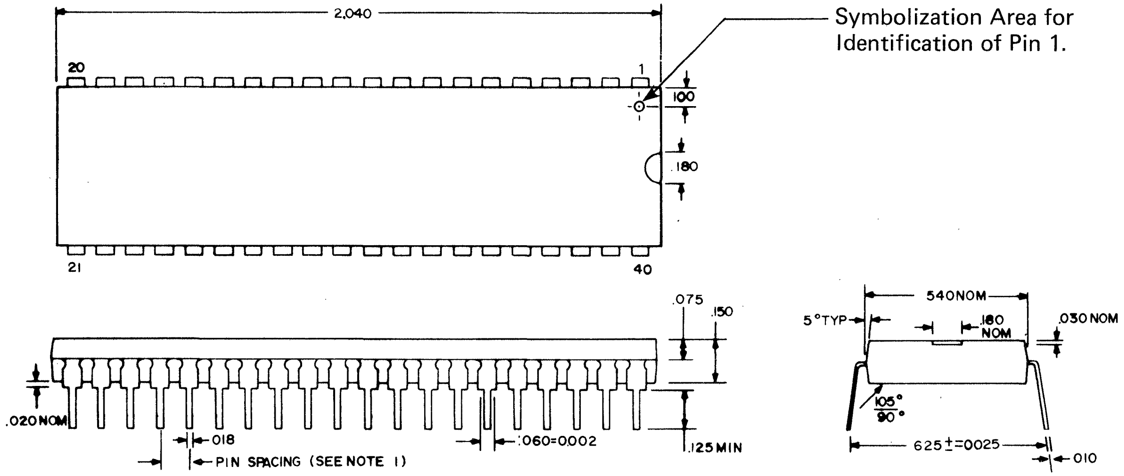




PACKAGE DESCRIPTION: 40-Pin Dual In-Line Ceramic Package



PACKAGE DESCRIPTION 40-Pin Dual-in-Line Plastic Package



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	TEMPERATURE RANGE
*MK3876(N)/16XXX	Plastic	0° C to +70° C
*MK3876(P)/16XXX	Ceramic	0° C to +70° C
+MK3876(N)/17XXX	Plastic	0° C to +70° C
+MK3876(P)/17XXX	Ceramic	0° C to +70° C

*Non Standby Device
+Standby Device

APPENDIX A

ORDERING INFORMATION

CUSTOM MK3876 OPTION SPECIFICATIONS

The custom MK3876 program may be transmitted to MOSTEK in any of the following media, listed in order of preference:

1) PROMs from the EMU-72
2) Punched paper tape
3) AID-80F Flexible Disk
4) Card Deck (IBM 80 column cards)

The program may be specified in the following forms:

PROMS with correct object code in each location

OBJECT CODE produced by one of MOSTEK's assemblers:

XFOR-50/70 Fortran IV Cross Assembler, SDB-50/70 resident assembler (ASMB-50/70), AID-80F F8 Cross-Assembler (FZCASM)

OBJECT CODE produced by the dump command from any of MOSTEK's F8 development hardware (SDB-50/70, AID-80F).

DATA DECK FORMAT as described in the Data Deck section

A completed cover letter (See page 31) must be attached. The information should be properly packed and mailed prepaid and insured to:

MOSTEK Corporation
Microcomputer Product Marketing
1215 West Crosby Road
Carrollton, Texas 75006

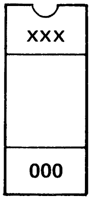
A second copy of the cover letter should be mailed separately to the above address.

PROMS

A 2716 type PROM, (5 volt only) programmed with the customer program (positive logic sense for addresses and data) may be submitted. See Fig. A-2 for marking. Include a three-letter customer ID on each PROM. After the PROM is removed from the EMU-72, it must be placed in a conductive IC carrier and securely packed.

Figure A-2

XXX = Customer ID



PAPER TAPE

Punched paper tapes (1" wide, 8 level ASCII) will be accepted. The tape must contain the absolute object output from the above mentioned F8 assemblers. Paper object tapes in absolute format generated by the "D" (dump) command of DDT-2 or the dump command of the AID-80F (F8 debug option) are also acceptable if the entire memory space is dumped continuously. Tapes may also be punched using the DATA DECK FORMAT. They must contain 80 characters per record with a CR (carriage return) and LF (line feed) separating each record. The tape must be clearly labeled with customer name, and format used. Fan fold tape is preferred. Tape transparency should be limited to 60% transmissivity (40% opaque). Specifically, thin yellow or white tape is error prone on photo-electric readers and must not be used.

FLEXIBLE DISKS

FLEXIBLE DISKS (Floppy Disks) produced on the MOSTEK AID-80F development station may be submitted. The format must be the absolute object output from the assembler or an object dump using the memory dump command (F8 Debug Option). The disk must be clearly labeled with the format of the data (object, or object dump) and the customer's name.

PUNCHED CARD DECK

Standard 80 column punched cards must be used. They must be punched in IBM 029 code. The deck must contain two types of cards:

COMMENT CARDS
DATA CARDS

3876 ORDERING INFORMATION

DATE________________ CUSTOMER PO NUMBER ________________

CUSTOMER NAME________________

ADDRESS________________

CITY __________ STATE __________ ZIP__________

COUNTRY________________

PHONE________________ EXTENSION________________

CONTACT ________________

CUSTOMER PART NUMBER________________

OPTIONS:

EXTERNAL INTERRUPT:	Pull-Up □	No Pull-Up □
RESET:	Pull-Up □	No Pull-Up □
STANDBY OPTION:	Yes □	No □

(Standby Power Option available only on the 3872 and 3876)

PORT OPTIONS:

	STANDARD TTL	OPEN DRAIN	DRIVER PULL-UP
P4-0	□	□	□
P4-1	□	□	□
P4-2	□	□	□
P4-3	□	□	□
P4-4	□	□	□
P4-5	□	□	□
P4-6	□	□	□
P4-7	□	□	□
P5-0	□	□	□
P5-1	□	□	□
P5-2	□	□	□
P5-3	□	□	□
P5-4	□	□	□
P5-5	□	□	□
P5-6	□	□	□
P5-7	□	□	□

PATTERN MEDIA

□ PROMS

(Customer can send in two extra PROM'S, MOSTEK will program the customer's code on these PROM'S for code verification in the Emulator-72.)

□ PAPER TAPE (DATA DECK)

□ PAPER TAPE (OBJECT)

□ CARD DECK (DATA DECK)

□ DISKETTE (OBJECT)

3870 Family

THESE ITEMS MAY AFFECT COST

BRANDING REQUIREMENT (If any, 10 Alpha-numeric digits allowed)

____

____

PROTOTYPE QUANTITY (10 pieces at no charge - higher quantity extra charge)

____

WAIVE PROTOTYPES (Customer accepts liability for all work in process)

Yes____________________ No ____________________

SIGNATURE__________________________________

TITLE __________________________________

COMMENT CARDS

Comment Cards must have an asterisk (*) in column 1. The remaining 79 columns may be any character. Comment Cards may be placed anywhere throughout the data deck.

DATA CARDS

These cards specify the actual ROM data. All fields are right justified.

COLUMN 1:	C (the letter C)
COLUMN 2-9:	ADDR
COLUMN 10-12:	BYTE
COLUMN 14-16:	DATA 1
COLUMN 17-19:	DATA 2
COLUMN 20-22:	DATA 3
.	
.	
.	
COLUMN 76-78:	DATA 21
COLUMN 77-79:	DATA 22 or SEQUENCE NUMBER

ADDR is the address of the first byte of data (DATA 1) contained on that card. Successive data bytes read from that card will be placed in successively greater address locations. BYTE is the number of data bytes to be read from that card (1 to 22).

If sequence numbers are used, the maximum number of bytes per card is 21. The base for ADDR and BYTE may be either decimal or hex but both must be the same. Data may be either in decimal or hex regardless of the base used for ADDR and BYTE. The base for sequence numbers (if they are used) is always decimal. The bases must be consistent throughout the deck. Data cards need not occur in order of increasing or decreasing addresses. Any unspecified address will be filled with zero. Any unpunched field will be read as a zero. If two data cards specify data for the same address, the one encountered second in the deck will override the first.

A portion of an example deck is shown.

```
  *   3876 DATA DECK
  *   MOSTEK CORP, EXAMPLE DECK
  *   ADDR/BYTE ARE IN DECIMAL
  *   DATA IS IN HEX
C  0  8  20  FF  OB  54  34  56  71  B6
C  8  8  1B  28  03  F3  4C  25  2E  94
C 16  8  04  29  01  00

  *   START OF SUBROUTINE ALPHA

C  1096   4  20  32  7C  53
C  1100   4  52  47  29  06
C  1104   1  07
```

VERIFICATION MEDIA

All original pattern media (PROMs, paper tape, etc.) are filed for contractural purposes and are not returned. Two copies of computer listings printed during the creation of the custom mask pattern are returned. One copy may be kept by the customer. The other copy should be checked thoroughly, signed, and returned to MOSTEK. The signed listing constitutes the contractual agreement for creation of the custom mask. Though the computer listing serves as the actual verification media, MOSTEK will program 2716 PROMs programmed from the data file used to create the custom mask to aid in the verification process. If programmed PROMs are desired, two blank 2716 type PROMs must be provided by the customer.

1979 MICROCOMPUTER DATA BOOK

F8
Family

MOSTEK®

F8 MICROCOMPUTER DEVICES

F8 Central Processing Unit MK 3850

August 1977

F8 Family

FEATURES

- □ N-channel Isoplanar MOS technology
- □ 2 μs cycle time
- □ 64 byte RAM on the CPU chip
- □ Two bi-directional, 8-bit I/O ports
- □ 8-bit arithmetic and logic unit, supporting both binary and decimal arithmetic
- □ Interrupt control logic
- □ Both external and crystal clock generating modes
- □ Over 70 instructions
- □ Low power dissipation—typically less than 330mW

GENERAL DESCRIPTION

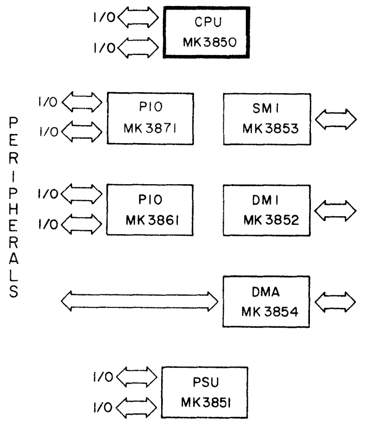
The MK3850 is the Central Processing Unit (CPU) for the F8 Microprocessor family. It is used in conjunction with other F8 family devices to configure the optimal microprocessor system for the amount of RAM, ROM/PROM, and I/O required in the users application. A minimum system may be configured with as few as two devices (CPU & PSU), while larger systems may have up to 64K bytes of memory, 128 I/O ports, direct memory acccess, and even multiple processors. Single chip microcomputer systems are also possible using the MK3870

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional (3-State)
Φ, WRITE	Clock Lines	Output
I/O 00-I/O 07	I/O Port Zero	Input/Output
I/O 10-I/O 17	I/O Port One	Input/Output
RC	RC Network Pin	Input
ROMC0-ROMC4	Control Lines	Output
EXT RES	External Reset	Input
INT REQ	Interrupt Request	Input
ICB	Interrupt Control Bit	Output
XTLX	Crystal Clock Line	Output
XTLY	External Clock Line	Input
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input

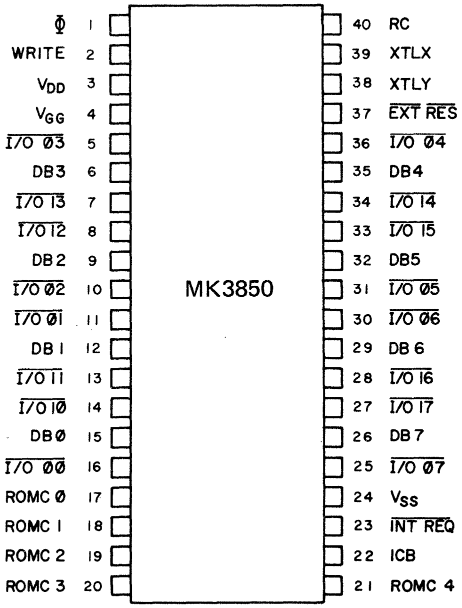
SINGLE CHIP MK3870

I/O
I/O
MK 3870
I/O
I/O

F8 FAMILY



PIN CONNECTIONS



FUNCTIONAL PIN DEFINITION

Φ and WRITE are clock outputs which drive all other devices in the F8 family.

XTLX and XTLY are used when generating the system clock in the Crystal mode. The XTLY pin is also used for operating in the External clock mode.

ROMC0 through ROMC4 are control outputs which control logic operations for other devices in the F8 family. ROMC0 through ROMC4 assume a state early in each machine cycle and hold that state for the duration of the cycle.

DB0 through DB7 are bi-directional data bus lines which link the 3850 CPU with all other F8 chips in the system. These are multiplexed lines, used to transfer data and addresses.

$\overline{\text{I/O 00}}$ through $\overline{\text{I/O 07}}$ and $\overline{\text{I/O 10}}$ through $\overline{\text{I/O 17}}$ are Input/Output port bits through which the CPU communicates with logic external to the microprocessor system.

$\overline{\text{EXT RES}}$ may be used to externally reset the system. When this line is pulled low, the program counter is set to address H '0000'.

$\overline{\text{INT REQ}}$ is used to signal the CPU that an interrupt is being requested. The 3851 PSU and 3853 SMI devices contain logic to initiate interrupt requests by pulling $\overline{\text{INT REQ}}$ low. The CPU acknowledges interrupt requests by outputting appropriate ROMC signal sequences.

$\overline{\text{ICB}}$ indicates whether or not the CPU is currently ignoring the INT REQ line. If ICB is low, the CPU will respond to interrupt requests, if $\overline{\text{ICB}}$ is high, the CPU will ignore interrupt requests.

RC is not used and should be connected to V_{SS} for normal operation.

V_{SS} = 0V
V_{DD} = +5V ± 5% @ 80mA max.
V_{GG} = +12V ± 5% @ 25mA max.

CPU ORGANIZATION

This section describes the basic functional elements of the MK3850 CPU. These elements are shown on the Functional Block Diagram of the CPU in Figure 3.

Instruction Register (IR)

The Instruction Register stores the instruction operation code during the instruction execution sequence. The OP Code is loaded into the Instruction Register from the data bus at the end of the execution sequence for the previous instruction. The last operation associated with each instruction is therefore the fetch of the OP code for the next instruction to be executed (unless an interrupt initiates the interrupt service sequence). The newly fetched OP code is latched into the Instruction Register at the start of the next machine cycle (as defined by the 1-0 transistion of the WRITE clock).

Most OP codes are either 4 or 8 bits long. For those instructions where the OP code may be completely specified using the upper 4 bits of the machine instruction, the lower 4 bits are used to specify an operand. This operand may specify a Scratch Pad Register, Port, or a 4-bit Immediate Constant. For this reason, the lower 4 bits of the instruction register are bussed to both the Scratch Pad Register Select logic and the Right Multiplexer Bus.

Control Unit

The Control Unit for the CPU consists of the Control ROM (CROM) and the State Counter. The CROM is responsible for generating all system timing and control signals required for controlling data flow within the F8 CPU and other F8 circuits.

The inputs to the CROM logic are the 8 bits from the instruction register, 4 bits from the State Counter, three internal status signals ("ALU RESULT = 0", "ISARL = 7", and the status of the Interrupt Control Bit (ICB) and two external conditions (INT REQ and Reset).

The IR inputs to the control logic identify which instruction is being executed, while the State Counter inputs define the machine cycle within the instruction execution sequence. The status of the ICB together with INT REQ are used to determine whether the interrupt sequence is to be initiated in lieu of fetching a new instruction. The reset input initiates the restart sequence. The remaining two internal signals are used to make branching decisions.

The outputs generated by the control logic fall into three groups.

- External Commands
- Next State Outputs
- Internal Commands

External commands are coded into the 5 system control lines (ROMC0 - ROMC4). Descriptions of these commands are shown in Table 2.

The next state outputs are 4 signals representing the next state of the State Counter. These signals are decoded during the present machine cycle and are strobed into the State Counter at the start of the next cycle. At that time these signals become the present state inputs to the CROM from the State Counter and new next State Outputs are generated.

The internal commands control data flow within the F8 CPU circuit. These commands include selecting the ALU operation to be performed, gating the proper input onto the Left and Right Multiplexer Busses, gating the Result Bus into the proper register or onto the Data Bus, selecting the proper Scratchpad Address input (either the ISAR or the lower 4 bits of the IR), and providing a signal to the timing circuits to force either a long or a short cycle.

Arithmetic And Logic Unit (ALU)

The 8-bit parallel ALU is the heart of the CPU. After receiving commands from the control circuits on the CPU circuit, the ALU performs the required arithmetic or logic operations (using the data presented on the two input busses) and provides the result on the Result Bus. The arithmetic operations that can be performed in the ALU are binary add, decimal adjust, add with carry, decrement, and increment. The logic operations that can be performed are "AND", "OR", "EXCLUSIVE OR", and "1's COMPLEMENT". Associated with the left input port to the ALU is a shifter, a complementer, and a low order carry (C_0). The shifter can shift the left Multiplexer Bus to the left or to the right by 1 or 4 bits. The complementer can perform the 1's complement of the left Multiplexer Bus before providing it as an input to the ALU. C_0 participates whenever the ALU performs the add with carry operation. Normally it is a zero, but may be forced to a 1 or may take the state of the carry bit in the W register. Besides providing the result on the Result Bus, the ALU also provides four signals representing the status of the result. These signals, stored in the Status (W) register, represent carry, overflow, sign and zero condition of the result of the operation. The Zero condition is also used by the control circuits during execution of the branch instructions. In addition to performing arithmetic or logic operations, the ALU sometimes acts simply as a passage way to allow the contents of the various internal registers to be placed on the Result Bus so that they may be transferred to another register. For example, when the W register is stored in the Scratchpad, it first passes unaltered through the ALU on to the Result Bus, then into the Scratchpad register.

The Accumulator

The Accumulator is the principle register for data manipulations within the CPU. Using the ALU, the 8-bit contents of the Accumulator may be complemented, incremented, or shifted left or right. Its contents may also be logically or arithmetically combined with the contents of the Scratchpad or memory locations, with the result replacing the original contents of the Accumulator.

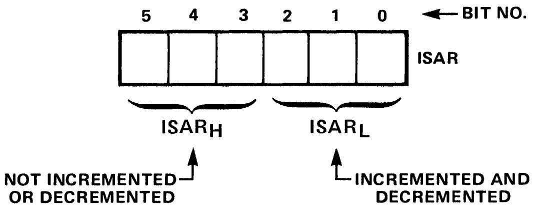
The Scratchpad And ISAR

The Scratchpad consists of 64 8-bit RAM data registers (H'00' thru H'3F') which are available to the programmer for the high speed access and manipulation of data. For most control/logic replacement this will provide all the data storage required.

All of the 64 Scratchpad registers are indirectly accessable through the use of the 6-bit Scratchpad address register, ISAR. In this way, any scratchpad register may be loaded to/from or added to the accumulator (binary or BCD); logically 'ANDED' or 'exclusive OR'ED' with the Accumulator; or decremented directly without disturbing the Accumulator. The contents of the least significant 3-bits of ISAR may be selectively auto-incremented, auto-decremented, or left unchanged (at the programmer's option) whenever the Scratchpad is accessed using ISAR (see Figure 1).

ISAR itself may be loaded either to/from the lower 6-bits of the accumulator, or loaded in 3-bit halves using the single byte immediate instructions LISU n and LISL n. The ability to independently modify the upper and lower halves of ISAR plus the auto-increment/auto-decrement options, can be used very effectively by the programmer to minimize the size of his programs.

FIGURE 1 – THE ISAR REGISTER



Additional saving may be further achieved by utilizing another key feature of the Scratchpad which permits the direct access of registers H'O' through H'B'. These registers should be reserved by the programmer for those variables most frequently accessed.

Scratchpad registers H'9' through H'F' (0 11' through O'17') have special significance since they have linkages directly with the status word (W), the Data Counter (DC), Stack Register (P) and Program Counter (P0) as shown in the F8 Programming Model (Figure 7). These linkages are implemented using single byte F8 instructions such as:

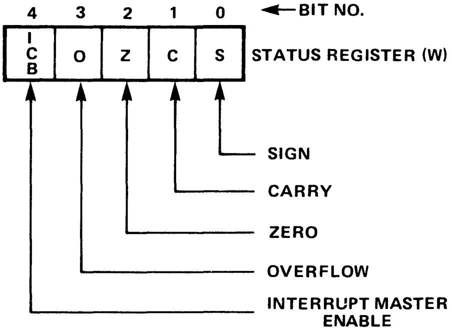
LR K,P

which transfers the 16-bit contents of the Stack Register (P) into the 'K' register pair (Scratchpad registers H'C' and H'D'). The contents of the accumulator are undisturbed by the execution by these instructions.

The Status Register

The status register (also called the W register) holds five status flags as shown in figure 2.

FIGURE 2 – THE STATUS REGISTER



Note that status flags are selectively modified following execution of different instructions. Table 4 defines the way in which individual F8 instructions modify status flags.

Sign (S BIT)

When the results of an ALU operation are being interpreted as a signed binary number, the high order bit (bit 7) represents the sign of the number.

At the conclusion of instructions that may modify the accumulator bit 7, the S bit is set to the complement of the accumulator bit 7.

Carry (C BIT)

The C bit may be visualized as an extension of an 8-bit data unit, i.e., the ninth of a 9-bit data unit. When two bytes are added, and the sum is greater than 255, then the carry out of the high order bit appears in the C bit. Here are some examples:

```
                          C  7 6 5 4 3 2 1 0 <—Bit Number
Accumulator contents:        0 1 1 0 0 1 0 1
         Value added:        0 1 1 1 0 1 1 0
                Sum:      0  1 1 0 1 1 0 1 1
There is no carry, so C is reset to 0.
                          C  7 6 5 4 3 2 1 0 <—Bit Number
Accumulator contents:        1 0 0 1 1 1 0 1
         Value added:        1 1 0 1 0 0 0 1
                Sum:      1  0 1 1 0 1 1 1 0
There is a carry, so C is set to 1.
```

Zero (Z BIT)

The Z bit is set whenever an arithmetic or logical operation generates a zero result. The Z bit is reset to 0 when an arithmetic or logical operation could have generated a zero result, but did not.

Overflow (O BIT)

When the results of an ALU operation are being interpreted as a signed binary number, since the high order bit (bit 7) represents the sign of the number, some method must be provided for indicating carries out of the highest numeric bit (bit 6). This is done using the O bit. After arithmetic operations, the O bit is set to the Exclusive-OR of carries out of bits 6 and bits 7. This simplifies signed binary arithmetic and is described in the Guide to Programming the F8. Here are some examples:

```
                        7 6 5 4 3 2 1 0 <—Bit Number
Accumulator contents:   1 0 1 1 0 0 1 1
         Value Added:   0 1 1 1 0 0 0 1
                Sum:    0 0 1 0 0 1 0 0
                  1
```

There is a carry out of bit 6 and out of bit 7, so the O bit is reset to 0 (1⊕1 = 0). The C bit is set to 1.

```
                        7 6 5 4 3 2 1 0 <—Bit Number
Accumulator contents:   0 1 1 0 0 1 1 1
         Value Added:   0 0 1 0 0 1 0 0
                Sum:    1 0 0 0 1 0 1 1
```

There is a carry out of bit 6, but no carry out of bit 7; the O bit is set to 1 (1⊕0 = 1). The C bit is reset to 0.

Interrupts (ICB BIT)

External logic can alter program execution sequence within the CPU by interrupting ongoing operations, however interrupts are allowed only when the ICB bit is set to 1.

TABLE 1 – SUMMARY OF STATUS BITS

OVERFLOW	=	$\text{CARRY}_7 \oplus \text{CARRY}_6$
ZERO	=	$\overline{\text{ALU}_7} \wedge \overline{\text{ALU}_6} \wedge \overline{\text{ALU}_5} \wedge \overline{\text{ALU}_4} \wedge \overline{\text{ALU}_3} \wedge \overline{\text{ALU}_2} \wedge \overline{\text{ALU}_1} \wedge \overline{\text{ALU}_0}$
CARRY	=	CARRY_7
SIGN	=	$\overline{\text{ALU}_7}$

External Reset

When the EXT RES (External Reset) signal is pulled low and then returned high, the Program Counter (P0) is set to 0, causing the program origined at memory location 0 to be executed. The Interrupt Control status bit is also set low, inhibiting interrupt acknowledgement. The system is locked in an idle state while EXT RES is held low.

Timing Circuit

The timing circuit generates all the timing signals for the entire microcomputer. The two primary timing signals are Φ and WRITE. The Instruction Execution Sequence for each instruction is timed with these signals. The falling edge of WRITE marks the beginning of a new machine cycle, while Φ is used to time the length of the individual machine cycles.

A machine cycle is either 4 or 6 Φ periods long, with all instructions requiring between 1 and 5 machine cycles to complete their execution sequence.

The Data Bus

The Data Bus is used for transfering all address and data information between F8 System components. This includes Port Addresses, Memory Addresses, Read/ Write Memory Data, and Input/Output Port Data. Memory Address transfers are accomplished using two successive 8 bit transfers to complete the 16-bit Memory Address. The three conditions requiring Memory Address transfers are:

1. When a three-byte instruction specifies a memory address in the second and third bytes.
2. When data is being moved between DC or P0 registers and associated scratchpad registers.
3. During the interrupt acknowledge sequence, when the interrupt vector is loaded into P0.

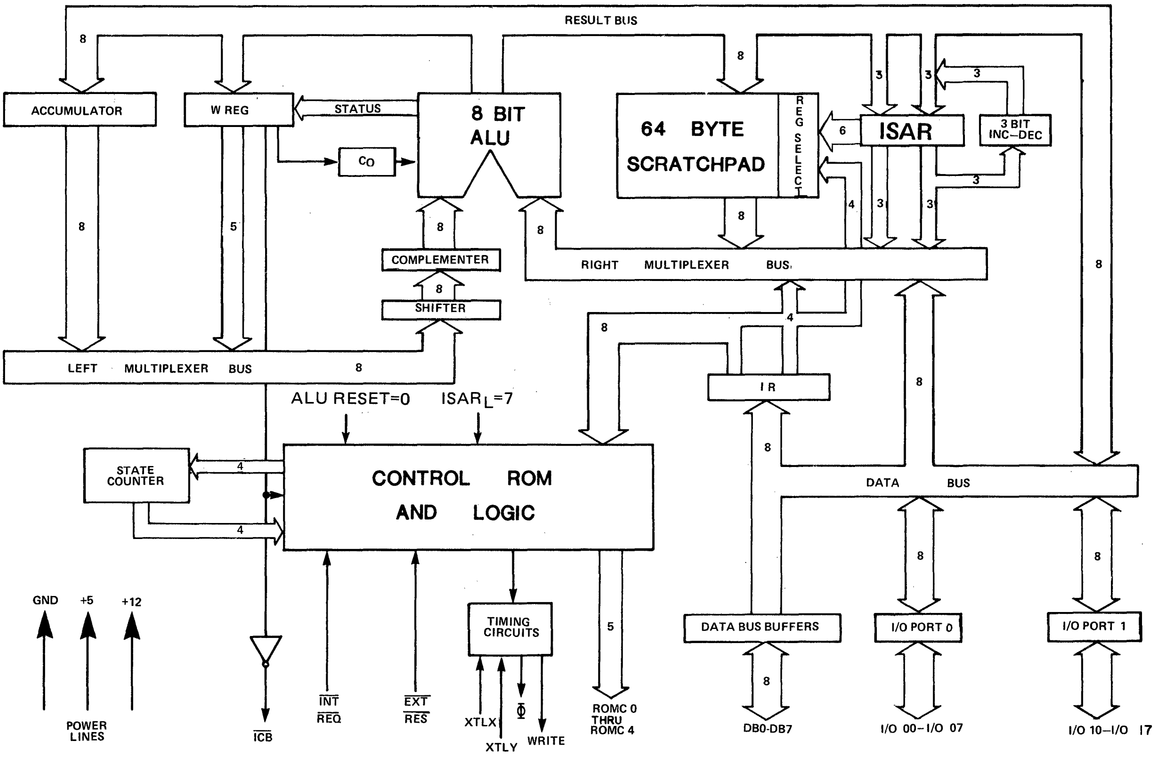
I/O Ports

The 16 address pins which most microprocessors require are used by the 3850 for two I/O ports. Data may be transferred, via these two I/O ports, between the 3850 CPU and logic external to the microprocessor system.

While other F8 devices provide additional I/O ports, the two I/O ports on the 3850 CPU execute data transfers twice as fast, since they do not use the external Data Bus.

Observe that the data path between the accumulator and the two CPU I/O ports is entirely within the 3850 CPU chip.

FIGURE 3 – MK 3850 CPU FUNCTIONAL DIAGRAM



INSTRUCTION EXECUTION SEQUENCE

All instructions are composed of long machine cycles (six Φ periods) and/or short machine cycles (four Φ periods). The long cycle is sometimes referred to as 1.5 cycles. Figure 8 illustrates the short cycle (PW_S) and the long cycle (PW_L). Observe that WRITE high appears at the end of each machine cycle.

The simplest instructions of the F8 instruction set execute in one short cycle while the most complex instruction (PI) requires two short cycles plus three long cycles. Every instruction's execution sequence ends with the next instruction OP code being fetched from memory. The OP code is loaded into the CPU's instruction register where it is decoded by the CPU's Control Unit.

The only instructions which may be executed in a single cycle are those which do not require the use of the Data Bus. This permits the Data Bus to be used to fetch the next instruction OP code simultaneously with the performance of the operation indicated by the current OP code. ROMC state 0 is used to specify the machine cycle during which a fetch is occurring, and therefore is used for all one cycle instructions.

Other instructions require more than one cycle to execute and use different ROMC states to specify the operation to be performed during each of the required cycles. The last cycle of each instruction, however, will always be the ROMC state 0 in order that the next OP code may be fetched.

The ROMC control signals are brought externally to the CPU itself in order to coordinate those operations which affect the memory referencing registers located on F8 devices other than CPU. Among these registers are the Program Counter, Stack Register and Data Counter. Most of the ROMC control states indicate those operations involving the contents of these registers, as shown in Table 2.

There are four different devices in the F8 Microprocessor family which contain the set of previously mentioned system registers (Program Counter, Stack Register, and Data Counter). These are the MK3853 SMI, MK3852 DMI, MK3851 PSU, and MK3871 PIO. Every F8 microprocessor system must contain at least one of these devices in addition to the MK3850 CPU. For those systems incorporating more than one of these devices, the resultant duplication of the Program Counter, Stack Register, and Data Counter is completely transparent to the user. This is accomplished since each device in the system receives the ROMC signals from the CPU and thus remains synchronized with all other devices.

INTERRUPTS

The Interrupt service sequence is initiated as the result of some other F8 device pulling the interrupt request (INT REQ) input to the CPU to V_{SS}. The interrupt service sequence begins during the last machine cycle of the first non-priviledged instruction to be executed after the interrupt request occurs. This is accomplished by modifying the ROMC state of the last machine cycle (which normally must be state 0 for the next OP code fetch) from state 0 to state 10 (Hex). Those instructions whose last machine cycle (ROMC state 0) is protected from being pre-empted by an interrupt request (and hence modified to ROMC state 10) are called PRIVILEGED instructions. These instructions are distinguished by the presence of an 'X' in the 'Interrupt' column of the instruction summary table (Table 4). The remainder of the interrupt service sequence requires three long and one short machine cycles as specified in Table 4. During this time, the high and low bytes of the Vector address from the interrupting device are transferred (via the Data Bus) into the Program Counter(s) and the Interrupt Control Bit (Bit 4 of the Status Register) is cleared to zero.

The response time for acknowledging an interrupt request can vary from 26 to 29 Φ periods if it is assumed that the CPU is executing a sequence of short cycle, non-privileged instructions during the time the interrupt request occurs (the minimum Φ period is 500 nS). The response time is defined as the duration from the 1-0 transition of INT REQ/ to the beginning of the execution sequence of the instruction stored at the Vector Address location in memory.

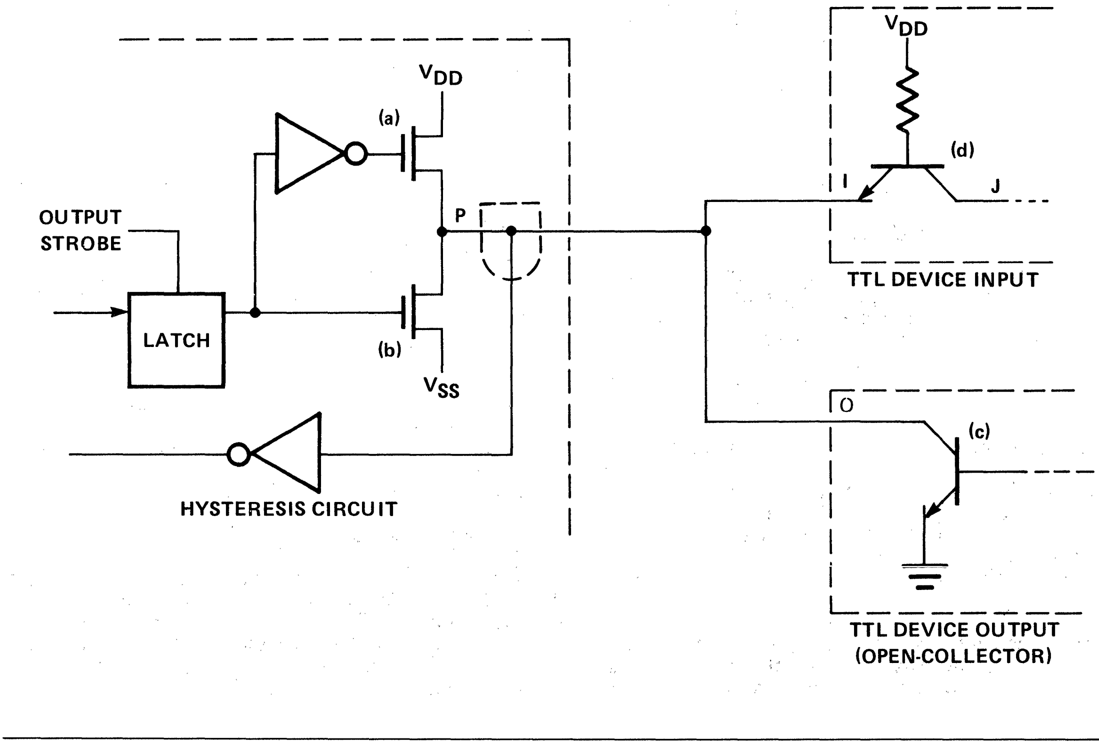
INPUT/OUTPUT INTERFACING

As illustrated in Figure 4, each I/O port pin is a "wire-AND" structure between an internal latch and any external signal. The latch is always loaded directly from the accumulator.

Each F8 I/O pin may be set high or low, under program control. If a 1 (high) is presented at the latch, then gate (b) will turn on and gate (a) will turn off, so that P will be at V_{SS} (low). If a 0 (low) is presented at the latch, then gate (a) will turn on and gate (b) will turn off, so that P will be at V_{DD} (high).

When outputting data through an I/O port, the pin can be connected directly to a TTL gate input ("TTL Device Input" in Figure 4).

FIGURE 4 – F8 I/O PORT BIT



Data is input to the pin from a "TTL Device Output" in Figure 4.

In normal operation, high or low levels at P drive the external TTL device input transistor (d). If a low level is set at P, transistor (d) conducts current through the path J, I, P, and FET (b). This is a low level to the TTL device. If the level at P is set high, transistor (d) does not conduct. This is a high level to the TTL device.

When data is input to the I/O pin, high or low levels at O drive the hysteresis circuit in the port, and result in logic 1's or 0's being transferred to the accumulator.

A port input should only be driven by devices which are incapable of sourcing more than 2 mA when pulled to V_{SS}. Ideally only open collector T^2L or open drain CMOS logic devices should be used to drive an I/O Port bit. This will prevent damage to the I/O Port output buffers should they be pulling to V_{SS} while the external device is holding the port bit to a high level through an excessively low impedance. This condition can not be avoided with software since the damage may occur when a port bit "Powers Up" to a V_{SS} level.

Since the I/O pin and the TTL device output at O are wire-ANDed, it is possible for the state of one to affect the transfer of data out from the I/O pin or in from the TTL device output. For example, if the latch in the I/O port is set so that the pin is clamped low by (b), then the level at O cannot pull P high. Conversely, if P is clamped to a low level by (c), setting the latch for a high level has no effect.

It can be seen, then, that all I/O port bits should be set for a high level, before data input, to prevent incoming logic 0's from being "masked" by logic 1's present at the port from previous outputs.

(Note: Logic 1 becomes a 0V electrical level at the I/O pin; likewise logic 0 corresponds to a high electrical level)

There are two types of programmed I/O operations that the F8 CPU may execute:

1. I/O via the two CPU ports (0 and 1),
2. I/O via ports on the other devices.

I/O operations that use the two CPU I/O ports execute in two instruction cycles. During the first cycle, the fetched instruction is decoded and data is either sent from the accumulator to the I/O latch or enabled from the I/O pin to the accumulator depending on whether the instruction is an output or an input. At the falling edge of WRITE (marking the end of the first cycle and beginning of the second cycle) the data is strobed into either the latch (OUTS) or the accumulator (INS) respectively. The second cycle is then used by the CPU for its next instruction fetch. Figure 9 indicates I/O timing.

Observe that for the data input (INS) the set-up and hold times specified are with respect to the WRITE pulse occurring at the end of the first cycle in the two cycle instruction. For output data (OUTS) the delay is specified with respect to the falling edge of WRITE marking the beginning of the second cycle in the two cycle instruction.

I/O instructions that address I/O ports with an I/O port address greater that H 'OF' occupy two bytes; the first byte specifies an IN or OUT instruction, while the second byte provides the I/O port address. Required timing at I/O port pins is given in the section of this manual that describes the device which contains the addressed I/O port.

CLOCK CIRCUITS

A unique feature of the F8 CPU is that clock logic is an integral part of the 3850 CPU chip.

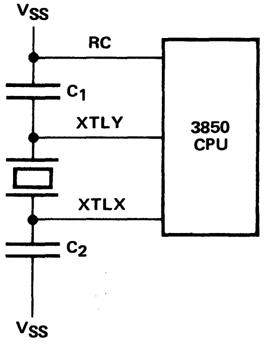
The 3850 CPU offers two alternate ways of generating a system clock; these are Crystal mode and External mode.

Crystal Mode

Figure 5 shows the pin configuration for clock generation using the crystal mode. A crystal in the 1 to 2 MHz range is placed across the XTLX and XTLY pins, along with two capacitors (C_1 and C_2), to provide a highly precise clock frequency. The external crystal (and capacitors), together with internal circuitry, combine to form a parallel resonant crystal oscillator. C_1 and C_2 capacitors should be approximately 15 pF. The characteristics of the crystal used in this mode of clock generation can be summarized as follows:

Frequency: 1 to 2 MHz, typical AT cut
Mode of Oscillation: Fundamental
Operating Temperature Range: 0°C to +70°C
Drive Level: 10 mW
Frequency Tolerance: f_O = 1 or 2 MHz
± 1000 ppm @ C_L=20pF

FIGURE 5 – CRYSTAL CONTROLLED CLOCK



External Mode

For F8 applications where synchronization with an external system clock is desired, the external clock mode may be used as shown in Figure 6. For example, a slave 3850 CPU may receive its timing from a master 3850 CPU, by having the master Φ output drive the slave XTLY input.

FIGURE 6 – EXTERNAL CLOCK

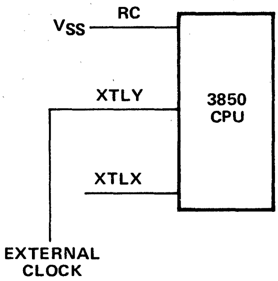


Figure 8 illustrates the AC characteristics of the clock signal needed for external mode clock generation, plus the AC characteristics of the Φ and WRITE signals generated by the CPU.

INSTRUCTION SET SUMMARY

The instruction set is summarized in Table 4. This table and the accompanying text explains the control signals and timing associated with the execution of every instruction.

The columns in Table 4 should be interpreted as follows:

OP CODE

This is the instruction mnemonic which appears in the mnemonic field of an assembly language instruction, and identifies the instruction.

OPERAND(S)

If the instruction contains any information in the operand field of the assembly language source code, the information is shown in this column. Arrows identify the portion of object code which represent the operand field. Any portion of object code that does not represent the operand field must represent the mnemonic field. Table 3 explains symbology used in the operand field.

OBJECT CODE

This is the hexadecimal representation of the instruction's object code. The first byte of object code, or in some cases the first hexadecimal digit of object code, represents the Op Code. The operand is represented by the second and third bytes of object code, if present, or in some cases by the second hexadecimal digit of the first object code byte. Table 3 explains symbology used in the object code field.

CYCLE

This column identifies each instruction cycle for every instruction. Every cycle is listed on a separate horizontal line, and is identified by the letter S for a short (4 clock period) cycle, or the letter L for a long (6 clock period) cycle. Thus the entry:

S

represents an instruction that executes in one short cycle. The entry:

S
L
S

represents an instruction that executes in three cycles; the first is a short cycle, the second is a long cycle, the third is a short cycle.

ROMC STATE

This is the state, as identified in Table 2 which is output by the 3850 CPU in the early stages of the instruction cycle.

TIMING

Timing for all instructions, except INS and OUTS accessing I/O ports 0 and 1, can be created out of Figures 12, 13 & 14. For the exceptions, Figure 9 is required. The ROMC lines are always set after a delay of td_3, as shown in Figure 12. The only timing variations for each instruction cycle are data bus timing variations. Therefore data bus timing is defined using the delays tdb_1 through tdb_6. With the exception of tdb_3, these time delays are unambiguous, in that they are keyed to either the leading edge, or to the trailing edge of WRITE high, for either a long instruction cycle, or for a short instruction cycle, as illustrated in Figure 14. There are two cases for tdb_3, however, as illustrated in Figures 12 and 13; these are identified in Table 4 as 3S for Figure 12, and 3L for Figure 13. Delays tdb_1 through tdg_6 are identified by the numbers 1 through 6.

Cycles that do not use the data bus are identified by 0 in the timing column; Figure 10 illustrates timing in this case. In summary:

0	represents Figure 10
1	represents tdb_1 in Figure 14
2	represents tdb_2 in Figure 14
3S	represents tdb_3 in Figure 12
3L	represents tdb_3 in Figure 13
4	represents tdb_4 in Figure 14
5	represents tdb_5 in Figure 14
6	represents tdb_6 in Figure 14

STATUS FLAGS

Status flags are identified as follows:

0 – Overflow
Z – Zero
C – Carry
S – Sign

Within each column, symbology is used as follows:

–	Status not effected
0	Status set to 0
I/O	Status set to either 1 or 0, depending on the results of the instruction's execution

INTERRUPT

An x in this column identifies an instruction that disallows interrupts at the end of the instruction's execution. A y identifies cycles in which the ICB bit is reset to 0 (cleared).

FUNCTION

The effect of each instruction cycle is described in this column using symbology given in Table 3.

Observe that instructions are described in Table 4 in order of ascending instruction (first byte) object code.

TABLE 2 – ROMC CONTROL STATES

ROMC (Hexadecimal)	OPERATION PERFORMED	COMMENT
00	DB ← ((P0)) ; P0←P0 +1	OP CODE, FETCH
01	DB ← ((P0)) ; P0←P0 +DB	BRANCH OFFSET FETCH
02	DB ←((DC)); DC←DC+1	
03	DB ←((P0)) ; P0←P0+1	IMMEDIATE OPERAND FETCH
04	P0 ←P	
05	((DC))←DB ; DC←DC+1	MK3851 :DC← DC+1 ONLY
06	DB←DCU	
07	DB←P U	
08	P←P0 ; DB←H'00'; P0L, P0H← DB	EXTERNAL RESET
09	DB ←DCL	
0A	DC ←DC+DB	
0B	DB ←PL	
0C	DB ←((P0)) ; DCL←DB	
0D	P←P0+1	
0E	DB ←((P0)) ; DCL←DB	
0F	P←P0 ; DB←IAL ; P0L← DB	LOWER BYTE OF ADDRESS VECTOR
10	FREEZE INTERRUPT STATUS	PREVENT ADDRESS VECTOR CONFLICTS
11	DB ←((P0)) ; DCU←DB	
12	P0L←DB ; P←P0	
13	DB←IAU; P0U←DB	UPPER BYTE OF ADDRESS VECTOR
14	P0U←DB	
15	PU←DB	
16	DCU←DB	
17	P0L ← DB	
18	PL←DB	
19	DCL←DB	
1A	((pp))←DB or ((p))←DB	
1B	DB←(pp)) or DB ←((p))	
1C	NO OPERATION	
1D	DC ⇄ DC1	MK3851 : NO OPERATION
1E	DB←P0L	
1F	DB ←P0U	

Definitions

DB	Data Bus	IA	Interrupt address vector
P0	Program Counter	L	Lower byte suffix
DC	Data Counter	U	Upper byte suffix
P	Stack Register	( )	Contents of
pp	Two hex digits (long I/O port address)	←	transfer to
p	One hex digit (short I/O port address)	⇄	exchange

TABLE 3 – SYMBOLOGY USED IN TABLES 2 and 4

SYMBOL	INTERPRETATION
( )	Contents of
A	The Accumulator contents.
a or H'a'	A single hexadecimal digit being interpreted as data.
aa or H'aa'	Two hexadecimal digits being interpreted as a single byte of data, or as the high order byte of 16 bits of data.
bb or H'bb'	Two hexadecimal digits being interpreted as the low order byte of 16 bits of data.
Binary	Binary arithmetic specified.
C	The carry status flag.
DB	F8 System Data Bus.
DC	The primary data counter register.
DCL	The low order byte of the primary data counter register.
DCU	The high order byte of the primary data counter register.
DC1	The auxiliary data counter register.
Decimal	Decimal arithmetic specified.
e or O'e'	A single octal digit being interpreted as data.
H	Scratchpad registers H'a' and H'b' contents.
ii or H'ii'	Two hexadecimal digits being interpreted as the high order byte of a 16-bit address, or as a simple byte address displacement.
IS	The six-bit scratchpad address register.
ISL	The low order three bits of ISAR.
ISU	The high order three bits of ISAR.
J	Scratchpad register H'9' contents.
jj or H'jj'	Two hexadecimal digits being interpreted as the low order byte of a 16-bit address.
K	Scratchpad registers H'c' and H'd' contents.
KL	Scratchpad register H'd' contents.
KU	Scratchpad register H'c' contents.
O	The overflow status flag.
p or H'p'	A single hexadecimal digit being interpreted as an I/O port address (short).
pp or H'pp'	Two hexadecimal digits being interpreted as an I/O port address (long).
PO	The program counter contents.
POL	The low order byte of the program counter
POU	The high order byte of the program counter
P	The stack register contents.
PL	The low order byte of the stack register
PU	The high order byte of the stack register
Q	Scratchpad registers H'e' and H'f'
QL	Scratchpad register H'f'
QU	Scratchpad register H'e'
r or H'r'	Single hexadecimal digit interpreted as scratchpad address: r = 0 through B for locations 0 through B in scratchpad. r = C or IS as address source with no change after access. r = D for IS as address source with ISL = ISL + 1 after access. r = E for IS as address source with ISL = ISL–1 after access. r = F is not allowed.

SYMBOLOGY USED IN TABLES 2 and 4 (continued)

SYMBOL	INTERPRETATION
?	Status flag has no meaning
S	The sign status flag.
t	A single hexadecimal digit identifying a status condition which will be tested by a "Branch on Condition" instruction.
W	The status register.
Z	The zero status flag.
	The logical OR of 8-bit quantities on each side of this symbol is specified.
⊕	The logical Exclusive-OR of 8-bit quantities on each side of this symbol is specified.
	The value to the right of this symbol is to be loaded into the location specified on the left of this symbol.
( )	The contents of the location within the brackets is specified.
(( ))	The contents of the memory word addressed by the contents of the location within the double brackets is specified.
+	The binary address of 8-bit quantities on each side of this symbol is specified.
←	Transfer to
⇄	Exchange

TABLE 4 – INSTRUCTIONS' EXECUTION AND TIMING

OP CODE	OPERAND(S)	OBJECT CODE	CYCLE	ROMC STATE	TIMING	STATUS FLAGS				INTERRUPT	FUNCTION
						O	Z	C	S		
LR	A, KU	00	S	0	3S	–	–	–	–		A ← (r12)
LR	A, KL	01	S	0	3S	–	–	–	–		A ← (r13)
LR	A, QU	02	S	0	3S	–	–	–	–		A ← (r14)
LR	A, QL	03	S	0	3S	–	–	–	–		A ← (r15)
LR	KU, A	04	S	0	3S	–	–	–	–		r12 ← (A)
LR	KL, A	05	S	0	3S	–	–	–	–		r13 ← (A)
LR	QU, A	06	S	0	3S	–	–	–	–		r14 ← (A)
LR	QL, A	07	S	0	3S	–	–	–	–		r15 ← (A)
LR	K, P	08	L	7	5	–	–	–	–		r12 ←(PU)
			L	B	5	–	–	–	–		r13 ←(PL)
			S	0	3S	–	–	–	–		
LR	P, K	09	L	15	2	–	–	–	–		PU ←(r12)
			L	18	2	–	–	–	–		PL ←(r13)
			S	0	3S	–	–	–	–		
LR	A, IS	0A	S	0	3S	–	–	–	–		A ←(ISAR)
LR	IS, A	0B	S	0	3S	–	–	–	–		ISAR ← (A)
PK		0C	L	12	2	–	–	–	–		P ←(P0)
											POL ←(r13)
			L	14	2	–	–	–	–		POU ←(r12)
			S	0	3S	–	–	–	–	x	
LR	P0, Q	0D	L	17	2	–	–	–	–		POL ←(r15)
			L	14	2	–	–	–	–		POU ← (r14)
			S	0	3S	–	–	–	–		

TABLE 4 – INSTRUCTIONS' EXECUTION AND TIMING (continued)

OP CODE	OPERAND(S)	OBJECT CODE	CYCLE	ROMC STATE	TIMING	STATUS FLAGS				INTERRUPT	FUNCTION
						O	Z	C	S		
LR	Q, DC	0E	L	6	5	–	–	–	–		r14 ← (DCU)
			L	9	5	–	–	–	–		r15 ← (DCL)
			S	0	3S	–	–	–	–		
LR	DC, Q	0F	L	16	2	–	–	–	–		DCU ← (R14)
			L	19	2	–	–	–	–		DCL ← (R15)
			S	0	3S	–	–	–	–		
LR	DC, H	10	L	16	2	–	–	–	–		DCU ← (R10)
			L	19	2	–	–	–	–		DCL ← (R11)
			S	0	3S	–	–	–	–		
LR	H, DC	11	L	6	5	–	–	–	–		r10 ← (DCU)
			L	9	5	–	–	–	–		r11 ← (DCL)
			S	0	3S	–	–	–	–		
SR	1	12	S	0	3S	0	1/0	0	1		Shift (A) right one bit position (zero fill)
SL	1	13	S	0	3S	0	1/0	0	1/0		Shift (A) left one bit position (zero fill)
SR	4	14	S	0	3S	0	1/0	0	1		Shift (A) right four bit positions (zero fill)
SL	4	15	S	0	3S	0	1/0	0	1/0		Shift (A) left four bit positions (zero fill)
LM		16	L	2	6	–	–	–	–		A ← ((DC))
			S	0	3S	–	–	–	–		
ST		17	L	5	1	–	–	–	–		(DC) ← (A)
			S	0	3S	–	–	–	–		
COM		18	S	0	3S	0	1/0	0	1/0		A ← (A) ⊕ H'FF' Complement accumulator
LNK		19	S	0	3S	1/0	1/0	1/0	1/0		A ← (A) + (C)
DI		1A	S	1C	0	–	–	–	–	y	Clear ICB.
			S	0	3S	–	–	–	–		
EI		1B	S	1C	0	–	–	–	–		Set ICB
			S	0	3S	–	–	–	–	x	
POP		1C	S	4	0	–	–	–	–		PO ← (P)
			S	0	3S	–	–	–	–	x	
LR	W, J	1D	S	1C	0	1/0	1/0	1/0	1/0		W ← (r9)
			S	0	3S	–	–	–	–	x	
LR	J, W	1E	S	0	3S	–	–	–	–		r9 ← (W)
INC		1F	S	0	3S	1/0	1/0	1/0	1/0		A ← (A) + 1
LI	aa	20	L	3	6	–	–	–	–		A ← H'aa'
	└	→ aa	S	0	3S	–	–	–	–		
NI	aa	21	L	3	4	0	1/0	0	1/0		A ← (A) ∧ H'aa'
	└	→ aa	S	0	3S	–	–	–	–		
OI	aa	22	L	3	4	0	1/0	0	1/0		A ← (A) ∨ H'aa'
	└	→ aa	S	0	3S	–	–	–	–		
XI	aa	23	L	3	4	0	1/0	0	1/0		A ← (A) ⊕ H'aa'
	└	→ aa	S	0	3S	–	–	–	–		
AI	aa	24	L	3	4	1/0	1/0	1/0	1/0		A ← (A) + H'aa'
	└	→ aa	S	0	3S	–	–	–	–		

TABLE 4 – INSTRUCTIONS' EXECUTION AND TIMING (continued)

OP CODE	OPERAND(S)	OBJECT CODE	CYCLE	ROMC STATE	TIMING	STATUS FLAGS				INTERRUPT	FUNCTION
						O	Z	C	S		
CI	aa	25	L	3	4	–	–	–	–		Perform H'aa' + $(\bar{A})$ + 1. Do not save result, but modify status flags to reflect result.
		aa	S	0	3S	1/0	1/0	1/0	1/0		
IN	PP	26	L	3	2	–	–	–	–		DB ← PP; P0 ← P0+1
		PP	L	1B	6	0	1/0	0	1/0		A ← (I/O Port PP)
			S	0	3S	–	–	–	–		
OUT	PP	27	L	3	2	–	–	–	–		DB ← PP
		pp	L	1A	1	–	–	–	–		I/O Port PP ← (A)
			S	0	3S	–	–	–	–	x	
PI	iijj	28	L	3	6	–	–	–	–		A ← H'jj'
		ii	S	D	0	–	–	–	–		P ← (P0) + 1
		jj	L	C	2	–	–	–	–		P0L ← H'jj'
			L	14	1	–	–	–	–		P0U ← (A)
			S	0	3S	–	–	–	–	x	
JMP	iijj	29	L	3	6	–	–	–	–		A ← H'ii'
		ii	L	C	2	–	–	–	–		P0L ← H'jj'
		jj	L	14	1	–	–	–	–		P0U ← (A)
			S	0	3S	–	–	–	–	x	
DCI	iijj	2A	L	11	2	–	–	–	–		DCU ← ii
		ii	S	3	0	–	–	–	–		(increment P0)
		jj	L	E	2	–	–	–	–		DCL ← jj
			S	3	0	–	–	–	–		(increment P0)
			S	0	3S	–	–	–	–		
NOP		2B	S	0	3S	–	–	–	–		P0 ← (P0) + 1
XDC		2C	S	1D	0	–	–	–	–		DC0 ⇆ DC_1
			S	0	3S	–	–	–	–		
DS	r	3r	L	0	3L	1/0	1/0	1/0	1/0		r ← (r) + H'FF' Decrement scratchpad byte
LR	A, r	4r	S	0	3S	–	–	–	–		A ← (r)
LR	r, A	5r	S	0	3S	–	–	–	–		r ← (A)
LISU	e	6e	S	0	3S	–	–	–	–		ISARU ← 0'e'
LISL	e	68 + e	S	0	3S	–	–	–	–		ISARL ← 0'e'
LIS	a	7a	S	0	3S	–	–	–	–		A ← H'0a'
BT	e, ii	8e	S	1C	0	–	–	–	–		Test e ∧ W. register
		ii	S	3	0	–	–	–	–		Res = 0 so P0 = (P0) + 2
			S	0	3S	–	–	–	–		
			S	1C	0	–	–	–	–		Test e ∧ W. register
			L	1	2	–	–	–	–		Res ≠ 0 so P0 = (P0) + H'ii' + 1
			S	0	3S	–	–	–	–		
AM		88	L	2	4	1/0	1/0	1/0	1/0		A ← (A) + ((DC)) Binary, DC ← (DC) + 1
			S	0	3S						
AMD		89	L	2	4	?	?	1/0	?		A ← (A)+((DC)) Decimal; DC ← (DC) + .1
			S	0	3S						

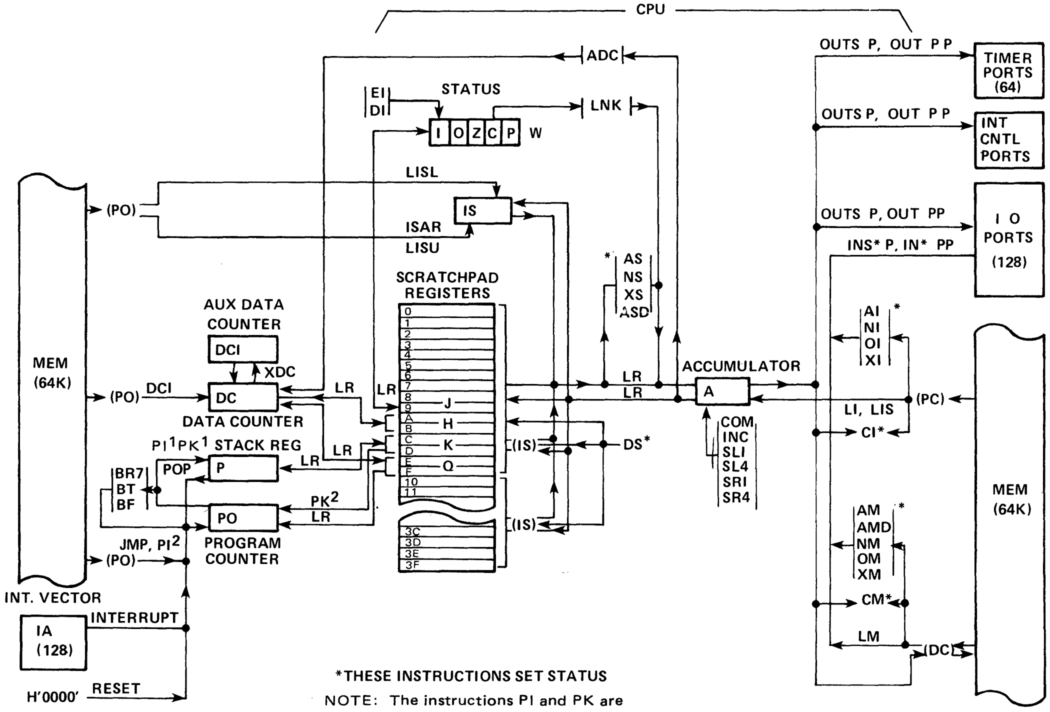
TABLE 4 – INSTRUCTIONS' EXECUTION AND TIMING (continued)

OP CODE	OPERAND(S)	OBJECT CODE	CYCLE	ROMC STATE	TIMING	STATUS FLAGS				INTERRUPT	FUNCTION
						O	Z	C	S		
NM		8A	L	2	4	0	1/0	0	1/0		A←(A) ∧ ((DC));
			S	0	3S						DC←(DC) + 1
OM		8B	L	2	4	0	1/0	0	1/0		A←(A) V ((DC)),
			S	0	3S						DC←(DC) + 1
XM		8C	L	2	4	0	1/0	0	1/0		A←(A) ⊕ ((DC));
			S	0	3S						DC←(DC) + 1
CM		8D	L	2	4	1/0	1/0	1/0	1/0		Set status flags on basis
			S	0	3S						of ((DC)) + (A) + 1;
											DC←(DC) + 1
ADC		8E	L	A	1	–	–	–	–		DC←(DC) + A
			S	0	3S	–	–	–	–		
BR7	ii	8F	S	3	0	–	–	–	–		P0←(P0) + 2
		ii	S	0	3S	–	–	–	–		because (ISARL) = 7
			L	1	2	–	–	–	–		P0←(P0) + H'ii' + 1
			S	0	3S	–	–	–	–		because (ISARL) ≠ 7
BF	t, ii	9t	S	1C	0	–	–	–	–		Test t ∧ W. register
		ii	L	1	2	–	–	–	–		Res ≠ 0 so P0 = (P0)
			S	0	3S	–	–	–	–		+ H'ii' + 1
			S	1C	0	–	–	–	–		Test t ∧ W. register
			S	3	0	–	–	–	–		Res ≠ 0 so P0 = (P0)
			S	0	3S	–	–	–	–		+ 2
INS	0 or 1	A0, A1	S	1C	0	0	1/0	0	1/0		A ← (I/O Port 0 or 1)
			S	0	3S	–	–	–	–		
INS	2	A2	L	1C	0	0	1/0	0	1/0		DB ← Port address (2
	through	through	L	1B	6	–	–	–	–		through 15)
	15	AF	S	0	3S	–	–	–	–		A ← (Port 2 through 15)
OUTS	0 or 1	B0, B1	S	1C	0	–	–	–	–		I/O Port 0 or 1 ← (A)
			S	0	3S	–	–	–	–		
OUTS	2	B2	L	1C	0	–	–	–	–		DB ← Port address (2
	through	through	L	1A	1	–	–	–	–		through 15)
	15	BF	S	0	3S	–	–	–	–	x	Port (2 through 15) ←
											(A)
AS	r	Cr	S	0	3S	1/0	1/0	1/0	1/0		A ← (A) + (r) Binary
ASD	r	Dr	S	1C	0	?	?	1/0	?		A ← (A) + (r) Decimal
			S	0	3S	–	–	–	–		
XS	r	Er	S	0	3S	0	1/0	0	1/0		A ← (A) ⊕ (r)
NS	r	Fr	S	0	3S	0	1/0	0	1/0		A ← (A) ∧ (r)
INTRPT		xx	L	1C	0	–	–	–	–		IDLE
			L	0F	2	–	–	–	–		P0L ← Int. address
											(lower byte); PC1←P0
			L	13	2	–	–	–	–	y	P0U ← Int. address
											(upper byte)
			S	0	3S	–	–	–	–	x	
RESET		xx	S	1C	0	–	–	–	–		IDLE
			L	8	1	–	–	–	–	y	P←P0, P0←0
			S	0	3S	–	–	–	–	x	

PROGRAMMING MODEL

Figure 7 shows a Programming Model of the F8 Microcomputer system. This diagram is intended to depict the various data transfers and manipulations which are facilitated by the instruction set of the F8. Every F8*system configuration will contain the basic functional elements shown in this diagram, with the exception of the Auxiliary Data Counter (DC1). The Auxiliary Data Counter is available only in those systems incorporating the MK3852 Dynamic Memory Interface, the MK3853 Static Memory Interface, or the MK3870 single chip F8 Microcomputer.

FIGURE 7 – F8 PROGRAMMING MODEL



*THESE INSTRUCTIONS SET STATUS

NOTE: The instructions PI and PK are shown in two sequential parts (PI^1, PI^2 and PK^1, PK^2).

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS (Above which useful life may be impaired)

V_{GG}	+15V to –0.3V
V_{DD}	+7V to –0.3V
RC, XTLX, and XTLY	+15V to –0.3V (RC with 5KΩ series resistor)
All other inputs	+7V to –0.3V
Storage temperature	–55°C to +150°C
Operating temperature	0°C to +70°C

NOTE: All voltages with respect to V_{SS}.

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		30	80	mA	f = 2 MHz, Outputs unloaded f - 2 MHz
I_{GG}	V_{GG} Current		15	25	mA	Outputs unloaded

TABLE 5 – AC CHARACTERISTICS

(V_{SS} = 0V, V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0°C to +70°C

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
P_x*	External Input Period	0.5		10	μs	
PW_x*	External Pulse Width	200		P_x–200	ns	t_r, $t_f \leqslant$ 30 nS
tx_1	Ext. to Φ – to – Delay	20		110	ns	
tx_2	Ext. to Φ + to + Delay	20		125	μs	
PΦ	Φ Period	0.5		10	μs	
PW_1	Φ Pulse Width	180		PΦ–180	ns	t_r, t_f = 50 nS; C_L= 100 pF
td_1	Φ to WRITE + Delay	60	150	250	ns	C_L = 100 pF
td_2	Φ to WRITE – Delay	60	150	225	ns	C_L = 100 pF
PW_2	WRITE Pulse Width	PΦ–100		PΦ	ns	t_r, t_f = 50 nS typ; C_L = 100 pF
PW_S	WRITE Period; Short		4PΦ			
PW_L	WR'TE Period; Long		6PΦ			
td_3	WRITE to ROMC Delay	80	300	550	ns	C_L = 100 pF
td_4*	WRITE to $\overline{\text{ICB}}$ Delay			410	ns	C_L = 50 pF
td_5	WRITE to $\overline{\text{INT REQ}}$ – Delay			430	ns	C_L = 100 pF
td_6	WRITE to $\overline{\text{INT REQ}}$ + Delay			1.65	μs	C_L = 100 pF
t_{sx}*	$\overline{\text{EXT RES}}$ set-up time	1.0			μs	C_L = 20 pF
t_{su}*	I/O set-up time	300			ns	
t_h*	I/O hold time	50			ns	
t_o*	I/O Output Delay			1.5	μs	C_L = 50 pF
tdb_0*	WRITE to data bus High Impedance		250	500	ns	
tdb_1*	WRITE to Data Bus Stable		0.6	1.3	μs	C_L = 100 pF
tdb_2	WRITE to Data Bus Stable	2PΦ		2PΦ+1.0	μs	C_L = 100 pF
tdb_3*	Data Bus Set-up	200			ns	
tdb_4*	Data Bus Set-up	300			ns	
tdb_5	Data Bus Set-up	500			ns	
tdb_6*	Data Bus Set-up	300			ns	

*The parameters which are starred in the table above represent those which are most frequently of importance when interfacing to an F8 system. These encompass I/O timing, external timing generation and possible external RAM timing. The remaining parameters are typically those that are only relevant between F8 chips and not normally of concern to the user.

Input and output capacitance is 3 to 5 pF typical on all pins except V_{DD}, V_{GG}, and V_{SS}.

TABLE 6 – DC CHARACTERISTICS
(V_{SS} = 0V, V_{DD} = +5V ± 5%; V_{GG} = +12V ± 5%)

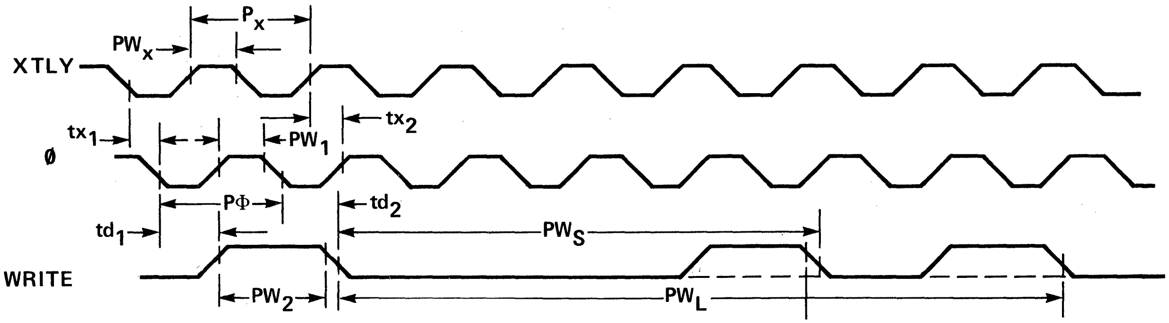
SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
Φ, WRITE	V_{OH}	Output High Voltage	4.4	V_{DD}	Volts	I_{OH} = –10 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	V_{OH}	Output High Voltage	2.9		Volts	I_{OH} = –100 μA
XTLY	V_{IH}	Input High Voltage	4.5	V_{GG}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IH}	Input High Current	5	50	μA	V_{IN} = V_{DD}
	I_{IL}	Input Low Current	–10	–80	μA	V_{IN} = V_{SS}
ROMC0	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
ROMC4						
DB0	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
DB7	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	I_{IH}	Input High Current		1	μA	V_{IN} = 7V 3-State mode
	I_{IL}	Input Low Current		–1	μA	V_{IN} = V_{SS}, 3-State mode
I/O 0	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –30 μA
	V_{OH}	Output High Voltage	2.9	V_{DD}	Volts	I_{OH} = –100 μA
I/O 17	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	V_{IH}	Input High Voltage (1)	2.9	V_{DD}	Volts	Internal pull-up to V_{DD}
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		1	μA	V_{IN} = V_{DD}
	I_{IL}	Input Low Current		–1.6	mA	V_{IN} = 0.4V (2)
$\overline{\text{EXT RES}}$	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	Internal pull-up to V_{DD}
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IL}	Input Low Current		–1.0	mA	V_{IN} = V_{SS}
$\overline{\text{INT REQ}}$	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	Internal pull-up to V_{DD}
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IL}	Input Low Current		–1.0	mA	V_{IN} = V_{SS}
$\overline{\text{ICB}}$	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 100μ A

(1) Hysteresis input circuit provides additional 0.3V noise immunity while internal pull-up provides TTL compatability.

(2) Measured while F8 port is outputting a high level.

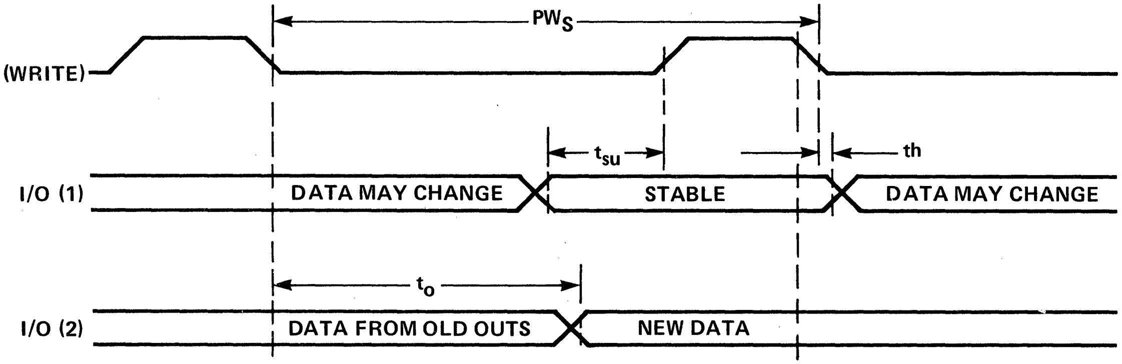
NOTE: Positive current is defined as conventional current flowing into the pin referenced.

FIGURE 8 – TIMING SIGNAL SPECIFICATIONS



Parameters are described in Table 5

FIGURE 9 – TIMING FOR DATA INPUT OR OUTPUT AT I/O PORT PINS

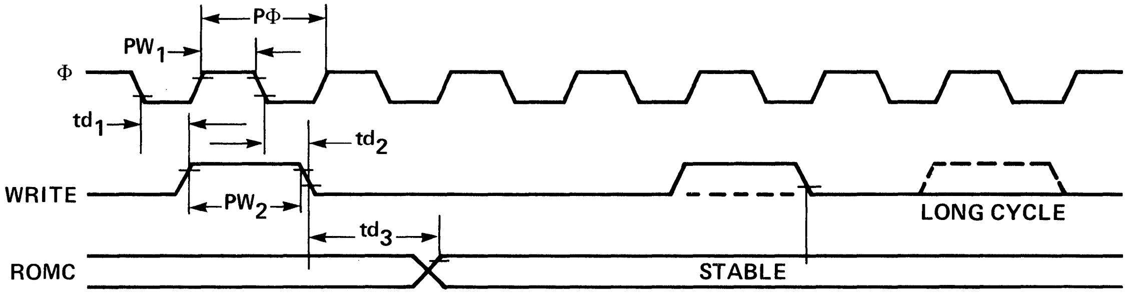


(1) This represents the timing for data at the I/O pin during the execution of the INS instruction, i.e., the CPU is inputting.

(2) This represents the timing for data being output by the CPU at the I/O pin.

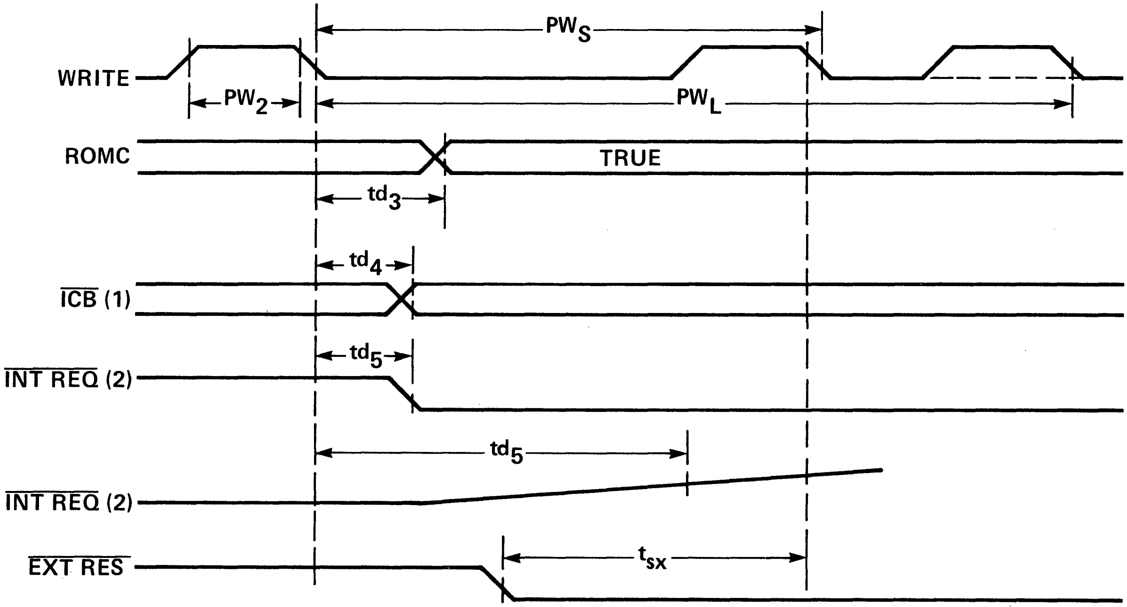
Symbols are defined in Table 5

FIGURE 10 – ROMC SIGNALS OUTPUT BY 3850 CPU



Symbols are defined in Table 5

FIGURE 11 – INTERRUPT SIGNALS TIMING

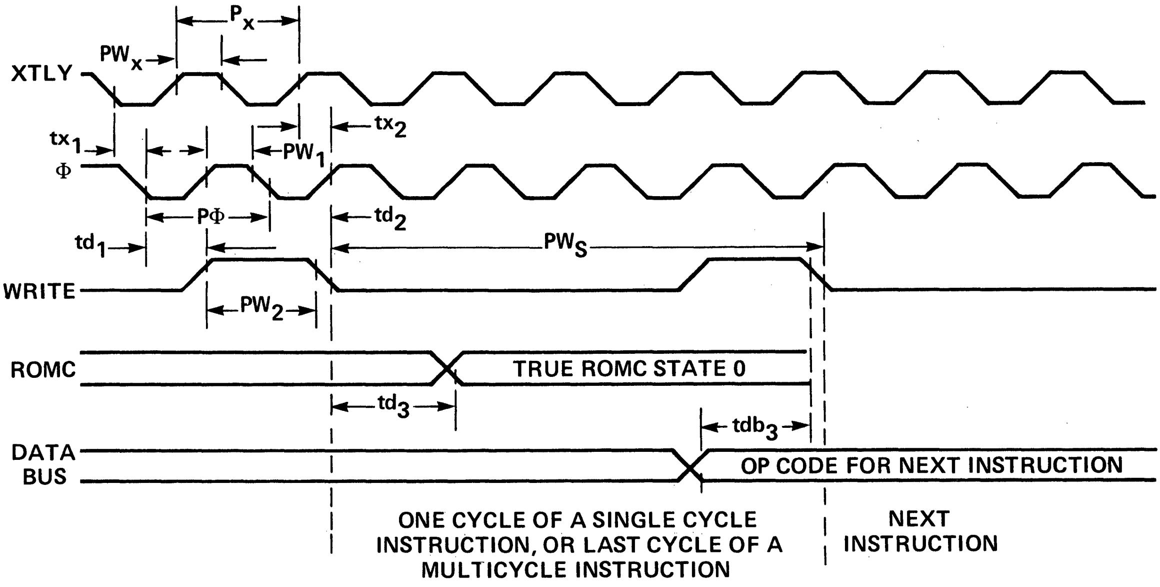


(1) ICB will go from a 1 to a 0 following the execution of the EI instruction and will go from a 0 to 1 following either the execution of the DI instruction or the CPU's acknowledgement of an interrupt.

(2) This is an input ot the CPU chip and is generated by a PSU or 3853 MI chip. The open drain outputs of these chips are all wire "ANDed" together on this line with the pull-up being located on the CPU chip. For a 0 to 1 transition the delay is measured to 2.0V.

Symbols are defined in Table 5

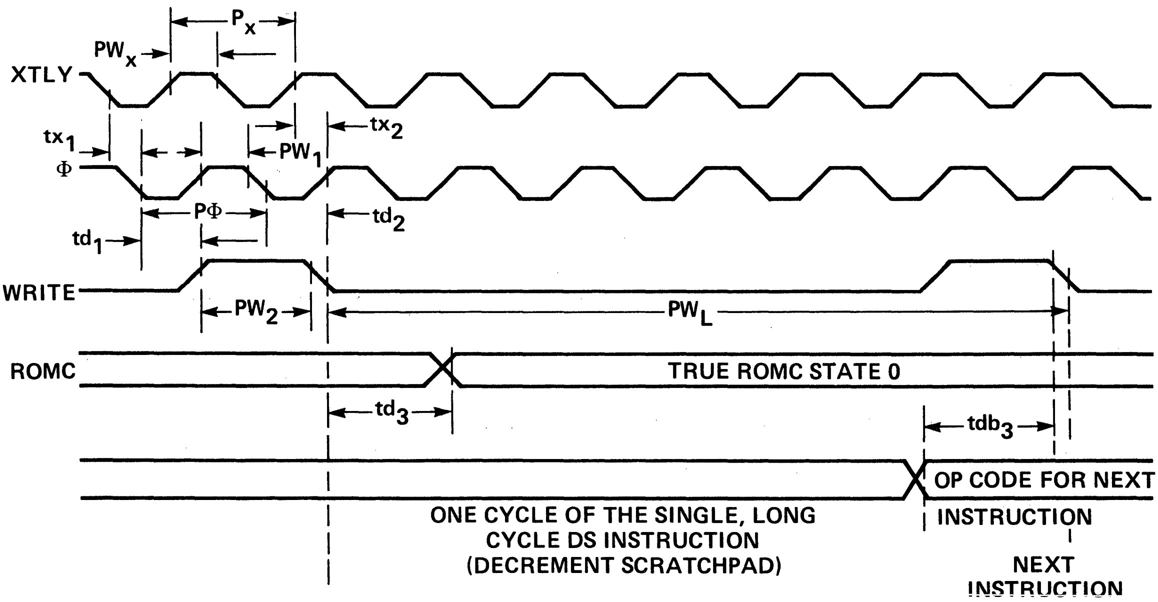
FIGURE 12 – A SHORT CYCLE INSTRUCTION FETCH



P_X
PW_X
XTLY
tx_1
tx_2
PW_1
Φ
PΦ
td_1
td_2
PW_S
WRITE
PW_2
ROMC
TRUE ROMC STATE 0
td_3
tdb_3
DATA BUS
OP CODE FOR NEXT INSTRUCTION
ONE CYCLE OF A SINGLE CYCLE INSTRUCTION, OR LAST CYCLE OF A MULTICYCLE INSTRUCTION
NEXT INSTRUCTION

Symbols are defined in Table 5

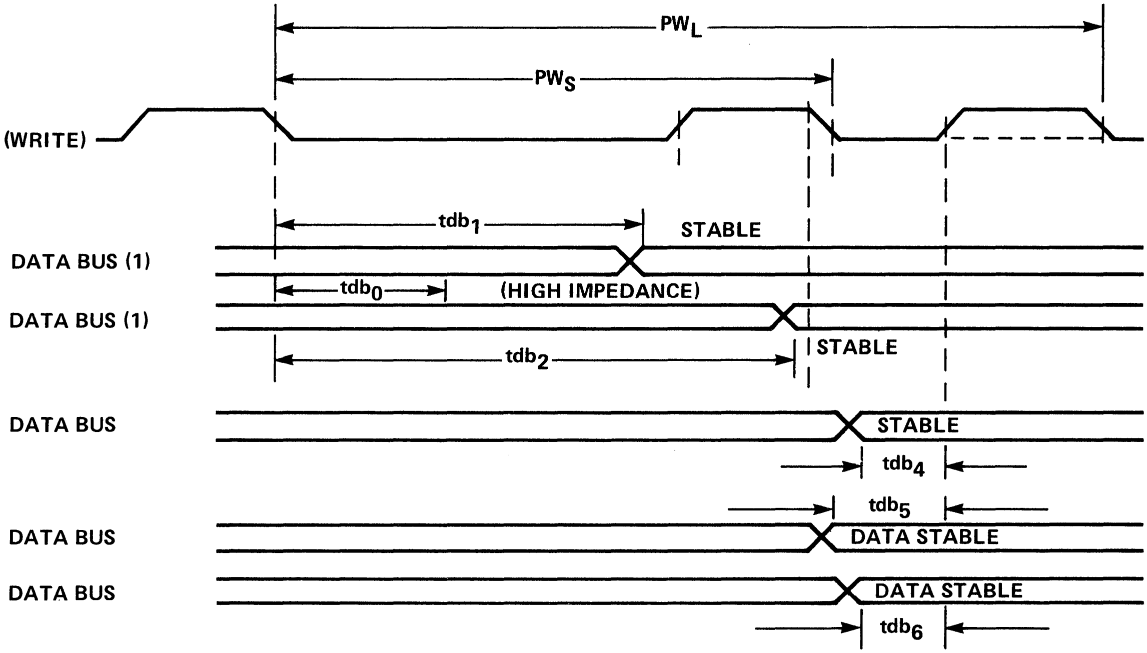
FIGURE 13 – A LONG CYCLE INSTRUCTION FETCH (DURING DS ONLY)



P_X
PW_X
XTLY
tx_1
tx_2
PW_1
Φ
PΦ
td_1
td_2
WRITE
PW_2
PW_L
ROMC
TRUE ROMC STATE 0
td_3
tdb_3
OP CODE FOR NEXT INSTRUCTION
ONE CYCLE OF THE SINGLE, LONG CYCLE DS INSTRUCTION (DECREMENT SCRATCHPAD)
NEXT INSTRUCTION

Symbols are defined in Table 5

FIGURE 14 – MEMORY REFERENCE TIMING



1. Timing for CPU outputting data onto the data bus.

   Delay tdb_1 is the delay when data is coming from the accumulator.

   Delay tdb_2 is the delay when data is coming from the scratchpad (or from a memory device).

   Delay tdb_0 is the delay for the CPU to stop driving the data bus.

2. There are four possible cases when imputting data to the CPU, via the data bus lines: they aepend on the data path and the destination in the CPU, as follows:

   tdb_3; Destination – IR (instruction Fetch) – See Figure 2-10 for details.
   tdb_4; Destination – Accumulator (with ALU operation – AM)
   tdb_5; Destination – Scratchpad (LR K,P etc.)
   tdb_6; Destination – Accumulator (no ALU operation – LM)

In each case a stable data hold time of 50 nS from the WRITE reference point is required.

Symbols are defined in Table 5

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS (Above which useful life may be impaired)

V_{GG}	+15V to –0.3V
V_{DD}	+7V to –0.3V
RC, XTLX, and XTLY	+15V to –0.3V (RC with 5KΩ series resistor)
All other inputs	+7V to –0.3V
Storage temperature	–55°C to +150°C
Operating temperature	0°C to +70°C

NOTE: All voltages with respect to V_{SS}.

SUPPLY CURRENTS (MK3850N-3, MK3850P-3)

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		30	80	mA	f = 2 MHz, Outputs unloaded
I_{GG}	V_{GG} Current		15	25	mA	Outputs unloaded

SUPPLY CURRENTS (MK3850N-13, MK3850P-13)

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		35	90	mA	f = 2 MHz, Outputs unloaded
I_{GG}	V_{GG} Current		20	33	mA	Outputs unloaded

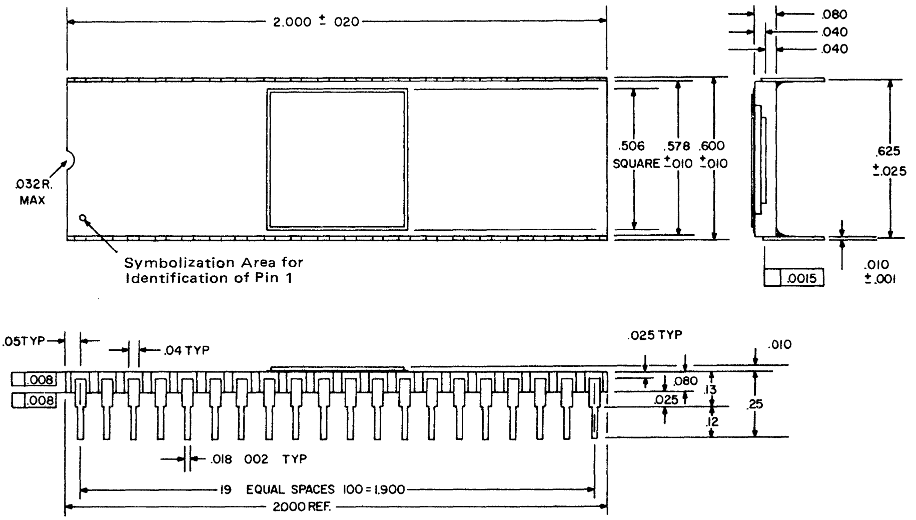
SUPPLY CURRENTS (MK3850P-23)

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		40	100	mA	f = 2 MHz, Outputs unloaded
I_{GG}	V_{GG} Current		25	40	mA	Outputs unloaded

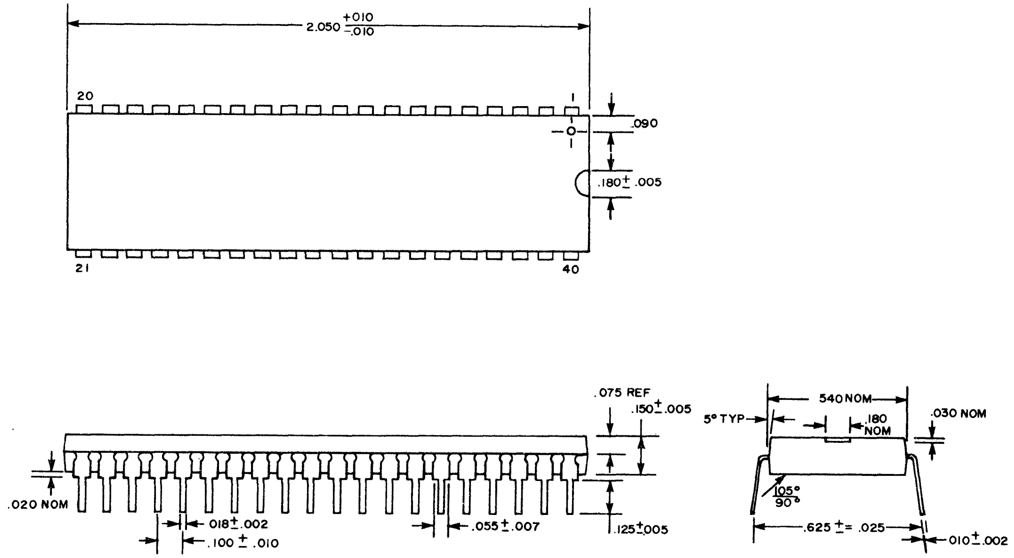
ORDER INFORMATION

PART NO.	PACKAGE TYPE	TEMPERATURE RANGE (T_A)	COMMENTS
MK3850N-3	Plastic	0°C to +70°C	
MK3850P-3	Ceramic	0°C to +70°C	
MK3850N-13	Plastic	–40°C to +85°C	
MK3850P-13	Ceramic	–40°C to +85°C	
MK3850P-23	Ceramic	–55°C to +125°C	

PACKAGE DESCRIPTION – 40-Pin Dual-In-Line Ceramic Package



PACKAGE DESCRIPTION – 40-Pin Dual-in-Line Plastic Package



INSTRUCTIONS FOR COMPLETION OF MASK PROGRAMMED PART FORM:

1. List customer name.
2. List customer address.
3. List customer city, state, and zip code.
4. List customer phone number and extension.
5. List a contact within the customer's company that can be called for reply to engineering questions.
6. List the responsible Disbributor should the order be placed through a Distributor.
7. List the ROM/PSU/3870 part number for example 3851/12XXX, 34XXX, or MK 3870/141XXX.
8. List the package type (plastic or ceramic) required by the customer for the production order (NOTE - prototypes will be Dallas assembled in ceramic).
9. List the customer part number.
10. List any special branding requirements desired by the customer (NOTE - usually the MOSTEK exclusive part will suffice for customer branding requirements).
11. List the customer specification number and indicate whether the customer intends to send a specification to MOSTEK for file. Should you circle NO this denotes that parts will be tested to the standard MOSTEK data sheet.
12. Should the customer request his specification to be on file with MOSTEK, please indicate the date that the customer spec was sent to MOSTEK.
13. Circle the pattern media that the customer wishes to use to transmit code to MOSTEK.
14. Indicate the verification media requested by the customer from MOSTEK. (NOTE - the listing is usually sufficient).
15. Check the port option requested by the customer (make reference to note #1).
16. Indicate the date that the customer's pattern was sent to MOSTEK.
17. Indicate whether the customer requires prototypes (NOTE - standard quantity of prototype Dallas assembled in ceramic is 10).
18. Indicate whether the customer requires pattern verification. Check YES or WAIVER.
19. Indicate whether the customer requires prototype verification. Indicate by checking YES or WAIVER.
20. Make any comments concerning waivers if stated above.
21. The customer purchase order to MOSTEK direct or to his Distributor.
22. List the date of the customer order.
23. List the Distributor purchase order number to MOSTEK should the order be placed through a Distributor.
24. Indicate the production quantity and price.
25. Indicate the delivery dates requested or committed to the customer; both prototypes and production. (NOTE - standard commitment is six weeks to prototype after verification of listing and twelve weeks from prototype verification to production).
26. Date this form was completed and forwarded to MOSTEK.
27. Name of Representative compieting this form.

Customer Name ____________________

Address ____________________

City ____________ State ____________ Zip ____________

Phone (_____) ____________ Extension ____________

Contact ____________________

Distributor ____________________

ROM Generic Type ____________________

Package Type ____________________

Customer Part Number ____________________

Branding Requirement ____________________

Customer Specification: ____________ □ Yes

____________________ □ No Parts to be tested to standard Data Sheet

____

Date customer spec sent to MOSTEK ____________________

PATTERN MEDIA

□ EMU-70
□ PROM
□ Paper Object Tape
□ Silent 700 Cassette
□ Card Deck
□ Tape of Card Deck

(Note 2) ____________________

VERIFICATION MEDIA

□ Listing
□ Other

PORT OPTION (Note 1)

□ Standard TTL
□ Open Drain
□ Driver Pullup

Date Pattern Data Sent to MOSTEK ____________________

Does Customer Require Prototypes □ Yes □ No

Pattern Verification Required by Customer □ Yes □ Waived

Prototype Verification Required by Customer □ Yes □ Waived

COMMENTS: (Waiver Explanation) ____________________

____

Customer Order Number ____________________

Date of Customer Order ____________________

Distributor Order Number to MOSTEK ____________________

Order Quantity and Price ____________________

Delivery Requested/Committed Prototypes ____________________

Production ____________________

Date Form Completed ____________________

Name of Representative Completing Form ____________________

MOSTEK®

F8 MICROPROCESSOR DEVICES

Program Storage Unit MK3851

FEATURES

- □ 1024 x 8 ROM storage
- □ Two 8-bit I/O Ports
- □ Programmable timer
- □ External/timer interrupt circuitry
- □ Low power dissipation < 275mW typical

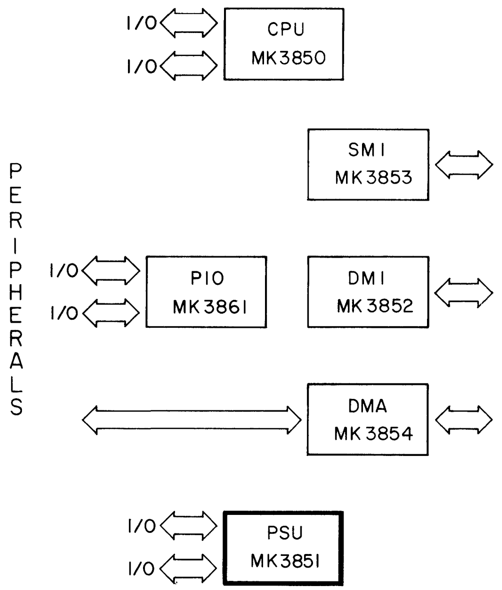
GENERAL DESCRIPTION

The MK 3851 program storage unit (PSU) provides 1024 bytes of read only memory (ROM) for the F8 system. Additionally each PSU provides two 8-bit I/O ports, a programmable timer and vectored timer and external interrupts. The PSU contains three 16-bit address registers and a 16-bit incrementer/adder. On command from the F8 CPU the MK 3851 accesses its internal memory using one of these three registers and increments or adds displacement to the register if required.

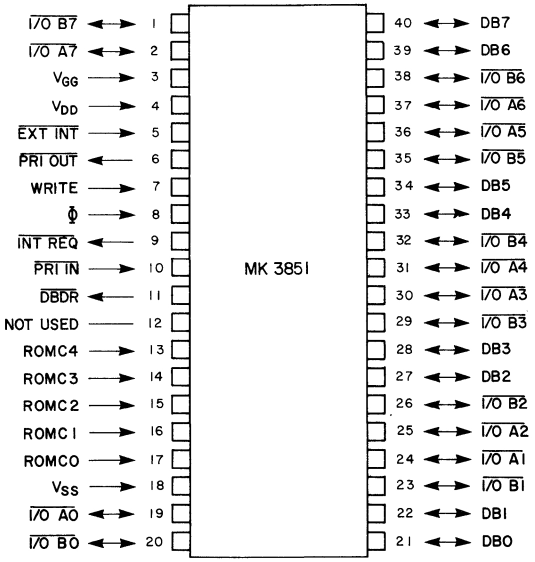
The MK 3851 PSU is manufactured using N-channel Isoplanar MOS technology. Power dissipation is very low, typically less than 275mW.

PIN NAME	DESCRIPTION	TYPE
I/O A0-I/O A7	I/O Port A	Bi-directional
I/O B0-I/O B7	I/O Port B	Bi-directional
DB0-DB7	Data Bus	Bi-directional, tri-state
ROMC0-ROMC4	Control Lines	Input
Φ, WRITE	Clock Lines	Input
EXT INT	External Interrupt	Input
PRI IN	Priority In	Input
PRI OUT	Priority Out	Output
INT REQ	Interrupt Request	Output
DBDR	Data Bus Drive	Output
V_{SS}, V_{DD}, V_{GG}	Power Supply Lines	Input

F8 FAMILY



PIN CONNECTIONS



FUNCTIONAL DIAGRAM

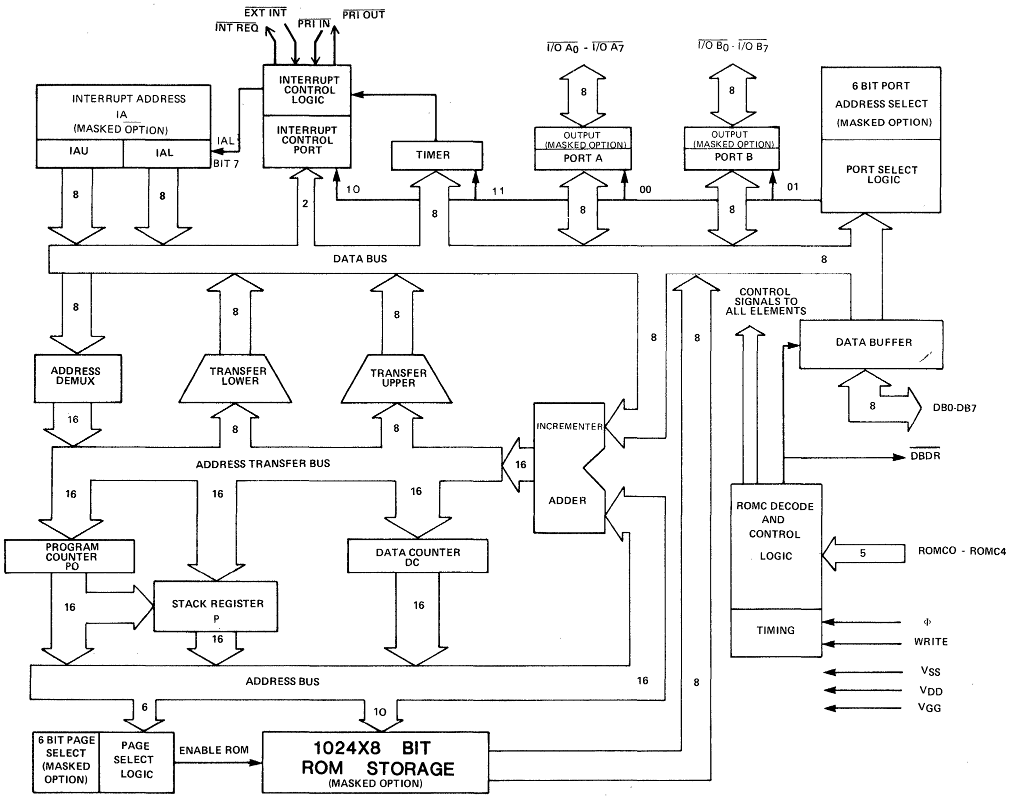


Figure 1

FUNCTIONAL PIN DESCRIPTION

Φ and WRITE are clock inputs generated by the MK 3850 CPU.

ROMCO through ROMC4 are control inputs generated by the MK 3850 CPU.

DBO through DB7 are bi-directional data bus lines which link the MK 3851 PSU with all other devices in the F8 system.

INT REQ. This signal is connected to the INT REQ input on the 3850 CPU. INT REQ is output low to interrupt the MK 3850 CPU. This occurs only if PRI IN is low, and MK 3851 PSU interrupt control logic is requesting an interrupt.

EXT INT. A high to low transition on this signal is recognized by the MK 3851 as an interrupt request from an external device.

PRI IN. This input must be low to allow the MK 3851 PSU to set INT REQ low in response to an interrupt.

PRI OUT. This signal is connected to PRI IN on the next device in the interrupt priority daisy chain. PRI OUT is output high unless PRI IN is entering the MK 3851 PSU low, and the MK 3851 PSU is not requesting an interrupt.

I/O A0 through I/O A7 and I/O B0 through I/O B7 are two Input/Output bi-directional ports through which the MK 3851 PSU communicates with logic external to the microprocessor system.

DBDR is low when the MK 3851 PSU is outputting data on the data bus (DB0-DB7). DBDR is an open drain signal.

DEVICE ORGANIZATION

This section describes the operation of the basic functional elements of the MK 3851 PSU. These elements are shown on the PSU functional block diagram. (Fig.1)

ROM STORAGE

The MK 3851 PSU has 1024 bytes of read-only memory. This ROM array may contain object program code and/or tables of non-varying data. Every MK 3851 PSU is implemented using a custom mask which specifies the state of every ROM bit, as well as certain address mask options which are external to the ROM array.

THE PROGRAM COUNTER (P0) AND DATA COUNTER (DC)

The MK 3851 PSU addressing logic consists primarily of two 16-bit registers: the program counter (P0) and the data counter (DC).

The program counter will at all times address the memory word from which the next object program code must be fetched. The data counter addresses memory words containing individual data bytes or bytes within data tables to be used as operands. The mechanism whereby an address is decoded by the MK 3851 PSU logic is identical, whether the address originated in P0 or in DC.

Recall that P0 always addresses the memory location out of which the next object program instruction byte will be read. If the instruction requires data (an operand) other than an immediate operand to be accessed, DC must address memory. P0 cannot be used to address a non-immediate operand since P0 is saving the address of the next instruction code.

THE STACK REGISTER

The MK 3851 PSU addressing logic contains a third 16-bit register, called the stack register. The stack register is labeled P on Figure 1. The stack register is a buffer for the program counter P0. The contents of the stack register are never used directly to address memory.

The following instructions access P:

LR K,P	MOVE THE CONTENTS OF P TO THE CPU SCRATCHPAD K REGISTERS
LR P,K	MOVE THE CONTENTS OF THE CPU K SCRATCHPAD REGISTERS TO P
PK	SAVE THE CONTENTS OF P0 IN P THEN MOVE THE CONTENTS OF CPU SCRATCHPAD REGISTERS 12 AND 13 TO P0
PI H'aaaa'	MOVE THE CONTENTS OF P0 TO P THEN LOAD THE HEXADECIMAL VALUE INTO P0
POP	MOVE THE CONTENTS OF P TO P0

In addition, when an interrupt is acknowledged, the contents of P0 are saved in P.

PAGE SELECT LOGIC

All memory addresses are 16-bits wide, whether the memory address originates in the program counter or the data counter. Addressing logic within the MK3851 PSU separates the 16-bit address into two portions. The low-order 10 bits address one of the PSU's 1024 bytes of ROM storage. The high order 6-bits constitute a page select.

Every MK 3851 PSU has a 6-bit page select register, which is a mask option that must be specified when the PSU ROM chip is ordered. If the high order six bits of the address match the page select mask, an enable signal will be generated which causes PSU logic to respond to a memory access request. If the high order 6-bits of the address do not match the page select, no enabling signal is generated and the PSU will not respond to memory access requests.

The 6-bit page select register may be looked upon as identifying the memory addressing space of the individual MK 3851 PSU device. Each of the 64 page select options allowed by the 6-bit page select register identifies a single address space consisting of 1024 contiguous memory addresses. Following are two examples:

Page Select Mask:
0 0 0 0 0 0

PSU Address Space:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 H'0000'
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 through H'03FF'

Page Select Mask:
0 0 1 0 1 1

PSU Address Space:
0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 H'2C00'
0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 through H'2FFF'

Six high order address bits   Ten low order address bits

INCREMENTER ADDER LOGIC

There are only two arithmetic operations that memory devices need to perform on the contents of memory address registers:

1. Increment by 1 the 16-bit value stored in an address register.
2. Add an 8-bit value, treated as a signed binary number (subject to twos complement arithmetic) to the 16-bit value stored in an address register.

The incrementer adder logic performs these two functions in the MK 3851 PSU.

INTERRUPT LOGIC

This logic responds to an interrupt request signal which may originate internally from timer logic, or be input by an external device. Based on priority considerations, the interrupt request is passed on to the MK 3851 CPU.

TIMER LOGIC

Every MK 3851 PSU has a polynomial shift register which may be used in conjunction with interrupt logic to generate real-time intervals.

Upon counting down to zero, the timer uses interrupt logic to signal that it has timed out.

The timer is programmable and is handled as though it were an I/O port. Using an OUT or OUTS instruction, a value may be loaded into the timer in order to determine the real-time period at the end of which a time-out interrupt will be generated.

THE DATA BUS

The 8-bit data bus is the main path for transfer of information between the MK 3850 CPU and other devices in the F8 microprocessor system. It is identified in Figure 1 by data lines DB0-DB7.

I/O PORTS

Every MK 3851 PSU has four, 8-bit I/O ports. Associated with the I/O ports is an I/O port address select register. This is a 6-bit register, the contents of which is a PSU mask option, that must be specified at the time the MK 3851 PSU is ordered.

Two of the four I/O ports, identified as I/O ports A and B in Figure 1, are used to transfer data to or from external devices. A third I/O port is assigned to the programmable timer while the fourth port is the Interrupt Control Port.

The four I/O ports of any MK 3851 PSU are addressed by an 8-bit I/O port address. The high order 6 bits are specified by the I/O port address select code with the remaining 2 bits identifying the particular I/O port as following:

XXXXXX00 I/O Port A
XXXXXX01 I/O Port B
XXXXXX10 Interrupt control
XXXXXX11 Programmable Timer

XXXXXX represents a six bit PSU mask option. For example, if the six are 000010, the four I/O port addresses are H'08', H'09', H'0A' and H'0B'.

When a logic "1" is output to I/O port A or B, it places a 0 volt level on the output pin. This same negative true logic also applies to input. The I/O ports, timer, and interrupt control ports are not initialized during the power on reset.

MASK OPTIONS

The following mask options must be specified for every MK 3851 PSU:

1. The 1024 bytes of ROM storage. This will reflect programs and permanent data tables stored in the PSU memory.
2. The 6-bit page select. This defines the PSU address space
3. The 6-bit I/O port address select. This defines the four PSU I/O port addresses.
4. The 16-bit interrupt address vector, excluding bit 7.
5. The I/O port output option. The choices are the standard Pull-up (Option A), the Open-Drain (Option B) and the Driver Pull-up (Option C)

OPERATIONAL DESCRIPTION

CLOCK TIMING

All timing within the MK 3851 PSU is controlled by Φ and WRITE, which are input from the MK 3850 CPU.

The WRITE clock refreshes and updates MK 3851 PSU address registers, which are dynamic.

The Φ clock drives sequencing logic to precharge the ROM matrix. The Φ clock also drives the programmable timer.

INSTRUCTION EXECUTION

The MK 3851 PSU responds to signals which are output by the MK 3850 CPU in the course of executing instruction cycles.

Table 1 summarizes the response of the MK 3851 PSU to the ROMC states.

MEMORY ADDRESSING

Those ROMC states which specify a memory access call for only one memory device to respond to the memory access operation. However, every memory device responds to ROMC states that call for modification of program counter or data counter register contents. Consider two examples:

1. ROMC state 5 specifies that the data counter (DC) register contents must be incremented. Every memory device will simultaneously receive this ROMC state, and will simultaneously increment the contents of its DC register.
2. ROMC state 0 is the standard instruction fetch. Only the memory device whose address space includes the current contents of the program counter (P0) registers will respond to this ROMC state by accessing memory and placing the contents of the addressed memory word on the 8-bit data bus. However, every memory device will increment the contents of its P0 register, whether or not the P0 register contents are within the memory space of the device.

When all memory devices connected to the 8-bit data bus of a MK 3850 CPU are also connected to the ROMC control lines of the same CPU, the memory devices simultaneously receive the same ROMC state signals from the CPU and respond to ROMC states by identically modifying the contents of memory address registers. Therefore the P0 register on all memory devices contain identical information. The same holds true for DC and P registers.

Only the memory device whose address space includes the specified memory address, will respond to any memory access request. To avoid addressing conflicts, it is necessary to insure that the following three conditions exist:

1. Memory devices must receive the ROMC state signals from one CPU.
2. Page select masks must not be duplicated. (More than one memory device cannot have the same memory space).
3. The memory address contained in the specified register (P0 or DC) must be within the memory space of a memory device.

DATA OUTPUT BY THE PSU

Figure 10 shows the timing when the MK 3851 PSU outputs data on the data bus. This timing applies whenever a MK 3851 PSU is the data source. The MK 3851 PSU always places data on the data bus in time for the set-up required by an MK 3850 CPU destination.

ROMC STATES

ROMC (Hexadecimal)	OPERATION PERFORMED	COMMENT
00	DB ← ((P0)) ; P0←P0 +1	OP CODE, FETCH
01	DB ← ((P0)) ; P0←P0 +DB	BRANCH OFFSET FETCH
02	DB ←((DC)); DC←DC+1	
03	DB ←((P0)) ; P0←P0+1	IMMEDIATE OPERAND FETCH
04	P0 ←P	
05	((DC))←DB ; DC←DC+1	MK 3851:DC← DC+1 ONLY
06	DB←DCU	
07	DB←PU	
08	P←P0 ; DB←H'00'; P0L, P0H← DB	EXTERNAL RESET
09	DB ←DCL	
0A	DC ←DC+DB	
0B	DB ←PL	
0C	DB←((P0)) ; DCL←DB	
0D	P←P0+1	
0E	DB←((P0)) ; DCL←DB	
0F	P←P0 ; DB←IAL ; P0L← DB	LOWER BYTE OF ADDRESS VECTOR
10	FREEZE INTERRUPT STATUS	PREVENT ADDRESS VECTOR CONFLICTS
11	DB←((P0)) ; DCU←DB	
12	P0L←DB ; P←P0	
13	DB←IAU; P0U←DB	UPPER BYTE OF ADDRESS VECTOR
14	P0U←DB	
15	PU←DB	
16	DCU←DB	
17	P0L ←DB	
18	PL←DB	
19	DCL←DB	
1A	((pp))←DB or ((p))←DB	
1B	DB←(pp)) or DB←((p))	
1C	NO OPERATION	
1D	DC ⇄ DC1	MK 3851: NO OPERATION
1E	DB←P0L	
1F	DB←P0U	

Definitions	DB -	Data Bus	IA -	Interrupt address vector
	P0 -	Program Counter	L -	Lower byte suffix
	DC -	Data Counter	U -	Upper byte suffix
	P -	Stack Register	( ) -	Contents of
	pp -	Two hex digits (long I/O port address)	← -	transfer to
	p -	One hex digit (short I/O port address)	⇄ -	exchange

Table 1

Observe that $\overline{DBDR}$ is low while data output by the MK 3851 PSU is stable on the data bus. Thus $\overline{DBDR}$ low indicates that the data bus currently contains data flowing from a MK 3851 PSU. For systems with more than one MK 3851 PSU the $\overline{DBDR}$ outputs may be wire-ORed and the result may be used as a bus data flow direction indicator. $\overline{DBDR}$ may remain low until td8 into the instruction cycle following the one in which $\overline{DBDR}$ was set low.

DATA INPUT TO THE PSU

The worst case timing for the MK 3851 PSU receiving data from the data bus is when the data must be added to a 16 bit number within the PSU's Incrementer Adder. This worst case corresponds to data coming from the accumulator in the CPU for an ADC instruction or from a memory device for a BR instruction. For this worst case, arriving data must allow sufficient time for 16-bit Adder logic.td4 in Figure 10 identifies this worst case timing.

INPUT/OUTPUT INTERFACING

The two PSU I/O ports with addresses xxxxxx00 and xxxxxx01 (xxxxxx is the 6-bit I/O port address select) may be used to transmit data between the PSU and external devices. IN and INS instructions

STANDARD PULL–UP CONFIGURATION

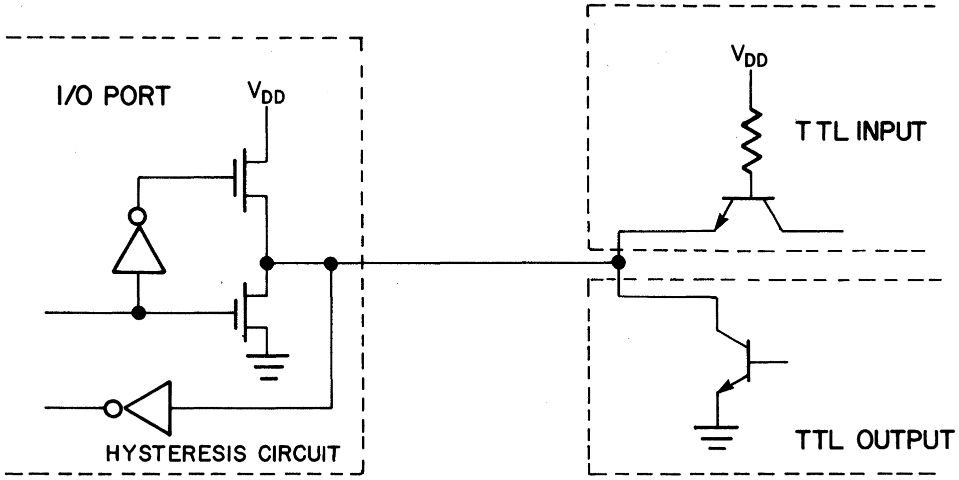


Figure 2

OPEN DRAIN CONFIGURATION

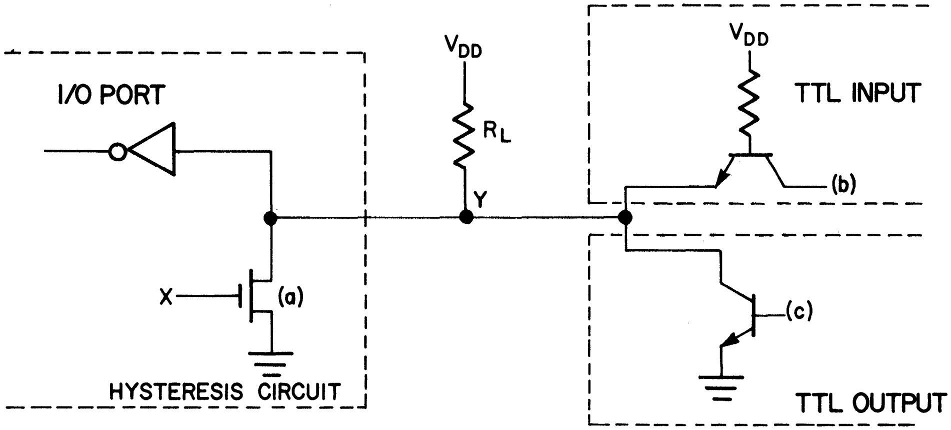


Figure 3

DRIVER PULL–UP CONFIGURATION

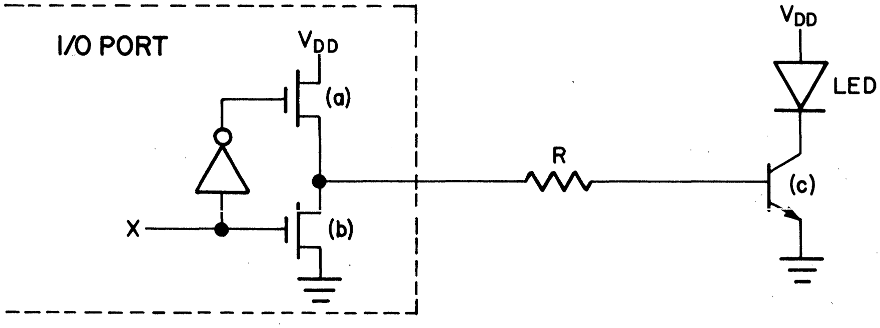


Figure 4

cause data at the I/O ports to be transmitted to the CPU. OUT and OUTS instructions cause data in the CPU's accumulator to be loaded into an I/O port latch.

Data bus timing associated with the execution of I/O instructions does not differ from data bus timing associated with any other data transfer to or from the PSU. However, timing at the I/O port depends on which port option is being used. Figures 2,3 and 4 illustrate the three port options. Figure 11 illustrates timing for the three cases. Figure 2 illustrates the standard pull-up configuration.

When the I/O port is configured as shown in Figure 3 the drain connection of FET (a) is "open", (not connected to V_{DD} through a pull-up transistor). This option is most useful in applications where several signals (possibly several I/O port lines) are to be wire-ORed together. A common external pull-up, R_L, is used to establish the 2 high output levels. Another advantage of this option is that the output (point Y) may be tied through a pull-up resistor to a voltage higher than V_{DD} (up to V_{GG}) for interfacing to external circuits requiring a higher level than V_{DD} would provide. The process of inputting and outputting with this configuration can be described as follows:

If a high level is present at point X, (this would be coming from the port latch), FET (a) will conduct and pull point Y to a low level by current flow through R_L. This low level at Y will cause transistor (b) to turn on and present a low level to the input TTL circuit. If a low level is present at X, FET (a) will turn off and point Y will be pulled toward V_{DD} by R_L. This causes transistor (b) to turn off and present a high level to the internal TTL circuits.

When data is input, a high level at the base of transistor (c) causes it to conduct and pull point Y low. This transfers a high level to the internal I/O port logic through the inverting hysteresis circuit. If a low level is present at the base of (c), conduction stops and point Y is pulled toward V_{DD} by R_L. This is then transferred as a low level to the internal I/O port logic through the hysteresis circuit.

Figure 4 shows the I/O port driver pull-up option shown driving a LED indicator. This application is typical of a front-panel address or data display, where a row of LED indicators shows the logic state of an I/O port. In this case, a high level at X turns FET (b) on and (a) off, providing a path for current through resistor R from the base of transistor (c). This stops (c) from conducting and the LED does not light. However, if a low level is present at X, (b) turns off and (a) turns on, providing a path for current from V_{DD} through (a) to R. This current through R turns on (c), causing the LED to conduct and be lit.

The three options for I/O port output configurations described above are provided to aid the designer in optimizing (minimizing) the system hardware for his particular application. The option is specified as a mask option by the designer.

THE PROGRAMMABLE TIMER

The MK 3851 PSU has an 8-bit shift register, addressable as I/O port xxxxxx11, which may be used as a programmable timer. (xxxxxx is the 6-bit I/O port address select, a PSU mask option.) Figure 5 illustrates the shift register logic and the exclusive-OR feedback path.

CONVERSION OF TIMER COUNTS INTO TIMER CONTENTS

	0	1	2	3	4	5	6	7	8	9
0	7F	BF	5F	2F	97	CB	E5	72	39	1C
1	0E	87	43	A1	D0	E8	F4	7A	3D	1E
2	0F	07	03	01	00	80	C0	60	B0	D8
3	EC	F6	7B	BD	5E	AF	D7	6B	35	1A
4	0D	06	83	41	A0	50	A8	54	AA	55
5	2A	15	8A	C5	E2	F1	F8	7C	3E	9F
6	CF	E7	73	B9	5C	AE	57	2B	95	CA
7	65	32	99	CC	66	B3	59	2C	16	0B
8	05	02	81	40	20	10	08	84	C2	61
9	30	98	4C	26	13	89	44	22	11	88
10	C4	62	B1	58	AC	56	AB	D5	6A	B5
11	5A	AD	D6	EB	75	BA	DD	6E	B7	5B
12	2D	96	4B	A5	D2	E9	74	3A	9D	CE
13	67	33	19	8C	C6	63	31	18	0C	86
14	C3	E1	70	38	9C	4E	27	93	C9	E4
15	F2	79	BC	DE	EF	77	BB	5D	2E	17
16	8B	45	A2	51	28	14	0A	85	42	21
17	90	48	24	12	09	04	82	C1	E0	F0
18	78	3C	9E	4F	A7	D3	69	34	9A	4D
19	A6	53	29	94	4A	25	92	49	A4	52
20	A9	D4	EA	F5	FA	7D	BE	DF	6F	37
21	1B	8D	46	23	91	C8	64	B2	D9	6C
22	B6	DB	6D	36	9B	CD	E6	F3	F9	FC
23	7E	3F	1F	8F	47	A3	D1	68	B4	DA
24	ED	76	3B	1D	8E	C7	E3	71	B8	DC
25	EE	F7	FB	FD	FE	FF halts timer				

Each timer count = 15.5 µs at 2MHz

Table 2

Based on the logic illustrated in Figure 5 binary values in the range 0 through 254 when loaded into the timer, are converted into "timer counts", as shown in table 2. Table 2 contains the actual (HEX) value loaded into the timer, and the column/row is the corresponding decimal number of time intervals the timer will take to time out. Data cannot be read out of the programmable timer I/O port.

Either the OUT or OUTS instruction is used to load "timer counts" into the programmable timer. The contents of the programmable timer can not be read using an IN or INS instruction. The timer will time out after a time interval given by the product:
(period of clock Φ) X (timer counts) X 31

For example, a value of H 'C8' loaded into the programmable timer becomes 215 timer counts. The timer will therefore time out in 3.33 milliseconds, if the period of clock signal Φ is 500 nanoseconds.

A value of H'FF' loaded into the programmable timer will stop the timer. This is because the timer shift

TIMER LOGIC DIAGRAM

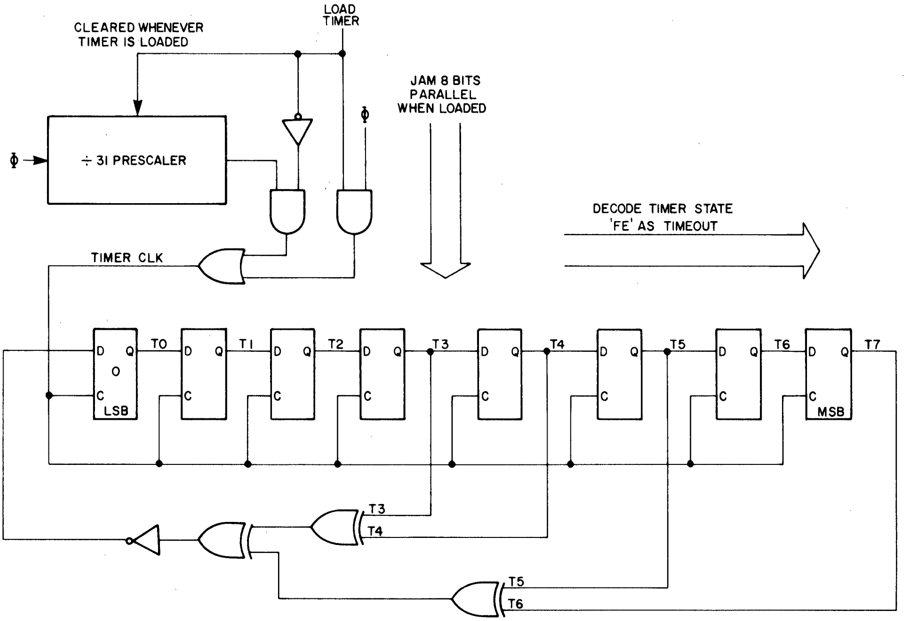

CLEARED WHENEVER TIMER IS LOADED
LOAD TIMER
JAM 8 BITS PARALLEL WHEN LOADED
Φ
÷ 31 PRESCALER
Φ
DECODE TIMER STATE 'FE' AS TIMEOUT
TIMER CLK
D Q
0
C
LSB
T0
T1
T2
T3
T4
T5
T6
T7
MSB
T3
T4
T5
T6

Figure 5

register feedback gates will always present a logic 1 to the D input of the LSB flip-flop (fig. 5). Therefore the timer will retain a value to H'FF' and a H'FE' will never be decoded to cause a time out.

The timer runs continuously unless it has been stopped by loading H'FF' into it. Upon timing out, the timer transmits an interrupt request to the interrupt logic. If proper interrupt logic conditions exist, the timer interrupt request is passed on to the CPU via INT REQ.

After the programmable timer has timed out it will again time out after 255 time counts. Therefore if the programmable timer is simply left running, it will time out every 7905 Φclock periods, or every 3.9525 milliseconds for a 500 nanosecond clock.

Whenever the timer and timer interrupt are being set to time a new interval, the timer should be loaded before enabling the timer interrupt. The act of loading the timer clears any pending timer interrupts. When the timer interrupt is enabled, any pending timer interrupt will be acknowledged and forwarded to the CPU. Since the timer runs continuously (unless stopped under program control) enabling the timer before loading a time count can cause a spurious interrupt. Time outs of the timer are latched in the interrupt logic of the PSU, even while timer interrupts are disabled. When the timer is enabled, an immediate interrupt acknowledge will occur if the continuous running timer timed out while timer interrupts were disabled.

If the timer is loaded just prior to enabling timer interrupts a spurious interrupt request will not exist when the timer interrupt is enabled.

Figure 6 illustrates a possible sequence for a timer which is initially loaded with H'C8' then allowed to run continuously.

INTERRUPT LOGIC ORGANIZATION

The Interrupt Control Port has the I/O port address xxxxxx10, where xxxxxx is the 6-bit I/O port address select. Data is loaded into this register (I/O port) using an OUT or OUTS instruction. Data cannot be read from this port. The contents of the Interrupt Control Port are interpreted as follows:

CONTENTS OF INTERRUPT CONTROL PORT	FUNCTION
B'xxxxxx00'	Disable all interrupts
B'xxxxxx01'	Enable external interrupt, disable timer interrupt
B'xxxxxx10'	Disable all interrupts
B'xxxxxx11'	Disable external interrupt, enable timer interrupt

In the above I/O port contents definitions x represents "don't care" bits.

Depending on the contents of the Interrupt Control Port, a MK 3851 PSU's interrupt control logic can be accepting timer interrupts, or external interrupts, or neither, but never both.

Figure 7 is a conceptual logic diagram of the PSU's interrupt logic. Between the EXT INT input or the time-out input and the output INT REQ, there are 4 flip-flops. EXT INT and the time-out interrupt input each have 2 synchronizing flip-flops to detect the active edge.

TIME OUT AND INTERRUPT REQUEST

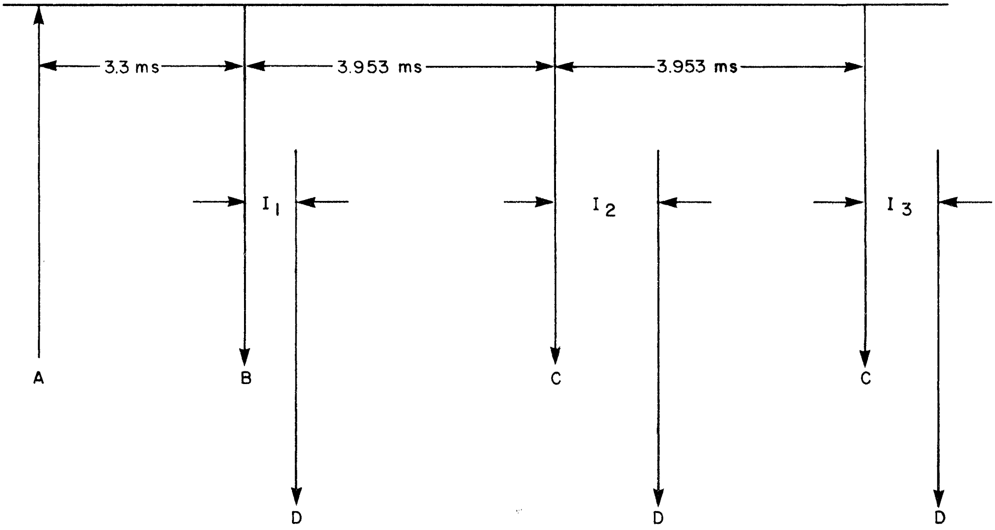


Figure 6

Each edge detect circuit is followed by its own INTERRUPT flip-flop which latches the true condition.

The outputs of the TIMER INTERRUPT flip-flop and the EXTERNAL INTERRUPT flip-flop are ORed to set the SERVICE REQUEST flip-flop, providing that an interrupt from some other PSU is not being acknowledged by the CPU.

INT REQ is the NAND of PRI IN and SERVICE REQUEST.

INT REQ is an open drain signal. The INT REQ signal of several PSU's may be tied together so that any one can force the line to 0V if it is requesting interrupt service. A pull-up to V_{DD} is provided by the MK 3850 CPU to INT REQ input pin.

PRI IN is part of the interrupt priority chain. The chain begins by a strap to V_{SS}. Each device in the chain has a PRI IN input and PRI OUT output. PRI OUT of the PSU will be true (0V) only if PRI IN is true (0V) and SERVICE REQUEST is false. This means that when PRI IN is true (0V), PRI OUT and INT REQ are always at opposite levels. PRI OUT is connected to PRI IN on the next device in the interrupt priority daisy chain, if there is one. The function of the priority daisy chain is to insure that just one device at a time be requesting interrupt service.

The SERVICE REQUEST flip-flop cannot be set if another interrupt request is in the process of being acknowledged anywhere in the system. If an interrupt request has been latched into the TIMER INTERRUPT flip-flop, or the EXTERNAL INTERRUPT flip-flop, the PSU logic waits until after the process of acknowledging the other interrupt has been completed before setting SERVICE REQUEST. This precaution is necessary to insure that the priority chain is not altered during acknowledgement. Chaos would result if half of the interrupt vector came from one device and the second half from some other device.

The SERVICE REQUEST flip-flop is cleared after an interrupt from the PSU has been acknowledged. It is also cleared whenever the PSU's interrupt control register is accessed by an output instruction.

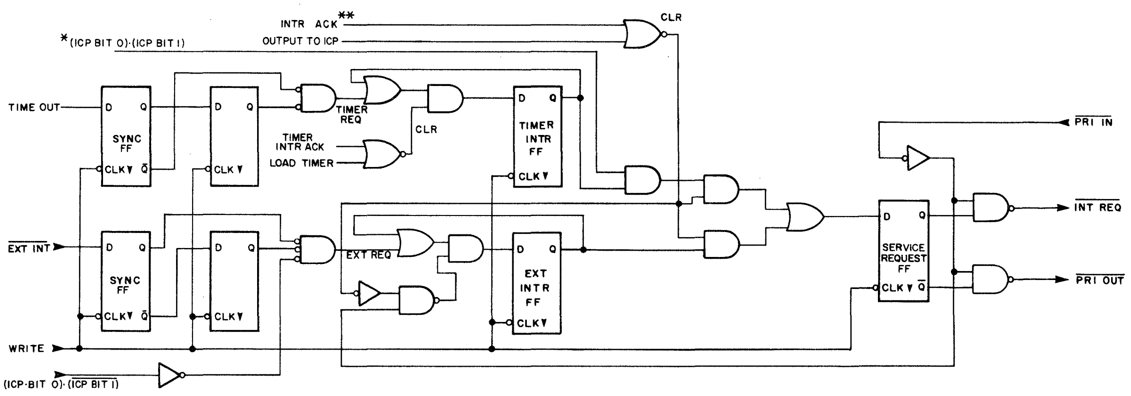
The conditions for setting the TIMER INTERRUPT flip-flop and the EXTERNAL INTERRUPT flip-flop differ slightly. External interrupts must be enabled before the EXTERNAL INTERRUPT flip-flop can be set by a negative going transition of EXT INT. However, TIMER INTERRUPT will be set by a timer TIME OUT independent of Interrupt Control Port bit 1. This means that the PSU can detect a time out interrupt that occurred while the external interrupt was enabled in the PSU.

The TIMER INTERRUPT flip-flop is cleared whenever the PSU's timer is loaded or when its timer interrupt has been acknowledged. The EXTERNAL INTERRUPT flip-flop is cleared whenever the PSU's interrupt control register is accessed by an output instruction, or when its external interrupt has been acknowledged.

INTERRUPT ACKNOWLEDGE SEQUENCE

Upon receiving an interrupt request, whether from an external source via EXT INT or from the internal timer, the PSU and CPU go through an interrupt

INTERRUPT LOGIC



* ICP-Interrupt Control Port
**1 During Event G in Fig 8

Figure 7

sequence which results in the execution of an interrupt service routine located at the memory address pointed to by the Interrupt Address Vector. Figures 8 and 9 illustrate the interrupt sequence for the two cases. Events occuring in these sequences are labeled with the letters A through H. Events are described as follows.

EVENT A

The initial interrupt request arrives. The falling edge of EXT INT pin identifies an external interrupt. The rising edge of interval timer output indicates a timeout.

EVENT B

The synchronizing flip-flop in the PSU control logic changes state.

EVENT C

The timer interrupt, or external interrupt flip-flop goes true, indicating the local interrupt logic's acknowledgement of the interrupt. The timer interrupt flip-flop will always respond and save the time-out occurrence, whereas the external interrupt flip-flop will only be set at this time if the external interrupt mode is enabled within the local control logic.

EVENT D

The INT REQ line is pulled low by the PSU, passing the request for servicing on to the CPU. The conditions that must be present for this to occur are:

The PRI IN pin must be low.

The proper enable state must exist in the local control logic for the type of interrupt (timer or external).

The system is not already into Event F due to servicing some other interrupt.

EVENT E

The CPU now begins its response to the INT REQ, line by outputting the unique ROMC state H'10' inhibiting modification of interrupt priority logic. This will only occur when the following conditions are satisfied:

The CPU is executing the last cycle of an instruction (beginning an instruction fetch).

The ICB is enabled (ICB = 0).

The current instruction fetch is not protected (not a privileged instruction).

EVENT F

The CPU generates the interrupt acknowledge sequence of ROMC states as follows:

ROMC STATE	
10	Inhibit modification of interrupt priority logic.
IC	No function
0F	Put lower byte of interrupt address vector on data bus
13	Put upper byte of interrupt address vector on data bus
00	Fetch instruction from memory (first instruction of interrupt service routine)

EVENT G

At this point the CPU begins fetching the first instruction of the interrupt service routine. In the PSU interrupt logic, the SERVICE REQUEST flip-flop and the appropriate INTERRUPT REQUEST flip-flop are cleared.

EVENT H

The CPU begins executing the first instruction of the interrupt service routine.

INTERRUPT ADDRESS VECTOR

During the interrupt acknowledge, the interrupting PSU provides a 16-bit interrupt address vector. The CPU causes this vector to be loaded into P0, so that program execution can branch to the routine that handles this particular interrupt. Fifteen bits of the interrupt vector are specified as a mask option. Bit 7 cannot be masked. It is set by the interrupt control logic to 0 if the timer interrupt is enabled or to a 1 if external interrupt is enabled. The interrupt vector is of the form:

WWWW, XXXX, 0YYY, ZZZZ for timer interrupt and WWWW, XXXX, 1YYY, ZZZZ for external interrupt where W,X Y and Z are the mask specified bits.

INTERRUPT SIGNALS TIMING

Timing for signals associated with the MK 3851 interrupt logic is shown in Figure 12.

TIMER INTERRUPT SEQUENCE

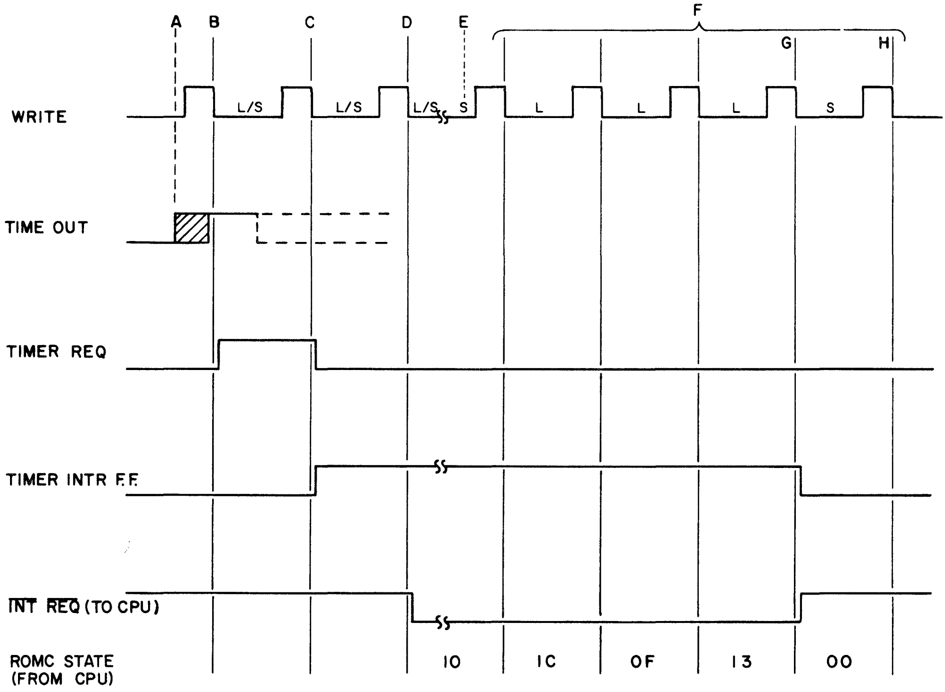


Figure 8

EXTERNAL INTERRUPT SEQUENCE

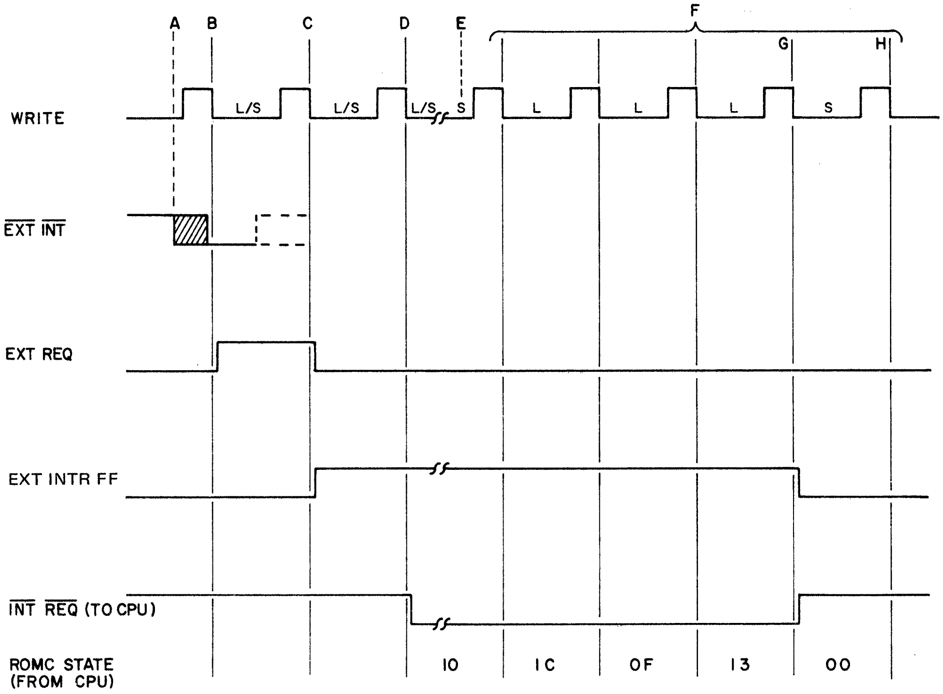


Figure 9

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS (Above which useful life may be impaired)

V_{GG}	+15V to –0.3V
V_{DD}	+7V to –0.3V
I/O Port Open Drain Option	+15V to –0.3V
External Interrupt Input	–1mA to +225 μA
All other inputs & outputs	+7V to –0.3V
Storage Temperature	–55°C to +150°C
Operating Temperature	0°C to +70°C

Note: All voltages with respect to V_{SS}.

DC CHARACTERISTICS

V_{SS} = 0V, V_{DD} = +5V ±5%; V_{GG} = +12V ± 5%, T_A = 0°C to + 70°C

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		30	70	mA	f = 2MHz, Outputs Unloaded
I_{GG}	V_{GG} Current		10	18	mA	f = 2MHz, Outputs Unloaded

DC SIGNAL CHARACTERISTICS
V_{DD} = +5V ± 5%, V_{GG} = + 12V ± 5%, V_{SS} = 0V T_A = 0-70°C

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
DATA BUS (DB0-DB7)	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = −100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6mA
	I_{IH}	Input High Current		1	μA	V_{IN} = V_{DD}, tri-state mode
	I_{OL}	Input Low Current		−1	μA	V_{IN} = V_{SS}, tri-state mode
CLOCK LINES (Φ,WRITE)	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		1	μA	V_{IN} = V_{DD}
PRIORITY IN AND CONTROL LINES ($\overline{\text{PRI IN}}$, ROMC0—ROMC4)	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		1	μA	V_{IN} = V_{DD}
PRIORITY OUT ($\overline{\text{PRI OUT}}$)	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = −100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 100 μA
INTERRUPT REQUEST ($\overline{\text{INT REQ}}$)	V_{OH}	Output High Voltage			Volts	Open Drain Output [1]
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1mA
	I_L	Leakage Current		1	μA	V_{IN} = V_{DD}
DATA BUS DRIVE ($\overline{\text{DBDR}}$)	V_{OH}	Output High Voltage				Open Drain Output
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6mA
	I_L	Leakage Current		1	μA	V_{IN} = V_{DD}
EXTERNAL INTERRUPT ($\overline{\text{EXT INT}}$)	V_{IH}	Input High Voltage	3.5	15	Volts	
	V_{IL}	Input Low Voltage	$-V_{SS}$	1.2	Volts	
	V_{IC}	Input Clamp Voltage		15	Volts	I_{IH} = 185 μA
	I_{IH}	Input High Current		10	μA	V_{IN} = V_{DD}
	I_{IL}	Input Low Current	−250	−750	μA	V_{IN} = V_{SS}
I/O PORT OPTION A	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = −30 μA
(STANDARD PULL-UP)	V_{OH}	Output High Voltage	2.9	V_{DD}	Volts	I_{OH} = −100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6mA
	V_{IH}	Input High Voltage	2.9	V_{DD}	Volts	Internal Pull-up to V_{DD} [3]
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Input Low Current		−1.6	mA	V_{IN} = 0.4V [4]
I/O PORT OPTION B (OPEN DRAIN)	V_{OH}	Output High Voltage				Open Drain Output
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6mA
	V_{IH}	Input High Voltage	2.9	V_{DD}	Volts	[3]
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IL}	Leakage Current		2	μA	V_{IN} = V_{DD}
I/O PORT OPTION (DRIVER PULL—UP)	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = −850 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6mA

Notes:

1. Pull-up resistor to V_{DD} on CPU.
2. Positive current is defined as conventional current flowing into the pin referenced.
3. Hysteresis input circuit typically provides additional 0.3V noise immunity while internal/external pull-up provides TTL compatibility.
4. Measured while I/O port is outputting a high level.

AC CHARACTERISTICS

V_{SS} = 0V, V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0°C to + 70°C

Symbols in this table are used by all timing diagrams.

Symbol	Parameters	Min	Typ	Max.	Units	Test Conditions/ Comments
$P\Phi$	ΦPeriod	0.5		10	µs	
PW_1	ΦPulse Width	180		$P\Phi$–180	ns	t_r, t_f = 50 ns typ
td_1	$\Phi$to WRITE + Delay	0		250	ns	C_L = 100pF
td_2	$\Phi$to WRITE – Delay	0		225	ns	C_L = 100pF
td_4	WRITE to DB Input Delay			$2P\Phi$+1.0	µs	
PW_2	WRITE Pulse Width	$P\Phi$–100		$P\Phi$	ns	t_r,t_f = 50 ns typ.
PW_S	WRITE Period; Short		$4P\Phi$			
PW_L	WRITE Period; Long		$6P\Phi$			
td_3	WRITE to ROMC Delay			500	ns	
td_6	WRITE to DB Output Delay	$2P\Phi$+100–td_2	$2P\Phi$+200	$2P\Phi$+800–td_2	ns	C_L = 100pF
td_7	WRITE to $\overline{\text{DBDR}}$ –Delay	$2P\Phi$+100–td_2	$2P\Phi$+200	$2P\Phi$+800–td_2	ns	C_L = 100pF
td_8	WRITE to $\overline{\text{DBDR}}$ + Delay		200		ns	Open Drain
tr_1	WRITE to $\overline{\text{INT REQ}}$ – Delay			430	ns	C_L = 100 pF [1]
tr_2	WRITE to $\overline{\text{INT REQ}}$ + Delay			430	ns	C_L = 100pF [3]
tpr_1	PRI IN to $\overline{\text{INT REQ}}$ – Delay			240	ns	C_L = 100pF [2]
tpr_2	PRI IN to $\overline{\text{INT REQ}}$ + Delay			430	ns	C_L = 100pF
tpd_1	PRI IN to $\overline{\text{PRI OUT}}$ –Delay			300	ns	C_L = 50pF
tpd_2	PRI IN to $\overline{\text{PRI OUT}}$ + Delay			365	ns	C_L = 50pF
tpd_3	WRITE to $\overline{\text{PRI OUT}}$ + Delay			700	ns	C_L = 50pF
tpd_4	WRITE to $\overline{\text{PRI OUT}}$ – Delay			640	ns	C_L = 50pF
t_{sp}	WRITE to Output Stable			1.7	µs	C_L = 50pF, Standard Pull-up
t_{od}	WRITE to Output Stable			1.7	µs	C_L = 50pF R_L = 12.5K Ω Open Drain
t_{dp}	WRITE to Output Stable		200	400	ns	C_L = 50pF, Driver Pull-up
t_{su}	I/O Setup Time	1.3			µs	
t_h	I/O Hold Time	0			µs	
t_{ex}	$\overline{\text{EXT INT}}$ Setup Time	400			ns	

Table 4

Notes:

1. Assume Priority In was enabled ($\overline{\text{PRI IN}}$ = 0) in previous F8 cycle before interrupt is detected in the PSU.
2. PSU has interrupt pending before priority in is enabled.
3. Assume pin tied to $\overline{\text{INT REQ}}$ input of the 3850 CPU.
4. The parameters which are shaded in the table above represent those which are most frequently of importance when interfacing to an F8 system. Unshaded parameters are typically those that are relevant only between F8 chips and not normally of concern to the user.
5. Input and output capacitance is 3 to 5pF typical on all pins except V_{DD}, V_{GG}, and V_{SS}.

PSU DATA BUS TIMING

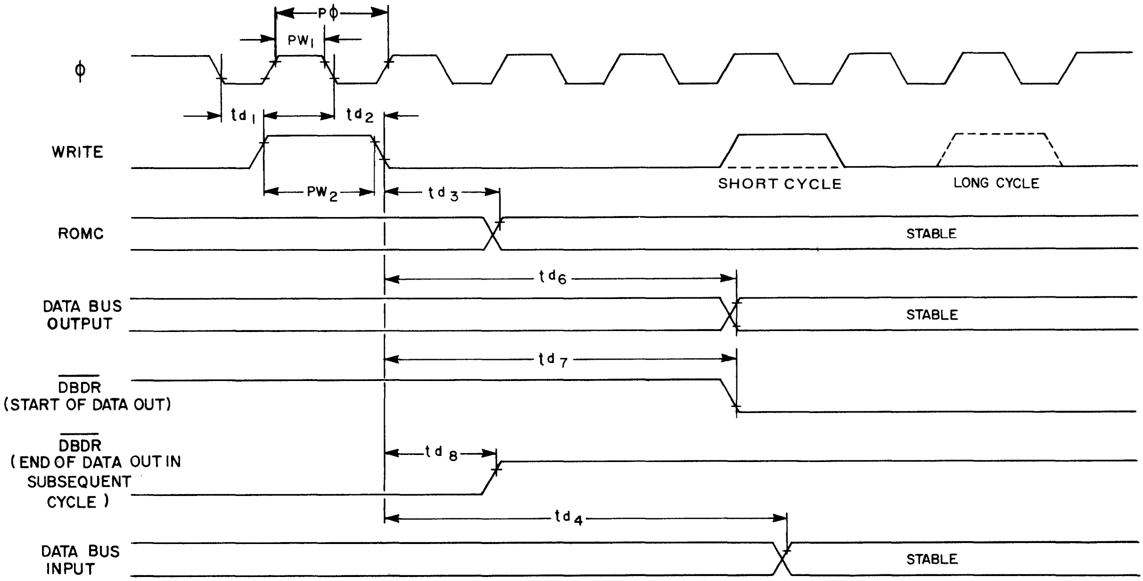
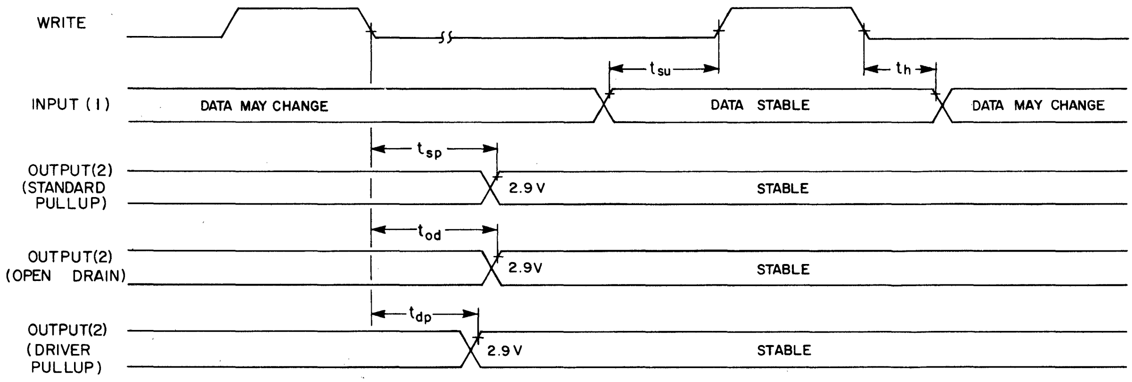


Figure 10

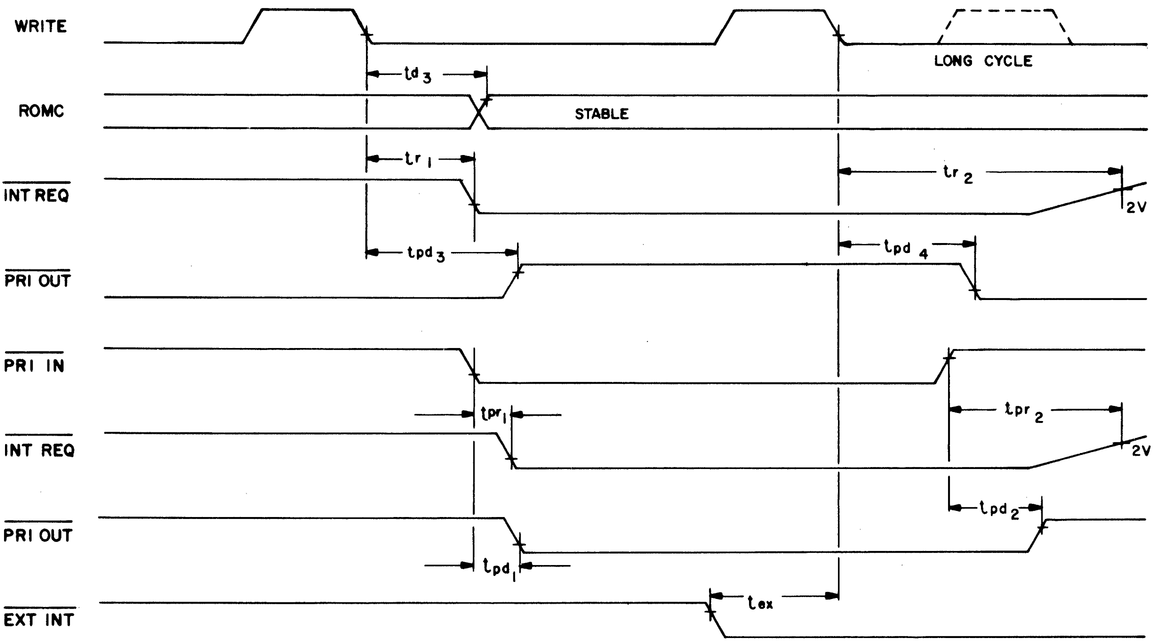
TIMING AT PSU I/O PORTS



1. The set-up and hold times specified are with respect to the end of the second long cycle during execution of the three cycle IN or INS instruction.
2. All delay times are specified with respect to the end of the second long-cycle during execution of the three cycle OUT or OUTS instruction.

Figure 11

INTERRUPT LOGIC SIGNALS TIMING



NOTE: Timing measurements are made at valid logic level of the signals referenced unless otherwise noted.

Figure 12

ORDER INFORMATION

PACKAGE SPECIFICATION

MK 3851N/12XXX	Plastic
MK 3851P/12XXX	Ceramic

The 12XXX number is assigned by MOSTEK when an MK 3851 is ordered. All mask options must also be specified as described in the next section.

OPTION SPECIFICATION

CARD FORMAT USED TO DEFINE MK 3851 PSU MASK OPTIONS

Mask options are specified using a card file which may include the following types of card:

- Option card,
- Comment cards,
- 'X' cards (text format commands), and
- 'C' cards (ROM truth table data).

OPTION CARD FORMAT

The option card must always be the first card in the input data file. The format of the option card follows:

Column	1-20	26-30	35-36	40-42	45	50-53	58-60	63-65
	User	SL	ROM	IO	Port	Timer	HEX DEC	HEX DEC

User is the customer name

SL is a 5-digit SL number for the device assigned by MOSTEK (Leave Blank)

ROM is the ROM number (0-63 decimal) Specifies ROM page

IO is the decimal number (n) of the lowest of the four I/O port addresses selected where: n = 4a, 1 ≤ a ≤ 63

Port is 1 for Standard I/O
2 for Open Drain
3 for Driver Pull-up (Output Only)

Timer is the Timer/External Interrupt Address Vector (4 Hexadecimal digits)

Columns 58-60 specify the desired number base for the address field on the output listing.

Columns 63-65 specify the desired number base for the data fields on the output listing. Each defaults to DECIMAL when not specified. All other fields on the option card must be specified.

COMMENT CARD FORMAT

Each comment card must have an asterisk (*) in column 1. All other columns are ignored. A comment card may occur any time after the option card in the input file. Comment cards are optional.

TEXT FORMAT CARD FORMAT

The text format commands are used to describe the format of the ROM data cards which follow. Text format commands should have the character 'X' in column 1 and should precede all ROM data cards. The valid text format commands are:

X SEQUENCE
indicates that the ROM has sequence numbers in columns 77-79. This command causes F8 ROM to do sequence checking.

X BASE HEX HEX
DEC DEC
specifies the number base of the ROM address input and the ROM data input respectively. If no X BASE card occurs, all fields are assumed to be decimal.

DATA CARD FORMAT

The data cards for F8 PSUs must have the character 'C' in column 1. The ROM truth table data card format is as follows:

Column	1	2-9	10-12	14-16	17-19	20-22	77-79
	C	Add	Bytes	Data 1	Data 2	Data 3. .	Data 22

Add is the ROM address of the first data field on the card

Bytes is the number of bytes of data on the card (< 23) Same number base as address.

Data n specifies the data to be coded at ROM address (Add + n – 1) for 0< n < = Bytes

Data 22 is a sequence number if an X SEQUENCE card has occurred

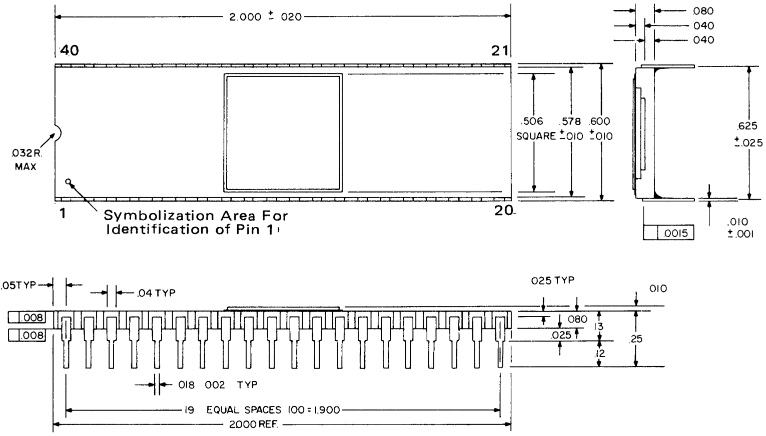
NOTE: All numeric fields must be right justified.

OTHER INPUT METHODS

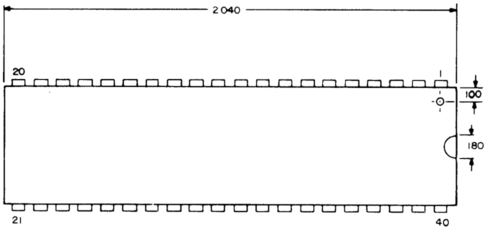
For information concerning other methods of input contact a MOSTEK representative.

PACKAGE DESCRIPTION

40-Pin Dual In-Line Ceramic

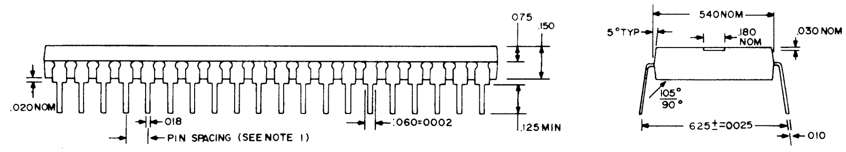


40-Pin Dual In-Line Plastic



NOTES:

1. The true position pin spacing is 0.100 between centerlines. Each pin centerline is located within ± 0010 of its true longitudinal position relative to pins 1 and 40.



MOSTEK®

F8 MICROCOMPUTER DEVICES

Dynamic Memory Interface MK 3852

FEATURES

□ Provides interface for 64K of dyanmic or static RAM

□ Interfaces with MK3854 for DMA channel

□ Provides automatic refresh for dynamic RAMs.

□ Low Power Dissipation Typically Less Than 335mW

GENERAL DESCRIPTION

The 3852 DMI provides all interface logic needed to include up to 64K bytes of dynamic or static RAM memory in an F8 microcomputer system. In response to control signals output by the 3850 CPU, the 3852 DMI generates address and control signals needed by standard static and dynamic RAM devices. The MK3852 DMI is manufactured using N-channel Isoplanar MOS technology.

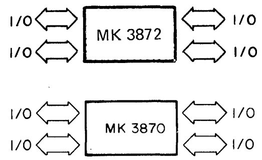
FUNCTIONAL PIN DESCRIPTION

Φ and WRITE are clock outputs from the 3850 CPU.

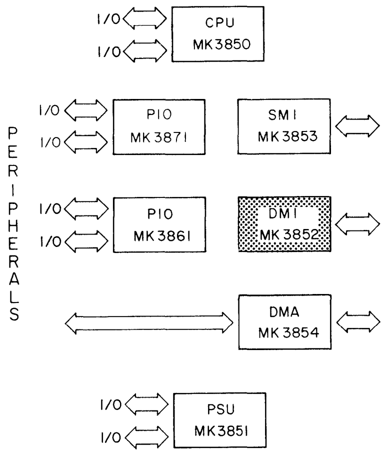
ROMCO through ROMC4 are the control signals output by the 3850 CPU.

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional(3-State)
ADDR0-ADDR15	Address Lines	Output (3-State)
Φ, WRITE	Clock LInes	Input
MEMIDLE	DMA Timing Line	Output
CYCLE REQ	RAM Timing Line	Output
CPU Slot	Timing Line	Input/Output
CPU READ	RAM Timing Line	Output
REGDR	Register Drive Line	Input/Output
RAM WRITE	Write Line	Output (3-State)
ROMC0-ROMC4	Control Lines	Input
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input

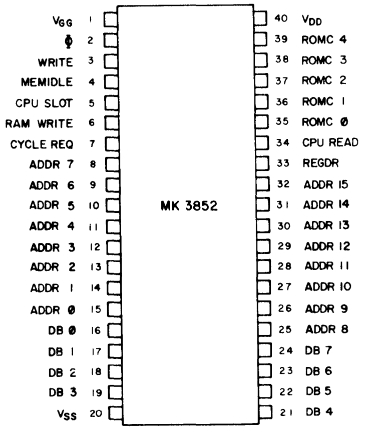
SINGLE CHIP 3870 MICROCOMPUTER FAMILY



F8 FAMILY



PIN CONNECTIONS



ADDR0 through ADDR15 are 16 address lines via which an address is transmitted to dyanmic RAM. The address may come from P0 or DC registers.

DB0 through DB7 are the bi-directional data bus lines which link the 3852 DMI with all other devices in the F8 system. Only data moving to or from 3852 DMI address registers and I/O ports use the 3852 DMI DB0-DB7 pins.

MEM IDLE high identifies portions of an instruction execution cycle during which the F8 system is not accessing memory, to read, write or refresh. MEM IDLE high therefore identifies the portion of an instruction cycle which is available for DMA operations. The 3852 DMI can inhibit DMA by holding MEM IDLE constantly low. The address drivers and $\overline{\text{RAM WRITE}}$ driver are always in a high impedance state when MEM IDLE is high, so that a DMA device may drive the address lines at this time.

RAM WRITE. When low, this signal specifies that data is to be written into RAM locations. When high, this signal is off; that is, $\overline{\text{RAM WRITE}}$ high does not necessarily specify a read operation.

CPU READ. When high, this signal specifies that data is to be read out of a RAM location. When low, this signal is off; that is, CPU READ low does not specify a write operation; that is done by $\overline{\text{RAM WRITE}}$ low.

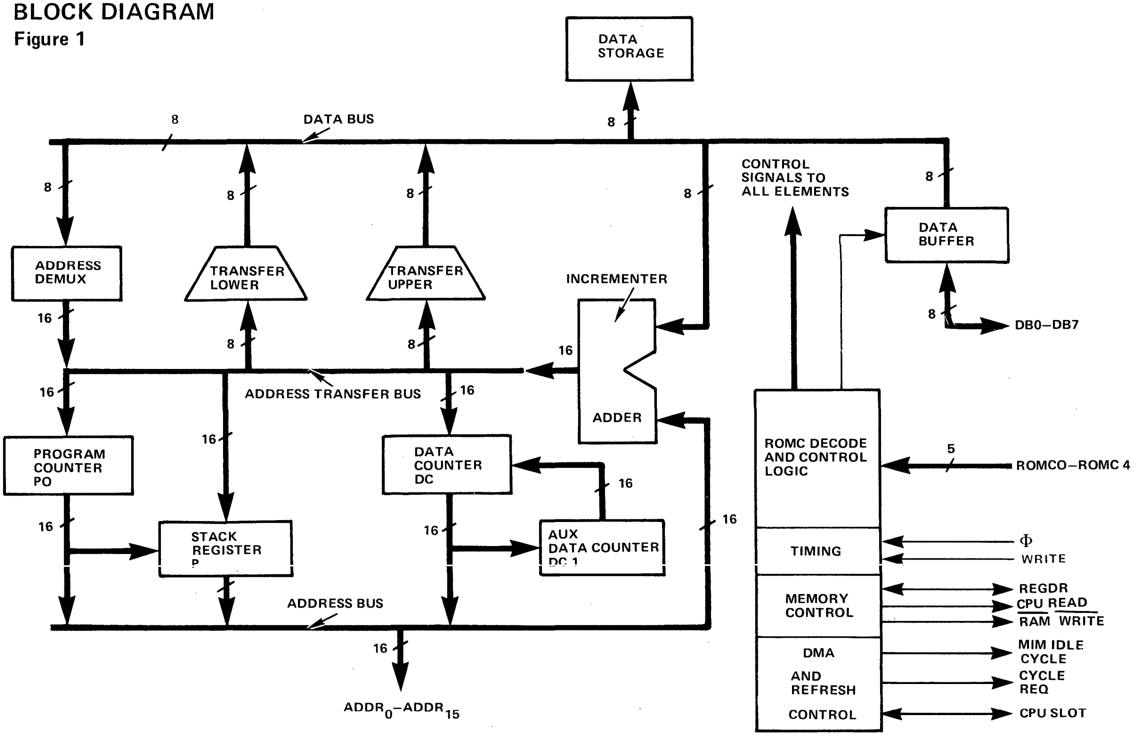
REGDR. This signal functions both as an input and an output. As an input, it can be clamped low by an external open collector gate. This prevents the 3852 DMI from placing a byte out of its P or DC registers onto the data bus. The DMI internally supplies a pull-up resistor. The signal, functioning as an output, can control data bus buffers. The DMI will internally clamp REGDR low except during those ROMC states during which the DMI is required to place a byte out of P or DC registers or either of its two control registers (I/O ports) onto the data bus.

CYCLE REQ. There may be either two or three memory access periods within one instruction cycle. CYCLE REQ identifies each memory access period by making a high to low transition at the start of the memory access period. CYCLE REQ does not identify events which are to occur during the memory access period. CYCL REQ is a divide-by-2 of Φ during all ROMC states except ROMC state 05 (store in memory); it can be used to generate the clock signals required by many dynamic RAMs.

CPU SLOT high identifies portions of an instruction execution cycle during which the 3850 CPU is reading data out of RAM, or writing data into RAM. CPU SLOT is a bi-directional signal. If held low by external logic, it causes the address line drivers and $\overline{\text{RAM WRITE}}$ driver to be held in a high impedance state.

BLOCK DIAGRAM

Figure 1



DEVICE ORGANIZATION

This section describes the basic functional elements of the MK3852 DMI. These elements are shown in the DMI functional block diagram (figure 1).

PROGRAM COUNTER (PO) AND DATA COUNTER (DC AND DC1)

The MK3852 DMI addressing logic consists of 3 16-bit registers, the Program Counter (PO) and the Data Counters (DC and DC1).

The Program Counter will at all times address the memory word from which the next object program code must be fetched. The Data Counter (DC) addresses memory words containing individual data bytes or bytes within data tables to be used as operands.

It is important to note that the 3852 DMI has an auxiliary Data Counter (DC1). The contents of DC can be saved in DC1 by using the instruction XDC (exchange data counters). This instruction puts the contents of DC into DC1 and the contents of DC1 into DC. DC⇄DC1.

PO will always address the memory location out of which the next object program instruction byte will be read. If the instruction requires data (an operand) other than an immediate to be accessed DC must address memory. PO cannot be used to address a NON-immediate operand since PO is saving the address of the next instruction code.

THE STACK REGISTER P

The MK3852 DMI addressing logic contains a fourth 16-bit register called the stack register (P). The stack register is a buffer for the program counter PO. The contents of the stack register are never used directly to address memory.

The following instructions access P

LR K, P	Move the contents of P to the CPU scratchpad K registers
LR P,K	Move the contents of the CPU K scratchpad registers to P
PK	Save the contents of PO in P then move the contents of CPU scratchpad registers 12 and 13 to PO
PI H'aaaa'	Move the contents of PO to P then load the hexadecimal value into PO
POP	Move the contents of P to PO

In addition, when an interrupt is acknowledged, the contents of PO are saved in P.

MEMORY CONTROLS

Memory Control logic generates appropriate timing and control signals needed by RAM to input or output data. Timing and control signals are generated in response to ROMC states, as decoded by the Control Unit.

INCREMENTER ADDER LOGIC

There are only two arithmetic operations that memory devices need to perform on the contents of memory address registers:

1. Increment by 1 the 16-bit value stored in an address register.

2. Add an 8-bit value, treated as a signed binary number (subject to twos complemented arithmetic) to the 16-bit value stored in address register.

The incrementer adder logic performs these two functions in the MK3852 DMI.

THE DATA BUS

The 8-bit data bus is the main path for transfer of information between the MK3850 CPU and other devices in the F8 microprocessor system.

ADDRESSABLE I/O PORTS

The 3852 DMI has four I/O port addresses reserved for its use. There are two versions of the 3852 DMI; one has I/O port addresses 0C, 0D, 0E and 0F for its four I/O ports; since these addresses are also used by the 3853 SMI, another version of the 3852 DMI uses I/O port address EC, ED, EE and EF. This allows an F8 microcomputer system to include both a 3852 DMI and a 3853 SMI.

I/O port addresses 0E and 0F (or EE and EF), though reserved for the 3852 DMI, are not used. Port 0C (or EC) is a general purpose, 8-bit data storage buffer which can be loaded with the OUT or OUTS instruction and read using the IN or INS instruction. Port 0D (or ED) is a control register which controls memory refresh and DMA operations.

DMA AND REFRESH CONTROL

Because of the organization of the F8 microcomputer system, there is a period within every instruction execution cycle when the CPU is not accessing memory.

DMA and Refresh Control logic generates timing and control signals that identify time periods when the CPU is not accessing memory; during these time periods memory is refreshed, or DMA data accesses occur.

OPERATIONAL DESCRIPTION

CLOCK TIMING

All timing within the MK3852 DMI is controlled by Φ and WRITE, which are input from the MK3850 CPU. Each machine cycle will contain either 4 Φ clock periods (short cycle) or 6 Φ clock periods (long cycle).

The WRITE clock refreshes and updates the MK3852 DMI. A machine cycle begins with the fall of the WRITE clock and the system control lines become stable shortly after the start of the cycle.

INSTRUCTION EXECUTION

The MK3852 DMI responds to signals which are output by the MK3850 CPU in the course of executing instruction cycles.

Table 1 summarizes the response of the MK3852 DMI to the ROMC states.

Table 1

ROMC STATES ROMC (Hexadecimal)	OPERATION PERFORMED	COMMENT
00	DB←((P0)); P0←P0 + 1	OP CODE, FETCH
01	DB←((P0)); P0←P0 + DB	BRANCH OFFSET FETCH
02	DB←((DC)); DC←DC + 1	
03	DB←((P0)); P0←P0 + 1	IMMEDIATE OPERAND FETCH
04	P0←P	
05	((DC))←DB; DC←DC + 1	
06	DB←DCU	
07	DB←PU	
08	P←P0; DB←H'00'; P0L, P0H←DB	EXTERNAL RESET
09	DB←DCL	
0A	DC←DC + DB	
0B	DB←PL	
0C	DB←((P0)); DCL←DB	
0D	P←P0 + 1	
0E	DB←((P0)); DCL←DB	
0F	NO OPERATION	
10	NO OPERATION	
11	DB←((P0)); DCU←DB	
12	P0L←DB; P←P0	
13	NO OPERATION	
14	P0U←DB	
15	PU←DB	
16	DCU←DB	
17	P0L←DB	
18	PL←DB	
19	DCL←DB	
1A	((pp))←DB or ((p))←DB	
1B	DB←(pp) or DB←((p))	
1C	NO OPERATION	
1D	DC⇄DC1	
1E	DB←P0L	
1F	DB←P0U	

Definitions:

DB	- Data Bus
P0	- Program Counter
DC	- Data Counter
DC1	- Aux Data Counter
P	- Stack Register
pp	- Two hex digits (long I/O port address)
p	- One hex digit (short I/O port address)
1A	- Interrupt address vector
L	- Lower byte suffix
U	- Upper byte suffix
( )	- Contents of
←	- transfer to
⇄	- exchange

MEMORY ADDRESSING

Any dynamic RAM which is controlled by the 3852 DMI will have a PAGE SELECT input, which must be true if the memory is to respond to read or write control signal sequences.

PAGE SELECT true is created by logic external to the 3852 DMI, and defines the dynamic RAM address space. PAGE SELECT true can be generated in any way; there are no special rules.

For example, consider an F8 system with 1K bytes of ROM on a 3851 PSU and 4K bytes of dynamic RAM controlled by a 3852 DMI; address ranges will be as follows:

1K bytes of ROM	0000_{16} to $03FF_{16}$
4K bytes of RAM	0400_{16} to $13FF_{16}$

In binary format, the dynamic RAM address space is defined by:

	ADDR15			ADDR12	ADDR13	ADDR10										ADDR0
Minimum:	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Maximum:	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1

PAGE SELECT may be the OR of ADDR 12, ADDR11 and ADDR10, which are shown above.

Depending on the way in which dynamic RAM is being used, PAGE SELECT may be a simple memory enable signal, or it may be ANDed with CPU READ and $\overline{\text{RAM WRITE}}$, to generate local versions of these two signals which are locally true only.

3852 DMI ADDRESS REGISTERS' ADDRESS SPACE

As described in Table 1, certain ROMC states require the contents of the high order, or low order half of P0, P, or DC to be placed on the data bus. If there is more than one memory device in an F8 system, only one device must respond to these ROMC states.

The 3851 PSU uses its address select mask to determine if it is to place address register contents on the data bus; for the 3851 PSU, therefore, the memory and address registers' address spaces must be identical.

The 3852 DMI address registers' address space is identified by the REGDR signal; if this signal is not clamped low, the 3852 will place data on the data bus in response to ROMC states that require data from P0, P or DC to be placed on the data bus. If REGDR is derived from the PAGE SELECT signal, then the RAM memory and the 3852 DMI address registers' address spaces will coincide.

On the other hand, it is a good idea to make the 3852 DMI address registers' address space cover all addresses that are not part of another memory device's address registers' address space. For example, the following address spaces would be desirable:

ADDRESS SPACES

	MEMORY	ADDRESS REGISTERS
3851 PSU	0000_{16}-$03FF_{16}$	0000_{16}-$03FF_{16}$*
3852 DMI	0400_{16}-$13FF_{16}$	0400_{16}-$FFFF_{16}$

*For the 3851 PSU, the two address spaces must be identical.

If the address space for the address registers covers all possible memory addresses, then instructions that read data out of address registers will always generate a valid response.

In the above illustration, if memory and address registers' address spaces coincided for the 3852, then in response to instructions that require data to be output from P0, P, or DC, no device would respond when the selected address register contains a value in excess of $13FF_{16}$; as a result, invalid values would be received by the 3850 CPU.

ADDRESS CONTENTIONS

When a 3852 DMI is present in an F8 system, memory addressing contentions are resolved as described in Memory Addressing, with one exception: the 3852 DMI has a DC1 register and the 3851 PSU does not.

The XDC instruction (ROMC state 1D) causes the contents of the DC0 and DC1 registers to be exchanged; having no DC1 register, the 3851 PSU does not respond to this ROMC state, therefore 3851 PSU and 3852 DMI devices can have different values in their DC registers, and each value can be within the different address spaces of the two memory devices. An instruction that requires data to be output from DC may now cause two devices to simultaneously place different data on the data bus. This may be illustrated as follows:

	PSU	DMI
Before XDC:	DC =XXXX	XXXX
	DC1=	YYYY
After XDC:	DC =XXXX	YYYY
	DC1	XXXX

If XXXX happens to be in a PSU's address space while YYYY is in the DMI address space, then address contentions will arise.

Even if XXXX is not in the PSU's address space, address contentions may arise due to the fact that memory reference instructions will increment different DC contents. Suppose two memory reference instructions are executed following one XDC, then another XDC is executed; this is what happens:

		PSU →	DMI
After first XDC:	DC =	XXXX	YYYY
	DC1 =		XXXX
After two memory reference instructions:	DC =	XXXX+2	YYYY+2
	DC1 =		XXXX
After second XDC:	DC =	XXXX+2	XXXX
	DC1 =		YYYY+2

An address contention may arise if DC contents approaches the boundary of the PSU address space. For example, if the address space boundary occurs at XXXX+1, the PSU and the DMI will both consider themselves selected.

The following coding instruction sequence shows how to use the DC instruction without encountering address contentions. The example allows use of a second address value YYYY, which is held in DC1, while using the H register to temporarily hold the first address value, XXXX. Address YYYY, which at the beginning of the example is held in DC1, must be in the DMI address space. The address XXXX may be in any address space.

INSTRUCTION	PSU	DMI	
	DC0	DC0	DC1
	XXXX	XXXX	YYYY
LR H,DC			
DCI ZZZZ	ZZZZ	ZZZZ	YYYY
XDC	ZZZZ	YYYY	ZZZZ
—			
—			
—			
Other Instructions			
—			
—			
—	ZZZZ+N	YYYY+N	ZZZZ
XDC	ZZZZ+N	ZZZZ	YYYY+N
LR DC,H	XXXX	XXXX	YYYY+N

For the above scheme to work, it is only necessary for ZZZZ through ZZZZ+N to be outside any PSU's address space.

If the value XXXX through XXXX+N is outside of any PSU's address space, then the DCI ZZZZ instruction may be omitted.

In many cases, it will not be necessary to restore the XXXX value; then the LR H,DC and LR DC, H instructions can also be omitted—letting a subsequent DC loading instruction synchronize the DC's.

Before a value held in DC1 can be used, it must first have been loaded into DC1. The XDC instruction is used to load DC1. Consider the following instruction sequence:

INSTRUCTION		PSU	DMI	
		DC	DC	P
		XXXX	XXXX	WWWW
DCI	YYYY	YYYY	YYYY	
XDC		YYYY	WWWW	YYYY
DCI	ZZZZ	ZZZZ	ZZZZ	YYYY

YYYY lies in the address space of the DMI, ZZZZ lies anywhere, XXXX and WWWW are arbitrary initial values. The DCI instructions could just as well be LR DC, H or LR DC, Q.

The exchange of DC and DC1 becomes most powerful when a series of swaps are used to add two blocks of memory, or to move data from one block to a second. The XDC instruction can be used to do this so long as neither block is in a PSU's address space. Notice that the DC of the PSU is out of step throughout the example.

INSTRUCTION		PSU	DMI	
		P0	DC	DC1
		XXXX	XXXX	YYYY
DCI	ZZZZ	ZZZZ	ZZZZ	YYYY
LM		ZZZZ+1	ZZZZ+1	YYYY
XDC		ZZZZ+1	YYYY	ZZZZ+1
ST		ZZZZ+2	YYYY+1	ZZZZ+1
XDC		ZZZZ+2	ZZZZ+ 1	YYYY+1
Other Instructions				
LM		ZZZZ+ΔZ+ΔY-1	ZZZZ+ΔZ	YYYY+ΔY
XDC		ZZZZ+ΔZ+ΔY-1	YYYY+ΔY-1	ZZZZ+ΔZ
ST		ZZZZ+ΔZ+ΔY	YYYY+ΔY	ZZZZ+ΔZ
DCI	WWWW	WWWW	WWWW	ZZZZ+ΔZ

In the above example ZZZZ and YYYY both lie in the address of a DMI. The space spanned by ZZZZ to ZZZZ + ΔZ + ΔY must be outside of any PSU's address space.

TIMING SIGNALS OUTPUT BY A 3852 DMI

Within an instruction cycle, there may be either two or three memory access periods, depending on whether the instruction cycle is long or short. A memory access period is equivalent to two Φ clock periods, and is identified by CYCLE REQ, which is a divide-by-two of Φ. Whether the instruction cycle is short, or long, depends on the source and destination of the data being transmitted during instruction execution.

During the first memory access period, the 3852 DMI outputs the contents of P0 on the address lines of ADDR0-ADDR15.

In effect, 3852 DMI logic beings by assuming that a memory read is to occur, with the memory address provided by P0.

While the contents of P0 are being output on the address lines, the 3852 DMI control unit, in parallel, decodes the ROMC state which has been received from the 3850 CPU.

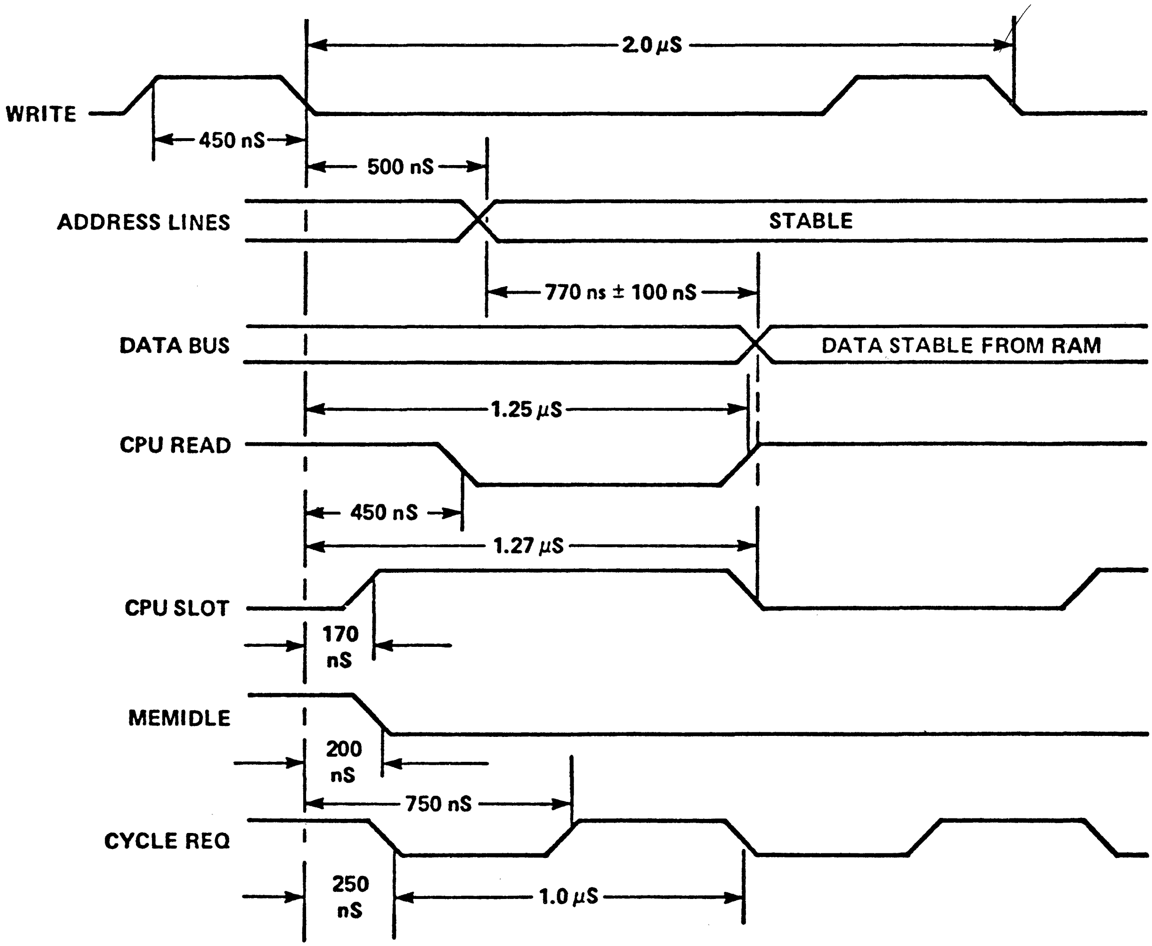
If the assumed logic proves to be correct, or if no memory access is to occur, then the second access period can be used for memory refresh or DMA.

If the instruction, once decoded by the CPU, specifies a memory read with another memory address, then the 3852 DMI wastes the first access period. The instruction cycle will always be long in this case. During the second access period, the required memory access is performed, while memory refresh occurs, or DMA is implemented in the third access period.

If a memory write instruction is decoded, then no access periods are available for memory refresh or DMA.

Four variations of the instruction cycle result. The timing diagrams illustrating the four variations represent worst cases, and assume td_2 = 150ns. These are the four variations:

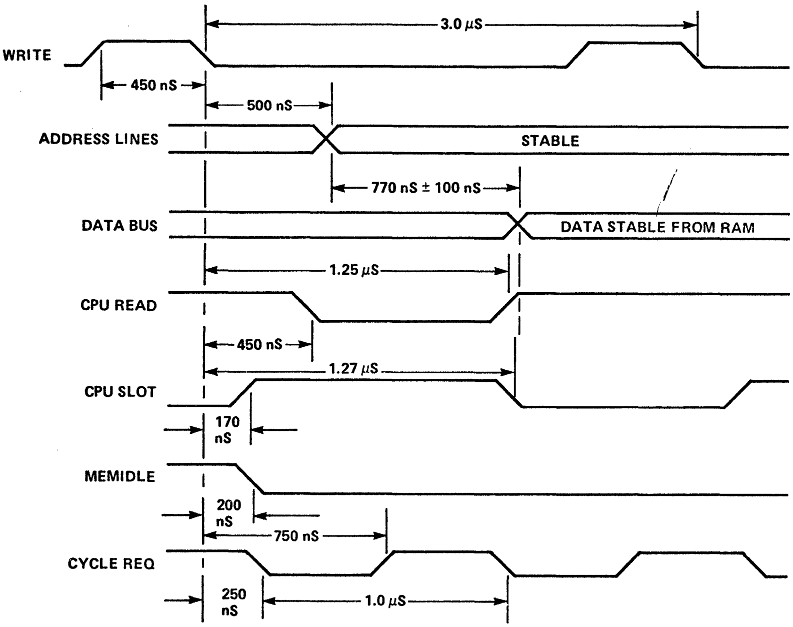
3852 DMI TIMING SIGNALS OUTPUT DURING A SHORT CYCLE MEMORY READ WITH ADDRESS FROM PO
Figure 2



1. The instruction fetch. The memory address originates in PO and the instruction cycle is short. Timing is shown in Figure 2.

3852 DMI TIMING SIGNALS OUTPUT DURING A LONG CYCLE MEMORY READ, WITH ADDRESS OUT OF PROGRAM COUNTER

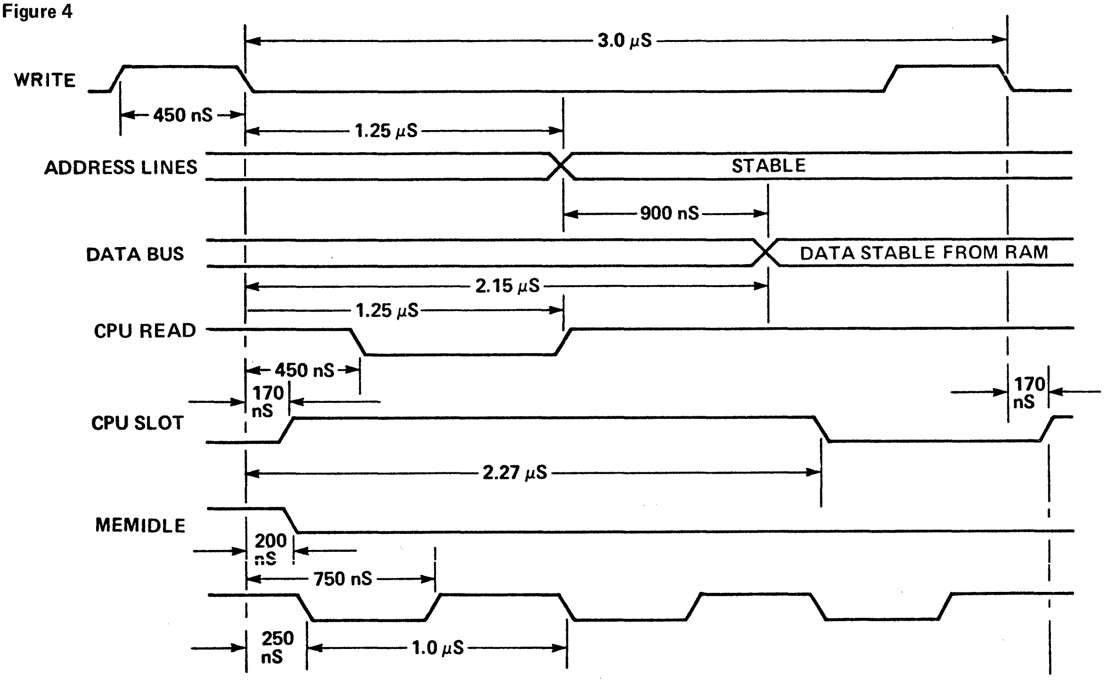
Figure 3



2. An immediate operand fetch. The memory address originates in P0, and the instruction cycle is long. Timing is shown in Figure 3.

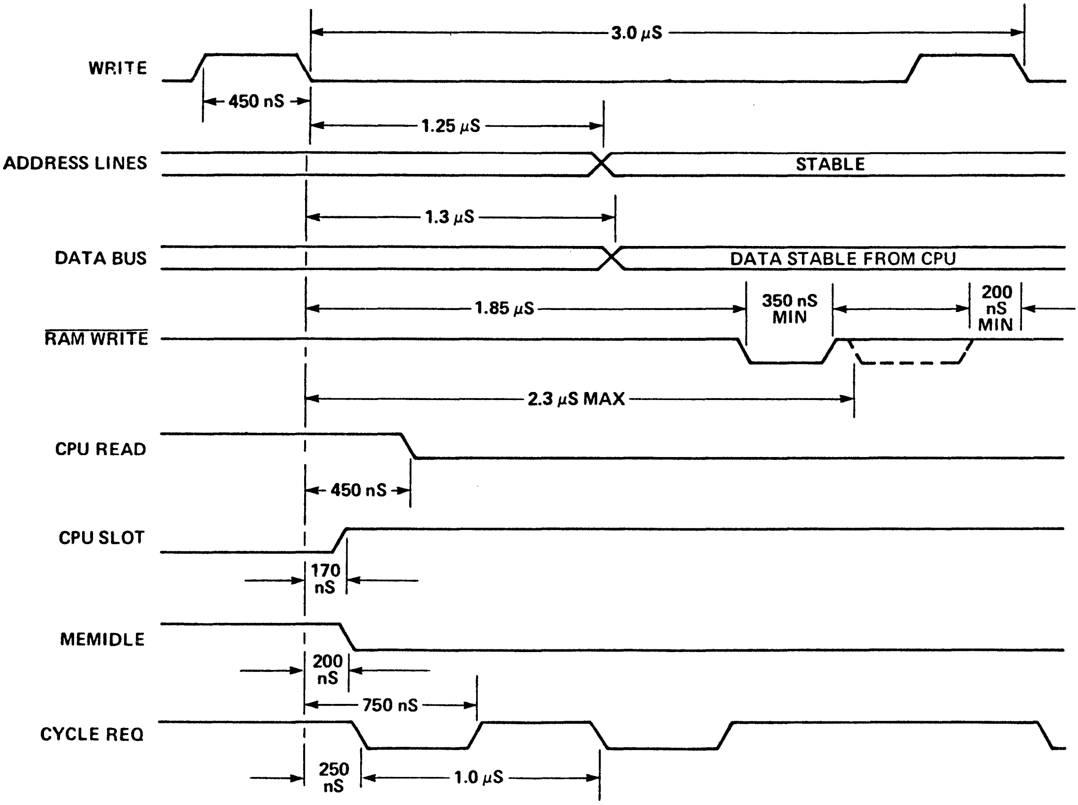
3852 DMI TIMING SIGNALS OUTPUT DURING A LONG CYCLE MEMORY READ, WITH ADDRESS OUT OF DATA COUNTER

Figure 4



3. A data fetch. A data byte is output from an address register, or the memory address originates in DC, therefore the instruction cycle is long. Timing is shown in Figure 4.

3852 DMI TIMING SIGNALS OUTPUT DURING A WRITE TO MEMORY
Figure 5



4. A memory write. Data is written into the RAM memory location addressed by DC. Timing is shown in Figure 5.

CPU SLOT and MEM IDLE identify the way in which a memory access period is being used. Figures 6 and 7 illustrate the relationship.

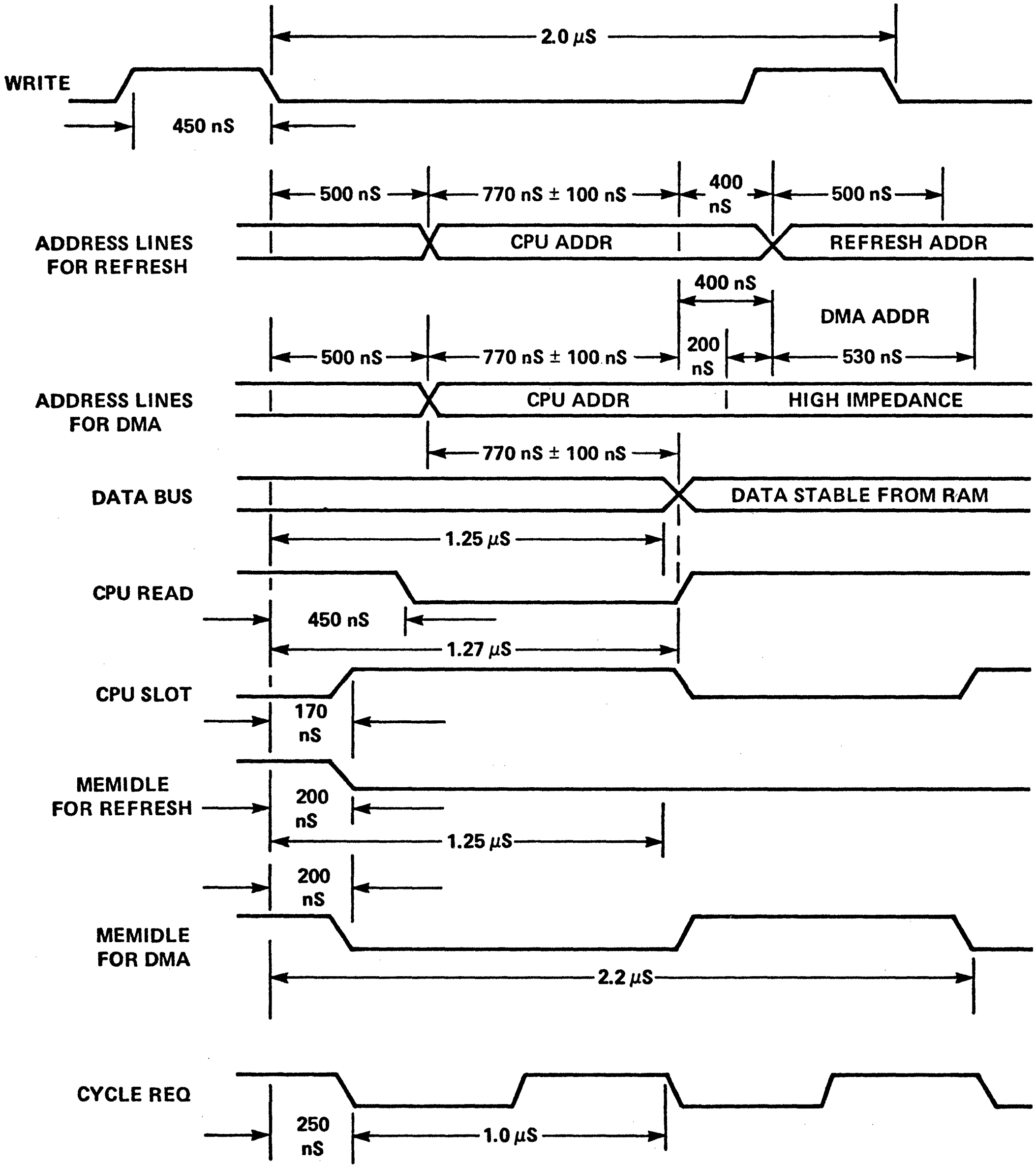
When the 3850 CPU is accessing memory, CPU SLOT is high; RAM WRITE and the address lines are driven at this time.

When memory is available for DMA access, CPU SLOT is low, and MEM IDLE is high.

When the 3852 DMI is refreshing dynamic memory CPU SLOT and MEM IDLE are both low.

3852 DMI logic is able to achieve two memory accesses within one instruction cycle by pursuing the logic sequence summarized in Table 2. Buffer/latches are placed on the F8 data bus lines between the RAM and the F8 system to hold the data fetched during the first access.

TIMING FOR MEMORY REFRESH AND DMA DURING A SHORT CYCLE MEMORY READ, WITH ADDRESS OUT OF PROGRAM COUNTER
Figure 6



TIMING FOR MEMORY REFRESH AND DMA DURING A LONG CYCLE MEMORY READ, WITH ADDRESS OUT OF PROGRAM COUNTER

Figure 7

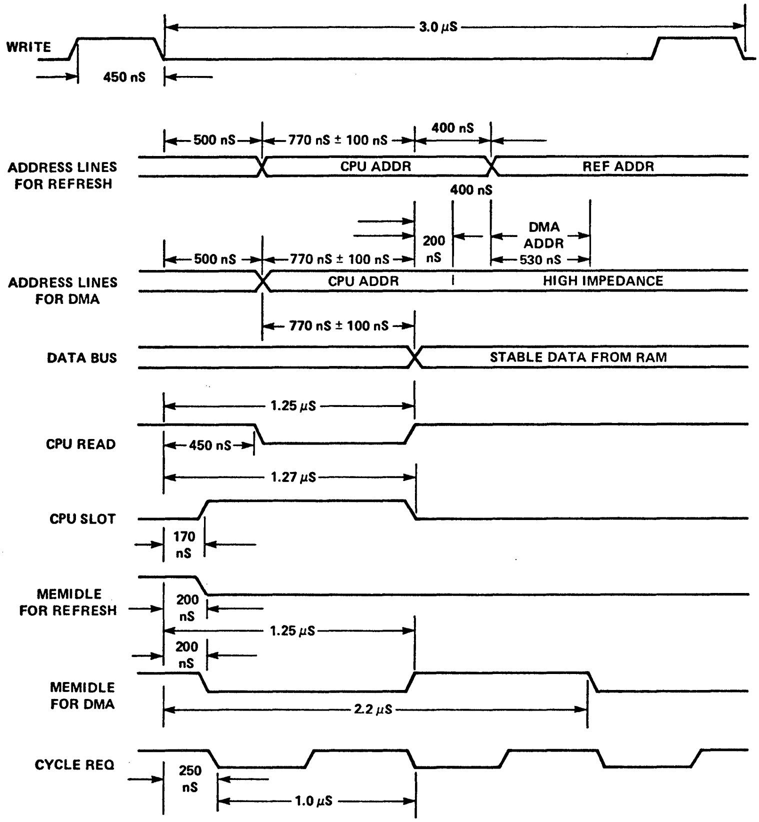


Table 2

OPERATION PERFORMED DURING INSTRUCTION CYCLE	FIRST ACCESS	SECOND ACCESS	THIRD ACCESS
No memory access, or read from memory addressed by P0. (See Figure 2.)	[P0]→A0 - A15	Latch data on F8 data bus. Second memory access for DMA or refresh.	None
No memory access, or read from memory addressed by P0. (See Figure 3.)	[P0]→A0 - A15	Latch data on F8 data bus. Second memory access for DMA or refresh.	Third memory access not used.
Read data from memory addressed by register other than P0, or read data from address register. (See Figure 4.)	[P0]→A0 - A15	[Other register]→A0 -15	Latch data on F8 data bus. Third memory access for DMA or refresh
Write data to memory. (See Figure 5.)	[P0]→A0 - A15	[DC]→A0 - A15	Access memory to write data. No DMA or refresh.

[ ] means "contents of register identified within square brackets.

MEMORY REFRESH AND DIRECT MEMORY ACCESS

These two topics are covered together, since in terms of 3852 DMI logic, they are similar operations.

CYCLE REQ identified 2 or 3 memory access periods witnin an instruction cycle.

Either the first or the second access period, as summarized in Table 2, is reserved for the instruction cycle being decoded. Let us refer to this as the "reserved" access period. If the ROMC state for the instruction cycle requires data to be read out of RAM, then the read occurs during the reserved access period. If the ROMC state for the instruction cycle requires data to be input to an address register, or if no data movement occurs on the data bus, then the reserved access period is not used for any memory access—it is, in effect, wasted.

One more memory access may occur within the instruction cycle; this occurs during either the second or third access period, as summarized in Table 2, while the data bus latches hold data accessed during the first period. We will refer to this as the "free" access period.

Some available free access periods must be used to refresh dynamic RAM. A refresh uses logic within the 3852 DMI. Therefore a refresh occurs in parallel to anything else that is going on.

If the free access period is not used to refresh dynamic RAM, it may be used by a 3854 DMA device to perform direct memory accesses. The DMA uses a separate data channel to access memory, so DMA can occur in parallel with anything else that is going on within the F8 system.

DATA OUTPUT BY RAM

Figures 2, 3, and 4 provide worst case timing when RAM, controlled by the 3852 DMI, outputs data onto the data bus. In these figures it is assumed that CPU SLOT is used to strobe the RAM data into the data bus latches.

CPU READ is output high by the 3852 DMI to enable transfer of data from the data bus buffers to the data bus. Recall that dynamic RAM have its own connection to the data bus via buffer/latches; data is not transferred via the 3852 DMI.

Observe that CPU READ high is similar to $\overline{\text{DBDR}}$ low—each is active when its respective data bus drivers are turned on.

DATA OUTPUT BY THE 3852 DMI

REGDR defines the address space of the address registers within the 3852 DMI.

If a ROMC state received by the 3852 DMI requires data to be output from an address register, then the 3852 DMI will become the selected data source if REGDR is allowed to go high.

DATA INPUT TO RAM

Figure 5 provides worst case timing when data is written into RAM. Data is transferred through tri-state buffers on the data bus and into RAM.

$\overline{\text{RAM WRITE}}$ is pulsed low by the 3852 DMI to enable the transfer of data off the data bus, into RAM. The tri-state buffers or multiplexers between data bus and RAM WRITE data lines are necessary if DMA sources are also allowed to write into RAM.

DATA INPUT TO THE 3852 DMI

Problems of addressing contention are posed by having duplicated address registers; one step in resolving this possible problem is to force every memory device to read onto its address registers whenever a ROMC state specifies any such operation. Address space concepts therefore do not apply when data is read into 3852 DMI address registers.

INPUT/OUTPUT

There are two versions of the 3852 DMI; each has four reserved I/O port addresses, implemented as follows:

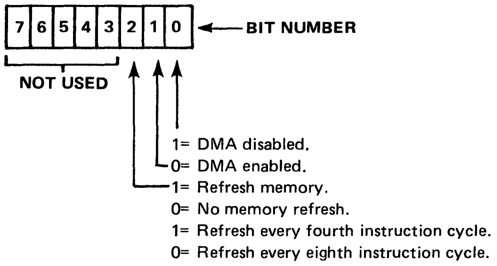
PORT ADDRESSES		FUNCTION
STANDARD	OPTION	
0C	EC	General purpose latch
0D	ED	Memory/DMA control
0E	EE	Not implemented
0F	EF	Not implemented

Option port addresses are used in F8 systems that include both a 3852 DMI and a 3853 SMI.

The implemented I/O ports are accessed via IN, INS, OUT and OUTS instructions, just like any other I/O port. However, the 3852 DMI I/O ports are internal latches, having no connection to I/O pins or external interface. REGDR, if not clamped low by an external device, will go high during IN or INS instructions that select either of the DMI ports. However, clamping REGDR low does not inhibit data bus driving during I/O as it did during the output of address registers.

I/O port 0C (or EC) is used as a general purpose, 8-bit data storage location.

I/O port 0D (or ED) controls memory refresh and DMA as follows:



SYSTEM INITIALIZATION

An F8 system is initialized by power on, or EXT RESET being pulsed low at the CPU.

When an F8 system is initialized, DMA is turned off and memory refresh is on, with refresh every fourth cycle selected.

Contents of all other registers are indeterminate; reading the control port 0D (or ED) also gives indeterminate results, although the DMA/refresh state of the DMI has been initialized.

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATING (All voltages with respect to V_{SS})*

V_{GG}	+15V to −0.3V
V_{DD}	+7V to −0.3V
All other inputs and outputs	+7V to −0.3V
Operating temperature, T_A (Ambient)	0°C to +70°C
Storage temperature - Ambient (Ceramic)	−65°C to +150°C
Storage temperature - Ambient (Plastic)	−55°C to +125°C

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

($0°C \leq T_A \leq 70°C$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{DD}	Supply	4.75	5.0	5.25	Volts	
V_{GG}	Voltage	11.4	12.0	12.6	Volts	
V_{SS}		0	0	0	Volts	

DC ELECTRICAL CHARACTERISTICS

($0°C \leq T_A \leq 70°C$) V_{DD} = +5V ± 5%; V_{GG} = +12V ± 5%; V_{SS} = 0V

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		35	70	mA	f=2MHz, Outputs unloaded
I_{GG}	I_{GG} Current		13	30	mA	f=2MHz, Outputs unloaded

DATA BUS (DB0-DB7)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	$I_{OH} = -100\mu A$
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	I_{OL} = 1.6mA
I_{IH}	Input High Current		1	µA	$V_{IN} = V_{DD}$, three-state mode
I_{IL}	Input Low Current		−1	µA	$V_{IN} = V_{SS}$, three-state mode
C_I	Capacitance		10	pF	Three-state mode

CONTROL LINES (ROMC0–ROMC4), AND CLOCK LINES (Φ, WRITE)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$
C_I	Capacitance		10	pF	

ADDRESS LINES (ADDR0-ADDR15) AND $\overline{\text{RAM}}$ $\overline{\text{WRITE}}$

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	4.0	V_{DD}	Volts	$I_{OH} = -1mA$
V_{OL}	Output Low Voltage		.4	Volts	$I_{OL} = 3.2mA$
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$

REGDR AND CPU SLOT

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	$I_{OH} = -300\mu A$
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	$I_{OL} = 2mA$
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	Internal Pull-up to V_{DD}
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
I_{IL}	Input Low Current	−3.5	−14.0	mA	V_{IN} = .4V and device outputting a logic "1"
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$

CPU READ, MEMIDLE, AND CYCLE REQ

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	$I_{OH}=-1mA$
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	$I_{OL} = 2mA$
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$

AC ELECTRICAL CHARACTERISTICS

($0°C \leqslant T_A \leqslant 70°C$) ($V_{DD}$ = +5V ± 5%; V_{GG} = +12V ± 5%; V_{SS} = 0V)

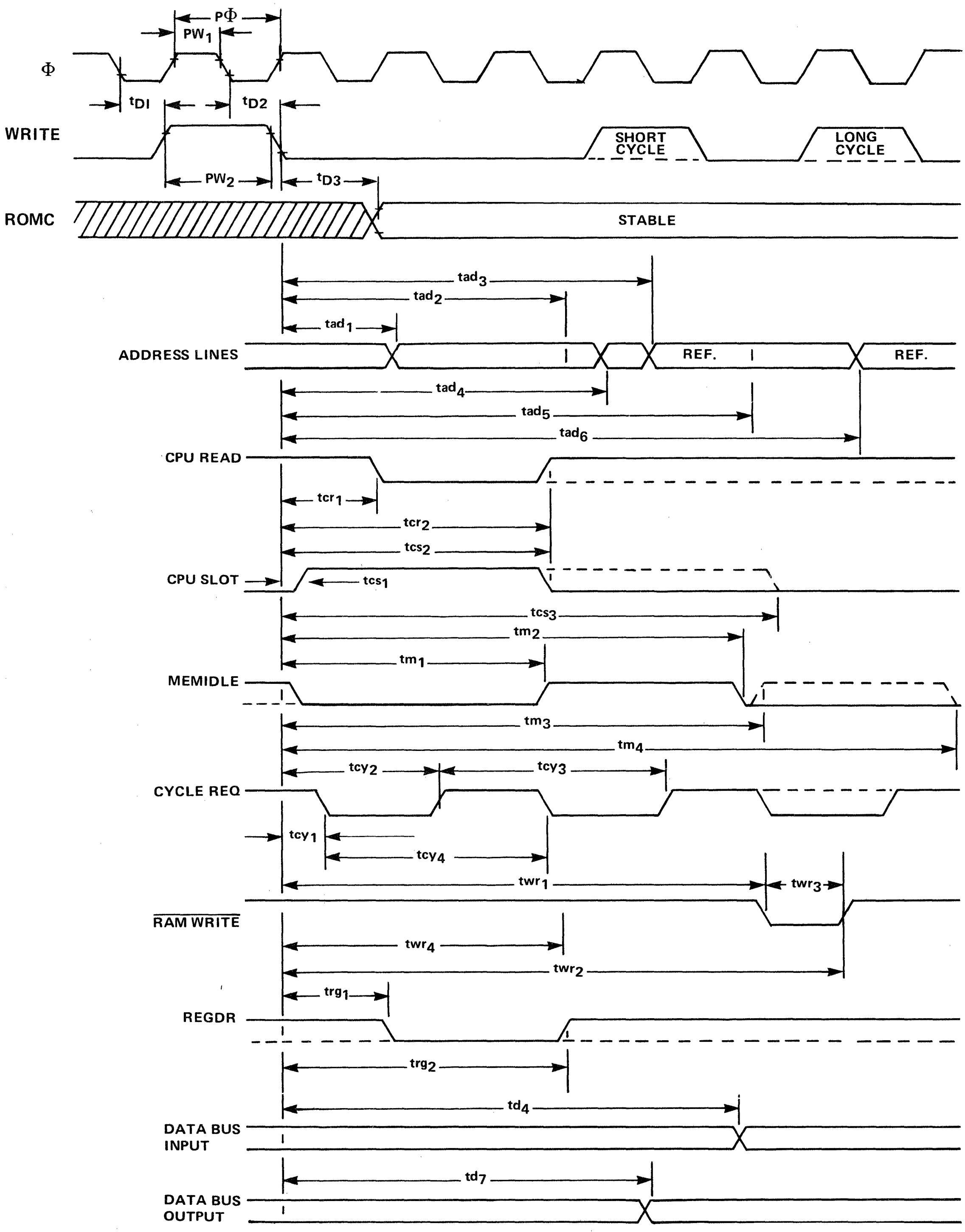
SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	TEST COND.
$P\Phi$	Φ Clock Period	0.5		10	μS	
PW_1	Φ Pulse Width	180		$P\Phi$–180	ns	
td_1	Φ to write + delay	0		300	ns	CL=100pF
td_2	Φ to write – delay	0		250	ns	CL=100pF
PW_2	Write Pulse Width	$P\Phi$–100		$P\Phi$	ns	
PW_S	Write Period; Short		$4P\Phi$		ns	
PW_L	Write Period; Long		$6P\Phi$		ns	
td_3	Write to ROMC Delay			750	ns	
tad_1	Address delay if PC0	50	300	500	ns	3
tad_2	Address delay to high Z(short cycle with DMA on)	tcs_2+50		tcs_2+200	ns	3
tad_3	Address delay to refresh(short cycle with REF on)	tcs_2+50		tcs_2+400	ns	3
tad_4	Address delay if DC	$2P\Phi$+50-td_2		$2P\Phi$+400-td_2	ns	3
tad_5	Address delay to high Z(long cycle with DMA on)	tcs_3+50		tcs_3+200	ns	3
tad_6	Address delay to refresh(long cycle with REF on)	tcs_3+50		tcs_3+400	ns	3
tcr_1	CPU READ-Delay	50	250	450	ns	1
tcr_2	CPU READ + Delay	$2P\Phi$+50-td_2		$2P\Phi$+400-td_2	ns	1
tcs_1	CPU SLOT + Delay	80-td_2		320-td_2	ns	1
tcs_2	CPU SLOT - Delay (PC0 access)	$2P\Phi$+60-td_2		$2P\Phi$+420-td_2	ns	1
tcs_3	CPU SLOT - Delay (DC access)	$4P\Phi$+60-td_2		$2P\Phi$+420-td_2	ns	1
tm_1	MEMIDLE + Delay (PC0 access)	$2P\Phi$+50-td_2		$4P\Phi$+400-td_2	ns	1
tm_2	MEMIDLE - Delay (PC0 access)	$4P\Phi$+50-td_2		$4P\Phi$+350-td_2	ns	1
tm_3	MEMIDLE + Delay (DC access)	$4P\Phi$+50-td_2		$4P\Phi$+400-td_2	ns	1
tm_4	MEMIDLE - Delay (DC access)	$6P\Phi$+50-td_2		$6P\Phi$+350-td_2	ns	1
tcy_1	WRITE to CYCLE REQ – Delay	80-td_2		400-td_2	ns	1,4
tcy_2	WRITE to CYCLE REQ + Delay	$P\Phi$+80-td_2		$P\Phi$+400-td_2	ns	1,4
tcy_3	CYCLE REQ + to + Edge Delay		$2P\Phi$			1,4
tcy_4	CYCLE REQ - to - Edge Delay		$2P\Phi$			1,4
twr_1	$\overline{\text{RAM WRITE}}$ - Delay	$4P\Phi$+50-td_2		$4P\Phi$+450-td_2	ns	3
twr_2	$\overline{\text{RAM WRITE}}$ + Delay	$5P\Phi$+50-td_2		$5P\Phi$+300-td_2	ns	3
twr_3	$\overline{\text{RAM WRITE}}$ Pulse Width	350		$P\Phi$	ns	3
twr_4	$\overline{\text{RAM WRITE}}$ to High Z Delay	tcs_2+40		tcs_2+200	ns	3
trg_1	REGDR - Delay	70	300	500	ns	1
trg_2	REGDR + Delay	$2P\Phi$+80-td_2		$2P\Phi$+500-td_2	ns	1
td_4	WRITE to Data Bus Input Delay			$2P\Phi$+1000	ns	
td_7	WRITE to Data Bus Output Delay	$2P\Phi$+100-td_2		$2P\Phi$+850-td_2		2

NOTES:

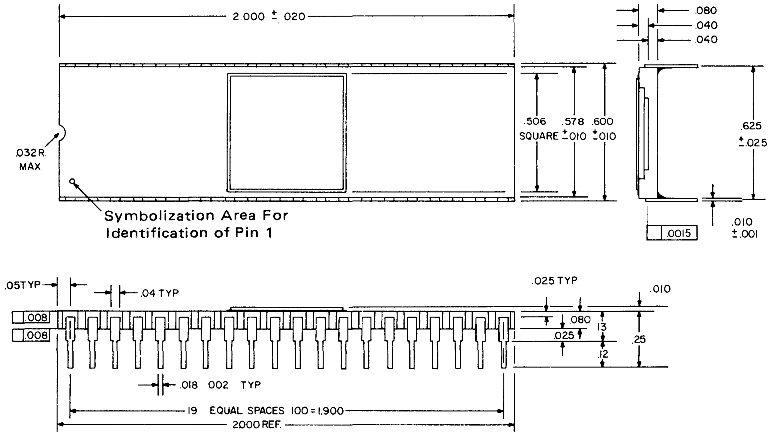
1. C_L = 50pF
2. C_L = 100pF.
3. C_L = 500pF.
4. CYCLE REQ is a divide-by-2 of Φ for all instructions except the STORE instruction.

TIMING DIAGRAM

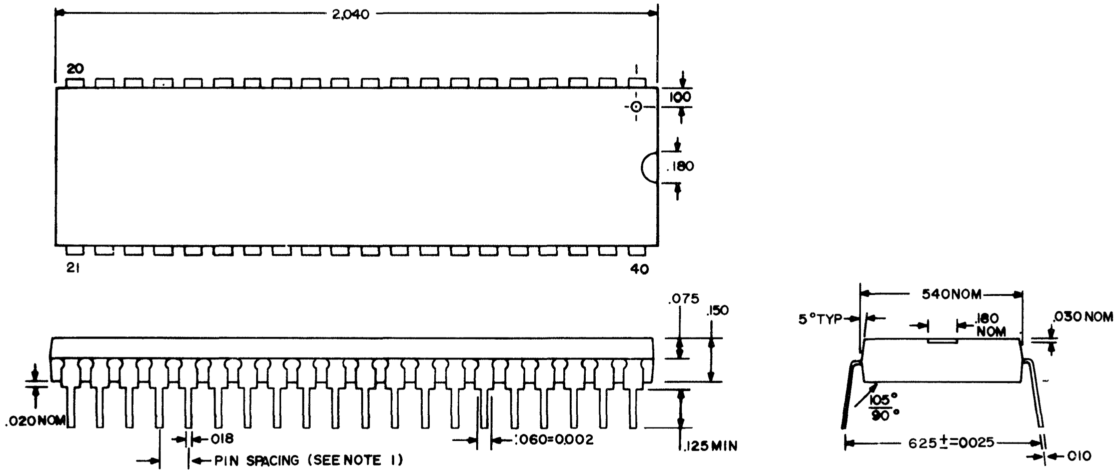
Figure 2



PACKAGE DESCRIPTION: 40-Pin Dual In-Line Ceramic Package



PACKAGE DESCRIPTION – 40-Pin Dual-in-Line Plastic Package



ORDERING INFORMATION

PART NUMBER	PACKAGE	TEMPERATURE RANGE
MK3852P	Ceramic	0° C to +70° C
MK3852N	Plastic	0° C to +70° C
MK3852P-10	Ceramic	-40° C to +85° C
MK3852N-10	Plastic	-40° C to +85° C

MOSTEK®

F8 MICROCOMPUTER DEVICES

Static Memory Interface MK 3853

FEATURES

- □ Static Memory Interface to RAM, ROM or PROM
- □ Programmable Timer
- □ Programmable Interrupt Vectors for Timer and External Interrupts
- □ Low Power Dissipation Typically Less Than 335 mw

GENERAL DESCRIPTION

The MK 3853 Static Memory Interface (SMI) provides all necessary address lines and control signals to interface up to 65,536 bytes of Static RAM, ROM or PROM to an F8 microcomputer system. When quantities do not justify the mask charges for the MK 3851 PSU, or a fast turn around is of high importance, the MK 3853 SMI can be used to interface the F8 to EPROM or fusible-link bipolar PROMs. The 3853 SMI along with standard PROM can emulate the memory function of the 3851 PSU, while the 3861 provides the I/O ports, interrupt and timer features of the 3851 PSU. The 3853 is a high performance MOS/LSI circuit using N-channel Isoplanar technology.

FUNCTIONAL PIN DESCRIPTION

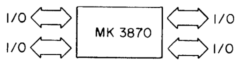
ADDR0-ADDR15 – The address bus provides the location of a memory read or write cycle.

DB0-DB7 – The Data Bus provides bi-directional communication between the 3850 F8 CPU and the 3853 SMI and all other F8 peripheral devices.

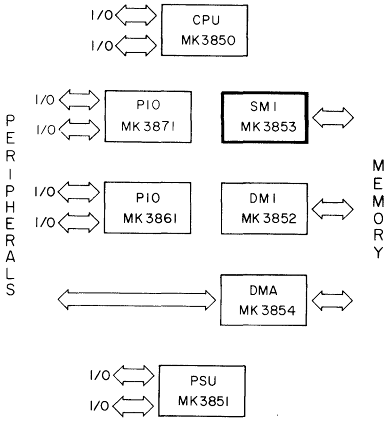
ROMC0-ROMC4 – These lines provide the 3853 SMI with control information from the 3850 F8 CPU.

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional, tri-state
ADDR0-ADDR15	Address Lines	Output
Φ, WRITE	Clock Lines	Input
INT REQ	Interrupt Request	Output
PRI IN	Priority In Line	Input
RAM WRITE	Write Line	Output
EXT INT	External Interrupt Line	Input
REGDR	Register Drive Line	Input/Output
CPU READ	CPU Read Line	Output
ROMC0-ROMC4	Control Lines	Input
V_{GG}, V_{DD}, V_{SS}	Power Supply Lines	Input

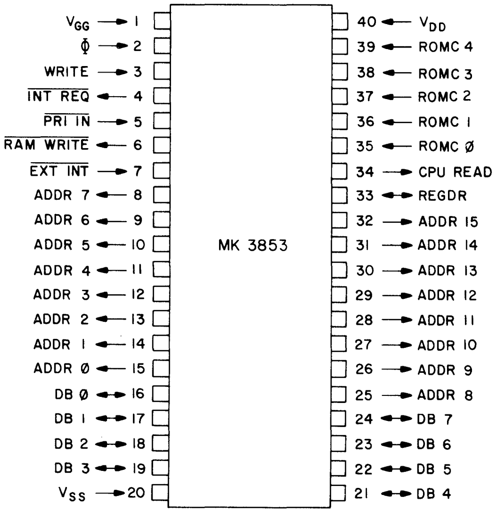
SINGLE CHIP MICROCOMPUTER



F8 FAMILY



PIN CONNECTIONS



BLOCK DIAGRAM

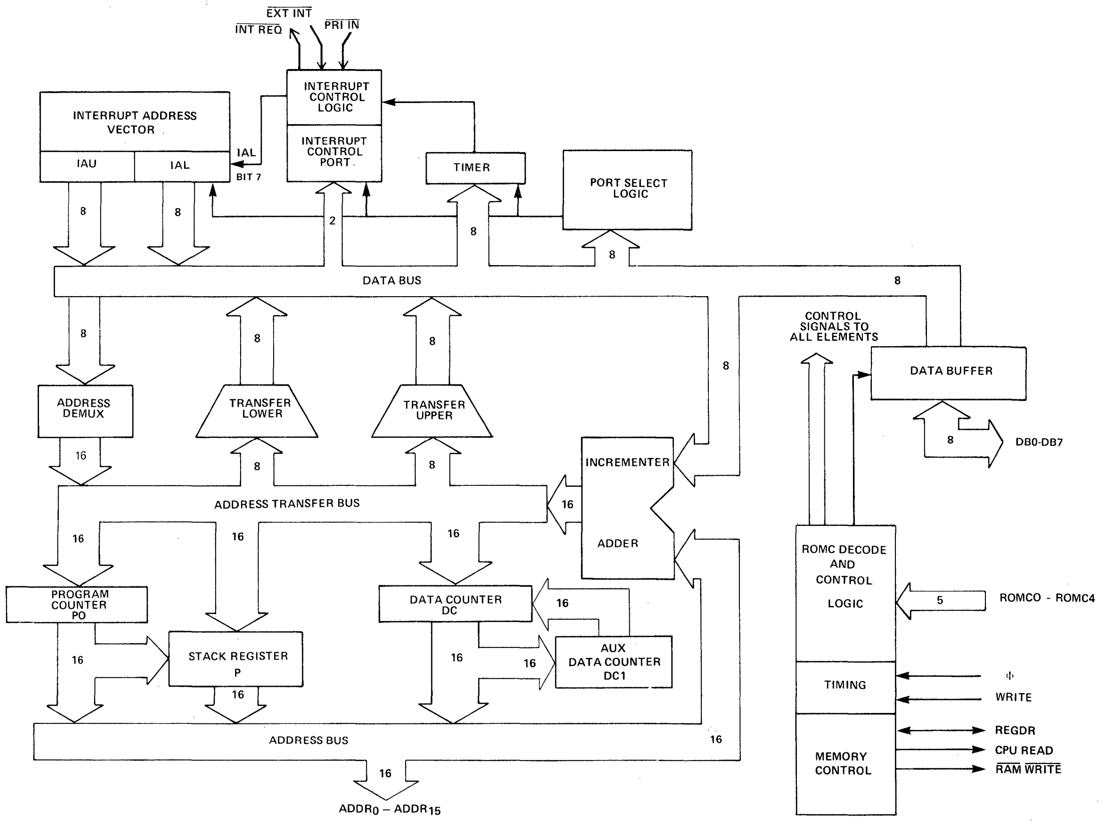


Figure 1

Φ– This is the system clock generated by the 3850 F8 CPU.

WRITE – This clock defines the machine cycle.

$\overline{\text{EXT INT}}$ – When an external circuit pulls this input "low", an external interrupt will be latched into the SMI if its interrupt control register has been set up to allow external interrupts. The SMI will then communicate this interrupt request to the CPU via $\overline{\text{INT REQ}}$ line.

$\overline{\text{PRI IN}}$ – This input signals the SMI that a higher priority peripheral has an interrupt request pending on the CPU. If the SMI has already requested an interrupt, the interrupt request will be maintained, but will not be serviced by the CPU until $\overline{\text{PRI IN}}$ is in the "low" state.

$\overline{\text{INT REQ}}$ – This is an open drain output that is wire ORed with the corresponding $\overline{\text{INT REQ}}$ outputs of all other peripherals to form the interrupt request input to the CPU.

$\overline{\text{RAM WRITE}}$ – This signal, when low, specifies that data from the data bus is to be written into a RAM location specified by the address bus.

CPU READ – This signal when high, specifies that data is to be read from the memory array interfaced to the SMI.

REGDR (OUTPUT/INPUT) – This signal functions both as an input and an output, to gate P0, DC, and I/O ports '0C' and '0D' onto the data bus at the proper time.

DEVICE ORGANIZATION

This section describes the basic functional elements of the MK 3853 SMI. These elements are shown in the SMI functional block diagram (figure 1).

PROGRAM COUNTER (PO) AND DATA COUNTERS (DC AND DC1)

The MK 3853 SMI addressing logic consists of 3 16-bit registers , the Program Counter (PO) and the Data Counters (DC and DC1)

The Program Counter will at all times address the memory word from which the next object program code must be fetched. The Data Counter (DC) addresses memory words containing individual data bytes or bytes within data tables to be used as operands.

It is important to note that the 3853 SMI has an auxiliary Data Counter (DC1). The contents of DC can be saved in DC1 by using the instruction XDC (exchange data counters). This instruction puts the contents of DC into DC1 and the contents of DC1 into DC. DC⇄DC1.

PO will always address the memory location out of which the next object program instruction byte will be read. If the instruction requires data (an operand) other than an immediate operand to be accessed, DC must address memory. PO cannot be used to address a NON-immediate operand since PO is saving the address of the next instruction code.

THE STACK REGISTER P

The MK 3853 SMI addressing logic contains a fourth 16-bit register called the stack register (P). The stack register is a buffer for the program counter PO. The contents of the stack register are never used directly to address memory.

The following instructions access P

LR K, P	Move the contents of P to the CPU scratchpad K registers
LR P,K	Move the contents of the CPU K scratchpad registers to P
PK	Save the contents of PO in P then move the contents of CPU scratchpad registers 12 and 13 to PO
PI H'aaaa'	Move the contents of PO to P then load the hexadecimal value into PO
POP	Move the contents of P to PO

In addition, when an interrupt is acknowledged, the contents of PO are saved in P.

MEMORY CONTROLS

The 3853 SMI provides three memory control outputs: RAM WRITE, CPU READ and REGDR.

$\overline{\text{RAM WRITE}}$ is used to control the read/write cycle of a static memory. $\overline{\text{RAM WRITE}}$ should be tied directly to the R/$\overline{\text{W}}$ line of the static memory.

CPU READ is a control signal that signifies that data is to be read out of a memory location. CPU READ and an externally generated address page select signal can be gated together to form a signal to enable the output of the memory array onto the F8 data bus.

REGDR is both an input and an output that is used to gate the program counter PO, data counter (DC), and I/O ports ('OC'H and 'OD'H) onto the data bus at the proper time. If the 3851 PSU or 3852 DMI are not used in the system, then REGDR may be left open. If one or more 3851 PSU's are used in a system without the 3852 DMI, then the signal DBDR from all PSU's in the system should be tied together and gated through an open collector AND gate and tied to REGDR of the SMI. If the 3852 DMI and the 3853 SMI are used in a system without the 3851 PSU, then REGDR of the SMI should be left open and REGDR of the 3852 DMI should be tied low to prevent a data bus conflict when the PO and DC registers are output onto the data bus.

INCREMENTER ADDER LOGIC

There are only two arithmetic operations that memory devices need to perform on the contents of memory address registers:

1. Increment by 1 the 16-bit value stored in an address register.
2. Add an 8-bit value, treated as a signed binary number (subject to twos complemented arithmetic) to the 16-bit value stored in address register.

The incrementer adder logic performs these two functions in the MK 3853 SMI.

INTERRUPT LOGIC

This logic responds to an interrupt request signal which may originate internally from timer logic, or be input by an external device. Based on priority considerations, the interrupt request is passed on to the MK 3850 CPU.

F8 Family

TIMER LOGIC

Every MK 3853 SMI has a polynomial shift register which may be used in conjunction with interrupt logic to generate real-time intervals.

Upon counting down to zero, the timer uses interrupt logic to signal that it has timed out.

The timer is programmable and is handled as though it were an I/O port. Using an OUT or OUTS instruction, a value may be loaded into the timer in order to determine the real-time period at the end of which a time-out interrupt will be generated.

THE DATA BUS

The 8-bit data bus is the main path for transfer of information between the MK 3850 CPU and other devices in the F8 microprocessor system.

ADDRESSABLE I/O PORTS

Every MK 3853 SMI has four, 8-bit I/O ports. Two of the I/O ports are used to store a programmable interrupt vector address. A third I/O port is assigned to a programmable timer while a fourth port is the Interrupt Control Port.

ROMC STATES

ROMC (Hexadecimal)	OPERATION PERFORMED	COMMENT
00	DB ← ((P0)) ; P0←P0 +1	OP CODE, FETCH
01	DB ← ((P0)) ; P0←P0 +DB	BRANCH OFFSET FETCH
02	DB ←((DC)); DC←DC+1	
03	DB ←((P0)) ; P0←P0+1	IMMEDIATE OPERAND FETCH
04	P0 ←P	
05	((DC))←DB ; DC←DC+1	
06	DB←DCU	
07	DB←PU	
08	P←P0 ; DB←H'00'; P0L, P0H← DB	EXTERNAL RESET
09	DB ←DCL	
0A	DC←DC+DB	
0B	DB←PL	
0C	DB←((P0)) ; DCL←DB	
0D	P←P0+1	
0E	DB←((P0)) ; DCL←DB	
0F	P←P0 ; DB←IAL ; P0L← DB	LOWER BYTE OF ADDRESS VECTOR
10	FREEZE INTERRUPT STATUS	PREVENT ADDRESS VECTOR CONFLICTS
11	DB←((P0)) ; DCU←DB	
12	P0L←DB ; P←P0	
13	DB←IAU; P0U←DB	UPPER BYTE OF ADDRESS VECTOR
14	P0U←DB	
15	PU←DB	
16	DCU←DB	
17	P0L ←DB	
18	PL←DB	
19	DCL←DB	
1A	((pp))←DB or ((p))←DB	
1B	DB←(pp)) or DB←((p))	
1C	NO OPERATION	
1D	DC ⇄ DC1	
1E	DB←P0L	
1F	DB←P0U	

Definitions	DB -	Data Bus	IA -	Interrupt address vector
	P0 -	Program Counter	L -	Lower byte suffix
	DC -	Data Counter	U -	Upper byte suffix
	DC1 -	Aux Data Counter	( ) -	Contents of
	P -	Stack Register	← -	transfer to
	pp -	Two hex digits (long I/O port address)	⇄ -	exchange
	p -	One hex digit (short I/O port address)		

Table 1

The four I/O ports of the MK 3853 SMI have the following port addresses:

H'0C'	Programmable Interrupt Vector (upper byte)
H'0D'	Programmable Interrupt Vector (lower byte)
H'0E'	Interrupt Control Port
H'0F'	Programmable Timer

OPERATIONAL DESCRIPTION

CLOCK TIMING

All timing within the MK 3853 SMI is controlled by Φ and $\overline{\text{WRITE}}$, which are input from the MK 3850 CPU. Each machine cycle will contain either 4 Φ clock periods (short cycle) or 6 Φ clock periods (long cycle).

The WRITE clock refreshes and updates the MK3853 SMI. A machine cycle begins with the fall of the WRITE clock and the system control lines become stable shortly after the start of the cycle.

INSTRUCTION EXECUTION

The MK 3853 SMI responds to signals which are output by the MK 3850 CPU in the course of executing instruction cycles.

Table 1 summarizes the response of the MK 3853 SMI to the ROMC states.

MEMORY ADDRESSING

Those ROMC states which specify a memory access call for only one memory device to respond to the memory access operation. However, every memory device responds to ROMC states that call for modification of program counter or data counter register contents. Consider two examples:

1. ROMC state 5 specifies that the data counter DC register contents must be incremented. Every memory device will simultaneously receive this ROMC state, and will simultaneously increment the contents of its DC register.

2. ROMC state 0 is the standard instruction fetch. Only the memory device whose address space includes the current contents of the program counter P0 registers will respond to this ROMC state by accessing memory and placing the contents of the addressed memory word on the 8-bit data bus. However, every memory device will increment the contents of its P0 register, whether or not the P0 register contents are within the memory space of the device.

When all memory devices are connected to the 8-bit data bus of a MK 3850 CPU and are also connected to the ROMC control lines of the same CPU, the memory devices simultaneously receive the same ROMC state signals from the CPU and respond to ROMC states by identically modifying the contents of memory address registers. Therefore the P0 register on all memory devices contains identical information. The same holds true for DC and P registers.

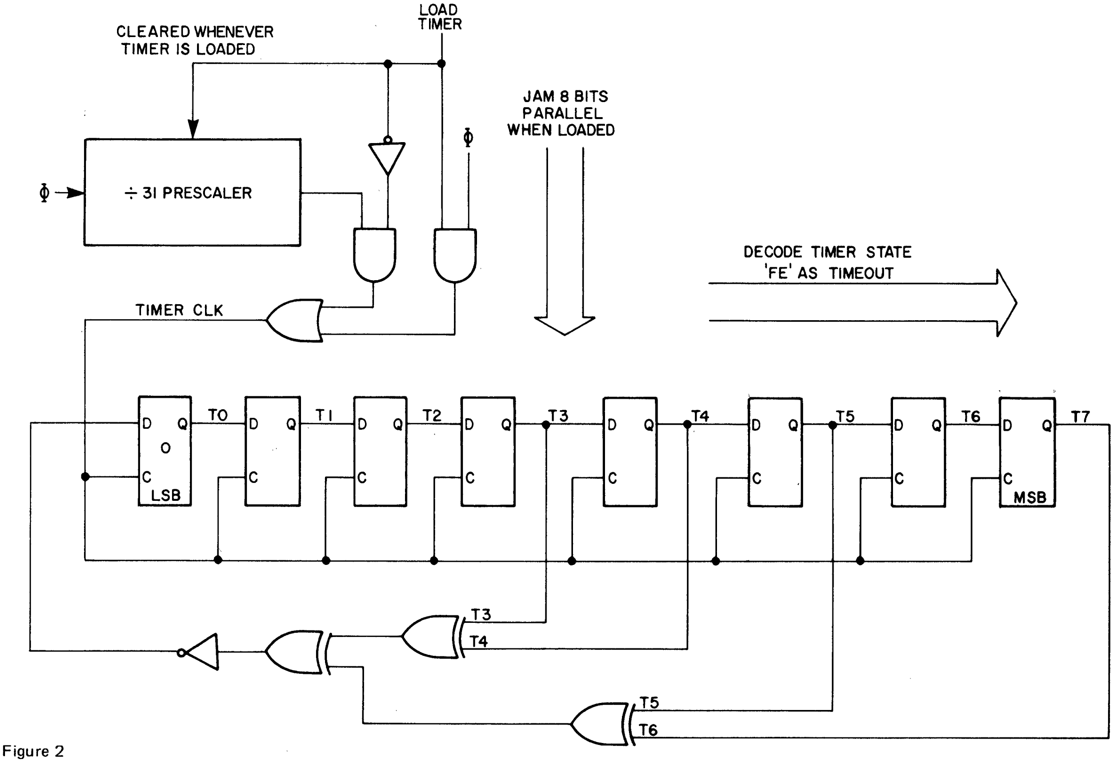
Only the memory device whose address space includes the specified memory address, will respond to any memory access request. To avoid addressing conflicts, it is necessary to insure that the following three conditions exist:

1. Memory devices must receive the ROMC state signals from one CPU.

2. Memory array decoding must not overlap. (More than one memory device cannot have the same memory space).

3. The memory address contained in the specified register (P0 or DC) must be within the memory space of memory device.

TIMER LOGIC DIAGRAM



LOAD TIMER
CLEARED WHENEVER TIMER IS LOADED
JAM 8 BITS PARALLEL WHEN LOADED
Φ
Φ
÷ 31 PRESCALER
DECODE TIMER STATE 'FE' AS TIMEOUT
TIMER CLK
D Q C O LSB
T0
T1
T2
T3
T4
T5
T6
T7
MSB
T3
T4
T5
T6

Figure 2

DATA INPUT TO THE SMI

The worst case timing for the MK 3853 SMI receiving data from the data bus is when the data must be added to a 16-bit number within the SMI's Incrementer Adder. This worst case corresponds to data coming from the accumulator in the CPU for an ADC instruction or from a memory device for a BR instruction. For this worst case, arriving data must allow sufficient time for 16-bit Adder logic.

THE PROGRAMMABLE TIMER

The MK 3853 SMI has an 8-bit shift register, addressable as an I/O port, of which may be used as a programmable timer. Figure 2 illustrates the shift register logic and the exclusive OR feedback path.

Based on the logic illustrated in Figure 2, binary values in the range 0 through 254, when loaded into the timer, are converted into "timer counts", as shown in Table 2. Table 2 contains the actual (HEX) value loaded into the timer, and the column/row is the corresponding decimal number of time intervals the timer will take to time out. Data cannot be read out of the programmable timer I/O port.

Either the OUT or OUTS instruction is used to load "timer counts" into the programmable timer. The contents of the programmable timer cannot be read using an IN or INS instruction. The timer will time out after a time interval given by the product:

$$(\text{period of clock}\ \Phi)\ \text{X}\ (\text{timer counts})\ \text{X}\ 31$$

For example, a value of H'C8' loaded into the programmable timer becomes 215 timer counts. The timer will therefore time out in 3.33 milliseconds, if the period of clock signal Φ is 500 nanoseconds.

A value of H'FF' loaded into the programmable timer will stop the timer. This is because the timer shift register feedback gates will always present a logic 1 to the D input of the LSB flip-flop (Fig. 2). Therefore, the timer will retain a value to H'FF' and a H'FE' will never be decoded to cause a time out.

The timer runs continuously unless it has been stopped by loading H'FF' into it. Upon timing out, the timer transmits an interrupt request to the interrupt logic. If proper interrupt logic conditions exist, the timer interrupt request is passed on to the CPU via INT REQ.

After the programmable timer has timed out it will again time out after 255 time counts. Therefore, if the programmable timer is simply left running, it will time out every 7905 Φ clock periods or every 3.9525 milliseconds for a 500 nanosecond clock.

Whenever the timer and timer interrupt are being set to time a new arrival, the timer should be loaded before enabling the timer interrupt. The act of loading the timer clears any pending timer interrupts. When the timer interrupt is enabled, any pending timer interrupt will be acknowledged and forwarded to the CPU. Since the timer runs continuously (unless stopped under program control) enabling the timer before loading a time count can cause a spurious interrupt. Time outs of the timer are latched in the interrupt logic of the SMI, even while timer interrupts are disabled. When the timer is enabled, an immediate interrupt acknowledge will occur if the continuous running timer timed out while timer interrupts were disabled.

If the timer is loaded just prior to enabling timer interrupts a spurious interrupt request will not exist when the timer interrupt is enabled.

Figure 3 illustrates a possible sequence for a timer which is initially loaded with H'C8' then allowed to run continuously.

CONVERSION OF TIMER COUNTS INTO TIMER CONTENTS

	0	1	2	3	4	5	6	7	8	9
0	7F	BF	5F	2F	97	CB	E5	72	39	1C
1	0E	87	43	A1	D0	E8	F4	7A	3D	1E
2	0F	07	03	01	00	80	C0	60	B0	D8
3	EC	F6	7B	BD	5E	AF	D7	6B	35	1A
4	0D	06	83	41	A0	50	A8	54	AA	55
5	2A	15	8A	C5	E2	F1	F8	7C	3E	9F
6	CF	E7	73	B9	5C	AE	57	2B	95	CA
7	65	32	99	CC	66	B3	59	2C	16	0B
8	05	02	81	40	20	10	08	84	C2	61
9	30	98	4C	26	13	89	44	22	11	88
10	C4	62	B1	58	AC	56	AB	D5	6A	B5
11	5A	AD	D6	EB	75	BA	DD	6E	B7	5B
12	2D	96	4B	A5	D2	E9	74	3A	9D	CE
13	67	33	19	8C	C6	63	31	18	0C	86
14	C3	E1	70	38	9C	4E	27	93	C9	E4
15	F2	79	BC	DE	EF	77	BB	5D	2E	17
16	8B	45	A2	51	28	14	0A	85	42	21
17	90	48	24	12	09	04	82	C1	E0	F0
18	78	3C	9E	4F	A7	D3	69	34	9A	4D
19	A6	53	29	94	4A	25	92	49	A4	52
20	A9	D4	EA	F5	FA	7D	BE	DF	6F	37
21	1B	8D	46	23	91	C8	64	B2	D9	6C
22	B6	DB	6D	36	9B	CD	E6	F3	F9	FC
23	7E	3F	1F	8F	47	A3	D1	68	B4	DA
24	ED	76	3B	1D	8E	C7	E3	71	B8	DC
25	EE	F7	FB	FD	FE	FF halts timer				

Each timer count = 15.5 μs at 2MHz

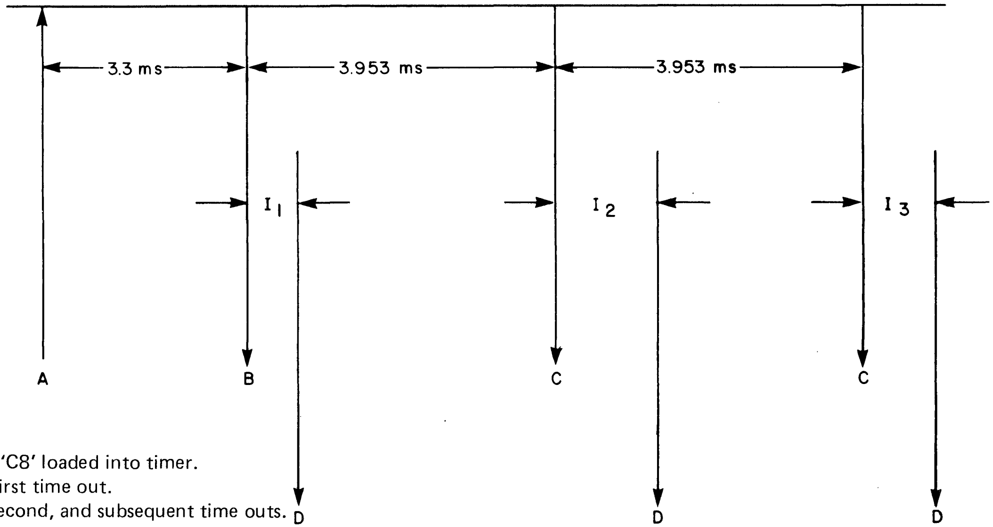
Table 2

INTERRUPT LOGIC ORGANIZATION

The interrupt Control Port has the I/O port address 'OE'H. Data is loaded into this register (I/O port) using an OUT or OUTS instruction. Data cannot be read from this port. The contents of the Interrupt Control Port are interpreted as follows:

CONTENTS OF INTERRUPT CONTROL PORT	FUNCTION
B'xxxxxx00'	Disable all interrupts
B'xxxxxx01'	Enable external interrupt, disable timer interrupt
B'xxxxxx10'	Disable all interrupts
B'xxxxxx11'	Disable external interrupt Enable timer interrupt

TIME OUT AND INTERRUPT REQUEST



A — H'C8' loaded into timer.
B — First time out.
C — Second, and subsequent time outs.
D — Interrupt Service Routines being entered by CPU.
I_1,I_2,I_3 — Intervals between time out interrupt request reaching interrupt logic and service routines being entered by CPU. These time intervals depend on the number of privileged instructions encountered from the time $\overline{\text{INT REQ}}$ goes low. If none are encountered, 34P Φ is the minimum interval (17 µs for P Φ = 500ns)

Figure 3

In the preceding I/O port contents definitions x represents "don't care" bits.

Depending on the contents of the Interrupt Control Port, a MK 3853 SMI's interrupt control logic can be accepting timer interrupts, or external interrupts, or neither, but never both.

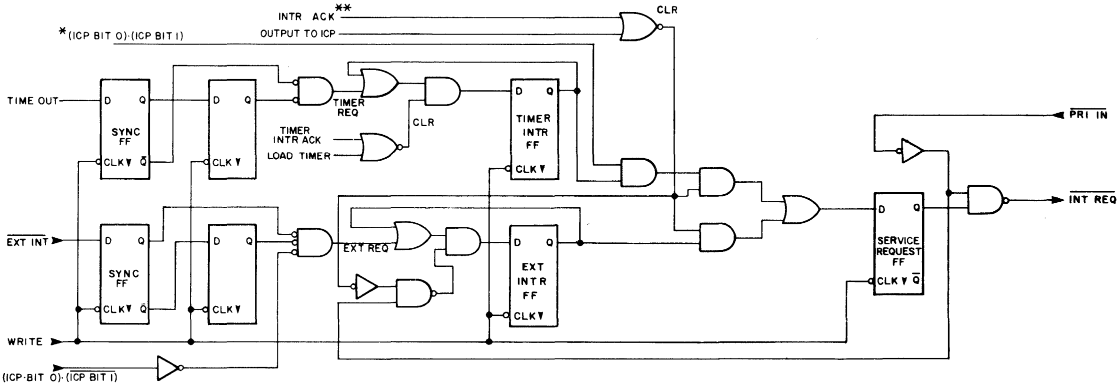
Figure 4 is a conceptual logic diagram of the SMI's interrupt logic. Between the $\overline{\text{EXT INT}}$ input or the time-out input and the output $\overline{\text{INT REQ}}$, there are 4 flip-flops. $\overline{\text{EXT INT}}$ and the time-out interrupt input each have 2 synchronizing flip-flops to detect the active edge.

Each edge detect circuit is followed by its own INTERRUPT flip-flop which latches the true condition.

The outputs of the TIMER INTERRUPT flip-flop and the EXTERNAL INTERRUPT flip-flop are ORed to set the SERVICE REQUEST flip-flop, provided that an interrupt from some other device is not being acknowledged by the CPU.

$\overline{\text{INT REQ}}$ is an open drain signal that is the NAND of PRI IN and SERVICE REQUEST. The $\overline{\text{INT REQ}}$ signal of several devices may be tied together so that any one can force the line to OV if it is requesting

INTERRUPT LOGIC



*ICP—INTERRUPT CONTROL PORT
**1 DURING EVENT G IN FIG 6

Figure 4

interrupt service. An internal pull-up to V_{DD} is provided by the MK 3850 CPU to the $\overline{\text{INT REQ}}$ input pin.

$\overline{\text{PRI IN}}$ is part of the interrupt priority chain. Each SMI has a $\overline{\text{PRI IN}}$ input but, it is important to note that the SMI does not have a $\overline{\text{PRI OUT}}$. This means that the SMI will be the last device in the daisy chain interrupt network. In a small system where only a CPU and a SMI are used, then $\overline{\text{PRI IN}}$ should be tied low. See Figure 5.

The SERVICE REQUEST flip-flop cannot be set if another interrupt request is in the process of being acknowledged anywhere in the system. If an interrupt request has been latched into the TIMER INTERRUPT flip-flop, or the EXTERNAL INTERRUPT flip-flop, the SMI logic waits until after the process of acknowledging the other interrupt before setting SERVICE REQUEST. This precaution is necessary to insure that the priority chain is not altered during acknowledgment. Chaos would result if half of the interrupt vector came from one device and the second half from some other device.

THE SERVICE REQUEST flip-flop is cleared after an interrupt from the SMI has been acknowledged. It is also cleared whenever the SMI interrupt control register is accessed by an output instruction.

The conditions for setting the TIMER INTERRUPT flip-flop and the EXTERNAL INTERRUPT flip-flop differ slightly. External interrupts must be enabled before the EXTERNAL INTERRUPT flip-flop can be set by a negative going transition of $\overline{\text{EXT INT}}$. However, TIMER INTERRUPT will be set by a timer TIME OUT independent of Interrupt Control Port bit 1. This means that the SMI can detect a time out interrupt that occurred while the external interrupt was enabled in the SMI.

The TIMER INTERRUPT flip-flop is cleared whenever the SMI's timer is loaded or when its timer interrupt has been acknowledged. The EXTERNAL INTERRUPT flip-flop is cleared whenever the SMI's interrupt control register is accessed by an output instruction, or when its external interrupt has been acknowledged.

INTERRUPT ACKNOWLEDGE SEQUENCE

Upon receiving an interrupt request, whether from an external source via $\overline{\text{EXT INT}}$ or from the internal timer, the SMI and CPU go through an interrupt sequence which results in the execution of an interrupt service routine located at the memory address pointed to by the Interrupt Address Vector. Figures 6 and 7 illustrate the interrupt sequence for the two cases. Events occurring in these sequences are labeled with the letters A through H. Events are described as follows.

EVENT A

The initial interrupt request arrives. The falling edge of $\overline{\text{EXT INT}}$ pin identified an external interrupt. The rising edge of interval timer output indicates a timeout.

EVENT B

The synchronizing flip-flop in the SMI control logic changes state.

EVENT C

The timer interrupt, or external interrupt flip-flop goes true, indicating the local interrupt logic's acknowledgment of the interrupt. The timer interrupt flip-flop will always respond and save the time-out occurrence, whereas the external interrupt flip-flop will only be set at this time if the external interrupt mode is enabled within the local control logic.

EVENT D

The $\overline{\text{INT REQ}}$ line is pulled low by the SMI, passing

INTERRUPT INTERCONNECTION

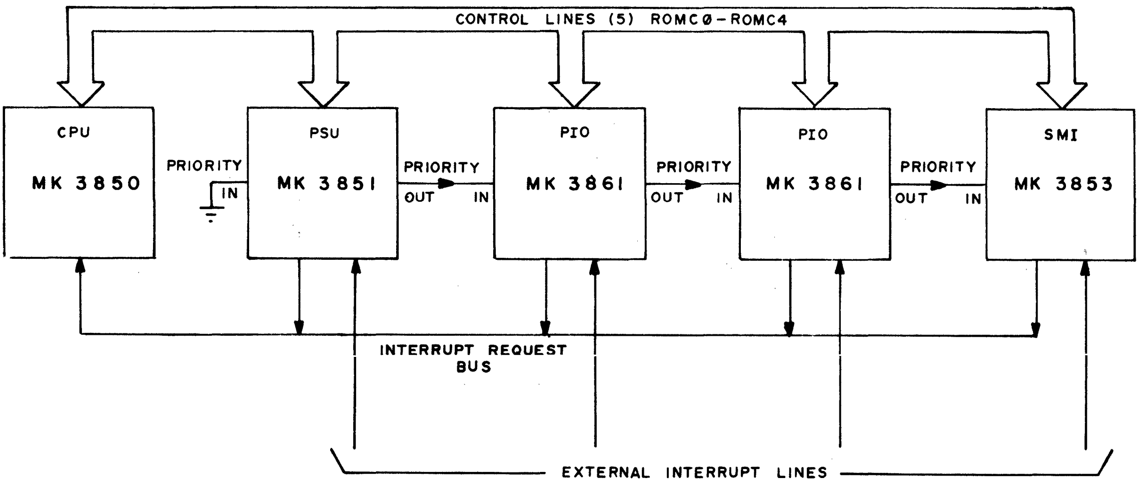


Figure 5

the request for servicing on to the CPU. The conditions that must be present for this to occur are:

The $\overline{\text{PRI IN}}$ pin must be low.

The proper enable state must exist in the local control logic for the type of interrupt (time or external). The system is not already into Event F due to servicing some other interrupt.

EVENT E

The CPU now begins its response to the $\overline{\text{INT REQ}}$, line by outputting the unique ROMC state H'10' inhibiting modification of interrupt priority logic. This will only occur when the following conditions are satisfied:

The CPU is executing the last cycle of an instruction (beginning an instruction fetch).

The ICB is enabled (ICB = 0).

The current instruction fetch is not protected (not a privileged instruction).

EVENT F

The CPU generates the interrupt acknowledge sequence of ROMC states as follows:

ROMC STATE

10	Inhibit modification of interrupt priority logic.
IC	No function
OF	Put lower byte of interrupt address vector on data bus
13	Put upper byte of interrupt address vector on data bus
00	Fetch instruction from memory (first instruction of interrupt service routine)

EVENT G

At this point the CPU begins fetching the first instruction of the interrupt service routine. In the SMI interrupt logic, the SERVICE REQUEST flip-flop and the appropriate INTERRUPT REQUEST flip-flop are cleared.

EVENT H

The CPU begins executing the first instruction of the interrupt service routine.

INTERRUPT ADDRESS VECTOR

During the interrupt acknowledge, the interrupting SMI provides a 16-bit interrupt address vector. The CPU causes this vector to be loaded into P0, so that program execution can branch to the routine that handles this particular interrupt. Fifteen bits of the interrupt vector are programmable from I/O ports 'OC'H and 'OD'H. Bit 7 cannot be programmed. It is set by the interrupt control logic to 0 if the timer interrupt is enabled or to a 1 if external interrupt is enabled. The interrupt vector is of the form: WWWW, XXXX, 0YYY, ZZZZ for timer interrupt and WWWW, XXXX, 1YYY, ZZZZ for external interrupt

TIMER INTERRUPT SEQUENCE

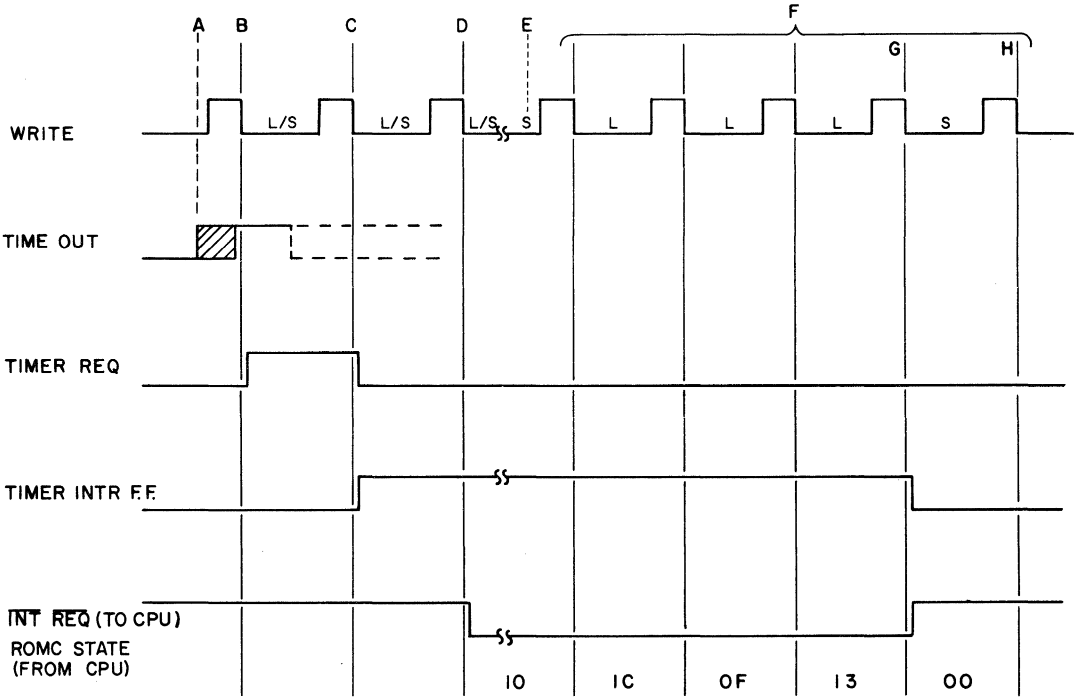


Figure 6

where W, X, Y and Z are the bits programmed by I/O ports 'OC'H and 'OD'H.

INTERRUPT SIGNALS TIMING

Timing for signals associated with the MK 3853 interrupt logic is shown in Figure 9.

EXTERNAL INTERRUPT SEQUENCE

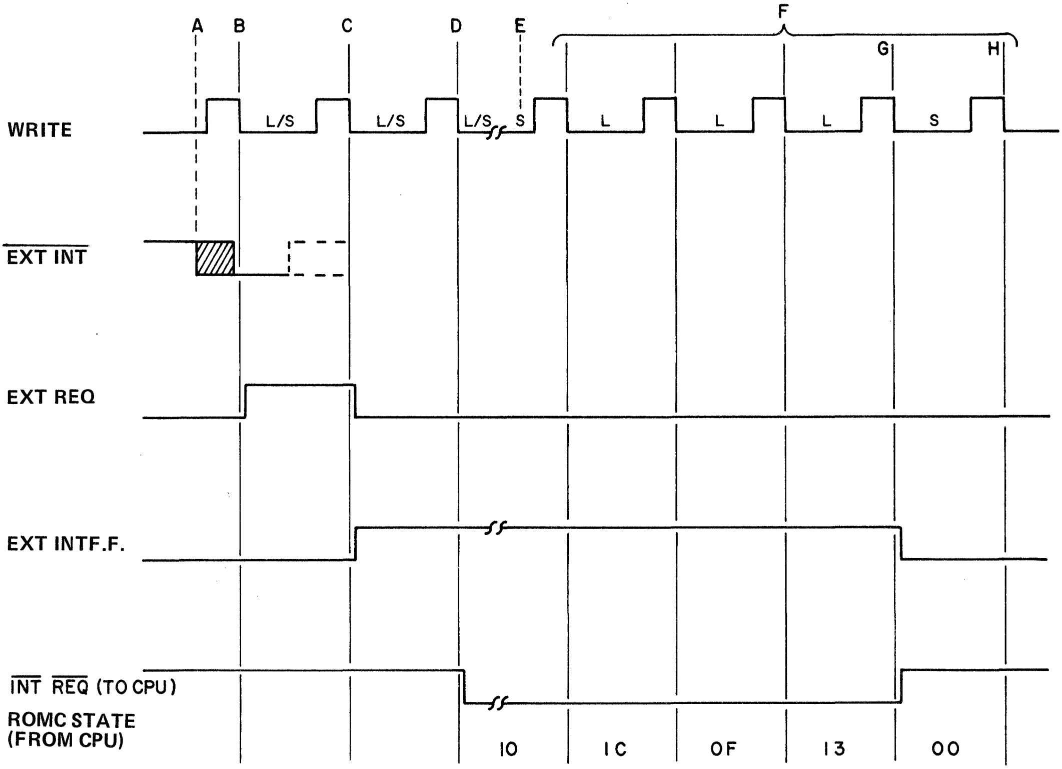


Figure 7

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATING (All voltages with respect to V_{SS}) *

V_{GG} .... +15V to –0.3V
V_{DD} .... +7V to –0.3V
All other inputs and outputs .... +7V to –0.3V
Operating temperature, T_A (Ambient) .... 0°C to +70°C
Storage temperature - Ambient (Ceramic) .... –65°C to +150°C
Storage temperature - Ambient (Plastic) .... –55°C to +125°C

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

($0°C \leqslant T_A \leqslant 70°C$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{DD}	Supply	4.75	5.0	5.25	Volts	
V_{GG}	Voltage	11.4	12.0	12.6	Volts	
V_{SS}		0	0	0	Volts	

DC ELECTRICAL CHARACTERISTICS

($0°C \leqslant T_A \leqslant 70°C$) ($V_{DD}$ = +5V ± 5%; V_{GG} = +12V ± 5%; V_{SS} = 0V)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		35	70	mA	f = 2 MHz, Outputs unloaded
I_{GG}	I_{GG} Current		13	30	mA	f = 2 MHz, Outputs unloaded

DATA BUS (DB0-DB7)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = −100μA
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	I_{OL} = 1.6mA
I_{IH}	Input High Current		1	μA	V_{IN} = V_{DD}, three-state mode
I_{IL}	Input Low Current		−1	μA	V_{IN} = V_{SS}, three-state mode
C_I	Capacitance		10	pF	Three-state mode

PRIORITY IN ($\overline{\text{PRI IN}}$), CONTROL LINES (ROMCO—ROMC4) AND CLOCK LINES (Φ, WRITE)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
I_L	Leakage Current		1	μA	V_{IN} = V_{DD}
C_I	Capacitance		10	pF	

ADDRESS LINES (ADDR0-ADDR15) and $\overline{\text{RAM WRITE}}$

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = −1mA
V_{OL}	Output Low Voltage		.4	Volts	I_{OL} = 3.2mA
I_L	Leakage Current		1	μA	V_{IN} = V_{DD}

F8 Family

INTERRUPT REQUEST ($\overline{\text{INT REQ}}$)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage				Open Drain Output [1]
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	I_{OL} = 1mA
I_L	Leakage Current		1	μA	V_{IN} = V_{DD}

EXTERNAL INTERRUPT ($\overline{\text{EXT INT}}$)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5	15	Volts	
V_{IL}	Input Low Voltage	V_{SS}	1.2	Volts	
V_{IC}	Input Clamp Voltage		15	Volts	I_{IH} = 185 μA
I_{IH}	Input High Current		10	μA	V_{IN} = V_{DD}
I_{IL}	Input Low Current	−250	−750	μA	V_{IN} = V_{SS}

Notes:

1. Pull-up resister to V_{DD} on CPU.

REGDR

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –300 μA
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	I_{OL} = 2mA
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	Internal Pull-up to V_{DD}
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
I_{IL}	Input Low Current	–3.5	–14.0	mA	V_{IN} = .4V and Device outputting a logic "1"
I_L	Leakage Current		1	μA	V_{IN} = V_{DD}

CPU READ

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS;
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –1mA
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	I_{OL} = 2mA
I_L	Leakage Current		1	μA	V_{IN} = V_{DD}

AC ELECTRICAL CHARACTERISTICS

($0°C \leqslant T_A \leqslant 70°C$) ($V_{DD}$ = + 5V ± 5%; V_{GG} = + 12V ± 5%; V_{SS} = 0V)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST COND
P Φ	Φ CLOCK Period	0.5		10	μS	
PW_1	Φ Pulse Width	180		PΦ–180	nS	
td_1	Φ to write + delay	0		300	nS	CL = 100pF
td_2	Φ to write – delay	0		250	nS	CL = 100pF
PW_2	Write Pulse Width	PΦ–100		PΦ	nS	
PW_S	Write Period; Short		4 PΦ		nS	
PW_L	Write Period; Long		6 PΦ		nS	
td_3	Write to ROMC Delay			750	nS	
td_4	Write to DB Input Delay			2PΦ+1.0	μS	
td_6	Write to DB Output Delay	2PΦ+100–td_2	2PΦ+200	2PΦ+800–td_2	nS	CL = 100pF
tad_1	Address delay if P0 (Instruction by immediate data)	50	300	500	nS	C_L = 500pF
tad_2	Address delay if DC (Operand fetch) or WRITE cycle	2Φ+50–td_2		2PΦ+620–td_2	nS	C_L = 500pF
tcr_1	CPU READ – Delay	50	250	450	nS	50pF
tcr_2	CPU READ + Delay	2PΦ+50–td_2		2PΦ+400–td_2	nS	50pF
tw_1	$\overline{\text{RAM WRITE}}$ – Delay	4PΦ+50–td_2		4PΦ+450–td_2	nS	500pF
tw_2	$\overline{\text{RAM WRITE}}$ + Delay	5PΦ+50–td_2		5PΦ+300–td_2	nS	500pF
tw_p	$\overline{\text{RAM WRITE}}$ Pulse Width	350		PΦ	nS	500pF
trg_1	WRITE to REGDR –Delay	70	300	500	nS	50pF
trg_2	WRITE to REGDR + Delay	2PΦ+80–td_2		2PΦ+500–td_2	nS	50pF

AC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
t_{r1}	WRITE to $\overline{\text{INT REQ}}$ – Delay			430	nS	C_L = 100 pF [1]
t_{r2}	WRITE to $\overline{\text{INT REQ}}$ + Delay			1.65	µs	C_L = 100 pF [3]
t_{pr1}	$\overline{\text{PRI IN}}$ to $\overline{\text{INT REQ}}$ – Delay			240	nS	C_L = 100 pF [2]
t_{pr2}	$\overline{\text{PRI IN}}$ to $\overline{\text{INT REQ}}$ + Delay			1.5	µs	C_L = 100 pF
tex	$\overline{\text{EXT INT}}$ Setup Time	400			nS	

NOTES:
1. Assume PRIORITY IN was enabled ($\overline{\text{PRI IN}}$ = 0) in previous F8 cycle before interrupt is detected in the SMI.
2. SMI has interrupt pending before $\overline{\text{PRIORITY IN}}$ is enabled.
3. Assume pin tied to $\overline{\text{INT REQ}}$ input of 3850 CPU.

TIMING DIAGRAM

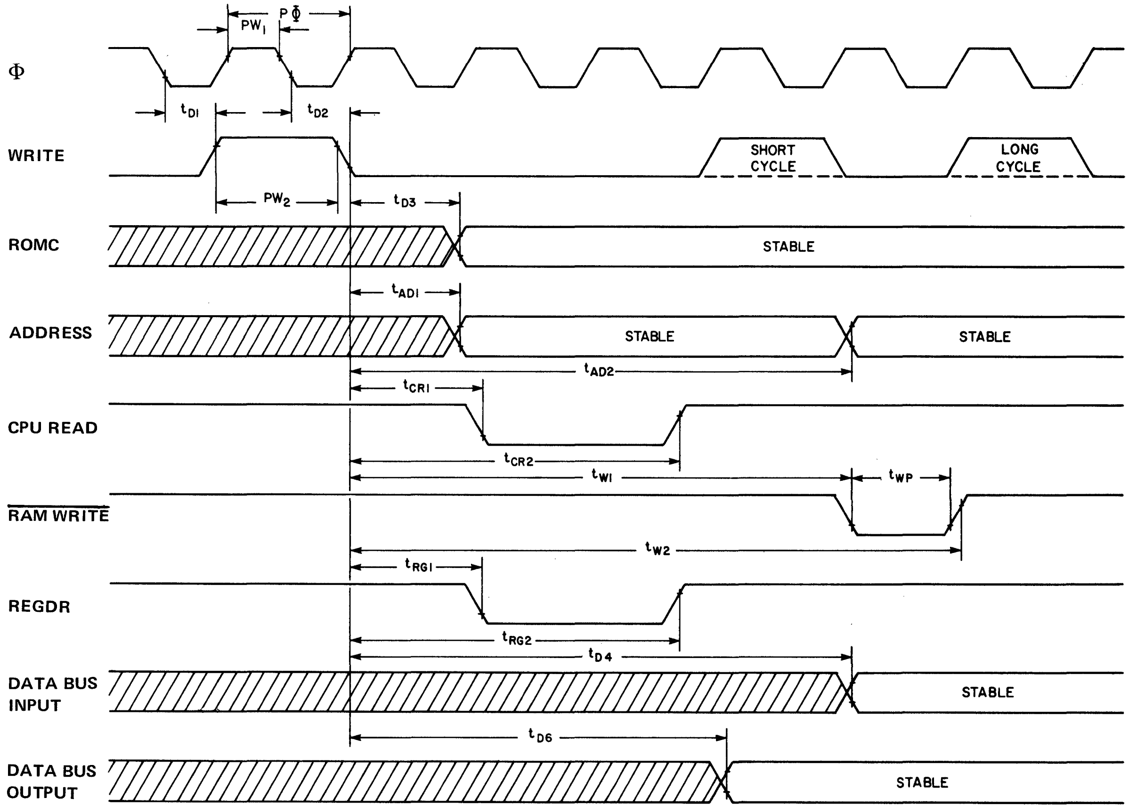


Figure 8

TIMING DIAGRAM

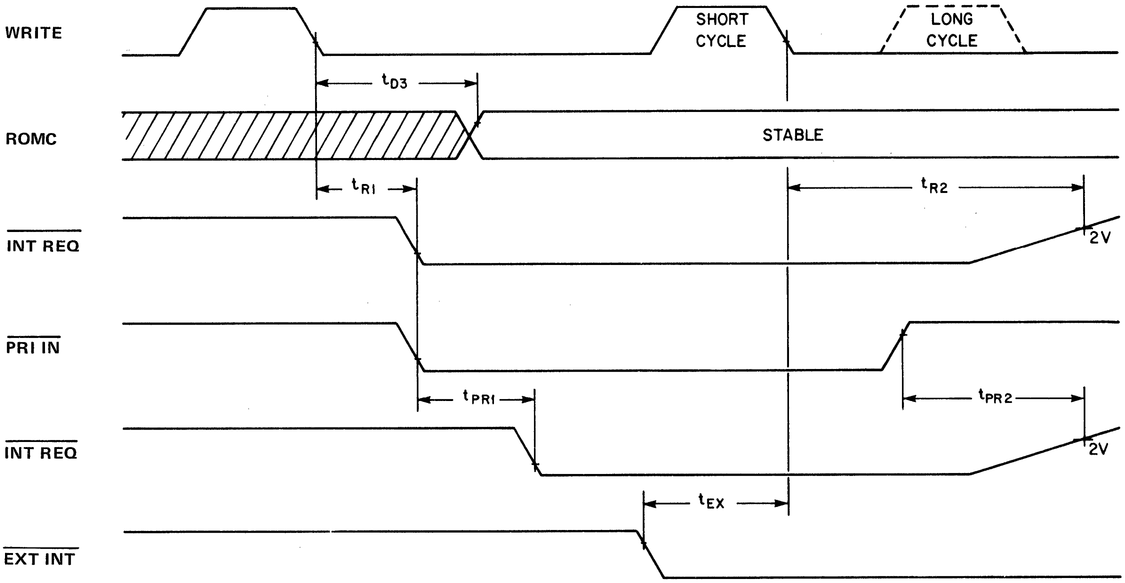
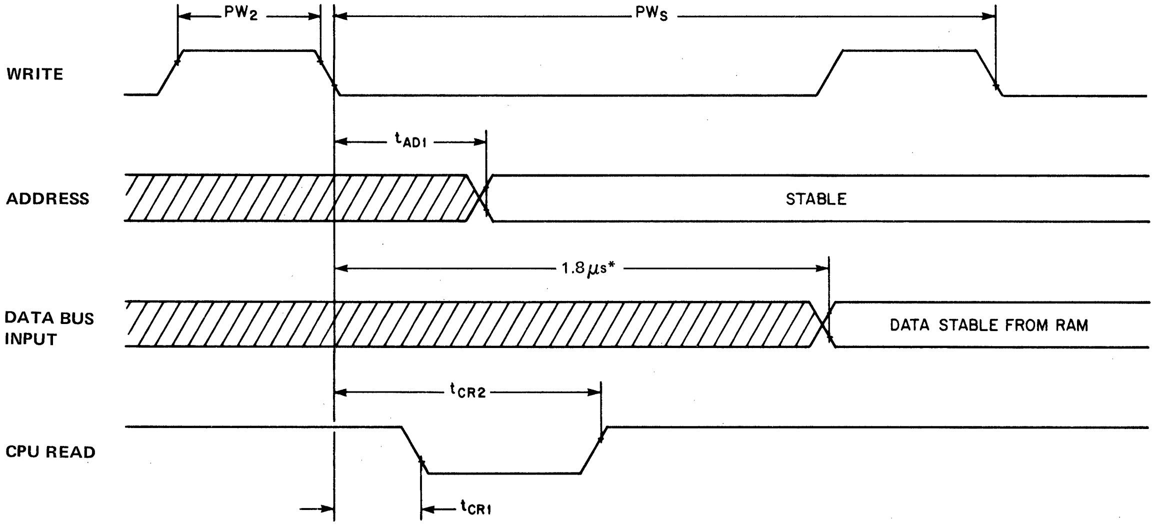

WRITE
SHORT CYCLE
LONG CYCLE
t_{D3}
ROMC
STABLE
t_{R1}
t_{R2}
$\overline{\text{INT REQ}}$
2V
$\overline{\text{PRI IN}}$
t_{PR1}
t_{PR2}
$\overline{\text{INT REQ}}$
2V
t_{EX}
$\overline{\text{EXT INT}}$

Figure 9

TIMING DIAGRAM

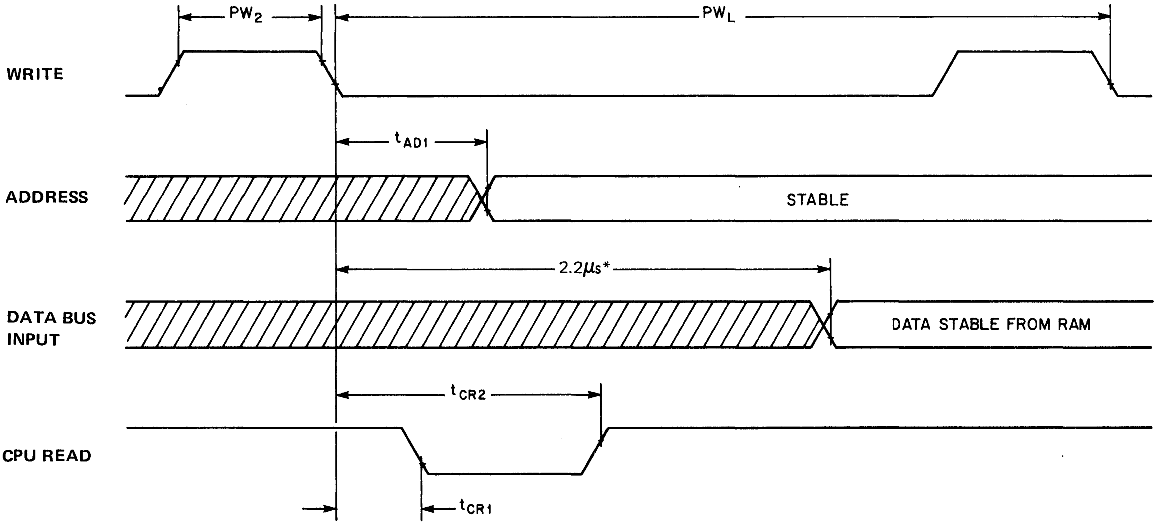


MK 3853 SMI TIMING SIGNALS OUTPUT DURING A SHORT CYCLE MEMORY READ USING P0

Figure 10

*NOTE: This is the time at which the CPU will strobe data in from the memory. (Φ = 2 MHz) Refer to MK3850 CPU data sheet for further information.

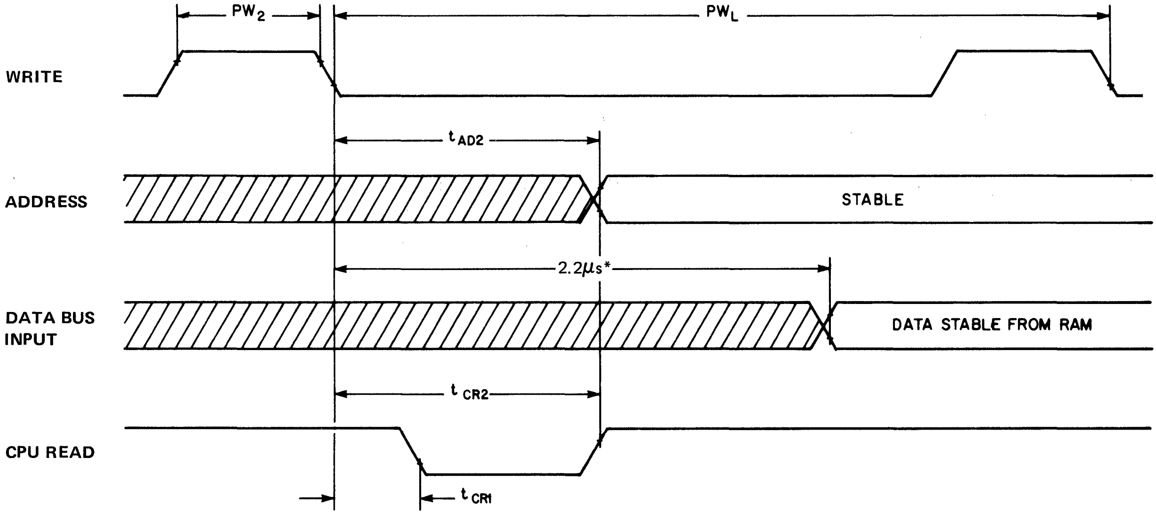
TIMING DIAGRAM



MK 3853 SMI TIMING SIGNALS OUTPUT DURING A LONG CYCLE MEMORY READ, WITH ADDRESS OUT OF PROGRAM COUNTER

Figure 11

TIMING DIAGRAM

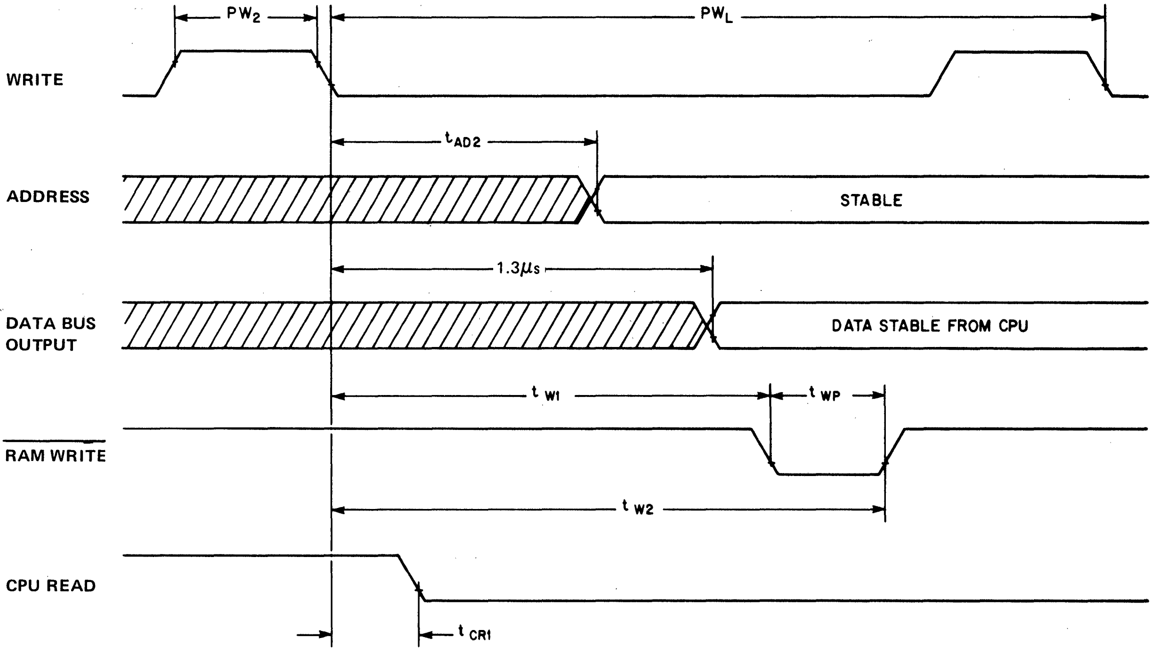


MK 3853 SMI TIMING SIGNALS OUTPUT DURING A LONG CYCLE MEMORY READ, WITH ADDRESS OUT OF DATA COUNTER

Figure 12

*NOTE: This is the time at which the CPU will strobe data in from the memory. (Φ = 2 MHz) Refer to MK3850 CPU data sheet for further information.

TIMING DIAGRAM



MK 3853 SMI TIMING SIGNALS OUTPUT DURING A WRITE TO MEMORY

Figure 13

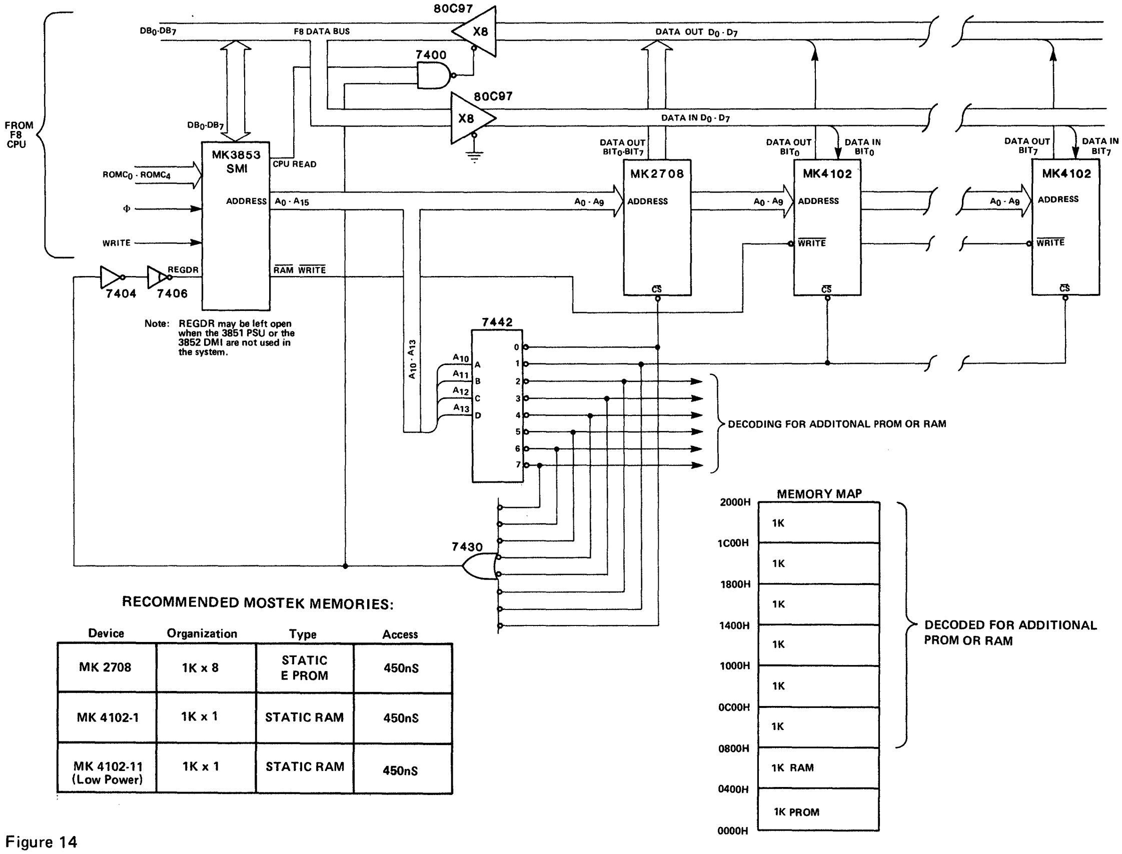
*NOTE: This is the time at which the CPU will output data to memory. (Φ= 2 MHz) Refer to MK3850 CPU data sheet for further information.

MK 3853 APPLICATION

Figure 9 shows a typical application for interfacing the MK 3853 SMI to static memories. This particular example shows a memory system using the MK 2708 1K x 8 EPROM and the MK 4102 1K x 1 Static RAM. Decoding is provided in 1K boundaries for up to 8K of memory. This should be more than adequate for most systems. However, if memory expansion is desired, decoders can be added to provide additional decoding.

The input to the memory array is isolated from the F8 data bus to avoid capacitive loading of the data bus and the output of the memory array is buffered with C/MOS drivers to meet the 3.5 V_{IH} requirement for the MK 3850 CPU.

MK 3853 APPLICATION



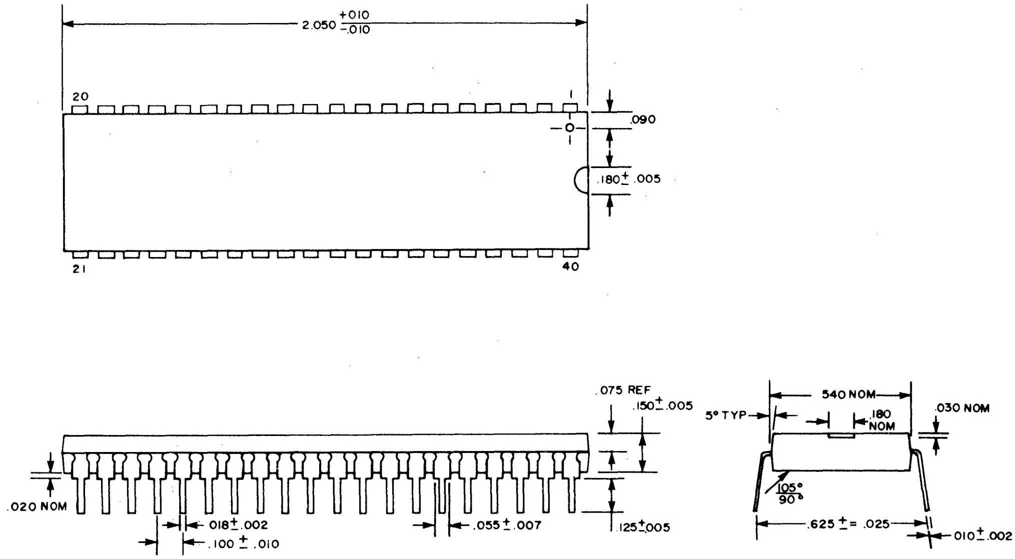
RECOMMENDED MOSTEK MEMORIES:

Device	Organization	Type	Access
MK 2708	1K x 8	STATIC E PROM	450nS
MK 4102-1	1K x 1	STATIC RAM	450nS
MK 4102-11 (Low Power)	1K x 1	STATIC RAM	450nS

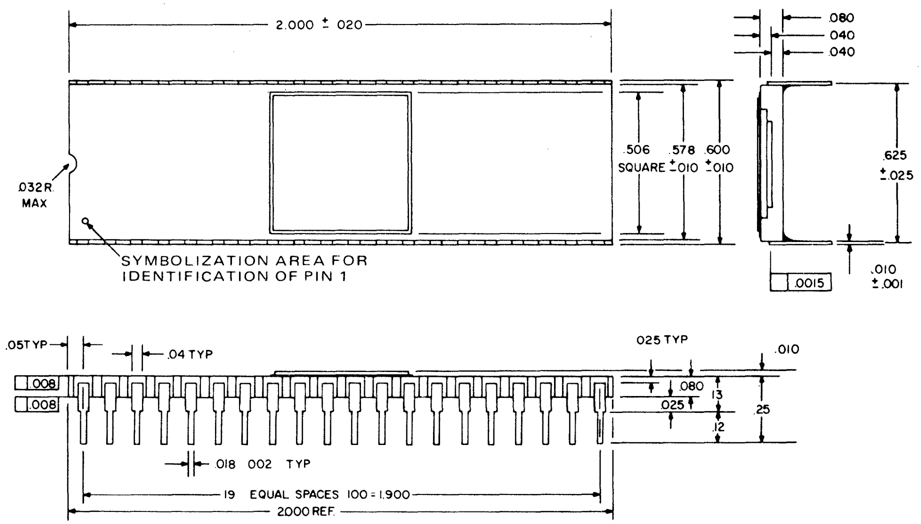
Figure 14

PACKAGE DESCRIPTION

40-lead plastic package



40-lead side-braze ceramic package



ORDERING INFORMATION		
MK3853P	0°C To 70°C	Ceramic
MK3853N	0°C To 70°C	Plastic
MK3853P-10	−40°C To +85°C	Ceramic
MK3853N-10	−40°C To +85°C	Plastic
MK3853P-20	−55°C To +125°C	Ceramic
MK3853N-20	−55°C To +125°C	Plastic

MOSTEK®

F8 MICROCOMPUTER DEVICES

F8 Direct Memory Access MK3854

FEATURES

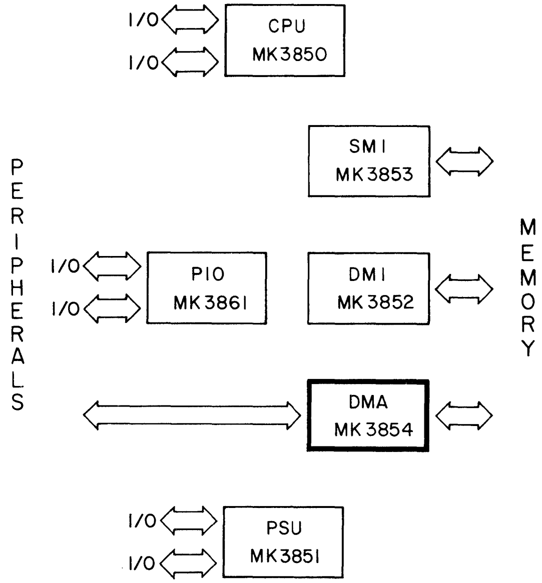
- □ 2 μsec cycle time
- □ Provides strobe for timing peripherals
- □ 16-bit address
- □ 12-bit byte count
- □ Control registers
- □ Port address selection
- □ +5V and +12V power supplies
- □ Low power dissipation—280mW

GENERAL DESCRIPTION

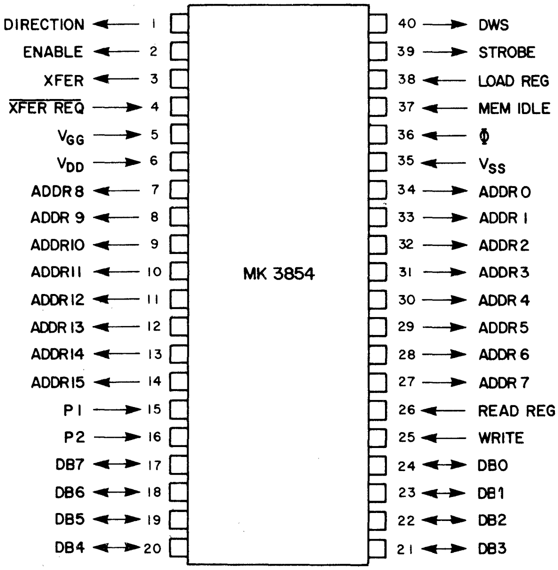
The MK 3854 Direct Memory Access (DMA) chip facilitates high speed data transfer between the main memory of an F8 system and peripherals. This transfer occurs without suspending normal operation of the processor, allowing DMA with no reduction of program execution speed. The MK 3854 DMA is manufactured using N-channel, Isoplanar MOS technology. Power dissipation is low, typically less than 280mW.

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data bus lines	Bidirectional three state
ADDR0-ADDR15	Address lines	Output three state
Φ, WRITE	Clock lines	Input
LOAD REG/ READ REG	Registers load/ read line	Input
P1, P2	Port address select	Input
MEM IDLE	Memory idle line	Input
XFER REQ	Transfer request line	Input
ENABLE, DIRECTION	Control status lines	Output
DWS, XFER	DMA Write slot, transfer	Output
STROBE	Output strobe line	Output
V_{SS}, V_{DD}, V_{GG}	Power lines	Input

F8 FAMILY



PIN CONNECTIONS



F8 Family

MK 3854 DMA FUNCTIONAL DIAGRAM

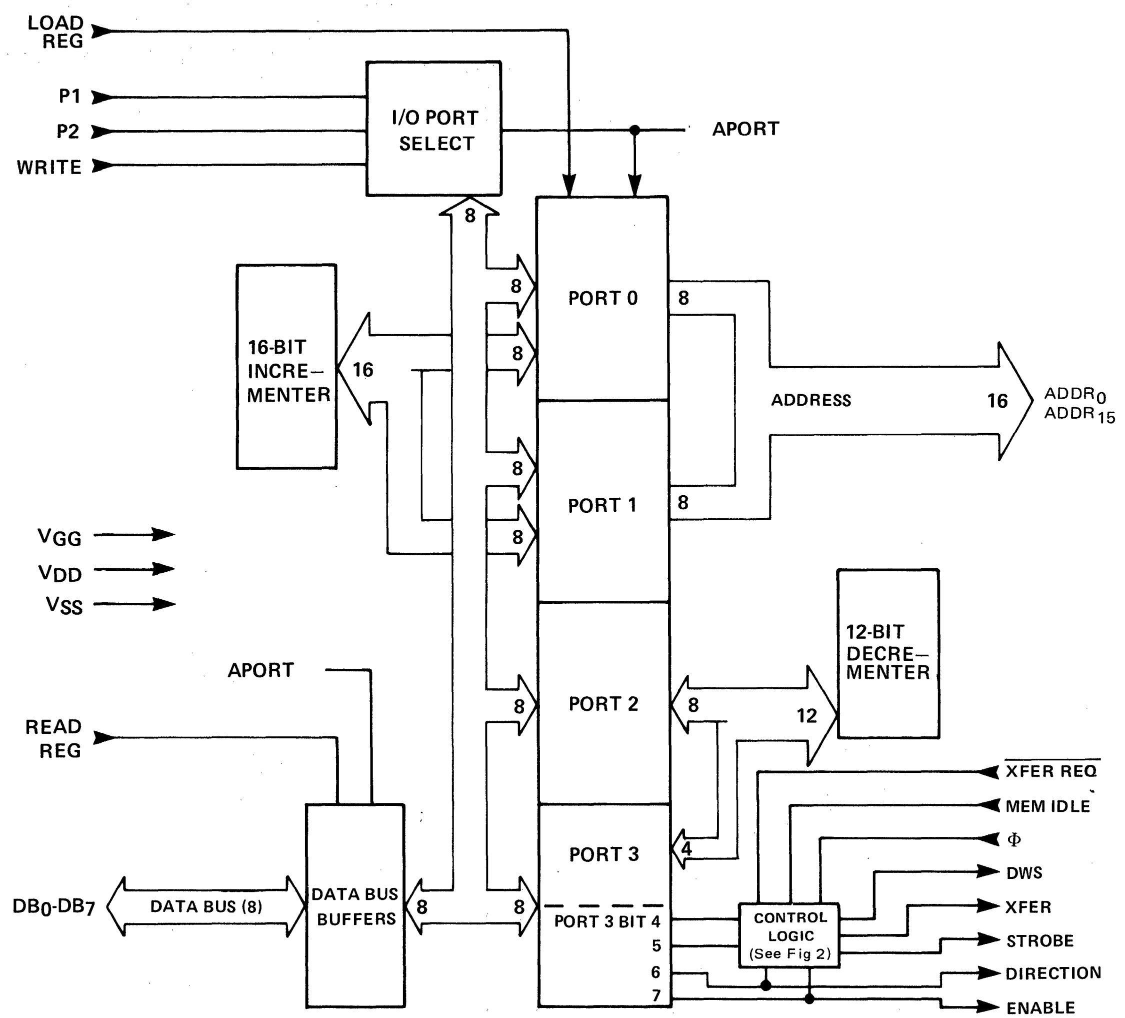


Figure 1

FUNCTIONAL PIN DESCRIPTION

Φand **WRITE** are clocks provided by the MK 3850 CPU. Φ is only used in the generation of STROBE. WRITE is only used for loading I/O ports and data bus monitoring for I/O match.

READ REG and LOAD REG are control signals that must be input to the MK 3854 DMA device in lieu of the five ROMC state signals. Since the MK 3854 DMA device only responds to ROMC states 1A and 1B, external logic must generate READ REG true for ROMC state 1B and LOAD REG true for ROMC state 1A, as follows:

READ REG = ROMC0-ROMC1-$\overline{\text{ROMC2}}$-ROMC3-ROMC4
LOAD REG = $\overline{\text{ROMC0}}$-ROMC1-$\overline{\text{ROMC2}}$-ROMC3-ROMC4

ADDR0 through ADDR15 are the 16 address lines which address the memory location to be accessed during the current DMA operation. This memory address originates in I/O ports 0 and 1 as illustrated in Figure 1. These lines are in a high impedance state when no DMA operation is taking place (XFER = 0).

MEM IDLE is a timing signal input to the MK 3854 DMA device from the MK 3852 DMI device. This signal is output high to identify time slots when memory is available for DMA access.

$\overline{\text{XFER REQ}}$ is a control signal which must be input to the MK 3854 DMA device by an external device which is controlling the DMA transfer rate (I/O port 3, bit 4 must be set to zero in this case). When low, this signal causes a byte of data to be transferred to or from memory during the next available DMA time slot. This signal is latched while MEM IDLE = 1. Changes during a DMA time slot are therefore ignored.

DB0 through DB7 are the bidirectional data bus lines which link the MK 3850 CPU with all other devices in the F8 system. Note that only data being transferred to or from one of the four MK 3854 I/O ports uses the data bus pins. Data being transferred to or from memory under DMA control completely bypasses the MK 3854 DMA device.

P1 and P2 must be strapped externally to determine the addresses of the four MK 3854 DMA device I/O ports as illustrated in the section titled 'I/O ports'.

XFER is a control output which identifies the time slots when a DMA data transfer is occurring . XFER is high whenever, MEM IDLE is high and other conditions specify that a DMA data transfer is to occur during the next available time slot. These conditions are that a DMA transfer is specified either by bit 4 of I/O port 3 being set to 1, or by XFER REQ being low while DMA has been enabled and the currently executed instruction is not attempting to access the DMA device's I/O ports. ENABLE is provided by I/O port 3 bit 7. DMA data transfers are inhibited while an instruction is accessing the I/O ports of the MK 3854 DMA device since these instructions may be in the process of modifying the parameters that control the DMA operation. This inhibit is generated by ANDing the LOAD REG input with an internal I/O port selected signal.

DIRECTION is a control output which reflects the contents of I/O port 3, bit 6. When high, data is being written into memory. When low, data is being read from memory.

ENABLE is a control output which reflects the contents of I/O port 3, bit 7. When high, DMA data transfers may occur. When low, DMA is disabled.

DWS is a DMA write slot signal. It is the logical AND of XFER and DIRECTION, thus it is true during any DMA write to memory.

STROBE is a DMA transfer signal output that is used for strobing data and for generating RAM WRITE. STROBE is high only during the second occurrence of Φclock high after MEM IDLE goes true, provided that XFER is also true.

DEVICE ORGANIZATION

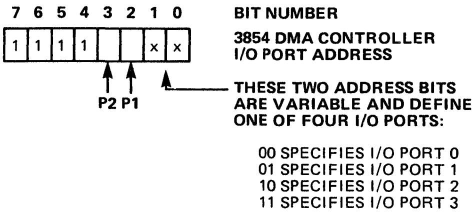
This section describes the operation of the basic functional elements of the MK 3854 DMA. These elements are shown on the DMA block diagram (fig. 1).

I/O PORTS

The MK 3854 DMA controller has four 8-bit registers which are addressed as I/O ports.

Since there may be up to four DMA controllers in an F8 system, 16 I/O port addresses are reserved for the exclusive use of DMA controllers, as shown in Table 1.

The four I/O port address used by a DMA are defined by the two signals (P1 and P2) which are input to the DMA controller and become bits 2 and 3 of the I/O port address. This may be illustrated as follows:



MK 3854 DMA I/O PORT ADDRESSES

FUNCTION OF I/O PORT	FIRST 3854	SECOND 3854	THIRD 3854	FOURTH 3854
Address, L.O. Byte (PORT0)	F0	F4	F8	FC
Address, H.O. Byte (PORT1)	F1	F5	F9	FD
Count, L.O. Byte (PORT2)	F2	F6	FA	FE
Count, H.O. Four bits, and Control (PORT3)	F3	F7	FB	FF

Table 1

The four I/O ports are not initialized during the power on reset.

DMA CONTROL LOGIC

This logic provides the control signals required to implement DMA data transfers. Figure 2 shows the detailed logic that generates these control signals.

LOAD REG/READ REG

The LOAD REG and READ REG signal inputs to the DMA require special mention.

Most F8 support devices have a control unit which decodes the five ROMC signals output by the MK 3850 CPU. However, the MK 3854 DMA controller will only respond to ROMC states 1A and 1B, which are "write to I/O port" and "read from I/O port" controls, respectively. All other states constitute "No Operations". Therefore, instead of having a control unit, external logic is used to decode these ROMC state signals, creating READ REG in response to state 1B, and LOAD REG in response to state 1A.

INCREMENT AND DECREMENT LOGIC

This logic is used to increment the address in ports 0 and 1 and to decrement the byte count in ports 2 and 3.

THE DATA AND ADDRESS BUSSES

Note carefully that whereas the address bus is used to output the address of the memory location which will be accessed during the next DMA operation, MK 3854 DMA controller's connection to the data bus is used only to transfer data between MK 3854 DMA device I/O ports and the CPU. The data bus is not used to transfer data bytes during a DMA operation.

DMA CONTROL SIGNALS

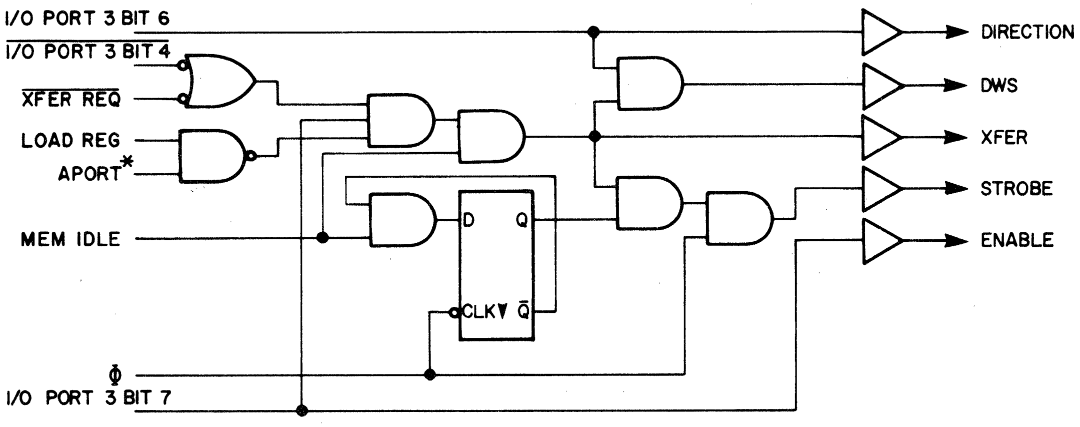


Figure 2

*AN I/O PORT WAS BEING SELECTED

OPERATIONAL DESCRIPTION

The MK 3854 DMA device makes use of time slots during which the CPU is not accessing memory. During these time slots, the MK 3854 DMA device generates data transfer control signals which enable data to be read out of memory, or to be written into memory. The MK 3852 DMI device outputs the MEM IDLE signal to identify time slots available for DMA access.

In addition to providing data transfer control signals, the MK 3854 DMA controller outputs the address of the memory location which is to be accessed.

The four I/O ports of a DMA device must be loaded with appropriate data to control the DMA operation. I/O ports are loaded using OUT instructions. The contents of I/O ports may be read at any time using IN instructions.

Before a DMA operation starts the beginning address of the memory buffer from which data will be read, or to which data will be written, must be loaded into I/O ports 0 and 1. I/O ports 2 and 3 are used to define the length of the memory buffer which is to be accessed plus various DMA options and controls, as illustrated in Figure 3.

With reference to Figure 3, observe that 12 bits are set aside to define the memory buffer length (byte count), therefore memory buffers up to 4096 bytes in length may be written into or read via DMA. A byte count of 01 transfers one byte; a count of 00 transfers 4096 bytes.

Bit 7 of I/O port 3 may be used at any time to start or stop DMA operations. During normal initiation sequence this bit will be zero while I/O ports 0, 1 and 2 are loaded with appropriate data. Then in order to initiate the DMA operations, I/O port 3 will be loaded with a data byte that includes a 1 in the high order bit. However, in the case of repeated block transfers, it may only be necessary to reload port 3, and port 2 will hold zero and the contents of port 0 and 1 will be the address of the last byte previously transferred plus 1.

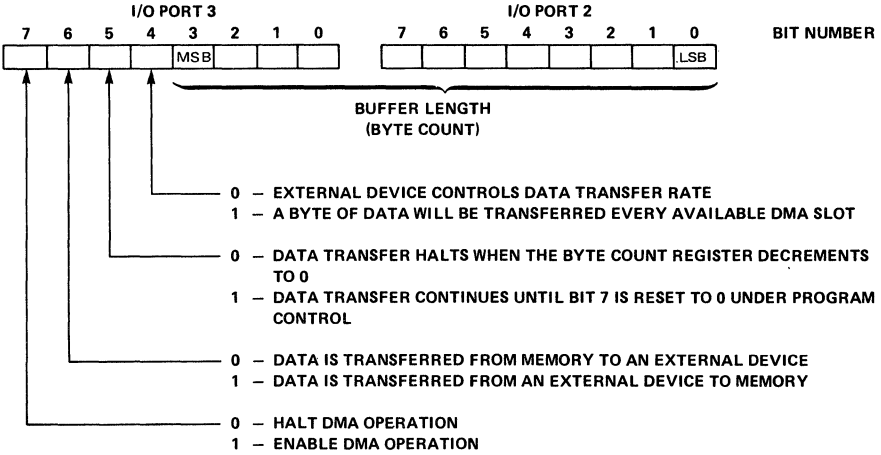
The direction of the DMA data transfer is determined by bit 6 of I/O port 3. If this bit is zero, data will be read out of memory by the external device. If this bit is one, data will be written into memory by the external device.

The rate of DMA data transfer is determined by bit 4 of I/O port 3. If this bit is zero, then the external device must provide a transfer request ($\overline{\text{XFER REQ}}$) signal whenever it is ready for a DMA data transfer. The actual data transfer will then occur during the next DMA slot, as identified by MEM IDLE high. The external device controls DMA transfer rate in this mode. If bit 4 of I/O port 3 is 1, the MK 3854 DMA controller assumes that external logic is ready for a DMA transfer whenever MEM IDLE high identifies a DMA slot. In this mode, the F8 system controls DMA transfer rate.

Each time a DMA data transfer occurs, logic within the MK 3854 DMA controller that is clocked by XFER increments the memory address in I/O ports 0 and 1 and decrements the byte counter in I/O ports 2 and 3. If bit 5 of I/O port 3 is zero, then DMA transfer will automatically halt and clear bit 7—the enable bit—as soon as the byte counter is decremented to zero. If bit 5 of I/O port 3 is 1, however, the byte count is ignored and DMA data transfer will continue until halted by an OUT instruction setting bit 7 of I/O port 3 to zero. If continuous DMA data transfer is specified by bit 5 of I/O port 3 being set to 1, then the memory address in I/O port 0 and 1 will still be incremented and the byte counter decremented after each DMA access even though the byte counter is ignored.

DMA registers are loaded and read when the MK 3850 CPU executes I/O instructions that access the DMA registers. The I/O instructions use the DATA BUS to transmit the I/O address in one instruction cycle and to transfer data during the following instruction cycle. The appropriate control signal, LOAD REG or READ REG, will become active during this second cycle. The DMA will load one of its registers during a cycle with LOAD REG high if the I/O address, which had been on the data bus during the previous cycle, matched a DMA port address. The register is loaded and the address comparator is up-dated by the WRITE clock. These are the only functions of WRITE in the MK 3854 DMA. Likewise a DMA chip will drive the contents of a selected register onto the DATA BUS only while READ REG is high if there was a similar address match during the prior cycle. I/O address assignment is made using pins P1 and P2.

USE OF PORT 2 AND PORT 3 AS DMA CONTROLS



0 – EXTERNAL DEVICE CONTROLS DATA TRANSFER RATE
1 – A BYTE OF DATA WILL BE TRANSFERRED EVERY AVAILABLE DMA SLOT

0 – DATA TRANSFER HALTS WHEN THE BYTE COUNT REGISTER DECREMENTS TO 0
1 – DATA TRANSFER CONTINUES UNTIL BIT 7 IS RESET TO 0 UNDER PROGRAM CONTROL

0 – DATA IS TRANSFERRED FROM MEMORY TO AN EXTERNAL DEVICE
1 – DATA IS TRANSFERRED FROM AN EXTERNAL DEVICE TO MEMORY

0 – HALT DMA OPERATION
1 – ENABLE DMA OPERATION

Figure 3

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS (Above which useful life may be impaired)

V_{GG}	+15V to –0.3V
V_{DD}	+7 to –0.3V
All other Inputs and Outputs	+7V to –0.3V
Storage Temperature	–55°C to + 150°C
Operating Temperature	0°C to + 70°C

Note: All voltages with respect to V_{SS}.

DC CHARACTERISTICS: V_{SS} = 0V, V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0 to + 70°C

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		20	40	mA	f = 2MHz, Outputs Unloaded
I_{GG}	V_{GG} Current		15	28	mA	f = 2MHz, Outputs Unloaded

DC SIGNAL CHARACTERISTICS

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
DATA BUS (DB0-DB7)	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –100 µA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6mA
	I_{IH}	Input High Current		1	µA	V_{IN} = 6V, three-state mode
	I_{IL}	Input Low Current		–1	µA	V_{IN} = V_{SS} three-state mode
ADDRESS LINES	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –1 mA
(ADDR0-ADDR15)	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 3.2 mA
	I_L	Leakage Current		1	µA	V_{IN} = 6V, three-state mode
ENABLE, DIRECTION	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –100 µA

Table 2

(continued)

(Table 2 continued)

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
DWS (DMA WRITE SLOT), XFER, STROBE	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	$I_{OL} = 1.6$ mA
	I_L	Leakage Current		1	µA	$V_{IN} = 6V$
MEM IDLE, $\overline{\text{XFER REQ}}$	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		1	µA	$V_{IN} = 6V$
LOAD REG, READ	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
REG, P1, P2	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		1	µA	$V_{IN} = 6V$
WRITE, Φ	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		1	µA	$V_{IN} = 6V$

NOTE: Positive current is defined as conventional current flowing into the pin referenced.

Table 2

AC CHARACTERISTICS

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	NOTES
PΦ	Φ Clock Period	.5		10	µs	Note 1
PW1	Φ Pulse Width	180		PΦ –180	ns	t_r, t_f = 50ns typical
t_{d1}	Φ to WRITE ↑ Delay	0		250	ns	Note 1
t_{d2}	Φ to WRITE ↓ Delay	0		200	ns	Note 1
PW2	WRITE Pulse Width	P Φ–100		P Φ	ns	t_r, t_f = 50ns typical
t_1	WRITE ↓ to CYCLE REQ ↑	P Φ+100-t_{d2}		P Φ +300-t_{d2}	ns	Note 4
t_2	WRITE ↓ to ENABLE & DIRECTION ↑			450	ns	
t_3	MEM IDLE ↓ to ENABLE ↓			400	ns	
t_4	$\overline{\text{XFER REQ}}$ ↓ to MEM IDLE ↑ Set-up	200			ns	
t_5	MEM IDLE ↑ to ADDR Valid	50	200	300	ns	$C_L = 500$ pF
t_6	MEM IDLE ↓ to ADDR Hi-Z	30		250	ns	$C_L = 500$ pF
t_7	MEM IDLE ↑ to XFER & DWS ↑	50		300	ns	$C_L = 50$ pF
t_8	MEM IDLE ↓ to XFER & DWS ↓	50		300	ns	$C_L = 50$ pF
t_9	MEM IDLE ↑ to STROBE ↑	600		$\frac{3P\Phi}{2} + 100$	ns	$C_L = 50$ pF
t_{10}	STROBE Pulse Width	200		$\frac{P\Phi}{2} + 30$	ns	$C_L = 50$ pF
t_{11}	DB Input Set-up Time	300			ns	
t_{12}	WRITE ↓ to READ/LOAD REG ↑			600	ns	
t_{13}	READ REG ↑ to DB Valid	40		300	ns	$C_L = 100$ pF
t_{14}	WRITE ↓ to MEM IDLE ↑	2P Φ +50-t_{d2}		2PΦ+300-t_{d2}	ns	Short Cycle
t_{15}	WRITE ↓ to MEM IDLE ↓	4P Φ+50-t_{d2}		4PΦ+300-t_{d2}	ns	Short Cycle
t_{16}	XFER & DWS ↓ to CYCLE REQ ↑	0		400	ns	Note 3

Table 3

1. These specifications are those of Φ and WRITE as supplied by the MK 3850 CPU.
2. Input and Output capacitance is 3 to 5pF typical on all pins except V_{DD}, V_{GG}, and V_{SS}.
3. If the next Cycle Req ↑ initiates a new read, XFER ↓ can be used to clock DMA read data into the peripheral.
4. Cycle Req is output by the MK 3852 DMI to initiate a memory READ/WRITE cycle.

MK 3854 DMA DEVICE TIMING

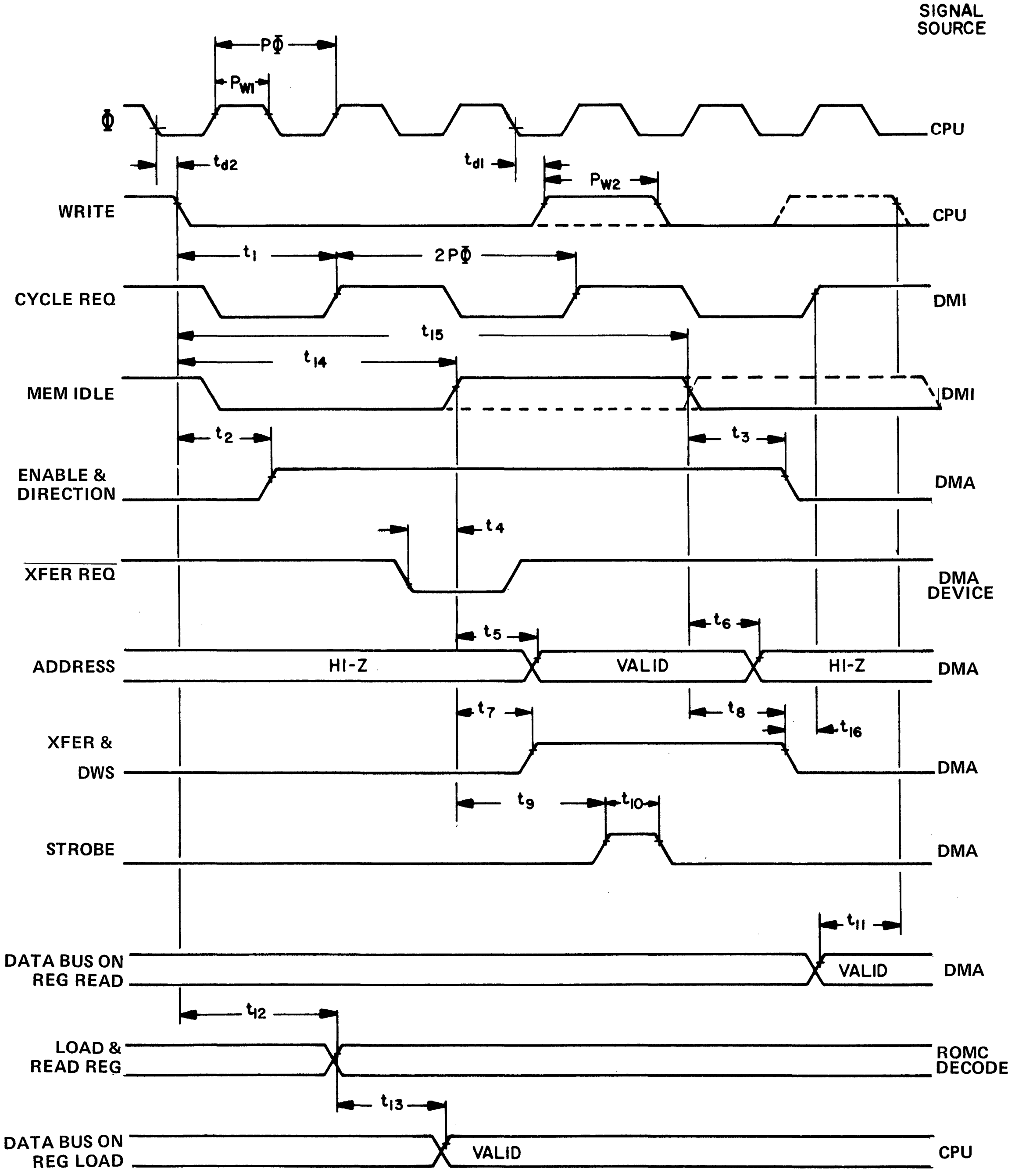
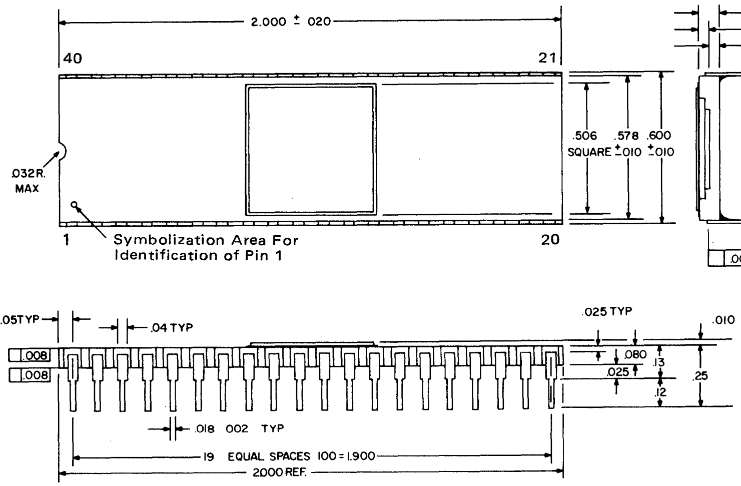
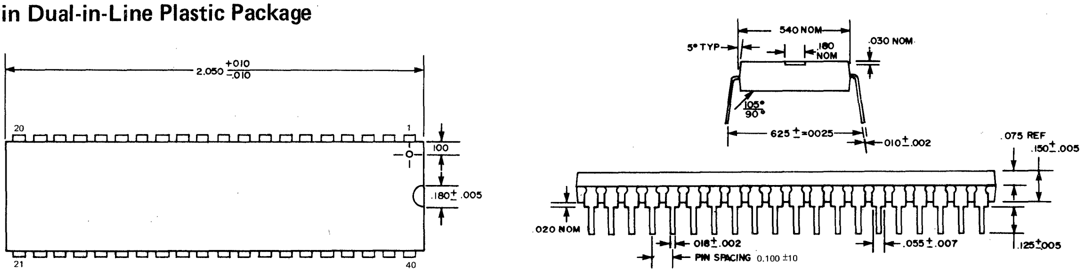


Figure 4

PACKAGE DESCRIPTION — 40-Pin Dual-in-Line Ceramic Package



40-Pin Dual-in-Line Plastic Package



ORDERING INFORMATION

PART NUMBER	PACKAGE	TEMPERATURE RANGE
MK3854(N)	Plastic	0° C to +70° C
MK3854(P)	Ceramic	0° C to +70° C
MK3854(N)-10	Plastic	-40° C to +85° C
MK3854(P)-10	Ceramic	-40° C to +85° C

MOSTEK®

F8 MICROCOMPUTER DEVICES

Peripheral Input/Output MK 3861

FEATURES

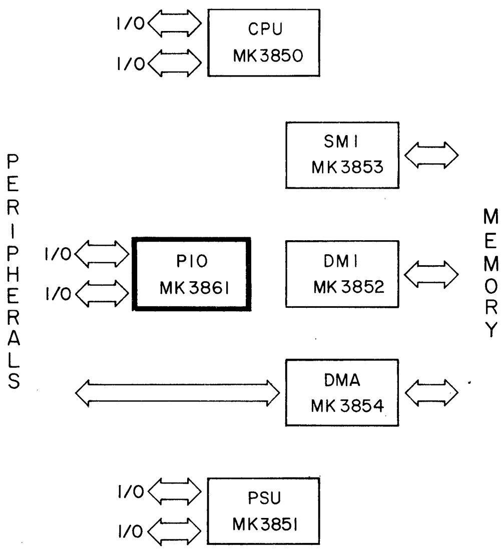
- □ Two 8-bit I/O ports
- □ Programmable timer
- □ External/timer interrupt control circuitry
- □ Low power dissipation-typically less than 200mW

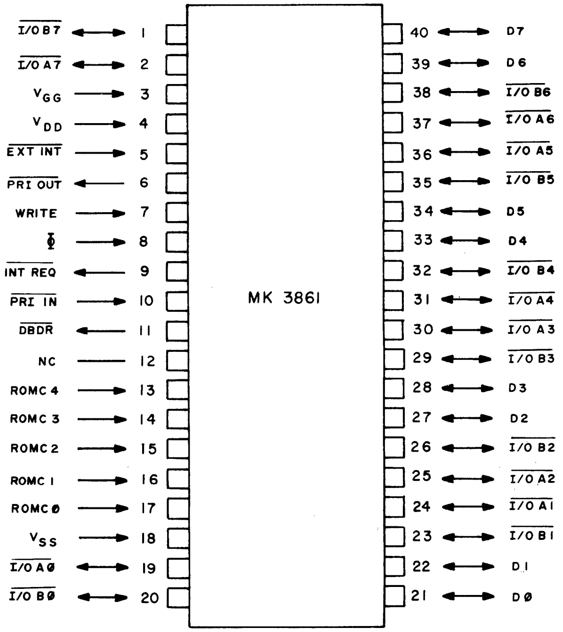
GENERAL DESCRIPTION

Each 3861 Peripheral Input/Output Circuit (PIO) provides two 8-bit I/O ports, a programmable timer and a vectored timer or external interrupt for the F8 system. The timer, I/O ports and interrupt circuitry are identical to those of the MK 3851 PSU. The 3861 may be used to provide extra I/O, timer, and interrupt functions compatible with those of the 3851 PSU, or the 3861 may be used as the only I/O peripheral in non PSU systems. This circuit in conjunction with the 3853 and standard PROM is particularly useful in prototyping a PSU system. The 3853 MI circuit along with standard PROM can emulate the memory functions of the PSU while the 3861 provides the I/O, interrupt, and timer features of the PSU. The 3861 is manufactured using the same high performance N-channel Isoplanar technology as the F8 CPU.

PIN NAME	DESCRIPTION	TYPE
D0-D7	Data Bus Lines	Bi-directional, Tri-State
I/O A0 - I/O A7	I/O Port A	Bi-directional
I/O B0 - I/O B7	I/O Port B	Bi-directional
ROMC0-ROMC4	System Control Lines	Input
ϕ, WRITE	Clock Lines	Input
EXT INT	External Interrupt	Input
PRI IN	Priority In	Input
PRI OUT	Priority Out	Output
INT REQ	Interrupt Request	Output
DBDR	Data Bus Drive	Output
V_{SS}' V_{DD}' V_{GG}	Power Lines	Input

F8 Family





F8 Family

FUNCTIONAL PIN DEFINITION

D0-D7 (BI-DIRECTIONAL, TRI-STATE)

DATA BUS: The Data Bus provides bi-directional communication between the F8 CPU and the 3861 and all other peripheral circuits for transfer of data. D0 is the least significant bit.

$\overline{\text{I/O A0}}$ – $\overline{\text{I/O A7}}$ and $\overline{\text{I/O B0}}$ – $\overline{\text{I/O B7}}$ (Bidirectional)

I/0 PORTS: Two 8-bit I/0 ports are located on the 3861 PIO. These ports are referred to as Port A and Port B herein, but the actual port number is determined by the version of the 3861 that is selected. These ports have output latches to hold output data, and hysteresis circuits are provided to add input noise immunity. Bit 0 of each port is the least significant bit.

ROMC0 – ROMC4 (INPUT)

SYSTEM CONTROL LINES: These lines provide the 3861 with control information from the F8 CPU. The CPU sets up these lines early in each machine cycle, and the PIO executes that command during that cycle.

Φ(INPUT)

Φ (PHI) CLOCK: This is the high frequency F8 system clock. It is generated by the F8 CPU. Each machine cycle contains either 4 Φ periods (short cycle) or 6 Φ periods (long cycle).

WRITE (INPUT)

WRITE CLOCK: This clock defines the machine cycle. The cycle starts with the fall of the WRITE clock. The system control lines become stable shortly after the start of the cycle and the PIO decodes and executes the command communicated by the control lines. All ROMC commands are started and completed within one cycle of WRITE.

$\overline{\text{EXT INT}}$ (INPUT)

EXTERNAL INTERRUPT: When an external circuit pulls this input "low" an external interrupt request will be latched into the PIO if its interrupt control register has been set up to allow external interrupts. The PIO will subsequently communicate this interrupt request to the CPU via its $\overline{\text{INT REQ}}$ line.

$\overline{\text{PRI IN}}$ (INPUT)

PRIORITY IN: This input signals the PIO that a higher priority peripheral has an interrupt request impending on the CPU. If the PIO has already requested an interrupt, it will maintain that request, but it will not be serviced by the CPU until its $\overline{\text{PRI IN}}$ input is in the "low" state. If an interrupt is received, it will be latched into the PIO but it will not be serviced until $\overline{\text{PRI IN}}$ is in the "low" state.

$\overline{\text{PRI OUT}}$ (OUTPUT)

PRIORITY OUT: This output signals lower priority peripherals that the PIO either has an interrupt request impending on the CPU, or that a still higher priority peripheral has requested an interrupt.

$\overline{\text{INT REQ}}$ (OUTPUT)

INTERRUPT REQUEST: This open drain output is wired ANDed with the corresponding output on all other peripherals to form the interrupt request input to the CPU.

$\overline{\text{DBDR}}$ (OUTPUT)

DATA BUS DRIVE: This output goes "low" whenever the PIO is driving the Data Bus as an output. It may be used to control tri-state buffers in a buffered Data Bus system and to signal other peripherals that the PIO has "control" of the Data Bus at that time.

V_{SS} (INPUT)

V_{SS}: This is system ground (0V.) V_{DD} and V_{GG} are referenced to V_{SS}.

V_{DD} (INPUT)

V_{DD}: Power line; +5V ± 5%.

V_{GG} (INPUT)

V_{GG}: Power line; +12V ± 5%.

PIO ARCHITECTURE

Figure 3.0.1 shows the various functional blocks and registers. The 3861 uses the clock signals 'Φ' and 'WRITE', which are generated by the CPU to control timing functions within the circuit. It also uses the contents on five control lines (ROMC's) as various commands to be performed within each cycle. A control ROM within the PIO decodes the five control lines and provides control within the circuit.

ADDRESSABLE PORTS

The 3861 has four addressable ports. They are linked to the accumulator of the CPU by the I/O instructions. Each port is referenced by an 8-bit address. The upper six bits of the address refer to the circuit on which the ports are located while the lower two bits select one of the four ports; hence, the port addresses are referred to as X0,X1,X2 and X3, where X is a six-bit binary number determined by the particular version of the 3861 that is selected. Each port on the device may be writtten into using output instructions. The contents of the I/O ports may be read using input instructions. These instructions initiate the transfer to contents between ports and the accumulator on the CPU. In the PIO circuit, two ports are used as 8-bit I/O ports. The remaining two ports are the 8-bit timer and the local interrupt control port. Table 3.1.1 lists the addressable ports and their respective functions.

PIO FUNCTIONAL DIAGRAM

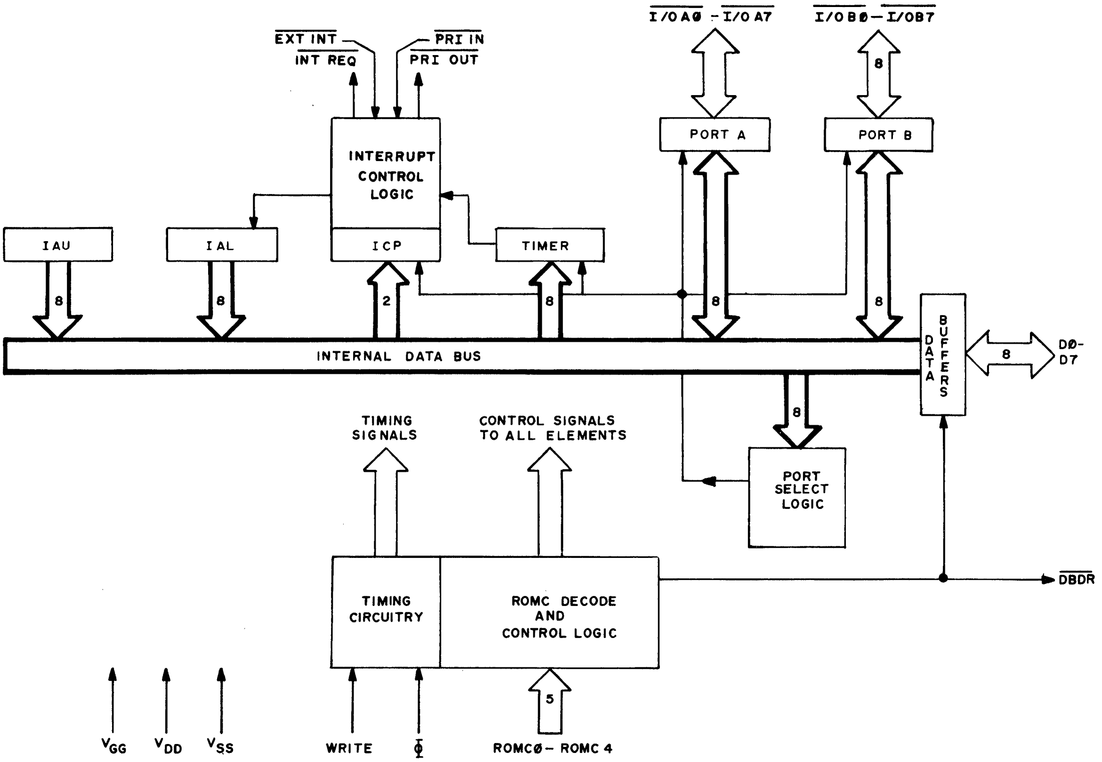


Figure 3.0.1

Table 3.1.1
3861 PIO PORT ASSIGNMENTS

PORT ADDRESS	
X00	PIO I/O Port A (READ–WRITE)
X01	PIO I/O Port B (READ-WRITE)
X10	PIO Local Interrupt Control (WRITE ONLY)
X11	PIO Timer (WRITE ONLY)

INPUT/OUTPUT PORTS

Each 3861 chip has two bidirectional 8-bit I/O ports. Each port's address, using binary notation, is XXXXXX00 or XXXXXX01, where the X binary digits are the chip's unique I/O port select code. Every 3861 used in a system must have a unique I/O port select code.

TIMER

The 3861 has a local timer to generate program initiated delays. To the programmer, the timer is an 8-bit register, addressable via F8 output instructions to the specified timer port address. Delay codes, calculated by the assembler, are loaded into the accumulator and then transferred to the timer (a polynomial shift register). An output instruction to the timer port number performs this function. After it is loaded, the timer counts down. A table of delay codes matched to delay times appears in the Appendix A.

The timer runs continuously. It signals the interrupt control circuitry after each timer cycle (3.953 ms in a 2MHz system). However, when an output instruction is executed with the timer port number as the operand, the timer is jammed with a specific count and the local interrupt control logic clears any stored timer interrupt. The timer then counts down from that count in a polynomial sequence (Appendix A) and generates an internal interrupt request when a count of H'FE' is reached. From that point, the timer continues to cycle every 3.953 milliseconds (for a 2MHz system) unless it is re-loaded as described above. If the interrupt is not set for timer interrupts, a timer initiated interrupt will be stored by the local interrupt control circuitry. When the local interrupt control logic is finally set to allow timer interrupts, the PIO will request interrupt service.

INTERRUPT INTERCONNECTION

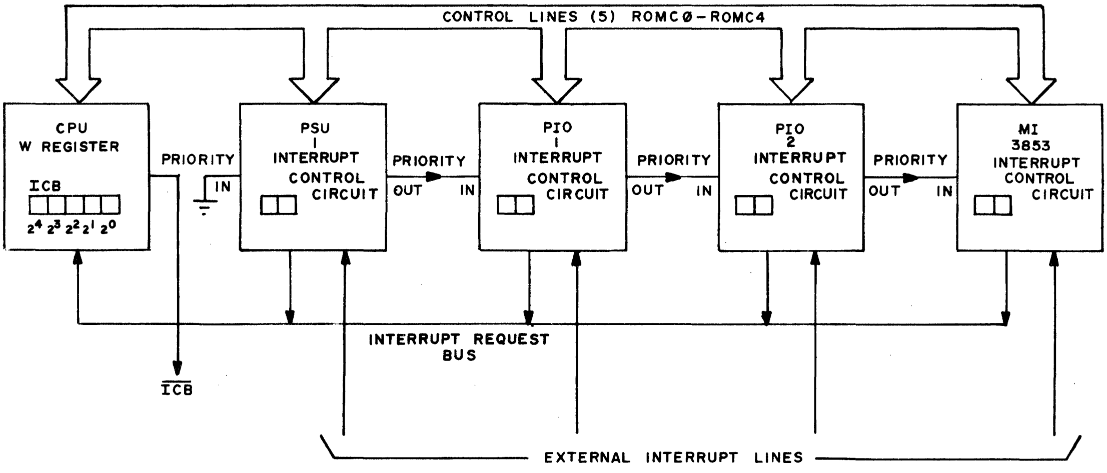


Figure 3.4.1

Time delays between 0 and 254 counts may be chosen. The timer is decremented once every 31 Φ clock cycles. Therefore, the counter may count as high as 7905 Φ clock cycles. (For a system at 2MHz, a clock cycle occurs every 500ns). Longer durations are achieved by counting multiple time interrupts. If the timer is loaded with all one's, it will stop counting.

INTERRUPT CONTROL PORT AND LOGIC

Figure 3.4.1 is a block diagram of the interrupt interconnection for a typical F8 system. The 3861 PIO, has either of two types of interrupts, internal or external. The internal interrupt may be generated by the programmable timer while the external interrupt is generated by external logic in the system. A local interrupt control cicuit containing two latches is included on each device. These latches are the Select Bit and the Interrupt Enable Bit.

Table 3.4.1

LOCAL INTERRUPT CONTROL BITS

2^1	2^0
Select Bit	Interrupt Enable Bit

These two bits have four possible states:

Select Bit	Interrupt Enable Bit	Function
0	0	No Interrupt
0	1	External Interrupt Enabled
1	0	No Interrupt
1	1	Timer (Internal) Interrupt Enabled

These control latches are loaded under program control using an output instruction. This loading clears the interrupt control logic, except for any pending timer interrupt. The operand for the OUT or OUTS instruction must be the predefined port number of the Interrupt Control Port (ICP). The two control bits allow each interrupt circuit to have independently controlled enable/disable capabilities. If enabled, the select bit may choose either internal (timer generated) interrupts or external interrupts.

Each PIO has a $\overline{\text{PRIORITY IN}}$ and a $\overline{\text{PRIORITY OUT}}$ line so that they may be daisy chained together in any order, to form a priority level of interrupts. When a PIO receives an interrupt (either timer or external) it pulls its $\overline{\text{PRI OUT}}$ output high, signaling all lower priority peripherals that it has a higher priority interrupt request impending on the CPU. Also when the PIO's $\overline{\text{PRI IN}}$ input is pulled high by a higher priority peripheral, signaling the PIO that there is a still higher priority interrupt request, it passes that signal along by pulling its $\overline{\text{PRI OUT}}$ high. When the CPU processes an interrupt request it commands the interrupting peripheral to place its interrupt vector address on the Data Bus. Only that peripheral whose $\overline{\text{PRI IN}}$ is low and who has an interrupt request impending will respond. Should there be another lower priority peripheral with an impending request, it will not respond at that time because its $\overline{\text{PRI IN}}$ input will be high.

To generate a timer interrupt, the timer must be set under program control. The PIO generates a timer interrupt request when the timer times out AND the interrupt control has been set (Select Bit = 1, Enable Bit = 1). The CPU will not process the request until 1, it is enabled to handle interrupts by setting the ICB bit in the status register, and 2, it has completed processing all higher priority interrupt requests. The timer may time out before ICB is set or the local interrupt control is enabled for internal interrupts; however, an interrupt will still be initiated after the required conditions have been met. Any pending timer interrupt is cleared whenver output instructions load the timer. The ICB is always cleared after the CPU has acknowledged an interrupt request.

The generation of an external interrupt request is also controlled by the local interrupt control circuit. If the Select Bit is set to zero and the Enable Bit is set to one, the control logic of the chip is responsive to the external interrupts. To guarantee an interrupt, the external interrupt line must drop from 1 (near V_{DD}) to 0 (near V_{SS}), and stay at zero for a minimum of two WRITE clock periods (4μs for a 500 ns system clock). The ICB may or may not be set when this occurs. If it is not set, the request will be stored by the local interrupt control logic until the ICB is reenabled; however, the stored external interrupt request will be lost whenever the control bits are reloaded. However, loading the control bits does not clear a stored timer interrupt. The stored external interrupt request will be cleared after that interrupt is serviced.

Within each local interrupt control circuit there is a 16-bit interrupt address vector. This vector is the address to which the program counter will be set after an interrupt is acknowledged; hence, it is the address of the first executable instruction of the interrupt routine. The 3861 has an interrupt address which is particular to the version of the 3861 selected by the user. Fifteen bits are fixed. These are bits 0 through 6 and 8 through 15. Bit seven (2^7) is dependent upon the type of interrupt. This bit will be a 0 for internal timer generated interrupts and a 1 for external interrupts. When the interrupt logic sends an interrupt request signal and the CPU is enabled to service it, the normal state sequence of the CPU is interrupted at the end of an instruction. The CPU signals the interrupt circuits via the five control lines. The requesting local interrupt circuit sends a 16-bit interrupt address vector (from the interrupt address generator) onto the Data Bus in two consecutive bytes. The address is made available to the program counter via the address demultiplexer circuits. Simultaneously, the address is also made available to all other devices connected to the data bus. It is the address of the next instruction to be executed. The program counter (PC0) of each memory device is set with this new address while the stack register (PC1) is loaded with the previous contents of the program counter. The information in PC1 is lost. Thus, the next instruction to be executed is determined by the value of the interrupt address vector.

The Interrupt Control Bit (ICB) of the CPU (loaded in the W register) allows interrupts to be recognized. Clearing the ICB prevents acknowledgement of interrupts. The ICB is cleared during power on, external reset, and after an interrupt is acknowledged. The interrupt status of the PSU, PIO or MI devices are not affected by the execution of the DISABLE INTERRUPT (DI) instruction. At the conclusion of most instructions, the fetch logic checks the state of the Interrupt Request Line. If there is an interrupt, the next instruction fetch cycle is suspended and the system is forced into an interrupt sequence.

The CPU allows interrupts after all F8 instructions except the following:

(PK)	PUSH K
(PI)	PUSH IMMEDIATE
(POP)	POP
(JMP)	JUMP
(OUTS)	OUTPUT SHORT (Excluding OUTS 00 and 01)
(OUT)	OUTPUT
(EI)	SET ICB
(LR W, J)	LOAD THE STATUS REGISTER FROM SCRATCHPAD
POWER ON	

As a result, it is possible to perform one more instruction after the above CPU instructions without being interrupted.

DATA FLOW

Table 3.5.1 shows the function performed by the PIO for each ROMC command. Each function is entirely performed within one machine cycle (one cycle of the WRITE clock)

TABLE 3.5.1

The following ROMC states are decoded by the 3861 as indicated. All other ROMC states are decoded as "NO–OPERATION' (NO–OP).

Binary					Hex	3861 FUNCTION
ROMC4	ROMC3	ROMC2	ROMC1	ROMC0		
0	1	1	1	1	0F	If this circuit is interrupting and no higher priority circuit is interrupting, move the lower half of the interrupt vector on to the Data Bus and signal Bus use with $\overline{DBDR}$.
1	0	0	0	0	10	Place interrupt circuitry on an inhibit state that prevents altering the interrupt chain.

1 0 0 1 1	13	If this circuit is interrupting and no higher priority circuit is interrupting move the upper half of the interrupt vector on to the Data Bus and signal Bus use with $\overline{\text{DBDR}}$. In any case, remove priority interrupt circuitry from inhibit state.
1 1 0 1 0	1A	If contents of Data Bus in the previous were an address of an I/O port, the timer, or the Interrupt Control Port, move current contents of the Data Bus into that port. (Output Command).
1 1 0 1 1	1B	If contents of Data Bus in the previous cycle was an I/O port address, move the contents of that port on to the Data Bus and signal Bus use with $\overline{\text{DBDR}}$ (Input Command).

3861 PIO VERSIONS

Each version of the 3861 is denoted by a MK 90----- number. This ninety thousand series number should be used when ordering or specifying a 3861 to insure that the proper version is understood. Thus, the complete part designation of a particular version of the 3861 is:

3861
MK90-------

The presently available versions of the 3861 are listed in table 4.0.1.

AVAILABLE VERSIONS OF THE 3861
TABLE 4.0.1

VERSION	PORT SELECT CODE	PORT NUMBERS (DERIVED FROM THE PORT SELECT CODE; HEX)	PORT OUTPUT TYPE	INTERRUPT ADDRESS	
				TIMER	EXTERNAL
MK 90001	000001	04 thru 07	Standard T^{2}Compatible	0600	0680
MK 90002	000010	08 thru 0B	Standard T^2 Compatible	0340	03C0
MK 90003	001000	20 thru 23	Standard T^2 Compatible	0320	03A0
MK 90004	001001	24 thru 27	Standard T^2 Compatible	0360	03E0
MK 90005	000001	04 thru 07	Standard T^2 Compatible	0020	00A0

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS*

V_{GG}, EXT INT	–.3V to +15V
V_{DD}	–.3V to +7V
I/O PORT OPEN DRAIN OPTION	–.3V to +15V
ALL OTHER INPUTS AND OUTPUTS	–.3V to +7V
STORAGE TEMPERATURE	–55°C to +150°C
OPERATING TEMPERATURE	0°C to 70°C

*All voltages are with respect to V_{SS}. Stresses above those listed may cause permanent damage to the device. Exposure to maximum rated stress for extended periods may impair the useful life of the device.

DC CHARACTERISTICS

V_{SS} = 0V, V_{DD} = 5V ± 5%, V_{GG} = 12V ± 5%
T_A = 0 to 70°C, unless otherwise noted.
Positive current is defined as conventional current flowing into the pin referenced.

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		25	60	mA	f= 2MHz, Outputs unloaded
I_{GG}	V_{GG} Current		8	15	mA	f= 2MHz, Outputs unloaded

DATA BUS (DB0-DB7)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	$I_{OH} = -100\mu A$
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.6mA [1]
I_{IH}	Input High Current	0		1	μA	V_{IN} = 6V, 3-State mode
I_{OL}	Input Low Current	0		–1	μA	$V_{IN} = V_{SS}$, 3-State mode
C_I	Input Capacitance			10	pF	3-State mode

CLOCK LINES (Φ WRITE)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	4.0		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			1	μA	V_{IN}= 6V
C_I	Input Capacitance			10	pF	

PRIORITY IN AND CONTROL ($\overline{\text{PRI IN}}$, ROMCO – ROMC4)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			1	μA	V_{IN} = 6V
C_I	Input Capacitance			10	pF	

PRIORITY OUT ($\overline{\text{PRI OUT}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	$I_{OH} = -100\mu A$
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	$I_{OL} = 100\mu A$

INTERRUPT REQUEST ($\overline{\text{INT REQ}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage				Volts	Open Drain Output [1]
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1mA
I_L	Leakage Current			1	μA	V_{IN} = 6V, Output device off
C_I	Input Capacitance			10	pF	Output device off

DATA BUS DRIVE ($\overline{\text{DBDR}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage					Open Drain Output
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1mA
I_L	Leakage Current			1	μA	V_{IN} = 6V, Output device off
C_I	Input Capacitance			10	pF	Output device off

F8
Family

EXTERNAL INTERRUPT ($\overline{\text{EXT INT}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5			Volts	Internal pullup exsists
V_{IL}	Input Low Voltage			1.2	Volts	
V_{IC}	Input Clamp Voltage			15	Volts	$I_{IH} = 185\mu A$
I_{IL}	Input Low Current	–250		–750	μA	$V_{IN} = V_{SS}$
C_I	Input Capacitance			10	pF	

I/O PORT OPTION A (STANDARD PULLUP)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	$I_{OH} = -30\mu A$
V_{OH}	Output High Voltage	2.9		V_{DD}	Volts	$I_{OH} = -100\mu A$
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	$I_{OL} =$ 2mA
V_{IH}	Input High Voltage	2.9		V_{DD}	Volts	Internal Pullup to V_{DD} [2]
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_{IL}	Input Low Current			–1.2	μA	$V_{IN} = .4V$[3]
C_I	Input Capacitance			10	pF	

I/O PORT OPTION (OPEN DRAIN)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage					External Pullup
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	$I_{OL} =$ 2mA
V_{IH}	Input High Voltage	2.9		V_{DD}	Volts	[2]
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			1	A	$V_{IN} = 6V$, Output device off
C_I	Input Capacitance			10	pF	

I/O PORT OPTION C (DRIVER PULLUP)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.75		V_{DD}	Volts	$I_{OH} =$-1 mA
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	$I_{OL} =$ 2 mA

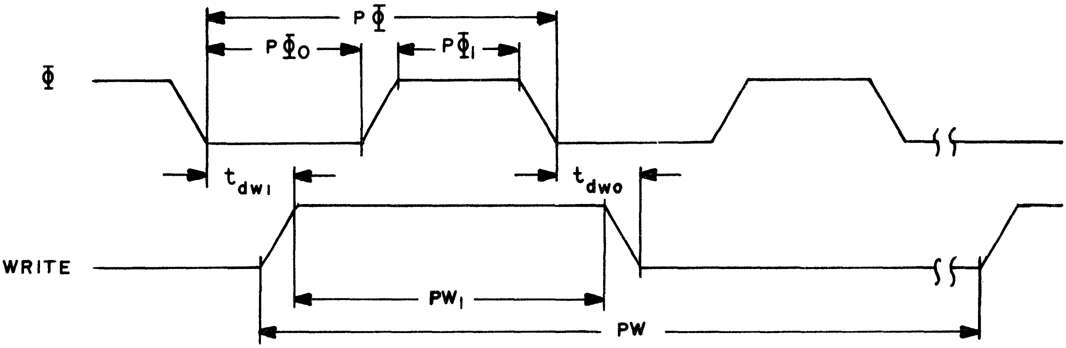
NOTES:

1. Pull up resistor to V_{DD} on CPU.
2. Hysteresis input circuit provides additional .3V noise immunity while internal/external pullup provides TTL compatibility.
3. Measured while I/0 port is outputting a high level.

TIMING

All timing specified at V_{SS} = 0V, V_{DD} = 5V ± 5%, V_{GG} = 12V ± 5%, T_A = 70°C to 0°C.

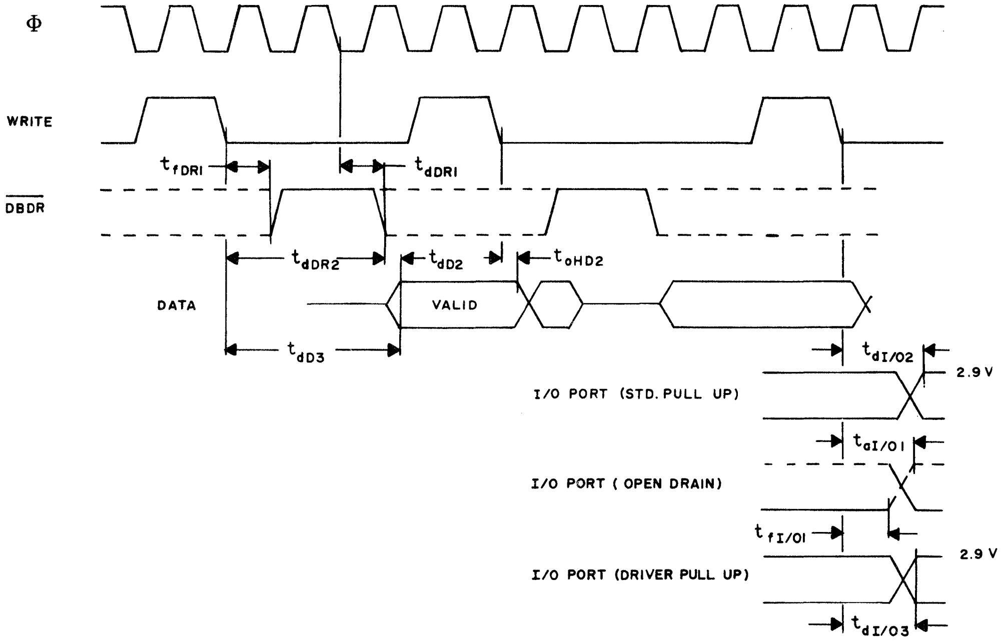
CLOCK TIMING



SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
P Φ	Clock Period	.5		10	μs	
P $Φ_0$	Low time	180			ns	
P $Φ_1$	High time	180			ns	
PW	WRITE Clock Period		4PΦ			Short cycle
PW_0	WRITE Clock Period		6PΦ			Long cycle
PW_1	WRITE Pulse Width	PΦ−100		PΦ		
t_{dw1}	Φ – to WRITE + delay			250	ns	
t_{dw0}	Φ – to WRITE – delay			225	ns	

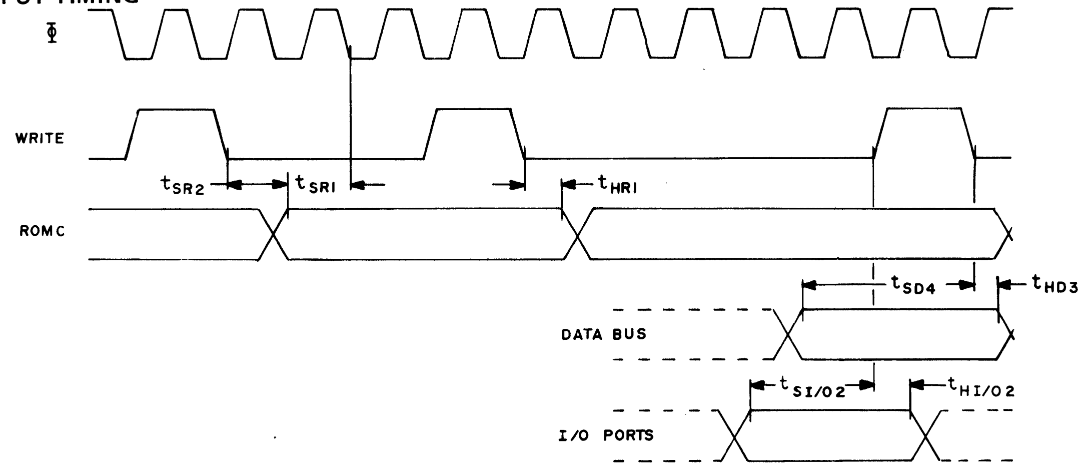
F8 Family

OUTPUT TIMING



SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
tfDR1	WRITE to DBDR floating			400	ns	
tdDR1	Φ to DBDR 1-0		200	625	ns	C_L = 100pF, R_L = 12.5K
tdDR2	WRITE to DBDR 1-0			2P Φ+ 625– tdw0	ns ns	C_L = 100pF, R_L = 12.5K C_L = 100pF
tdD3	WRITE to DATA VALID	2P Φ– tdW0	2PΦ– 400	2P Φ+ 700– tdW0	ns	C_L = 100pF
tOHD2	Guaranteed Data Hold Time After Fall of WRITE	30			ns	
tdI/02	WRITE to I/0 Port Valid			1.5	μs	STD Pull up, C_L = 50pF
tdI/03	WRITE to I/0 Port Valid			400	ns	Driver Pullup, C_L = 50pF
td I/01	WRITE to I/0 Port-Actively Pulled Down			400	ns	Open Drain R_L = 12.5K, C_L = 50pF
tfI/01	WRITE to I/0 Port-Floating			375	ns	Open Drain

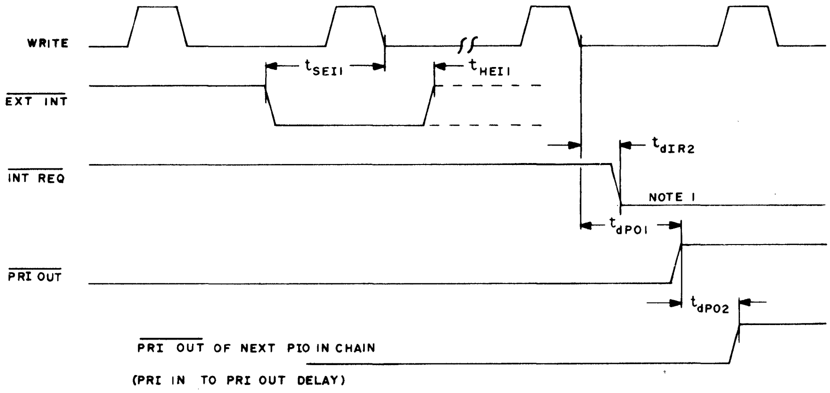
INPUT TIMING



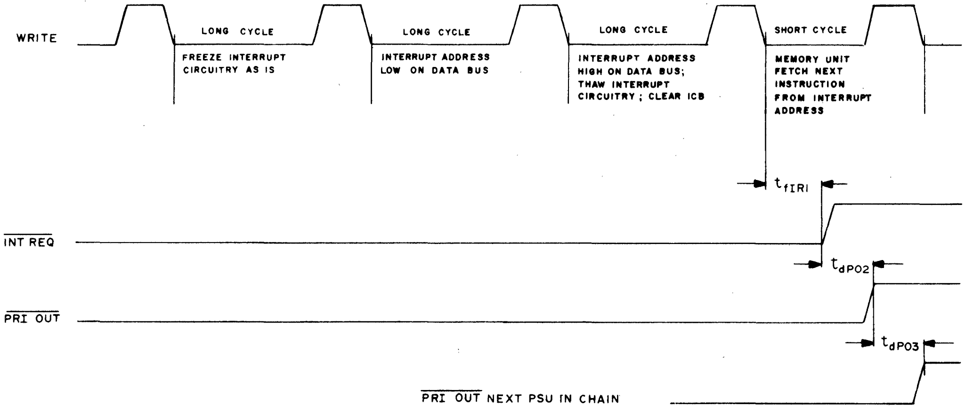
SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{SR1}	ROMC Setup Time	225			ns	
t_{SR2}	ROMC Valid Measured From Fall of WRITE			550	ns	
t_{HR1}	ROMC Required Hold After Fall Of WRITE	20			ns	
t_{SD4}	Data Bus Set-up Time				ns	
t_{HD3}	Data Input Hold Time	20			ns	
$t_{SI/02}$	I/0 Input Set-up Time	1.3			ns	
$t_{HI/02}$	I/0 Input Hold Time	20			ns	

5.2.4 INTERRUPT TIMING

A. Request Made



B. Request Allowed By CPU

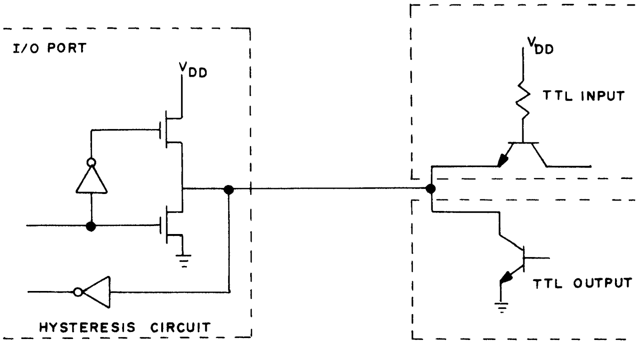

WRITE
LONG CYCLE
FREEZE INTERRUPT CIRCUITRY AS IS
LONG CYCLE
INTERRUPT ADDRESS LOW ON DATA BUS
LONG CYCLE
INTERRUPT ADDRESS HIGH ON DATA BUS; THAW INTERRUPT CIRCUITRY ; CLEAR ICB
SHORT CYCLE
MEMORY UNIT FETCH NEXT INSTRUCTION FROM INTERRUPT ADDRESS
tfIR1
INT REQ
tdPO2
PRI OUT
tdPO3
PRI OUT NEXT PSU IN CHAIN

NOTES.

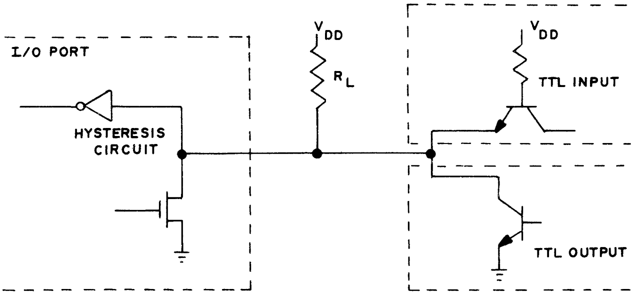
1. Assuming PRI IN is already low. If not, INT REQ 1–0 transition will be delayed 240ns max from the time PRI IN is enabled, and PRI OUT 0–1 transition will be delayed t_{dPO2} from the time PRI IN is enabled.

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{SEI1}	EXT INT Setup Time			1.3	ns	
t_{HEI}	EXT INT Hold Time	30			ns	
t_{dIR2}	WRITE to INT REQ Delay			430	ns	C_L = 100pF
t_{dPO1}	WRITE to PRI OUT Delay			640	ns	C_L = 50pF
t_{dPO2}	PRI IN to PRI OUT 0–1 Delay			300	ns	C_L = 50pF
t_{fIR1}	WRITE to INT REQ Float by PSU			640	ns	Open Drain Output
t_{dPO3}	PRI IN to PRI OUT 1–0 Delay			365	ns	C_L = 50pF

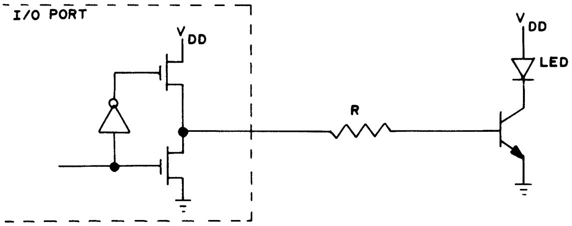
INTERFACING
STANDARD CONFIGURATION



OPEN DRAIN CONFIGURATION



DRIVER PULL–UP CONFIGURATION



APPENDIX A-TIMER COUNTS

CONTENTS OF COUNTER	COUNTS TO INTERRUPT	CONTENTS OF COUNTER	COUNTS TO INTERRUPT	CONTENTS OF COUNTER	COUNTS TO INTERRUPT
FE	254	A4	198	70	142
FD	253	49	197	E1	141
FB	252	92	196	C3	140
F7	251	25	195	86	139
EE	250	4A	194	0C	138
DC	249	94	193	18	137
B8	248	29	192	31	136
71	247	53	191	63	135
E3	246	A6	190	C6	134
C7	245	4D	189	8C	133
8E	244	9A	188	19	132
1D	243	34	187	33	131
3B	242	69	186	67	130
76	241	D3	185	CE	129
ED	240	A7	184	9D	128
DA	239	4F	183	3A	127
B4	238	9E	182	74	126
68	237	3C	181	E9	125
D1	236	78	180	D2	124
A3	235	F0	179	A5	123
47	234	E0	178	4B	122
8F	233	C1	177	96	121
1F	232	82	176	2D	120
3F	231	04	175	5B	119
7E	230	09	174	B7	118
FC	229	12	173	6E	117
F9	228	24	172	DD	116
F3	227	48	171	BA	115
E6	226	90	170	75	114
CD	225	21	169	EB	113
9B	224	42	168	D6	112
36	223	85	167	AD	111
6D	222	0A	166	5A	110
DB	221	14	165	B5	109
B6	220	28	164	6A	108
6C	219	51	163	D5	107
D9	218	A2	162	AB	106
B2	217	45	161	56	105
64	216	8B	160	AC	104
C8	215	17	159	58	103
91	214	2E	158	B1	102
23	213	5D	157	62	101
46	212	BB	156	C4	100
8D	211	77	155	88	99
1B	210	EF	154	11	98
37	209	DE	153	22	97
6F	208	BC	152	44	96
DF	207	79	151	89	95
BE	206	F2	150	13	94
7D	205	E4	149	26	93
FA	204	C9	148	4C	92
F5	203	93	147	98	91
EA	202	27	146	30	90
D4	201	4E	145	61	89
A9	200	9C	144	C2	88
52	199	38	143	84	87

CONTENTS OF COUNTER	COUNTS TO INTERRUPT
08	86
10	85
20	84
40	83
81	82
02	81
05	80
0B	79
16	78
2C	77
59	76
B3	75
66	74
CC	73
99	72
32	71
65	70
CA	69
95	68
2B	67
57	66
AE	65
5C	64
B9	63
73	62
E7	61
CF	60
9F	59
3E	58
7C	57
F8	56
F1	55
E2	54
C5	53
8A	52
15	51
2A	50
55	49
AA	48
54	47
A8	46
50	45
A0	44
41	43
83	42
06	41
0D	40
1A	39
35	38
6B	37
D7	36
AF	35
5E	34
BD	33
7B	32
F6	31
EC	30
D8	29
B0	28
60	27
C0	26
80	25
00	24
01	23
03	22
07	21
0F	20
1E	19
3D	18
7A	17
F4	16
E8	15
D0	14
A1	13
43	12
87	11
0E	10
1C	9
39	8
72	7
E5	6
CB	5
97	4
2F	3
5F	2
BF	1
7F	0

F8
Family

MOSTEK®

F8 MICROPROCESSOR DEVICES

Peripheral Input/Output MK 3871

FEATURES

- □ Two 8-bit I/O ports
- □ Programmable binary timer
- □ External/timer interrupt control circuitry
- □ Low power dissipation — typically less than 200mW

GENERAL DESCRIPTION

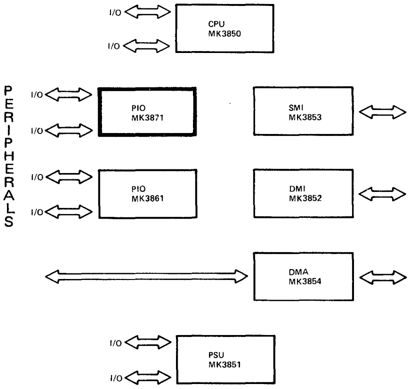
The MK3871 Peripheral Input/Output Circuit (PIO) provides two 8-bit I/O ports and a programmable timer for an F8 multi-chip system (MK3850 family). The MK3871 has the same improved timer and ready strobe output as are on the MK3870 single-chip microcomputer. Thus, for software compatibility with the MK3870, the MK3871 PIO should be used in F8 multi-chip configurations rather than the MK3861 PIO. The MK3871 is manufactured using the same N-channel silicon-gate technology as the single chip MK3870 and the multi-chip F8 family.

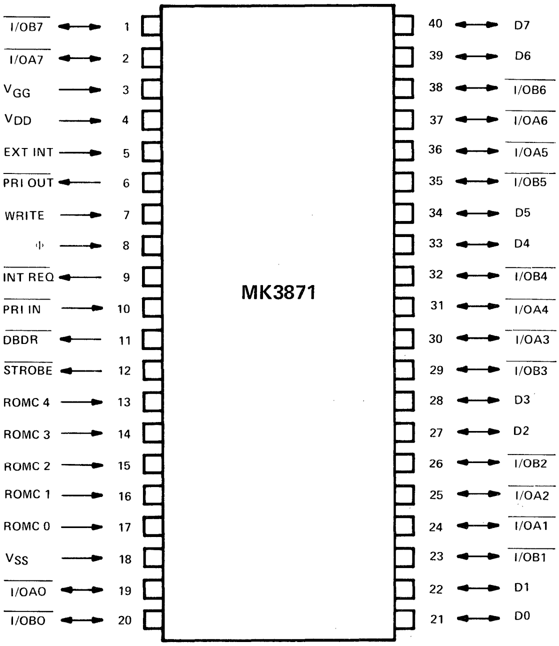
PIN NAME	DESCRIPTION	TYPE
D0-D7	Data Bus Lines	Bi-Directional, Tri-State
$\overline{\text{I/O A0 - I/O A7}}$	I/O Port A	Bi-Directional
$\overline{\text{I/O B0 - I/O B7}}$	I/O Port B	Bi-Directional
ROMC 0 - ROMC 4	System Control Lines	Input
Φ, WRITE	Clock Lines	Input
EXT INT	External Interrupt	Input
$\overline{\text{PRI IN}}$	Priority In	Input
$\overline{\text{PRI OUT}}$	Priority Out	Output
$\overline{\text{INT REQ}}$	Interrupt Request	Output
$\overline{\text{DBDR}}$	Data Bus Drive	Output
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input
$\overline{\text{STROBE}}$	Ready Strobe	Output

SINGLE CHIP MK3870

I/O
I/O
MK3870
I/O
I/O

F8 FAMILY





FUNCTIONAL PIN DEFINITION

D0 – D7 (BI-DIRECTIONAL, TRI-STATE)

DATA BUS: The Data Bus provides bi-directional communication between the F8 CPU and the 3871 and all other peripheral circuits for transfer of data. D0 is the least significant bit.

I/O A0 – I/O A7 and I/O B0 – I/O B7 (Bi-directional)

I/O PORTS: Two 8-bit I/O ports are located on the 3871 PIO. These ports are referred to as Port A and Port B herein, but the actual port number is determined by the version of the 3871 that is selected. These ports have output latches to hold output data.

ROMC 0 – ROMC 4 (INPUT)

SYSTEM CONTROL LINES: These lines provide the 3871 with control information from the F8 CPU. The CPU sets up these lines early in each machine cycle, and the PIO executes that command during that cycle.

Φ (INPUT)

Φ (PHI) CLOCK: This is the high frequency F8 system clock. It is generated by the F8 CPU. Each machine cycle contains either 4 Φ periods (short cycle) or 6 Φ periods (long cycle).

WRITE (INPUT)

WRITE CLOCK: This clock defines the machine cycle. The cycle starts with the fall of the WRITE clock. The system control lines become stable shortly after the start of the cycle and the PIO decodes and executes the command communicated by the control lines. All ROMC commands are started and completed within one cycle of WRITE.

EXT INT (INPUT)

EXTERNAL INTERRUPT: This is the external interrupt input. It may also be used in conjunction with the timer for pulse width measurement and event counting. Its active state is software programmable.

$\overline{\text{PRI IN}}$(INPUT)

PRIORITY IN: This input signals the PIO that a higher priority peripheral has an interrupt request pending on the CPU. If an interrupt is received, it will be latched into the PIO but it will not be serviced until PRI IN is in the "low" state.

$\overline{\text{PRI OUT}}$ (OUTPUT)

PRIORITY OUT: This output signals lower priority peripherals that the PIO either has an interrupt request pending on the CPU, or that a still higher priority peripheral has requested an interrupt.

$\overline{\text{INT REQ}}$ (OUTPUT)

INTERRUPT REQUEST: This open drain output is wired OR ed with the corresponding output on all other peripherals to form the interrupt request input to the CPU.

$\overline{\text{DBDR}}$ (OUTPUT)

DATA BUS DRIVE: This output goes "low" whenever the PIO is driving the Data Bus as an output. It may be used to control tri-state buffers in a buffered Data Bus system and to signal other peripherals that the PIO has "control" of the Data Bus at that time.

V_{SS} (INPUT)

V_{SS}: This is system ground (0V.) V_{DD} and V_{GG} are referenced to V_{SS}.

V_{DD} (INPUT)

V_{DD}: Power line; +5V ± 5%.

V_{GG} (INPUT)

V_{GG}: Power line; +5V ± 5% or +12V ± 5%. With V_{GG} at +5V the Data Bus output levels are TTL compatible; however, for a CMOS or MOS higher output level V_{GG} may be connected to +12V.

$\overline{\text{STROBE}}$ (OUTPUT)

PORT A READY STROBE: This pin which is normally high provides a single low pulse after valid data is present on the $\overline{\text{I/O A0}}$ – $\overline{\text{I/O A7}}$ pins during an output instruction.

PIO FUNCTIONAL DIAGRAM

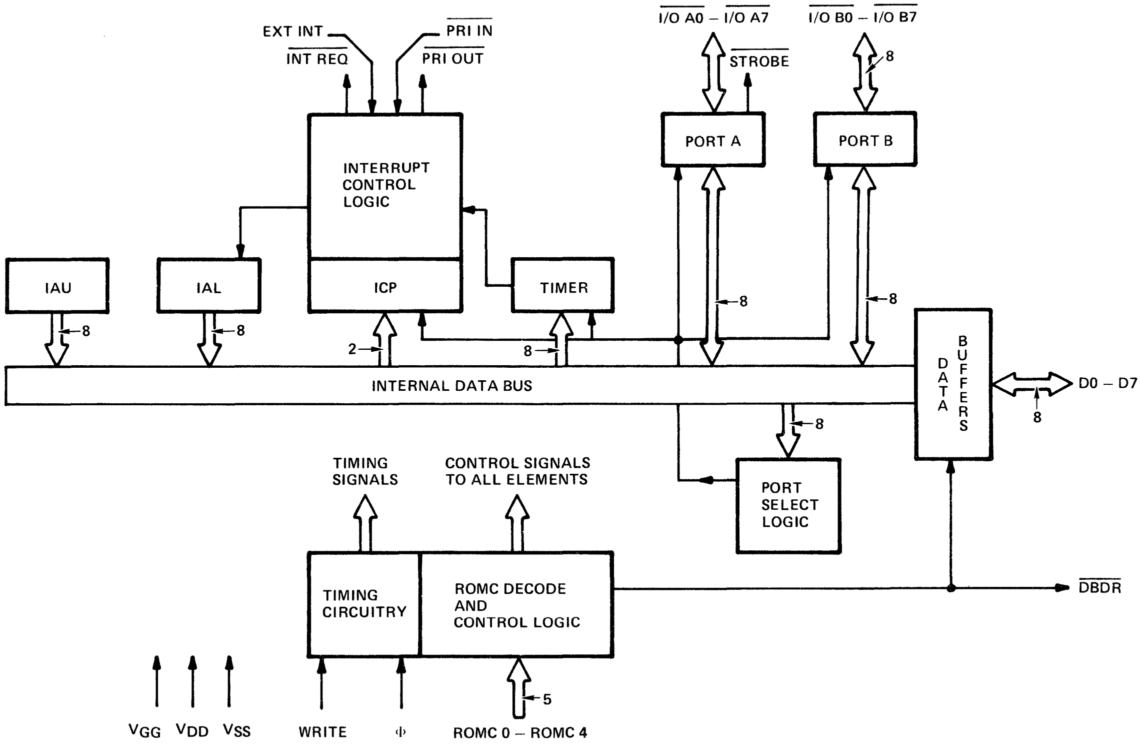


Figure 1.

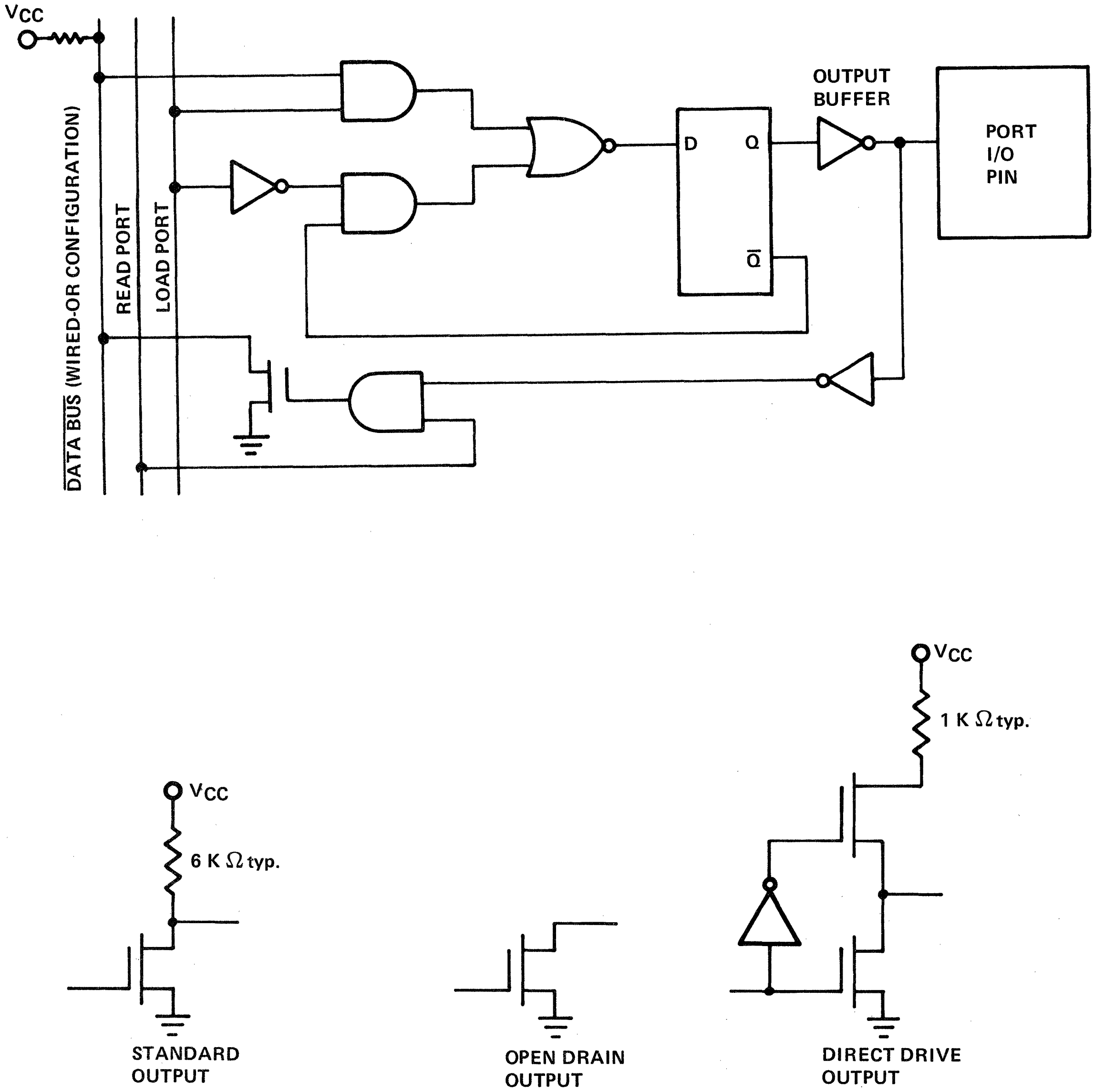
INPUT/OUTPUT PORTS

Each 3871 chip has two bi-directional 8-bit I/O ports. Using binary notation, Port A's address is XXXXXX00 and Port B's address is XXXXXX01, where the X binary digits are the chip's unique I/O port select code. If the port select code, for example, is chosen to be 000001, then Port A may be called Port 4 and Port B may be called Port 5. (The PIO port select code is not permitted to be all 0's since Ports 0 and 1 are reserved for the MK3850 CPU). In addition, the Interrupt Control Port is addressed as port XXXXXX10 and the binary timer is addressed as port XXXXXX11 (which become Ports 6 and 7 for the port select code example given above).

An output instruction (OUT or OUTS) causes the contents of the Accumulator to be latched into the addressed port. An input instruction (IN or INS) transfers the contents of the port to the Accumulator (the Interrupt Control Port is an exception which is described later). The I/O pins on the 3871 are logically inverted. The two I/O ports may both be any of the three output options shown in Figure 2.

An output ready strobe is associated with Port A. This strobe may be used to signal a peripheral device that the 3870 has just completed an output of new data to Port A. The strobe provides a single low pulse shortly after the output operation is completely finished. The $\overline{\text{STROBE}}$ output is always configured similar to a Standard Output (see Figure 2) except that it is capable of driving 3 TTL loads.

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS

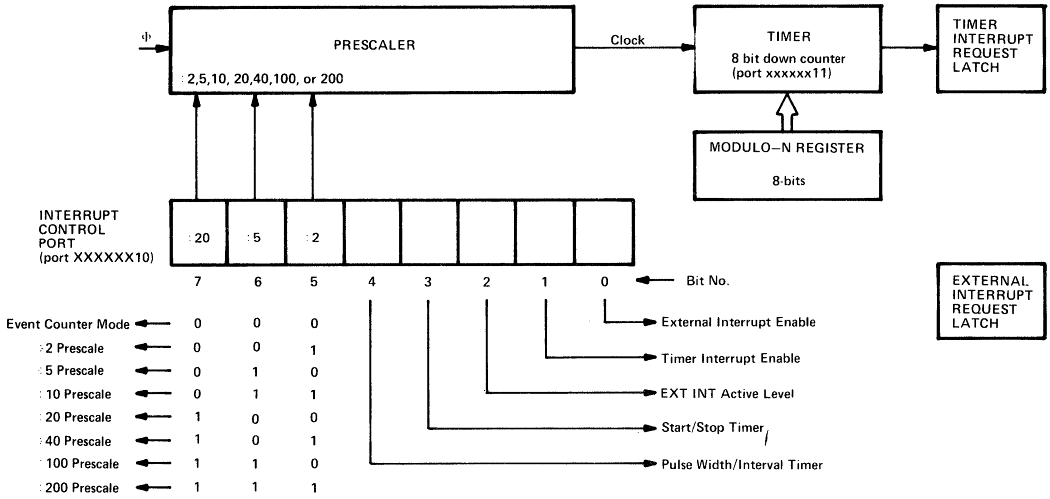


Direct drive ports may be used only as outputs.

The $\overline{\text{STROBE}}$ output is always configured similar to a Standard Output except that it is capable of driving 3 TTL loads.

Figure 2.

TIMER & INTERRUPT CONTROL PORT BLOCK DIAGRAM



	7	6	5
Event Counter Mode	0	0	0
÷2 Prescale	0	0	1
÷5 Prescale	0	1	0
÷10 Prescale	0	1	1
÷20 Prescale	1	0	0
÷40 Prescale	1	0	1
÷100 Prescale	1	1	0
÷200 Prescale	1	1	1

Figure 3.

TIMER

Timer and Interrupt Control Port

The Timer is an 8-bit binary down counter which is software programmable to operate in one of three modes: the Interval Timer Mode, the Pulse Width Measurement Mode, or the Event Counter Mode. As shown in Figure 3, associated with the Timer are an 8-bit register called the Interrupt Control Port, a programmable prescaler, and an 8-bit modulo-N register.

The desired timer mode, prescale value, starting and stopping the timer, active level of the EXT INT pin, and local enabling or disabling of interrupts are selected by outputting the proper bit configuration from the Accumulator to the Interrupt Control Port with an OUT or OUTS instruction. Bits within the Interrupt Control Port are defined as follows:

Interrupt Control Port (Port XXXXXX10)

Bit 0 – External Interrupt Enable
Bit 1 – Timer Interrupt Enable
Bit 2 – EXT INT Active Level
Bit 3 – Start/Stop Timer
Bit 4 – Pulse Width/Interval Timer
Bit 5 – ÷ 2 Prescale
Bit 6 – ÷ 5 Prescale
Bit 7 – ÷ 20 Prescale

A special situation exists when reading the Interrupt Control Port (with an IN or INS instruction). The Accumulator is not loaded with the content of the ICP; instead, Accumulator bits 0 through 6 are loaded with 0's while bit 7 is loaded with the logic level being applied to the EXT INT pin, thus allowing the status of EXT INT to be determined without the necessity of servicing an external interrupt request. This capability is useful in establishing a high speed polled handshake procedure or for using EXT INT as an extra input pin if external interrupts are not required and the Timer is used only in the Interval Timer Mode. However, if it is desirable to read the content of the ICP, then one of the 64 scratchpad registers may be used to save a copy of whatever is written to the ICP.

The rate at which the timer is clocked in the Interval Timer Mode is determined by the frequency of the Φ clock and by the division value selected for the prescaler. If ICP bit 5 is set and bits 6 and 7 are cleared, the prescaler divides Φ by 2. Likewise, if bit 6 or 7 is individually set the prescaler divides Φ by 5 or 20 respectively. Combinations of bits 5, 6 and 7 may also be selected. For example, if bits 5 and 7 are set while 6 is cleared the prescaler will divide by 40. Thus possible prescaler values are: ÷ 2, ÷ 5, ÷10, ÷20, ÷40, ÷100, and ÷200.

Any of three conditions will cause the prescaler to be reset: (1) Whenever the timer is stopped by clearing ICP bit 3; (2) Execution of an output instruction to the timer (port address XXXXXX11); or (3) On the trailing edge transition of the EXT INT pin when in the Pulse Width Measurement Mode. These last two conditions are explained in more detail below.

An OUT or OUTS to Port XXXXXX11 will load the content of the Accumulator to both the Timer and the 8-bit modulo-N register, reset the prescaler, and clear any previously stored timer interrupt request. As previously noted, the Timer is an 8-bit down counter which is clocked by the prescaler in the Interval Timer Mode and in the Pulse Width Measurement Mode. The prescaler is not used in the Event Counter Mode. The modulo-N register is a buffer whose function is to save the value which was most recently outputted to port XXXXXX11. The modulo-N register is used in all three timer modes.

Interval Timer Mode

When ICP bit 4 is cleared (logic 0) and at least one prescale bit is set the Timer operates in the Interval Timer Mode. When bit 3 of the ICP is set the Timer will start counting down from the modulo-N value. After counting down to H'01', the Timer returns to the modulo-N value at the next count. On the transition from H'01' to H'N' the Timer sets a timer interrupt request latch. Note that the interrupt request latch is set by the transition to H'N' and not by the presence of H'N' in the Timer, thus allowing a full 256 counts if the modulo-N register is preset to H'00'. If bit 1 of the ICP is set, the interrupt request is passed on to the CPU via $\overline{\text{INT REQ}}$. However, if bit 1 of the ICP is a logic 0 the interrupt request is not passed on to the CPU, but the interrupt request latch remains set. If ICP bit 1 is subsequently set, the interrupt request will then be passed on to the CPU. Only two events can reset the timer interrupt request latch; when the timer interrupt request is acknowledged by the CPU, or when a new load of the modulo-N register is performed.

Consider an example in which the modulo-N register is loaded with H '64' (decimal 100). The timer interrupt request latch will be set at the 100th count following the timer start and the timer interrupt request latch will repeatedly be set on precise 100 count intervals. If the prescaler is set at ÷ 40, the timer interrupt request latch will be set every 4,000 Φclock periods. For a 2 MHz Φ clock this will produce 2 millisecond intervals.

The range of possible intervals is from 2 to 51,200 Φ clock periods (1 μs to 25.6 ms for a 2 MHz Φ clock). However, approximately 50 Φ periods is a practical minimum because the time between setting the interrupt request latch and the execution of the first instruction of the interrupt service routine is at least 27 Φ periods (the response time is dependent upon how many privileged instructions are encountered when the request occurs). To establish time intervals greater than 51,200 Φ clock periods is a simple matter of using the timer interrupt service routine to count the number of interrupts, saving the result in one or more of the scratchpad registers until the desired interval is achieved. With this technique virtually any time interval or several time intervals may be generated.

The Timer may be read at any time and in any mode using an input instruction (IN or INS) and may take place "on the fly" without interfering with normal timer operation. Also, the Timer may be stopped at any time by clearing bit 3 of the ICP. The Timer will hold its current contents indefinitely and will resume counting when bit 3 is again set. Recall, however, that the prescaler is reset whenever the Timer is stopped; thus a series of starting and stopping will result in a cumulative truncation error.

A summary of other timer errors is given in the timing section of this specification. For a free running timer in the Interval Timer Mode, the time interval between any two interrupt requests may be in error by ±6 Φclock periods although the cumulative error over many intervals is zero. The prescaler and Timer generate precise intervals for setting the timer interrupt request latch but the time out may occur at any time within a machine cycle. (There are two types of machine cycles; short cycles which consist of 4 Φ clock periods and long cycles which consist of 6 Φclock periods. The WRITE clock corresponds to a machine cycle). Interrupt requests are synchronized with the WRITE clock, thus giving rise to the possible ±6 Φ error. Additional errors may arise due to the interrupt request occuring while a privileged instruction or multicycle instruction is being executed. Nevertheless, for most applications all of the above errors are negligible, especially if the desired time interval is greater than 1 ms.

Pulse Width Measurement Mode

When ICP bit 4 is set (logic 1) and at least one prescale bit is set, the Timer operates in the Pulse Width Measurement Mode. This mode is used for accurately measuring the duration of a pulse applied to the EXT INT pin. The Timer is stopped and the prescaler is reset whenever EXT INT is at its inactive level. The active level of EXT INT is defined by ICP bit 2; if cleared, EXT INT is active low; if set, EXT INT is active high. If ICP bit 3 is set, the prescaler and Timer will start counting when EXT INT transitions to the active level. When EXT INT returns to the inactive level the Timer then stops, the prescaler resets, and if ICP bit 0 is set an external interrupt request latch is set. (Unlike timer interrupts, external interrupts are not latched if the ICP Interrupt Enable bit is not set).

As in the Interval Timer Mode, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, the prescaler and ICP bit 1 function as previously described, and the Timer still functions as an 8-bit binary down counter with the timer interrupt request latch being set on the Timer's transition from H '01' to H 'N'. Note that the EXT INT pin has nothing to do with loading the Timer; its action is that of automatically starting and stopping the Timer and of generating external interrupts. Pulse widths longer than the prescale value times the molulo-N value are easily measured by using the timer interrupt service routine to store the number of timer interrupts in one or more scratchpad registers.

As for accuracy, the actual pulse duration is typically slightly longer than the measured value because the status of the prescaler is not readable and is reset when the Timer is stopped. Thus for maximum accuracy it is advisable to use a small division setting for the prescaler.

Event Counter Mode

When ICP bit 4 is cleared and all prescale bits (ICP bits 5, 6, and 7) are cleared, the Timer operates in the Event Counter Mode. This mode is used for counting pulses applied to the EXT INT pin. If ICP bit 3 is set, the Timer will decrement on each transition from the inactive level to the active level of the EXT INT pin. The prescaler is not used in this mode; but as in the other two timer modes, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, ICP bit 1 functions as previously described, and the timer interrupt request latch is set on the Timer's transition from H '01' to H 'N'.

Normally ICP bit 0 should be kept cleared in the Event Counter Mode; otherwise, external interrupts will be generated on the transition from the inactive level to the active level of the EXT INT pin.

For the Event Counter Mode, the minimum pulse width required on EXT INT is 2 Φclock periods and the minimum inactive time is 2 Φ periods; therefore, the maximum repetition rate is 500 KHz.

EXTERNAL INTERRUPTS

When the timer is in the Interval Timer Mode the EXT INT pin is available for non-timer related interrupts. If ICP bit 0 is set, an external interrupt request latch is set when there is a transition from the inactive level to the active level of EXT INT. (EXT INT is an edge-triggered input). The interrupt request is latched until either acknowledged by the CPU section or until ICP bit 0 is cleared (unlike timer interrupt requests which remain latched even when ICP bit 1 is cleared). External interrupts are handled in the same fashion when the Timer is in the Pulse Width Measurement Mode or in the Event Counter Mode, except that only in the Pulse Width Measurement Mode the external interrupt request latch is set on the trailing edge of EXT INT; that is, on the transition from the active level to the inactive level.

INTERRUPT HANDLING

Figure 4 is a block diagram of the interrupt interconnection for a typical F8 system.

Each PIO has a $\overline{\text{PRIORITY IN}}$ and a $\overline{\text{PRIORITY OUT}}$ line so that they may be daisy chained together in any order, to form a priority level of interrupts. When a PIO receives an interrupt (either timer or external) it pulls its $\overline{\text{PRI OUT}}$ output high, signaling all lower priority peripherals that it has a higher priority interrupt request pending on the CPU. Also, when the PIO's $\overline{\text{PRI IN}}$ input is pulled high by a higher priority peripheral, signaling the PIO that there is a still higher priority interrupt request, it passes that signal along by pulling its $\overline{\text{PRI OUT}}$ high. When the CPU processes an interrupt request it commands the interrupting peripheral to place its interrupt vector address on the Data Bus. Only that peripheral whose $\overline{\text{PRI IN}}$ is low and which has an interrupt request pending will respond. Should there be another lower priority peripheral with a pending request, it will not respond at that time because its $\overline{\text{PRI IN}}$ input will be high.

If there is both a timer interrupt request and an external interrupt request when the CPU section starts to process the requests, the timer interrupt is handled first.

Within each local interrupt control circuit there is a 16-bit interrupt address vector. This vector is the address to which the program counter will be set after an interrupt is acknowledged; hence, it is the address of the first executable instruction of the interrupt routine. The 3871 has an interrupt address which is particular to the version of the 3871 selected by the user. Fifteen bits are fixed. These are bits 0 through 6 and 8 through 15. Bit seven (2^7) is dependent upon the type of interrupt. This bit will be a 0 for internal timer generated interrupts and a 1 for external interrupts. When the interrupt logic sends an interrupt request signal and the CPU is enabled to service it, the normal state sequence of the CPU is interrupted at the end of an instruction. The CPU signals the interrupt circuits via the five control lines. The requesting local interrupt circuit sends a 16-bit interrupt address vector (from the interrupt address generator) onto the Data Bus in two consecutive bytes. The address is made available to the program counter via the address demultiplexer circuits. Simultaneously, the address is also made available to all other devices connected to the data bus. It is the address of the next instruction to be executed. The program counter (PO) of each memory device is set with this new address while the stack register (P) is loaded with the previous contents of the program counter. The information in P is lost. Thus, the next instruction to be executed is determined by the value of the interrupt address vector.

The Interrupt Control Bit (ICB) of the CPU (loaded in the W register) allows interrupts to be recognized. Clearing the ICB prevents acknowledgement of interrupts. The ICB is cleared during power on, external reset, and after an interrupt is acknowledged. The interrupt status of the PSU, PIO or MI devices is not affected by the execution of the DISABLE INTERRUPT (DI) instruction. At the conclusion of most instructions, the fetch logic checks the state of the Interrupt Request Line. If there is an interrupt, the next instruction fetch cycle is suspended and the system is forced into an interrupt sequence.

The CPU allows interrupts after all F8 instructions except the following:

(PK)	PUSH K
(PI)	PUSH IMMEDIATE
(POP)	POP
(JMP)	JUMP
(OUTS)	OUTPUT SHORT (Excluding OUTS 00 and 01)
(OUT)	OUTPUT
(EI)	SET ICB
(LR W, J)	LOAD THE STATUS REGISTER FROM SCRATCHPAD

POWER ON

As a result, it is possible to perform one more instruction after the above CPU instructions without being interrupted.

INTERRUPT INTERCONNECTION

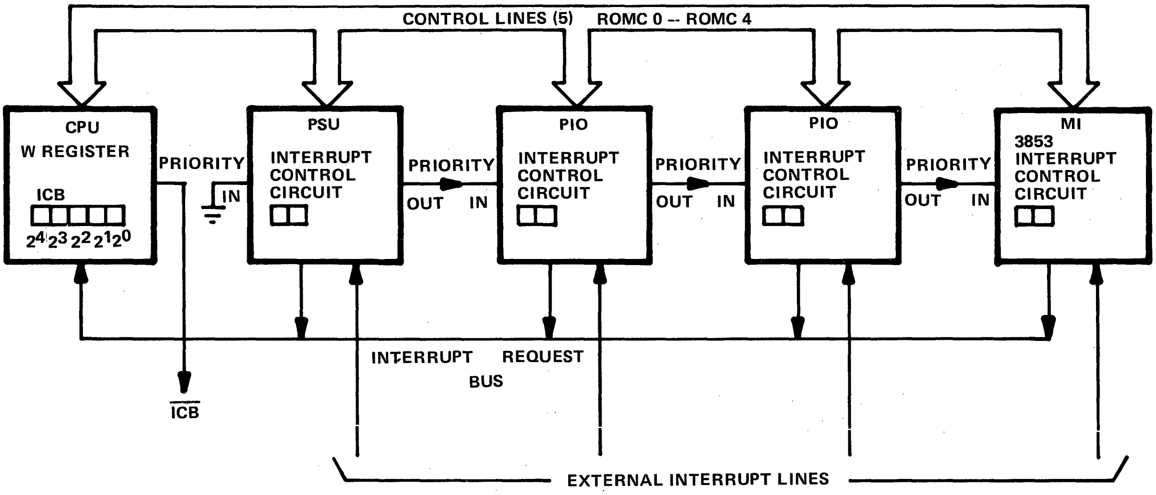


Figure 4.

INTERRUPT SEQUENCE

Figure 5 details the interrupt sequence which occurs whether the interrupt request is from an external source via EXT INT or from the PIO's internal timer. Events are labeled with the letters A through G and are described below.

Event A

An interrupt request must satisfy a setup time requirement as specified on page 19. If not satisfied, INT REQ will delay going low until the next negative edge of the WRITE clock.

Event B

Event B represents the instruction being executed when the interrupt occurs. The last cycle of B is normally the instruction fetch for the next cycle. However, if B is not a privileged instruction and the CPU's Interrupt Control Bit is set, then the last cycle becomes a "freeze" cycle rather than a fetch. At the end of the freeze cycle the interrupt request latches are inhibited from altering the interrupt daisy-chain so that sufficient time will be allowed for the daisy-chain to settle. (If B is a privileged instruction, the instruction fetch is not replaced by a freeze cycle; instead, the fetch is performed and the next instruction is executed. Although unlikely to be encountered, a series of privileged instructions will be sequentially executed without Interrupt. One more instruction, called a 'protected' instruction, will always be executed after the last privileged instruction. The last cycle of the protected instruction then performs the freeze.)

The dashed lines on EXT INT illustrate the last opportunity for EXT INT to cause the last cycle of a non-protected instruction to become a freeze cycle. The freeze cycle is a short cycle (4 Φ clock periods) in all cases except where B is the Decrement Scratchpad instruction, in which case the freeze cycle is a long cycle (6 Φclock periods).

$\overline{\text{INT REQ}}$ goes low on the next negative edge of WRITE if both $\overline{\text{PRI IN}}$ is low and the appropriate interrupt enable bit of the Interrupt Control Port is set.

Event C

A NO-OP long cycle to allow time for the PRI IN/PRI OUT chain to settle. At a 2 MHz Φ clock rate a total of 7 PIO, PSU, or MI devices may be daisy-chained without the need for look-ahead logic.

INTERRUPT SEQUENCE

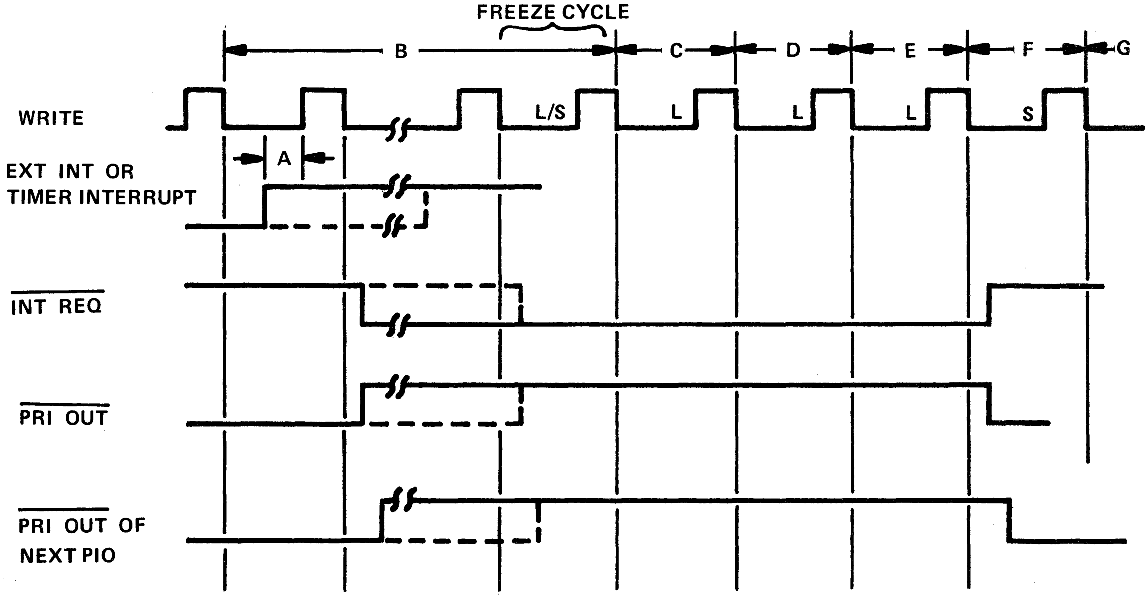


Figure 5.

Event D

In PSU circuits the program counter (PO) is pushed to the stack register (P) in order to save the return address. The interrupting PIO places the lower 8 bits of the interrupt vector address onto the data bus. This is always a long cycle.

Event E

A long cycle in which the PIO places the upper 8 bits of the interrupt vector address onto the data bus.

Event F

A short cycle in which the PIO's interrupting interrupt request latch is cleared. Also, the CPU's Interrupt Control Bit is cleared, thus disabling interrupts until an EI instruction is performed. Additionally, during EVENT F the $\overline{\text{PRI IN/PRI OUT}}$ daisy-chain freeze is removed since the interrupt vector address has been passed to the CPU. Another action is the fetch of the instruction from the interrupt address.

Event G

Begin execution of the first instruction of the interrupt service routine.

Summary Of Interrupt Sequence

For the MK3871 the interrupt response time is defined as the time elapsed between the occurrence of EXT INT going active (or the Timer transitioning to H'N') and the beginning of execution of the first instruction of the interrupt service routine. The interrupt response time is a variable dependent upon what the microprocessor is doing when the interrupt request occurs. As shown in Figure 5, the minimum interrupt response time is 3 long cycles plus 2 short cycles plus one WRITE clock pulse width plus a setup time of EXT INT prior to the leading edge of the WRITE pulse – a total of 27 Φ clock periods plus the setup time. At 2 MHz this is 14.25 μs. Although the maximum could theoretically be infinite, a practical maximum is 35 μs (based on the interrupt request occurring near the beginning of a PI and LR K, P sequence).

DATA FLOW

Table 1 shows the function performed by the PIO for each ROMC command. Each function is entirely performed within one machine cycle (one cycle of the WRITE clock).

TABLE 1

The following ROMC states are decoded by the 3871 as indicated. All other ROMC states are decoded as "NO-OPERATION" (NO-OP).

Binary					Hex	3871 Function
ROMC 4	ROMC 3	ROMC 2	ROMC 1	ROMC 0		
0	1	1	1	1	OF	If this circuit is interrupting and no higher priority circuit is interrupting, move the lower half of the interrupt vector on to the Data Bus and signal Bus use with $\overline{\text{DBDR}}$.
1	0	0	0	0	10	Place interrupt circuitry in an inhibit state that prevents altering the interrupt chain.
1	0	0	1	1	13	If this circuit is interrupting and no higher priority circuit is interrupting move the upper half of the interrupt vector on to the Data Bus and signal Bus use with $\overline{\text{DBDR}}$. In any case, remove priority interrupt circuitry from inhibit state.
1	1	0	1	1	1B	If contents of Data Bus in the previous cycle was an I/O port address, move the contents of that port on to the Data Bus and signal Bus use with $\overline{\text{DBDR}}$ (Input Command).
1	1	0	1	0	1A	If contents of Data Bus in the previous cycle were an address of an I/O port, the timer, or the Interrupt Control Port, move current contents of the Data Bus into that port. (Output Command).
0	1	0	0	0	08	Reset command. Load Port A, Port B, the Interrupt Control Port, and the timer with contents of the Data Bus; CPU is outputting H'00'.

3871 PIO VERSIONS

Each version of the 3871 is denoted by a 90XXX number.

The presently available versions of the 3871 are listed in Table 2.

AVAILABLE VERSIONS OF THE 3871
TABLE 2

VERSION	PORT SELECT CODE	PORT NUMBERS (DERIVED FROM THE PORT SELECT CODE; HEX)	PORT OUTPUT TYPE	INTERRUPT ADDRESS	
				TIMER	EXTERNAL
90070	000001	04 thru 07	Direct Drive	0020	00A0
90071	000001	04 thru 07	Standard	0020	00A0
90072	000001	04 thru 07	Open Drain	0020	00A0
90077	000010	08 thru 0B	Standard	4420	44A0

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS

V_{GG}	–.3V to +15V
V_{DD}	–.3V to +7V
Open Drain Option Ports	–.3V to +13.2V
All Other Inputs and Outputs	–.3V to +7V
Storage Temperature	–55° C to +150° C
Operating Temperature	0° C to 70° C

*All voltages are with respect to V_{SS}. Stresses above those listed may cause permanent damage to the device. Exposure to maximum rated stress for extended periods may impair the useful life of the device.

DC CHARACTERISTICS

V_{SS} = 0V, V_{DD} = 5V± 5%, V_{GG} = 12V± 5%
T_A = 0 to 70°C, unless otherwise noted.

Positive current is defined as conventional current flowing into the pin refrenced.

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		25	60	mA	f = 2 MHz, Outputs unloaded
I_{GG}	V_{GG} Current		3	8	mA	f = 2 MHz, Outputs unloaded

DATA BUS (DB0-DB7)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	2.0		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	$I_{OH} = -100\ \mu A$
V_{OH}	Output High Voltage	2.4			Volts	$I_{OH}=-100\mu A$, $V_{GG}=5V\pm5\%$
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.6 mA [1]
I_{IH}	Input High Current	0		1	μA	V_{IN} = 6V, 3-State mode
I_{OL}	Input Low Current	0		−1	μA	$V_{IN} = V_{SS}$, 3-State mode
C_I	Input Capacitance			10	pF	3-State mode

CLOCK LINES (Φ, WRITE)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	2.0		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			±1	μA	$V_{IN}= V_{SS}$ to +6V
C_I	Input Capacitance			10	pF	

PRIORITY IN AND CONTROL ($\overline{\text{PRI IN}}$, ROMC0 – ROMC4)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	2.0		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			1	μA	$V_{IN} = V_{SS}$ to 6V
C_I	Input Capacitance			10	pF	

PRIORITY OUT ($\overline{\text{PRI OUT}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	I_{OH} - $-100\ \mu A$
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA

INTERRUPT REQUEST ($\overline{\text{INT REQ}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage				Volts	Open Drain Output [1]
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA
I_L	Leakage Current			1	μA	V_{IN} = 6V, Output device off
C_I	Input Capacitance			10	pF	Output device off

DATA BUS DRIVE ($\overline{\text{DBDR}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage					Open Drain Output
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA
I_L	Leakage Current			1	μA	V_{IN} = 6V, Output device off
C_I	Input Capacitance			10	pF	Output device off

EXTERNAL INTERRUPT (EXT INT)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	2.0			Volts	Internal pullup exists
V_{IL}	Input Low Voltage			0.8	Volts	
I_{IL}	Input Low Current			−1.6	mA	V_{IN} = 0.4V
I_{IH}	Input High Current	−100			μA	V_{IN} = 2.4V
C_I	Input Capacitance			10	pF	

READY STROBE ($\overline{\text{STROBE}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	2.4		V_{DD}	Volts	I_{OH} = −300 μA
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 5.0 mA

I/O PORT (STANDARD OUTPUT OPTION)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	I_{OH} =-30μA
V_{OH}	Output High Voltage	2.4		V_{DD}	Volts	I_{OH} = −100μA
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA
V_{IH}	Input High Voltage	2.0		V_{DD}	Volts	Internal Pullup to V_{DD}
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_{IL}	Input Low Current			−1.6	mA	V_{IN} = .4V[2]
C_I	Input Capacitance			10	pF	

I/O PORT (OPEN DRAIN OUTPUT OPTION)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage			13.2	Volts	External Pullup
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA
V_{IH}	Input High Voltage	2.0		13.2	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			5	μA	V_{IN} = 13.2V, Output device off
C_I	Input Capacitance			10	pF	

I/O PORT (DIRECT DRIVE OUTPUT OPTION)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	1.5		V_{DD}	Volts	I_{OH} = −1.5 mA
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA
I_{OH}	Output High Current	−1.5	−4.0	−9.0	mA	V_{OH} = 0.7V to 1.5V

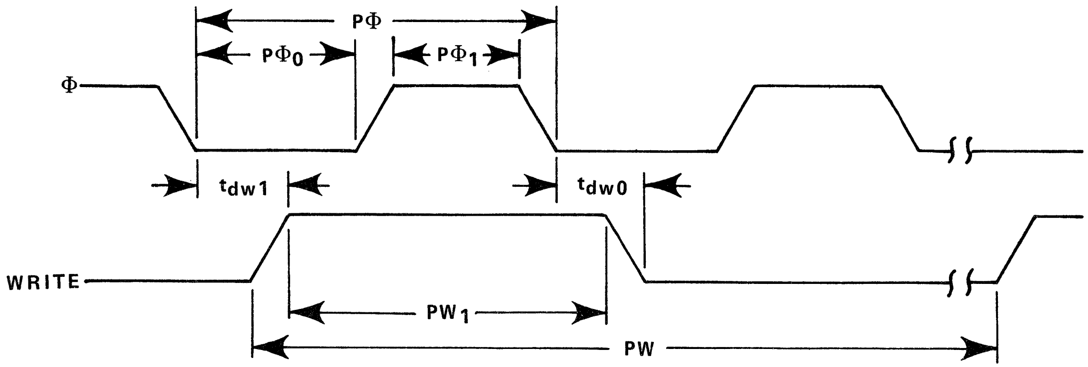
NOTES:

1. Pull up resistor to V_{DD} on CPU.
2. Measured while I/O port is outputting a high level.

TIMING

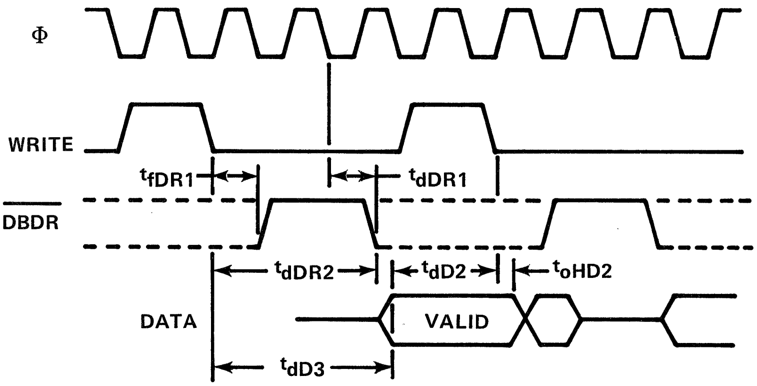
All timing specified at V_{SS} = 0V, V_{DD} = 5V ± 5%, V_{GG} = 12V ± 5%
T_A = 70°C to 0°C

CLOCK TIMING



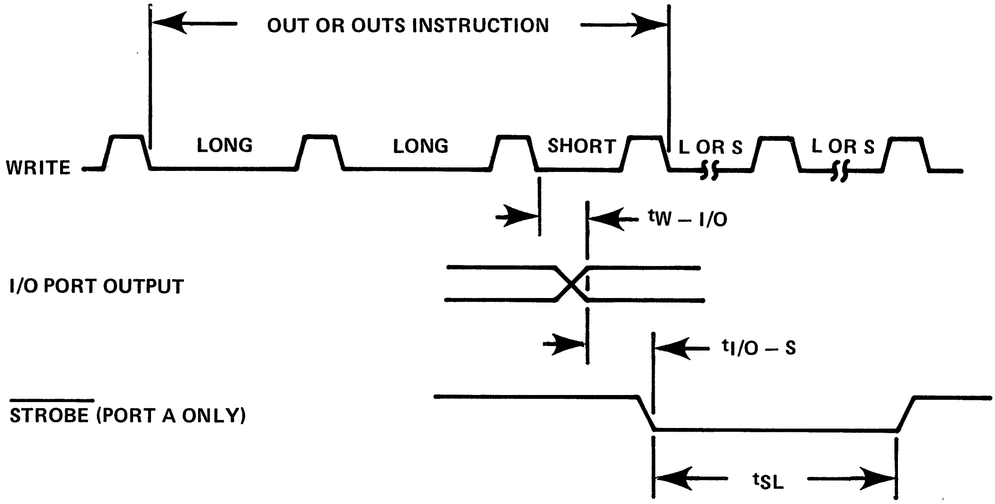
SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
PΦ	Clock Period	.5		10	μs	
PΦ	Low time	180			ns	
$P\Phi_1$	High time	180			ns	
PW	WRITE Clock Period		4PΦ			Short cycle
PW_0	WRITE Clock Period		6PΦ			Long cycle
PW_1	WRITE Pulse Width	PΦ−100		PΦ		
t_{dw1}	Φ − to WRITE + delay			250	ns	
t_{dw0}	Φ − to WRITE − delay			225	ns	

OUTPUT TIMING



SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{fDR1}	WRITE to $\overline{DBDR}$ floating			400	ns	
t_{dDR1}	Φ to $\overline{DBDR}$ 1-0		200	625	ns	C_L = 100pF R_L = 12.5K
t_{dDR2}	WRITE to $\overline{DBDR}$ 1-0			2P Φ+ 625− tdw0	ns ns	C_L = 100pF R_L = 12.5K C_L = 100pF
t_{dD3}	WRITE to DATA VALID	2P Φ− tdW0	2P Φ− 400	2P Φ+ 700− tdW0	ns	C_L = 100pf
t_{oHD2}	Guaranteed Data Hold Time After Fall of WRITE	30			ns	

OUTPUT TIMING (CONT'D)

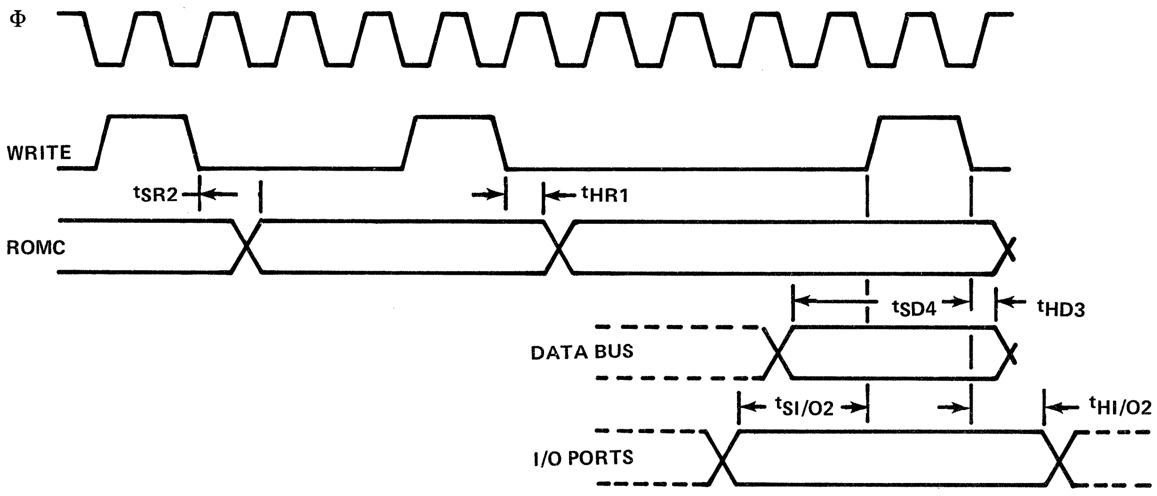


SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
$\overline{\text{STROBE}}$	$t_{I/O\text{-}S}$	Port Output to $\overline{\text{STROBE}}$ Delay	3t Φ-1000	3t Φ + 250	ns	Note 1
	t_{SL}	$\overline{\text{STROBE}}$ Pulse Width, Low	8t Φ-250	12tΦ+250	ns	
I/O PORT	$t_{W\text{-}I/O}$	WRITE to I/O Port Output Valid		1000	ns	Note 2

NOTES:

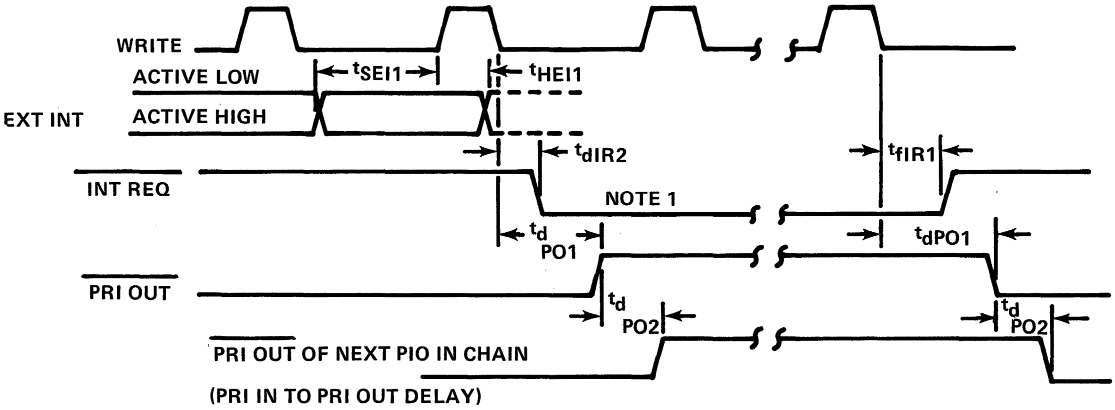
1. Load is 50 pF plus 3 standard TTL inputs.
2. Load is 50 pF plus 1 standard TTL input.

INPUT TIMING



SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{SR2}	ROMC Valid Measured from Fall of WRITE			550	ns	
t_{HR1}	ROMC Required Hold After Fall of WRITE	20			ns	
t_{SD4}	Data Bus Set-Up Time				ns	
t_{HD3}	Data Input	20			ns	
$t_{SI/02}$	I/O Input Set-Up Time	1.3			ns	
$t_{HI/02}$	I/O Input Hold Time	20			ns	

INTERRUPT TIMING



NOTES:

1. Assuming $\overline{\text{PRI IN}}$ is already low. If not, $\overline{\text{INT REQ}}$ 1–0 transition will be delayed 240 ns max from the time $\overline{\text{PRI IN}}$ is enabled, and $\overline{\text{PRI OUT}}$ 0–1 transition will be delayed t_{dPO2} from the time $\overline{\text{PRI IN}}$ is enabled.

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{SEI1}	EXT INT Setup Time	750			ns	
t_{HEI}	EXT INT Hold Time	30			ns	
t_{dIR2}	WRITE to INT REQ Delay			430	ns	C_L = 100pF
t_{dPO1}	WRITE to PRI OUT Delay			640	ns	C_L = 50pF
t_{dPO2}	PRI IN to PRI OUT Delay			350	ns	C_L = 50pF
t_{fIR1}	WRITE to INT REQ Float by PIO			640	ns	Open Drain Output

TIMER CHARACTERISTICS

Definitions:

Error = Indicated time value — actual time value

tpsc = $t\Phi$ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3)	$\pm 6t\Phi$
Cumulative interval error, free running (Note 3)	0
Error between two Timer reads (Note 2)	$\pm(tpsc + t\Phi)$
Start Timer to stop Timer error (Notes 1, 4)	$+t\Phi$ to $-(tpsc + t\Phi)$
Start Timer to read Timer error (Notes 1, 2)	$-5t\Phi$ to $-(tpsc + 7t\Phi)$
Start Timer to interrupt request error (Notes 1, 3)	$-2t\Phi$ to $-8t\Phi$
Load Timer to stop Timer error (Note 1)	$+t\Phi$ to $-(tpsc + 2t\Phi)$
Load Timer to read Timer error (Notes 1, 2)	$-5t\Phi$ to $-(tpsc + 8t\Phi)$
Load Timer to interrupt request error (Notes 1, 3)	$-2t\Phi$ to $-9t\Phi$

Pulse Width Measurement Mode:

Measurement accuracy (Note 4)	$+t\Phi$ to $-(tpsc + 2t\Phi)$
Minimum pulse width of EXT INT pin	$2t\Phi$

Event Counter Mode:

Minimum active time of EXT INT pin	$2t\Phi$
Minimum inactive time of EXT INT pin	$2t\Phi$

NOTES:

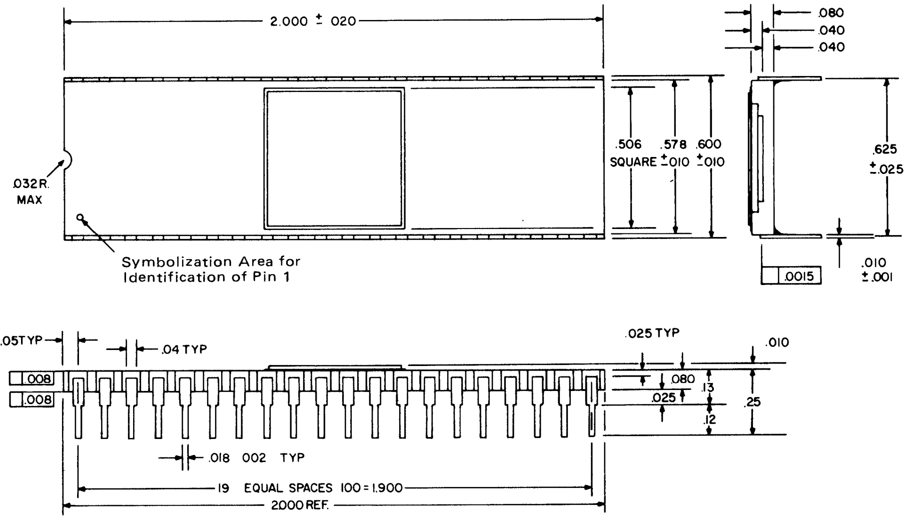
1. All times which entail loading, starting, or stopping the Timer, are referenced from the end of the last machine cycle of the OUT or OUTS instruction.

2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.

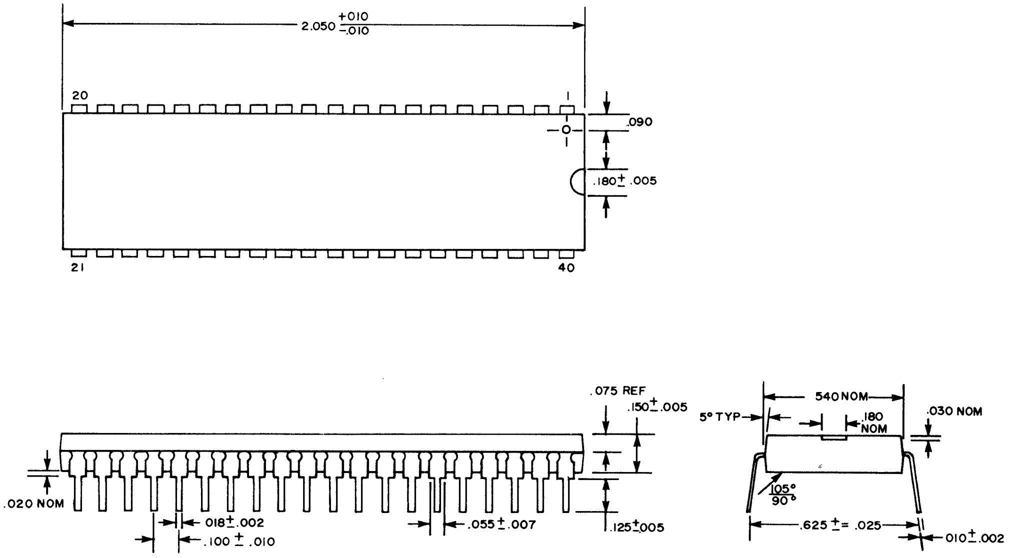
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.

4. Error may be cumulative if operation is repetitively performed. (That is, if the counter if used to total the width of several pulses the error associated with each pulse width measurement will accumulate in the total.)

PACKAGE DESCRIPTION – 40-Pin Dual-In-Line Ceramic Package



PACKAGE DESCRIPTION – 40-Pin Dual-In-Line Plastic Package



ORDERING INFORMATION

Part No.	Package Type
MK3871N/90XXX*	Plastic
MK3871P/90XXX*	Ceramic

*Refer to Table 2 on Page 11 for available 90XXX versions.

MOSTEK®

F8 MICROCOMPUTER SUPPORT

Application Note

USING MOSTEK'S F8 IN A SCANNED KEYBOARD APPLICATION

Using Mostek's F8 In A Scanned Keyboard Application

BLOCK DIAGRAM OF
4x4 KEYBOARD MATRIX

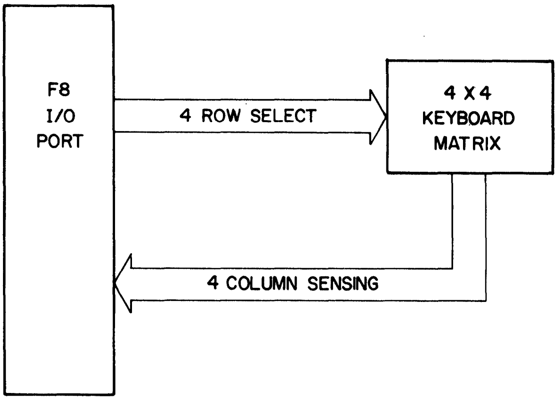


Figure 1

INTRODUCTION

Many microprocessor based systems require input from a keyboard of some type. The hardware required to encode a keyboard outside of the processor can be eliminated by using a keyboard scanning technique. With one F8 port, a 16 switch keyboard can be scanned (see fig. 1) using no external hardware. This is because of the bi-directional quality of the F8 ports.

THEORY OF OPERATION

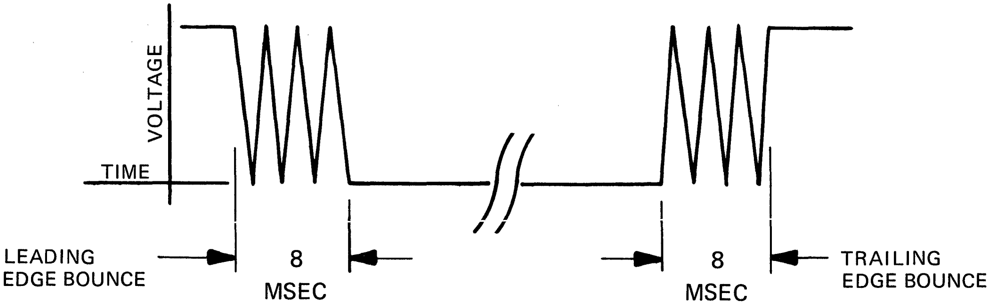
When scanning the keyboard, one of the four row select bits is turned on supplying a ground return for one row of switches. The column data is then read back into the processor via the four column bits. These four bits will indicate the condition of all four switches in the selected row. Each of the four rows is selected, one at a time, continously providing current status of all 16 switches.

"BOUNCE" is a problem encountered when using mechanical switches (see fig. 2). In order to prevent multiple detection of the switch closure, the bounce must be filtered out. A conventional solution to the bounce problem was to use an R—C filter and attempt to eliminate it electrically. However, when using the F8 scanning technique the switch bounce can be filtered in software by taking multiple samples of the switch to verify switch depression and release.

Since the software must usually scan all switches continuously, a register (or half) can be used to maintain the status of each switch.

A common requirement for keyboards is "N—key rollover", meaning that if more than one switch is depressed at a time, all switch closures will be detected. This requirement can be met when using the scanning technique as described above. Since all switches are continuously scanned, the condition of each switch is always available to the processor.

SWITCH BOUNCE



4 x 3 KEY MATRIX

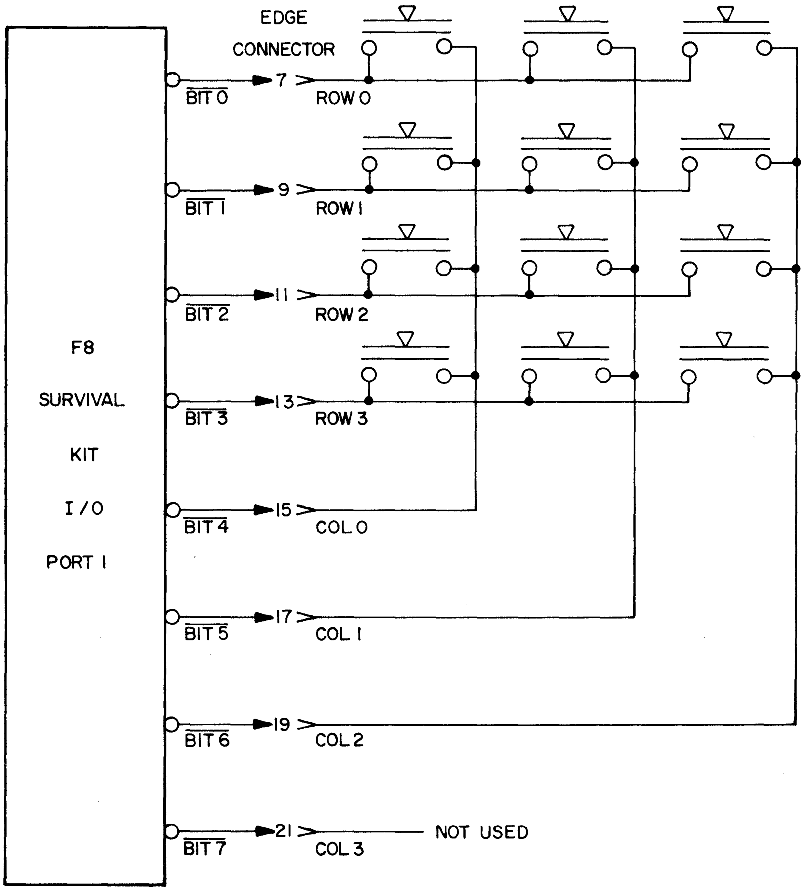


Figure 3

EXAMPLE HARDWARE DESIGN

The example in figure 3 shows a 4x3 matrix interfaced to an F8 port. This arrangement will provide N–key rollover input to the processor unless three keys are depressed simultaneously to form an L configuration. Then erroneous input could occur. If this presents a problem for a given application, one germanium diode (1N 270) should be added on the column pole of each switch (see fig. 4).

The operation of this keyboard (fig. 3) is simple. To sense the condition of row 0, a Hex '01' is written to port 1. Port 1 is then read back. The state of bits 4, 5 and 6 (COL 0, COL 1, COL 2) will be 1 if the respective switches in row 0 are closed and 0 if

FOR SOME APPLICATIONS DIODES ARE NECESSARY

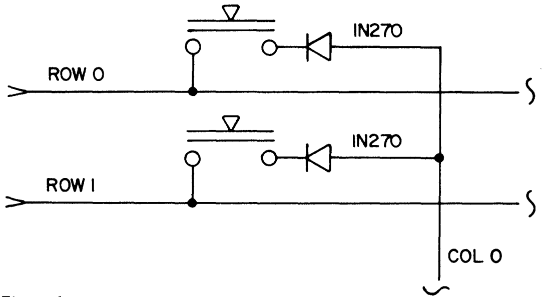


Figure 4

KEYBOARD SCAN ROUTINE (4 x 3 MATRIX)

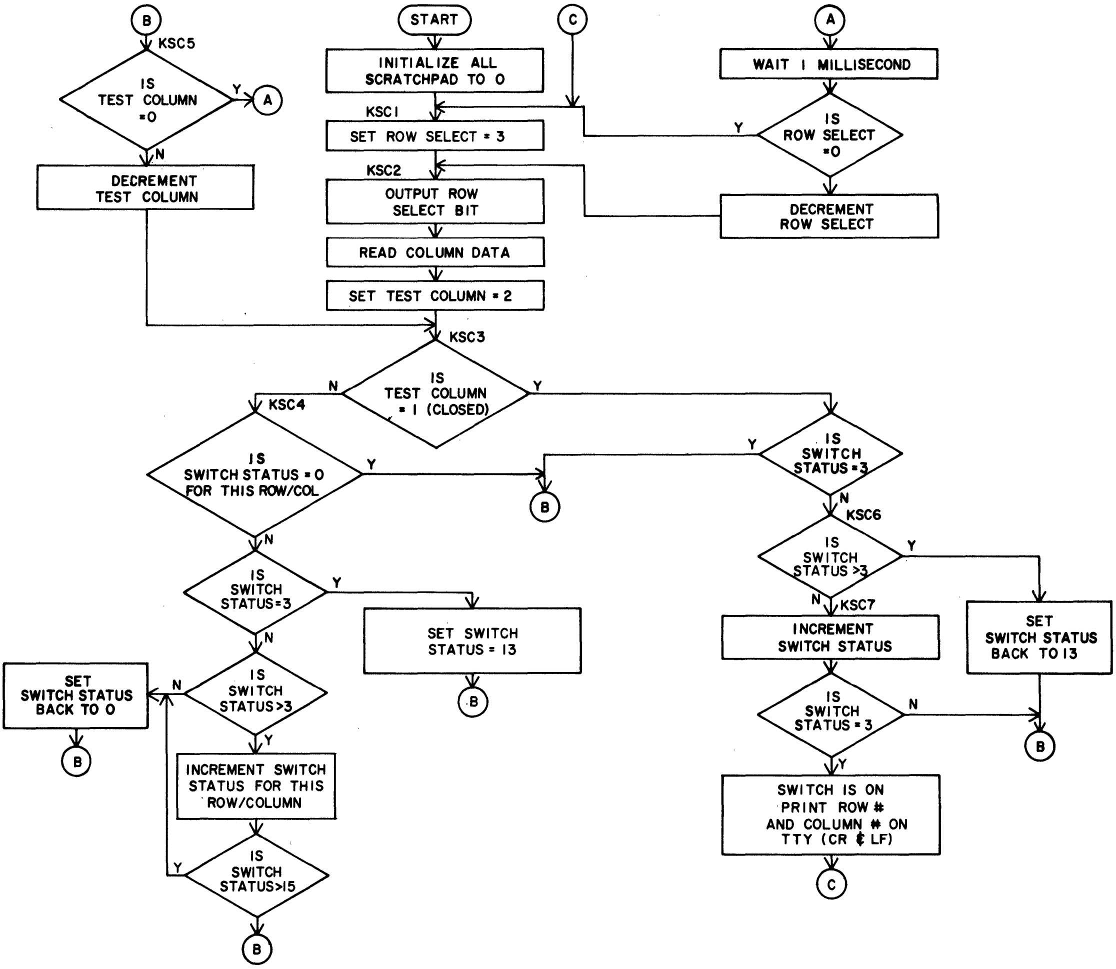


Figure 5

open. (Note: The F8 I/O ports contain internal pull-ups). The other three rows are read similarly.

EXAMPLE SOFTWARE FOR THE 4x3 MATRIX KEYBOARD

An example program was written to run on the F8 Survival Kit to demonstrate software switch sensing and debounce.

One scratchpad register is used to maintain current status for each switch. When a switch is inactive it maintains a status of 0. In order for the switch to be processed, three consecutive scans must occur in which the switch is sensed to be closed.

When a switch is first sensed closed, its status is incremented to 1. In succeeding scans its status is either incremented (if sensed closed) or reset to 0 (if sensed open) until the status reaches 3, thus requiring three consecutive scans with the switch closed.

The switch is then processed,which in the example means the column number and row number are printed on the TTY (terminal).

A status of 3 is maintained by the switch until the first time it is sensed open. At that time its status is set to 13. Then three consecutive scans with the switch open are required to get the switch back to inactive status (0). This is accomplished by incrementing the status (if sensed open) or resetting the status to 13 (if sensed closed) until it reaches 15. The status is then reset to 0. As long as bounce occurs, however, the status will be reset to 13.

The flowchart (fig. 5) shows the logic described above. Note that at the end of each row scan there is a one millisecond delay which effects an interscan delay of 4 milliseconds for each switch. This means that the switch must be on 'solid' for 8 milliseconds before being processed and off 'solid' 8 milliseconds before becoming inactive again; so the switch will only be processed one time per depression. This debounce time sets the max keyboard entry rate for a given switch at 1 entry/24 milliseconds.

Figure 6 shows the scratchpad register assignments used by the example program.

For an instruction by instruction description of the example program see the listing (fig. 7).

ALTERNATE DESIGN APPROACHES

When more than 16 switches are needed, an additional chip must be used. By adding a 4 to 16 decoder (see figure 8) to select 1 of 16 rows, up to 64 switches can be scanned.

Many off-the-shelf keyboards are available which have a 4x3 or 4x4 physical arrangement, but all switches have one common pole (on the P.C. Board). This type of keyboard can be scanned by using a 4 bit code to select one of up to 16 switches. The code is then decoded by a 4 to 16 decoder which supplies a ground return to the selected switch. The switch common line is then read to sense the condition of that switch (see figure 9).

SCRATCHPAD REGISTER ASSIGNMENTS

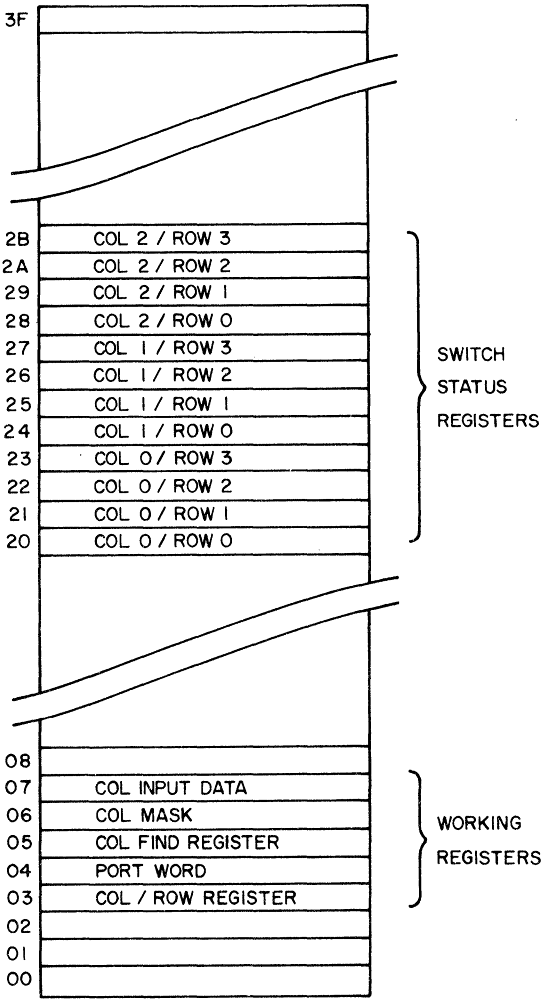


Figure 6

If more ports can be assigned to the keyboard interface, other options may become advantageous. For example, with two ports 16 switches can be read without scanning. The basic requirements such as switch debounce and N-Key rollover will remain regardless of which option is taken. The best approach to a given design application will be determined by the system requirements and structure.

16 x 4 KEYBOARD

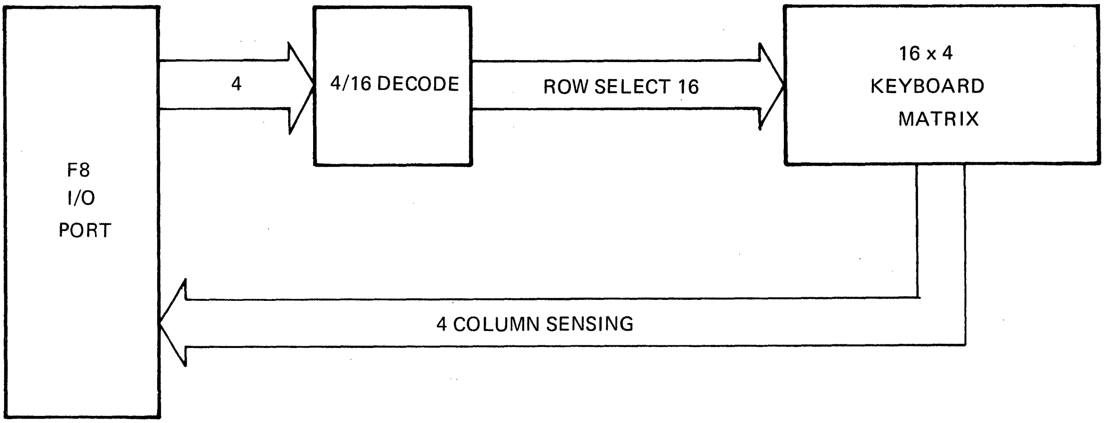


Figure 8

KEYBOARD WITH COMMON POLE

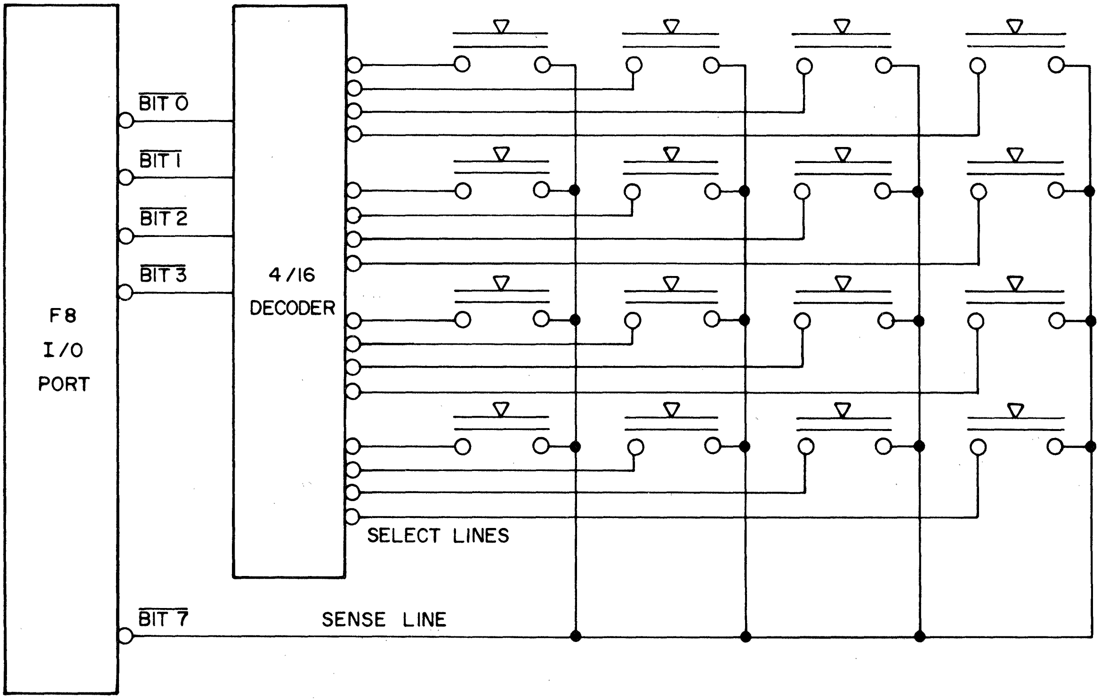


Figure 9

MOSTEK

F8 MICROPROCESSOR SUPPORT

APPLICATION NOTE

F8 Family

USING MOSTEK'S F8 IN A SCANNED SEVEN-SEGMENT DISPLAY APPLICATION

Using Mostek's F8 In A Scanned Seven-Segment Display Application

by Dan Hammond

INTRODUCTION

Many microprocessor based devices require a numeric display as an integral part of the system. For reasons of cost and reliability, it is usually desirable to keep the chip count as low as possible with the microprocessor performing the control logic in software. Time multiplexed digit scanning is a common solution and works very well using a single F8 port for up to 8 digits.

THEORY OF OPERATION

An eight digit display can be scanned with one F8 port (fig. 1) by using half of the port for the BCD number and half for the digit select. When using the digit scanning technique an 'image' of the display must be maintained in memory, with a byte (or half byte) of memory containing the BCD number to be displayed in each of the eight digits. The following five steps show the basic control the software is required to execute:

Step 1 Output digit select and BCD number for this digit (from 'image')

Step 2 Turn on strobe

Step 3 Delay

Step 4 Turn off strobe

Step 5 Increment digit select, return to step 1

The scan rate should be fast enough to prevent the display from 'flickering'. It has been found that a 80 to 100Hz rate is sufficient for a stationary display. An approximate 100Hz rate is achieved in an eight digit display by making the delay in step 3, 1.25 milliseconds.

Maximum brilliance will be provided by leaving the strobe on for the whole delay time. This provides a 1/8 or 12.5% duty cycle. Reducing the strobe width will reduce the duty cycle and cause the display to be dimmer.

Interdigit blanking to prevent a blurring effect is accomplished by strobing the digit decoder after digit select/BCD number data is present on the port and removing the strobe before changing the data (see fig. 2)

EXAMPLE HARDWARE DESIGN

The example design in Figure 3 shows the hardware simplicity in an LED display scanning circuit interfaced to the F8. Bits 0-2 are used to select the digit, bit 3 as a strobe and bits 4-7 for the BCD number.

In this eight digit display the current required from the segment drivers and anode drivers is approximately 6-8 times what it would be for a static non-scanned display of equal brillance because only one digit is receiving current at a time (12.5% Duty Cycle).

NUMERIC DISPLAY BLOCK DIAGRAM

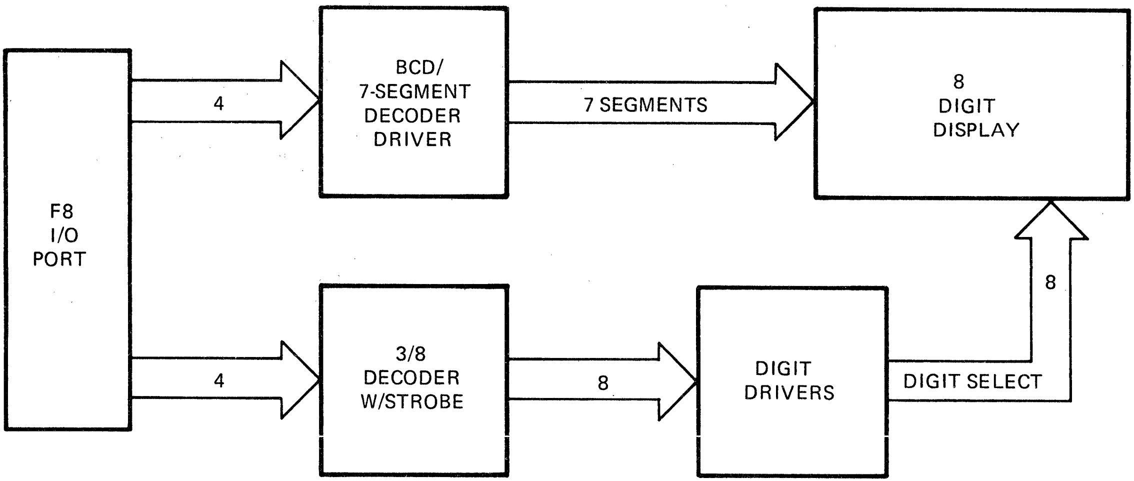


Figure 1

INTERDIGIT BLANKING

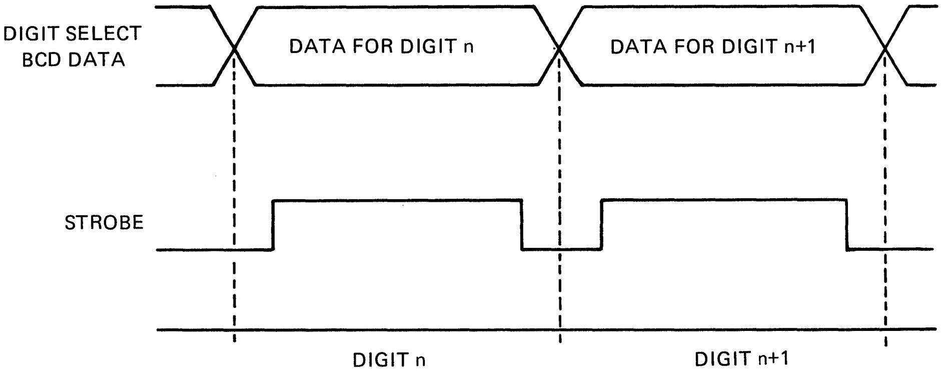


Figure 2

The SN7447 seven segment decoder/driver sinks 40mA per segment which will supply a maximum average current of 5mA per segment to each digit. This is an acceptable current level for many 7 segment LED displays such as the .43 inch HP7650.

Since the anode transistors must drive seven segments, they will be required to source 280mA peak at a 12.5% duty cycle. Many discrete transistors (such as the 2N2907) and transistor arrays will handle this load.

7 SEGMENT DISPLAY INTERFACE

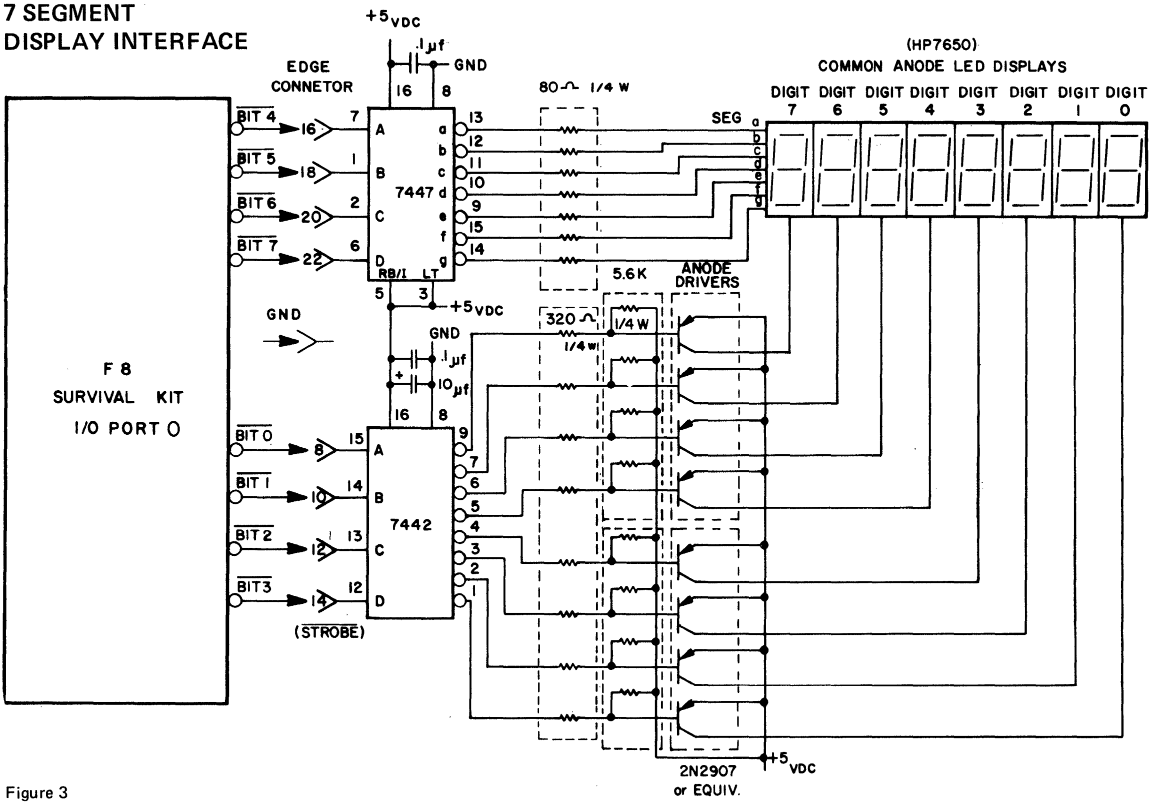


Figure 3

EXAMPLE CONTROL SOFTWARE

The 'MAIN PROGRAM' flow chart (Fig. 4) shows the initialization needed to start the scanning process. The main program must provide a means of entering numbers into the RAM image of the display in addition to the other processing required by the system.

As the BCD numbers are entered into the 'image' the interrupt service routine named 'SCAN' displays them. Note that the flow chart (Fig. 5) for 'SCAN' contains the five basic steps described in The Theory of Operations section (page 2).

The timer in the SMI (Static Memory Interface) chip provides periodic interrupts resulting in a continuous scan of the display. Therefore, only the SMI timer constant has to be changed in order to adjust the scan cycle delay. A key advantage to this interrupt scheme is that it effects a very minimal time burden on the processor. Specifically, every 1.5 milliseconds the interrupt routine takes less than .1 milliseconds to maintain the display scan, using less than 6% of F8 processor time. (It should be noted here that if a scanned keyboard is in the system, the timer interrupt service routine could also scan the keyboard and maintain its status).

In the example program the last eight F8 scratchpad registers are assigned to be used for the display image, register 0 for the display port image, register 1 for saving the display ISAR (Indirect Scratchpad Address Register), register 2 for saving the 'MAIN PROGRAM' ISAR, register 8 for saving the accumulator, and register J (9) for saving the status word (w), (See Fig. 6).

All six bits of ISAR are used to address the 'image' with the least significant three bits also defining the digit in which the addressed 'image' data is to be displayed. The instruction on line number 12 of figure 7 (LR A, I) loads the contents of the location in the scratchpad 'image' addressed by ISAR into the accumulator, then increments ISAR (preparing ISAR for the next interrupt).

MAIN PROGRAM

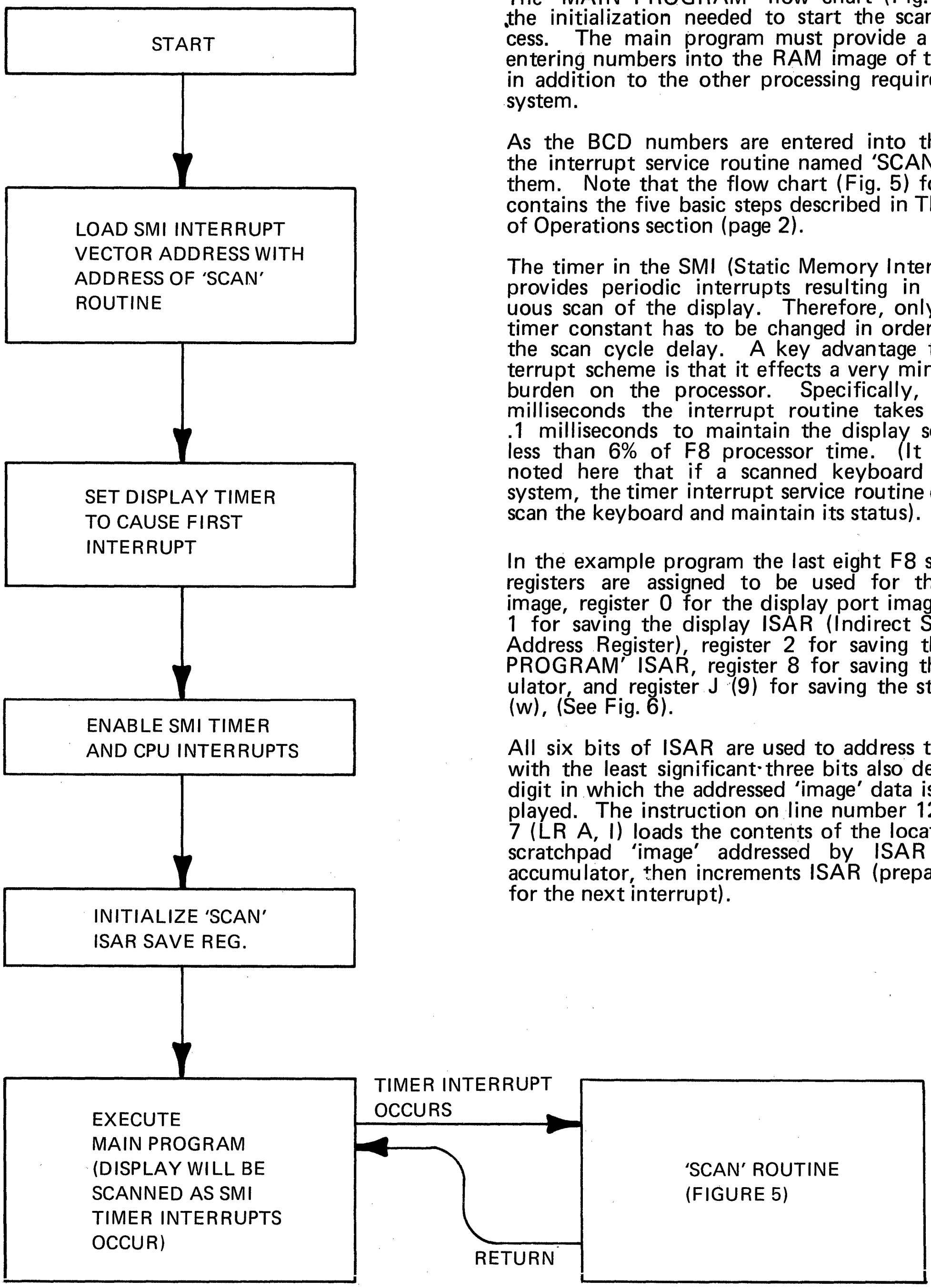

START
LOAD SMI INTERRUPT VECTOR ADDRESS WITH ADDRESS OF 'SCAN' ROUTINE
SET DISPLAY TIMER TO CAUSE FIRST INTERRUPT
ENABLE SMI TIMER AND CPU INTERRUPTS
INITIALIZE 'SCAN' ISAR SAVE REG.
EXECUTE MAIN PROGRAM (DISPLAY WILL BE SCANNED AS SMI TIMER INTERRUPTS OCCUR)
TIMER INTERRUPT OCCURS
RETURN
'SCAN' ROUTINE (FIGURE 5)

Figure 4

INTERRUPT SERVICE ROUTINE FOR NUMERIC DISPLAY

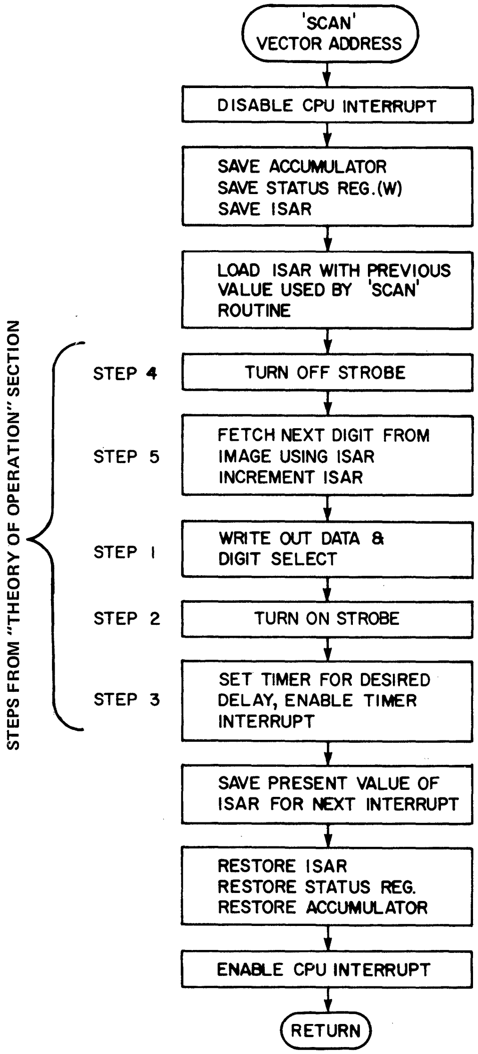
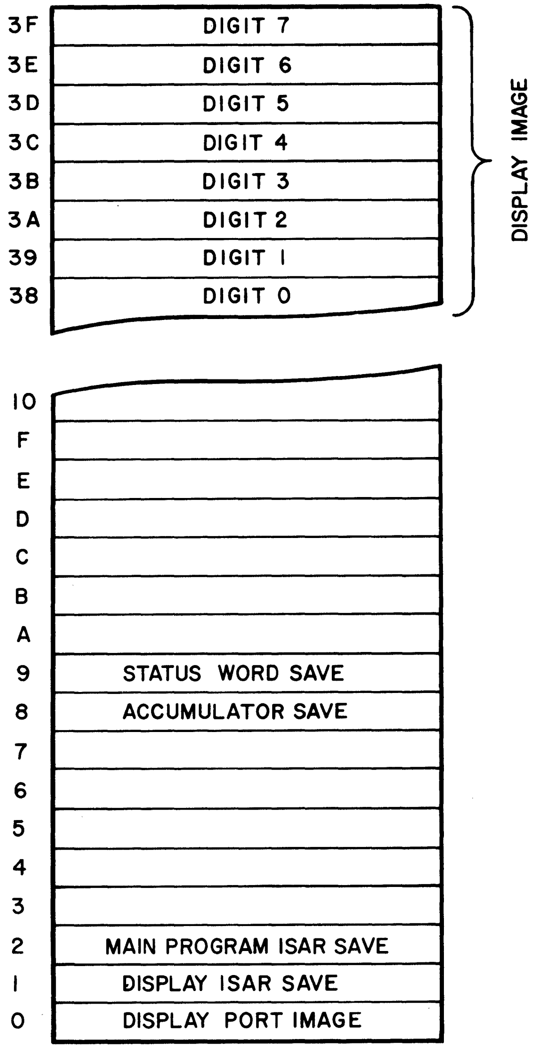

'SCAN' VECTOR ADDRESS
DISABLE CPU INTERRUPT
SAVE ACCUMULATOR
SAVE STATUS REG.(W)
SAVE ISAR
LOAD ISAR WITH PREVIOUS VALUE USED BY 'SCAN' ROUTINE
STEP 4
TURN OFF STROBE
STEP 5
FETCH NEXT DIGIT FROM IMAGE USING ISAR
INCREMENT ISAR
STEP 1
WRITE OUT DATA & DIGIT SELECT
STEP 2
TURN ON STROBE
STEP 3
SET TIMER FOR DESIRED DELAY, ENABLE TIMER INTERRUPT
STEPS FROM "THEORY OF OPERATION" SECTION
SAVE PRESENT VALUE OF ISAR FOR NEXT INTERRUPT
RESTORE ISAR
RESTORE STATUS REG.
RESTORE ACCUMULATOR
ENABLE CPU INTERRUPT
RETURN

Figure 5

F8 SCRATCHPAD REGISTER USAGE MAP



3F	DIGIT 7
3E	DIGIT 6
3D	DIGIT 5
3C	DIGIT 4
3B	DIGIT 3
3A	DIGIT 2
39	DIGIT 1
38	DIGIT 0
10	
F	
E	
D	
C	
B	
A	
9	STATUS WORD SAVE
8	ACCUMULATOR SAVE
7	
6	
5	
4	
3	
2	MAIN PROGRAM ISAR SAVE
1	DISPLAY ISAR SAVE
0	DISPLAY PORT IMAGE

Figure 6

Output port H'F' is the timer constant register in the SMI chip (see line 1C in figure 7). Port H'E' is a register used to enable the timer interrupt in the SMI (line 1F). Note also that all outputs to the display port are 'OUTS 0' selecting port 0 (line E, line 17 & line 19).

The program listing (Fig. 7) contains comments that specify the purpose of each instruction.

SDB RESIDENT ASSEMBLER LISTING Figure 7

```
                 OBJECT
LINE # ADDRESS   CODE                    SOURCE CODE       COMMENTS
0000                       *                               INTERRUPT SERVICE ROUTINE
0001                       *                               FOR NUMERIC DISPLAY
0002                       *
0003                       *
0004                             ORG   H'700'
0005 0700 1A               SCAN  DI                        DISABLE CPU INTERRUPTS
0006 0701 58                     LR    8,A                 SAVE ACCUMULATOR
0007 0702 1E                     LR    J,W                 SAVE STATUS REG
0008 0703 0A                     LR    A,IS                LOAD ISAR INTO ACCUMLATOR
0009 0704 52                     LR    2,A                 SAVE ISAR FROM MAIN PROGRAM
000A 0705 41                     LR    A,1                 LOAD ACCUMULATOR WITH PREV ISAR
000B 0706 0B                     LR    IS,A                LOAD ISAR FOR SCAN
000C 0707 40                     LR    A,0                 LOAD PREVIOUS DISPLAY PORT DATA
000D 0708 21 F7                  NI    H'F7'               MASK OUT STROBE BIT
000E 070A B0                     OUTS  0                   TURN OFF STROBE
000F 070B 0A                     LR    A,IS                LOAD ISAR INTO ACCUMULATOR
0010 070C 21 07                  NI    7                   MASK OUT ISAR(U)
0011 070E 50                     LR    0,A                 ISAR(L) TO RO FOR DIGIT # SELECT
0012 070F 4D                     LR    A,I                 GET BCD DATA USING ISAR,INC ISAR
0013 0710 15                     SL    4                   MOVE IT TO MS HALF OF ACCUMULATO
0014 0711 C0                     AS    0                   ADD DIGIT # TO BCD DATA
0015 0712 18                     COM                       INVERT DATA SINCE PORTS NEG TRUE
0016 0713 21 F7                  NI    H'F7'               MASK OUT STROBE BIT
0017 0715 B0                     OUTS  0                   WRITE NEW DATA OUT (NO STROBE)
0018 0716 22 08                  OI    8                   STROBE BIT ON
0019 0718 B0                     OUTS  0                   TURN ON STROBE
001A 0719 50                     LR    0,A                 SAVE DISPLAY PORT DATA
001B 071A 20 C4                  LI    H'C4'               TIMER CONSTANT
001C 071C BF                     OUTS  H'F'                WRITE TO SMI TIMER
001D 071D 73                     LIS   3                   LOCAL INTERRUPT ENABLE BITS
001E 071E BE                     OUTS  H'E'                ENABLE LOCAL INTERRUPTS
001F 071F 0A                     LR    A,IS                LOAD ISAR INTO ACCUMULATOR
0020 0720 51                     LR    1,A                 SAVE DISPLAY SCAN ISAR
0021 0721 42                     LR    A,2                 LOAD MAIN PROGRAM ISAR VALUE
0022 0722 0B                     LR    IS,A                RESTORE ISAR WITH IT
0023 0723 1D                     LR    W,J                 RESTORE STATUS REG
0024 0724 48                     LR    A,8                 RESTORE ACCUMULATOR
0025 0725 1B                     EI                        ENABLE CPU INTERRUPTS
0026 0726 1C                     POP                       RETURN TO MAIN PROGRAM
0027                             END
00

SCAN 0700
```

ALTERNATE DESIGN APPROACHES

There are several other approaches to a numeric display interface with the F8. For example, the BCD to seven segment conversion and 3/8 digit decoding could be done in software. This approach (Fig. 8) uses two ports.

If four ports are available, the display could also be driven statically, with each port controlling two digits. This approach (Fig. 9) would require one BCD to 7-segment decoder/driver (and 7 resistors) for each digit.

The best design approach depends on the application and the number of F8 ports available.

ALTERNATE SCANNING APPROACH

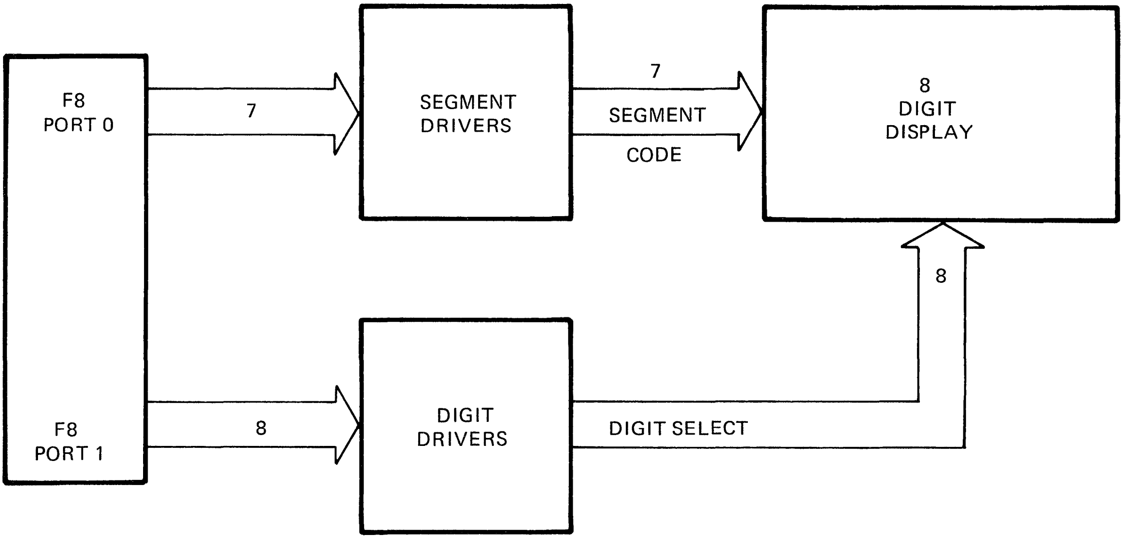


Figure 8

STATIC DISPLAY APPROACH

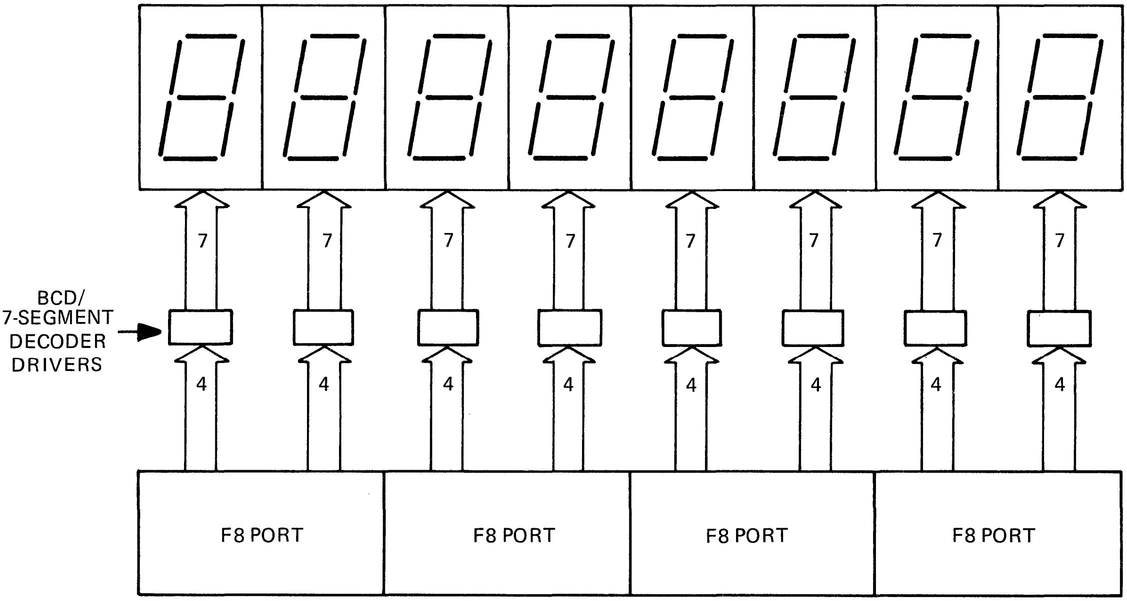


Figure 9

EXPANDING MOSTEK'S F8 EXTERNAL INTERRUPT CAPABILITIES

Expanding Mostek's F8 External Interrupt Capabilities

by Jim Vittera

INTRODUCTION

One of the considerations involved in the design of any microprocessor based system is how to structure the interface between the peripherals (inputs or devices being controlled) and the CPU. The data line interface is usually dictated by the peripheral itself (e.g., a paper tape reader is eight bits of parallel data, a teletype is two lines of serial data, and a switch or front panel lamp usually only requires one line of data). The control lines of these peripherals however, can be handled in one of two basic ways by the system designer. The first method of handling these control lines, which is probably the most common, is to have the CPU periodically scan the control lines (connected to a I/O Port) to see if they require service. This is done by a small program which inputs the control lines through an I/O Port into the accumulator. They are then tested to determine if a line is active and the program flow diverted to service the active control line. The second method is to allow these control lines to interrupt the processor and divert program flow to service that peripheral. Servicing of these control inputs in a F8 based system is the topic of this application note with particular emphasis placed on implementing interrupt driven systems.

SCANNED VS INTERRUPT DRIVEN SYSTEMS

The basic difference between scanned and interrupt driven systems is that in a scanned system the peripherals are checked periodically to see if they need service. This periodic interval can be determined by the count down of a hardware timer (a software timer could be used, but the CPU would be tied up implementing a ripple counter—not a very effective use of the microprocessor). This technique is good for peripherals which can wait for service by the CPU (the maximum time would be the time between counter outputs), and good examples are any peripheral activated or observed by a human. For example a keyboard/display might be scanned at 1 ms intervals, as determined by the timer, which would be slow by microprocessor standards but exceedingly fast by human standards (after pressing a key or throwing a switch an extra 1 ms delay in service would not be noticeable).

FLOWCHART FOR SCANNING N CONTROL LINES

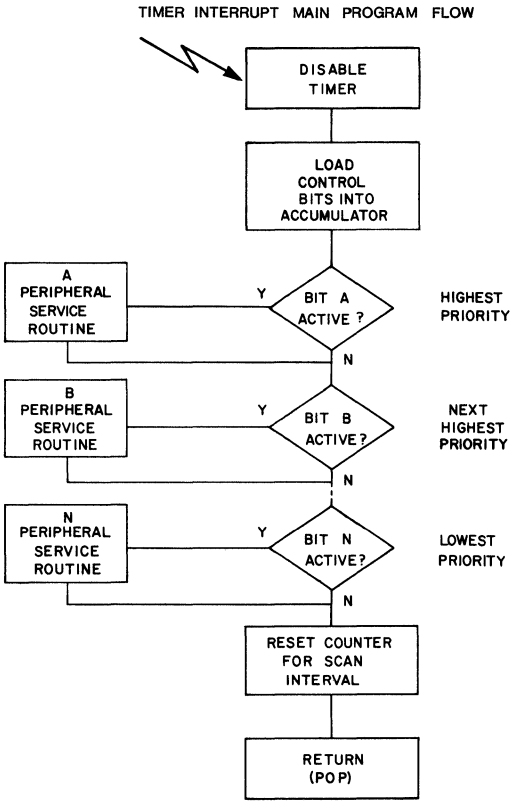

TIMER INTERRUPT MAIN PROGRAM FLOW
DISABLE TIMER
LOAD CONTROL BITS INTO ACCUMULATOR
A PERIPHERAL SERVICE ROUTINE
Y
BIT A ACTIVE ?
N
HIGHEST PRIORITY
B PERIPHERAL SERVICE ROUTINE
Y
BIT B ACTIVE?
N
NEXT HIGHEST PRIORITY
N PERIPHERAL SERVICE ROUTINE
Y
BIT N ACTIVE?
N
LOWEST PRIORITY
RESET COUNTER FOR SCAN INTERVAL
RETURN (POP)

Figure 1

On the other hand many microprocessors are involved in the control of fast peripherals (Floppy Disk) or real time systems where quick response by the processor is required. In these situations, interrupt driven systems are mandatory, because the processor can be diverted from its present task to service the interrupting device in the order of tens of microseconds. Scanned systems are usually perferred by the system designer because they usually require less hardware, especially when implemented in a F8 System with its hardware timers. Figure 1 is a flow chart of a scanned system where the interval between scans is determined by the value preset into the timer. Note that priority is established by the order in which the control bits are tested and can be changed entirely by software.

HARDWARE VECTORED INTERRUPTS

The interrupt technique used by F8 Family devices capable of interrupting the CPU (PSU, PIO, or SMI) is to have the interrupting device provide to the CPU a Interrupt Vector unique to that interrupt. The CPU then loads this vector directly into the program counter (saving the previous program counter in P) directing the CPU to the service routine for this interrupt. This technique provides a fast response to the interrupt because no time is consumed in polling to locate the interrupting device. In addition to providing automatic vectoring of the interrupts, the F8 devices provide automatic prioritizing of the interrupts. Priority is determined by the placement of the interrupting device in a daisy chain structure — a location closer to the CPU means higher priority — as shown in Figure 3. ICB is an output from the F8 CPU to indicate if interrupts have been enabled by the use of an EI instruction in the program being executed. ICB goes low when interrupts have been enabled, thereby enabling the daisy chain of interrupting devices. One or more of the three EXT INT inputs shown goes low signaling a request(s) for service by one or more of the peripherals. The device or devices that have EXT INT low now pull their INT REQ line low (assuming interrupts are not disabled at the local level) signaling the processor to begin an interrupt service sequence. The status of the INT REQ line is tested by the CPU at the end of every instruction which is not privileged. Privileged instructions cannot be interrupted so the CPU waits until the end of the next instruction (which is not privileged) to test the INT REQ line. When the CPU finds the INT REQ line low it begins the interrupt sequence by saving the Program Counter in P and using the ROM Control Lines to command the interrupting peripheral to transfer its vector address to the Program Counter. The 3851 is the highest priority device in Figure 3 and if its EXT INT line is low it sets its PRI OUT signal high thereby disabling all lower priority devices and outputs its vector address on the Data Bus. Should the PSU not be the interrupting device, it leaves its PRI OUT signal low passing the request to the second device in the chain (the PIO in this case). If the PIO is interrupting, it raises its PRI OUT line to a logic one and outputs its vector address. PRI OUT going high prevents all devices of lower priority from outputing their vector address even though they may be trying to interrupt. Twenty two cycles of the Φ clock are required to complete this interrupt vector fetch sequence. The next event that occurs is an instruction fetch from the location specified from the vector address. The SMI doesn't have a PRI OUT signal therefore it must be the lowest priority device in the system. The time required to get to an interrupt service routine can be calculated as shown in Figure 2 (at a 2 MHz Φ rate).

INTERRUPT VECTOR FETCH

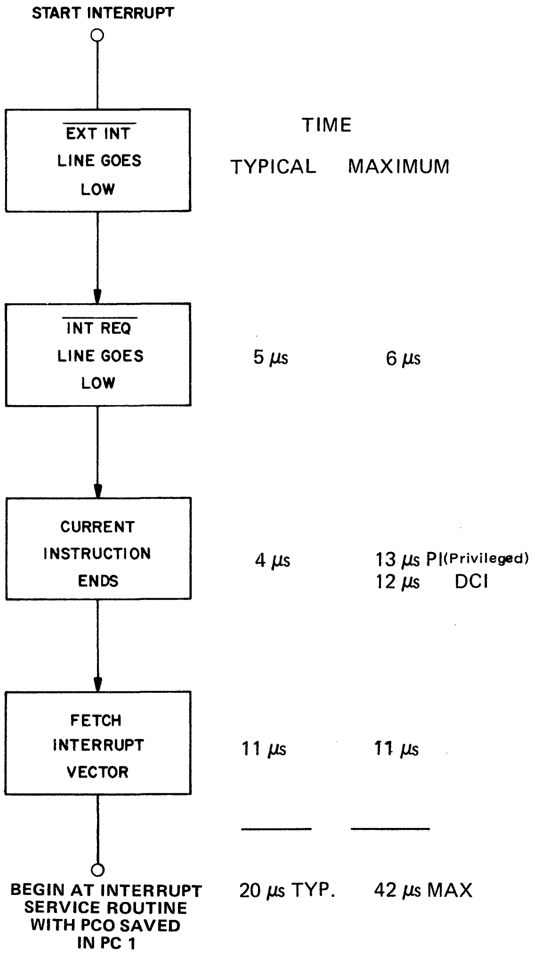


Figure 2

The time from an interrupt striking to the start of execution of its service routine is highly dependent on the instruction being executed at the time of the interrupt. The maximum number was based on a long privileged instruction such as PI followed by a long non-privileged instruction such as DCI. The typical instruction time is based on a 2 cycle instruction although many F8 instructions are one byte/one cycle instructions. The 6.0µs max number represents the propagation delay through the peripheral device from EXT INT to INT REQ (interrupts from the timer do not incur this delay). Once the INT REQ is recognized, 22 cycles are required to stack the program counter and fetch the interrupt vector. One technique that can be used to minimize the maximum delay that would be incurred upon an interrupt is to constrain the instructions that are executed when the interrupt is expected. A method that would reduce the maximum delay from the interrupt striking to the execution of the first instruction of the service routine would be to put the processor in a branch on self loop (BR*). This essentially provides a wait for interrupt situation with the CPU running in a loop waiting for the interrupt.

EXPANDING INTERRUPT INPUTS IN A MINIMUM SYSTEM

In a two-chip F8 microcomputer system (MK 3850 CPU and MK 3851 PSU) the system can be interrupted by either the timer in the PSU or the EXT INT line of the PSU. Thirty-two lines of bidirectional I/O are available and it may be desirable to have more than one input capable of interrupting

F8 SYSTEM INTERRUPT CONNECTION

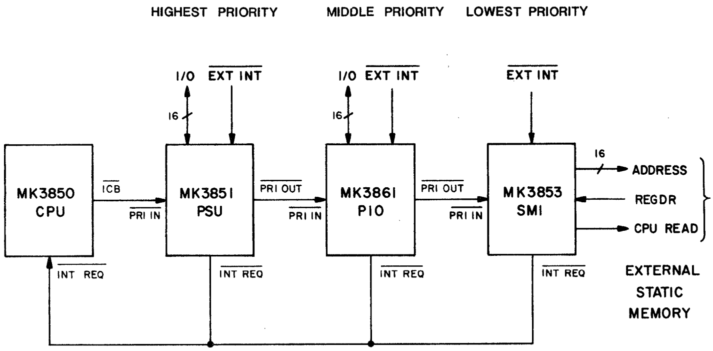
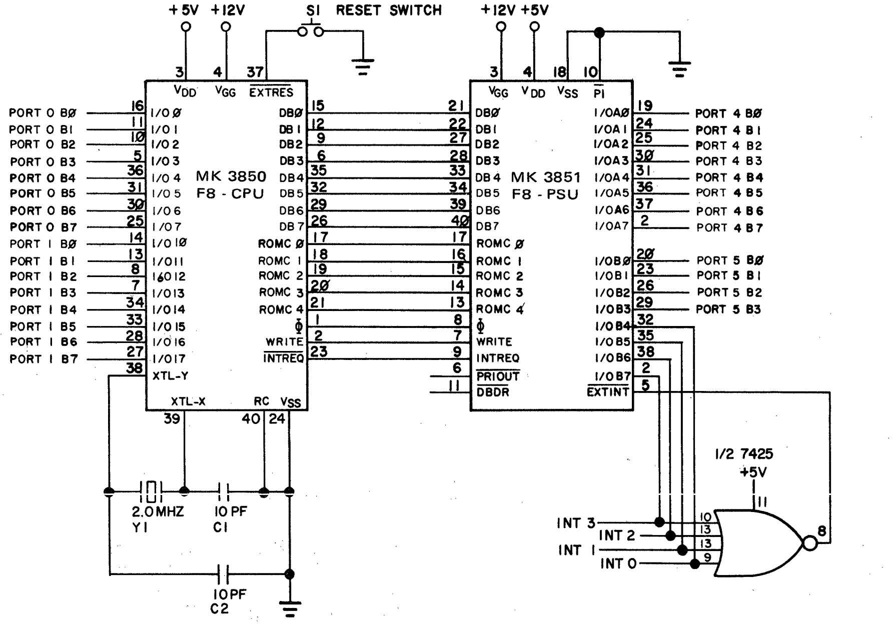


Figure 3

rupting the system. Figure 4 depicts this minimum F8 system, with four signals (INTO-INT3) capable of interrupting the system. The four external interrupting signals are defined active high and the presence of any one in the high state causes the output of the NOR gate to go low causing the interrupt. The interrupt service routine flowchart to locate the interrupting input is shown in Figure 5, with the actual program in Figure 6.

This service routine is entered with the interrupts automatically disabled at the CPU so that no further interrupts can occur until the interrupt is cleared by its service routine. The port containing the INTO-INT3 signals is loaded into the accumulator and tested to determine if bit 7 is low (a positive number). If bit 7 is low INT3 is active and the branch is taken to the service routine for INT3 (SERV3) (there is an inversion from the Port to the accumulator). If bit 7 is high a shift left one instruction is performed on the accumulator and it is again tested for bit 7 = 0 (bit 6 shifted). This process continues until all four of the interrupt lines have been tested. If an active interrupt bit has been found the proper service routine is branched to in order to service the active device and

EXPANSION OF INTERRUPT INPUTS IN A MINIMUM F8 SYSTEM



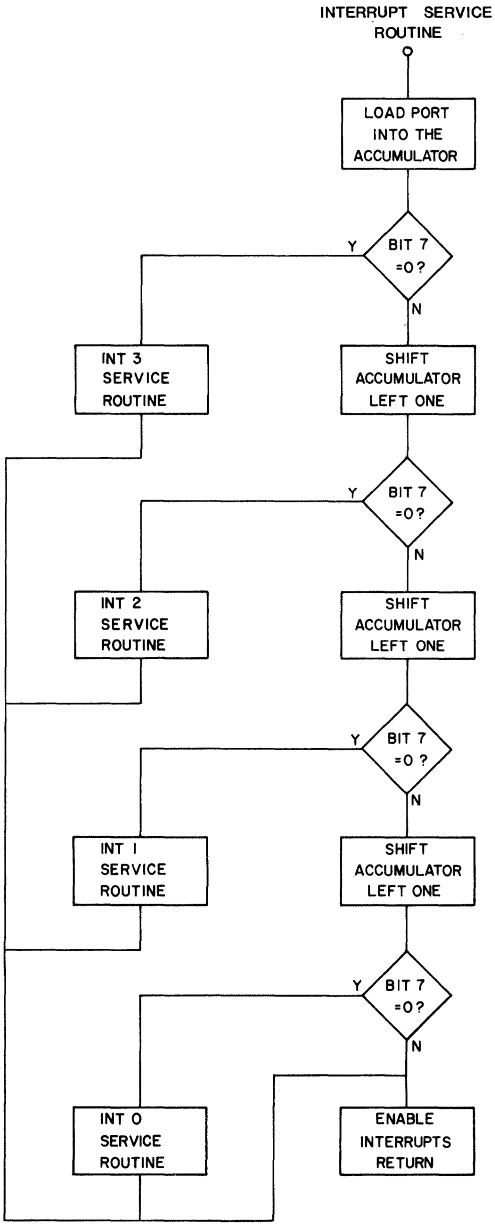
NOTE: MK 3870 Single Chip F8 will replace this two chip minimum system.

Figure 4

clear the interrupt. The routine ends by enabling the interrupts at the CPU and returning to the main program flow should no interrupt be found.

The additional time required to locate the active interrupt is a function of which interrupt is active due to the polling used. As shown in Figure 6, the additional delay to service interupts produced by polling varies from 15 µs for the highest priority device to 42 µs for the lowest priority device. To these times must be added the delays calculated earlier of 20 µs typical and 42 µs maximum which is required to get to the polling routine.

FLOWCHART OF INTERRUPT SERVICE ROUTINE



INTERRUPT SERVICE ROUTINE TO LOCATE INTERRUPTING DEVICE

CUMM µS	µS				
	8	INTSVC	INS	PORT	GET SIGNALS
INT 3 15	7		BP	SERV 3	INT 3 ACTIVE 7
	2		SL	1	NO, SHIFT LEFT
INT 2 24	7		BP	SERV 2	INT 2 ACTIVE 7
	2		SL	1	NO, SHIFT LEFT
INT 1 33	7		BP	SERV 1	INT 1 ACTIVE 7
	2		SL	1	NO, SHIFT LEFT
INT 0 42	7		BP	SERV 0	
	4		EI		ENABLE INTERRUPTS
	4		POP		RETURN

Figure 6

SINGLE CHIP MICROCOMPUTER

The MK 3870 Single Chip Microcomputer is the natural evolution of the F8 chip set. It will combine the functions of the 3850/3851 onto a single chip with the additions of another 1K bytes of ROM storage and an improved timer/interrupt structure. The techniques discussed in this application note apply also to the single chip F8 as it is software and hardware compatible with the multiple chip F8 family.

F8 Family

MOSTEK®

3870/F8 MICROCOMPUTER SUPPORT

Application Note

F8 Family

SUBROUTINE NESTING AND MULTIPLE INTERRUPT HANDLING

INTRODUCTION

The 3870 and F8 Microcomputer Families are quickly becoming recognized as a cost effective method of placing computing power into types of equipment which couldn't have justified computer control just a short time ago. The falling cost per computer function afforded by advances in Metal-Oxide Semiconductor-Large Scale Integration (MOS—LSI) has brought computer technology and techniques into areas where until now, mechanical controller's, random logic and relay logic predominated. The availability of a large number of Input/Output pins in the 3870 Microcomputer coupled with its minimum system configuration of just one device, makes it an ideal replacement for many previously used control devices. The purpose of this note is to discuss the use and implementation of subroutines and interrupts as they apply to programming an F8 based microcomputer system. The intent of this note is to discuss the use of subroutines and interrupts for the hardware designer who might not be totally familiar with the programming of a computer.

SUBROUTINES

A subroutine is a sequence of computer instructions or mnemonics which can be called or used in several portions of the computer program. The purpose of a subroutine is to reduce the total length of a computer program by consolidating in one portion of the program a sequence of instructions that are used in several different areas of the program. When this subroutine is required the program counter contents are replaced with the starting address of the subroutine. At the end of the subroutine the program is transferred back to the main body of the program.

Figure 1 depicts program flow when using subroutines. The main program calls a subroutine which causes the program counter to be loaded with the address of the subroutine. The last statement of the subroutine causes a return back to the main program flow by retrieving the saved program counter value, forcing a return to the main program flow. The subroutine is called again any place in the main program flow where the sequence of instructions contained in the subroutine is required. Every time the subroutine is called a savings in program length (and ROM size) equal to the length of the subroutine (minus three) is realized compared to a program which doesn't use subroutines. Many times a subroutine will call another subroutine resulting in what is referred to as nested or multi-level subroutines. Nested subroutines in an F8 system will be discussed in this note.

INTERRUPTS

Interrupts are used in a microcomputer system to make it responsive to the device it is controlling.

PROGRAM FLOW WHEN USING SUBROUTINES

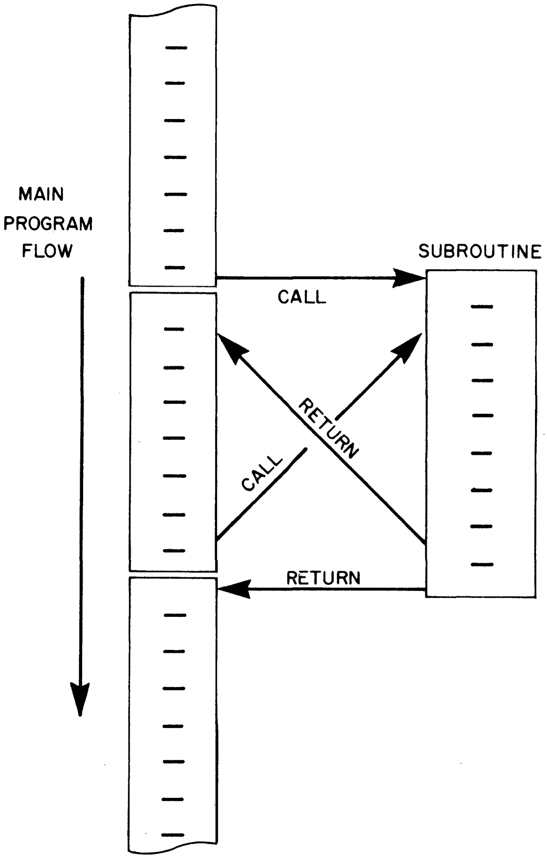


Figure 1

By interrupting the microcomputer the I/O device can signal its requirement for attention or service by the microcomputer. As in the case of the subroutine, the interrupt can divert the main program flow to a sequence of instructions called the Interrupt Service Routine (See Figure 2). This routine either inputs or outputs data to the device being controlled. At the end of this service routine the program counter value at the time of the system interrupt is retrieved from a temporary register and reloaded into the program counter to cause a return to the main

PROGRAM FLOW WHEN INTERRUPTED

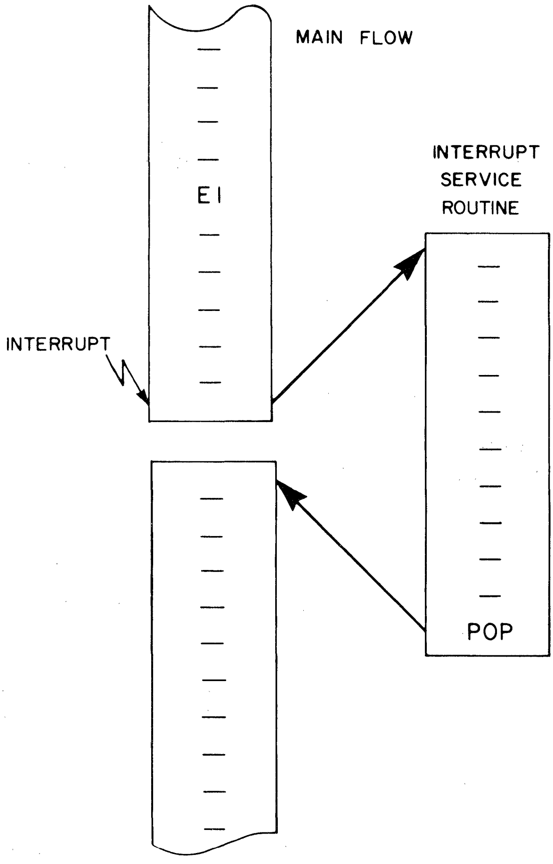


Figure 2

program flow. Interrupts, like subroutines, can be nested because an Interrupt Service Routine could be interrupted by a higher priority device or an Interrupt Service Routine may call a subroutine, which in either case causes nesting.

The F8 instructions which are used to transfer program flow to or from subroutines or interrupts are shown in Figure 3. The Program Counter (PO) holds the address of the next instruction to be executed by the microcomputer while the Stack Register (P)* is a temporary storage location for the Program Counter. In addition two pairs of registers in the Scratchpad have been designated K and Q with instructions that link them to P0 and P. The instructions that link and affect these registers are the following:

(a) Call to subroutine immediate —PI—an instruction which causes the next two bytes in the program to be loaded into the Program Counter (P0) in order to transfer control to a subroutine and saves the old program counter value (return address) in the Stack Register (P).

(b) Call to subroutine—PK—an instruction which causes the contents of the K register to be loaded into the Program Counter while the Program Counter is saved in the Stack Register.

(c) Return from subroutine—POP—an instruction used at the end of a subroutine or interrupt service routine to load the Stack Register back into the Program Counter to return program flow back to the main program. The previous value of the Program Counter is overwritten and lost.

(d) Load—LR P,K—a pair of instructions which
LR K,P
allows the transfer of the Stack Register (P) to the K register in the Scratchpad or vice versa. This switch is useful to save P in preparation for a subroutine or interrupt.

(e) Load — LR P0,Q which allows the transfer of the Program Counter (P0) to the Q register in the Scratchpad.

The following sections of this note will discuss the use of these instructions and registers as well as the general F8 architecture to handle Subroutines, Interrupts and the tradeoffs in doing so.

SUBROUTINES AND/OR INTERRUPTS UP TO TWO LEVELS

Many applications can be handled by two levels of subroutines and/or interrupts. Two levels means that only two return addresses need be saved, which can be handled easily by registers within the F8 for this purpose. The calling of subroutines is under control of the programmer and thus only the return addresses need be saved as other registers (such as the Data Counter) can either be saved by the calling or the called routines if the registers are needed by the subroutine. Interrupts are under control of the programmer only to the extent that they can be masked or enabled. Assuming interrupts are enabled, upon entry to an Interrupt Service Routine, it may not be known which registers in the CPU contain data which cannot be overwritten. In this case these registers should be stored in the scratchpad during the Interrupt Service Routine and be restored before exiting this routine. Examples of using the ISAR to store CPU registers in a push down stack are given in this note but in many cases the programmer will tailor the status saving routine for the specific circumstances of his sytem design (by using specific Scratchpad Registers to save CPU Registers).

Figure 4 shows the instructions usually used to call a subroutine (one level deep) in an F8 system. SUBA1 is the symbolic name of the two byte address of the subroutine and PI causes the return address (XXXX) to be saved in P. POP reverses the procedure at the end of the subroutine causing P0 to be reloaded with the address saved in P causing the program flow to return to the next instruction in the main flow (XXXX). Response to an interrupt from the main flow is similar to this example except that the interrupt causes a path similar to 1 to the Interrupt Service Routine with the address (vector) being supplied by the interrupting device and loaded into P0.

To call a second subroutine or to respond to an interrupt from SUBA1 the instructions in Figure 5 could be used. In this case PI SUBA1 transfers the

* Earlier versions of F8 literature labled the Program Counter PC0 and the Stack Register PC1.

F8 REGISTERS USED IN SUBROUTINES AND INTERRUPTS

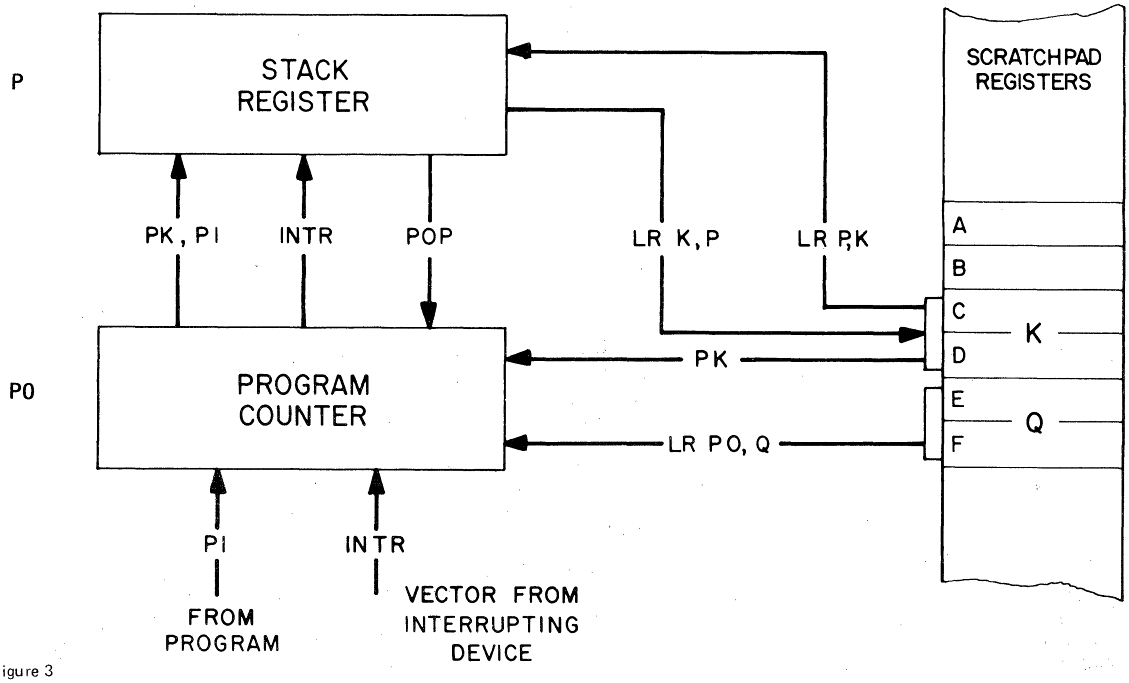


Figure 3

ONE LEVEL SUBROUTINES OR INTERRUPTS

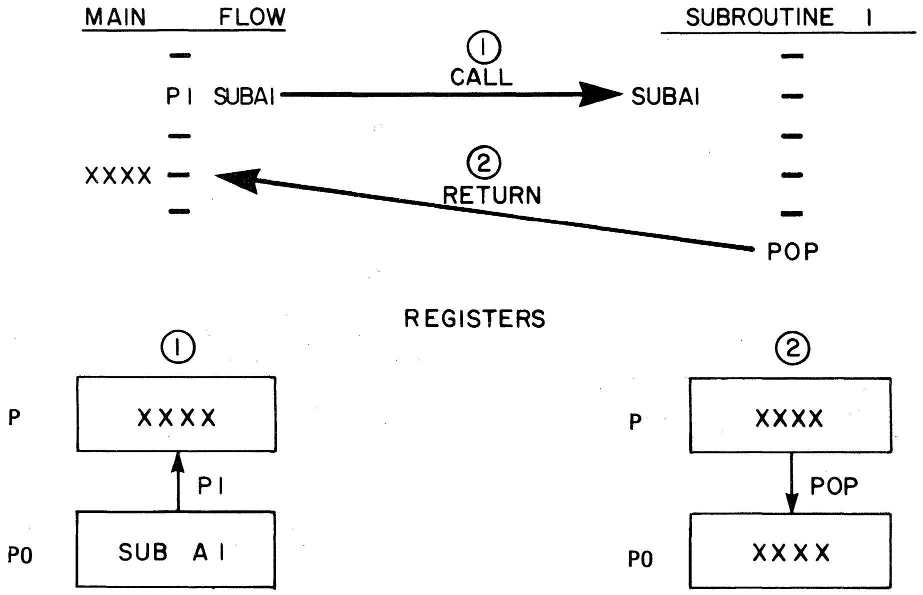


Figure 4

program flow to SUBA1 while saving the return address (XXXX) in the Stack Register (P). Subroutine 1 now transfers P to K in preparation for another subroutine or an interrupt (note that if an interrupt occurs during the PI SUBA1, LR K,P sequence it'will not be serviced until after the LR K,P instruction because PI is privileged). Subroutine 2 is called by PI SUBA2 which saves YYYY in P which was just vacated. Program flow transfers to Subroutine 2 and the POP instruction reloads P0 with YYYY from P. At the end of Subroutine 1 the return address is now in K so a PK is used to load XXXX into P0, thereby returning to the main program. Note that LR K,P followed by POP could have been used in place of the PK instruction, but would be 1 byte longer.

Three levels of subroutines or interrupts can be handled by using the Q register to save a return address. Figure 6 shows programming with three levels of subroutines (three levels of interrupts would be handled in the same manner). The first subroutine is called from the main program and the return address is saved in the Stack Register P. At the beginning of SUB1, P is transferred to the K register in preparation for the second subroutine call or interrupt. This second call uses the just vacated P register for storage of the return address to SUB1. Upon entering SUB 2 both P and K contain valid return addresses so that interrupts must be disabled while the contents of K register are moved to the Q register and the contents of P are moved to the K register. Once P is clear, interrupts can be enabled by the use of the EI instruction, allowing a third level of subroutine nesting (as shown in Figure 6) or an interrupt. The return from SUB3 is accomplished by executing the POP instruction which loads the Program Counter with the value RTN2 from the Stack Register P. During the first portion of SUB2, P was moved to K so that a PK instruction will load the Program Counter with RTN1 from the K register. The return address for SUB1 (RTN) was moved to the Q register during the first portion of SUB2 and can be transferred to the Program Counter by the execution of LR P0,Q instruction.

MULTILEVEL INTERRUPTS OR SUBROUTINES

At a minimum when using the F8 in a system with greater than 3 levels of interrupts or subroutines a consistent method of placing return address into the scratchpad must be used to allow their recovery. In many cases it will be desirable to stack more registers than just the return addresses. Previous examples have shown 3 levels deep with the three return addresses in P, K, and Q registers. Any further nesting would destroy either P, K, or Q so the technique to be described is to move K into the scratchpad to make room for another level.

Figure 7 shows a generalized subroutine which automatically transfers P to K and then K into the scratchpad registers. The routines in Figure 7 assume that ISAR (Indirect Scratchpad Address Register) has been initialized at an odd value, probably H'3F'which

TWO LEVEL SUBROUTINES OR INTERRUPTS

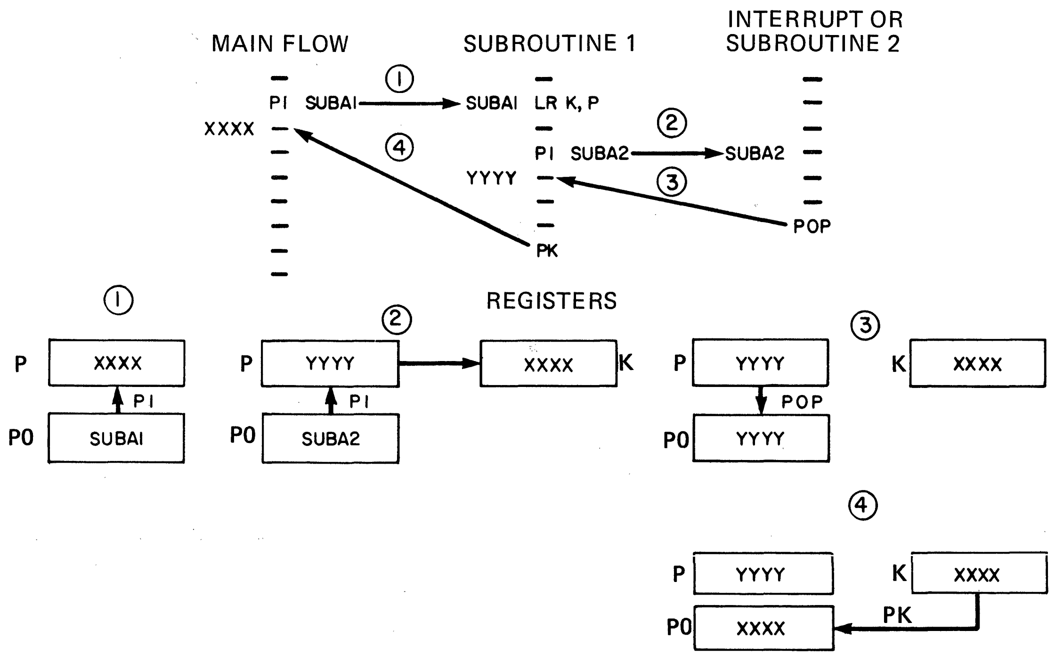


Figure 5

THREE LEVELS OF SUBROUTINES OR INTERRUPTS

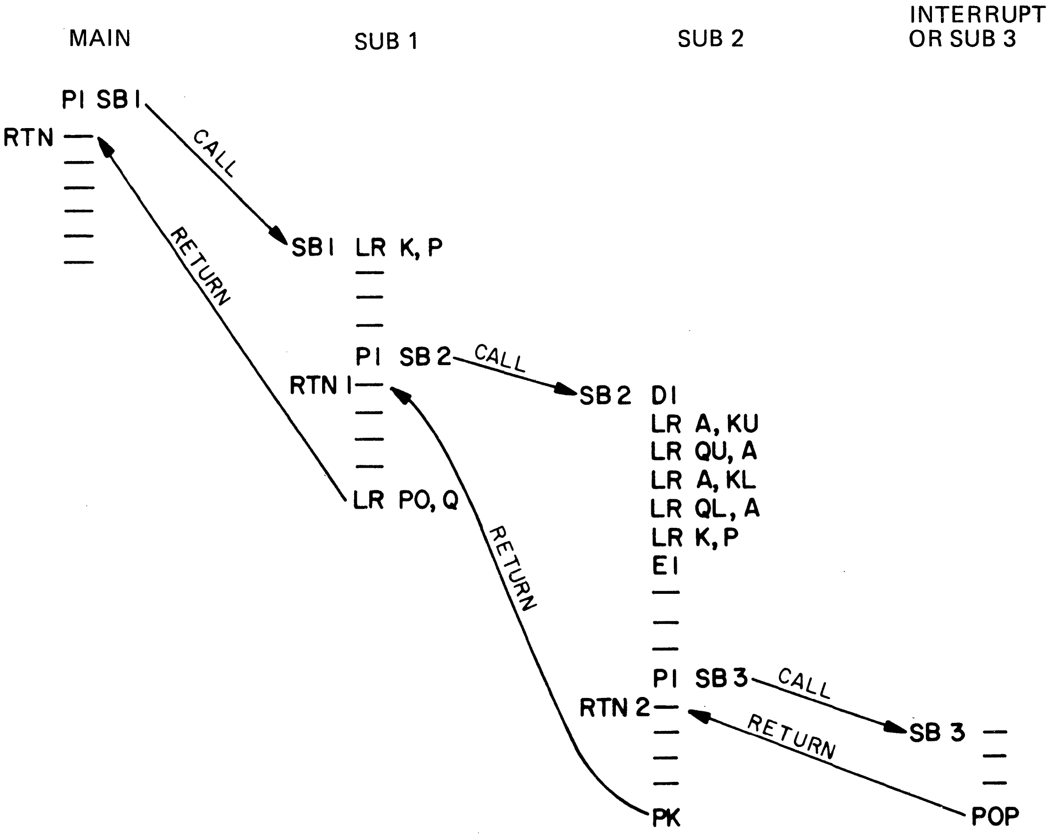


Figure 6

STACKING OF ACCUMULATOR AND STATUS REGISTERS

```
                        SUBX  DI          )
                              LR K,P      }  K to
                              PI  PSHK    )  Stack
                              ________    )
                              ________    }  Body of
                              ________    )  Subroutine
                              PI POPK     )  K from
                                          }  Stack
                              PK          )
      TO STACK                                      FROM STACK

      PSHK   LR  A, KL                              POPK   LR  A, IS
             LR  D, A     KL to stack                      INC            Increment ISAR
             LR  A,KU                                      LR  IS,A
             LR  S,A      KU to stack                      LR  A,I
             LR  A,IS                                      LR  KU,A       Recover KU
             AI  H'FF'    Decrement ISAR                   LR  A,S
             LR  IS, A                                     LR  KL,A       Recover KL
             LR  W,J                                       LR  W,J
             POP          Return                           POP            Return
```

Figure 7

is the top of the scratchpad registers. The odd starting value is required to insure that ISAR is not pointing to an 8 byte buffer boundary when the LR D, A instruction is executed. (The LR D,A instruction loads the accumulator from the scratchpad location pointed to by ISAR and then does a modulo 8 decrement of ISAR. This means that only the lower three bits of ISAR are decremented resulting in an 8 byte range for these auto decrementing and autoincrementing instructions which does not allow crossing of page boundarys. By initializing ISAR at an odd value, every time the LR D,A instruction is executed ISAR will be odd and therefore will not have to cross page boundaries which are even.) The decrement from even values is accomplished by loading ISAR into the accumulator and adding hexidecimal FF to it, which results in an 8 bit decrement of ISAR's contents. PSHK then moves the contents of K onto the stack and leaves ISAR pointing to the next empty location thus implementing a push-down stack. When in the body of the subroutine both P and K are clear, allowing a call to another subroutine of this format or the enabling of interrupts to allow interrupting out of this subroutine (return address would be held in P). POPK is called to recover the subroutine return address and place it in the K register. Note here that the 8 bit increment (INC) is done first to cross the page boundary and point to the last byte stored on the stack. ISAR is used to pull the return address off the stack and place it in the K register. The PK instruction reloads P0 and program flow is returned to the calling routine.

If interrupts are enabled during the body of the program they must be disabled during execution of POPK because this routine is using both P and K registers. The LR W,J instruction allows the user the option to control whether interrupts will be enabled or disabled after execution of PSHK or

SUBROUTINE OR INTERRUPTS NESTED 4 LEVELS DEEP

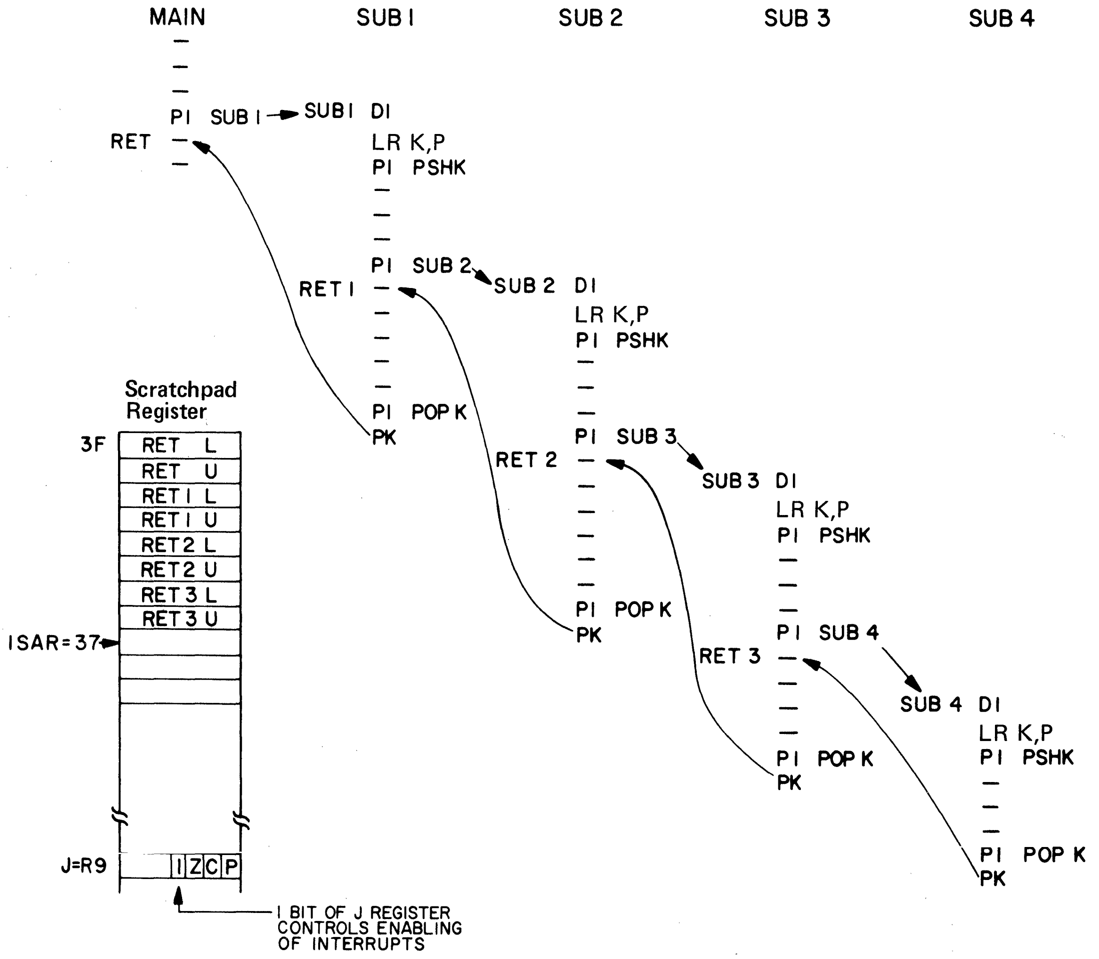


Figure 8

POPK by setting or clearing bit 4 in the J register. Thus if interrupts are desired during a subroutine the following instructions would be used to call SUBX.

MAIN PROGRAM

```
LR A,IS
LR 0,A    SAVE ISAR IN R0
LI  H'09'
LR IS,A   POINT ISAR TO J
LR A,S    GET J INTO A
OI H'10'  SET 1 BIT, INTERRUPTS ENABLE D
LR S,A    A INTO J
LR A,0    RESTORE ISAR
LR IS,A
PI  SUBX  CALL SUBROUTINE
```

If interrupts are not desired during the nesting of subroutines the software can be simplified as follows: Assuming interrupts are disabled instructions DI and LR W,J can be deleted from the PSHK and POPK routines. Also the above calling sequences will not be required because bit 4 in the W register will already be clear.

Figure 8 shows the effects of PSHK used to save the return address on the stack. Note that the Stack Pointer (ISAR) is pointing to the next empty location on the stack and that bit 4 of the J register is controlling whether interrupts are enabled or not through the use of the LR W,J instruction in PSHK and POPK. Bit 4 of the J register is used to control whether interrupts are enabled or not in order to allow two different callers to use this subroutine. If one of the callers was in an interrupt driven portion of the program, bit 4 of the J register could be set to allow interrupts upon returning from the subroutine. If one of the other callers of the subroutine did not want interrupts enabled bit 4 of the J register could be cleared so that no interrupts would be recognized upon returning from the subroutine.

At the end of each subroutine, PI POPK followed by PK will be executed to unload the stack and return program flow back to the correct return address. Note also that interrupts will be allowed at any time except during the execution of the PSHK or POPK routines, their return address being stored in the P register. If the interrupting device service routine needs to call a subroutine, P will have to be pushed onto the stack using the methods previously described.

In many cases it may be desirable to save the major registers within the CPU whenever an interrupt is serviced. Saving registers on the stack frees up all of the computing power of CPU for use by the Interrupt Service Routine. Saving the registers upon the stack rather than in direct scratchpad locations makes the subroutine or interrupt service routine re-entrant (i.e., the routine calls itself without destroying scratch locations). The same philosophy as before can be used to save accumulator and status register on the stack (see Figure 9). Other registers within the machine could be saved using the technique; however, they must be pushed in pairs in order to leave ISAR pointing to an odd register location (since K, DC and DC1 are all 16 bit registers this should not be a limitation).

STACKING OF ACCUMULATOR AND STATUS REGISTERS

```
PSHAW     LR D,A     Accm to stack
          LR A,J
          LR S,A     J Reg to stack
          LR J,W     W reg to J reg
          LR A,IS    ISAR to A
          AI H'FF'   Decrement A
          LR IS,A    A to ISAR

POPAW     LR A, IS   ISAR to A
          INC        Increment A
          LR IS,A    A to ISAR
          LR W,J     J reg to W
          LR A,I     J from stack
          LR J,A     into J reg
          LR A,S     Accm from stack
          POP        Return to caller
```

Figure 9

CONCLUSION

This application note has discussed a general method of handling subroutines and interrupts in an F8 system. Many applications for which the F8 is suited will have minimal subroutines or a minimum number of interrupts so that the internal P and K registers can be used to hold return addresses.

In the cases where deeply nested subroutines or multiple interrupts must be handled a push-down stack can be created in the scratchpad registers or external memory. Software routines were discussed to save return addresses in this stack as well as methods to save the general purpose registers. The user has the option in a F8 system to stack only what is necessary to accomplish his design goals in an optimum manner.

1979 MICROCOMPUTER DATA BOOK

MOSTEK®

MICROPROCESSOR SUPPORT

Z80 Software Development Board (SDB-80E)

HARDWARE FEATURES

□ Available as board or complete system

□ 4K bytes of RAM, expandable on board to 16K Bytes

□ Four 8-bit I/O ports with handshake lines

□ Serial ASCII interface (110-9600 BAUD)

□ Fully buffered for system expandability

□ Four counter/timer channels

□ On board capacity from 5K bytes of PROM to 20K bytes of ROM

□ Double euro-card format

SOFTWARE FEATURES

□ 2K x 8 Operating System in ROM (DDT-80)

□ 8K x 8 assembler/editor in ROM (ASMB-80)

□ Channeled I/O for user convenience

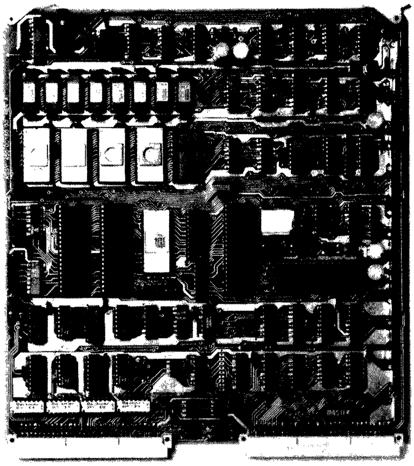
□ Double euro-card format

GENERAL DESCRIPTION

The SDB-80 is a stand-alone microcomputer designed by MOSTEK around the advanced Z80 microprocessor familiy. It contains more on-board firmware and RAM memory than any previously offered single board microcomputer, plus all the features of the industries most sophisticated microprocessor. This board represents the very latest in state-of-the-art technology by utilizing MOSTEK's new 16K Dynamic RAM memories. The SDB-80 also is the first single board microcomputer to offer a complete package of software development aids in ROM. This 10K byte firmware package is included with the SDB-80 and provides the ability to generate, edit, assemble, load, execute, and debug Z80 programs for all types of applications.

USING THE SDB-80

In addition to functioning as a stand-alone development aid, the SDB-80 is fully expandable through the addition of optional add-on circuit boards. It may also be utilized directly in OEM applications by inserting custom programmed ROM or PROM memories into the sockets provided on the board. For these OEM applications, partially populated versions of the SDB-80 (designated OEM-80) are available without the standard system firmware, and with quantity discounts.



SYSTEM FIRMWARE

A standard feature of the SDB-80 is a complete package of development software aids which are resident in the five MK 34000, 2k x 8 ROM memories located on the board. This firmware includes a sophisticated operating system, debug package, assembler, and text editor. The presence of this software in ROM provides instant access to these development aids, eliminating the time-consuming requirement of loading the software from some perpheral device into RAM.
Another key feature of having the development aid software in ROM is that entire RAM space is available for the user's programs.

Debug (DDT-80) includes:

□ object program Load/Dump

□ Memory or Port Examine/Change

□ Breakpoint/Execute

□ Logical/Physical I/O mapping (with user expandable drivers)

□ Drivers for Standard Peripherals

SD/E Series

Z80 SYSTEM SUPPORT

SYS-80E	SDB-80E with MK 78039	
	4k byte	MK 78040
	16k byte	MK 78041
MOSTEK TERMINAL	Complete Video Display Unit	MK 78037

Z80 PROCESSOR ELEMENTS

OEM-80E	with 4k bytes RAM	MK 78122
	with 16k bytes RAM	MK 78124
DDT-80	Debug 2k Byte ROM	MK 78118
ASMB-80 EDIT-80	Assembler four 2k Byte ROM's (including the Editor)	MK 78119
SDB-80E	with OEM-80 + DDT-80 + ASMB-80 + EDIT-80 + documentation	
	4k byte RAM	MK 78103
	16k byte RAM	MK 78104

Z80 HARDWARE SUPPORT

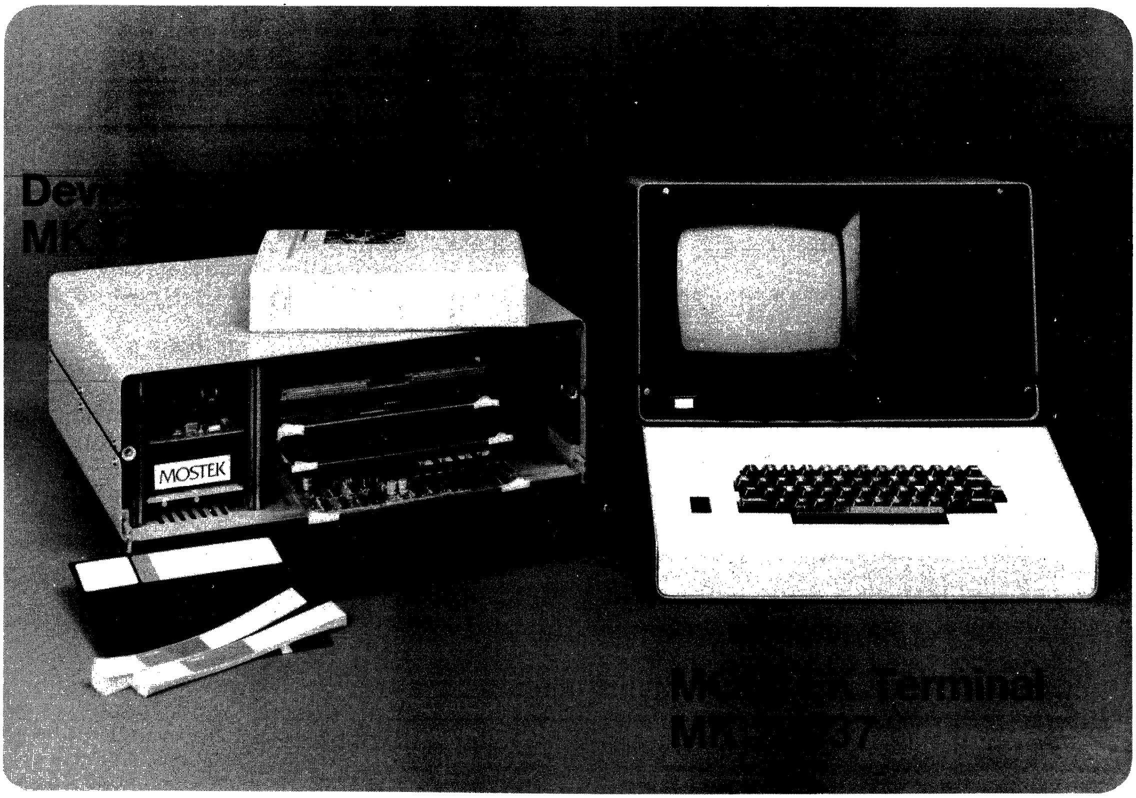
RAM-80AE	16k RAM	MK 78109
RAM-80BE	16k RAM, 2 PIO	MK 78110
XRAM-80	16k expander for RAM-80BE	MK 78126
AIM-80E	I.C.E. (In-Circuit-Emulation)	MK 78106
FLP-80E	Floppy Interface	MK 78112
RIO-80E	16k PROM, 2 PIO, 1-CTC, UART	MK 78128
Universal	Serial UDI-S	MK 78033
Display	Parallel UDI-P	MK 78035
Interface	Screen read option	MK 78036
	EIA Interface Cable For SDB-80E	MK 79058
	TTY Cable for SYS-80E	MK 79059
PPG-08	PROM Programmer for MK 2708 1kx8 UV PROM's with enclosure (requires MK 79060)	MK 79033
	PPG-08 Cable for SYS-80E	MK 79060
	Wire Wrap Card	MK 79063
	Extender Card	MK 79062
BACK-80E	Backplane Card, 6 slot	MK 79054
	Development Station Z80 6 total slots, power supply, no cards	MK 78039

Z80 SOFTWARE SUPPORT

XFOR-80	Fortran IV Cross Assembler requires 20k, 16 bit words	
	Card Deck	MK 78117C
	Paper Tape	MK 78117P
XMDS-80	8080 MDS Cross Assembler Paper Tape	MK 78115
XMDS-80D	8080 MDS Cross Assembler Soft sectored diskette	MK 78116
	Listing for DDT-80	MK 78534
	Listing for ASMB-80	MK 78536

DOCUMENTATION

Z80 CPU	Manual	MK 78070
Z80 PIO	Manual	MK 78071
SDB-80E	Manual	MK 78548
RAM-80E	Manual	MK 78545
AIM-80E	Manual	MK 78546
SDB-80E	Literature Package includes CPU, PIO, SDB-80 manuals plus data sheets	MK 78549
PPG-08	Manual	MK 78532
Z80	Pocket reference manual	MK 78516
Z80	Programming manual	MK 78515



MOSTEK TERMINAL MK 78037

FEATURES

- □ Self contained visual terminal
- □ 24 line, 80 character per line display
- □ Baud rate selection 110–9600
- □ Current loop or V24
- □ Comprehensive commands

GENERAL

A keyboard and a monitor provide together a "teletype" replacement video terminal for MOSTEK development systems, that can also be used in other applications. The terminal is completely self contained with its own power supply and electronics, requiring only the serial communication lines to the computer.

KEYBOARD

The keyboard obtains its power from the display unit, the coded keyboard information and power connections are made over a 25 pin type D connector.

DISPLAY ELECTRONICS

The display electronics uses the MOSTEK universal display interface board (MK 78033), power being provided within the terminal itself. The set of available functions is fully described in the MK 78033 data sheet; the key features are:

- □ 24 lines with 80 characters per line
- □ Cursor movements, absolute and relative
- □ Serial communication, 110–9600 baud
- □ Upper and lower case characters
- □ Clear screen, clear line etc.
- □ Tabulate
- □ A 9" diagonal display is used.

MECHANICAL

The display and keyboard are separate units connected by a cable. The display dimensions are:

B 43 cm H 26 cm T 32 cm

The keyboard dimensions are:

B 43 cm H 4,5 cm T 24 cm
H 9,0 cm

The Assembler (ASMB-80) includes:

□ 1, 2 or 3 pass operation

□ conditional Assembly

□ Relocatable object module generation

□ Relocatable linking loader

□ Drivers for Silent 700 Cassette

The Text Editor (EDIT-80) includes:

□ Line or character operation

□ Macro commands

ELECTRICAL SPECIFICATIONS

Operating Temperature Range . . . 0 °C to 50 °C

Power Supply requirements (Typical)

+12V ± 5%	175 mA
+ 5V ± 5%	1.5 A
−12V ± 5%	100 mA

Interface Levels . . . TTL Compatible

MECHANICAL SPECIFICATIONS

Extended double Eurocard

Board Size: 250 mm x 233.4 mm x 18 mm

Connector: Dual 64 pin Eurocard Connector DIN 41612 form D; A and C pinned.

COMPATIBLE ADD-ON BOARDS

RAM-80AE Add-on RAM card for the SDB-80E. This card supplies 16k bytes of MK 4027 dynamic RAM Memory.

RAM-80BE Add-on RAM/IO card for the SDB-80E. This card supplies 16k (expandable to 64k) bytes of MK 4116 dynamic RAM Memory, plus 4 fully buffered I/O ports using 2 MK 3881 PIO's. On-card bank switching allows expansion of SDB-80E memory space beyond 64k bytes.

XRAM-80 Expansion Kit for RAM-80BE. Consists of 8-MK 4116 RAMs.

AIM-80E In-circuit-emulation capability is added to the SDB-80 by using the AIM-80 board also provides other debugging capabilities such as TRACE and SINGLE STEP.

FLP-80E The FLP-80 interfaces the SDB-80 to two soft-sectored floppy disk drives. Full file handling software and firmware is provided with the card.

RIO-80E The RIO-80E includes 2-buffered PIO's, l-UART, l-CTC, and sockets for 16k bytes of MK 2708 PROM.

NON RESIDENT SOFTWARE AVAILABLE

XFOR-80 Fortran IV Cross Assembler. Assembles Z80 programs but is written in Fortran IV. It is useful for persons desiring to perform Z80 assembly in mini-computers such as the PDP-11. It is furnished in Fortran IV source as a card deck or paper tape.

XMDS-80 8080A Cross Assembler. Performs the same function as the Fortran IV Cross Assembler, except that it is designed to be used with an Intel MDS system. It is furnished as an object tape in Intel compatible Hex format.

XMDS-80D This is identical to the XMDS-80 except that it is compatible with Intel MDS systems which use floppy disk. It is furnished as object code on an MDS compatible floppy diskette.

OTHER ACCESSORIES AVAILABLE

PPG-08 PROM Programmer module for programming MK 2708 UV erasable PROM memories. Interfaces directly with the SDB-80. Enclosure included.

OEM USERS CARDS AVAILABLE

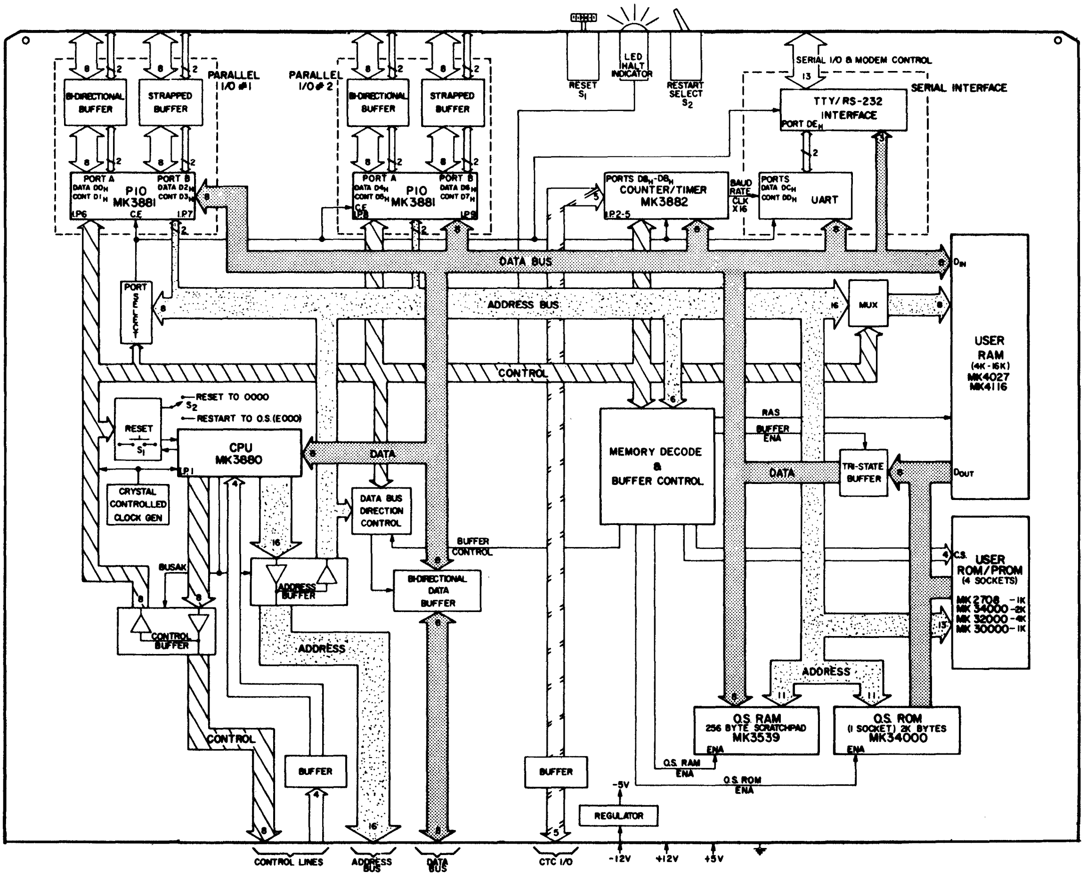
OEM-80E SDB-80E without Software. Available with 4 or 16k RAM Memory. 5 PROM/ROM sockets are free for user programs.

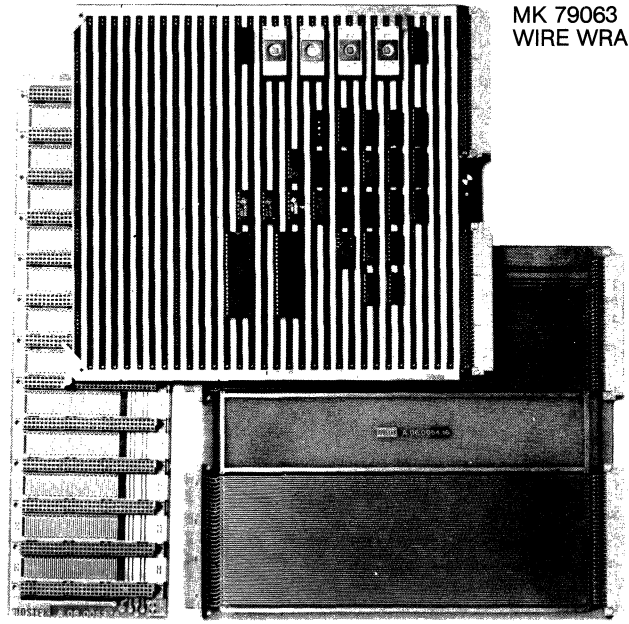
Parallel Universal Display Interface UDI-P This double Eurocard CRT/Keyboard Interface is bus compatible with the SDB-80E. A MK 3881 PIO on the card allows writing to the CRT Display at up to 3.300 characters per second.The CRT-80E provides 24 lines of 80 characters. The standard ASCII 96 character font is provided, other fonts may be programmed using MK 2708 PROM's. The command set includes TAB and cursor control.

Serial Universal Dislplay Interface UDI-S This is identical to the parallel except it operates over a 4 wire serial current loop connection. Useful in remote terminal applications.

BACK-80E 12 slot prewired printed circuit backplane for the SDB-80E family. This card greatly simplifies system construction.

SDB-80 FUNCTIONAL BLOCK DIAGRAM





MK 79063
WIRE WRAP CARD

EXTENDER CARD
MK 79062

BACKPLANE CARD
MK 79054

DEVELOPMENT STATION Z80 MK 78039

FEATURES

- □ Accepts upto 6 total boards
- □ Protected power supplies
- □ 11 I/O connectors for Peripheral equipment
- □ Double Europaformat boards

GENERAL

The MOSTEK development station has been designed to house and provide power for all MOSTEK boards with the double european format as in the detailed description of the SDB-80E. When used in conjunction with the MOSTEK terminal it forms a powerful development system.

POWER SUPPLIES

Plug-in power supplies are used, the supplies being one card with +5 volt 7 ampere and one card with ±12 volt 1 ampere. All supplies have overvoltage protection and current limiting.

MECHANICAL

The housing has the following dimensions:

B 52 cm H 18 cm T 36 cm

The front panel has quick release fastners to give free access to the boards.

INPUT/OUTPUT

11 connectors, 25 pin type D, are available for peripheral equipment. For the SDB 80E some of these are already commited as listed here below:

Connector	SDB 80 E function
1	Terminal
2	C T C
3	Floppy disc controller (1)
4	Uncommitted.
5	Uncommitted.
6	Paper tape reader
7	Paper tape punch
8	Line printer (2)
9	Uncommitted.
10	PROM programmer-PPG08
11	Uncommitted.

(1) with FLP 80E
(3) with RAM 80BE (or RIO80E or use PROM prog. Connector)

MOSTEK Z80 DEVELOPMENT SYSTEM

Includes:

MK 78103	SDB-80 Package A or	(4k byte RAM)
MK 78104	SDB-80 Package B with 256 byte static RAM DDT 80 Operating System ASMB 80 Resident Assembler and Text Editor and documentation	(16k byte RAM)
MK 78037	MOSTEK Terminal	
MK 78039	Development Station Z80	

PRELIMINARY

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Random Access Memory Board (RAM-80E)

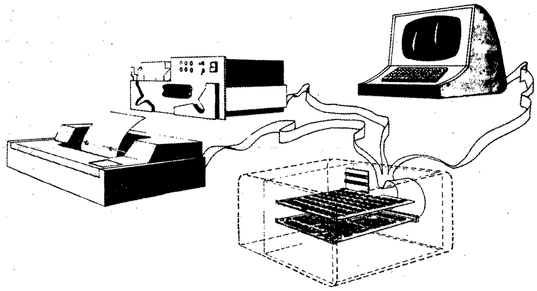
FEATURES

□ Memory Capacity
- RAM-80AE – 16,384 (16K) bytes using MK4027 RAM's
- RAM-80BE –16,384 (16K) bytes expandable to 65,536 (65K) bytes using MK4116 RAM's
- RAM-80BE under page mode operation – up to 1 megabyte of memory

□ I/O Capacity (RAM-80BE only)
Four 8-bit ports with handshake lines

□ Memory Access Time – 345ns (maximum)

□ Memory Cycle Time – 450ns (minimum)



GENERAL DESCRIPTION

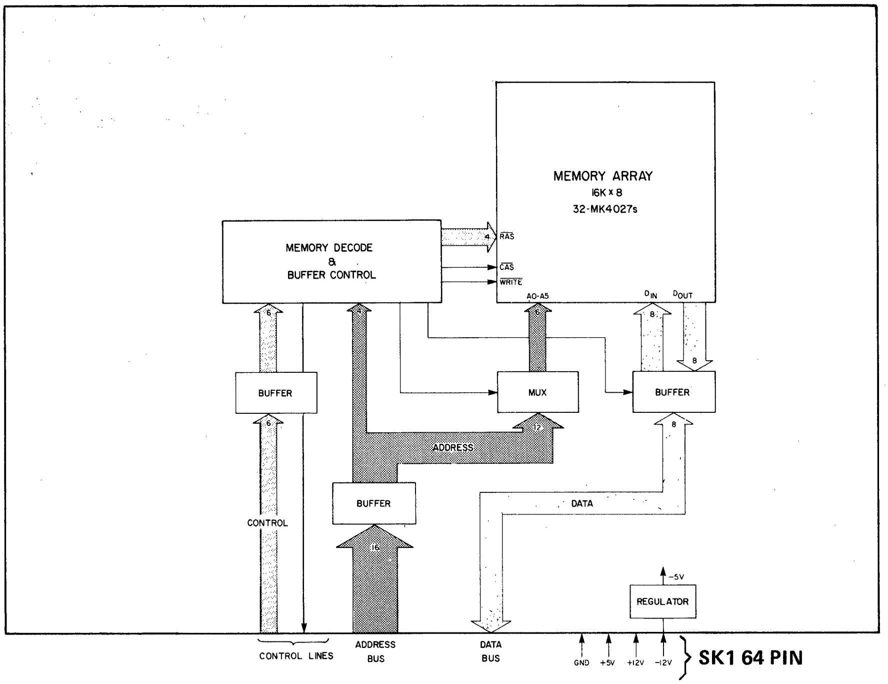
The RAM-80E is designed to provide RAM expansion capability for the Z80 based SDB-80E Microcomputer. For user flexibility, it is offered in two basic configurations designated RAM-80AE and RAM-80BE.

The RAM-80AE is the basic 16K byte RAM board for users requiring the most economical means for adding RAM to an SDB-80E Microcomputer. It is designed using the high performance MK4027-4, 4096 x 1 bit dynamic RAM, and includes address strapping options for positioning the decoded memory space to start on any 4K incremental address boundary.

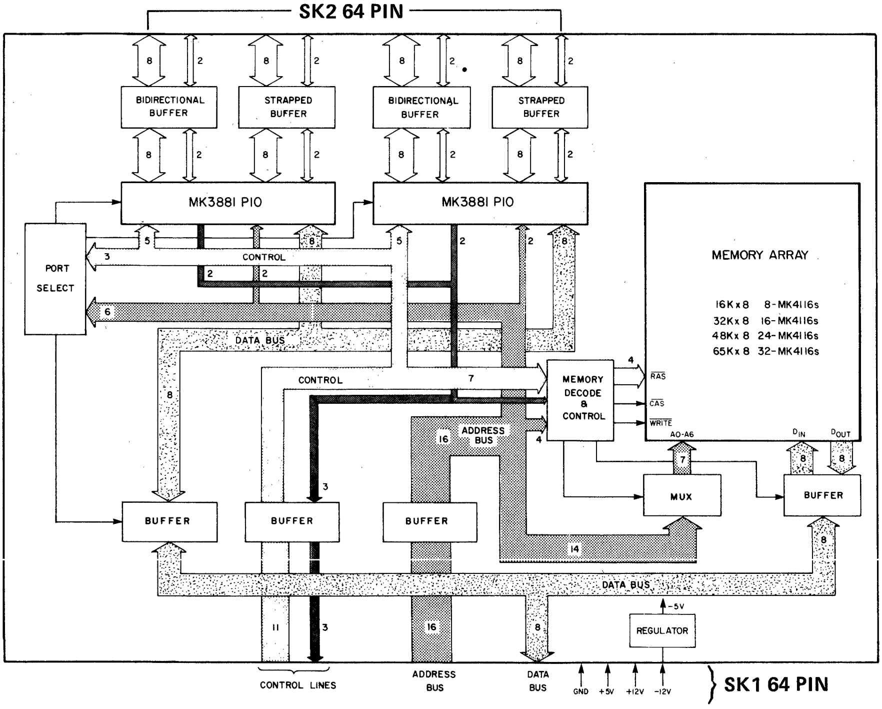
The RAM-80BE is a combination memory and I/O expansion board. The memory may be configured to have a memory capacity of l6K, 32K, 48K, or 65K bytes of RAM. This on-board memory expandability is made possible by population options of either eight, sixteen, twenty-four or thirty-two MK4116-4 (16,384 x 1 MOS dynamic RAM) memories. The RAM-80BE provides strapping options for positioning the decoded memory space to start on any 16K address boundary. In addition to the add-on memory, the RAM-80BE provides four 8-bit I/O ports from the two on-board MK3881 Z80 PIO circuits. Each I/O port is fully TTL buffered and has two handshake lines per I/O port. The RAM-80BE also includes logic for a "Page Mode Operation" which permits up to 1 megabyte of memory (sixteen 65K x 8 RAM-80B's) to be used in a single SDB-80E system.

A complete set of documentation for each RAM-80E board is available to ensure easy utilization.

RAM-80AE FUNCTIONAL DIAGRAM



RAM-80BE FUNCTIONAL DIAGRAM



ELECTRICAL SPECIFICATIONS

Memory Access Time—345 ns (maximum)

Memory Cycle Time—450 ns (minimum)

Operating Temperature: 0° C to 50° C

Power Supply Requirements

Voltage	RAM-80AE	RAM-80BE
+12V±5%	200 mA typ.	200 mA typ.
	575 mA max	575 mA max
−12V±5%	25 mA typ	25 mA typ
	30 mA max.	30 mA max.
+5V±5%	370 mA typ.	1.1A typ.
	550 mA max.	1.5A max.

MECHANICAL SPECIFICATIONS

U.S. Version

Board Size: 8.5" x 12.0" x 0.65"

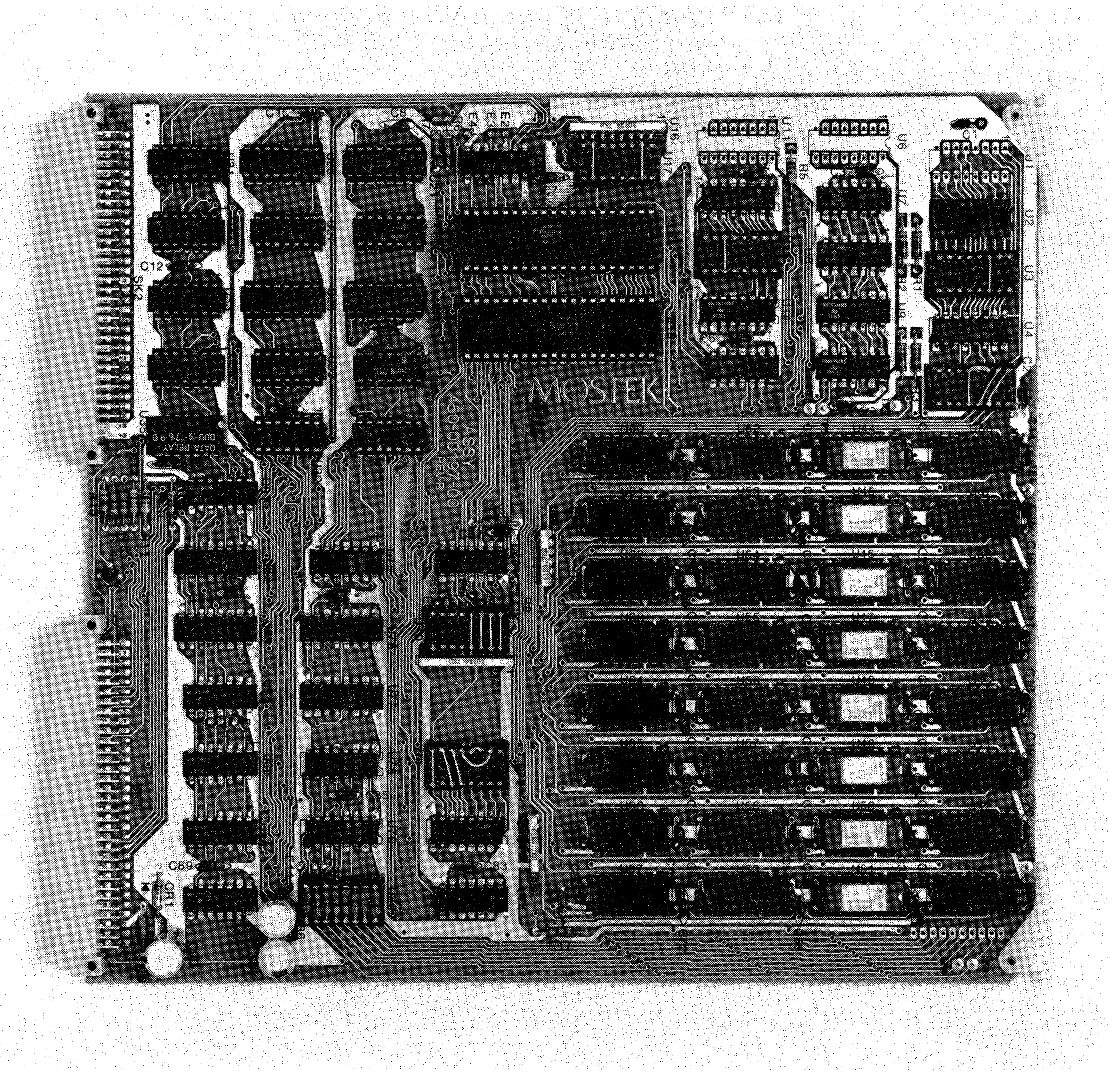
Bottom Connector: 100 pin, 125 mil centers

Top Parallel Connectors (RAM-80B): 50 pin, 100 mil centers

Double Eurocard Version

Board Size: 250 mm x 233.4 mm x 18 mm

Connector: Dual 64 pin Eurocard Connector



ORDERING INFORMATION

NAME	DESCRIPTION	PART NO.*
RAM-80AE	16,384 byte RAM board with 32-MK4027's.	MK78109
RAM-80BE	Expandable 16,384 byte RAM board with 8-MK4116's, sockets for additional MK4116's, 2-MK3881 Z80 PIO's, plus page mode capability.	MK78110
XRAM-80B	RAM-80B Expansion Package. Includes 8-MK4116 RAMs for insertion in the RAM-80B board plus a blank strapping header and documentation.	MK78126
Operations Manual	Complete description of the electrical specifications, timing, and circuit operation of the RAM-80BE. Also includes a detailed schematic diagram, assembly drawing and parts list.	MK78555
Operations Manual	Complete description of the electrical specifications, timing, and circuit operation of the RAM-80AE. Also includes a detailed schematic diagram, assembly drawing, and parts list.	MK78574

*Part numbers are for the Eurocard versions of RAM-80AE and RAM-80BE.

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Flexible Disk Drive Controller (FLP-80E)

HARDWARE FEATURES

- □ Soft sector format compatible with IBM 3740 data entry system format.
- □ Capable of controlling up to four flexible disk drives per subsystem.
- □ Full disk initialization (Formatting)
- □ Full sector (128 bytes) FIFO buffering for data.
- □ Double buffering for control and status.
- □ Automatic track seek with verification.
- □ Completely interruptable for real time systems.

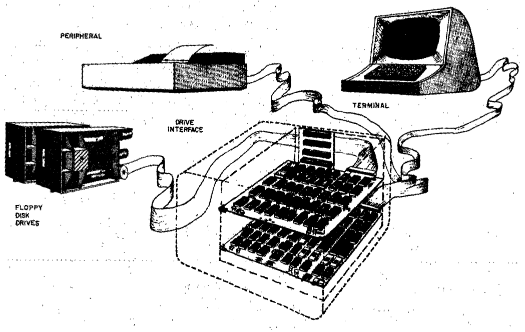
APPLICATIONS

- □ Flexible disk drive interface for use with MOSTEK's Software Development Board (SDB-80) in a disk based Z80 Development System (AID-80F)
- □ Single or multiple flexible disk drive controller/ formatter for disk based OEM systems using the OEM-80 Single Board Computer.

GENERAL DESCRIPTION

The FLP-80E is an add-on flexible disk controller used to interface up to four flexible disk drives to the MOSTEK Software Development Board (SDB-80E).

The FLP-80E provides the necessary electronics to accomplish track selection, head loading, data transfer, error detection, flexible drive interface, status reporting and format generation/recognition. The FLP-80E is designed to operate with either Shugart SA-800 Single Sided or SA-850 Double Sided Flexible Disk Drives. In addition to functioning as a add-on card to the SDB-80E system, the FLP-80 may be utilized directly in OEM applications to control/ format up to four flexible disk drives of either single or dual sided type in 8080A or Z80 systems.



AVAILABLE SOFTWARE

Software for the FLP-80E disk controller is the MOSTEK Disk Operating System (FLP-80DOS). A user can easily design his own OEM software package around 20 powerful disk operating system commands permitting complex record insertion, deletion, and position manipulation. Other software includes application packages such as an advanced monitor and debugger, disk-based Text Editor, Z80 Assembler, Relocating Linking Loader, Peripheral Interchange Program, and a channelized I/O system for each peripheral interface. These programs provide state-of-the-art software for developing Z80 programs as well as establishing a firm basis for OEM products. Further information is provided in the FLP-80DOS Data Sheet MK78556.

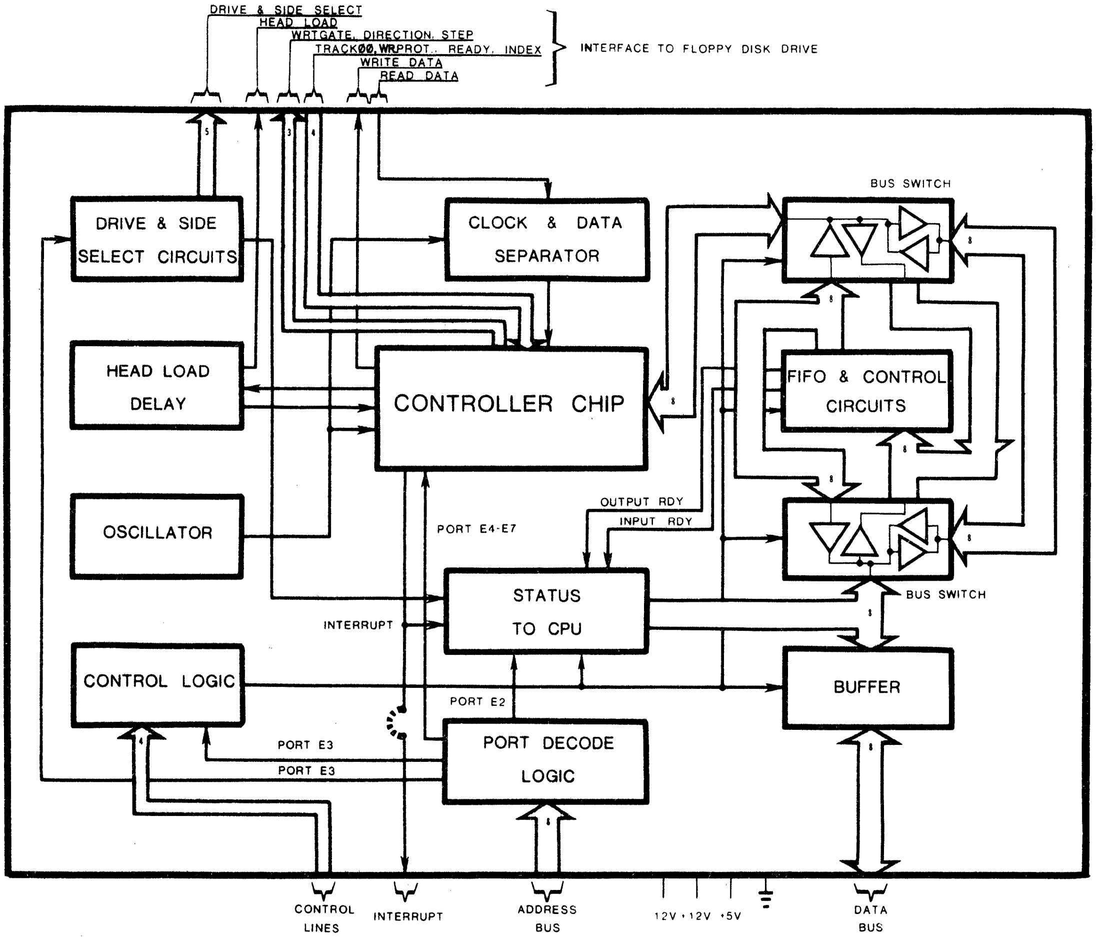
ELECTRICAL SPECIFICATIONS

Operating Temperature Range - 0° C to 50° C
Power Supply Requirements (Typical)
+12V ± 5% @ .006A
+5V ± 5% @ 1.1A
-12V ± 5% @ .03A
Interface Levels - TTL Compatible

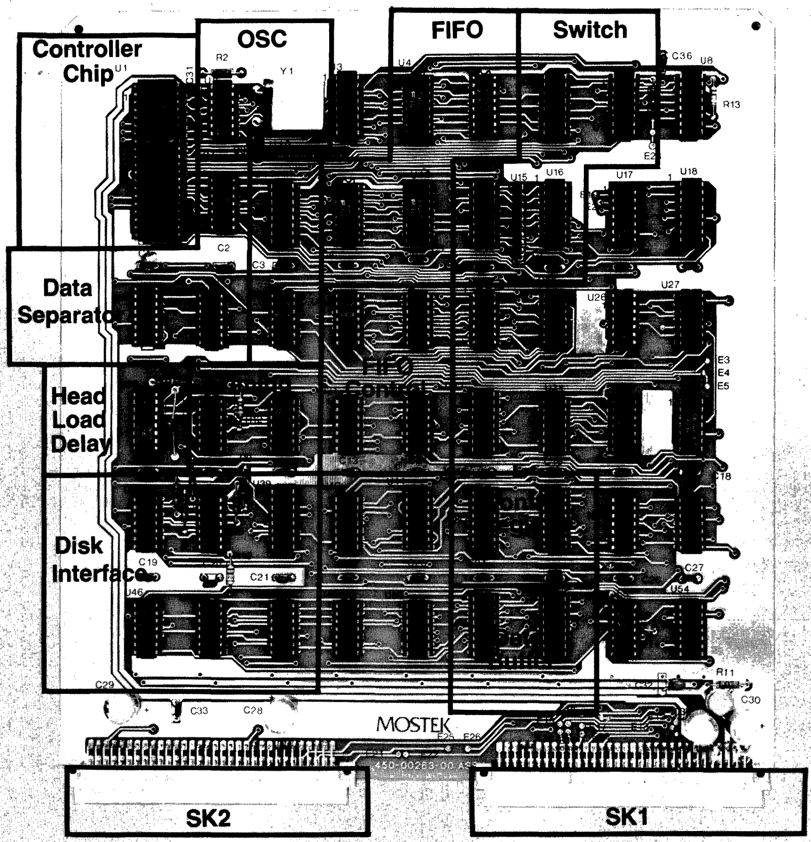
MECHANICAL EUROCARD

Board Size: 250mm x 233.4mm x 18mm
Bottom Connector: Dual 64 pin Eurocard Connector DIN 41612 form D; A and C pinned.

FLP-80E BLOCK DIAGRAM



FLP-80E BOARD



ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
FLP-80E	FLP-80E Disk Controller Board with Operations Manual, FLP-80DOS bootstrap, PROMs and diskette with FLP-80DOS Operations Manual.	MK78112
FLP-80E Operations Manual	Detailed description of use and operation of FLP-80E.	MK78561
FLP-80DOS Data Sheet	Disk Operating System data sheet.	MK78556
FLP-80DOS Operations Manual	Detailed description of the use and operation of FLP-80DOS.	MK78557

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

SYS-80F Flexible Disk Operating System (FLP-80DOS)

INTRODUCTION

The MOSTEK FLP-80DOS software package is designed for the MOSTEK dual floppy disk Z80 Development System (SYS-80F). Further information on this system can be found in the SYS-80F Data Sheet, MK78575. FLP-80 DOS is a software package that consists of two collections of programs: DSS-80, the Development System Software, and DOPS-80, the Disk Operating Software. FLP-80DOS includes:

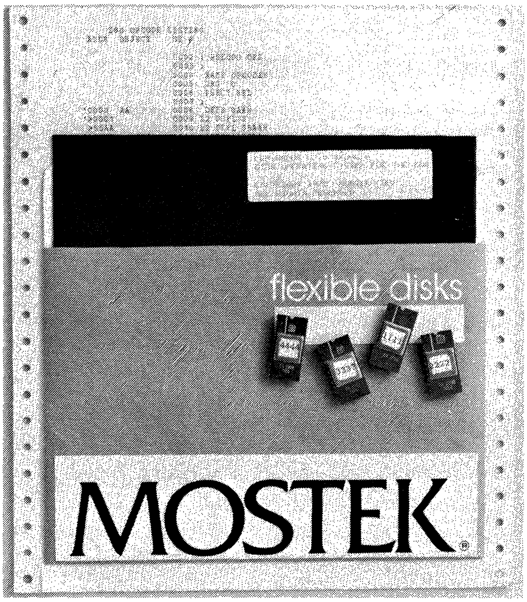
□ Monitor
□ Debugger
□ Text Editor
□ Z80 Assembler
□ Relocating Linking Loader
□ Peripheral Interchange Program
□ Linker
□ A Generalized I/O System For Peripherals

These programs provide state-of-the-art software for developing Z80 programs as well as establishing a firm basis for OEM products.

DEVELOPMENT SYSTEM SOFTWARE-DSS-80

Monitor

The Monitor provides user interface from the console to the rest of the software. The user can load and run system programs, such as the Assembler, using one simple command. Programs in object and binary format can be loaded into and dumped from RAM. All I/O is done via channels which are identified by Logical Unit Numbers. The Monitor allows any software device handler to be assigned to any Logical Unit Number. Thus, the software provides complete flexibility in configuring the system with different peripherals. The Monitor also allows two character mnemonics to represent 16-bit address values. Using mnemonics simplifies the command language. Certain mnemonics are reserved for I/O device handlers such as 'DK' for the flexible disk handler. The user can create and assign his own mnemonics at any time from the console, thus simplifying the command language for his own use. The Monitor also allows "batch mode operation" from any input device or disk file.



The Monitor commands are:

$ASSIGN - assign a Logical Unit Number to a device.
$CLEAR - remove the assignment of a Logical Unit Number to a device.
$RTABLE - print a list of current Logical Unit Number to device assignments.
$DTABLE - print default Logical Unit Number to device assignments.
$LOAD - load object modules into RAM.
$GTABLE - print a listing of global symbol table.
$GINIT - initialize global symbol table.
$DUMP - dump RAM to a device in object format.
$GET - load a binary file into RAM from disk.
$SAVE - save a binary file on disk.
$BEGIN - start execution of a loaded program.
$INIT initialize disk handler.
$DDT - enter DDT debug environment.
IMPLIED RUN COMMAND - get and start execution of a binary file.

Designer's Development Tool - DDT

The DDT debugger program is supplied in a combination of PROM and on the FLP-80DOS diskette. It provides a complete facility for interactively debugging relative and absolute Z80 programs. Standard commands allow displaying and modifying memory and CPU registers, setting breakpoints, and executing programs. Additional commands allow use of the MOSTEK AIM-80 to interactively debug a target system. Mnemonics are used to represent Z80 registers, thus simplifying the command language.

The allowed commands are:

B - Insert a breakpoint in user's program.
C - Copy contents of a block of memory to another location in memory.
E - Execute a program.
F - Fill an area of RAM with a constant.
H - 16-bit hexadecimal arithmetic.
L - Locate and print every occurrence of an 8-bit pattern.
M - Display, update, or tabulate the contents of memory.
P - Display or update the contents of a port.
R - Display the contents of the user's registers.
S - Hardware single step - requires MOSTEK's AIM-80 board.
W - Software single step.
V - Verify memory (compare two blocks and print differences).

Text Editor - EDIT

The FLP-80DOS Editor permits random access editing of ASCII character strings. The Editor works on blocks of characters which are rolled in from disk. It can be used as a line or character oriented editor. Individual characters may be located by positon or context. Each edited block is automatically rolled out to disk after editing. Although the Editor is used primarily for creating and modifying Z80 assembly language source statements, it may be applied to any ASCII text delimited by "carriage returns".

The Editor has a pseudo-macro command processing option. Up to two sets of commands may be stored and processed at any time during the editing process. The Editor allows the following commands:

An - Advance record pointer n records.
Bn - Backup record pointer n records.
Cn dS1dS2d - Change string S1 to string S2 for n occurrences.
Dn - Delete the next n records.
En - Exchange current records with records to be inserted.
Fn - If n = 0, reduce printout to console device (for TTY and slow consoles).
G - Get a file and insert it after the current line.
I - Insert records.
Ln - Go to line number n.
Mn - Enter commands into one of two alternate command buffers (pseudo-macro).
Pn - Put n records out to another file.
Q - Quit - Return to Monitor.
Sn dS1d - Search for nth occurrence of string S1.
T - Insert records at top of file before first record.
Vn - Output n records to console device.
Wn - Output n records to Logical Unit Number five (LUN 5) with line numbers.
Xn - Execute alternate command buffer n.

Z80 Assembler - ASM

The FLP-80DOS Assembler reads standard Z80 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The assembly listing shows address, machine code, statement number, and source statement. The code is in industry standard hexadecimal format modified for relocatable, linkable assemblies.

The Assembler supports conditional assemblies, global symbols, relocatable programs, and a printed symbol table. It can assemble any length program, limited only by a symbol table size of over 400 symbols. Expressions involving arithmetic and logical operations are allowed. Although normally used as a two pass assembler, the Assembler can also be run as a single pass assembler or as a learning tool. The following pseudo-ops are supported:

DEFB - define byte.
DEFL - set label.
DEFM - define message (ASCII).
DEFS - define storage.
DEFW - define word.
END - end statement.
ENDIF - end of conditional assembly.
EQU - equate label.
GLOBAL - global symbol definition.
IF - conditional assembly.
INCLUDE - include another file within an assembly.
NAME - program name definition.
ORG - program origin.
PSECT - program section definition.
EJECT - eject a page of listing.
TITLE - place heading at top of each page of listing.
LIST - turn listing on.
NLIST - turn listing off.

Relocating Linking Loader - RLL

The MOSTEK FLP-80DOS Relocating Linking Loader provides state of the art capability for loading programs into memory. Loading and linking of any number of relocatable or nonrelocatable object modules is done in one pass. A non-relocatable module is always loaded at its starting address as defined by the ORG pseudo-op during assembly. A relocatable object module can be positioned anywhere in memory at an offset address.

The Loader automatically links and relocates global symbols which are used to provide communication or linkage between program modules. As object modules are loaded, a table containing global symbol references and definitions is built up. The symbol table can be printed to list all global symbols and their load addresses. The number of object modules which can be loaded by the Loader is limited only by the amount of RAM available for the modules and the symbol table.

The Loader also loads industry standard non-relocatable, non-linkable object modules.

Linker - LINK

The Linker provides capability for linking object modules together and creating a binary (RAM image) file on disk. A binary file can be loaded using the Monitor GET or IMPLIED RUN command. Modules are linked together using global symbols for communication between modules. The linker produces a global symbol table and a global cross reference table which may be listed on any output device.

The Linker also provides a library search option for all global symbols undefined after the specified object modules are processed. If a symbol is undefined the Linker searches the disk for an object file having the file-name of the symbol. If the file is found, it is linked with the main module in an attempt to resolve the undefined symbol.

Peripheral Interchange Program - PIP

The Peripheral Interchange Program provides complete file maintenance facilities for the system. In addition, it can be used to copy information from any device or file to any other device or file. The command language is easy to use and resembles that used on DEC minicomputers. The following commands are supported:

COMMAND	FUNCTION
APPEND	Append files.
COPY	Copy files from any device to another device or file.
DIRECT	List directory of specified disk unit.
ERASE	Delete a file.
FORMAT	Format a disk.
INIT	Initialize the disk handler.
RENAME	Rename a file.
STATUS	List number of used and available sectors on specified disk unit.
QUIT	Return to Monitor.

The first letter only of each command may be used.

DISK OPERATING SOFTWARE - DOPS-80

The disk software, as well as being the heart of the AID-80F development system, can be used directly in OEM applications. The software consists of two programs which provide a complete disk handling facility.

Input/Output Control System - IOCS

The first package is called the I/O Control System (IOCS). This is a generalized blocker/deblocker which can interface to any device handler. Input and output can be done via the IOCS in any of four modes:

1. single byte transfer.
2. line at a time, where the end of a line is defined by carriage return.
3. multibyte transfers, where the number of bytes to be transferred is defined as the logical record length.
4. continuous transfer to end-of-file, which is used for binary (RAM-image)files.

The IOCS provides easy application of I/O oriented packages to any device. There is one entry point, and all parameters are passed via a vector defined by the calling program. Any given handler defines the physical attributes of its device which are, in turn, used by the IOCS to perform blocking and deblocking.

Floppy Disk Handler - FDH

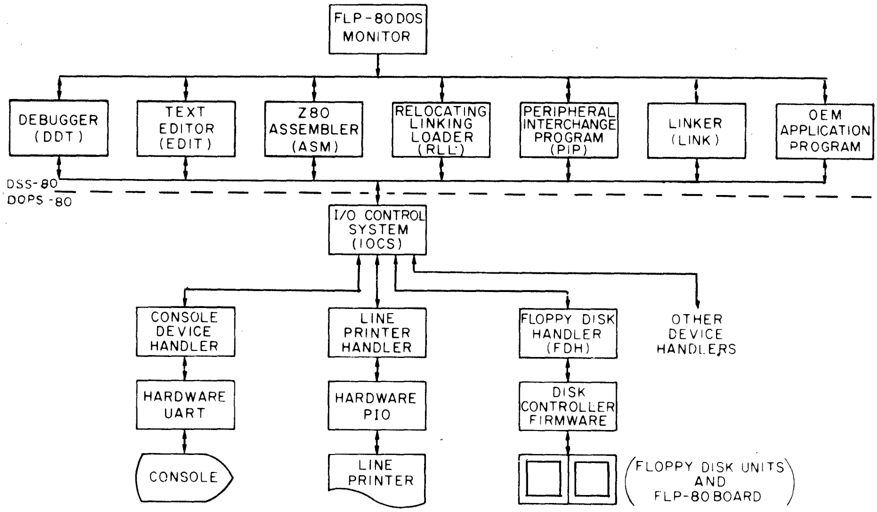
The Floppy Disk Handler (FDH) interfaces from the IOCS to a firmware controller for up to 4 floppy disk units. The FDH provides a sophisticated command structure to handle advanced OEM products. The firmware controller interfaces to MOSTEK's FLP-80E Controller Board. The disk format is IBM 3740 soft sectored. The software can be easily adapted to double-sided disks.The Floppy Disk Handler commands include:

— erase file
— create file
— open file
— close file
— rename file
— rewind file
— read next n sectors
— reread current sector
— read previous sector
— skip forward n sectors
— skip backward n sectors
— replace (rewrite) current sector
— delete n sectors

The FDH has advanced error recovery capability. It supports a bad sector map and an extensive directory which allows multiple users. The file structure is doubly linked to increase data integrity on the disk and a bad file can be recovered from either its start or end.

SD/E Series

FLP-80DOS FLOW CHART



ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
DSS-80	DSS-80 Development System Software. This package is supplied to OEM's under a non-exclusive licensing agreement. The software is supplied on paper tape as source, on Mostek diskette as a source, relocatable object, and binary, and in the form of complete listings. FLP-80DOS Operations Manual is included.	MK78135
DOPS-80	DOPS-80 Disk Operating Software. The Software is supplied on paper tape as source, on Mostek Diskette as source, relocatable object, and binary, and in the form of complete listings. FLP-80DOS Operations Manual is included.	MK78136
SYS-80F	AID-80F Z80 Development System. Includes all FLP-80DOS programs in binary and DSS-80 source listings described above.	MK78134
FLP-80EDOS Operation Manual	Detailed description of the operation and use of FLP-80DOS.	MK78557
SYS-80F Data Sheet	Z80 Development System Data Sheet.	MK78575
FLP-80E Data Sheet	Flexible Disk Drive Controller Data Sheet.	MK78572

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Software Development Board (SDB-80)

HARDWARE FEATURES

□ Available with choice of either 4K or 16K bytes of RAM

□ Four 8-bit I/O ports with handshake lines

□ Serial ASCII interface (110-9600 BAUD)

□ Fully buffered for system expandability

□ Four counter/timer channels

□ On board capacity for 5K bytes of PROM or 20K bytes of ROM

SOFTWARE FEATURES

□ 2K x 8 Operating System in ROM (DDT-80)

□ 8K x 8 assembler/editor in ROM (ASMB-80)

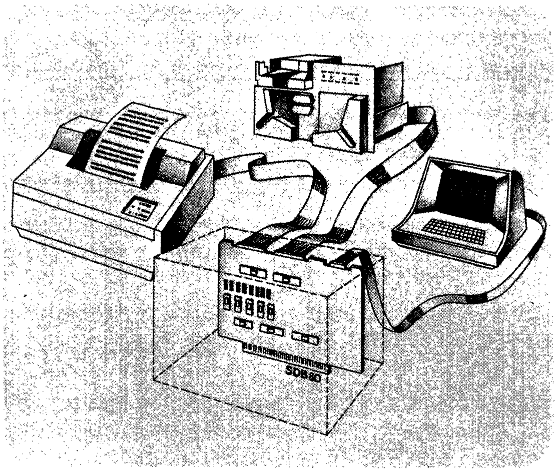
□ Channeled I/O for user convenience

GENERAL DESCRIPTION

The SDB-80 is a stand-alone microcomputer designed by MOSTEK around the advanced Z80 microprocessor family. It contains more on-board firmware and RAM memory than any previously offered single board microcomputer, plus all the features of the industries' most sophisticated microprocessor. This board represents the very latest in state-of-the-art technology by utilizing MOSTEK's new 16K Dynamic RAM memories. The SDB-80 also is the first single board microcomputer to offer a complete package of software development aids in ROM. This 10K byte firmware package is included with the SDB-80 and provides the ability to generate, edit, assemble, load, execute, and debug Z80 programs for all types of applications.

USING THE SDB-80

In addition to functioning as a stand-alone development aid, the SDB-80 is fully expandable through the addition of optional add-on circuit boards. It may also be utilized directly in OEM applications by inserting custom programmed ROM or PROM memories into the sockets provided on the board. For these OEM applications, partially populated versions of the SDB-80 (designated OEM-80) are available without the standard system firmware, and with quantity discounts.



SYSTEM FIRMWARE

A standard feature of the SDB-80 is a complete package of development software aids which are resident in the five MK34000, 2K x 8 ROM memories located on the board. This firmware includes a sophisticated operating system, debug package, assembler, and text editor. Among the many features provided are execute and breakpoint commands, console routines for examining and/or modifying memory and port locations, object load capability for both absolute and relocatable object modules, I/O driver routines for a variety of standard peripheral devices, and channeled I/O for user defined peripheral drivers. The presence of this software in ROM provides instant access to these development aids, eliminating the time-consuming requirement of loading the software from some peripheral device into RAM. Another key feature of having the development aid software in ROM is that the entire RAM space is available for the user's programs.

ELECTRICAL SPECIFICATIONS

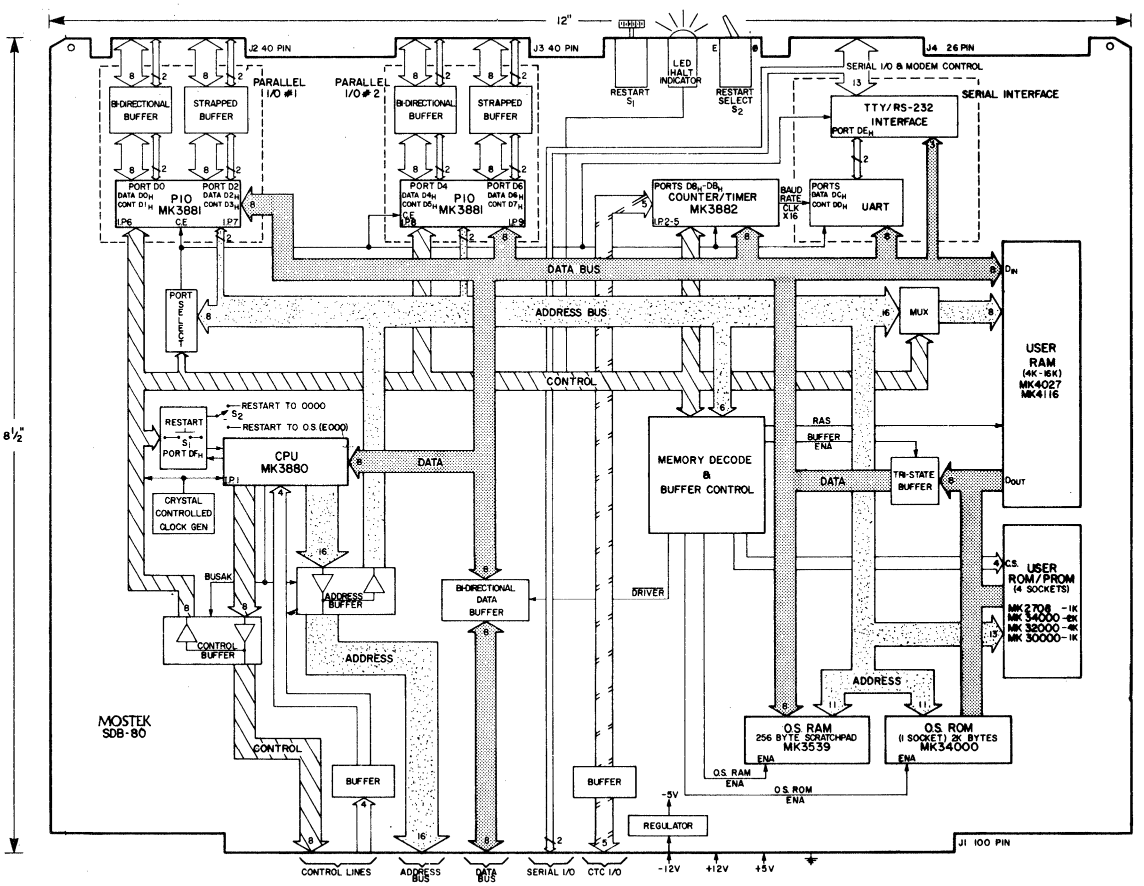
Operating Temperature Range . . .0°C to 50°C

Power Supply Requirements (Typical)

+12V ± 5% @ 175 mA
+ 5V ± 5% @ 1.5 A
−12V ± 5% @ 100 mA

Interface Levels . . . TTL Compatible

SDB-80 FUNCTIONAL BLOCK DIAGRAM



NON-RESIDENT SOFTWARE AVAILABLE

XFOR-80 Fortran IV Cross Assembler. Assembles Z80 programs but is written in Fortran IV. It is useful for persons desiring to perform Z80 assembly on mini-computers such as the PDP-11. It is furnished as a Fortran IV source deck. (MK78117)

XMDS-80 8080A Cross Assembler. Performs the same function as the Fortran IV Cross Assembler, except that it is designed to be used with an Intel MDS system. It is furnished as an object tape in Intel compatible Hex format. (MK78115)

XMDS-80D This is identical to the XMDS-80 except that it is compatible with Intel MDS systems which use floppy disks. It is furnished as object code on an MDS compatible floppy diskette. (MK78116)

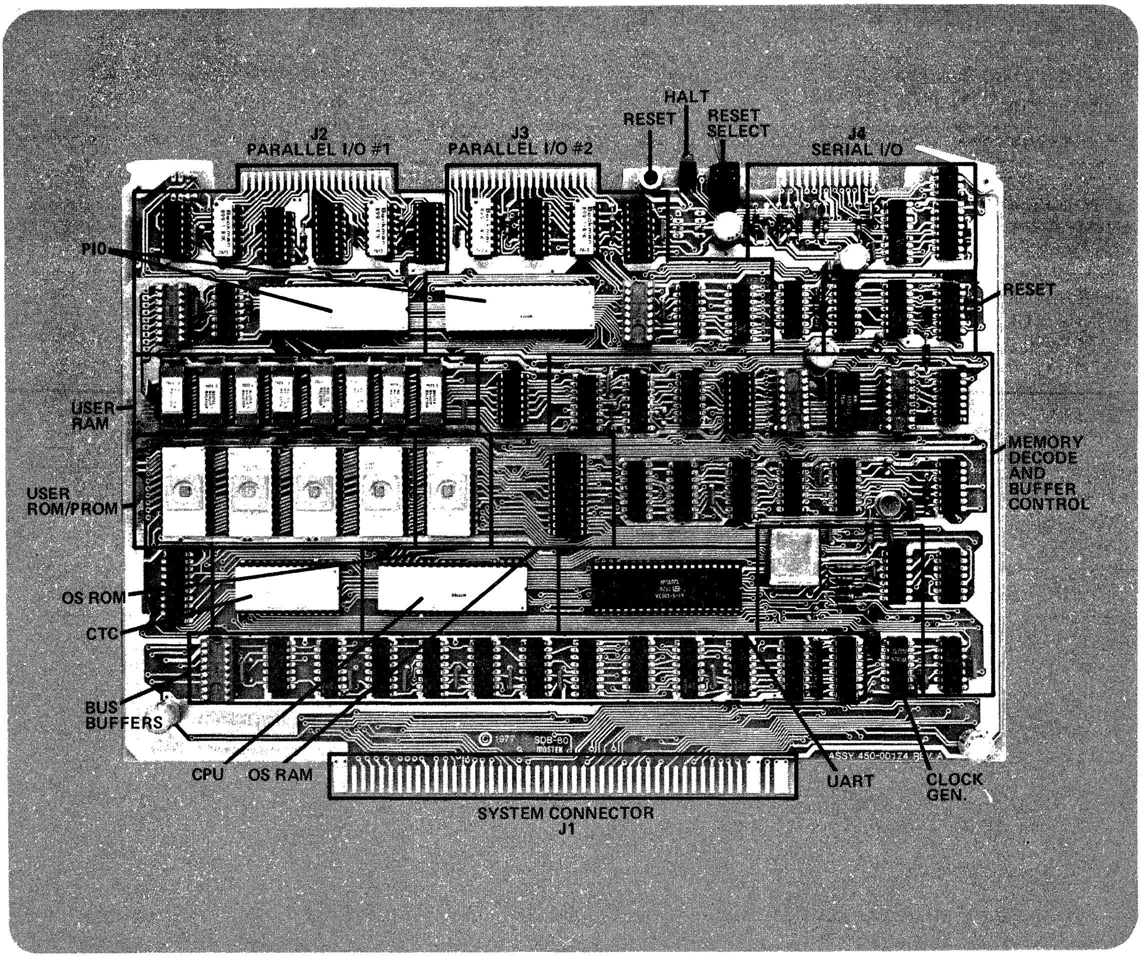
COMPATIBLE ADD-ON BOARDS

RAM-80A A 16K byte RAM board for users requiring the most economical means for expanding memory. (MK78107)

RAM-80B Combination memory and I/O expansion board. The memory may be configured to have a capacity of 16K, 32K, 48K, or 65K bytes of RAM. The board also provides four 8-bit I/O ports from two Z80 PIO circuits. (MK78108)

AIM-80 In-circuit emulation capability is added to the SDB-80 by using the AIM-80 board (Application Interface Module). This board also provides other debugging capabilities such as TRACE and SINGLE STEP, and a DISASSEMBLER. (MK78132)

AIM-72 Provides in-circuit emulation capability for emulating the MK3870 family of single chip microcomputers. Compatible with SDB-80 only in AID-80F floppy disk system environment.



FLP-80 The FLP-80 interfaces the SDB-80 to two floppy disk drives. Software drivers are included with the board. (MK78111)

OTHER ACCESSORIES AVAILABLE

PPG-08 PROM Programmer module for programming MK 2708 UV Erasable PROM memories. Interfaces directly with the SDB-80. (MK79033)

XAID-100 System package which includes a 13-slot card cage, enclosure and power supply. (MK79034)

AID-80F Complete Z80 Microcomputer system which includes enclosure with 6 slot card cage, power supply, cooling fan, OEM-80, RAM-80B, FLP-80, two Floppy Disk drives, and FLP-80DOS software package. (MK78125)

XAID-102 Three-slot card cage. (MK79028)

XAID-103 Wire wrap card (MK79023)

XAID-104 Extender card (MK79024)

MECHANICAL SPECIFICATIONS

Domestic Version

Board Size: 8.5" x 12.0" x 0.65"
Bottom Connector: 100 pin, 125 mil centers
Top Serial Connector: 26 pin, 100 mil centers
Top Parallel Connectors: 40 pin, 100 mil centers

Double Eurocard Version Available

Board Size: 250mm x 233.4mm x 18mm
Connector: Dual 64 pin Eurocard Conn.

ORDER INFORMATION FOR THE SDB-80 AND OEM-80

Z80 Microcomputer System Components		SDB-80 Package 'A' MK78101 includes:	SDB-80 Package 'B' MK78102 includes:
DESCRIPTION	PART NO.		
OEM-80* with 256 bytes of static RAM, 4K bytes of dynamic RAM, and sockets for ROM and PROM.	78121	X	
OEM-80* with 256 bytes of static RAM, 16K bytes of dynamic RAM and sockets for ROM and PROM.	78123		X
DDT-80 operating system in 1-MK34000 2K x 8 ROM.	78118	X	X
ASMB-80 Resident Assembler and Text Editor in 4 MK34000 2K x 8 ROMs.	78119	X	X
TTY Interface Cable (XAID-800).	79036	X	X
EIA/RS-232 Interface Cable (XAID-802).	79038	X	X

* The Circuit Board for the SDB-80 and the OEM-80 are identical and include 2 MK3881 PIOs, 1 MK3882 CTC, and 1 UART, plus the associated circuitry for control and buffering of all bus and I/O signals. Sockets are provided for expansion of on-board system RAM and ROM/PROM.

SYSTEM DATA SHEETS

MK78519	RAM-80A/B
MK78537	AIM-80
MK78538	FLP-80
MK79576	AIM-72

MK79081	PPG-8/16
MK78568	AID-80F
MK79081	PPG-8/16
MK79552	XAID-103
MK79552	XAID-104

PRELIMINARY

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Random Access Memory Board (RAM-80)

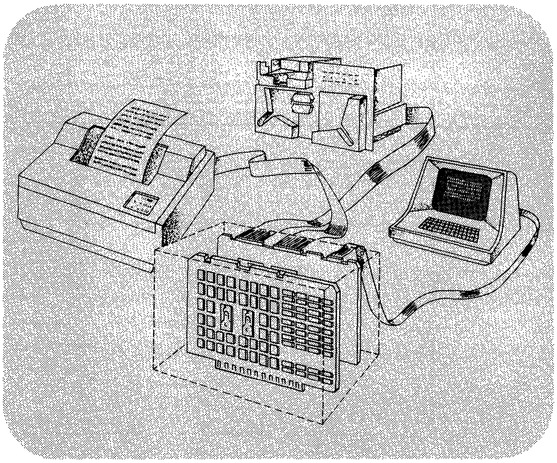
FEATURES

□ Memory Capacity
- RAM-80A – 16,384 (16K) bytes using MK4027 RAM's
- RAM-80B–16,384 (16K) bytes expandable to 65,536 (65K) bytes using MK4116 RAM's
- RAM-80B's under page mode operation – up to 1 megabyte of memory

□ I/O Capacity (RAM-80B only)
Four 8-bit ports with handshake lines

□ Memory Access Time – 345ns (maximum)

□ Memory Cycle Time – 450ns (minimum)



GENERAL DESCRIPTION

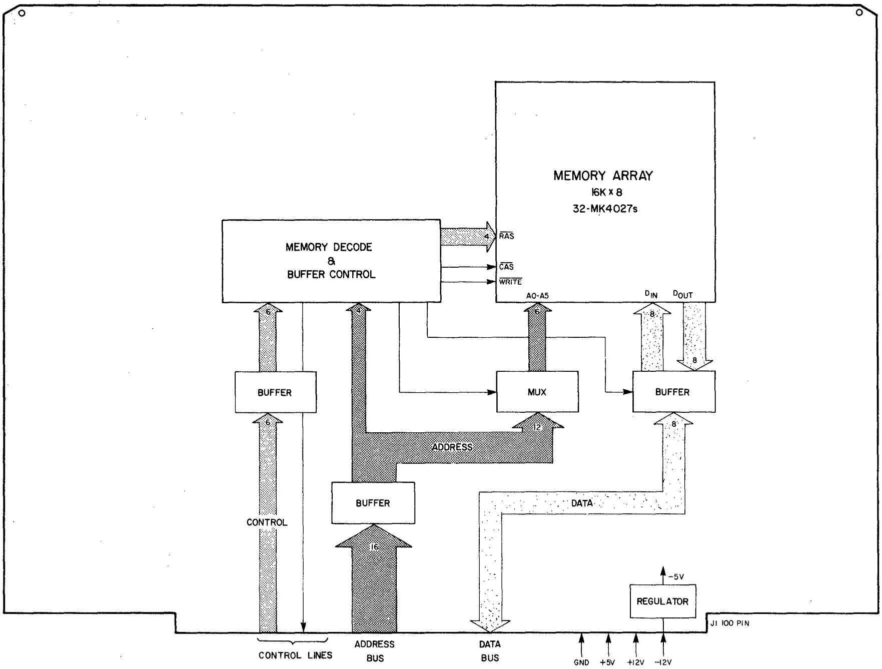
The RAM-80 is designed to provide RAM expansion capability for the Z80 based SDB-80 Microcomputer. For user flexibility, it is offered in two basic configurations designated RAM-80A and RAM-80B.

The RAM-80A is the basic 16K byte RAM board for users requiring the most economical means for adding RAM to an SDB-80 Microcomputer. It is designed using the high performance MK4027-4, 4096 x 1 bit dynamic RAM, and includes address strapping options for positioning the decoded memory space to start on any 4K incremental address boundary.

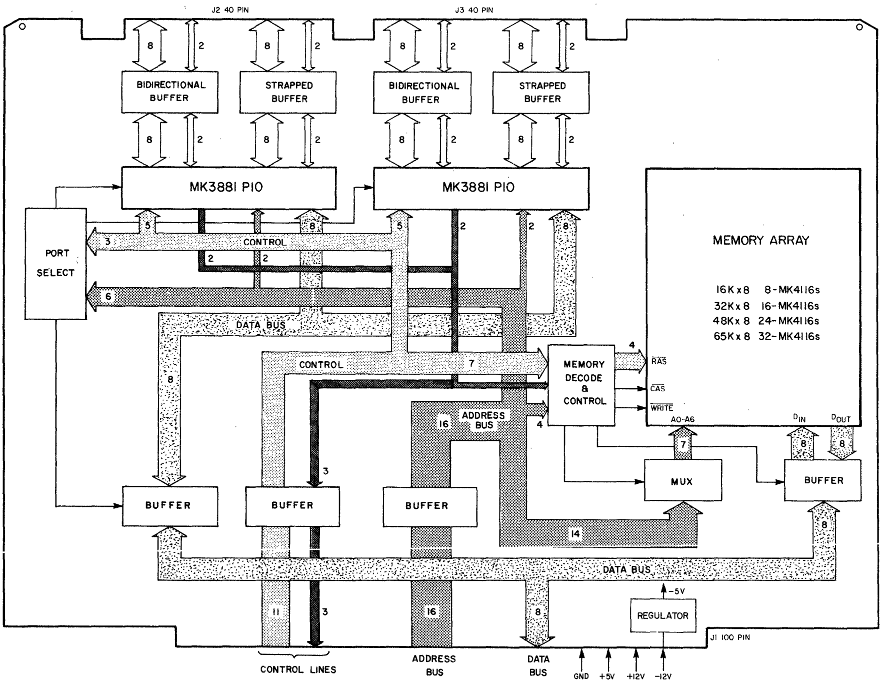
The RAM-80B is a combination memory and I/O expansion board. The memory may be configured to have a memory capacity of I6K, 32K, 48K, or 65K bytes of RAM. This on-board memory expandability is made possible by population options of either eight, sixteen, twenty-four or thirty-two MK4116-4 (16,384 x 1 MOS dynamic RAM) memories. The RAM-80B provides strapping options for positioning the decoded memory space to start on any 16K address boundary. In addition to the add-on memory, the RAM-80B provides four 8-bit I/O ports from the two on-board MK3881 Z80 PIO circuits. Each I/O port is fully TTL buffered and has two handshake lines per I/O port. The RAM-80B also includes logic for a "Page Mode Operation" which permits up to 1 megabyte of memory (sixteen 65K x 8 RAM-80B's) to be used in a single SDB-80 system.

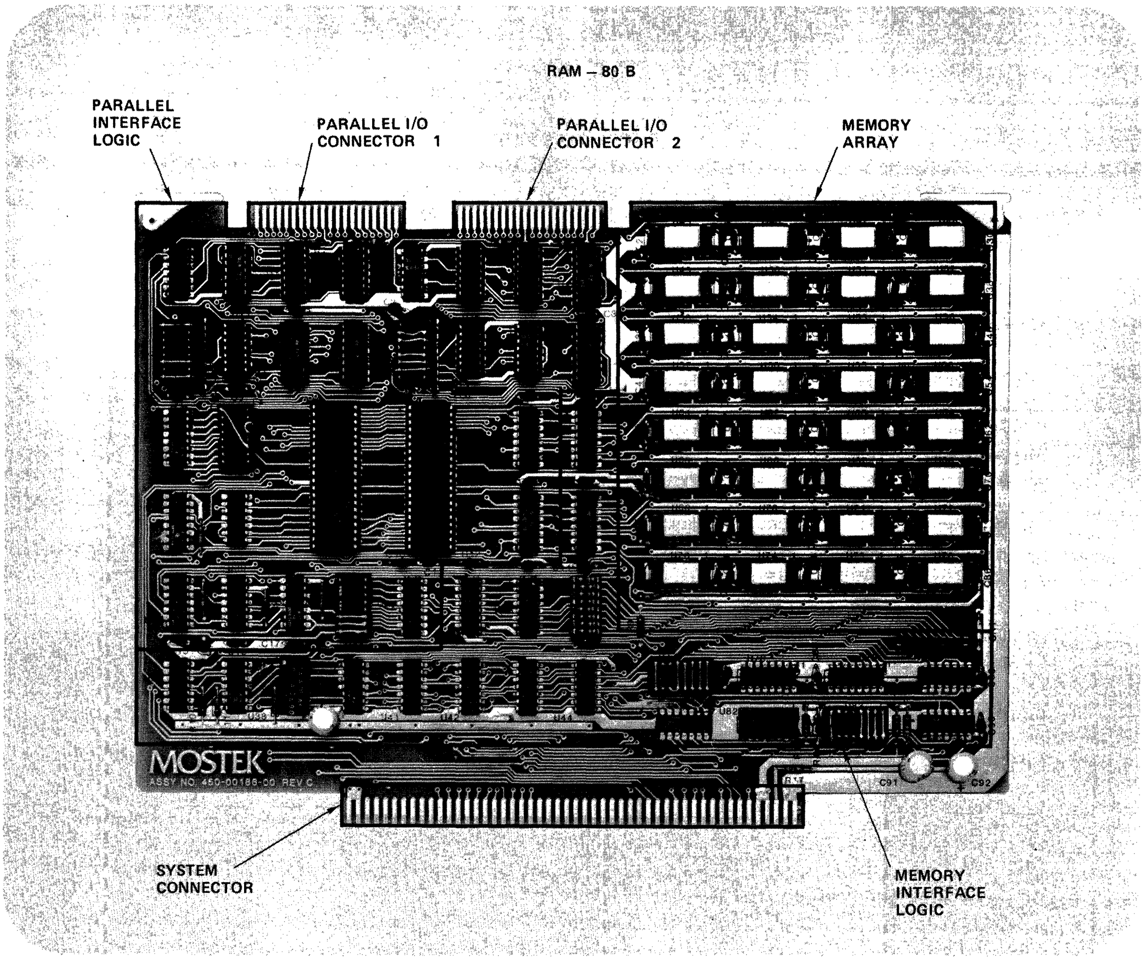
A complete set of documentation for the RAM-80 is available to ensure easy utilization of the board.

RAM-80A FUNCTIONAL DIAGRAM



RAM-80B FUNCTIONAL DIAGRAM





ELECTRICAL SPECIFICATIONS

Memory Access Time—345 ns (maximum)
Memory Cycle Time—450 ns (minimum)
Operating Temperature: 0°C to 50°C
Power Supply Requirements

Voltage	RAM-80A	RAM-80B
+12V±5%	200 mA typ. 575 mA max	200 mA typ. 575 mA max
−12V±5%	25 mA typ 30 mA max.	25 mA typ 30 mA max.
+5V±5%	370 mA typ. 550 mA max.	1.1A typ. 1.5A max.

MECHANICAL SPECIFICATIONS

U.S. Version

Board Size: 8.5″ x 12.0″ x 0.65″
Bottom Connector: 100 pin, 125 mil centers
Top Parallel Connectors (RAM-80B): 40 pin, 100 mil centers

Double Eurocard Version

Board Size: 250 mm x 233.4 mm x 18 mm
Connector: Dual 64 pin Eurocard Connector

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold
on condition that the Purchaser agrees to
the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.

4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject To This Agreement:

Order Number Description Price*

____

____

Ship To: ______________________ Bill To: ______________

____

Method of Shipment: ______________________ Customer P.O. Number: ______________

Agreed To:

PURCHASER MOSTEK CORPORATION

By: ______________________ By: ______________
Title: ______________________ Title: ______________
Date: ______________________ Date: ______________

*Prices Subject to change Without Notice

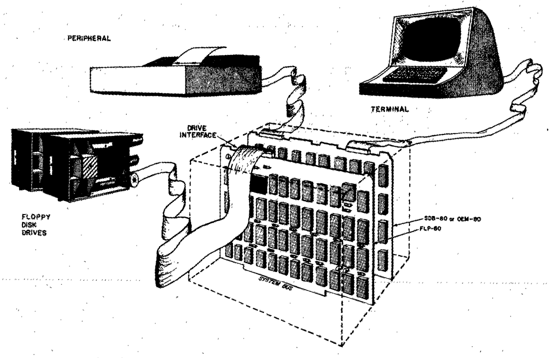
MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Flexible Disk Drive Controller (FLP-80)

HARDWARE FEATURES

- □ Soft sector format compatible with IBM 3740 data entry system format.
- □ Capable of controlling up to four flexible disk drives per subsystem.
- □ Double sided drive capability.
- □ Full disk initialization (Formatting).
- □ Full sector (128 bytes) FIFO buffering for data.
- □ Double buffering for control and status.
- □ Automatic track seek with verification.
- □ Completely interruptable for real time systems.



APPLICATIONS

- □ Flexible disk drive interface for use with MOSTEK's Software Development Board (SDB-80) in a disk based Z80 Development System (AID-80F).
- □ Single or multiple flexible disk drive controller/formatter for disk based OEM systems using the OEM-80 Single Board Computer.

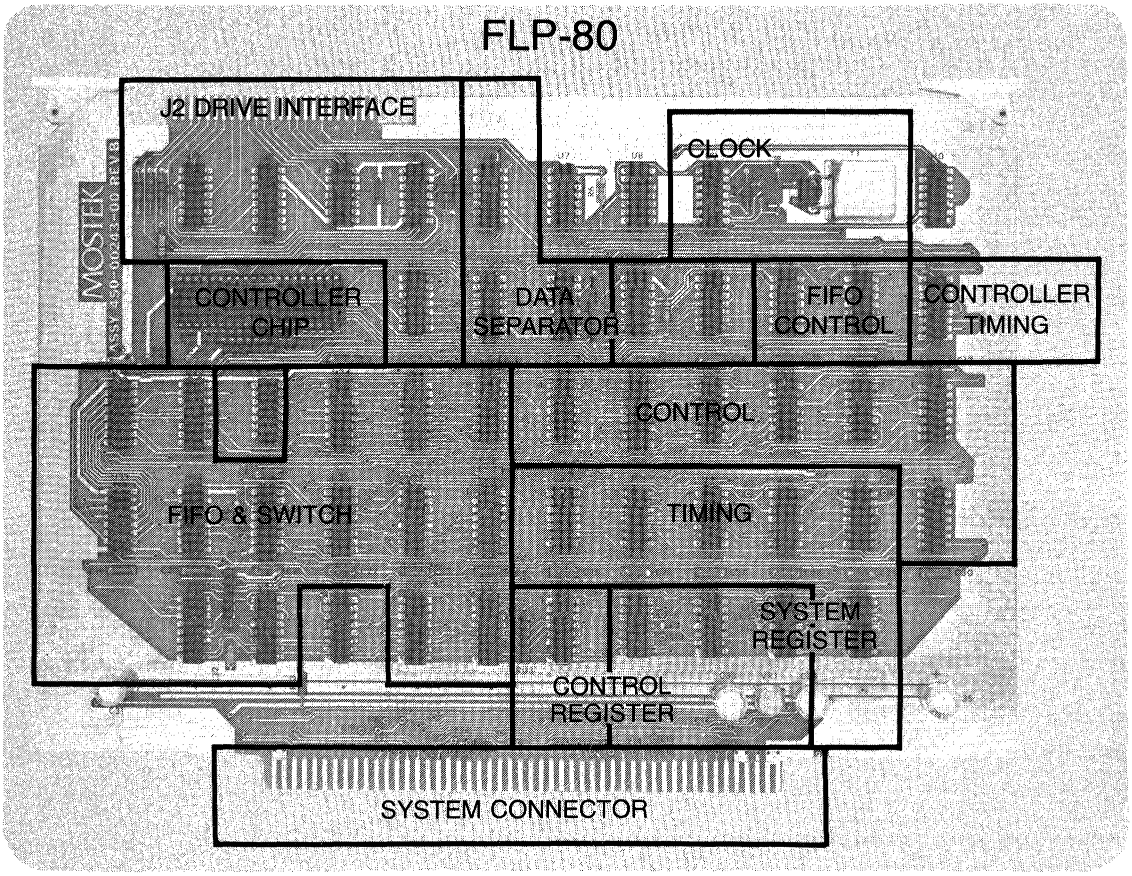
GENERAL DESCRIPTION

The FLP-80 is an add-on flexible disk controller used to interface up to four flexible disk drives to the MOSTEK Software Development Board (SDB-80).

The FLP-80 provides the necessary electronics to accomplish track selection, head loading, data transfer, error detection, flexible drive interface, status reporting and format generation/recognition. The FLP-80 is designed to operate with either Shugart SA-800 Single Sided or SA-850 Double Sided Flexible Disk Drives. In addition to functioning as an add-on card to the SDB-80 system, the FLP-80 may be utilized directly in OEM applications to control/format up to four flexible disk drives of either single or dual sided type in 8080A or Z80 systems.

AVAILABLE SOFTWARE

Software for the FLP-80 Disk controller is the MOSTEK Disk Operating System (FLP-80DOS). A user can easily design his own OEM software package around 20 powerful disk operating system commands permitting complex record insertion, deletion, and position manipulation. Other software includes application packages such as an advanced monitor and debugger, disk-based Text Editor, Z80 Assembler, Relocating Linking Loader, Peripheral Interchange Program, and a channelized I/O system for each peripheral interface. These programs provide state-of-the-art software for developing Z80 programs as well as establishing a firm basis for OEM products. Further information is provided in the FLP-80DOS Data Sheet MK78556.



ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
FLP-80	FLP-80 Disk Controller Board with Operations Manual, FLP-80DOS bootstrap PROMs and diskette, and FLP-80 DOS Operations Manual.	MK78111
FLP-80 Operations Manual	Detailed description of the use and operation of FLP-80.	MK78560
FLP-80DOS Operations Manual	Detailed description of the use and operation of FLP-80DOS.	MK78557
FLP-80DOS Data Sheet	Disk operating system data sheet	MK78556
AID-80F Data Sheet	Disk system data sheet	MK78568

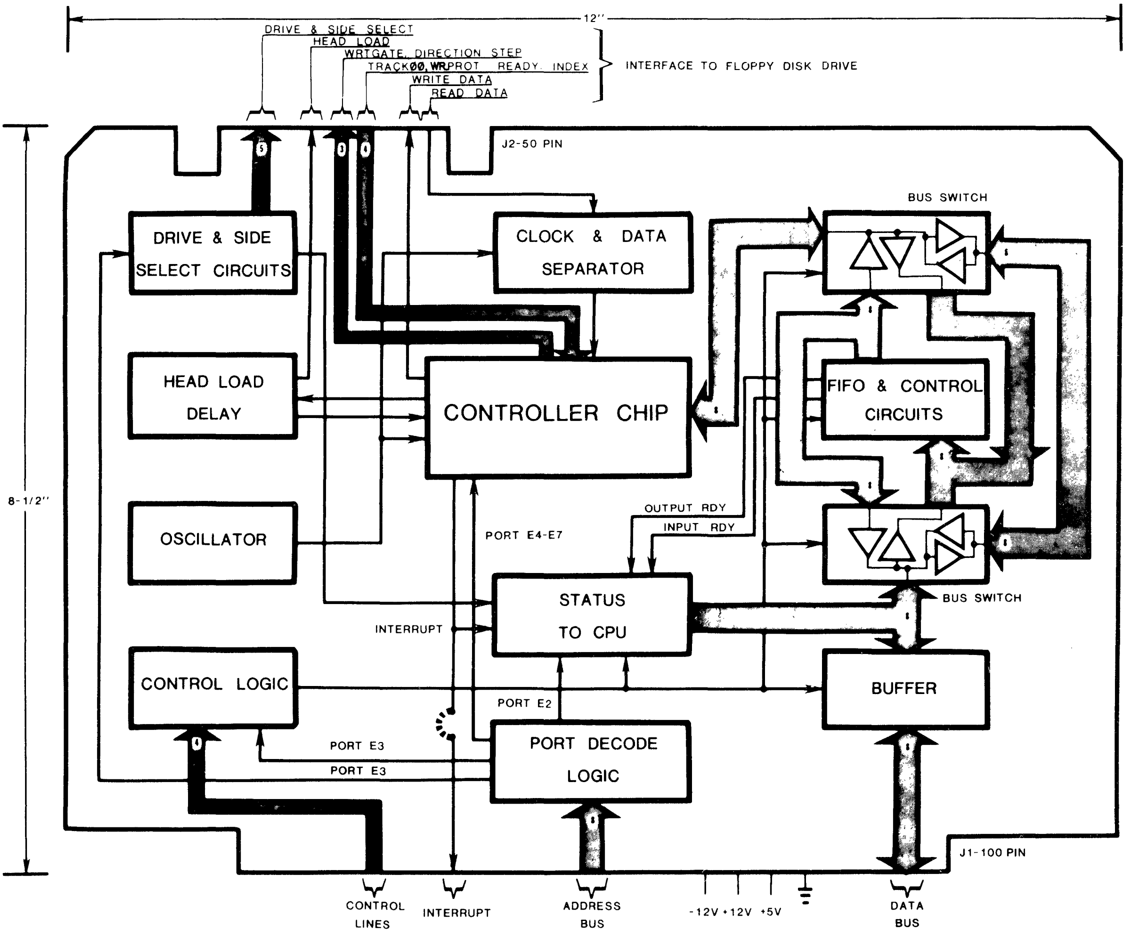
MECHANICAL SPECIFICATIONS

Board Size: 8.5" x 12.0" x 0.65"
*Bottom Connector: 100 pin 125 mil centers
*Top Connector: 50 pin 100 mil centers
*Supplied with Bottom Card edge connector and Top Ribbon Cable Assembly for two drives.

ELECTRICAL SPECIFICATIONS

Operating Temperature Range - 0° C to 50° C
Power Supply Requirements (Typical)
+12V ± 5% @ .006A
+5V ± 5% @ 1.1A
−12V ± 5% @ 0.03A
Interface Levels — TTL Compatible

FLP-80 BLOCK DIAGRAM



ORDER INFORMATION

NAME	DESCRIPTION	PART NO.
RAM-80A	16,384 byte RAM board with 32-MK4027's.	MK 78107
RAM-80B	Expandable 16,384 byte RAM board with 8-MK4116's, sockets for additional MK4116's, 2-MK3881 Z80 PIO's, plus page mode capability.	MK 78108
XRAM-80B	RAM-80B Expansion Package. Includes 8-MK4116 RAMs for insertion in the RAM-80B board plus a blank strapping header and documentation.	MK 78126
Operations Manual	Complete description of the electrical specifications, timing, and circuit operation of the RAM-80. Also includes a detailed schematic diagram, assembly drawing, and parts list.	MK 78545

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Flexible Disk Operating System (FLP-80DOS)

INTRODUCTION

The Mostek FLP-80DOS software package is designed for the MOSTEK Z80 dual floppy disk systems. FLP-80DOS is a software package that consists of two collections of programs: DSS-80, the Development System Software, and DOPS-80, the Disk Operating Software. FLP-80DOS includes:

Monitor
Debugger
Text Editor
Z80 Assembler
Peripheral Interchange Program
Linker
A Generalized I/O System For Peripherals

These programs provide state-of-the-art software for developing Z80 programs as well as establishing a firm basis for OEM products.

DEVELOPMENT SYSTEM SOFTWARE-DSS-80

Monitor

The Monitor provides communication with the user via the console terminal enabling him to load and start execution of either system (e.g., PIP, EDITOR, ASM, LINKER) or user programs. In addition, the Monitor provides utility functions such as re-assignment of logical unit devices and the creation of RAM image files. After power up or reset, the system automatically enters the Monitor environment awaiting entry of user commands.

The Monitor commands are:

$ASSIGN - assign a Logical Unit Number to a device.
$CLEAR - remove the assignment of a Logical Unit Number to device.
$RTABLE - print a list of current Logical Unit Number to device assignments.
$DTABLE - print default Logical Unit Number to device assignments.
$DUMP - dump RAM to a device in object format.
$GET - load a binary file into RAM from disk.
$SAVE - save a binary file on disk.
$BEGIN - start execution of a loaded program.
$INIT - initialize disk handler.
$DDT - enter DDT debug environment.
IMPLIED RUN COMMAND - get and start execution of a binary file. The file name of the system or user program to be executed is entered as the Monitor command.

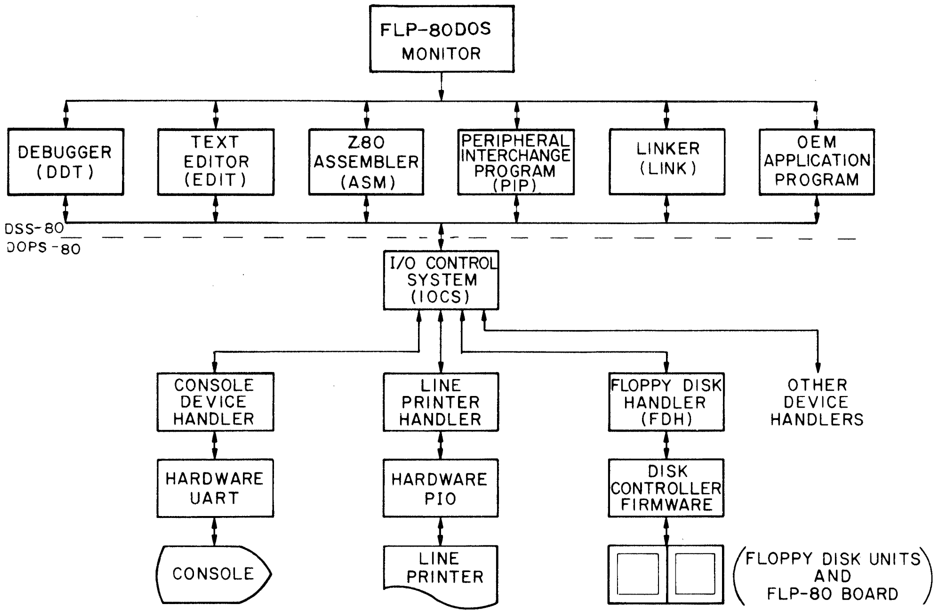
Designer's Development Tool - DDT

The DDT debugger program is supplied in PROM. It provides a complete facility for interactively debugging relative and absolute Z80 programs. Standard commands allow displaying and modifying memory and CPU registers, setting breakpoints, and executing programs. Additional commands allow use of the MOSTEK AIM-80 to interactively debug a target system. Mnemonics are used to represent Z80 registers, thus simplifying the command language.

The allowed commands are:

B - Insert a breakpoint in user's program.
C - Copy contents of a block of memory to another location in memory.
E - Execute a program.
F - Fill an area of RAM with a constant.
H - 16-bit hexadecimal arithmetic.
L - Locate and print every occurrence of an 8-bit pattern.
M - Display, update, or tabulate the contents of memory.
P - Display or update the contents of a port.
R - Display the contents of the user's registers.
S - Hardware single step - requires MOSTEK's AIM-80 board.
W - Software single step.
V - Verify memory (compare two blocks and print differences).

SOFTWARE BLOCK DIAGRAM



Text Editor - EDIT

The FLP-80DOS Editor permits random access editing of ASCII character strings. The Editor works on blocks of characters which are rolled in from disk. It can be used as a line of character-oriented editor. Individual characters may be located by position or context. Each edited block is automatically rolled out to disk after editing. Although the Editor is used primarily for creating and modifying Z80 assembly language source statements, it may be applied to any ASCII text delimited by "carriage returns".

The Editor has a pseudo-macro command processing option. Up to two sets of commands may be stored and processed at any time during the editing process. The Editor allows the following commands:

An	- Advance record pointer n records.
Bn	- Backup record pointer n records.
Cn dS1dS2d	- Change string S1 to string S2 for n occurrences.
Dn	- Delete the next n records.
En	- Exchange n records with records to be inserted.
Fn	- If n = 0, reduce printout to console device (for TTY and slow consoles).
G	- Get a file and insert it after the current line.
I	- Insert records.
Ln	- Go to line number n.
Mn	- Enter commands into one of two alternate command buffers (pseudo-macro).
Pn	- Put n records out to another file.
Q	- Quit - Return to Monitor.
Sn dS1d	- Search for nth occurrence of string S1.
T	- Insert records at top of file before first record.
Vn	- Output n records to console device.
Wn	- Output n records to Logical Unit Number five (LUN 5) with line numbers.
Xn	- Execute alternate command buffer n.

Z80 Assembler - ASM

The FLP-80DOS Assembler reads standard Z80 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The assembly listing shows address, machine code, statement number, and source statement. The code is in industry-standard, hexadecimal format modified for relocatable, linkable assemblies.

The Assembler supports conditional assemblies, global symbols, relocatable programs, and a printed symbol table. It can assemble any length program, limited only by a symbol table size which is dependent on available RAM. Expressions involving arithmetic and logical operations are allowed. Although normally used as a two-pass assembler, the Assembler can also be run as a single-pass assembler. The following pseudo-ops are supported:

DEFL	- set label.
DEFM	- define message (ASCII).
DEFS	- define storage.
DEFW	- define word.
END	- end statement.
ENDIF	- end of conditional assembly.
EQU	- equate label.
GLOBAL	- global symbol definition.
IF	- conditional assembly.
INCLUDE	- include another file within an assembly.
NAME	- program name definition.
ORG	- program origin.
PSECT	- program section definition.
EJECT	- eject a page of listing.
TITLE	- place heading at top of each page of listing.
LIST	- turn listing on.
NLIST	- turn listing off.

Linker - LINK

The Linker provides capability for linking object modules together and creating a binary (RAM image) file on disk. A binary file can be loaded using the Monitor GET or IMPLIED RUN command. Modules are linked together using global symbols for communication between modules. The linker produces a global symbol table and a global cross-reference table which may be listed on any output device.

The Linker also provides a library search option for all global symbols undefined after the specified object modules are processed. If a symbol is undefined the Linker searches the disk for an object file having the file name of the symbol. If the file is found, it is linked with the main module in an attempt to resolve the undefined symbol.

Peripheral Interchange Program - PIP

The Peripheral Interchange Program provides complete file maintenance facilities for the system. In addition, it can be used to copy information from any device or file to any other device or file. The command language is easy to use and resembles that used on DEC minicomputers. The following commands are supported:

COMMAND	FUNCTION
APPEND	Append files
COPY	Copy files from any device to another device or file.
DIRECT	List directory of specified disk unit.
ERASE	Delete a file.
FORMAT	Format a disk.
INIT	Initialize the disk handler.
RENAME	Rename a file.
STATUS	List number of used and available sectors on specified disk unit.
QUIT	Return to Monitor

The first letter only of each command may be used.

DISK OPERATING SOFTWARE - DOPS-80

The disk operating software as well as being the heart of MOSTEK disk systems, can be used directly in OEM applications. The software consists of two programs which provide a complete disk-handling facility.

Input/Output Control System - IOCS

The first package is called the I/O Control System (IOCS). This is a generalized blocker/deblocker which can interface to any device handler. Input and output can be done via the IOCS in any of four modes:

1. single byte transfer.
2. line at a time, where the end of a line is defined by carriage return.
3. multibyte transfers, where the number of bytes to be transferred is defined as the logical record length.
4. continuous transfer to end-of-file, which is used for binary (RAM-image) files.

The IOCS provides easy application of I/O oriented packages to any device. There is only one entry point, and all parameters are passed via a vector defined by the calling program. Any given handler defines the physical attributes of its device which are, in turn, used by the IOCS to perform blocking and deblocking.

Floppy Disk Handler - FDH

The Floppy Disk Handler (FDH) interfaces from the IOCS to a firmware controller for up to 4 floppy disk units. The FDH provides a sophisticated command structure to handle advanced OEM products. The firmware controller interfaces to MOSTEK's FLP-80 Controller Board. A firmware controller is also available for MOSTEK's MDX-FLP Controller Board. The disk format is IBM 3740 type, soft-sectored. The software can be easily adapted to double-sided disks. The Floppy Disk Handler commands include:

- erase file
- create file
- open file
- close file
- rename file
- rewind file
- read next n sectors
- reread current sector
- read previous sector
- skip forward n sectors
- skip backward n sectors
- replace (rewrite) current sector
- delete n sectors

The FDH has advanced error recovery capability. It supports a bad sector map and an extensive directory which allows multiple users. The file structure is doubly linked to increase data integrity on the disk and a bad file can be recovered from either its start or end.

HARDWARE CONFIGURATIONS

FLP-80DOS software is offered for three different hardware configurations. The minimum hardware configurations required are described in the following.

SD SERIES

FLP-80DOS software is available in the MK78111 package which includes the FLP-80 Disk Controller Module. The minimum system configuration is the following:

a). OEM-80/16 (MK78123) - SD Series single board computer with 16K bytes of RAM.

b). FLP-80 (MK78111) - SD Series Disk controller board which includes the FLP-80DOS software. Supports 8 inch, soft-sectored floppy disks only. Recommended drives are Shugart SA 800-2 or SA850 (double-sided).

c). RAM-80 A or B (MK78107 or MK78108) - Optional RAM expansion modules

SD/E SERIES

FLP-80DOS Software is available in the MK78112 package which includes the FLP-80E disk controller modules. The /E modules are identical to the SD Series in function but are on a double Eurocard form factor. The minimum system configuration is the following.

a). OEM-80/16E (MK78124) - SD/E Series Single Board Computer with 16K bytes of RAM.

b). FLP-80E (MK78112) - SD Series discontroller board which includes the FLP-80DOS software. Supports 8 inch, soft-sectored floppy disks only. Recommended drives are Shugart SA 800-2 or SA 850 (double-sided).

c). RAM-80 AE or RAM80BE (MK78109 or MK78110) - Optional RAM expansion modules

d). MDX-DRAM (16K Bytes MK77751) (32K Bytes MK77752) - MD Series RAM modules. FLP-80DOS/MDX requires at least 16K bytes of RAM

MDX SERIES

FLP-80DOS/MDX software is available for the MDX Series of OEM Microcomputer Modules. The minimum system configuration is the following.

a). FLP-80 DOS/MDX (MK77962) - FLP-80DOS/MDX bootstrap PROMs and diskette containing all FLP-80DOS/MDX programs in Binary and FLP-80DOS Operations Manual.

b). MDX-CPU1 (MK77850) - MD Series CPU Board. Has PROM sockets for above mentioned bootstrap PROMs.

c). MDX-EPROM/UART (MK77753) - MD Series Serial interface module.

d). MDX-FLP (MK77652) - MD Series disk controller module for single and double-sided drives (Both 5" and 8" Drives). Recommended drives are Shugart SA 800-2, SA 850 (double-sided), SA 400 and SA 450 (mini double-sided).

SD Series

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
FLP-80	FLP-80 Disk Controller Board with Operations Manual, FLP-80DOS Bootstrap PROMs and diskette containing all FLP-80DOS programs in binary, and FLP-80DOS Operation Manual. Requires at least 16K bytes of RAM and OEM-80.	MK78111
FLP-80E	Same as above but on double Eurocard form factor	MK78112
FLP-80DOS/ MDX	FLP-80DOS/MDX Bootstrap PROMs and diskette containing all FLP-80DOS/ MDX programs in binary and FLP-80DOS/ Operations Manual. Requires MDX-CPU1, MDX-EPROM/UART, MDX-FLP modules and MDX-DRAM module with at least 16K bytes of RAM.	MK77962
FLP-80DOS Operation Manual	Detailed description of the operation and use of FLP-80DOS and FLP-80DOS/MDX software packages.	MK78557
DSS-80	DSS-80 Development System Software available to OEM's under a non-exclusive licensing agreement. The software is supplied on paper tape as source, on Mostek diskette as source, relocatable object, and binary, and in the form of complete listings. FLP-80DOS Operations Manual is included.	MK78135
DOPS-80	DOPS-80 Disk Operating Software is supplied on paper tape as source, on paper tape as source, on Mostek diskette as source, relocatable object, and binary, and in the form of complete listings. FLP-80DOS Operations Manual is included.	MK78136
MOSTEK BASIC	BASIC Interpreter high-level language to run on FLP-80DOS or FLP-80DOS/MDX. Requires 32K bytes of RAM.	MK78157
MOSTEK FORTRAN	FORTRAN IV high-level language compiler to run on FLP-80DOS or FLP-80DOS/MDX. Requires 48K bytes of RAM.	MK78158
FLP-80DOS Software Library	Twenty-three useful programs for users running FLP-80DOS. Both source and binary supplied for each program on diskette.	MK78164
AID-80F	AID-80F Z80 Development System with 32K of RAM. Includes all FLP-80DOS programs in binary	MK78125
SYS-80FT	Complete dual disk system with 58K RAM. Standard software and 30 x 24 display terminal.	MK78042

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Analog/Digital Converter (A/D-80)

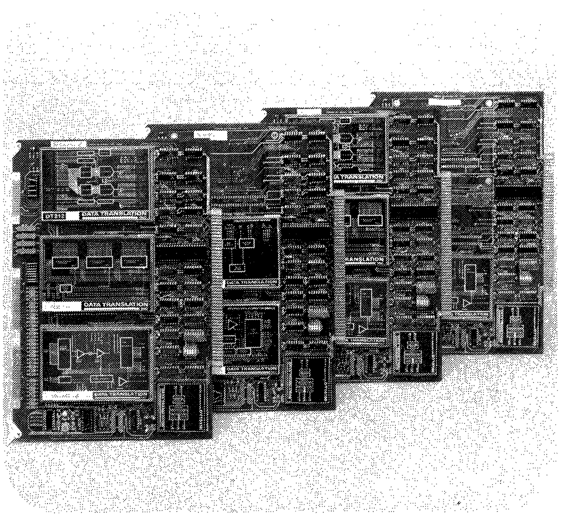
FEATURES

- □ Complete line of Analog I/O Systems for Mostek's SD Series
- □ Three Analog input versions available:
  - A/D-80/1792
    - High-level (±5V, ±10V, 0-10V, 0-5V)
    - Choice of 16 SE, 8 DI, 64 SE, or 32 DI channels
    - 12 bit A/D converter
    - Programmable gain option
    - Current loop input option
  - A/D-80/1794
    - Low-level, wide range (±10mV to ± 10V)
    - Choice of 16SE, 8DI, 64SE, or 32DI channels
    - 12 bit A/D converter
  - A/D-80/1798
    - Wide-range, isolated inputs (± 10mV to ±10V)
    - 4 DI or 12 DI channels
    - Programmable gain option
    - Current loop input option.
- □ Three Analog I/O versions available:
  - A/D-80/1791
    - Same analog input features as 1792
    - 2 D/A output channels with scope control
  - A/D-80/1795
    - Same analog input features as 1794
    - 2 D/A output channels with scope control
  - A/D-80/1799
    - Same analog input features as 1798
    - 2 D/A output channels with scope control
- □ One Analog Output Version available
  - A/D-80/1796
    - 2 D/A output channels with scope control.

GENERAL DESCRIPTION

The System Design Series (SD Series™) of OEM microcomputer boards offers powerful features and versatility to the OEM. Utilizing the MOSTEK Z80 and MOSTEK's industry-standard memories, the SD Series enables the user to construct high-performance, memory-intensive systems for a wide range of application.

The A/D-80 line of analog I/O systems is offered as a part of the SD Series. Available in seven different versions with various user-specified options, the A/D-80 can be configured with the right combination of analog I/O to meet the system designer's needs.



MOSTEK offers three types of low-cost analog input systems for three different input voltage ranges. The 1792, 1794, and 1798 handle the input voltage ranges as shown under "SPECIFICATIONS". The user can order up to 64 single-ended analog input channels on either the A/D-80/1792 or the A/D-80/1798. In addition, a programmable gain option and current loop input option is available on the A/D-80/1792 and A/D-80/1798.

The A/D-80/1791, A/D-80/1795, and A/D-80/1798, have the same A/D input features as the A/D-80/1792, A/D-80/1794, and A/D-80/1798, respectively, with the addition of two D/A output channels with scope control. These three boards are complete analog I/O systems.

The A/D-80/1796 is an analog output-only system featuring two D/A output channels with scope control.

The interface to the A/D-80 contains all the control circuitry for Z axis and scope control mode bits.

The A/D-80 line of analog I/O boards has been designed using DATAX-II™ data acquisition modules manufactured by Data Translation, Inc. Each DATAX module is a complete self-contained unit with multiple shielding for operation in a microprocessor system. This eliminates ground-loop and noise problems inherent in interconnection of separate modules.

SD Series

A/D DATA

A/D data is directly addressable in two Bytes. The data has the following formats:

D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
MSB ____ ____ ____MSB					10	9	8	7	6	5	4	3	2	1	LSB
SIGN FILL					A/D DATA										

D/A CONVERTER DATA FORMAT

D/A data can be loaded directly in two bytes. DAT 1 must be loaded prior to DAT 2 so that the D/A will not glitch on the output. Double buffering is provided on the DAT 1 data.

DAT 1								DAT 2							
D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
Select X/Y DAC	Z	Mode Select		MSB	10	9	8	7	6	5	4	3	2	1	LSB
				DAC DATA											

The following is an explanation of the various D/A control BITS.

BIT	DESCRIPTION
X/Y select	When set = Y DAC When reset = X DAC
Z	When set will generate a Z output upon loading DAC DATA BYTE 2
MODE	Two Mode Select Bits provide 4 Mode outputs

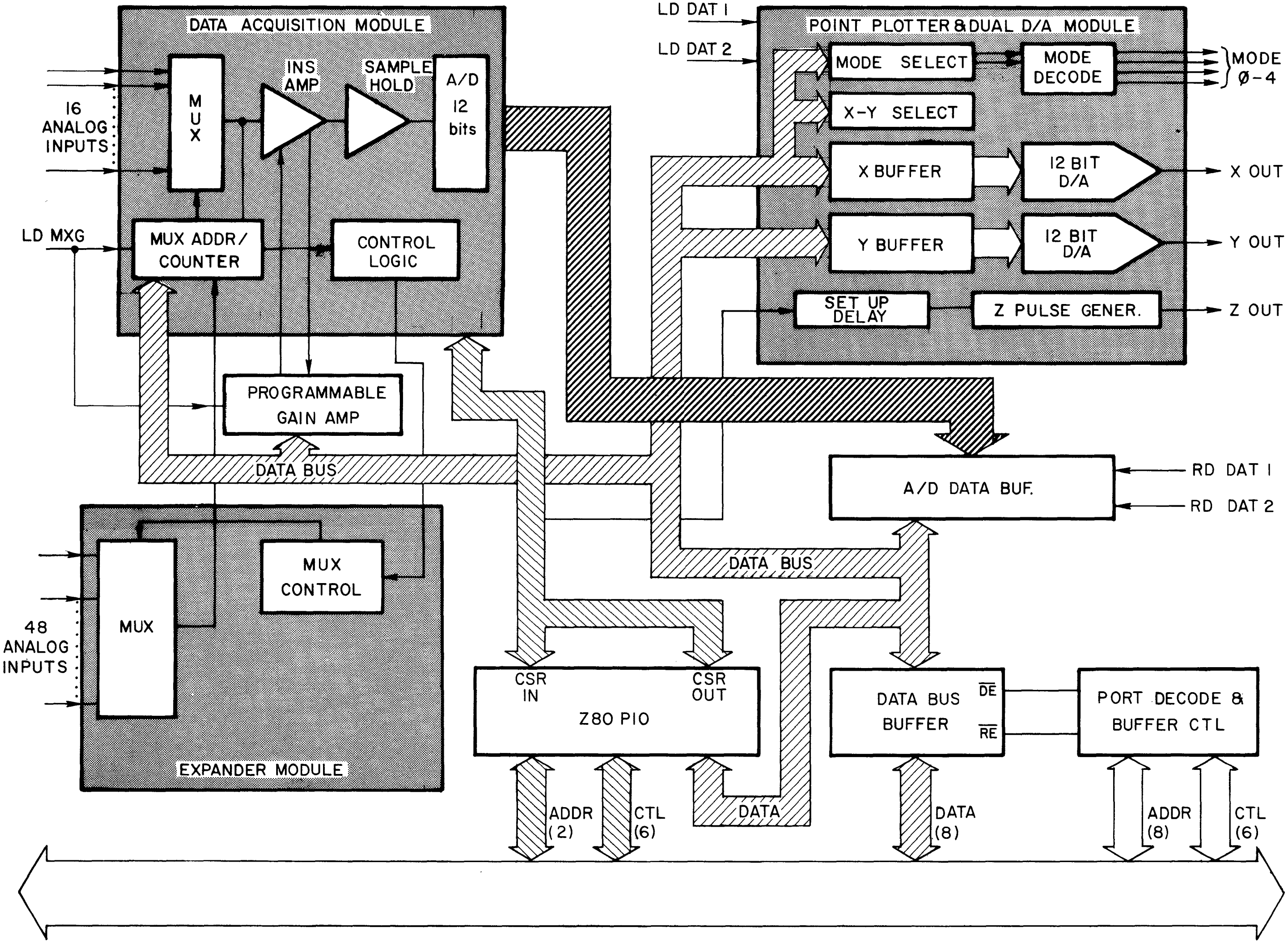
MULTIPLEXER/GAIN REGISTER
MXR (Multiplexer & Gain Register)

D7	D6	D5	D4	D3	D2	D1	D0
GS1	GS0	MXAD 5-0					

The MUX-GAIN Register is directly addressable and contains this information as follows:

BIT	NAME	DESCRIPTION			
7-6	GAIN SELECT	SET/RESET via program control			
				GAIN	GAIN
		BIT 7	BIT 6	1791	1798
		0	0	1	1
		0	1	2	10
		1	0	4	100
		1	1	8	500
5-0	MUX ADDRESS	Select 1 of 64 multiplier address SET/RESET by program control.			

A/D BLOCK DIAGRAM



A/D-80
ANALOG INPUT
SPECIFICATION

	A/D-80/1791 & 1792	A/D-80/1794 & 1795	A/D-80/1798 & 1799
Number of channels	Up to 64 single ended or 32 differential	*	4 differential or 12 differential
Input Impedance	100 megOhm	*	10 megOhm
Input Overvoltage	±35V non-destructive	±15V non-destructive	15V DC max.
Input Range	0-5V, ±5V, 0-10V, ±10V All jumper selectable	0-10mV, 0-10V, ±10mV, ±10V selectable via a single register	0-10V unipolar, ±10V bipolar
Optional programmable gain amplifier	gains: 1, 2, 4, 8	Not Available	gains: 1, 10, 100, 500
Conversion resolution	12 bits	*	*
Linearity	±½LSB	±½LSB	±½LSB
Inherent quantizing error	±½LSB	*	*
Stability Tempco	± 25ppm/°C, F.S.R.	± 30ppm/°C	Zero - ± 20 microVolt/°C Full Scale - ± 30ppm/°C
Throughput	35KHz stand. 100KHz optional	31KHz	20 Hz random 40 Hz sequential
Power Requirements	+5V @ 2.0 A Max	*	*
Mechanical printed circuit board	8.5" x 12.0" x .65"	*	*
Temperature	0° - 70°C	*	*
Implementation	Programmed I/O and Interrupt Functions	*	*
Device address	Selectable via jumper	*	* * - Same as 1791 ** - Same as 1795

Analog Output Specifications

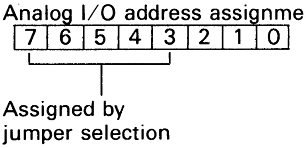
Resolution -	12 Bits
Linearity -	±1/2 LSB
Range -	±10V, 0 - 10 V; @ 25 mA minimum current output, all jumper selectable.
Relative Accuracy -	±0.025%
Full Scale Settling -	0.1% - 1 microsecond, 0.01% -3 microsecond into 50ft, coaxial cable terminated with 470 Ohm
Temperature Coefficient -	25 ppm/°C
Z Axis Control -	The Interface contains all the control circuitry of Z axis and scope control mode bits.
Z Output (Intensity) -	LO (0.8V) to HI (2.4V) TTL compatible into 50 Ohm termination
Z Risetime -	100 nsec into 50 ft. of terminated COAX
Z Pulse Width -	Jumper Selectable a. 0.5 microsecond b. 5 microsecond c. external RC 1 microsecond to 0.5 msec

A/D-80 INTERFACE

The Z80-PIO chip and some external logic are utilized to provide the interface for the A/D-80. In this manner, the Z80-PIO chip is used to provide all the interrupt circuitry for Z80- Mode 2 operation, i.e., a vectored daisy chain priority interrupt structure.

The I/O addresses are jumper selectable in groups of 8 addresses anywhere in the I/O address space of the Z80. Once bits 7 thru 3 are assigned the low order bits 2,1, and 0 have the following assignments.

Analog I/O address assignments



<u>Z 1 0</u>

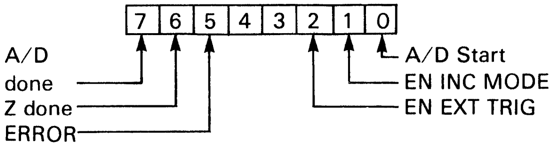
0 0 0 - PIO CHIP PORT A CSR IN
0 0 1 - PIO CHIP PORT B CSR OUT
0 1 0 - PIO CHIP PORT A CONTROL
0 1 1 - PIO CHIP PORT B CONTROL
1 0 0 - DAT 1 I/O (A/D INPUT DATA - D/A OUTPUT DATA)
1 0 1 - DAT 2 I/O (A/D INPUT DATA - D/A OUTPUT DATA)
1 1 0 - MXG (MUX - GAIN REGISTER)
1 1 1 - Not used

*CSR = Control and Status Register

PROGRAMMING

The following is a description of the I/O ports used in programming the A/D-80.

CONTROL & STATUS REGISTER (CSR)



Port A and Port B of the PIO chip are utilized to implement the CSR function. Port A of the PIO chip in Mode 3 operation is utilized for the input of the CSR and Port B of the PIO chip in Mode 0 is utilized for the output function to the CSR.

The control and status register provides all the control of the interface as follows:

BIT	NAME	DESCRIPTION
7	A/D Done	Set by A/D data ready, reset by read A/D Data Byte 2. This bit is READ only, and is reset by initialization.
6	Z Done	Set by the trailing edge of the Z output pulse, reset by program control.
5	ERROR	This bit is used when external triggers are utilized to start the A/D conversion. It will be set with the following conditions, if EXT start is received. 1. During MUX settling time 2. During A/D conversion 3. Before A/D DATA has been READ. The bit can be reset by program control and is reset by initialize.
4-3	UNUSED	
2	EN EXT TRIG	When set, this bit will enable the user of the external trigger inputs to start A/D conversions. Reset by initialize, controlled via program.
1	EN INC MD	When set via program control this bit allows the A/D multiplex to run in increment mode as follows: 1. When A/D start is set (Bit O CSR) the A/D will increment to the next sequential channel and start a conversion on that channel. 2. If external trigger enable is set, when any external trigger is received the channel will be incremented and a conversion started on that channel. Reset by program control and initialize.
0	A/D START	When set by program control this bit will start the A/D converter. It is WRITE only, and always reads as O.

A/D PART NUMBERS, OPTIONS, & PRICES

ORDER INFORMATION			
PART NO.	OPTIONS	DESIGNATION	DESCRIPTION
78155	A B C D	A/D-80/1791	High level analog I/O board
78166	A B C D	A/D-80/1792	High level analog aboard
78167	A X X D	A/D-80/1794	Low level, nonisolated analog input board
78168	A X X D	A/D-80/1795	Low level, non-isolated analog I/O board
78169	X X X X	D/A-80/1796	Analog output board
78170	*A B C D	A/D-80/1798	Wide range, isolated analog board
78171	*A B C D	A/D-80/1799	Wide range, isolated analog I/O board

*Available with 4 or 12 differential input only

OPTION	EXPLANATION
A	16 = 16 single ended inputs 64 = 64 single ended inputs **4 = 4 differential inputs 8 = 8 differential inputs **12 = 12 differential inputs 32 = 32 differential inputs
B	0 = No programmable 1 = programmable gain

**Available on 78170 and 78171 only

OPTION	EXPLANATION
*** C	0 = No current loop input 1 = Current loop input
D	0 = Standard 35KHz throughput 1 = Optional 100KHZ throughput
X	= Not available option

***Supplied in differential configuration only

SAMPLE

MK78166 - $\frac{A}{32}\ \frac{B}{1}\ \frac{C}{1}\ \frac{D}{1}$

MOSTEK®

MICROCOMPUTER SYSTEMS

Video Adaptor Board (VAB-2)

FEATURES

□ Complete video interface system on one board

□ Single supply (+5VDC or 12.6VAC) operation

□ On board rectifier and regulator for 12.6VAC operation

□ 16 lines of 64 characters

□ Full ASCII character set - 128 symbols including upper/lower case letters

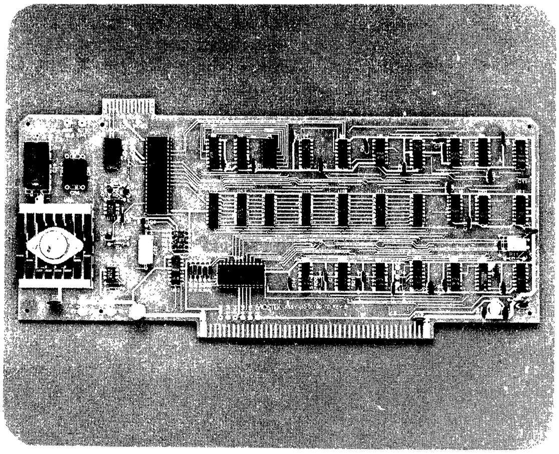
□ Full cursor controls: ↑ ↓ ← → home, screen clear, carriage return, erase to end of line/screen; plus direct X-Y addressing

□ 8 bit ASCII or 5 bit Baudot operation

DESCRIPTION

The VAB-2 is a single board video terminal based on the MOSTEK MK3870 single chip microcomputer. It functions as an interface between a 20mA full duplex serial data loop, an ASCII encoded keyboard, and an EIA standard video monitor. The only other external component required is a 12.6 volt transformer.

The P.C. board 'form factor' facilitates installation within most standard keyboard housings. Alternatively, the 2 inch power supply section may be cut off the P.C. board allowing the board to be inserted into a standard 12" card rack (such as Mostek's XAID-100 MK79034) for system use.



SPECIFICATIONS

Operating Temperature 0° C – 50° C

Power Supply Requirements
5VDC±5% @ 0.75A max.
or
8 – 14 VAC rms @ 0.75A rms max.

Board size (with power supply) 14" x 6.5" x 1"
(without power supply) 12" x 6.5" x 1"

Video output 1.5Vp-p into 75Ω (EIA RS-170)

Current loop input/output 20mA nominal opto-isolated 240V max loop to ground

Keyboard inputs – standard TTL compatible

CUSTOMER SUPPLIED EQUIPMENT

Keyboard – Cherry B70-4753 or equivalent

Monitor – SC Electronics, Inc. 10M915 or equivalent

Transformer – Stancor P8384 or equivalent

SD Series

MICROCOMPUTER BASED

The heart of the VAB-2 is the MK3870 single chip microcomputer. The MK3870 provides the following functions:

Serial data link interface
Control character decoding
Cursor positioning
Keyboard interface

ASCII OPERATION

In ASCII mode, the VAB-2 receives and transmits an 8 bit code (parity bit = 0 on transmit, ignored on receive). Two stop bits are transmitted by the VAB-2, but only one stop bit is required by the VAB-2 receiver. The VAB-2 works equally well with external systems transmitting one, two, or more stop bits. Available Baud rates for ASCII are 300 and 110.

See also Figure 1 — ASCII character set, and Table 1 — ASCII control characters.

BAUDOT OPERATION

In Baudot mode, the VAB-2 receives and transmits a 5 bit code (compatible with Model 15, Model 28, or similar Teletypes™). Two stop bits are transmitted, but only one stop bit is required by the VAB-2 receiver. The VAB-2 works equally well with external systems transmitting one, 1.5, or more stop bits. Available Baud rates for Baudot are 74.2 and 45.45. In Baudot mode, the only control codes available are carriage return and line feed. The Baudot "Letters" and "Figures" shift characters are generated automatically as required. Keys on the ASCII keyboard which generate codes having no equivalent Baudot code are ignored. ASCII code "Rubout" ($7F_{16}$ or 177_8) generates a "Letters" shift to facilitate synchronization of the distant end receiver.

ASCII CHARACTER SET

α β γ δ ϵ θ ι λ μ ν π Σ ϕ ψ ω Ω 0 1 2 3 ⁰ ² _ ÷ ≏ √ ∫ | ← ↑↓
! " # $ % & ' ( ) * + , – ; / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ] ^
` a b c d e f g h i j k l m n o p q r s t u v w x y z { ¦ } ~ ▒

Figure 1

BAUDOT CHARACTER SET

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
– ? : * 3 $ & # 8 ( ) . , 9 0 1 4 ! 5 7 ; 2 / 6 "

Figure 2

FUNCTIONAL DIAGRAM

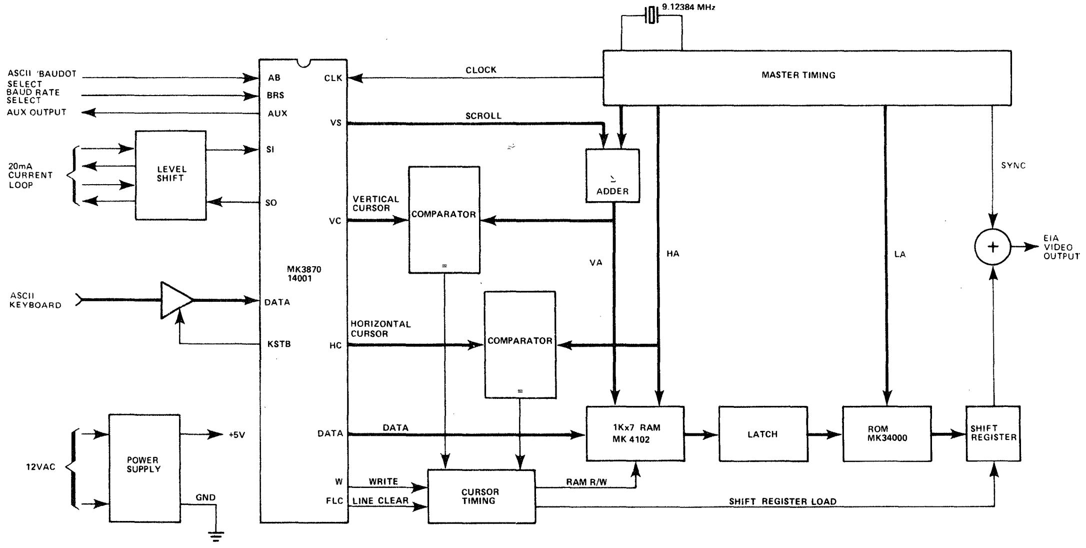


Figure 3

OCTAL	HEX	CNTL	FUNCTION	
004	04	D	HOM	Home – moves cursor to upper left corner of screen
005	05	E	EOL	Erase end of line – erases current line from right margin to current cursor position (1600mS max)
006	06 1	F	EOS	Erase end of screen – erases lines from bottom of screen to, but not including, current line (400mS max)
010	08	H	BS	Back space – move cursor left one column unless already in left most column
011	09	I	HT	Horizontal tab – moves cursor right one column unless already in right most column
012	0A	J	LF	Line feed – moves cursor down one line, scrolls screen up if already on bottom line
013	0B	K	VT	Vertical tab – moves cursor up one line, scrolls screen down if already on top line
014	0C	L	FF	Form feed – clears screen and homes cursor (400mS)
015	0D	M	CR	Carriage return – moves cursor to left margin
020	10	P	DS	Down shift sequence – causes character following DS to be interpreted as printable rather than control. Required for lower 32 symbols (Greek and math), but may be used with any characters.
021	11	Q	DC1	Device control – sets AUX bit
023	13	S	DC3	Device control – clears AUX bit
033	1B		ESC	Start cursor sequence – ESC + ΔV ΔH adds ΔV modulo 16 to vertical cursor address ΔH modulo 64 to horizontal cursor address ESC = ΔV ΔH sets vertical cursor address to ΔV modulo 16 horizontal cursor address to ΔH modulo 64
177	7F		DEL	Delete – moves cursor left one column, unless cursor was already on leftmost column; erases new position

TABLE 1. – ASCII CONTROL CHARACTERS

CHARACTER GENERATOR

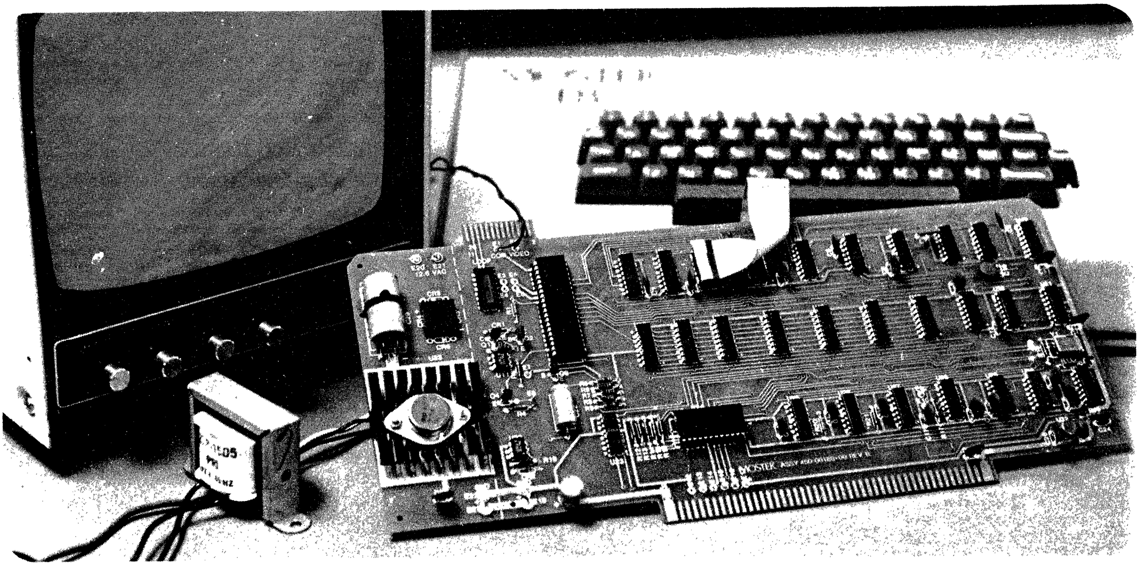
The VAB-2 is shipped with an MK34073 (2K x 8) character generator ROM, providing 128 displayable characters (see Figure 1 – ASCII character set). For custom applications, the MK34073 ROM may be removed and an MK2708 type PROM (1K x 8) installed, programmed with the user's custom font (external +12V and -5V or -12V supplies required for some PROMs). Alternatively, for high volume applications, a new ROM mask may be ordered. The MK34000 series can provide two complete 128 character sets per ROM. Provision is made for wiring the AUX bit to the ROM for program-selectable character font.

AUXILLARY BIT OUTPUT

A special output (AUX) is provided for custom control applications. AUX is capable of driving one TTL load, and is brought out to the P.C. edge connector. AUX is cleared upon power up and each time a DC3 character is recieved. AUX is set upon receipt of a DC1 character.

KEYBOARD

The VAB-2 interfaces directly with standard ASCII encoded keyboards. Although normally used with active high data and strobe keyboards, provision is made for active low keyboards.



CUSTOMER SELECTABLE OPTIONS

- □ 50/60 Hz (Strap option)
- □ 110/300 Baud ASCII (strap option)
- □ 74.2/45.45 Baud Baudot (strap option)
- □ MK34000 series ROM or MK2708 type PROM character generator (strap and population option; MK34073 standard)
- □ 5VDC or 12VAC operation (strap and population option; 12 VAC standard)
- □ Serial loop connector – 16 pin DIP socket or 26 pin edge connector
- □ Active high or active low keyboard input (population option; active high standard)
- □ Custom features and/or character generator for high volume OEM applications (one-time mask charge applicable)

ORDER INFORMATION

NAME	DESCRIPTION	PART NO.	PRICE
VAB-2 Operations Manual	Detailed description of the use and operation of VAB-2	MK79560	$ 1.50
VAB-2 Source Listing	Source Listing of the 3870 Firmware used in VAB-2	MK79561	$ 15.00
MK3870/ 14001 Firmware Package	Pre-programmed 3870 used with VAB-2 plus the Operations Manual and Source Listing described above	MK79056	$ 50.00
VAB-2	Assembled and tested VAB-2 Circuit Board plus the Operations Manual and Program Source Listing	MK79052	$195.00

*Prices are subject to change without notice and apply only toU.S. and Canada.

SD Series

1979 MICROCOMPUTER DATA BOOK

MOSTEK®

REFERENCE GUIDE

Military Products

INTRODUCTION

The Defense and Aerospace industries are more concerned than ever today about the price and performance of the integrated circuits they purchase. The traditional military IC manufacturers have met the stringent reliability requirements of the military at the cost of being several years behind the state of the art in commercial products. Mostek, which has long been known for being at the leading edge in MOS technology, has a separate Military Products Department serving the special needs of the Defense and Aerospace industries. This organization has the primary objective of providing Mostek's state of the art products screened to MIL. STD. 883, Methods 5004.4 and 5005.5. As MIL-M-38510 slash sheets are issued, the Military Products Department will qualify Mostek's products in the JAN 38510 program.

Mostek is currently pursuing the qualification of the industry standard 16K dynamic RAM, MK4116. This effort will result in the QPL listing of the most advanced MOS device to date.

This brochure describes each of the products Mostek currently offers to MIL. STD. 883B, Method 5004.4, Class B. They are prefixed "MKB" rather than "MK" to designate Class B. Class C and extended temperature devices are available via special order.

PRODUCT OFFERINGS

MKB 4116

The industry standard 16K x 1 Dynamic RAM, Mostek's MK 4116, is available for military requirements as the MKB 4116. Screened to full 883B, Method 5004.4, Class B requirements, Mostek offers the MKB 4116 in three packages; the 16 pin CERDIP ("J"), the 16 pin flat package ("F"), and the leadless chip carrier ("E"). Mostek has received DESC Line Certification for the 4116. A supplementary data sheet (MKB4116-(P/J)-82/83/84) is available.

Device	Temp. Range	Access Time	Notes
MKB 4116J93	-55° to 85°C	200ns	1ms refresh
MKB 4116J83	-55° to 85°C	200ns	
MKB 4116J84	-55° to 85°C	250ns	
MKB 4116J2	0° to 70°C	150ns	
MKB 4116J3	0° to 70°C	200ns	
MKB 4116J4	0° to 70°C	250ns	
MKB 4116F84	-55° to 85°C	250ns	16 Pin Flat
MKB 4116E84	-55° to 85°C	250ns	Leadless chip carrier

MKB 4027

Mostek's industry standard 4K x 1 Dynamic RAM is available for military applications as the MKB 4027. Two versions are available, the standard 16 pin Cer-DIP ("J") and Mostek's 16 pin flat pack ("F"). Both are screened to MIL. STD. 883B. Because the pinout is Mostek's standard, easy upgrading to the MKB 4116 is possible, allowing the corresponding increase in system densities. Supplementary data sheets for the MK4027J 83/84 are available.

Device	Temp. Range	Access Time	Notes
MKB 4027J83	-55° to 85°C	200ns	
MKB 4027J84	-55° to 85°C	250ns	
MKB 4027J2	0° to 70°C	150ns	
MKB 4027J3	0° to 70°C	200ns	
MKB 4027J4	0° to 70°C	250ns	
MKB 4027F84	-55° to 85°C	250ns	16 Pin Flat

MKB 4104

Mostek's popular 4K x 1 static RAM, the MK 4104, has earned its stripes as a full temperature range military device. Featuring Mostek's Edge-Activated™ technology, the MKB 4104 offers a rare combination of low power and high speed among static MOS 4K RAMs.

Reliability is greatly enhanced by the low power dissipation which causes a maximum junction rise of only 6.6 °C at 1.6 MHz operation. The device is packaged in a standard 18 pin CERDIP, and is screened 100% to the requirements of MIL. STD. 883B, Method 5004.4, Class B. A supplementary data sheet, the MK4104P85/86, is available on request.

Device	Temp. Range	Access Time	Notes
MKB 4104J85	-55° to 125°C	300ns	
MKB 4104J86	-55° to 125°C	350ns	
MKB 4104J4	0° to 70°C	250ns	
MKB 4104J5	0° to 70°C	300ns	
MKB 4104J6	0° to 70°C	350ns	

Military ROMs

The state of the art in ROMs is available from Mostek, with military processing.

The MKB 34000 2K x 8 ROM, and the MKB 36000 8K x 8 ROM are available with the full military temperature range and full 883B processing. Both are packaged in standard 24 pin DIPs which are 2708/2716 compatible, and receive rapid prototype turnaround from Mostek's ROM services group. Because the user can select from three temperature ranges, the cost of Mostek's military mask programmable ROMs can be substantially less than E PROMs. Supplementary data sheets are available for the extended temperature versions.

Device	Temp. Range	Access Time	Notes
MKB 36000P84	-55° to 125°C	300ns	
MKB 36000P80	-40° to 85°C	300ns	
MKB 36000P	0° to 70°C	250ns	
MKB 34000P84	-55° to 125°C	450ns	
MKB 34000P80	-40° to 85°C	450ns	
MKB 34000P	0° to 70°C	350ns	

MKB 3880

Leading Mostek's microprocessor batallion is the MKB 3880, the Z80 CPU which is fully software compatible with the popular 8080A. Thus, without the significant investment in software often required, the user can now design in the most advanced 8 bit microcomputer system on the market, while meeting stringent military requirements. Mostek is firmly committed to supporting the Z80 family of peripheral chips with 883B screening, and will pursue JAN qualification of the Z80 should a slash sheet be issued.

Device	Temp. Range	Fre-quency	Max I_{CC}
MKB 3880P20	-55° to 125°C	2.5MHz	200mA
MKB 3880P14	-40° to 85°C	4.0MHz	200mA
MKB 3880P10	-40° to 85°C	2.5MHz	200mA
MKB 3880P4	0° to 70°C	4.0MHz	200mA
MKB 3880P	0° to 70°C	2.5MHz	150mA

MKB 3870

The 3870 is now the green beret of military microprocessors, performing on one chip what formerly required three. The 3870 is software compatible with Mostek's popular F8 family, and is supported by a full line of development systems.

Furthermore, Mostek's ISE concept (In Socket Expandability) ensures that 3870 users will not be outflanked by future advances: the 3872, 3874, and 3876 all will use the 3870 pinout. Mostek will pursue JAN qualification of the 3870 when a slash sheet is issued, and is actively pursuing design modifications to expand the temperature range of the 3870.

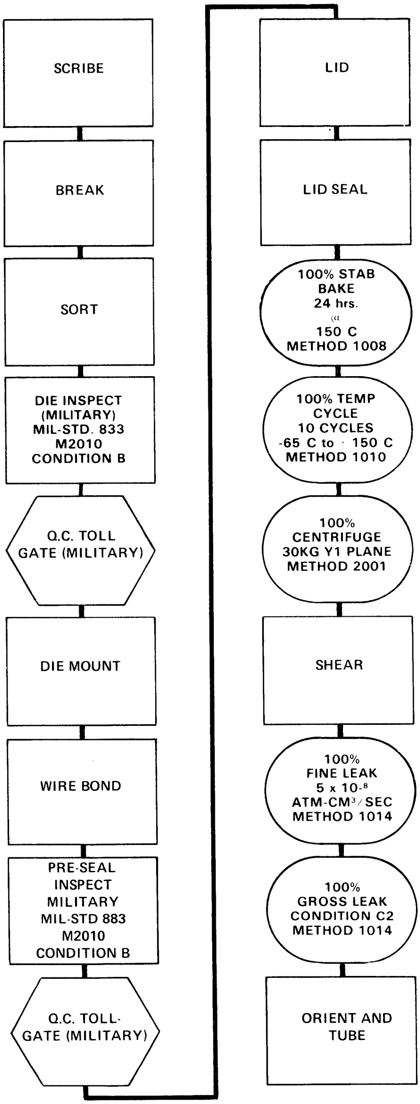
Device	Temp. Range	Speed	Notes
MKB 3870P10	-40° to 85°C	4MHz	
MKB 3870P	0° to 70°C	4MHz	

NEW PRODUCT STATUS

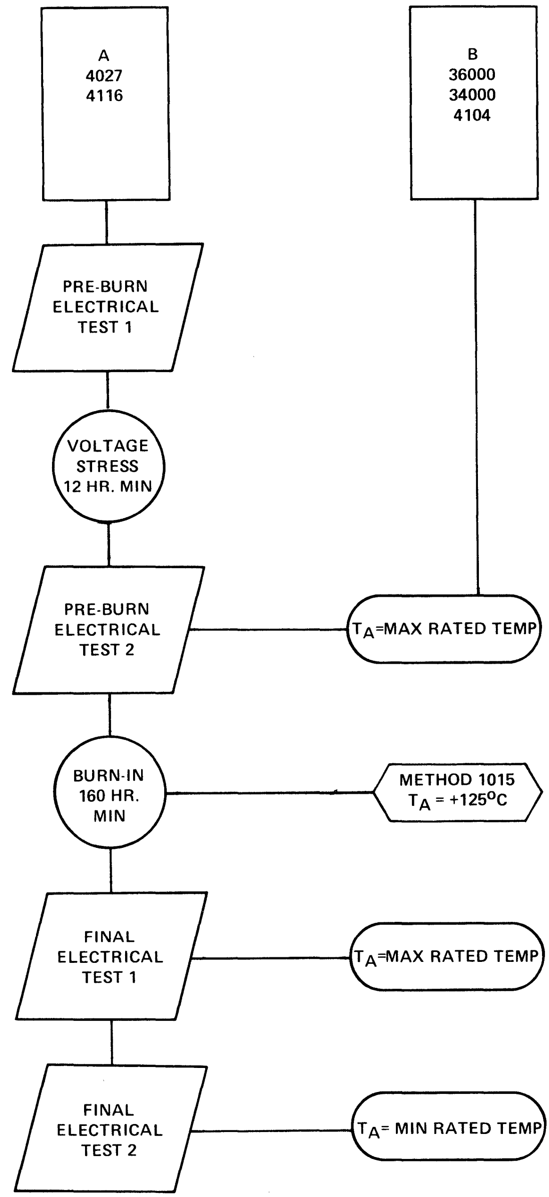
Device	Temp. Range	Speed	Sample Avail-ability
MKB 3881P10	-40° to 85°C	2.5MHz	2Q 79
MKB 3881P14	-40° to 85°C	4.0MHz	TBA
MKB 3881P20	-55° to 125°C	2.5MHz	TBA
MKB 3882P10	-40° to 85°C	2.5MHz	2Q 79
MKB 3882P14	-40° to 85°C	4.0MHz	TBA
MKB 3882P20	-55° to 125°C	2.5MHz	TBA
MKB 2716T	TBA	TBA	3Q 79
MKB 4118P	TBA	TBA	3Q 79

TBA = to be announced

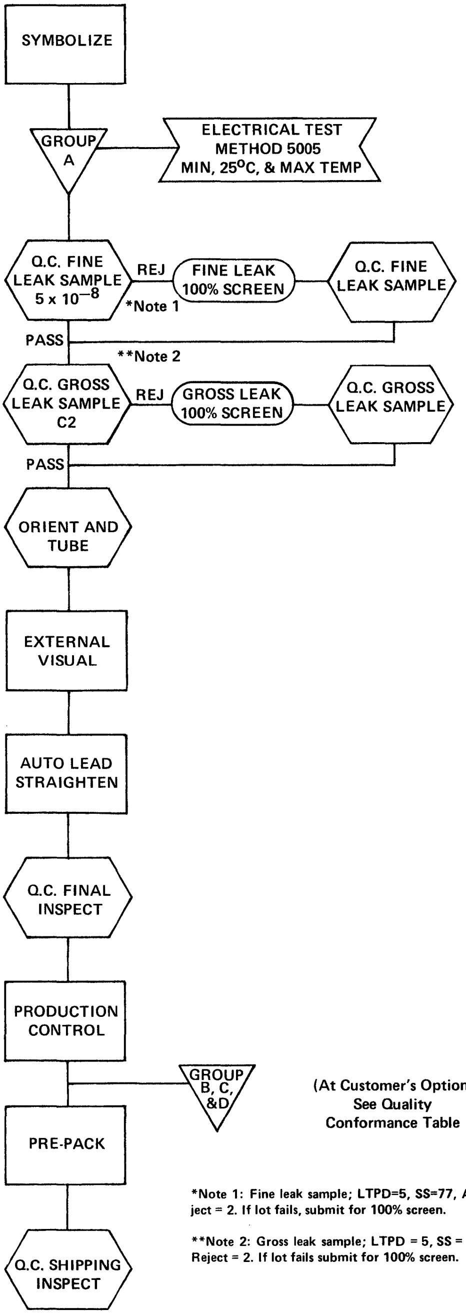
Mostek "MKB" FLOW CHART
Mil. Std. 883B, Method 5004, Class B


SCRIBE
BREAK
SORT
DIE INSPECT (MILITARY) MIL-STD. 833 M2010 CONDITION B
Q.C. TOLL GATE (MILITARY)
DIE MOUNT
WIRE BOND
PRE-SEAL INSPECT MILITARY MIL-STD 883 M2010 CONDITION B
Q.C. TOLL-GATE (MILITARY)
LID
LID SEAL
100% STAB BAKE 24 hrs. @ 150 C METHOD 1008
100% TEMP CYCLE 10 CYCLES -65 C to · 150 C METHOD 1010
100% CENTRIFUGE 30KG Y1 PLANE METHOD 2001
SHEAR
100% FINE LEAK 5 x 10-8 ATM-CM3/SEC METHOD 1014
100% GROSS LEAK CONDITION C2 METHOD 1014
ORIENT AND TUBE

Mostek "MKB" FLOW CHART
Mil. Std. 883B, Method 5004, Class B



Mostek "MKB" FLOW CHART
Mil. Std. 883B, Method 5004, Class B



(At Customer's Option)
See Quality
Conformance Table

*Note 1: Fine leak sample; LTPD=5, SS=77, Accept = 1, Reject = 2. If lot fails, submit for 100% screen.

**Note 2: Gross leak sample; LTPD = 5, SS = 77, Accept = 1, Reject = 2. If lot fails submit for 100% screen.

MOSTEK MILITARY DEVICE ORDERING INFORMATION-GENERAL

All Devices - Part Numbering

The part number consists of:
-Prefix- -Basic Part Number- -Electrical Sort-

Prefix - MKB designates processing to MIL-STD-883B, Method 5004, Class B, with 100% screening at the minimum, room, and maximum rated temperatures.

Basic Part Number - Is the same as for the generic device.

Electrical Sort - Specifies the temperature range, and in some cases other electrical characteristics.

ELECTRICAL SORT SYSTEMS

Microprocessors

	Temp.	3870	3880
- 0	0° - 70°C	4.0MHz	2.5MHz
- 4	0° - 70°C	N/A	4.0MHz
-10	-40° to 85°C	4.0MHz	2.5MHz
-14	-40° to 85°C	N/A	4.0MHz
-20	-55° to 125°C	N/A	2.5MHz

RAMs & EPROMs

- X	0° to 70°C. Commercial Speed as Specified per "X"
-8X	Min to Max rated temperature, speed per "X"
-9X	Special attributes as defined

ROMs

- 0	0° to 70°C
-80	-40° to +85°C
-84	-55° to +125°C

SUPPLEMENT

MOSTEK®

Z80 CENTRAL PROCESSING UNIT

Extended Operating Range

MKB3880(P)-20/34/14

FEATURES

□ Screened per MIL-STD 883, Method 5004 Class B

□ Extended operating ranges:
- -55° to +125°C (-20)
- -55° to +100°C (-34)
- -40° to +85°C (-14)

□ Single 5-Volt supply and single-phase clock required

□ Z80 CPU and Z80 A CPU

□ Software compatible with 8080A CPU

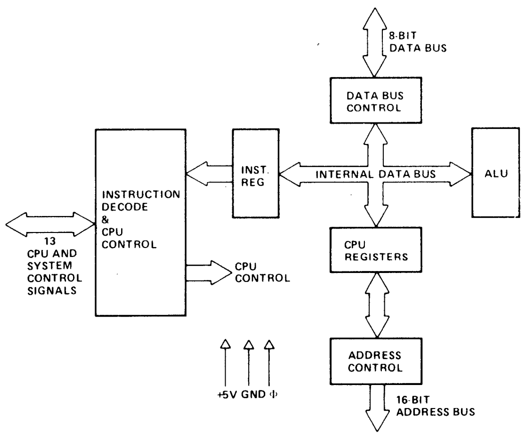
□ Complete development and OEM system product support

DESCRIPTION

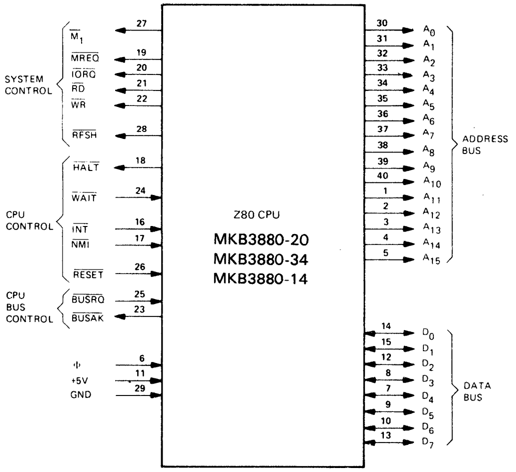
The Mostek Z80 family of components is a significant advancement in the state-of-art of microcomputers. These components can be configured with any type of standard semiconductor memory to generate computer systems with an extremely wide range of capabilities. For example, as few as two LSI circuits and three standard TTL MSI packages can be combined to form a simple controller. With additional memory and I/O devices, a computer can be constructed with capabilities that only a minicomputer could deliver previously. This wide range of computational power allows standard modules to be constructed by a user that can satisfy the requirements of an extremely wide range of applications.

The CPU is the heart of the system. Its function is to obtain instructions from the memory and perform the desired operations. The memory is used to contain instructions and, in most cases, data that is to be processed. For example, a typical instruction sequence may be to read data from a specific peripheral device, store it in a location in memory, check the parity, and write it out to another peripheral device. Note that the Mostek component set includes the CPU and various general purpose I/O device controllers, as well as a wide range of memory devices. Thus, all required components can be connected together in a very simple manner with virtually no other external logic. The user's effort then becomes primarily one of the software development. That is, the user can concentrate on describing his problem and translating it into a series of instructions that can be loaded into the microcomputer memory. Mostek is dedicated to making this step of software generation as simple as possible. A good

Z80-CPU BLOCK DIAGRAM



Z80 PIN CONFIGURATION



example of this is our assembly language in which a simple mnemonic is used to represent every instruction that the CPU can perform. This language is self-documenting in such a way that from the mnemonic the user can understand exactly what the instruction is doing without constantly checking back to a complex cross listing.

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	Specified Operating Range
Storage Temperature	−65°C to +150°C
Voltage on Any Pin with Respect to Ground	−0.3V to +7V
Power Dissipation	1.5W

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

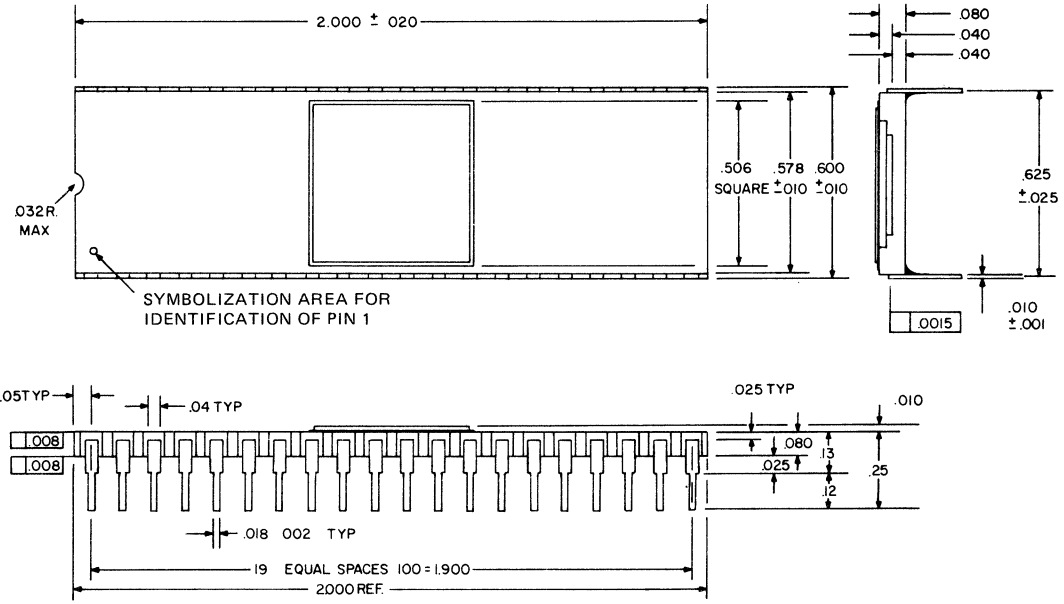
MKB3880(P)-20 TA = −55°C to 125°C
MKB3880(P)-34 TA = −55°C to 100°C
MKB3880(P)-14 TA = −40°C to 85°C
V_{CC} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min	Typ	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	−0.3		0.8	V	
V_{IHC}	Clock Input High Voltage	V_{CC}−.6		V_{CC}+.3	V	
V_{IL}	Input Low Voltage	−0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −250 microA
I_{CC}	Power Supply Current			200	mA	
I_{LI}	Input Leakage Current			10	microA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	microA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			−10	microA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current In Input Mode			±10	microA	$0 < V_{IN} < V_{CC}$

CAPACITANCE
$T_A = 25°C, f = 1MHz$

Symbol	Parameter	Max	Unit	Test Conditions
CΦ	Clock Capacitance	35	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

40 PIN CERAMIC PACKAGE DESCRIPTION



SUPPLEMENTAL DATA SHEET TO BE USED IN CONJUNCTION WITH MK3880 DATA SHEET

SUPPLEMENT

MOSTEK®

Z80 PARALLEL I/O CONTROLLER

Extended Operating Range

MKB3881(P)-20/14

FEATURES

□ Z80 PIO and Z80 A PIO

□ Two independent, 8-bit, bidirectional, peripheral interface ports with "handshake" data transfer control

□ Interrupt driven "handshake" for fast response

□ Any one of four distinct modes of operation may be selected for a port including:
- Byte output
- Byte input
- Byte bidirectional bus (Available on Port A only)
- Bit control mode
- All with interrupt controlled handshake

□ Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic

□ Eight outputs are capable of driving Darlington transistors

□ All inputs and outputs fully TTL compatible

□ Single 5 volt supply and single phase clock required

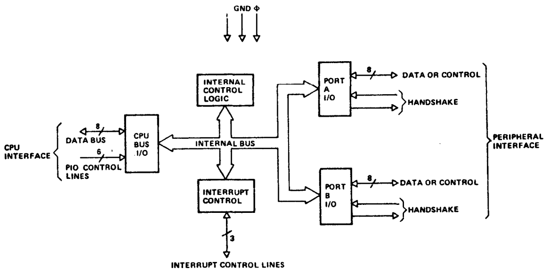
□ Screened per MIL.STD 883, method 5004, Class B.

□ Extended operating ranges:
- -55° to +125°C (-20)
- -40° to +85°C (-14)

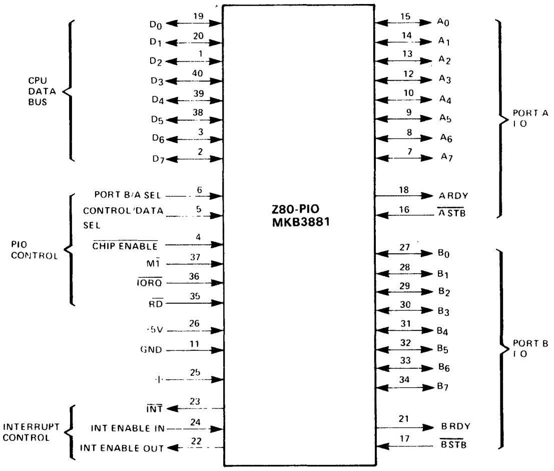
DESCRIPTION

The Z80 Parallel I/O Circuit is a programmable, two-port device which provides a TTL compatible interface between peripheral devices and the Z80-CPU. The CPU can configure the Z80-PIO to interface with a wide range of peripheral devices with no other external logic required. Typical peripheral devices that are fully compatible with the Z80-PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc. The Z80-PIO utilizes N-channel, silicon gate depletion load technology and is packaged in a 40 pin DIP.

PIO BLOCK DIAGRAM



PIO PIN CONFIGURATION



ELECTRICAL SPECIFICATIONS
ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .... Specified operating range.
Storage Temperature ....65°C to +150°C
Voltage On Any Pin With
Respect to Ground... 0.3V to +7.0V
Power Dissipation 6W

*Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

MKB3881(P)-20 - T_A = -55° to 125°C
MKB3881(P)-14 - T_A = -40° to 85°C
V_{CC} = 5V ± 5% unless otherwise specified

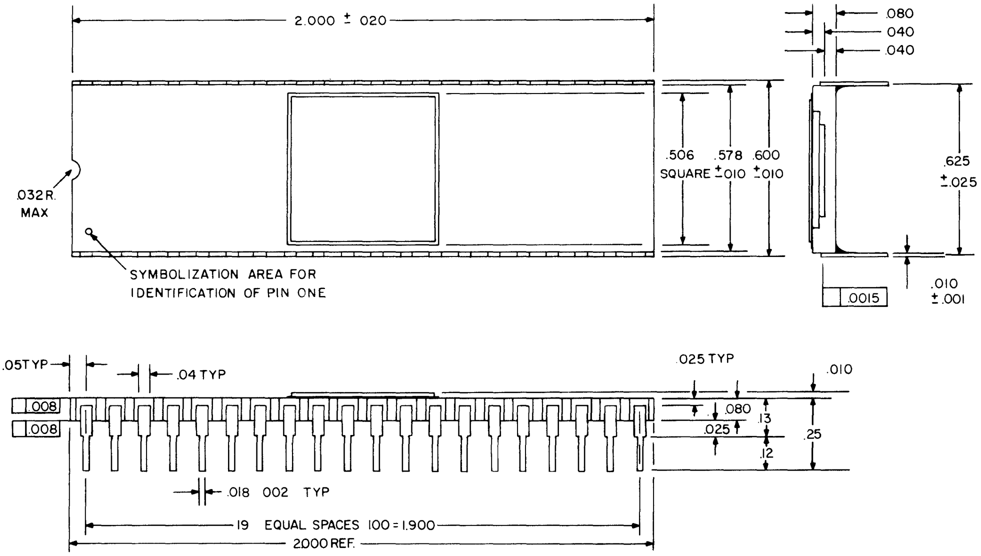
Symbol	Parameter	Min	Typ	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6		V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 2.0mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250 microA
I_{CC}	Power Supply Current			150	mA	
I_{LI}	Input Leakage Current			10	microA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current In Float			10	microA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current In Float			-10	microA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current In Input Mode			±10	microA	$0 < V_{IN} < V_{CC}$
I_{OHD}	Darlinton Drive Current	-1.5			mA	V_{OH} = 1.5V Port B Only

Military
Hi-Rel

CAPACITANCE
T_A = 25°C, f = 1MHz

Symbol	Parameter	Max	Unit	Test Conditions
C	Clock Capacitance	10	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

40 PIN CERAMIC PACKAGE DESCRIPTION



SUPPLEMENTAL DATA SHEET TO BE USED IN CONJUNCTION WITH MK3881 DATA SHEET

SUPPLEMENT

MOSTEK®

Z80 COUNTER TIMING CIRCUIT

Extended Operating Range

MKB3882(P)-20/14

FEATURES

□ Z80 CTC & Z80A CTC

□ All inputs and outputs fully TTL compatible

□ Each channel may be selected to operate in either Counter Mode or Timer Mode

□ Used in either mode, a CPU-readable Down Counter indicates number of counts-to-go until zero.

□ Selectable positive or negative trigger initiates time operation in Timer Mode. The same input is monitored for event counts in Counter Mode

□ Three channels have Zero Count/Timeout outputs capable of driving Darlington transistors

□ Interrupts may be programmed to occur on the zero count condition in any channel

□ Daisy chain, priority interrupt logic included to provide for automatic interrupt vectoring without external logic

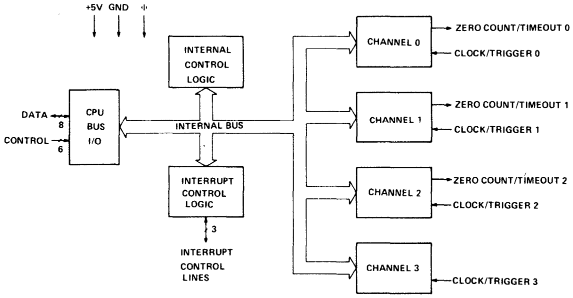
□ Screened per MIL.STD 883, Method 5004, CLASS B

□ Extended Operating Range:
- -55° to +125°C (-20)
- -40° to + 85°C (-14)

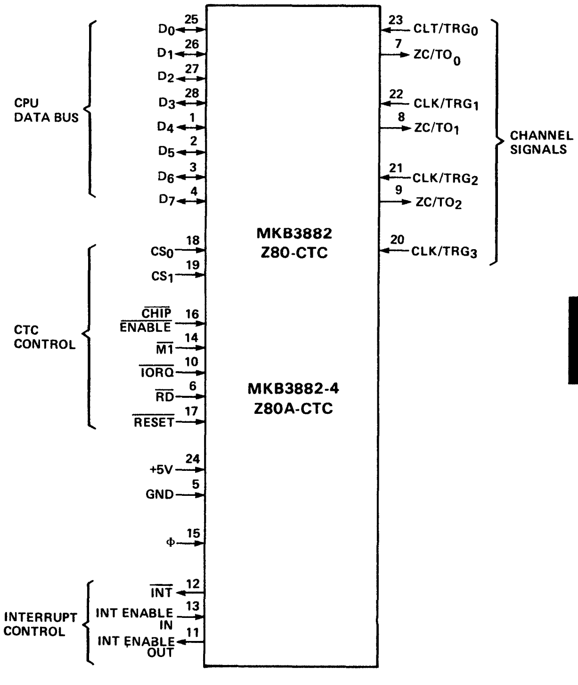
DESCRIPTION

The Z80-Counter Timer Circuit (CTC) is a programmable component with four independent channels that provide counting and timing functions for microcomputer systems based on the Z80-CPU. The CPU can configure the CTC channels to operate under various modes and conditions as required to interface with a wide range of devices. In most applications, little or no external logic is required. The Z80-CTC utilizes N-channel silicon gate depletion load technology and is packaged in a 28-pin DIP. The Z80-CTC requires only a single 5-Volt supply and a one-phase, 5-Volt supply and a one-phase, 5-volt clock.

CTC BLOCK DIAGRAM



Z80-CTC PIN CONFIGURATION



Military
Hi-Rel

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .... Specified Operating Range
Storage Temperature .... -65°C to +150°C
Voltage on Any Pin with Respect to Ground .... -0.3V to +7V
Power Dissipation 0.8W

*Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

MKB3882(P)-20 = -55°C to 125°C
MKB3882(P)-14 = -40°C to 85°C
V_{CC} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min	Typ	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6		V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 2.0mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250 microA
I_{CC}	Power Supply Current			200	mA	T_C = 400nsec**
I_{LI}	Input Leakage Current			10	microA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current In Float			10	microA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current In Float			-10	microA	V_{OUT} = 0.4V
I_{OHD}	Darlington Drive Current	-1.5			mA	V_{OH} = 1.5V

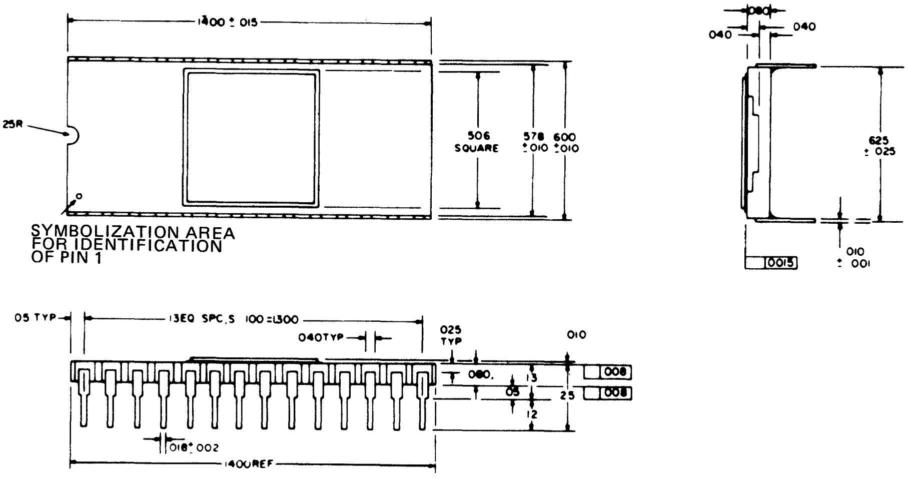
**T_C = 250 nsec for MKB3882-14

CAPACITANCE

T_A = 25°C, f = 1MHz

Symbol	Parameter	Max	Unit	Test Condition
C	Clock Capacitance	20	pF	Unmeasured Pins Returned to Ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

28 PIN PACKAGE DESCRIPTION



SUPPLEMENTAL DATA SHEET TO BE USED IN CONJUNCTION WITH MK3882 TECHNICAL MANUAL

MOSTEK®

MILITARY/HI-REL DEVICES

Quality Specification

1.0 PURPOSE – To provide a general quality specification for Military/Hi-Rel to be used with the applicable detail specification to ensure a higher than commercial level of device screening, product assurance and quality control.

2.0 SCOPE –

2.1 Statement of Scope. This specification establishes the GENERAL requirements for Military/Hi-Rel monolithic MOS/LSI microcircuits supplied by MOSTEK. This document is applicable only to devices with MKB, MKM, or MKX product designator prefixes.

2.2 Product Assurance Levels. This specification provides for three (3) levels of product assurance and screening as outlined below and in 3.4 and 3.4.1.

2.2.1. MKB MOSTEK product designator for a device processed to MIL-STD-883, Method 5004, Class B.

2.2.2. MKM MOSTEK product designator for a device processed to MIL-STD-883, Method 5004, Class B, except as modified in 3.4 and 3.4.1 (basic difference from MKB is single pass correlated hi-temp testing with guard band to guarantee 25°C and low temp).

2.2.3. MKX MOSTEK product designator for a custom military device purpose built to a customer P.O. that has some degree of military processing. See 3.4 and 3.4.1.

2.3. Applicable Documents. The following documents of issue in effect on the date of release of the MOSTEK Sales Order form a part of this specification to the extent specified herein.

A. MIL-M-38510 Microcircuits General Specification For
B. MIL-STD-883 Test Methods and Procedures for Microelectronics.
C. MIL-STD-1313 Microelectronics Terms and Definitions
D. MIL-C-45662 Calibration System Requirements
E. MOSTEK Sales Order
F. CUSTOMER Purchase Order
G. Detail Specification of Applicable Device Type (Military Data Sheet)

2.3.1. Document Hierarchy. In the event of any conflict between this document and the referred documents, the following order of precedence shall apply:

A. MOSTEK Sales Order
B. Customer Purchase Order
C. This document
D. Detail Specification (Military Data Sheet)
E. Referenced documents

Military Hi-Rel

3.0 GENERAL

3.1. General. MOSTEK, in compliance with this specification, shall have and use production and test facilities and a quality and reliability assurance program adequate to assure successful compliance with the provisions of this specification and the detail specification. The individual item requirements shall be as specified herein, and in the detail specification or drawing.

3.1.1. Reference to Detail Specification. For purposes of this specification, when the term "as specified" is used without additional reference to a specific location or document, the intended reference shall be to the detail specification or drawing number which constitutes the applicable individual device specification.

3.1.2. Terms, Definitions, and Symbols. For the purpose of this specification, the terms, definitions, and symbols of MIL-STD-883, MIL-STD-1313, and MIL-STD-1331 and those contained herein shall apply and shall be used in the applicable detail specification wherever they are pertinent.

A. **Production Lot.** A production lot shall consist of devices manufactured on the same production line(s) by means of the same production technique, materials, controls and design. Where a production lot identification is terminated upon completion of wafer or substrate processing, or at any later point prior to device dealing, it shall be permissible to process more than a single device type in a single production lot provided traceability is maintained by assembling devices into inspection lots as defined herein, at the point where production lot identification is terminated.

B. **Inspection Lot.** A quantity of microcircuits submitted at one time for inspection to determine compliance with the requirements and acceptance criteria of the applicable device specification. Each inspection lot shall consist of microcircuits of a single type, in a single package type, outline and lead finish, or may consist of inspection sublots of several different types, in a single package type, outline and lead finish defined by a single detail specification. Each inspection lot shall be manufactured on the same production line(s) through final seal by the same production techniques, and to the same device design rules and case with the same material requirements, and sealed within the same period not exceeding six weeks.

C. **Inspection Sublot** - An inspection sublot shall consist of microcircuits of a single type in a single package type, outline and lead finish, contained on a single detail specification, manufactured on the same production line(s) through final seal by the same production techniques, and to the same device design rules and package with the same material requirements, and sealed within the same period not exceeding six weeks.

D. **Microcircuit Group** - Microcircuits which are designed to perform the same type of basic circuit function, which are designed for the same supply, bias and signal voltages and for input/output compatibility and which are fabricated by use of the same basic die construction and metallization; the same die-attach method; and by use of bonding interconnects of the same size, material and attachment method.

E. **Percent Defective Allowable (PDA).** Percent defective allowable is the maximum observed percent defective which will permit the lot to be accepted after the specified 100 percent test.

F. **Delta (Δ) Limit.** The maximum change in a specified parameter reading which will permit a device to be accepted on a specified test, based on a comparison of the present measurement with a specified previous measurement. NOTE: When expressed as a percentage value, it shall be calculated as a proportion of the previous measured value.

G. **Rework.** Any processing or reprocessing operation, other than testing, applied to an individual device, or part thereof, and performed subsequent to the prescribed nonrepairing manufacturing operations which are applicable to all devices of that type at that stage.

H. **Final Seal.** That manufacturing operation which completes the enclosure of a device so that further internal processing cannot be completed without disassembling the device.

I. **Device Type.** The term device type refers to a single specific microcircuit configuration. Samples of the same device type will be electrically and functionally interchangeable with each other at the die level and environmental limits will be the same for a given device type even though the device class, the case outline, and the lead finish and the lot identification code may be different. A given type shall appear on only one device specification but that detail specification may also specify other similar devices.

3.2. Item Requirements. The individual item requirements for microcircuits delivered under this specification shall be documented in the detail specification or drawing. Unless otherwise specified, all microcircuits shall have an operating ambient temperature range from -55°C to +125°C and any reference to minimum or maximum operating temperatures shall refer to the respective lower and upper limits of this range.

3.3 Classification of Requirements. The requirements of the microcircuits are classified herein as follows:

REQUIREMENTS	PARAGRAPH
Product Assurance	3.4
Screening	3.4.3
Quality conformance inspection	3.4.4
Traceability	3.4.5
Design and construction	3.5
Marking	3.6
Workmanship	3.7

3.4. Product Assurance Requirements. Three levels of microcircuit quality and reliability assurance are provided for in this specification. MKB, MKM and MKX devices shall be those which have been subjected to, and passed all applicable requirements, tests, and inspections detailed herein, for the specified class. Where shown, method references are per MIL-STD-883. For general guidance, the following table summarizes these requirements for the respective device classes:

3.4 Cont. **SCREENING PER METHOD 5004 of MIL-STD-883**

TEST	MIL-STD-883A	CONDITION	MKB	MKM
Internal Visual	2010	Cond. B	100%	100%
Stabilization Bake	1008	24 Hrs @ +150°C	100%	100%
Temperature Cycling	1010	10 cycles min. –65°C to +150°C	100%	100%
Constant Acceleration	2001	30 KG Y_1 Plane Only	100%	100%
Seal Fine	1014	5 x 10^{-8} ATM-CM^3/SEC	100%	100%
Gross	1014	Condition C2	100%	100%
Pre-Burn-in Electrical 1/	Static & Dynamic		100%	100%
Burn-In	1015	MOSTEK Dynamic +160 hours minimum T_A = +125°C	100%	100%
Final Electrical Test		Static and Dynamic per Detail Spec	100% Max, 25°C and min rated Temp	100% Max Rated Temp
External Visual	2009		100%	100%
Quality Conformance	5005	Group A See Quality Conformance	Max.,25°C and Min. Rated Temp	7%LTPD 3/Max Rated Temp

1/ Manufacturer's Option

2/ Delete Subgroups 9, 10 and 11

3/ Subgroups 2, 5, and 8 combined at maximum rated temp.

NOTE: MKX is a custom flow, built per the customer drawing and may contain all, some or none of the above flow.

3.4.1. MIL-STD-883A QUALITY CONFORMANCE TEST

	TEST METHOD OF	LTPD IN	TEST CONDITIONS		
Group A (Each Lot)	MIL-STD-883	%	MKB Series	MKM Series	NOTES
Static Test	——		Per Detail Spec at min., 25°C and max temperature 1/	Max Rated Temp 2/	
Dynamic Test	5005				
Functional Test	——				
GROUP B (Each Lot)					
Physical Dimensions	2016	2 Dev	Per Detail Spec	Per Detail Spec	3, 4
Resistance to Solvents	2015	3 Dev	Marking Durability	Marking Durability	3, 5
Internal Visual	2014	2 Dev	Internal Construction Verification	Internal Construction Verification	3, 6
Bond Strength	2011	15	Condition D	Condition D	3, 7
Solderability	2003	15	260 ± 10°C	260 ± 10°C	3, 8
GROUP C (9/)					9
Operating Life	1005	5	+125°C 1000 Hrs.	+125°C 1000 Hrs	
End Point Electrical			Max Rated Temp	Max Rated Temp	
Temp Cycle	1010		−65°C to +150°C	−65°C to +150°C	
Constant Acceleration	2001	15	Cond. E	Cond. E	
Fine and Gross Leak	1014		5 x 10^{-8} ATM-CM^3/SEC	5 x 10^{-8} ATM-CM^3/SEC	
End Point Electrical			Max Rated Temp	Max Rated Temp	
GROUP D (9/)					9
Physical Dimensions	2016	15	Per Detail Spec	Per Detail Spec	
Lead Integrity	2004	15	Condition B2	Condition B2	
Fine and Gross Leak	1014	15	5 x 10^{-8} ATM-CM^3/SEC	5 x 10^{-8} ATM-CM^3/SEC	
Thermal Shock	1011	15	Condition B	Condition B	
Temperature Cycling	1010	15	Condition C -100 Cycles	Condition C -100 Cycles	
Moisture Resistance	1004	15	10 Cycles	10 Cycles	
Fine and Gross Leak	1014	15	5 x 10^{-8} ATM-CM^3/SEC	5 x 10^{-8} ATM-CM^3/SEC	
End Point Electrical			Max Rated Temp	Max Rated Temp	
Mechanical Shock	2002	15	Condition B	Condition B	
Vibration	2007	15	Condition A	Condition A	
Constant Acceleration	2001	15	Condition D	Condition D	
Fine and Gross Leak	1014	15	5 x 10^{-8} ATM-CM^3/SEC	5 x 10^{-8} ATM-CM^3/SEC	
End Point Electrical			Max Rated Temp	Max Rated Temp	
Salt Atmosphere	1009	15	Condition A	Condition A	

3.4.1. Cont:

(1/)Delete Subgroups 9, 10, 11 (2/)Subgroups 2, 5, and 8 at Maximum Rated Temp to a combined LTPD of 7% (3/) Electrical Rej. may be used (4/) 2 Devices from each lot will be tested. Accept on 0/, reject on 1. (5/) 3 Devices from each lot will be tested. Accept on 0/, reject on 1. (6/) 1 Device from lot will be tested. Accept on 0/, reject on 1. (7/)Test Sample may be pulled prior to sealing. (8/) Solderability sample must have seen time/temp exposure or burn-in. (9/) Group C and D tests will be performed "only" when specified on the Customer Purchase Order.

3.4.2 Change of Qualified Product. MOSTEK shall notify the customer prior to the implementation of any major change of the product or product assurance program which may affect performance, quality-reliability and interchangeability.

3.4.3. Screening. All microcircuits to be delivered in accordance with this specification shall have been subjected to, and passed, all the screening tests detailed in Paragraph 3.4 for the type of microcircuit and product assurance level (device class) specified. Sampling inspections shall not be an acceptable substitute for any specified screening test.

3.4.4. Quality Conformance Inspection. Microcircuits shall not be accepted or approved for delivery until the inspection lot has passed quality conformance inspection. (See 4.3.)

3.4.5. Tracability. See 3.1.2. (A)

3.5. Design and Construction. Microcircuit design and construction shall be in accordance with all the requirements specified herein and in the detail specification or drawing.

3.5.1. Package. All devices supplied under this specification shall be hermetically sealed in glass, metal or ceramic (or combinations of these) packages. No organic or polymetic materials (lacquers, varnishes, coatings, adhesives, greases, etc.) shall be used inside the microcircuit package, and no desiccants shall be contained in the microcircuit package unless otherwise specified. Polymer impregnations (backfill, docking, etc.) of the microcircuit packages shall not be permitted.

3.5.2. Metals. External metal surfaces shall be corrosion-resistant or shall be plated or treated to resist corrosion. External leads shall meet the requirements specified in 3.5.5.

3.5.3. Other Materials. External parts, elements or coatings including markings shall be inherently non-nutrient to fungus and shall not blister, crack, outgas, soften, flow or exhibit defects that adversely affect storage, operation or environmental capabilities of microcircuits delivered to this specification under the specified test conditions.

3.5.4 Internal Conductors. Internal thick film conductors on silicon die or substrate (metallization stripes, contact areas, bonding interfaces, etc.) shall be designed so that no properly fabricated conductor shall experience in normal operation (at worst case specified operating conditions), a current density in excess of the maximum allowable value shown below for the applicable conductor material:

CONDUCTOR MATERIAL	MAXIMUM ALLOWABLE CURRENT DENSITY
Aluminum (99.99% pure or doped) without glassivation	2 x 10^5 A/cm^2
Aluminum (99.99% pure or doped) glassivated	5 x 10^5 A/cm^2
Gold	6 X 10^5 A/cm^2
All other (unless otherwise specified)	2 x 10^5 A/cm^2

The current density shall be calculated at the point(s) of maximum current density (i.e. greatest current (see 3.5.5 (a)) per unit cross section) for the specific device type and schematic or configuration.

(a) Use a current value equal to the maximum continuous current (at a full fanout for digitals or at maximum load for linears) or equal to the simple time-averaged current

Military Hi-Rel

obtained at maximum rated frequency and duty cycle with maximum load, whichever results in the greater current value at the point(s) of maximum current density. This current value shall be determined at the maximum recommended supply voltage(s) and with the current assumed to be uniform over the entire conductor cross sectional areas.

3.5.5. Lead Material and Finish.

3.5.5.1. Lead Material. Lead material shall conform to one of the following chemical compositions:

A. Type A

Iron	53 percent, nominal
Nickel	29 ± 1 percent
Cobalt	17 ± 1 percent
Manganese	0.65 percent, maximum
Carbon	0.06 percent, maximum
Silicon	0.20 percent, maximum
Aluminum	0.10 percent, maximum
Magnesium	0.10 percent, maximum
Zirconium	0.10 percent, maximum
Titanium	0.10 percent, maximum

(Combined total of aluminum, magnesium, zirconium and titanium to be a maximum of 0.20 percent).

B. Type B

Nickel	40-43 percent
Manganese	0.08 percent, maximum
Silicon	0.30 percent, maximum
Carbon	0.10 percent, maximum
Chronium	0.25 percent, maximum
Cobalt	0.50 percent, maximum
Phosphorous	0.025 percent, maximum
Sulfur	0.025 percent, maximum
Aluminum	0.10 percent, maximum
Iron	Remainder

3.5.5.2. Lead Finish. Lead finish shall conform to one of the following as applicable.

A. Hot solder dip - The hot solder dip shall be homogeneous with a minimum thickness at the crest of the major flats of 200 microinches (50.8 nm) of solder (SN60 to SN63) over the preliminary finishes in accordance with (b) or (c) below or over nickel plate with a plating thickness of 100 microinches (25.4 nm) minimum and 200 microinches (50.8 nm) maximum.

B. Bright acid tin plate - Thickness of 100 microinches (25.4 nm) minimum and 400 microinches (101.6 nm) maximum. Optional electroless or electrolytic nickel or copper underplating, if used, shall be a minimum of 10 microinches (25.4 nm) in thickness. NOTE: It is recognized that "bright acid tin plate", a term which refers to the process as well as the appearance, can yield a range of texture or reflectivity. It is intended that this finish be dense and continuous and that it will meet the solderability and environmental requirements of this specification.

C. Gold plate - Gold plating shall be a minimum of 99.7 percent gold (0.3 percent maximum for all impurities and other metals combined). Gold plating shall be a minimum of 50 microinches (12.7 nm) and a maximum of 225 microinches (57.4 nm) thick. Optional electroless or electrolytic nickel or copper underplating, if used, shall be a minimum of 10 microinches (2.54 nm) and a maximum of 100 microinches (25.4 nm) in thickness.

3.5.6. Die Thickness. Unless otherwise specified, the minimum die thickness for all microcircuits shall be 0.006 inch (.15 mm).

Military
Hi-Rel

3.6. Marking of Microcircuits. Marking shall be in accordance with the requirements of this specification, and the identification and marking provisionsof the detail specification or drawing. The marking shall be legible, and complete and shall meet the resistance to solvents requirements of MIL-STD-883, Method 2015. If any special marking is used, it shall in no way interfere with the marking required herein, and shall be visibly separated therefrom. The following marking shall be placed on each microcircuit unless otherwise specified:

A. Index point (3.6.1)
B. Part number
C. Inspection lot identification code (3.6.2)
D. Manufacturer's identification
E. Country of origin (3.6.3)
F. Serialization, when applicable (3.6.4)

3.6.1. Index Point. The index point, tab or other marking indicating the starting point for numbering of leads or for mechanical orientation shall be as specified.

3.6.2. Inspection Lot Identification Code. Microcircuits shall be marked by a unique code to identify the inspection lot (see 3.1.3 (b) and 3.1.3 (c)) and identify the first or the last week of the period (six weeks maximum) during which devices in that inspection lot were sealed. The first two numbers in the code shall be the last two digits of the number of the year, and the third and fourth numbers shall be two digits indicating the calendar week of the year. When the number of the week is a single digit, it shall be preceded by a zero. Reading from left to right or from top to bottom, the code number shall designate the year and week, in that order. When more than one lot of a type is to be identified within the same week, an inspection lot identification suffix letter, representing each lot identified during that week and lettered uniquely shall appear on each microcircuit immediately following the inspection lot data code so that each inspection lot is identified by the inspection lot date code and by the lot identification suffix letter, if one is required.

3.6.3. Country of Origin. The phrase "Made in U.S.A." shall be marked in small characters below or adjacent to the other marking specified, except that for microcircuits made in a foreign country the phrase shall be changed accordingly. If there is limited space, the marking may be shortened to "U.S.A." or to the appropriate accepted abbreviation for the country of origin.

3.6.4. Serialization. Prior to the first recorded electrical measurement in screening, when specified, each microcircuit shall be marked with a unique serial number assigned consecutively within the inspection lot. This serial number allows traceability of test results down to the level of the individual microcircuit within that inspection lot.

3.6.5. Marking Location and Sequence. Unless otherwise specified, the part number, inspection lot identification code, and serialization (where applicable), shall be located on top surface of flat packages or dual-in-line configurations and on either the top or side of cylindrical packages (TO-96 and similar configurations). The index point shall be marked as specified. The balance of the markings may be placed in any suitable location so as to perform their required functions and not interfere with the other markings.

3.6.6. Marking on Initial Container. All of the markings specified in 3.6, except the index point and serialization shall appear on the initial protection or wrapping for delivery (container, carton, box, plastic envelope, etc.) and this marking shall be in accordance with MIL-STD-129.

3.6.7. Marking Option for Controlled Storage. Where microcircuits are subjected to testing and screening in accordance with some portion of the product assurance requirements and stored in controlled storage areas pending receipt of orders requiring conformance to the same or a different level, the inspection lot identification code shall be placed on the microcircuit package along with the other markings specified in 3.6 sufficient to assure identification of the material. As an alternative, if the microcircuits are stored together with sufficient data to assure traceability to processing and inspection records, all markings may be applied after completion of all inspections to the specified level.

3.6.8. Marking Procedure Option. MOSTEK has the option of marking the entire lot or only the sample devices to be submitted to qualification or Groups B, C, and D quality conformance inspection as applicable. If the manufacturer exercises the option to mark only the sample devices, the procedures shall be as follows:

A. The sample devices shall be marked prior to performance of Groups B, C and D quality conformance inspections, as applicable.

B. At the completion of inspection, the marking of the sample devices shall be inspected for conformance with the requirements of 3.6.

C. The inspection lot represented by a conforming inspection sample shall then be marked and any specified visual and mechanical inspection performed.

D. The marking materials and processing applied to the inspection lot shall be to the same specifications as those used for the inspection sample.

3.7. Workmanship. Microcircuits shall be manufactured, processed, and tested in a careful and workmanlike manner in accordance with good engineering practice and with the requirements of this specification.

3.7.1. Rework Provisions. All rework permitted on microcircuits procurred under this specification shall be accomplished in accordance with procedures and safeguards documented and available for review. No delidding or package opening for rework shall be permitted for microcircuits of any class. Allowable rework of sealed packages includes recleaning of any microcircuit or portion thereof, rebranding to correct defective marking and lead straightening (provided the reworked devices meet the requirements of 4.6.2 for conditions of leads).

3.7.1.1. Rebonding of Monolithic Devices. Unless otherwise specified, rebonding of monolithic microcircuits shall be permitted with the following limitations:

A. No scratched, open or discontinuous metallization paths or conductor patterns shall be repaired by bridging with or addition of bonding wire or ribbon.

B. All rebonds shall be placed on at least 50% undisturbed metal and no more than one rebond attempt at any design bond location shall be permitted at any pad or post and no rebonds shall be made directly over an area where metallization of intended bond areas has been lifted.

C. The total number of rebond attempts shall be limited to a maximum of 10 percent of the total number of bonds in the microcircuit. The 10 percent limit on rebonds may be interpreted as the nearest whole number of bonds in the microcircuit. A bond shall be defined as a wire to post or wire to pad bond (i.e. for a 14 lead wire bonded package there are 28 bonds). Bond-offs required to clear the bonder after an unsuccessful first bond attempt need not be considered as rebonds provided they can be identified as bond-offs by being made physically off the plated post or if they contain a non-typical number of wedge marks. The initial bond attempt need not be visible. A rebond attempt at one end of the wire counts as one rebond; a replacement of a wire bonded at both ends, or an unsuccessful bond attempt of a wire already bonded at the other end, counts as two rebonds. A bond on top of another bond is not permissible.

4.0 PRODUCT ASSURANCE PROVISIONS

4.1. Safety Requirements - Not Applicable

4.1.1. Responsibility for Tests and Inspections. Unless otherwise specified in the contract or purchase order, MOSTEK is responsible for the performance of all tests and inspection requirements as specified herein and in the detail specification. Except as otherwise specified in the contract or order, the manufacturer may use his own or other suitable facilities.

4.1.2. Inspection During Manufacture. MOSTEK shall establish and maintain inspection at appropriately located points in the manufacturing process in accordance with the procedures described in 20.1.1 of Appendix A of MIL-M-38510 to assure continuous control of quality of materials, subunits and parts during fabrication and testing. This inspection shall be adequate to assure compliance with the applicable procurement documentation and quality standards for microcircuits manufactured to this specification and the applicable detail specification.

4.1.3. Control and Inspection of Procurement Sources. MOSTEK shall be responsible for assuring that all supplies and services used in the manufacture and test of microcircuits conform to all the requirements of this specification, the detail specification, and other provisions of the applicable procurement documentation.

4.1.4. Inspection Records.

4.1.4. Inspection Records. MOSTEK shall maintain adequate records of all examinations, inspections, and tests accomplished in accordance with 4.0. Records shall be retained as specified in 20.1.2 of Appendix A of MIL-M-38510.

4.2. General Inspection Conditions. The general requirements of MIL-STD-883 shall apply.

4.2.1. Classification of Examinations and Tests. The examinations and tests required to assure conformance to the specified product assurance levels of microcircuits or lots thereof are classified as follows:

Requirement	**Paragraph**
Quality Conformance Inspection	4.3
Screening	4.4
Data reporting	4.6

4.2.2. Sampling. Statistical sampling for quality conformance inspections shall be in accordance with the sampling procedures of appendix B of MIL-M-38510, and as specified in the detail specification or drawing, as applicable. Reserve sample devices may be tested with the subgroups to provide replacements in the case of test equipment failure or operator error. These devices shall be used in predesignated order.

4.2.2.1. Disposal Of Samples. Devices subjected to destructive tests or which fail any test shall not be shipped on the contract or purchase order as acceptable product. They may, however, be delivered at the request of the procuring activity if they are isolated from, and clearly identified so as to prevent their being mistaken for acceptable product. Sample microcircuits, form lots which have passed product assurance inspections or tests and which have been subjected to mechanical or environmental tests specified in Groups B, C and D inspection and not classified as destructive, may be shipped on the contract or purchase order provided the test has been proved to be nondestructive (see 4.2.2.3) and each of the microcircuits subsequently passes final electrical tests per the applicable device specification.

4.2.2.2. Destructive Tests. Unless otherwise specified, the following MIL-STD-883 tests shall be classified as destructive:

Internal visual and mechanical (Method 2014)
Bond strength.
Solderability.
Moisture resistance.
Lead integrity.
Salt atmosphere.
SEM inspection for metallization.
Steady state life test (accelerated).
Die shear strength test.

All other mechanical or environmental tests (other than those listed in 4.2.2.3), shall be considered destructive initially, but may subsequently be considered nondestructive. The accumulation of data from five repetitions of the specified test on the same sample of product, without evidence of cumulative degradation or failure to pass the specified test requirements in any microcircuit in the sample, is considered sufficient evidence that the test is nondestructive. Any test specified as a 100 percent screen shall be considered nondestructive for the stress level and duration or number of cycles applied as a screen.

4.2.2.3. Nondestructive Tests. Unless otherwise specified, the following tests are classified as nondestructive:

Barometric pressure
*Steady state life
*Intermittent life
Seal
External visual
Internal visual (pre-cap)
*Burn-in screen
Radiography

*When the test temperature exceeds the maximum specified junction temperature for the device (including maximum specified for operation or test), these tests shall be considered destructive.

4.2.3. Formation of Lots. Microcircuits shall be segregated into identifiable production lots as defined in 3.1.3(a) as required to meet the production control and inspection requirements of Appendix A of MIL-M-38510. Microcircuits shall be assembled into inspection lots as defined in 3.1.3(b) and 3.1.3(c) as required to meet the product assurance inspection and test requirements of this specification.

4.2.3.1. Resubmission of Failed Lots. Resubmitted lots shall be kept separate from new lots and shall be clearly identified as resubmitted lots. When any lot submitted for quality conformance inspection fails any subgroup requirement of group A, B, C or D tests, it may be resubmitted once for that particular subgroup using tightened inspection criteria (as defined in 30.2.6 of Appendix B of MIL-M-38510). A second resubmission using tightened inspection criteria is permitted only if failure analysis is performed to determine the mechanism of failure for each failed microcircuit from the prior submissions and it is determined that failure(s) is due to:

A. A defect that can be effectively removed by rescreening the entire lot, or
B. Random type defects which do not reflect poor basic device design or poor basic processing procedures.
C. Testing errors resulting in electrical damage to the device.

In all instances where analysis of the failed devices indicates that the failure mechanism is due to poor basic processing procedures, a basic design fault or non-screenable defects, the lot shall not be resubmitted.

4.2.4. Test Method Deviation. Deviations from test methods or tests circuits specified are allowed provided that such deviations in no way relax the requirements of this specification.

4.2.5. Procedure in Case of Test Equipment Failure or Operator Error. Whenever a microcircuit is believed to have failed as a result of faulty test equipment or operator error, the failure shall be entered in the test record which shall be retained for review along with a complete explanation verifying why the failure is believed to be invalid.

4.2.5.1. Procedure for Sample Tests. When it has been established that a failure is due to test equipment failure or operator error and it has been established that the product has not been damaged or degraded, a replacement microcircuit from the same inspection lot may be added to the sample. The replacement microcircuit shall be subject to all those test to which the discarded microcircuit was subjected prior to its failure and to any remaining specified tests to which the discarded microcircuit was not subjected prior to its failure. The manufacturer, at his own risk, has the option of replacing the failed microcircuit and continuing with the tests before the validity of the test equipment failure or operator error has been established.

4.2.5.2. Procedure for Screening Tests. When it has been established that a lot failure(s) during the screening test(s) is due to operator or equipment error and it has been established that the remaining product has not been damaged or degraded, the lot or surviving portion of the lot, as the case may be, may be resubmitted to the corrected screening test(s) in which the error occurred. Failures verified as having been caused by test equipment failure or operator error shall not be counted in the PDA calculation (when applicable).

4.3. Quality Conformance Inspection.

4.3.1. General. Quality conformance inspection shall be conducted in accordance with the applicable requirements of Groups A, B, C and D of Method 5005, MIL-STD-883, for the specified device class. (See 3.4.1.)

4.3.2. Group A Inspection. Group A inspection shall be performed on each inspection lot in accordance with Method 5005 of MIL-STD-883 and shall consist of electrical parameter tests specified for the specified device class. Group A inspection may be performed in any order. If an inspection lot is made up of a collection of sublots, each sublot shall pass Group A inspection as specified.

4.3.3. Group B Inspection. Group B inspection shall be performed on each inspection lot, for each different package type (i.e. case outline, materials and lead finish), on each different device specification. Group B shall consist of mechanical and environmental tests in accordance with Method 5005 of MIL-STD-883 for the specified device class. Testing of one device type sublot in any subgroup shall be considered as complying with the requirements for that subgroup for all types in the inspection lot. Different device types may be used for each subgroup. A different device type sublot shall be tested for subgroup 2 at each successive Group B inspection until all qualified device types on that detail specification, being submitted for acceptance, have been tested. Except as otherwise specified, this inspection shall be applied only to completed and fully marked devices from lots which have been subjected to and passed the Group A tests.

4.3.3.1. Group B Sample Selection. Samples for Group B subgroups shall be chosen at random from any sublot which has completed the screening requirements of paragraph 4.4 and been submitted to quality conformance inspection (see 30.1.1 of Appendix B of MIL-M-38510).

4.3.4. Group C Inspection. Group C inspection (die related tests) shall be in accordance with Method 5005 of MIL-STD-883 and shall include those tests specified which are performed periodically. Group C tests shall be performed only when specified on the Purchase Order.

4.3.4.1. Group C Sample Selection. Samples for subgroups in Group C shall be chosen at random from any inspection lot of a particular microcircuit group which is submitted to and passes Group A tests for quality conformance inspection during the week in which the first lot of that microcircuit group is submitted in each specified Group C inspection period. Samples from the lot may be subjected to Group C inspection whether or not the specified inspection lot has passed Group B quality conformance inspection. Testing of one device type for each subgroup shall be considered as complying with the requirements for that subgroup for all types on the detail specification(s) within that same microcircuit group. A different device type shall be tested at each successive inspection interval until all device types qualified on the detail specification(s) with the microcircuit group have been tested.

When none of the lots passing Group A during the week in which the first lot is submitted contains the devices type which is due to be tested, the samples for inspection shall be chosen from those types in the lot being tested which have not been used for the longest time for Group C die-related inspection. The next lot submitted which contains the skipped type shall be subjected to Group C inspection as part of its quality conformance inspection. Successful completion of Group C inspection shall initiate a new Group C die-related inspection period. For nonconformance see 4.3.7.

4.3.5. Group D Inspection. Group D inspection (package-related tests) shall be in accordance with Method 5005 of MIL-STD-883 and shall include those package or case-related tests which are performed periodically. The group D tests shall be performed periodically as specified on the Purchase Order for each different package, case or construction.

4.3.5.1. Group D Sample Selection. Samples for subgroups in Group D shall be chosen at random from any inspection lot containing the intended package, case or construction which is submitted to and passes Group A tests for quality conformance inspection during the week in which the first lot containing the intended package is submitted in each Group D inspection period. Testing of a subgroup using a single device type enclosed in the intended package shall be considered as complying with the requirements for that subgroup for all detail specifications utilizing that package. Different types from the inspection lot may be used for each subgroup. Testing of different types on a rotation basis is not required. Successful completion of Group D inspection shall initiate a new group D package-related inspected period. For non conformance see 4.3.7.

4.3.6. End Point Tests for Groups C and D Inspection. Specified post-test parameters shall be measured for each microcircuit for the sample after completion of all other specified tests in the subgroup. Additional measurements may be made at the discretion of the manufacturer.

At the end of each Group C and D subgroup, end point measurements shall include visual examination without magnification to assure marking on each microcircuit tested is legible and complete (see 3.6). Damage to marking caused by mechanical fixturing or handling during tests shall not be cause for lot rejection, but devices so damaged shall be individually remarked or shall be rejected for shipment.

4.3.7. Nonconformance. Samples which fail subgroup requirements of Groups A, B, C, or D may be resubmitted in accordance with the provisions of 4.2.3.1. However, if the lot is not resubmitted or fails the resubmission of 4.2.3.1 the lot shall not be shipped. Samples from subsequent lots of the same microcircuit group for Group C failures or the same package type for Group D failures shall then be subjected to all the tests in the subgroup in which the failure occurred, on a lot-by-lot basis until three successive lots pass the failed subgroup. The testing may then return to periodic testing. A device type which fails a Group C inspection shall not be accepted until the device type which failed successfully completes the failed Group C subgroup(s). No other device types in the group represented by the failed device type may be accepted until the Group C inspection requirements have been satisfied with a device type in the group. A package type which fails a Group D inspection shall not be accepted until the package type which failed successfully completes the failed Group D inspection subgroup(s).

4.4 Screening. Each microcircuit shall have been subjected to and passed all the screening tests detailed in Paragraph 3.4. for the specified product assurance level and type of microcircuit in order to be acceptable for delivery. When a PDA (see 3.1.2(c) or delta limits (see 3.1.2(f)) have been specified or other conditions for lot acceptance have been imposed, the required data shall be recorded and maintained as a basis for lot acceptance. Devices which fail any test criteria in the screening sequence shall be removed from the lot at the time of observation or immediately at the conclusion of the test in which the failure was observed. Once rejected and verified as a device failure, no device may be retested for acceptance.

4.4.1. Burn-in. Burn-in shall be performed on all microcircuits where specified and the specified pre- and post-burn-in electrical parameters shall be measured.

4.4.1.1. Lots Resubmitted for Burn-In. Unless otherwise specified, lots may be resubmitted for burn-in one time only and may be resubmitted only when the observed percentage of parts which were in the original lots. Resubmitted lots shall be kept separate from new lots and shall be inspected for all specified characteristics using a tightened inspection PDA equal to the next lower number in the LTPD series.

4.5. External Visual Screen. The final external visual screen shall be conducted in accordance with Method 2009 of MIL-STD-883 after all other 100 percent screens have been performed to determine that no damage to, or contamination of the package exterior has occurred.

4.6 Data Recording. The results of all quality conformance tests and inspections and the results of all required failure analyses shall be recorded and maintained in the manufacturer's facility. The disposition of all lots or samples submitted for screening (where PDA is specified), or quality conformance inspection shall be fully documented and lots which fail any specified requirement shall be recorded at failed lots whether resubmitted or withdrawn. Disposition of resubmitted lots shall likewise be recorded so that a complete history is available for every lot tested from initial submission to final disposition including all failures, resubmissions and withdrawals.

4.7. Inspection of Preparation for Delivery. Sample packages and packs shall be selected and inspected in accordance with MIL-M-55565, or as specified in the contract or order.

5.0 PREPARATION FOR DELIVERY.

5.1 Preservation-packaging and Packing. Microcircuits shall be prepared for delivery in accordance with preservation-packaging and packing conforming to Level A, B or C requirements of MIL-M-55565 unless otherwise specified in the procurement document (see 6.1(e)).

5.1.1. Packaging and Packing. No packaging or packing material that is used shall crumble, flake, powder or shred. The cushioning material near or in contact with the microcircuits shall not be fibrous in form which might cause the microcircuit leads to be caught and damaged upon removal. Individual microcircuits shall be separated from all others, physically restrained from vibration and mechanically isolated from shock that might cause damage or degradation to the part. Leads must be supported to prevent vibration and retain their shape and position.

5.1.2. Unit Container. When specified on the detail drawing or purchase order (see 6.1(e)) individual microcircuits shall be supplied mounted in the specified carrier or unit container. Leads must be secured to protect against vibration and retain their shape. Marking on the unit pack, carrier container, or initial contained shall be as specified in 3.6.10.

6.0 NOTES.

6.1 Ordering Data. Procurement documents should specify the following:

A. Part number.
B. Title, number and date of this specification.
C. Title, number and date of applicable detail specification or drawing and identification or the originating design activity.
D. Test data to be furnished.
E. Selection of applicable level of packaging and packing required (see 5.0). Specification of unit container, when applicable (see 5.1), and special marking when applicable.
F. Product assurance level and product assurance options, when applicable (see 3.4)
G. Design documentation to be furnished (see 3.5.4)
H. Lead finish letter when required (see 3.5.5)
I. Requirements for failure analysis, when applicable.
J. Requirements for notification of change (see 3.4.2) to the procuring activity, when applicable.

6.2. Re-evaluation of Lot Quality. The specified LTPD method is designed for source inspection and provides a high degree of assurance that a lot has a proportion defective no greater than the specified LTPD value. Re-evaluation of any given lot to the same LTPD and acceptance number has the net effect of increasing the probability of rejection or the manufacturer's risk. This is especially true when the initial sampling plan is based on a low acceptance number or when lot re-evaluation is done using a lower acceptance number than was used in the initial sampling plan. Table B-I of Appendix A of MIL-M-38510 provides examples of the approximate quality levels required to satisfy any selected sampling plan. To minimize the effect of re-evaluation on the manufacturer's risk, whenever the quality of a lot is re-evaluated by sampling inspection subsequent to the manufacturer's demonstration of compliance with the quality requirements, the sampling plan shall be based on the next higher acceptance number (for the same LTPD) above that used in the initial lot evaluation. If the initial acceptance number is not known, or if the original inspection was conducted as a screening or 100 percent inspection, then the lot being re-evaluated shall not be rejected using an acceptance number of less than 3. Lots may, however, be accepted on re-evaluation using an acceptance number as low as 0. When deemed necessary, the purchase order may specify detailed criteria for lot re-evaluation and disposition other than the above. Government sources inspection procedures or resubmission of failed lots shall not be considered as re-evaluation of lot quality but rather as a part of the initial quality conformance procedure.

MIL STD 105D

TABLE I
Sample size code letters

Lot or batch size	General inspection levels I	NORMAL LEVEL II	III
2 to 8	A	A	B
9 to 15	A	B	C
16 to 25	B	C	D
26 to 50	C	D	E
51 to 90	C	E	F
91 to 150	D	F	G
151 to 280	E	G	H
281 to 500	F	H	J
501 to 1200	G	J	K
1201 to 3200	H	K	L
3201 to 10000	J	L	M
10001 to 35000	K	M	N
35001 to 150000	L	N	P
150001 to 500000	M	P	Q
500001 and over	N	Q	R

TABLE II-A
Single sampling plans for normal inspection (Master table)

Sample size code letter	Sample size	Acceptable Quality Levels (normal inspection)																	
		.010	.015	.025	.040	.065	0.10	0.15	0.25	0.40	0.65	1.0	1.5	2.5	4.0	6.5	10	15	25
		Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re	Ac Re
A	2	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	0 1	↓	↓	1 2
B	3	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	0 1	↑	↓	1 2	2 3
C	5	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	0 1	↑	↓	1 2	2 3	3 4
D	8	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	0 1	↑	↓	1 2	2 3	3 4	5 6
E	13	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8
F	20	↓	↓	↓	↓	↓	↓	↓	↓	↓	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8	10 11
G	32	↓	↓	↓	↓	↓	↓	↓	↓	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8	10 11	14 15
H	50	↓	↓	↓	↓	↓	↓	↓	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8	10 11	14 15	21 22
J	80	↓	↓	↓	↓	↓	↓	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8	10 11	14 15	21 22	↑
K	125	↓	↓	↓	↓	↓	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8	10 11	14 15	21 22	↑	↑
L	200	↓	↓	↓	↓	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8	10 11	14 15	21 22	↑	↑	↑
M	315	↓	↓	↓	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8	10 11	14 15	21 22	↑	↑	↑	↑
N	500	↓	↓	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8	10 11	14 15	21 22	↑	↑	↑	↑	↑
P	800	↓	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8	10 11	14 15	21 22	↑	↑	↑	↑	↑	↑
Q	1250	0 1	↑	↓	1 2	2 3	3 4	5 6	7 8	10 11	14 15	21 22	↑	↑	↑	↑	↑	↑	↑
R	2000	↑	↑	1 2	2 3	3 4	5 6	7 8	10 11	14 15	21 22	↑	↑	↑	↑	↑	↑	↑	↑

↓ = Use first sampling plan below arrow. If sample size equals, or exceeds, lot or batch size, do 100 percent inspection
↑ = Use first sampling plan above arrow.

Re = Rejection number. Ac = Acceptance number.

TABLE C—1. LTPD sampling plans 1/2/

Minimum size of sample to be tested to assure, with a 90 percent confidence, that a lot having percent-defective equal to the specified LTPD will not be accepted (single sample).

Max. Percent Defective (LTPD) or λ	20	15	10	7	5	3	2	1.5	1	0.7	0.5	0.3
Acceptance Number (C) (r = c + 1)	Minimum Sample Sizes (For device-hours required for life test, multiply by 1000)											
0	11 (0.46)	15 (0.34)	22 (0.23)	32 (0.16)	45 (0.11)	76 (0.07)	116 (0.04)	153 (0.03)	231 (0.02)	328 (0.02)	461 (0.01)	767 (0.007)
1	18 (2.0)	25 (1.4)	38 (0.94)	55 (0.65)	77 (0.46)	129 (0.28)	195 (0.18)	258 (0.14)	390 (0.09)	555 (0.06)	778 (0.045)	1296 (0.027)
2	25 (3.4)	34 (2.24)	52 (1.6)	75 (1.1)	105 (0.78)	176 (0.47)	266 (0.31)	354 (0.23)	533 (0.15)	759 (0.11)	1065 (0.080)	1773 (0.045)
3	32 (4.4)	43 (3.2)	65 (2.1)	94 (1.5)	132 (1.0)	221 (0.62)	333 (0.41)	444 (0.31)	668 (0.20)	953 (0.14)	1337 (0.10)	2226 (0.062)
4	38 (5.3)	52 (3.9)	78 (2.6)	113 (1.8)	158 (1.3)	265 (0.75)	398 (0.50)	531 (0.37)	798 (0.25)	1140 (0.17)	1599 (0.12)	2663 (0.074)
5	45 (6.0)	60 (4.4)	91 (2.9)	131 (2.0)	184 (1.4)	308 (0.85)	462 (0.57)	617 (0.42)	927 (0.28)	1323 (0.20)	1855 (0.14)	3090 (0.085)
6	51 (6.6)	68 (4.9)	104 (3.2)	149 (2.2)	209 (1.6)	349 (0.94)	528 (0.62)	700 (0.47)	1054 (0.31)	1503 (0.22)	2107 (0.155)	3509 (0.093)
7	57 (7.2)	77 (5.3)	116 (3.5)	166 (2.4)	234 (1.7)	390 (1.0)	589 (0.67)	783 (0.51)	1178 (0.34)	1680 (0.24)	2355 (0.17)	3922 (0.101)
8	63 (7.7)	85 (5.6)	128 (3.7)	184 (2.6)	258 (1.8)	431 (1.1)	648 (0.72)	864 (0.54)	1300 (0.36)	1854 (0.25)	2599 (0.18)	4329 (0.108)
9	69 (8.1)	93 (6.0)	140 (3.9)	201 (2.7)	282 (1.9)	471 (1.2)	709 (0.77)	945 (0.58)	1421 (0.38)	2027 (0.27)	2842 (0.19)	4733 (0.114)
10	75 (8.4)	100 (6.3)	152 (4.1)	218 (2.9)	306 (2.0)	511 (1.2)	770 (0.80)	1025 (0.60)	1541 (0.40)	2199 (0.28)	3082 (0.20)	5133 (0.120)

1/ Sample sizes are based upon the Poisson exponential binomial limit.

2/ The minimum quality (approximate AQL) required to accept (on the average) 19 of 20 lots is shown in parenthesis for information only.

1979 MICROCOMPUTER DATA BOOK

Develop Systems US

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

AID-80F

INTRODUCTION

The Mostek AID-80F* is a complete state-of-the-art, floppy disk-based computer. Not only does it provide all the necessary tools for software development, but it provides complete hardware/software debug through Mostek's AIM* series of in-circuit emulation cards for the Z80 as well as the 3870 family of single chip microcomputers. The AID-80F has at its heart the powerful OEM-80 (Single Board Computer), RAM-80 (RAM (I/O add on board), and the FLP-80 (floppy controller board). Because these boards and software are available separately to OEM users, the AID-80F serves as an excellent test bed for developing systems applications.

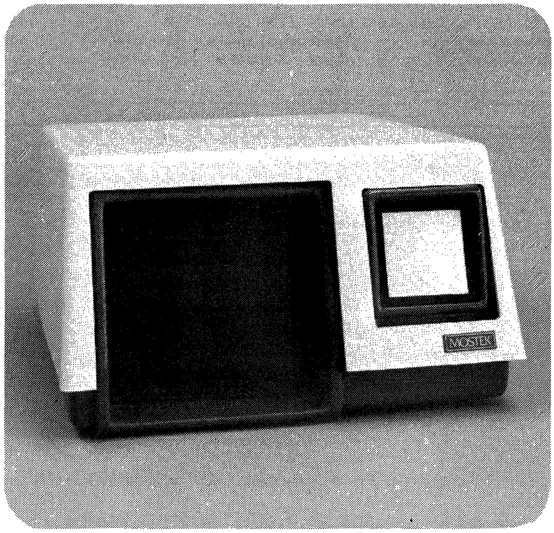
The disk-based system eliminates the need for other mass storage media and provides ease of interface to any peripheral normally used with computers. The file-based structure for storage and retrieval consolidates the data base and provides a reliable portable media to speed and facilitate software development.

The FLP-80DOS Disk Operating System is designed for maximum flexibility both in use and expansion to meet a multitude of end user or OEM needs.

Development System Features

The AID-80F is an excellent integration of both hardware and software development tools for use throughout the complete system design and development phase. The software development is begun by using the combination of Mostek's Text Editor with "roll in-roll out" virtual memory operation and the Mostek relocating assembler. Debug can then proceed inside the AID-80F domain using its resources as if they were in the final system. Using combinations of the Monitor, Designer's Debugging Tool, execution time breakpoints, and single step/multistep operation along with a formatted memory dump provides control for attacking those tough problems. The use of the Mostek AIM-80 option provides extended debug with versatile hardware breakpoints on memory or port locations, a buffered in-circuit emulation cable for extending the software debug into its own natural hardware environment, and a 256x32 history memory to capture bus transactions in real time for later examination.

The relocatable and linking feature of the assembler enables the use of contemporary modular design techniques whereby major system alterations can be made in small tractable modules. Using the Linker, the



small modules can be combined to form a run-time module without major reassembly of the entire program.

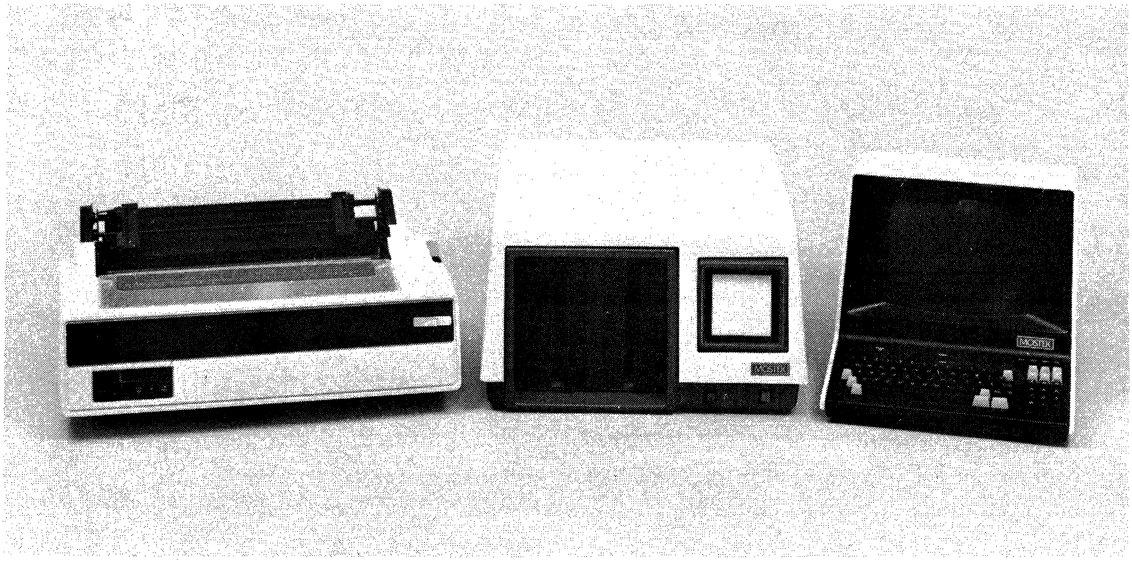
Packaged System Features

From a system standpoint, the AID-80F has been designed to be the basis of an end-product, small business/industrial computer. The flexibility provided in the FLP-80DOS operating system permits application programs to be as diverse as a high-level language compiler to a supervisory control system in the industrial environment. Other hardware options are available, with even more to be added. Expansion of the disk drive units to a total of four single-sided or double-sided units provides up to two megabytes of storage. This computer uses the third generation Z80 processor supported with the power of a complete family of peripheral chips. Through the use of its 158 instructions, including 16-bit arithmetic, bit manipulation, advanced block moves and interrupt handling, almost any application from communication concentrators to general purposed accounting systems is made easy.

OEM Features

The hardware and software basis for the AID-80F is also available separately to the OEM purchaser. Through a software licensing agreement, all Mostek Software can be utilized on these OEM series of cards. A growing line

*Trademark of Mostek Corporation



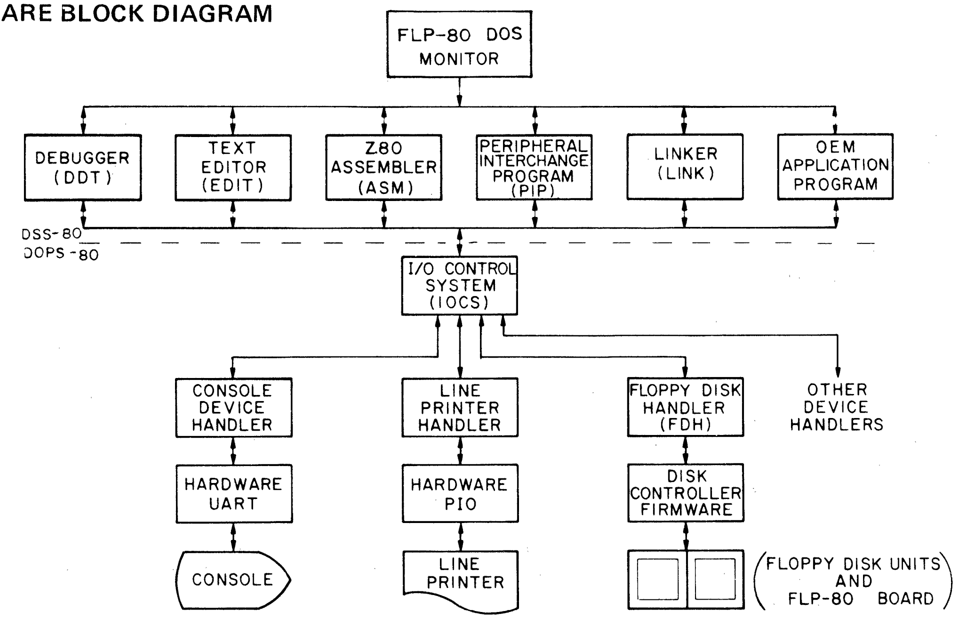
of support cards and card cages permits the user to configure a multitude of different systems.

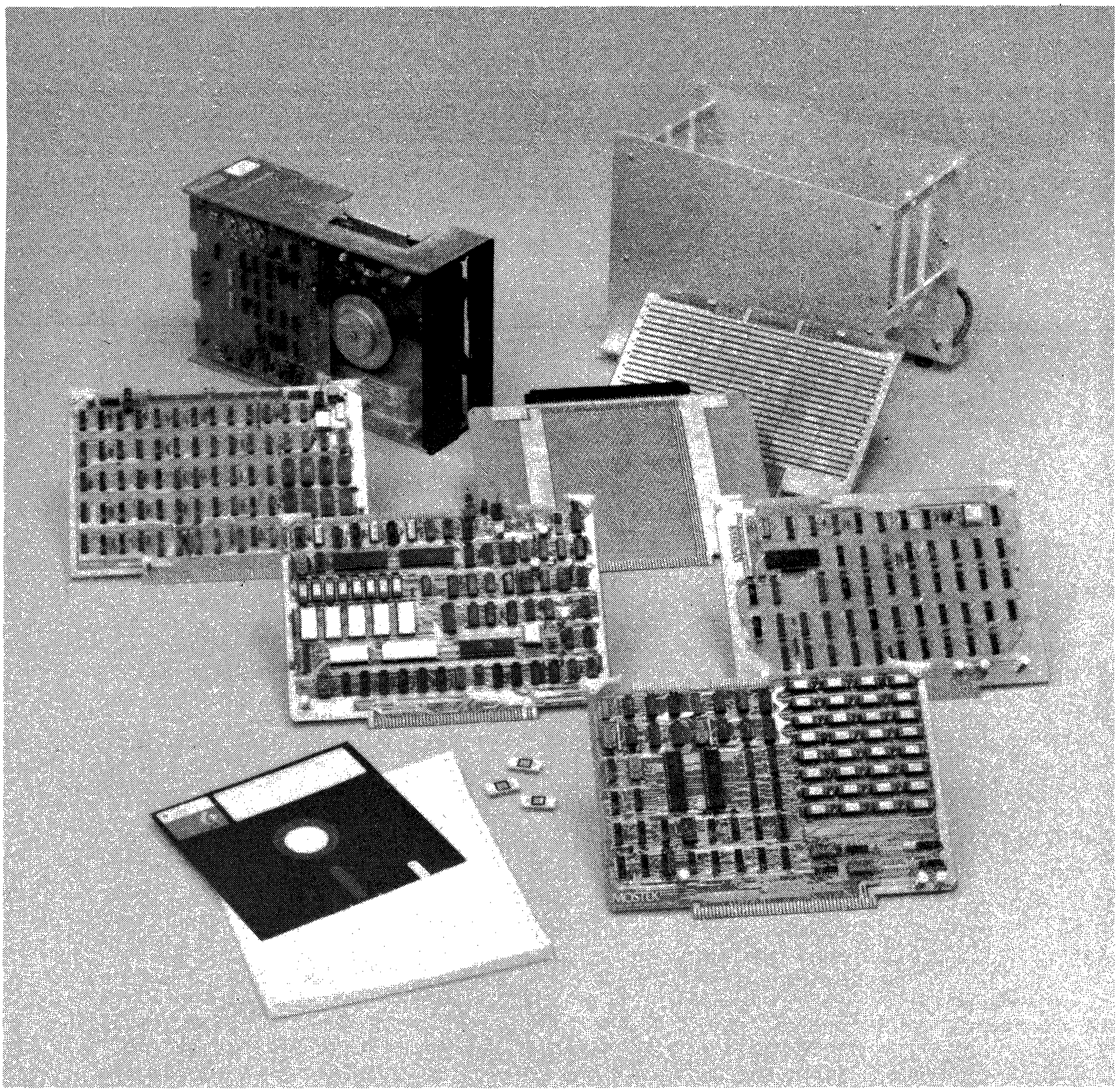
AID 80F RESIDENT SOFTWARE (FLP-80DOS)

A totally integrated package of resident software is offered in conjunction with the AID-80F consisting of:

- Monitor
- DDT-80 with extended debug through AIM-80
- Text Editor
- Z80 Relocating Assembler
- Peripheral Interchange Program
- Linker
- I/O Control System
- Floppy Disk Handler
- Device Driver Library
- Batch Mode Operation

SOFTWARE BLOCK DIAGRAM





Monitor

The FLP-80DOS Monitor is the environment from which all activity in the system initiates. From the Monitor, any system routine such as PIP or a user-generated program is begun by simply entering the program name. FLP-80DOS I/O is done in terms of logical unit numbers, as is commonly done in FORTRAN. A set of logical units is pre-assigned to default I/O drivers upon power up or reset. From the console the user can reassign any logical unit to any new I/O device and can also display logical unit assignments. Executable file creation can be done by the Save command as well as printable absolute object files can be produced using the Dump command.

Text Editor

The Text Editor permits editing/creating of any source file independent of the language being written. The Editor is both line and string oriented to give maximum utility and user flexibility. The Editor, through its virtual memory "roll in-roll out" technique, can edit a file whose length is limited only by maximum diskette storage. Included in the repertoire of I5 commands are macro commands to save time when encountering a redundant editing task. The Editor is also capable of performing in one operation all the commmands which will fit into an 80-column command buffer.

Summary of Editor Commands

Advance N - Advance line pointer N line
Backup N - backs up N lines
Change N/S1/S2 ehange N occurrences of string I to string 2
Delete N - Delete current line plus next N-I lines of text
Exchange N - Exchanges current line plus next N-1 lines with lines to be inserted while in insert mode.
Get file Reads another file and inserts it into the file being edited after the current line.
Insert - Place Editor in insert mode. Text will be inserted after present line.
Line N - Place line pointer on Line N.
Macro I or Macro 2 - Defines Macro I or Macro 2 by the following string of Text Editor commands.
Put n file - Outputs n lines of the file being edited to another disk file.
Quit - Stores off file under editing process and returns to Monitor environment
Search N/SI - Searches from existing pointer location until nth occurrence of string SI is located and prints it.
Top - Inserts records at top of file before first line.
Verify N - Print current record to console plus next N-I records while advancing pointer N records ahead.
Write N - Prints current records plus next N-I records to source output device while advancing pointer N records.
eXecute N - Executes Macro I or Macro 2 as defined by Macro command.

Z80 Assembler

The Z80 Resident Assembler generates relocatable or absolute object code from source files. The assembler recognizes all I58 Z80 instructions as well as 20 powerful pseudo operators. The object code generated is industry-standard, absolute or relocatable format. With the relocating feature, large programs can be easily developed in smaller sections and linked using the Linker. Because the assembler utilizes the I/O Control System, object modules or list modules can be directed to disk files, paper tape, console, or line printer. Portability of output media eliminates the requirement for a complete set of peripherals at every software/hardware development system. The assembler run-time options include sorted symbol table generation, no list, no object, pass 2 only, quit, cross reference table, and reset symbol table. The assembler is capable of handling I4 expression operators including logical, shift, multiplication,division, addition and subtraction operations. These permit complex expressions to be resolved at assembly time by the assembler rather than manually by the programmer. Comments can be placed anywhere but must be preceded by a semi-colon. Error messages are integrated with the listing file but can be directed to the console device. In addition, assembler pseudo operators are:

GLOBAL - for global symbol definition.
PSECT operator - to generate relocatable or absolute modules
IF expression - conditional assembly IF expression is true
INCLUDE dataset - to include other datasets (files) as in-line source code anywhere in source file.

Peripheral Interchange Program

PIP provides complete file maintenance activity for operations such as copy file from disk to disk, disk to peripheral, or any peripheral to any other peripheral supporting both file-structured and character-oriented devices. Key operations such as renaming, appending, and erasing files also exist along with status commands for diskette ID and vital statistics. PIP can search the diskette directories for any file or a file of a specific name, extension, and user number. The PIP operations are:

Append - appends file 1 to file 2 without changing file I.
Copy - copies input files or data from an input device to an output file or device. The Copy command can be used for a variety of purposes such as listing files, concatenating individual files, or copying all the files or a single file from one disk unit(e.g. DKO) to a second disk unit(e.g. DK1).
Directory - lists the directory of a specified disk unit (DKO, DKI, etc.). The file name, extension, and user number are listed for each file in the directory. The user can also request listing-only files of a specified name, only files of a specified extension, or only files of a specified user number. The list device can be any device supported by the system as well as a file.
Erase - erases a single file or files from a diskette in a specified disk unit. The user has the option to erase all files, only files of a specified file name, or only files of a specified user number.
Format - takes completely unformatted soft-sectored diskettes, formats to IBM 3740, and prepares to be a system diskette. Operation is performed on diskette unit 1 and a unique 11-character name is assigned to that diskette.

Init - initializes maps in the disk handler when a new diskette has been changed while in the PIP environment.
Rename - renames a file, its extension, and user number to a file of name X, extension Y and user Z.
Status - lists all vital statistics of a disk unit to any device. These include the number of allocated records, the number of used records, and the number of bad records.
Quit - returns to Monitor Environment.

DOS/Disk Handler

The heart of the FLP-80DOS software package is the Disk Operating System. Capable of supporting 4 double-sided units, the system provides a file-structure orientation timed and optimized for rapid storage and retrieval. Program debug is enhanced by complete error reporting supplied with the DOS. Additionally, extensive error recovery and bad sector allocation insure data and file integrity. The DOS not only provides file reading and writing capability, but special pointer manipulation, record deletions, record insertions, skip records both forward and backward as well as directory manipulation such as file creation, renaming, and erasure. The DOS is initiated by a calling vector which is a subset of the I/O control system vector or through the standard IOCS calling sequence to elect buffer allocation, blocking, and deblocking of data to a user-selectable, logical record type.

A unique dynamic allocation algorithm makes optimal use of disk storage space. Run time (Binary) files are given first priority to large blocks of free space to eliminate overhead in operating system and overlay programs. The algorithm marks storage fragments as low priority and uses them only when the diskete is nearing maximum capacity. The DOS permits 7 files to be opened for operations at any one time, thus permitting complex application programs as well as multi-user operation of the DOS.

I/O Control System

The I/O Control System provides a central facility from which all calls to I/O can be structured. This permits a system applications program to dissolve any device dependence by utilizing the logical unit approach of large, main-frame computers. For example, a programmer may want to structure the utility to use logical unit No. 5 as the list device which normally in the system defaults to the line printer. He may , however, assign at run time a different device for logical No. 5. The application program remains unchanged.

Interface by a user to IOCS is done by entering a device mnemonic in a table and observing the calling sequence format. IOCS supplies a physical buffer of desired length, handles buffer allocation, blocking, deblocking, and provides a logical record structure as specified by the user.

DDT

The Designer's Debugging Tool consists of commands for facilitating an otherwise difficult debugging process. The AID-80F's rapid source changes through the editor and re-assemblies, followed by DDT operations close the loop on the debug cycle. The DDT commands include:

Memory - display, update, or tabulate memory
Port - display, update or tabulate I/O ports
Execute - execute user's program
Hexadecimal - performs 16 bit add/sub
Copy - copy one block to another
Breakpoint - sets software trap in user code for interrupting execution in order to examine CPU registers
Register - displays contents of user's registers
Offset - enters address adder for debug of relocatable modules
Fill - fills specified portion of memory with 8 bit byte
Verify - compares two blocks of memory
Walk - software single step/multistep
Quit - returns to Monitor

LINKER

The Linker program provides the capability of linking assembler-generated, absolute or relocatable object modules together to create a binary or run-time file. This process permits generation of programs which may require the total memory resources of the system. The linking process includes the library search option which, if elected, will link in standard library object files (device drivers, math pack functions) from disk to resolve undefined global symbols. Another option selects a complete global symbol cross-reference listing.

Batch- Mode Operation

In Batch-Mode Operation, a command file is built on disk or assigned to a peripheral input device such as a card reader. The console input normally taken from the keyboard is taken from this batch device or batch file. While operating under direction from a batch file, the console output prompts the user as normal or the prompting can be directed to any other output device. The Batch operation is especially useful for the execution of redundant procedures not requiring constant attention of the operator and for allowing several programmers to use one system.

HARDWARE DESCRIPTION OEM-80

The OEM-80, also available as a complete, single-board development system (SDB-80), provides the essential power of the system. While using the Z80 as the central processing unit, the OEM-80 is provided with other Z80 family peripheral chip support. Two Z80 PIO's give the system 4 completely programmable, 8-bit parallel I/O ports with handshake from which the standard system peripherals are interfaced. Also, in the system is the

Z80-CTC counter timer circuit which has 3 free flexible channels to perform critical counting and event counter/timing functions. Along with 16K or RAM, the OEM-80 provides 5 ROM/PROM sockets which can be utilized for 10/20K of ROM or 5/10K PROM. Four sockets contain the firmware portion of FLP-80DOS. The remaining socket can be strapped for other ROM/PROM elements. The OEM-80 is particularly flexible for system expansion. Expansion of memory, (ROM,PROM, or RAM) is made easy by off-board select logic or by the on-board strapping flexibility.

RAM-80B

The RAM-80B adds additional memory with Mostek's MK4116 16K dynamic memory along with more I/O. These two fully programmable 8-bit I/O ports with handshake provide additional I/O expansion as system RAM memory needs grow.

FLP-80

Integral to the AID-80F system is the floppy controller. The FLP-80 is a complete IBM 3740 single-density/double-sided controller for up to 4 drives. The controller has 128 bytes of FIFO buffer resulting in a completely interruptable disk system.

AIM-80

The AIM-80 module provides extended debug for the AID-80F. In Z80 development, real time in-circuit emulation permits debug of the hardware and the software at the most intimate level. Hardware single-step/multi-step with register trace, execution intercept on memory access, port access, or external trigger provides the absolute control over any system, regardless of how complex it is. The "pushbutton intercept" enables the programmer to perform a controlled recovery for those extremely difficult-to-trace processor lock-out loops. With the memory clock selectable history module, any past 256 events of data, address, or control bus operation are captured in real time and are displayed.

The AIM-80 includes 8K bytes of ROM firmware introducing unique software including a mnemonic disasembler for inverse assembly of history module contents of single step/multistep operations. "In-line" code disassembled to language mnemonics provides insight into execution results as if examining an assembler-generated listing. Extra added capability is the ROM resident self test of OEM-80 or target RAM.

AIM-72

The AIM-72 module provides debug and in-circuit emulation capabilities for the 3870 series microcomputers (3870, 3872 3874, and 3876) on the AID-80F. Multiple breakpoint capability and single-step operation allows the designer complete control over the execution of the 3870 series microcomputer. Register and Port display and modification capability provides information needed to find system "bugs". All I/O in the user's system is connected to AIM-72 by a 40-pin interface cable. Program storage on the AIM-72 is in RAM so that the results of disk-based Editing and Assembly can be quickly loaded to the AIM-72 memory for debug with the user's I/O devices.

Software supporting the AIM-72 in the AID-80F system is a 3870/F8 Cross Assembler. This assembler produces either relocatable or absolute object code. All AID-80F editing and utility software is available to the user to speed the process of programming 3870 series single-chip microcomputers.

MECHANICAL SPECIFICATIONS

AID-80F Enclosure	
Overall Dimension	- 20"w x 22"1 x 12"h
Material	- NORYLEN 185
Color Composition	- White GE No. 8385; Blue GE No.2283
Weight	- 60 lbs.
Front Panel Dimensions	- 3.75" x 3.75"
Read-End Panel Dimensions	- 4.25" x 4.62" 4.25" x 2.00"
Fan Capacity	- 52 CFM

POWER SUPPLY

Input - 115 VAC 60Hz
Outputs - +5VDC at 10 Amps Max/
-5VDC at 0.15 Amps Max.
+12VDC at 3 Amps Max.
-12VDC at 0.5 Amps Max.
+24VDC at 3 Amps Max.

ORDERING INFORMATION

NAME	DESCRIPTION	PART NO.
AID-80F	Z80 floppy disk development system with power supply, cooling fans, enclosure, six slot card cage, 32K dynamic RAM, FLP-80DOS disk operating system,* and two single-sided disk drives	MK78125
AID-80F-48	AID-80F with 48K dynamic RAM	MK78125-2
AID-80F-64	AID-80F with 64K dynamic RAM	MK78125-3
AID-80F-P	PPG-8/16 integrated into AID-80F	MK78125-01
AID-80F-DS	Double-sided drive option	MK78125-001
AID-80F-FQ	50Hz, 220v, option	MK78125-0001

NAME	DESCRIPTION	PART NO.
AID-80F-16	Z80 floppy disk development system described above with 16K dynamic RAM only. Note: 16K RAM system will not run Mostek BASIC FORTRAN; requires addition of RAM-80B board	MK78125-1
AIM-80	RAM based Z80 In-Circuit emulation module with buffer box and cable	MK78132
AIM-72	RAM based in-circuit emulation module for 3870 series of single chip microcomputers (3870, 3872, 3874 and 3876)	MK79076
FZCASM	Relocatable 3870/F8 cross assembler to run on AID-80F	MK79079
PPG-08	PROM programmer for 2708 PROMs	MK79033
PPG-8/16	PROM programmer for 2708,2758 and 2716 PROMs(order MK79125-01 for factory integration)	MK79081
XAID-805	Connecting cable from PPG-08 or PPG-8/16 to AID-80F (only required if not factory integrated)	MK79041

NAME	DESCRIPTION	PART NO.
RAM-80B	Dynamic RAM board with 16K bytes of RAM expandable to 64K; includes four 8-bit I/O ports with handshake control	MK78108
MOSTEK BASIC	BASIC interpreter to run on AID-80F; requires 32K bytes of memory	MK78157
MOSTEK FORTRAN	Fortran IV high-level language compiler to run on AID-80F-48; requires 48K bytes of memory	MK78158
MACRO-80	Z80 MACRO Assembler requires 32K bytes of memory	MK78165
LIB-80	Volume 1 of Z80 Software Library includes both source and binary	MK78164
MOSTEK CRT	110-9600 Baud CRT upper and lower case 7 x 10 dotmatrics, EIARS 232 and 20mA current loop. Includes cables	MK78149
MOSTEK	7x7 dot matix Line Printer with 120 character per second operation. Includes cables	
AID-80F-	AID-80F Operations Manual	MK78569

*The FLP-80DOS software package includes binary run time files of all system software described. Source to FLP-80DOS is available subject to the terms and conditions of the Mostek OEM Software License Agreement.

Develop Systems US

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Application Interface Module (AIM*-80)

HARDWARE FEATURES

□ Direct Interface with SDB-80

□ Single step/multistep with register trace

□ Execution intercept (breakpoint) intercepts on memory access, port access, or external trigger

□ Pushbutton execution intercept

□ 256x32 history memory which samples Data Bus, Address Bus, $\overline{\text{M1}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{IORQ}}$, and four external probes

□ History memory clock selectable from $\overline{\text{M1}}$, $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, or INTERRUPT ACKNOWLEDGE

□ Selectable history memory clock conditions: read only, write only, DMA only, or external probe only (high or low)

□ 8K x 8 ROM memory (firmware)

SOFTWARE FEATURES

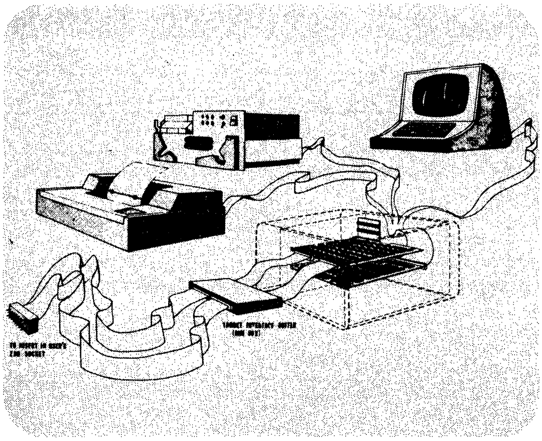
□ ROM resident mnemonic dis-assembler

□ ROM resident RAM test for SDB or target RAM

GENERAL DESCRIPTION

AIM-80 provides Z80 system debug assistance for both software and hardware via in-circuit emulation. (See Block Diagram).

Single step/multistep allows the programmer to trace through a program and display the CPU registers after each instruction. The execution intercept feature allows suspending program execution on the nth occurrence of an address or other specified condition. If the program has begun an unknown sequence, the intercept pushbutton will return the system to the single step mode. Single step and execution intercept (breakpoint) operate in RAM or ROM/PROM.



Hardware debugging is aided greatly by use of the 256x32 history memory which monitors bus transactions for a specified period. This information may then be displayed on the console. The data bus, address bus, $\overline{\text{M1}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{IORQ}}$, and inputs from four probes are sampled and stored in the history memory upon every occurrence of the user specified clock ($\overline{\text{M1}}$, $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, or interrupt acknowledge) qualified by the user specified conditions (read only, write only, DMA only, probe High only, or probe Low only). Upon the occurrence of the selected intercept, AIM-80 returns control to the system debug (DDT-80). The history memory may then be displayed (See AIM-80 print-out example 1) with or without mnemonic dis-assembly.

USING THE AIM-80

AIM-80 may be added directly to any SDB-80 system. All system bus signals are wired one to one between SDB-80 and AIM-80. Voltage requirements for the AIM-80 are the same as for the SDB-80. Programs may be debugged in SDB-80 memory space or with the target interface buffer box (AIM-80X) may be debugged right in the target environment. Dynamic memory mapping allows target memory to be simulated using SDB-80 system RAM. All peripheral devices of the SDB-80 are still functional with the AIM-80.

*Trademark of Mostek Corporation

SYSTEM FIRMWARE

To minimize the impact of the AIM-80 on the users memory space, all AIM-80 firmware is resident in one MK36000, 8K x 8 ROM. This firmware is completely compatible with DDT-80 firmware and includes five new commands for control of the AIM-80. The interactive nature of the commands makes operation simple and avoids operator errors. The ROM resident dis-assembler makes correlation with the user's source listing easier and reduces the necessity of memorizing op codes.

ELECTRICAL SPECIFICATIONS

Operating Temperature Range 0°C to +50°C
Power Supply Requirements (Typical)
+12V ± 5% @ 12 mA
+5V ± 5% @ 1.0A
Interface - SDB-80 compatible
Operating Frequency - 1-2.5 MHz (with SDB-80)

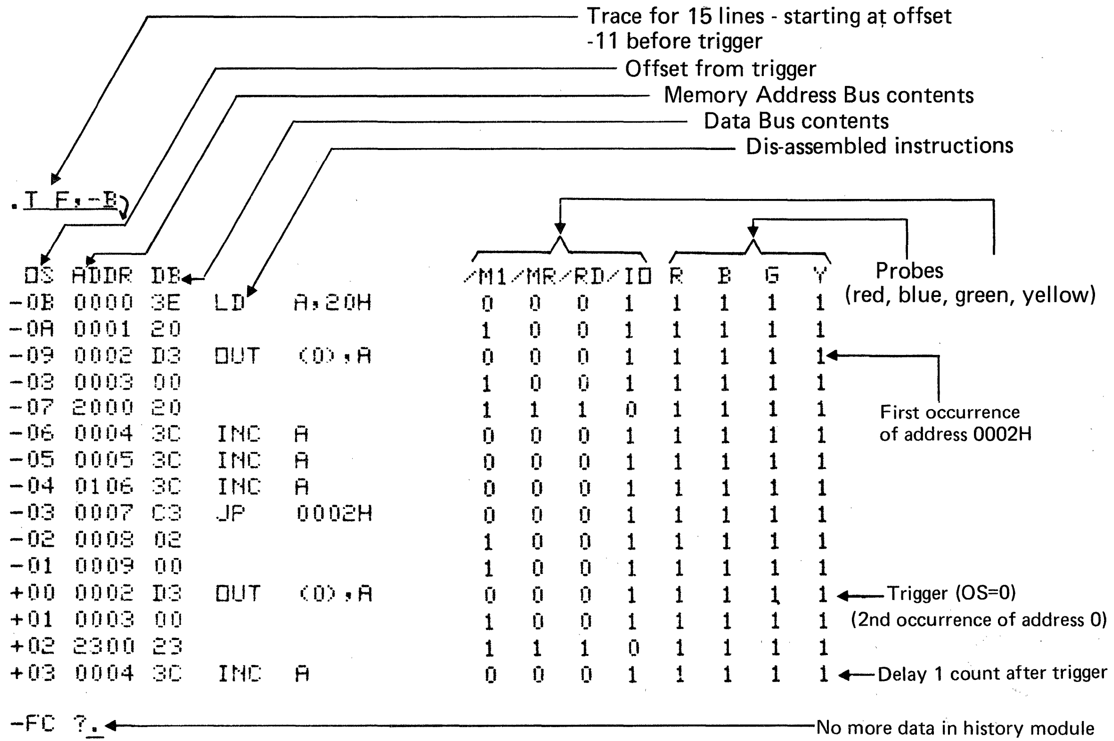
MECHANICAL SPECIFICATIONS

Board Size: 8.5" x 12.0" x 0.65"
Bottom Connector: 100 Pin 125 mil centers
Top Connectors: One 40 pin 3M ribbon
One 50 pin 3M ribbon

AIM-80 PRINT-OUT EXAMPLE
(user entries underlined)

`.I 2,T`	Set intercept at address 0002H with trigger option
`TRIG ON (MREQ/IORQ/+/-) M`	Trigger on $\overline{\text{MREQ}}$
`EVENT CNT (1-FF) 2`	After 2 occurences
`DELAY CNT (0-FF) 1`	Delay 1 clock after trigger
`CLOCK ON (M1,MREQ,IORQ,INTA) MR IO`	Clock history (sample) memory on $\overline{\text{MREQ}}$ or $\overline{\text{IORQ}}$
`ONLY IF (RD/WR/DMA/H/L)`	No qualifying conditions selected
`.E 0`	Begin execution at address 0000H
`0005 2420`	Intercept occurs at second occurrence of address 2 with a delay of one.

At this point the history memory contains the bus transactions which occurred before the intercept.



Trace for 15 lines - starting at offset -11 before trigger
Offset from trigger
Memory Address Bus contents
Data Bus contents
Dis-assembled instructions

```
.T F,-B

 OS  ADDR DB                       /M1/MR/RD/IO  R   B   G   Y
-0B  0000 3E   LD    A,20H          0  0  0  1   1   1   1   1
-0A  0001 20                        1  0  0  1   1   1   1   1
-09  0002 D3   OUT   (0),A          0  0  0  1   1   1   1   1
-08  0003 00                        1  0  0  1   1   1   1   1
-07  2000 20                        1  1  1  0   1   1   1   1
-06  0004 3C   INC   A              0  0  0  1   1   1   1   1
-05  0005 3C   INC   A              0  0  0  1   1   1   1   1
-04  0106 3C   INC   A              0  0  0  1   1   1   1   1
-03  0007 C3   JP    0002H          0  0  0  1   1   1   1   1
-02  0008 02                        1  0  0  1   1   1   1   1
-01  0009 00                        1  0  0  1   1   1   1   1
+00  0002 D3   OUT   (0),A          0  0  0  1   1   1   1   1
+01  0003 00                        1  0  0  1   1   1   1   1
+02  2300 23                        1  1  1  0   1   1   1   1
+03  0004 3C   INC   A              0  0  0  1   1   1   1   1

-FC ?.
```

Probes
(red, blue, green, yellow)

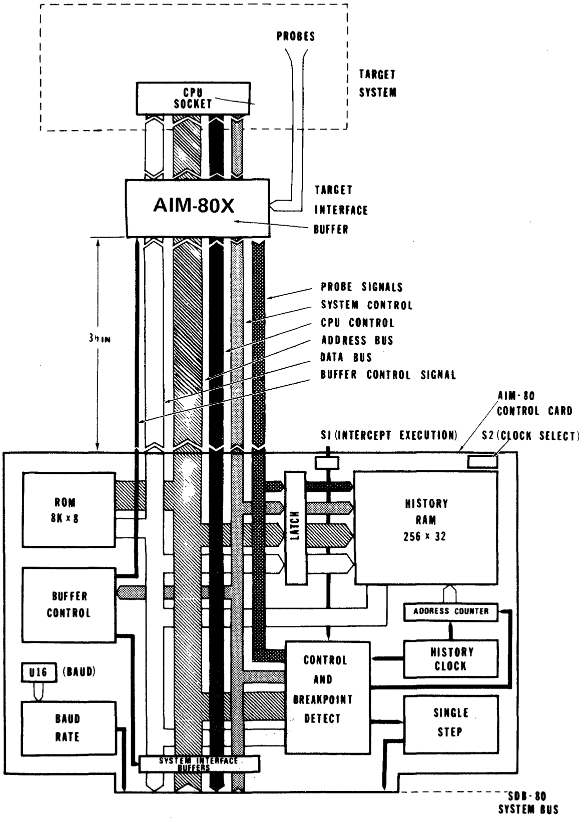
First occurrence of address 0002H

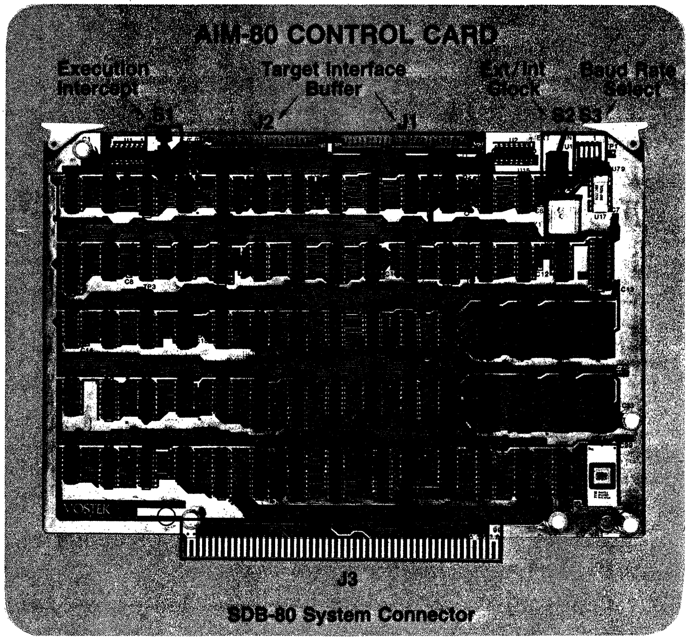
Trigger (OS=0)
(2nd occurrence of address 0)

Delay 1 count after trigger

No more data in history module

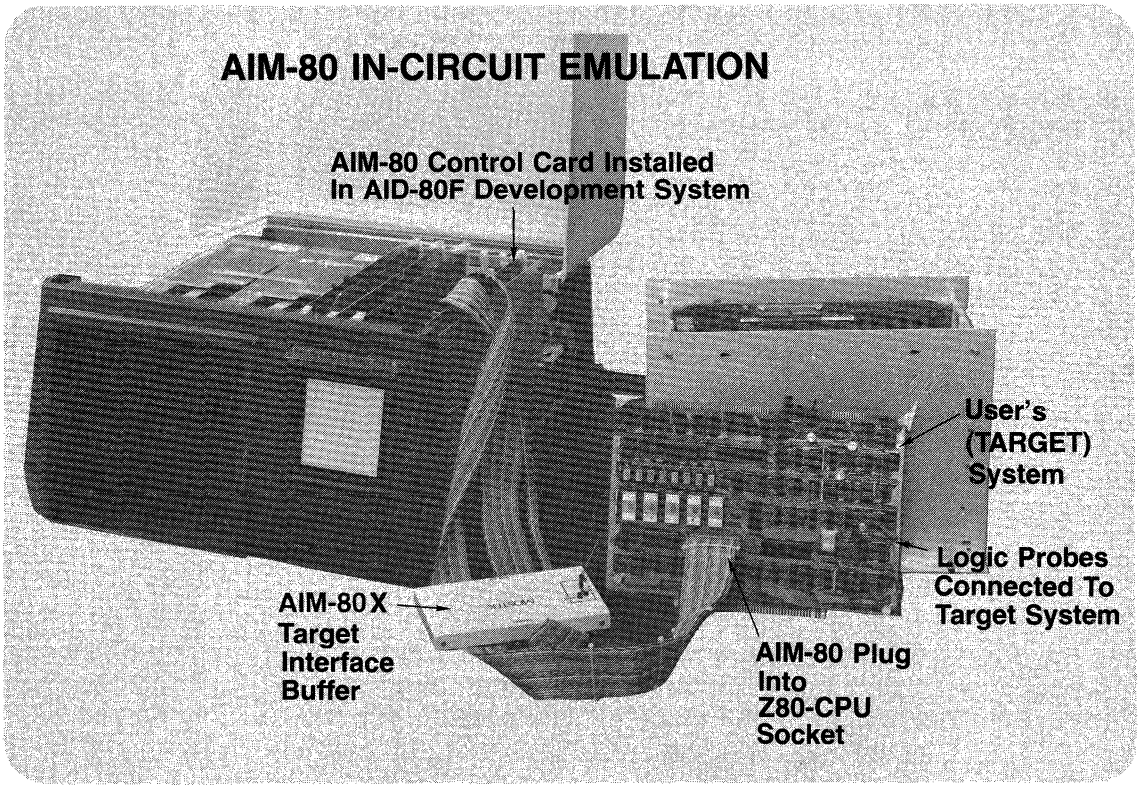
AIM-80 BLOCK DIAGRAM





ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
AIM-80	AIM-80 Circuit Board includes: AIM-80 Firmware AIM-80 Operations Manual 100 pin edge connector Target Interface Buffer (AIM-80X) includes: cables connectors 4 probe clips	MK78132
	AIM-80 Operations Manual	MK78546


AIM-80 IN-CIRCUIT EMULATION
AIM-80 Control Card Installed In AID-80F Development System
User's (TARGET) System
Logic Probes Connected To Target System
AIM-80X Target Interface Buffer
AIM-80 Plug Into Z80-CPU Socket

MOSTEK®

3870 MICROCOMPUTER SYSTEMS

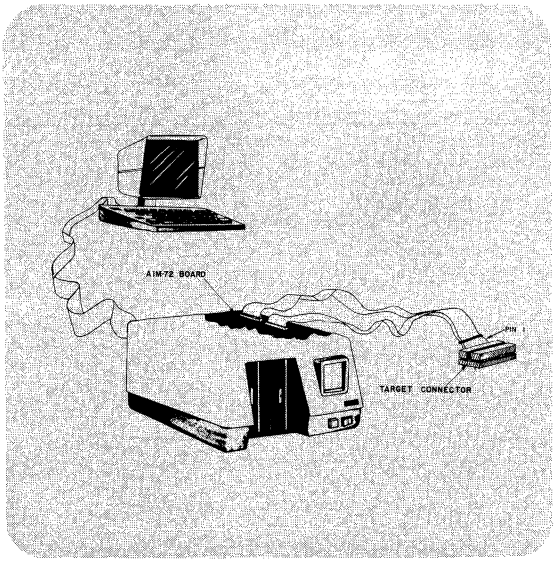
Application Interface Module (AIM-72)

FEATURES

- □ Real time in-circuit emulation of Mostek's 3870 family of microcomputers, including MK3870 MK3872 and MK3876
- □ Direct interface to Mostek's AID-80F Dual Floppy Disk Microcomputer with ZAIM-72 software supplied on floppy disk
- □ Direct interface with Mostek's SDB-50/70 (Software Development Board) with FAIM-72 software supplied on paper tape
- □ Standard features include:
  - ● Breakpoint insertion, memory display and modification, register display and modification, port display and modification, and single step
  - ● Execution intercept from user keyboard with the ESCAPE key
  - ● Debugging or emulation mode is selectable from the user's console
  - ● Debugging of 3870 and F8 programs up to 8K long can be done without a target system

GENERAL DESCRIPTION

AIM-72 (Application Interface Module) is a unique development aid for debugging 3870 Series Microcomputer applications in the actual hardware and software configuration of the user's final system (referred to as the 'Target'.) To accomplish this, it is first necessary to emulate the Target ROM with RAM. This RAM must appear as ROM to the application while retaining the ability to be loaded, debugged, and modified using peripherals independent of the Target. It is the purpose of AIM-72, used in conjunction with the AID-80F Disk Based Microcomputer or the SDB-50/70, to provide these capabilities. With AIM-72, all of the peripheral and debugging capabilities of the user's development system may be applied directly to either the prototype or final production configuration of any 3870, 3872 or 3876 application; no modifications to the user's hardware, software, or mechanical package are required.



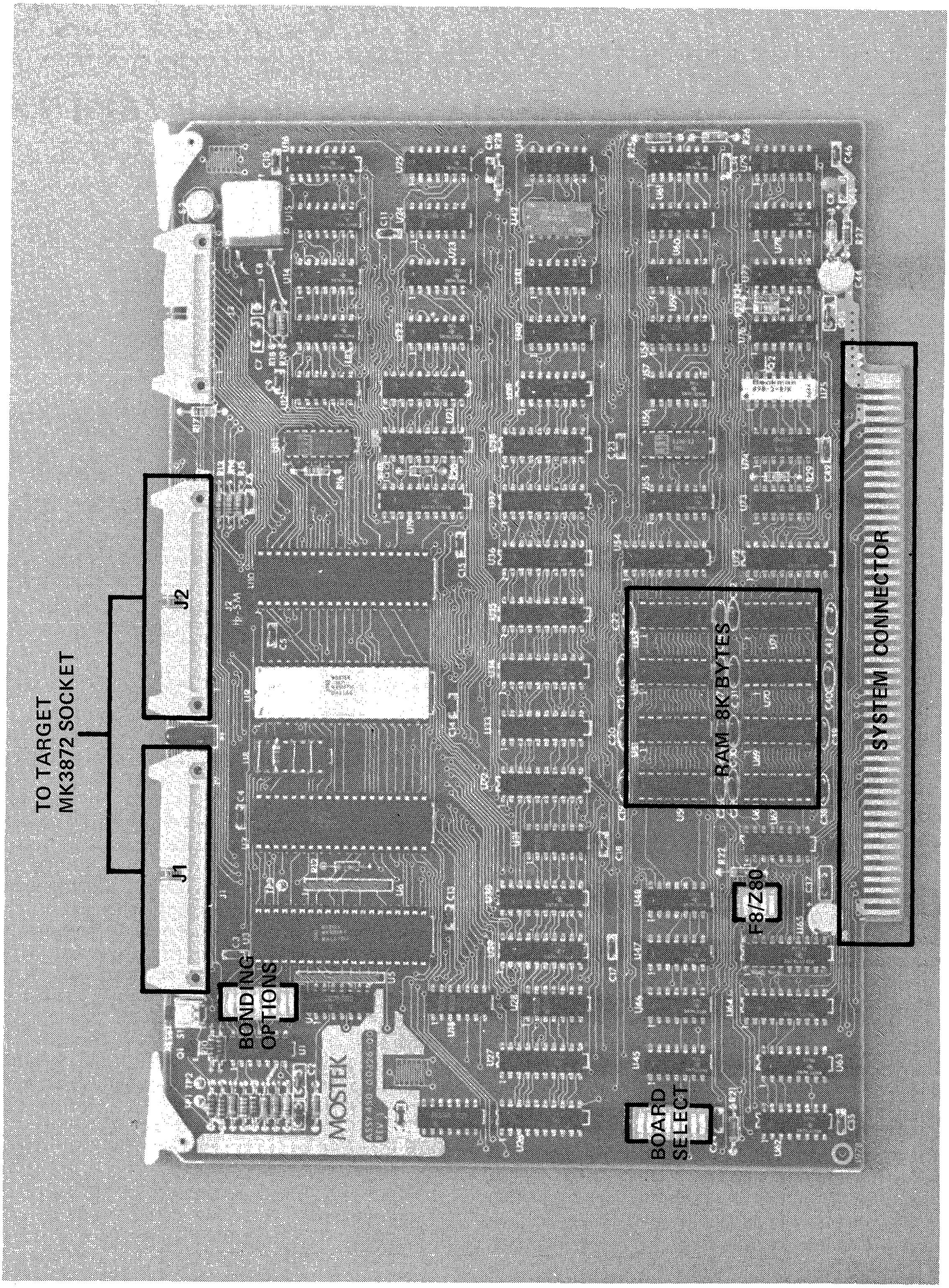
USING AIM-72

The pictorial diagram above shows how AIM - 72 would typically be used during system development. Because the AIM-72 is an exact functional emulation of the 3870 family, it may be directly inserted into the 3870, 3872, or 3876 socket in the target system. Also, since the Target can be a production version of the user's application, product revisions and enhancements may be easily implemented. As shown in the diagram, the AIM board is usually mounted in the card cage of the user's development system. It is the purpose of the SDB to provide the user with the means for accessing and controlling the target system (via the AIM board) during the program development phase. This provides access to all the debugging software and peripherals of the development system without having to introduce any perturbations to the target system environment. AIM does not affect the peripheral expansion capabilities of the development system.

BLOCK DIAGRAM DESCRIPTION

As shown in the block diagram, the AIM-72 contains all the functional elements necessary to emulate 3870 Series Microcomputers. Target Ports are emulated with the CPU and PIO Ports. Target ROM

AIM-72 PHOTO



Develop
Systems
US

ZAIM-72 SOFTWARE DESCRIPTION

ZAIM-72 is the software designed to operate the AIM-72 board on Mostek's AID-80F Dual Floppy Disk Microcomputer. It is supplied on a standard FLP-80DOS diskette. The software has the same command structure as other Mostek debuggers. The commands available with ZAIM-72 are summarized below. Designations s,f, and d stand for operands.

,A s,f Assign data byte f to target memory location s.

,B s Set a breakpoint at target memory location s. Up to 8 breakpoints can be set at once.

,C s, f, d Copy the target memory block s to f to target memory starting at d.

,E s Execute target program at location s.

,F s, f, d Fill target memory locations s through f with data d.

,G s Get binary file s and load it into Target memory.

,H Hexadecimal arithmetic.

,I Reinitialize target system.

,L s, f, d Locate data d in target memory range s through f.

,M s Display and update target memory at location s.

,M s, f, d Tabulate target memory locations s through f. Option d specifies additional printout of ASCII characters or disassembly.

,O s Set relative offset equal to s for all address operands.

,P s Display and update target port number s.

,Q Quit and return to FLP-80DOS Monitor.

,R s, f Display target registers, Option s allows a heading to be printed and option f specifies the number of scratchpad registers to be displayed.

,S s, f Single step starting at target location s for f number of steps.

,V s, f, d Verify target memory block s through f against target memory block starting at location d.

Target system programs are developed using the Mostek AID-80F Cross Assembler for 3870/F8 Microcomputers (FZCASM-MK79079). Then ZAIM-72 is used to debug the completed program on the user's target system. The software features multiple breakpoints, single step, and in-line disassembly. Target system memory, ports, and registers may be displayed and updated.

SPECIFICATIONS

Operating Temperature Range	0°C to 50°C
Power Supply Requirement	+5V ± 5% @ 1.5A max. +12 V ± 5% @ 100mA max. −12 V ± 5% @ 30mA max.
Board Size	8.5" x 12.0" x .75"
Connectors/Cables	40-Pin Ribbon Cable (24" long)

and RAM are emulated with the 8K x 8 RAM which can also be accessed directly by the control system via the bottom edge connector. System memory accesses are transparent to the Target system execution. Thus, there is no impact on target execution timing. The Target memory map can be controlled from the system allowing 2K, 4K or 8K Bytes of memory to be available in the Target System. Debug firmware in a PSU on the AIM-72 interfaces with the system to implement the breakpoint, single step and other functions. Trap control circuitry allows the use of a single byte breakpoint, providing complete flexibility when using break points in tight programming loops. Execution is at full speed, determined only by the user's crystal frequency - no speed reduction is introduced by the AIM's operating system. The AIM-72 clock may be implemented from the Target system, from an on-board crystal oscillator, or from the SDB-50/70 clock.

MULTI 3870 SERIES APPLICATIONS

Up to eight AIM-72 boards may be installed in one control system with each AIM-72 used to emulate a different 3870 Series Microcomputer. The debug functions on each AIM-72 may be enabled one at a time and each program developed until all Target programs are functional. Only one AIM-72 may be in the active debug mode at a time; other AIM-72's will be in the emulator mode.

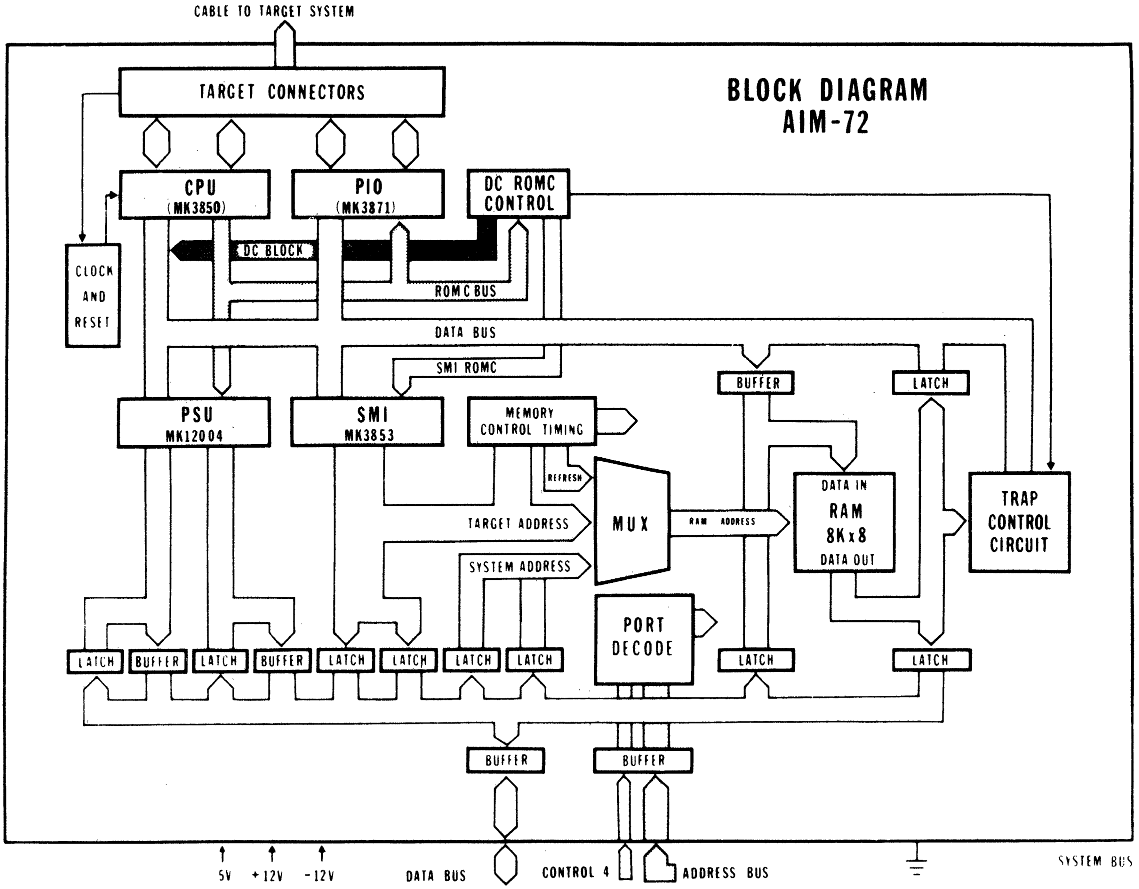
FAIM-72 SOFTWARE

FAIM-72 is the software designed to operate the AIM-72 board with the SDB-50/70 Software Development Board. It is supplied on a paper tape or cassette for loading into the SDB-50/70 memory.

The hardware and software associated with AIM have been designed to retain the same command structure as the SDB. The only difference is that all operands (Memory Addresses or Port Addresses) which correspond to the 'Target' system must be preceded by the letter 'T'. The commands available with FAIM-72 are summarized. Designations s, f, and d stand for operands.

,B Ts	Set breakpoint to exit target program at address s
,C Ts, Tf, Td ,C s, f, Td ,C Ts, Tf, d	Copy memory block from address s thru address f in the SDB or target to the memory block starting at address d in the SDB or target
,D Ts, Tf	Dump formatted tape from target memory block from address s thru address f.
,E Ts	Execute target program at address s
,I	Re-initialize AIM-72
,L T	Load formatted tape into target memory
,M Ts	Display and update target memory at address s
,M Ts, Tf	Tabulate target memory block from address s thru address f
,P Ts	Display and update target port s
,P Ts, Tf	Tabulate target ports s thru f
,Q	Return to DDT-2
,S Ts	Begin single step execution at address s in target program

Each of these SDB commands may be applied to any portion of the target system's port or memory map. This is accomplished by means of a 'handshaking' procedure between the CPU on the AIM and the SDB. Handshaking is initiated automatically when the system is initialized or whenever single-step execution of a target instruction has been completed or when a breakpoint is encountered. Also, whenever handshaking is initiated, the contents of all target system registers (Scratchpad, Status, Accumulator, etc.) are transferred to a designated portion of the SDB memory map where they may be examined or modified. This portion of the SDB memory is called the 'Register Map' and is also used to initialize the target system register whenever execution is initiated (or resumed) in the target system.


CABLE TO TARGET SYSTEM
BLOCK DIAGRAM
AIM-72
TARGET CONNECTORS
CPU
(MK3850)
PIO
(MK3871)
DC ROMC
CONTROL
CLOCK
AND
RESET
DC BLOCK
ROMC BUS
DATA BUS
SMI ROMC
BUFFER
LATCH
PSU
MK12004
SMI
MK3853
MEMORY
CONTROL TIMING
REFRESH
TARGET ADDRESS
SYSTEM ADDRESS
MUX
RAM ADDRESS
DATA IN
RAM
8Kx8
DATA OUT
TRAP
CONTROL
CIRCUIT
PORT
DECODE
LATCH
BUFFER
LATCH
BUFFER
LATCH
LATCH
LATCH
LATCH
LATCH
LATCH
BUFFER
BUFFER
5V
+12V
-12V
DATA BUS
CONTROL 4
ADDRESS BUS
SYSTEM BUS

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO
AIM-72 Operations Manual	Contains a complete description of the use and operation of AIM-72.	MK79577
AIM-72	Includes the AIM-72 circuit board, the AIM-72 Operations Manual, the ZAIM-72 software on diskette, and the FAIM-72 software on paper tape for developing 3870 Series Applications.	MK79076
AID-80F	A complete dual floppy disk development system (less terminal and line printer). Order FZCASM to provide 3870 series assembly capability. Order AIM-72 to provide 3870 Series debug capability.	MK78125
FZCASM	AID-80F Cross Assembler for 3870/F8 Microcomputers. Provides disk-based assembly for 3870 assembly language programs on the Mostek AID-80F Microcomputer.	MK79079
SDB-50/70	Includes the SDB-50/70 circuit board with complete documentation. The SDB-50/70 is used with the AIM-72 as a stand alone microcomputer with resident software for 3870 series program assembly and debug.	MK79019
FAIM-72	Optional cassette tape based AIM-72 software for use with Silent 700 terminal and SDB-50/70 development system.	MK79083

MOSTEK®

MICROCOMPUTER SUPPORT

Prom Programmer (PPG-08)

FEATURES

- □ Programs, reads, and verifies MK 2708 PROMS
- □ Directly interfaces to SDB-50/70 and SDB-80
- □ Driver software included
- □ Zero insertion force socket
- □ Power and programming indicators

GENERAL DESCRIPTION

The MK 2708 PROM Programmer (PPG-08) is a peripheral which provides a low-cost means of programming MK 2708 UV erasable PROMs. The PPG-08 has a generalized computer interface (two 8-bit I/O ports) allowing it to be controlled by most types of host computers with user-generated driver software. It is directly compatible with MOSTEK's F8 Software Development Board (SDB-50/70) and Z80 Software Development Board (SDB-80). Driver software in paper-tape form and source listings for the SDB-50/70 and SDB-80 are included with the purchase of the PPG-08. A complete set of documentation is also provided with the PPG-08 which describes the internal operation and details user's operating procedures. Interface cables for the SDB-50/70 and SDB-80 may be purchased separately. Another optional accessory is a TI Silent 700 compatible cassette tape containing control software for the SDB-50/70 and SDB-80.

SPECIFICATIONS

Interface

40 pin control connector (.1" centers card edge)
12 pin power connector (.156" centers card edge)
All control signals are TTL compatible.

Power requirements

+12 VDC @ 250 mA typical
+ 5 VDC @ 100 mA typical
−12 VDC @ 50 mA typical

Operating Temperature	0° to 50°C
Programming time (maximum)	2.5 minutes
Physical Dimensions	5" x 7" x 2"

ORDER INFORMATION

NAME	DESCRIPTION	PART NO.
PPG-08	MK 2708 PROM Programmer	MK 79033
XAID-805	Cable for interface to SDB-80	MK 79041
XAID-705	Cable for interface to SDB-50/70	MK 79046
SWD-1	Driver software on TI Silent 700 compatible cassette tape for SDB-50/70 and SDB-80	MK 79051

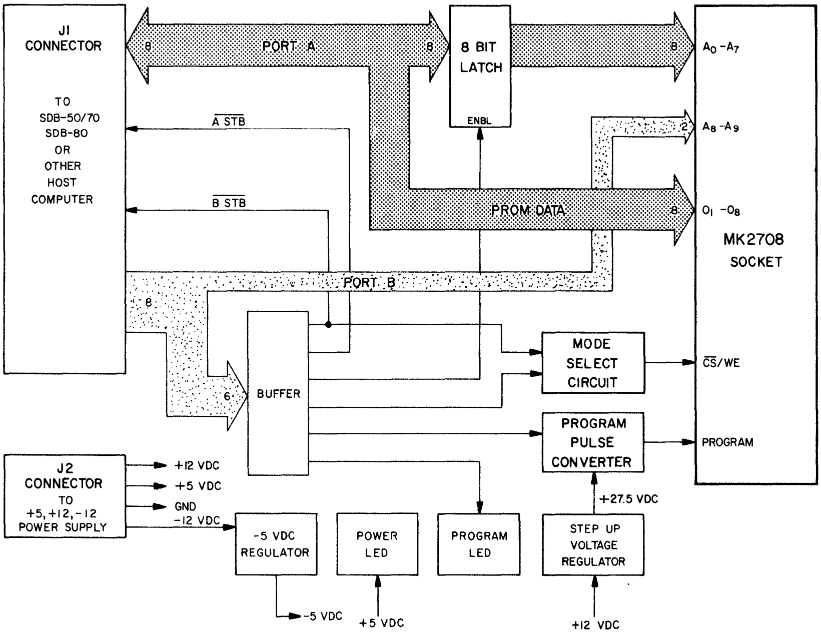
CONTROL CONNECTOR (J1) PIN-OUT

PIN #	Signal Name	Direction	Description
All Odd Pins (1-39)	GND		Logic Ground
J1-2	ASTB	Output	"LOW"when Port A (PA0-PA7) is in output mode
	PA0 - PA7	Bidirectional	PORT A (PA0-PA7) is used to output the lower 8 bits of PROM address to latch, output PROM data during programming and input PROM data during read sequence.
J1-24	BSTB	Output	"LOW" when Port A (PA0-PA7) is in input mode.
J1-26	PB0/ADDR8	Input	PROM address bit 8
J1-28	PB1/ADDR9	Input	PROM address bit 9
J1-30	PB2/PAIN	Input	"HIGH" when Port A (PA0-PA7) is in input mode and PROM is in read mode.
J1-32	PB3/PROG MODE	Input	"HIGH" during program mode.
J1-34	PB4/PROG PULSE	Input	Programming Pulse
J1-36	PB5/PA OUT	Input	"HIGH" when Port A (PA0-PA7) is in output mode.
J1-38	PB6/CLK LATCH	Input	Clock to strobe address bits 0-7 into latch
J1-40	PB7/PROG LED	Input	Control line for programming indicator

POWER CONNECTOR (J2) PIN-OUT

J2-1,A	$+5V_{DC}$	J2-4, 5, D, E	$+12V_{DC}$
J2-2, 3, B, C	GND	J2-6, F	$-12V_{DC}$

BLOCK DIAGRAM



MOSTEK®

MICROCOMPUTER SYSTEM

PROM Programmer (PPG-8/16)

FEATURES

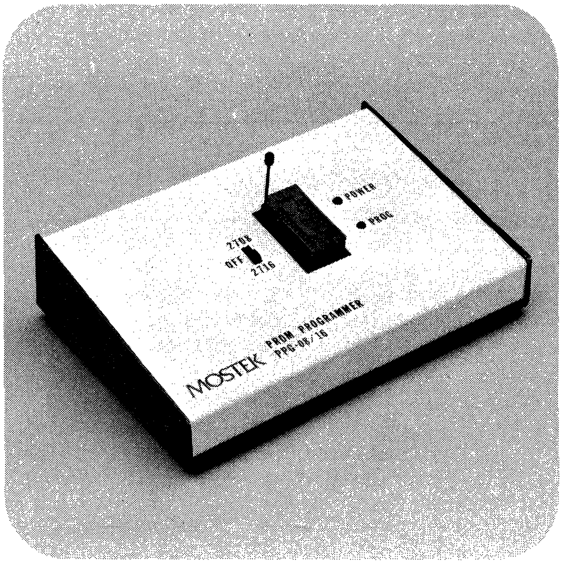
□ Programs, reads, and verifies 2708, 2758, and 2716 type PROMs (2758 and 2716 PROMS must be 5 Volt only type)

□ Directly interfaces to SDB-50/70, SDB-80, AID-80F and SYS-80F

□ Driver software included

□ Zero insertion force socket

□ Power and programming indicators



DESCRIPTION

The PPG-8/16 PROM Programmer is a peripheral which provides a low-cost means of programming 2708, 2758, or 2716 PROMs. It is directly compatible with MOSTEK's F8 Software Development Board (SDB-50/70), Z80 Software Development Board (SDB-80), Z80 MDX system and Z80 Microcomputer Development System (AID-80F or SYS-80F). The PPG-8/16 has a generalized computer interface (two 8-bit I/O ports) allowing it to also be controlled by other types of host computers with user-generated driver software. A complete set of documentation is provided with the PPG-8/16 which describes the internal operation and details user's operating procedures.

The PPG-8/16 is available in two packaging configurations: (1) In a metal enclosure for use with the SDB-50/70, SDB-80 or SYS-80F, or alternatively (2) fully integrated into the AID-80F. Interface cables for either the SDB-80 or SDB-50/70 must be purchased separately.

SOFTWARE DESCRIPTION

The driver software available for the SDB-80, SDB-50/70, AID-80F, and SYS-80F accomplishes four basic operations. These are: (1) loading data (object tapes for SDB-80 and SDB-50/70 or binary files for the AID-80F or SYS-80F) into host computer memory, (2) reading the contents of a PROM into host computer memory, (3) programming a PROM from the contents the host computer memory, and (4) verifying the contents of a PROM with the contents of the host computer memory.

The driver software is provided in the form of paper-tape for both the SDB-50/70 and the SDB-80. An optional accessory is a TI Silent 700 compatible cassette object tape containing control software for the SDB-50/70 and SDB-80. Users of MOSTEK's AID-80F or SYS-80F who wish to upgrade their systems with a PPG-8/16 will find the driver software on their system diskette (version 2.0 or later). The user documentation provided with PPG-8/16 fully explains programming procedures which enable a user to develop a software driver on a different host computer.

SPECIFICATIONS

INTERFACE

25 Pin control connector (D type) for MDX or SYS-80F
40 Pin control connector (.1 in. centers card edge) for AID-80F, SDB-80, or SDB-50/70
12 pin power connector (.156 in. centers card edge)
All control signals are TTL compatible.

POWER REQUIREMENTS

+12VDC @ 250 mA typical
+5VDC @ 100 mA typical (Supplied by AID-80F or SYS-80F)
-12VDC @ 50 mA typical

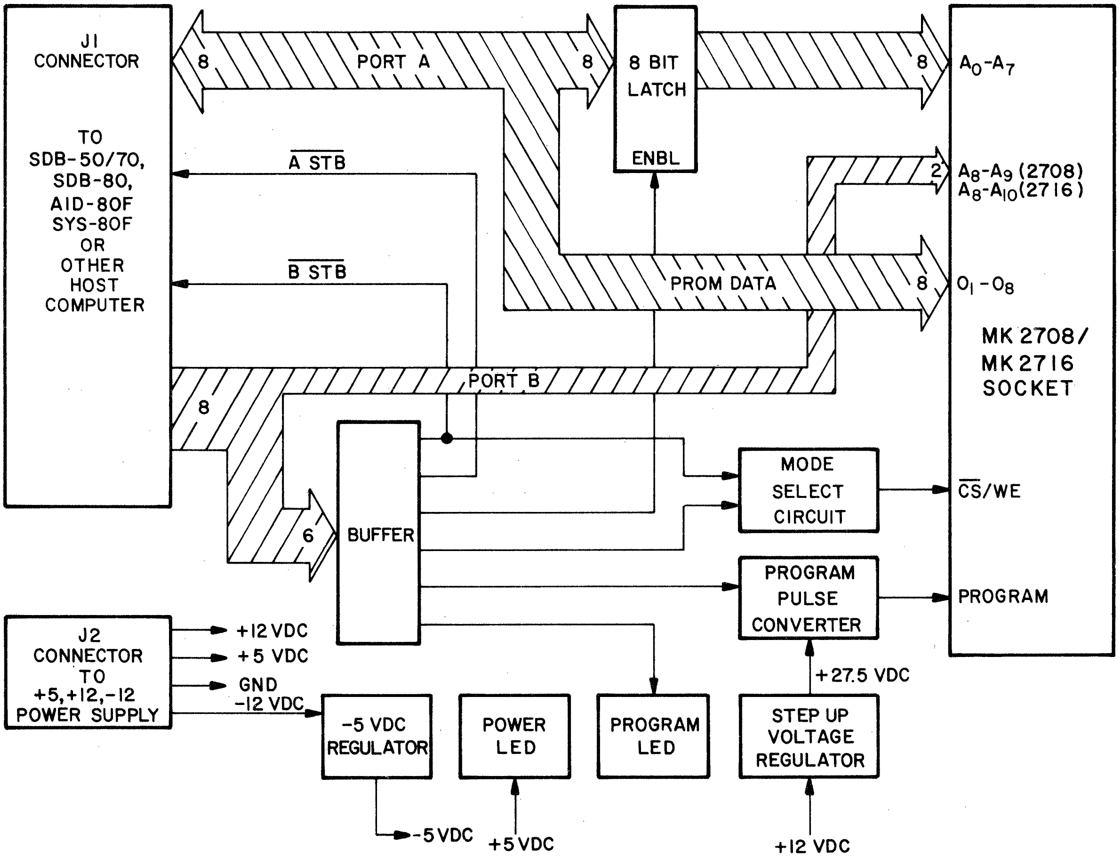
OPERATING TEMPERATURE

0°C - 50°C

PROGRAMMING TIME

2708	2.5 minutes
2758	0.9 minutes
2716	1.8 minutes

PPG 8/16 BLOCK DIAGRAM



ORDERING INFORMATION

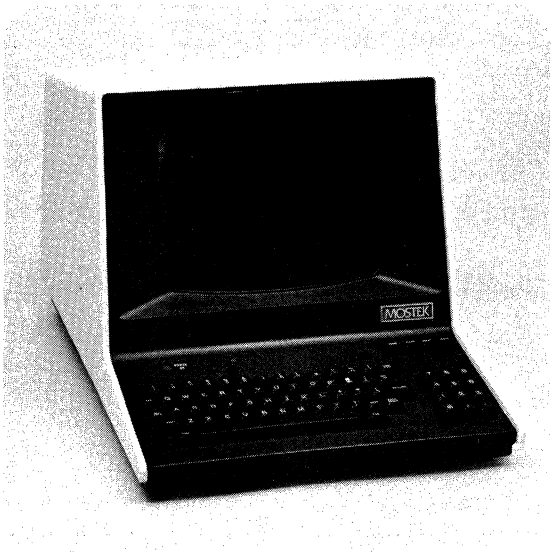
DESIGNATOR	DESCRIPTION	PART NO.
PPG-8/16*	PROM Programmer for 2708/2758/2716 PROMs with Operations Manual and paper tape drivers for SDB-50/70 and SDB-80. Does not include cables.	MK79081
	PROM Programmer for 2708/2758/2716 PROMs with Operations Manual and paper tape drivers for SDB-50/70 and SDB-80. Includes cable for interface to SYS-80F. (Europe only).	MK79082
XAID-805	PPG-8/16 Interface Cable for SDB-80	MK79041
XAID-705	PPG-8/16 Interface Cable forSDB-50/70	MK79046
MD-PPG	PPG-8/16 Interface Cable for MDX-PIO	MK77957
SWD-2	PPG-8/16 Object programs for SDB-50/70 & SDB-80 on Silent 700 cassette tape.	MK79084
PPG-RETRO	Installation package for installing PPG 8/16 into the AID-80F (also can be used for PPG-08 integration into AID-80F). Includes cable.	MK78154
	PPG-8/16 Operations Manual	MK79603

*NOTE: The PPG-8/16 will only program the 2708, 2758, and 2716 PROMs. The 2758 and 2716 are 5 Volt only type PROMs. **THE PPG-8/16 WILL NOT PROGRAM THE TI 2716 MULTIPLE VOLTAGE 2K x 8 PROM.**

Develop Systems US

CRT

FEATURES



- □ Interfaces directly to AID-80F or SYS-80F
- □ All 128 ASCII codes
- □ 94 displayable characters including lower case
- □ ANSI standard keyboard layout
- □ Separate numeric pad
- □ Dual intensity
- □ Cursor addressing and sensing
- □ EIA and 20 mA interface
- □ Baud rates up to 19.2KB
- □ Auxiliary EIA output
- □ Standard or reverse video

The MOSTEK CRT is a high-performance, keyboard display unit that is fully compatible with the AID-80F microcomputer system.

Characters for the MOSTEK CRT are formed using a TV raster-scan technique. Spacing of 2 dots between characters and 3 lines between upper case rows provides a highly legible character with excellent definition. The character set consists of 94 displayable upper and lower-case characters with lower-case descenders. The display may be switch-selected to be standard video (white on black) or reverse video (black on white).

The MOSTEK CRT can be interfaced to any computer system that provides a RS-232 or 20 mA serial asynchonous interface.

OPERATING CHARACTERISTICS

TERMINAL CONTROL

	Keyboard	Remote Command
CLEAR SCREEN	•	•
CLEAR FOREGROUND	•	•
CLEAR TO END OF LINE	•	•
CLEAR TO END OF SCREEN	•	•
CLEAR TO END OF SCREEN (BACKGROUND SPACES)		
AUDIBLE ALARM	•	•
BACKSPACE		•
KEYBOARD LOCK		•
KEYBOARD UNLOCK		•
INSERT LINE		•
DELETE LINE		•
TAB	•	•
*BACKGROUND FOLLOWS		•
**FOREGROUND FOLLOWS		•

*Background spaces are low intensity.
**Foreground spaces are high intensity.

CURSOR CONTROL

	Keyboard	Remote Command
CURSOR ADDRESS (XY)	•	•
INCREMENTAL CURSOR CONTROL	•	•
READ CURSOR ADDRESS	•	•
HOME CURSOR	•	•

SPECIFICATION

DISPLAY CHARACTERISTICS

Characters per line: 80
Lines per display: 24
Screen Capacity: 1920 characters
All 128 ASCII Codes
94 Displayable Characters Including Lower Case
Character size: 182 in. High x .088 in. Wide (nominal)
Refresh rate: 60 frames/sec
Display: P4 phosphor white on black or black on white
Cursor: Block
Dual Intensity

INTERFACE

Full or Half Duplex (W.E. modem 103A compatible or W.E. Modem 202C/D using character turnaround).

EIA RS232C connector, or Current Loop connector, 20mA externally sourced.

Eight Baud Rates: 110, 300, 1200, 2400, 4800, 9600, 19.2K

Parity: Odd, Even, 1 or 0

No. of Stop Bits: one (two at 110 Baud)

EXTERNAL CONTROLS

Contrast
Power On/Off
Half Duplex/Full Duplex
Auto LF/CR Control
Reverse Video or Standard Video
Lower Case Disable
Parity
Baud rate
EIA or Current Loop

ELECTRICAL

Power consumption: 115 watts nominal
Power input: 115V,60 Hz

MECHANICAL

Size (nominal): 13.5 in (34.3cm) high, 15.5 in (40 cm) wide, 20.5 in (52.1cm) deep

Weight: 35 lbs. (15.9 kg)

ENVIRONMENTAL

Temperature: 10°C to 40°C
Storage Temperature: -20°C to 65°C
Humidity: Up to 80% relative, non-condensing
Shock: Up to 40 g's on three axes (in carton)

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MOSTEK	Mostek CRT terminal featuring all 128 ASCII codes, 94 displayable characters including lower case, 80 characters by 24 lines, ANSI standard keyboard layout, cursor addressing and sensing, EIA and 20 Ma interface and Baud rates to 19.2K B. Includes RS-232 interface cable (MK79038).	MK78149
	AID-80F or SYS-80F Interface cable (available separately).	MK78152

Develop
Systems
US

MOSTEK®

MICROCOMPUTER SUPPORT

Line Printer

FEATURES

□ Interfaces directly to AID-80F

□ Prints 120 characters per second

□ Up to 132 characters per line

□ Prints original plus five copies

□ Character elongation

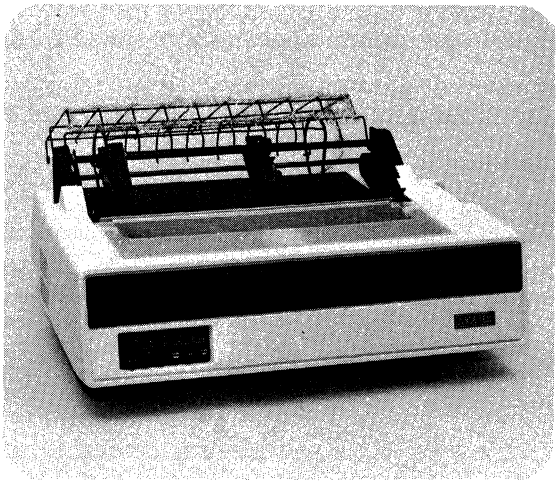
□ 8 inches per second paper slew rate

□ Ribbon cartridge

□ 7x7 dot matrix, 64 character ASCII

□ Tractor feed/Pin feed platen

□ Parallel interface



DESCRIPTION

The MOSTEK line printer is a state-of-the-art microprocessor controlled, dot-matrix line printer that prints at the rate of 120 characters per second. The printer has a maximum print width of 132 characters with a horizontal format of 10 characters per inch and 6 lines per inch vertical. Elogated (double width) characters are software selectable.

The Mostek line printer interfaces directly to the AID-80F or SYS-80F Microcomputer Systems and can be interfaced easily to other computer systems supporting parallel I/O.

SPECIFICATIONS

Print Performance - Minimum Throughout

Printer Model	Print Speed (cps)	Max. Print Width	10Char/ Line (lpm)	80Char/ Line (lpm)	132Char/ Line (lpm)
702	120	13.2 in. (335mm)	260	74	47

Characters
7x7 dot matrix, 64 character U.S. ASCII

Format
10 Characters per inch horizontal
6 Lines per inch vertical
Elongated (double width) characters software selectable

Forms Handling
Tractor feed, for rear or bottom feed forms

8 ips slew rate
Usable paper 4 in. (102mm) to 17.3 in. (439mm) width
Paper tension adjustment

Ribbon System
Ribbon cartridge
Continuous ribbon 9/16 in. (14mm) wide, 20 yards (18.3 meters) long.
Mobius loop allows printing on upper and lower portion on alternate passes.

Panel Indicators
Power On: Indicates AC power is applied to printer.
Select: Indicates printer can receive data.
Alert: Indicates operator-correctable error condition.

Operator Controls
Select/deselect
Forms thickness

Top of form
Horizontal forms positioning
Vertical forms positioning
Power ON/OFF
Single line feed
Paper empty override
Self-test

Internal Controls

Auto motor control: turns stepping motors off when no data is received.

Electronic top of form: allows paper to space to top of form when command is received.

Preset for 11 in. (279mm) or 12 in. (305mm) forms
Opt. VFU must be used for other form lengths.

Data Input

7 or 8 bit ASCII parallel; microprocessor electronics; TTL levels with strobe.
Acknowledge pulse indicates that data was received.

Electrical Requirements

50/60 Hz, 115/230 VAC;+10%/-15% of Nominal Tappable Transformer (100, 110, 115, 120, 200, 220, 230, 240 VAC).

Physical Dimensions

	Model 702
Weight:	60 lbs. (27 Kg)
Width:	24.5 in. (622mm)
Height:	8 in. (203mm)
Depth:	18 in. (457mm)

Temperature

Operating:	40° to 100°F (4.4° to 37.7°C)
Storage:	-40° to 160°F (-40° to 71.1°C)

Humidity

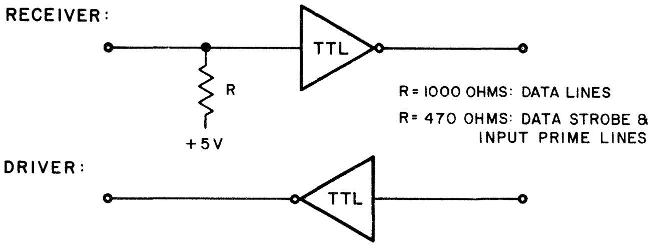
Operating:	20% to 90% (No condensation)
Storage:	5% to 95% (No condensation)

INTERFACING

INTERFACE DRIVERS AND RECEIVERS

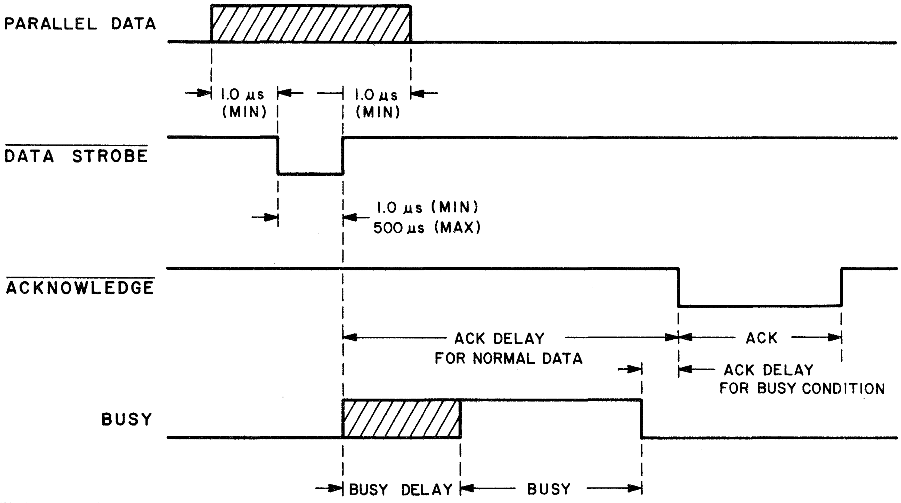
ALL INPUT/OUTPUT SIGNALS ARE TTL COMPATIBLE

LO: 0.4 VOLTS HI: 2.4 VOLTS



CONNECTOR: AMPHENOL 57. 40360 SERIES, 36-PIN
(CENTRONICS 31310019)

INTERFACING TIMING



INTERFACE TIMING CONT'D 702

NORMAL DATA INPUT TIMING	ACK DELAY	2 - 6 μsec.
	ACK	4 μsec.
BUSY CONDITION TIMING	BUSY DELAY	0 - 1.5 μsec.
	ACK DELAY	1 - 6 μsec.
	ACK	4 μsec.
	BUSY DURATION:	
	Line Feed	350 - 500 μsec.
	Vertical Tab (1-in.)	135 - 145 msec.
	Form Feed (11-in.)	1.48 - 1.50 sec.
	Delete	160 - 400 μsec.
	Bell	0
	Select*	0 - 1.5 μsec.
	Deselect	Until Printer is selected
	Printer	8.33 msec/char.; plus 148 msec. non-printing time/line.

*No busy if inhibit prime on select option is used.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MOSTEK LP	Mostek line printer featuring 120cps operation, 7x7 dot matrix, 10 cpi, and paper slew rate of 8 ips. Includes AID-80F cable.	MK78150
	AID-80F parallel interface cable for line printer (Centronics 702).	MK78153

Develop Systems US

MOSTEK®

MICROCOMPUTER SYSTEMS

AID-80F Cross Assembler for 3870/F8 (FZCASM)

FEATURES

□ Assembles all standard 3870/F8 family source statements

□ Object output in industry standard hexadecimal format extended for relocatable and linkable programs

□ Allows the following pseudo-ops:

ORG	- program origin
EQU	- equate label
DC	- define constant
DEFL	-. define label
DEFM	- define message
DEFB	- define byte
DEFW	- define word
DEFS	- define storage
END	- end statement
IF	- conditional assembly
ENDIF	- end of conditional assembly
INCLUDE	- include another dataset within current assembly
NAME	- program name definition
PSECT	- program section definition
GLOBAL	- global symbol definition

□ Supports the following assembler directive pseudo-ops:

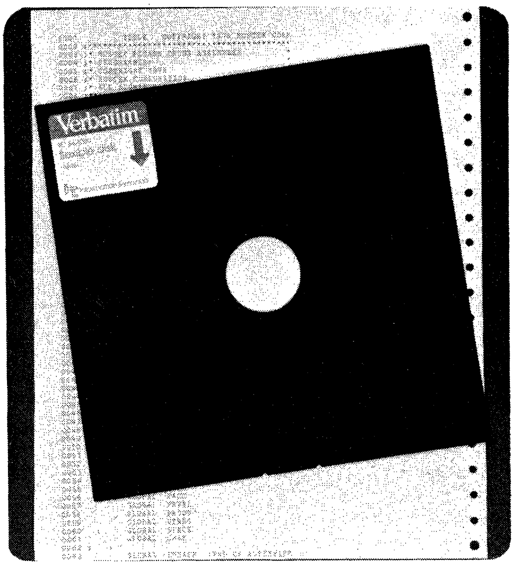
EJECT	- eject a page of listing
TITLE	- place heading at top of each page of listing
LIST	- turn listing on
NLIST	- turn listing off

□ Complete assembly in two passes with second pass repeatable

□ Size of program to be assembled limited only by memory available for symbol table

□ Supports conditional assembly, relocatable and linkable modules, symbol table and cross reference listings

□ Supplied on a standard FLP-80DOS diskette for use with the MOSTEK AID-80F floppy disk based development system



DESCRIPTION

The purpose of the 3870/F8 Cross Assembler is to assemble source language programs for the MOSTEK 3870 Series and F8 microcomputers. The Cross Assembler is designed to run on the MOSTEK AID-80F Dual Disk Development System with the FLP-80DOS operating system. The Cross Assembler is supplied on flexible diskette. The Assembler reads F8 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The assembly listing shows address, machine code, statement number, and source statement. The object code is in industry standard hexadecimal format modified for relocatable, linkable assemblies. A conversion utility (F8DUMP) is supplied to produce object code in F8 format for users of the MOSTEK SDB-50/70. The Assembler supports conditional assemblies, global symbols, relocatable programs, a printed symbol table and cross reference listing. It can assemble any length program limited only by a symbol table size of over 400 symbols. Expressions involving mathematical and logical operations are allowed. Conditional assembly allows the user to suspend assembly for a portion of the program depending upon the result of an expression. A global symbol is catagorized as "internal" if it appears as a label in the program; otherwise it is an "external" symbol. The printed symbol table and cross reference listing show which symbols are internal and which are external. The Cross Assembler allows the user to select relocatable or non-relocatable assembly via the "PSECT" pseudo-op. Relocation records are placed in the object output for relocatable assemblies.

The Assembler can be run as a single pass assembler or as a learning tool. (In this mode, global symbols and forward references are not allowed).

In conjunction with the FLP-80DOS Text Editor and Linker, FZCASM provides the means for editing, assembling and linking F8 or 3870 family programs.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NUMBER
FZCASM Cross Assembler	Includes the 3870/F8 Cross Assembler on a FLP-80 DOS system diskette, and the FZCASM Operations Manual.	MK79079
FZCASM Operations Manual	Describes in detail the operation of the 3870/F8 Cross Assembler	MK78582
AID-80F Data Sheet	Describes the MOSTEK AID-80F Dual Disk Development System	MK78568
FLP-80DOS Data Sheet	Describes the operating system used on the AID-80F System	MK78556
FLP-80DOS Operations Manual	Describes in detail the software and operating system used to run FZCASM on the AID-80F System	MK78557
AIM-72 Data Sheet	Describes the MOSTEK AIM-72 Application Interface Module used to provide in-circuit emulation capability for 3870 series Microcomputers on the AID-80F.	MK79576

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold
on condition that the Purchaser agrees to
the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.

4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTIC–ULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject To This Agreement:

Order Number	Description	Price*

Ship To: ____________________

Bill To: ____________________

Method of Shipment: ____________________

Customer P.O. Number: ____________________

Agreed To:

PURCHASER	MOSTEK CORPORATION
By: ____________	By: ____________
Title: ____________	Title: ____________
Date: ____________	Date: ____________

* Prices Subject To Change Without Notice

Develop Systems US

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

BASIC Software Interpreter

FEATURES

□ Direct access to CPU I/O Ports

□ Ability to read or write any memory location (PEEK, POKE)

□ Arrays with up to 255 dimensions

□ Dynamic allocation and de-allocation of arrays

□ IF...THEN...ELSE and nested IF...THEN...ELSE

□ Direct (immediate) execution of statements

□ Error trapping, with error messages in English

□ Four variable types: Integer, string, real and double precision real.

□ **Full PRINT USING capabilities for formatted output**

□ Extensive program editing facilities

□ Trace facilities

□ Can call up to 10 assembly language subroutines

□ Boolean (logical) operations

□ **Supports up to 6 sequential and random access files** on floppy disk.

□ **Complete set of file manipulation statements**

□ Occupies only 19K bytes, not including operating system

□ Supports console and line printer I/O

□ Allows console output to be redirected to the line printer

DESCRIPTION

MOSTEK BASIC is an extensive implementation of Microsoft BASIC for the Z80 microprocessor. Its features are comparable to those BASICs found on minicomputers and large mainframes. Mostek BASIC is among the fastest microprocessor BASICs available.

Designed to operate on the Mostek Dual Disk Development System with FLP-80DOS and 32K bytes or more memory. Mostek BASIC provides a sophisticated software development tool.

Mostek BASIC is implemented as an interpreter and is highly suitable for user interactive processing. In a 32K byte system, about 7K bytes of free storage area are available to the user. Programs and data are stored in a compressed internal format to maximize memory utilization. By adding more memory to the system, the user's program and data storage area may be increased to as much as 31K bytes.

Unique features include long variable names, substring assignments and hexadecimal and octal constants. Many other features ease the task of programming complex functions. The Programmer is seldom limited by array size (up to 255 dimensions, with run time allocation and deallocation) or I/O restrictions. Full PRINT USING capabilities allow formatted output, while both input and output may be performed with multiple sequential and random files on floppy disk as well as with the CPU I/O ports. Editing, error trapping, and trace facilities greatly simplify program debugging.

Develop Systems US

Commands:

AUTO	CLEAR	CONTST	DELETE	EDIT
FILES	LIST	LLIST	LOAD	MERGE
NEW	NULL	RENUM	RESET	RUN
SAVE	SYSTEM	TRON	TROFF	WIDTH

Program Statement:

DEFNx	DEFDBL	DEFINT	DEFSNG	DRFSTR
DIM	END	ERASE	ERROR	FOR
GOSUB	GOTO	IF...THEN(ELSE)	LET	NEXT
ON...ERROR	ON...GO SUB	ON...GOTO	OUT	POKE
REM	RESUME	RETURN	STOP	SWAP
WAIT				

Input/Output Statements:

CLOSE	DATA	FIELD	GET	INPUT
KILL	LINEINPUT	LSET	NAME	OPEN
PRINT	PUT	READ	RESTORE	RESET

Operators

=	-	+	*	/
↑	\	MOD	NOT	AND
OR	XOR	IMP	EQV	<
>	<=	>=	<>	

Arithmetic Functions

ABS	ATN	CDBL	CINT	COS
CSNG	ERL	ERR	EXP	FRE
INP	INT	LOG	LPOS	PEEK
POS	RND	SGN	SIN	SPC
SQR	TAB	USRn	VARPTR	

String Functions

ASC	CHR$	FRE	HEX$	INSTR
LEFT$	LEN	MID$	OCT$	RIGHT$
SPACE$	STRING$	STR$	VAL	

Input/Output Functions

CVD	CVI	CVS	EOF	LOC
LOF	MKD$	MKI$	MKS$	

In order to receive Mostek BASIC, the attached Mostek BASIC Non-disclosure agreement should be signed and returned with each purchase order.

ORDERING INFORMATION		
Designator	**Description**	**Part No.**
MOSTEK BASIC	BASIC INTERPRETER high-level Language to run on FLP-80DOS. Requires 32K bytes of memory.	MK78157
AID-80F	Floppy disk development system for Z80 and and 3870/F8 systems. Includes FLP-80DOS and 32K bytes of RAM	MK78125

MOSTEK BASIC NON-DISCLOSURE AGREEMENT

The party below agrees that it is receiving a copy of Mostek BASIC for use on a single computer only, as designated on its registration card. The party agrees to fill out and mail in the registration card before making use of Mostek BASIC. The party agrees that all copies will be strictly safeguarded against disclosure to or use by persons not authorized by Mostek to use Mostek BASIC, and that the location of all copies will be reported to Mostek at Mostek's request. The party agrees that is agreement shall insure to the benefit of any third party holding any right, title or interest in the Mostek BASIC cr any software from which it was derived.

"Party"________________ (Date)________________
Company:________________________________
Address:________________________________
Phone:________________________________

Return to:

Mostek Corp.
Microcomputer Dept.
Software Librarian
MS 501
1215 W. Crosby Road
Carrollton, Texas 75006

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

FORTRAN IV Compiler

FEATURES

□ All of ANSI standard FORTRAN IV (X3.9-1966) except complex data type.

□ Generates relocatable linkable object code.

□ Subroutines may be compiled separately and stored in a system library.

□ Compiles several hundred statements per minute in a single pass.

□ Enhancements include

1. LOGICAL variables which can be used as integer quantities
2. LOGICAL DO loops for tighter, faster execution of small valued integer loops.
3. Mixed mode arithmetic.
4. Hexadecimal constants.
5. Literals and Holleriths allowed in expressions.
6. Logical operations on integer data. .AND., .OR., .NOT. and .XOR. can be used for 16-bit or 8-bit Boolean operations.
7. READ/WRITE End-of-File or Error Condition transfer. END=n and ERR=n (where n is the statement number) can be included in READ or WRITE statements to transfer control to the specified statement on detection of an error or end-of-file condition.
8. ENCODE/DECODE for FORMAT operations to memory.

□ Long descriptive error messages.

□ Extended optimizations

□ Z80 assembly language subprograms may be called from FORTRAN programs

DESCRIPTION

Mostek's FORTRAN IV Compiler package provides new capabilities for users of Z80–based microcomputer systems. Mostek FORTRAN is comparable to FORTRAN compilers on large mainframes and minicomputers. All of ANSI Standard FORTRAN X3.9-1966 is included except the COMPLEX data type. Therefore, users may take advantage of the many applications programs already written in FORTRAN.

Mostek FORTRAN IV is unique in that it provides a microprocessor FORTRAN development package that generates relocatable object modules. This means that only the subroutines and system routines required to run FORTRAN programs are loaded before execution. Subroutines can be placed in a system library so that users can develop a common set of subroutines that are used in their programs. Also, if only one module of a program is changed, it is necessary to re-compile only that module.

The standard library of subroutines supplied with FORTRAN includes:

ABS	IABS	DABS	AINT
INT	IDINT	AMOD	MOD
AMAXO	AMAX1	MAXO	MAX1
DMAX1	AMINO	AMIN1	MINO
MIN1	DMIN1	FLOAT	IFIX
SIGN	ISIGN	DSIGN	DIM
IDIM	SNGL	DBLE	EXP
DEXP	ALOG	DLOG	ALOG10
DLOG1O	SIN	DSIN	COS
DCOS	TANH	SQRT	DSQRT
ATAN	DATAN	ATAN2	DATAN2
DMOD	PEEK	POKE	INP
OUT			

The library also contains routines for 32-bit and 64-bit floating point additon, subtraction, multiplication, division, etc. These routines are among the fastest available for performing these functions on the Z80.

A minimum system size of 48K bytes (including FLP-80DOS) is required to provide efficient optimization. The Mostek FORTRAN compiler optimizes the generated object code in several ways:

1. Common subexpression elimination. Common subexpressions are evaluated once, and the value is substituted in later occurrences of the subexpression.

2. Peephole Optimization. Small sections of code are replaced by more compact, faster code in special cases.

3. Constant folding. Integer constant expressions are evaluated at compile time.

4. Branch Optimizations. The number of conditional jumps in arithmetic and logical IFs is minimized.

Long descriptive error messages are another feature of the compiler. For instance:

?Statement unrecognizable

is printed if the compiler scans a statement that is not an assignment or other FORTRAN statement. The last twenty characters scanned before the detected error are also printed.

As an option, the compiler generates a fully symbolic listing of the machine language to be generated. At the end of the listing, the compiler produces an error summary and tables showing the addresses assigned to labels, variables and constants.

LINKER

A relocating linking loader (LINK-80) and a library manager (LIB-80) are included in the Mostek FORTRAN package.

LINK-80 resolves internal and external references between the object modules loaded and also performs library searches for system subroutines and generates a load map of memory showing the locations of the main program, subroutines and common areas.

LIBRARY MANAGER

LIB-80 allows users to customize libraries of object modules. LIB-80 can be used to insert, replace or delete object modules within a library, or create a new library from scratch. Library modules and the symbol definitions they contain may also be listed.

CP/M UTILITY

A utility program (XCPM) is included which allows the user to copy FORTRAN source programs from CP/M diskettes to FLP-80DOS diskettes. At this point the programs can be compiled using the Mostek FORTRAN compiler.

FTRANS UTILITY

FTRANS allows the user to convert object programs produced by the Mostek Z80 assembler to a form that is linkable to FORTRAN programs.

DESIGNATOR	DESCRIPTION	PART NO.
MOSTEK FORTRAN IV	FORTRAN IV high level compiler to run on FLP-80DOS. Requires 48K bytes of RAM. Includes Operations Manual.	MK78158
Mostek FORTRAN IV User's Manual	Operations Manual only	MK79643

MOSTEK®

Z80 MICROCOMPUTER SYSTEM

FLP-80DOS Software Library Vol. 1 (LIB-80-V1)

FEATURES

□ Includes 23 useful subroutines and programs for the Z80, including:

- Lawrence Livermore Lab's Basic
- Generalized sort program for up to 8 fields per record
- 8080 - Z80 source code converter
- Fast disk-to-disk copy utility
- Hexadecimal Dump Utility to dump memory on files
- Assembly Language Formatter Utility to format Z80 source into columns
- Word Processor Program Version 2.0, used to format documents
- Disk Recovery Utility used to recover bad disk files

□ All programs are supplied in source, object, and binary format with complete documentation on a standard FLP-80DOS diskette.

□ Requires FLP-80DOS Version 2.0 or higher.

DESCRIPTION

The Mostek FLP-80DOS Software Library is a collection of programs of general utility that run under FLP-80DOS Version 2.0 or higher. These programs are used quite extensively at Mostek. They are being offered in source format on diskette so that the user may not only use them as supplied, but may use them as a base for individually tailored software.

This software library differs from other libraries in that all programs in the library have been developed or modified in-house. All programs in the library are in use at Mostek and all have some utility.

The FLP-80DOS Software Library Volume 1 consists of a User's Guide and two diskettes containing the source and binary (or object for subroutines) forms for each one of the twenty-three included programs. In order to reduce the cost of the library, printed source listings are not supplied. The user can obtain a source listing easily by assembling the required source program. A brief User's Guide is a part of each program source.

The FLP-80DOS Software Library is a "Level 2" product. "Level 2" software products are supplied by Mostek but are not supported in the areas of technical assistance or updates.

ORDERING INFORMATION

PART NO.	DESCRIPTION
MK78164	LIB-80 Volume 1 - FLP-80DOS Software Library, including source, object, and binary formats on diskette, and a printed user's guide.

1979 MICROCOMPUTER DATA BOOK

Microcomputer Development System (SYS-80FT)

INTRODUCTION

The Mostek SYS-80FT is a complete state of the art disk based computer. Not only does it provide all the necessary tools for software development but it provides complete hardware/software debug through Mostek 's AIM series of in-circuit emulation cards for the Z80, the 3870 family and future Mostek microprocessors.

The disk based system eliminates the need for other mass storage media but provides ease of interface to any peripheral normally used with computers. The file oriented floppy disk structure for data storage and retrieval provides a reliable, portable media to speed and facilitate software development.

The FLP-80DOS Disk operating system is designed for maximum flexibility both in use and expansion to meet a multitude of end user or OEM needs.

Because FLP-80DOS is supported by Mostek's SD and MD Series of OEM boards, software designed on the SYS-80FT can be directly used in OEM board applications.

DEVELOPMENT SYSTEM FEATURES

The SYS-80FT is an excellent integration of both hardware and software development tools for use throughout the complete system design and development phase. The software development is begun by using the combination of Mostek's Text Editor with « roll in - roll out » virtual memory operation and the Mostek relocating assembler. Debug can then proceed with the SYS-80FT using its resources as if they were in the final system. Using combinations of the Monitor, Designer's Debugging Tool, execution time breakpoints, and single step/multistep operation along with a formatted memory dump provides control for attacking those tough problems. The use of the Mostek AIM option provides extended debug with versatile hardware breakpoints on memory or port locations, a buffered in-circuit emulation cable for extending the software debug into its own natural hardware environment, as well as a history memory to capture bus transactions in real time for later examination and external probes for analysis of signals not directly related to the CPU bus. The relocatable and linking feature of the assembler enables the use of contemporary modular design techniques whereby major system alterations can be made in

Develop Systems Europe

small tractable modules. Using the linker, the small modules can be combined to form a run time module without major reassembly of the entire program.

PACKAGED SYSTEM FEATURES

From a system standpoint, the SYS-80FT has been designed to be the basis of an end product small business/industrial computer. The flexibility provided in the FLP-80DOS operating system permits application programs to be as diverse as a high level language compiler or a supervisory control system in the industrial environment. Other hardware options are available, with even more to be added.

Expansion of the disk drive units to a total of four single sided or double sided units provide up to two megabytes of storage. This computer uses the third generation Z80 processor supported with the power of a complete family of peripheral chips. Through use of its 158 instructions (including : 16-bits arithmetic, bit manipulation, advanced block moves and interrupt handling), almost any application from communication concentrators to general purpose accounting systems is made easy.

OEM FEATURES

The hardware and software basis for the SYS-80FT is also available separately to the OEM purchaser. Through a software licensing agreement, all Mostek Software can be utilized on these OEM series of cards. A growing line of support cards and card cages, permits the user to configure a multitude of different systems.

SYS-80FT RESIDENT SOFTWARE (FLP-80DOS)

A totally integrated package of resident software is offered in conjunction with the SYS-80FT consisting of :

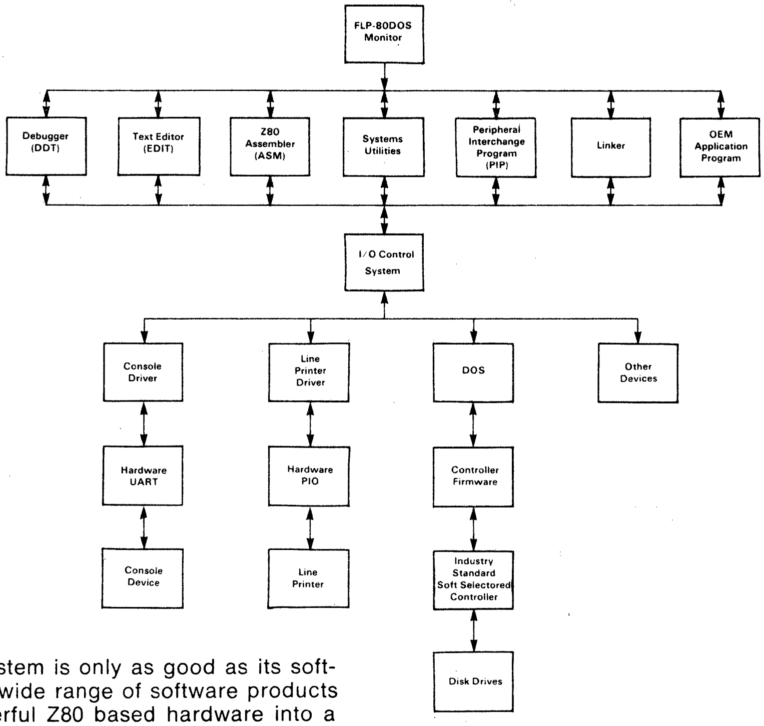
Standard software

Monitor
DDT-80 with extended debug through AIM-80
Text Editor
Z80 Relocating Assembler
Peripheral Interchange Program
Linker
I/O control system
Floppy disk handler
Device driver library
Batch mode operation

Optional software includes :

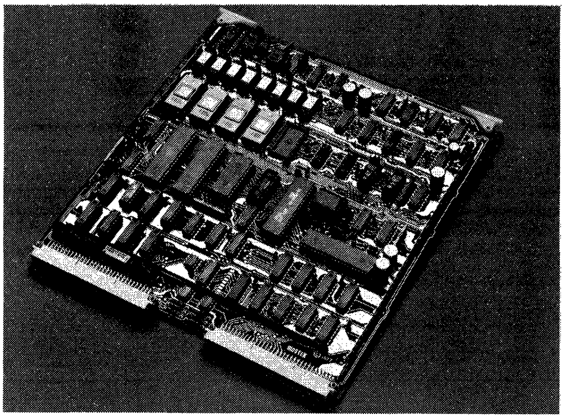
Macro Assemblers Z80 + 3870
Basic
Fortran
Utility package
Block structured high level language

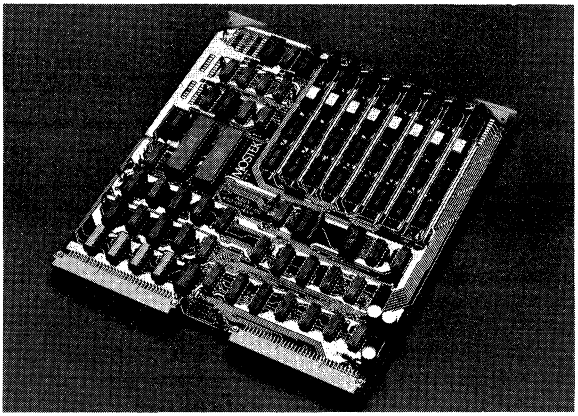
SOFTWARE BLOCK DIAGRAM

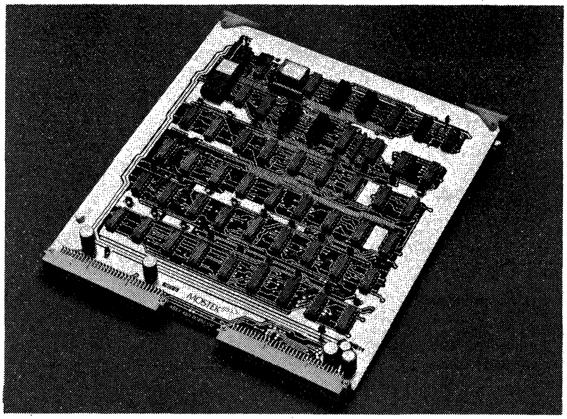


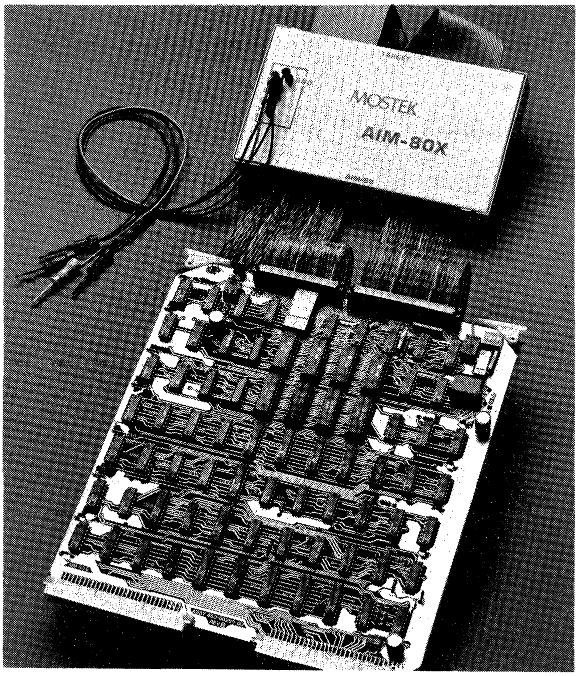
A microcomputer system is only as good as its software. Mostek has a wide range of software products which turn the powerful Z80 based hardware into a system of minicomputer capabilities.

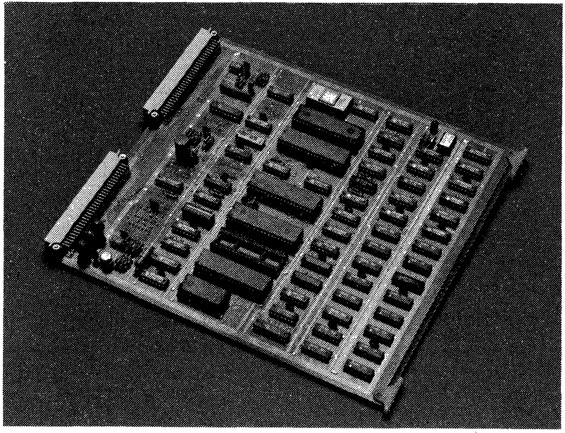
Develop Systems Europe

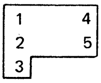












1- SDB - 80E
2- FLP - 80E
3- VDI - S/P
4- RAM - 80BE
5- AIM - 80E

Monitor

The FLP-80DOS Monitor is the environment from which all activity in the system initiates. From the Monitor ,any system routine such as PIP or a user generated program is begun by simply entering the program name. FLP-80DOS I/O is done in terms of logical unit numbers, as is commonly done in FORTRAN. A set of logical units are preassigned to default I/O drivers upon power or reset. From the console the user can reassign any logical unit to any new I/O device and can also display logical unit assigments. Executable file creation can be done by the Save command, and Hex object files can be produced using the Dump command. User generated binary-files can be loaded with the GET command or can be directly executed by entering the program name.

Text Editors

The Text Editors permit editing/creating of any source file independent of the language being written.
Two are provided. The Editor for TTY type I/O.
The Video Editor for CRT I/O.
The Editor is both line and string oriented to give maximum utility and user flexibility.
The Editor through its virtual memory "roll in roll out" technique can edit a file whose length is limited only by maximum diskette storage.
Included in the repertoire of 16 commands are macro commands to save time when encountering redundant editing tasks. The Editor is also capable of performing in one operation all the commands which will fit into an 80 column command buffer.

Summary of Editor commands

Advance N — Advance line pointer N line
Backup N — Backs up N lines
Change N/S1/S2 — Change N occurrences of string 1 to string 2
Delete N — Delete current line plus next N-1 lines of text
Exchange N — Exchanges current line plus next N-1 lines with lines to be inserted while in insert mode.
Get — Inserts a file from the disk into the text.
Insert — Place Editor in insert mode. Text will be inserted after present line.
Line N — Place line pointer on Line N.
Macro 1 or Macro 2 — Defines Macro 1 or Macro 2 by the following string of text Editor commands.
Put — Writes a section of code to the disk.
Quit — Stores off file under editing process and returns to Monitor environment.
Search N/S 1 — Search from existing pointer location until Nth occurrence of string S1 is located and print it.
Top — Inserts records at top of file before first line.
Verify N — Print current record to console plus next n-1 records while advancing pointer N records ahead.
Window N — Prints current records plus next N-1 records to source output device while advancing pointer N records.
eXecute N — Executes Macro 1 or Macro 2 as defined by Macro command.

The video Editor displays the text to be edited directly on the CRT screen as if it were a window into memory.
The window and cursor need only be positioned over the character to be changed. Then the new text can be added or old deleted with the changed data dispayed immediatly on the screen. The video Editor allows programs (or any text) to be entered or corrected much quicker and easier than possible with TTY oriented Editors.
The video Editor provides all the commands of the Editor except E,I,M,V,W,X.

Z80 Assembler

The Z80 Resident Assembler generates relocatable or absolute object code from source files independent of source medium. The assembler recognizes all 158 Z80 instructions as well as 20 powerful pseudo operators. The object code generated is industry standard absolute or relocatable format. With the relocating feature, large programs can easily be developed in smaller sections and linked using the system Relocating Linking Loader or Linker. Because the assembler utilizes the I/O Control System, object modules or list modules can be directed to disk files, paper tape, console, or line printer. Portability of output media eliminates the requirement for a complete set of peripherals at every software/hardware development system.

The assembler run time options include sorted symbol table generation, no list, no object, pass 2 only, quit, cross reference table and reset symbol table. The assembler is capable of handling 14 expression operators including logical, shift, multiplication, division, addition and subtraction operations.
These permit complex expressions to be resolved at assembly time by the assembler rather than manually by the programmer. Comments can be placed anywhere but must be preceded by a semicolon. Error messages are integrated with listing file but can be directed to console device. Additions to standard assembler pseudo operators are :

GLOBAL — For global definition.
PSECT operator — To generate relocatable or absolute modules.
IF expression — Conditional assembly IF expression is true.
INCLUDE dataset — To include other datasets (files) as in-line code anywhere in source file.

Peripheral Interchange Program

PIP provides complete file maintenance activity for operations such as copy file from disk to disk, disk to peripheral, or any peripheral to any other peripheral supporting both file structured and character oriented devices. Key operations such as renaming, appending, and erasing files also exist along with status commands for diskette ID and vital statistics. PIP can search the diskette directories for any file or a specific name, extension, and user number. The PIP operations are :

Append — Appends file 1 to file 2 without changing file 1.

Copy — Copies input files or data from an input device to an output file or device. The Copy command can be used for a variety of purposes such as listing files, concatenating individual files, or copying all the files or a single file from one disk unit (e.g. DK0) to a second disk unit (e.g. DK1).

Directory — Lists the directory of a specified disk unit (DK0, DK1, etc...). The file name, extension, and user number is listed for each file in the directory. The user can also request listing only files of a specified name, only files of a specified extension or only files of a specified user number. The list device can be any device supported by the system as well as a file.

Erase — Erases a single file or files from a diskette in a specified disk unit. The user has the option to erase all files, only files of a specified file name, only files of a specified extension or only files of a specified user number.

Format — Takes completely unformatted soft sectored diskettes. Formats to IBM 3740, and prepares to be system diskette. Operation is performed on diskette unit 1 and a unique 11 character name is assigned to that diskette.

Init — Initializes maps in disk handler when a new diskette has been changed while in the PIP environment.

Rename — Renames a file, its extension, and its user number to a file of name X, extension Y, and user Z.

Status — Lists all vital statistics of a disk unit to any list device. These include number of allocated records, number of used records, and number of bad records.

Quit — Returns to Monitor Environment.

DOS/Disk Handler

The heart of the FLP-80DOS software package is the Disk Operating System. Capable of supporting 4 double sided units, the system provides a file structured orientation timed and optimized for rapid storage and retrieval. Thorough error reporting exists from the DOS to enable an application programmer to quickly debug his program as wel as extensive error recovery and bad sector allocation which insures data and file integrity. The DOS not only provides file reading and writing capability but special pointer manipulation, record deletions, record insertions, skip records both forward and backward as well as directory manipulation such as file creation, renaming,and erasure. The DOS is initiated by a calling vector which is a subset of the I/O control system vector or through the standard IOCS calling sequence to elect buffer allocation, blocking and deblocking of data to a user selectable logical record type.

A unique dynamic allocation algorithm makes optimal use of disk storage space. Run time (Binary Files) are given first priority to large blocks of free space to eliminate any such overhead in operation system and overlay programs. The algorithm marks storage fragments as low priority and uses them only when diskette is nearing maximum capacity. The DOS permits 7 files to be active at any one time, thus permitting complex application programs as well as multiuser operation of the DOS.

I/O Control system

The I/O control system provides a central facility from which all calls to I/O can be structured. This permits a system applications program to dissolve any device dependence by utilizing the logical unit approach of large main frame computers. For example, a programmer may want to structure his utility to use logical unit no 5 as his list device which normally in his system defaults to the line printer. He may, however, assign at run time a different device for logical unit No 5. His application program remains unchanged.

Interface by a user to IOCS is done simply be entering a device mnemonic in a table and observing the calling sequence format. IOCS supplies a physical buffer of desired length, handles buffer allocation, blocking, deblocking and provides a logical record structure as specified by the user.

DDT

The Designer's Debugging Tool consists of commands for facilitating an otherwise difficult debugging process. The SYS-80FT support of fast program modifications through editing and re-assemblies, followed by DDT operations close the loop on the debug cycle. The DDT commands include :

Memory — Display, update, or tabulate memory.

Port — Display, update or tabulate I/O ports.

Execute — Performs 16 bit add/sub.

Hexadecimal — Execute user's program.

Copy — Copy one block to another.

Breakpoint — Set software trap in user code for interrupting execution in order to examine CPU registers.

Register — Display contents of user's registers.

Offset — Enter address adder for debug of relocatable modules.

Fill — Fill specified portion of memory with 8 bit byte.

Verify — Compare two blocks of memory.

Walk — Software single step/multistep.

Quit — Return to Monitor.

Linker

The Linker program provides the capability of linking assembler generated absolute or relocatable object modules together to create a binary or run time file. The linking process includes the library search option that if elected, will link in standard library object files (device drivers, math pack functions IOCS features) on disk to resolve undefined global symbols. By selecting an option a complete cross reference table will be generated and stored in a separate file, a list of undefined global symbols will be printed, and/or the global symbol table will be generated and stored in the same file as the cross reference symbol table.

Batch Mode Operation

In Batch Mode Operation, a command file is built on disk or assigned to a peripheral input device such as a card reader. The console input normally taken from the keyboard is taken from this batch device or batch file.

While operating under direction from a batch file, either the console device prompts the user or the prompting can be directed to any other output device. The Batch file definable operation is especially useful to execute rendundant procedures not requiring constant attention of the operator and to allow several programmers to use one system.

SYSTEM SPECIFICATIONS

- □ Z80 CPU.
- □ 4K byte PROM bootstrap and Z80 debugger.
- □ 58K bytes RAM.
- □ 4K bytes PROM bootstrap
- □ 8 x 8 bit I/O ports (4 x PIO) with user definable drivers/receivers.
- □ Serial port. RS 232 and current loop.
- □ 4 channel counter/timer (CTC).
- □ 2 disk drives 250K bytes per floppy disk.
- □ 80 x 24 display terminal with Cursor Addressing and Inverse Video. Full ASCII keyboard.
- □ 3 free positions for AIM modules, A/D cards, Modem interfaces, etc.
- □ Cross Assembler and debug ability for 3870 and future Mostek microprocessors.
- □ Device drivers for paper tape readers, punches, card readers, line printers, Silent 700's, teletypes and CRT's are included. Others can be added.
- □ PROM programer I/O port. Programer itself is optional.
- □ Bus compatible with Mostek SD series of OEM boards.

HARDWARE DESCRIPTION

CPU Module (OEM-80E)

The OEM-80 provides the essential CPU power of the system. While using the Z80 as the central processing unit, the OEM-80 is provided with other Z80 family peripheral chip support. Two Z80 PIO's give 4 completely programmable 8 bit parallel I/O ports with handshake from which the standard system peripherals are interfaced. Also on the card is the Z80-CTS counter time circuit which as 3 free flexible channels to perform critical counting and event counter timing functions. Along with 16K of RAM, the OEM-80 provides 5 ROM/PROM sockets which can be utilized for 10/20K of ROM or 5/10K PROM. Four sockets contain the firmware portion of FLP-80DOS. The remaining socket can be strapped for other ROM/PROM elements.

RAM-80BE

The RAM 80B adds additional memory with Mostek's MK 4116 16K dynamic memory along with more I/O. The four fully programmable 8 bit I/O ports with handshake provide the additional I/O ports in the system.

FLP-80E

Integral to the SYS-80FT system is the floppy controller. The FLP-80 is a complete IBM 3740 single density/double sided controller for up to 4 drives. The controller has 128 bytes of FIFO buffer resulting in a completely interruptable disk system.

AIM80E + AIM-80/4

The AIM-80 module provides extended debug for the SYS-80FT in Z80 development, real time in-circuit emulation permits debug of the hardware and the software at the most intimate level. Hardware single step/multistep with register trace, execution intercept on memory access, port access, or external trigger provides the absolute control over any system no matter how complex. The "pushbutton intercept" enables the programmer to perform a controlled recovery for those extremely difficult to trace processor lock out loops. With the memory clock selectable history module, any past 256 events of data, address, or control bus operation are captured in real time and displayable.

The AIM-80 includes 8K bytes of ROM firmware introducing unique software including a mnemonic disassembler for inverse assembly of history module contents or single step/multistep operations. "In line" code disassembled to language mnemonics provides insight into execution results as if examining an assembler generated listing. Extra added capability is the ROM resident self test of OEM-80 or target RAM.

AIM-80/4 provides upgraded operation to 4.0 MHz.

AIM - 72E + AIM - 73

The AIM-72 module provides debug and in-circuit emulation capabilities for the 3870 series microcomputers (3870, 3872 and 3876) on the SYS-80FT. Multiple breakpoint capability and single step operation allows the designer complete control over the execution of the 3870 Series microcomputer.

Register and Port display and modification capability provides information needed to find system "bugs" A11 I/O is in the user's system connected to AIM-72 by a 40-pin interface cable.

The debugging operation is controlled by ZAIM-72 a mnemonic debugger which controls the interaction between the Z80 host computer and the 3870 slave. For 3873 debugging, the AIM-73 can be used. It includes a history module for the last 1024 CPU cycles and also supports all 3870 family circuits.

Assembly and linking is done using the Macro-70 Assembler and the standard FLP-80DOS linker.

MECHANICAL SPECIFICATIONS

SYS-80FT Enclosure

Overall Dimensions : (in cm) 60 w x 42 d x 29 h.
Weight : 40 kg.
Fan capacity : 110 CFM.
Card Cage Capacity : 7 Double Eurocards.
Card Connectors : Din 41612.
Operating Temperature Range : + 10° C to + 35° C.
Power Supply.
Input : 230 V ± 10 % - AC - 50/60 Hz.
Outputs : + 5 VDC at 15 Amps Max.
+ 12 VDC at 1.2 Amps Max.
— 12 VDC at 1.2 Amps Max.
+ 24 VDC at 3 Amps Max.

STANDARD SOFTWARE LICENSING AGREEMENT

All Mostek Corporation products are protected by copyright and are sold on condition that the Purchaser agrees to the following terms :

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to : object code, source code and program listings.
2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.
3. The Purchaser may not incorporate Mostek software products or any portions thereof into his/her own products without written permission of Mostek.
4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.
5. In no event will Mostek Corporation be held liable for consequential damage resulting from the use or misuse of any Mostek software product.
   Due to the expense of generating software, Mostek must have a copy of this agreement, signed by the customer, before shipping certain software products to the customer. Additionally, the above rules apply to all Mostek software, regardless whether or not a signed license has been received. Signing the license confirms the customer's understanding, and allows Mostek to comply with point 4.

Order Number   Description

1)____

2)____

3)____

4)____

5)____

Agreed to :

PURCHASER	MOSTEK CORPORATION
By : ________________	By : ________________
Title : ________________	Title : ________________
Date : ________________	Date : ________________
Address : ________________	

ORDERING INFORMATION

SOFTWARE

Name	Description	Part No.
FLP-80DOS	Floppy disk operating system for OEM use.	Contact Mostek
MACRO-80	Most powerful Z80 Macro Assembler available for microcomputer development systems.	MK78165
MACRO-70	Syntax compatible to the Macro-80 this 3870(F8) Macro Assembler eases the development of 3870 programs.	MK79085
FZCASM	Non-Macro 3870 assembler.	MK79075
BASIC	17K byte disk based extended scientific BASIC interpreter. Random disk access.	MK78157
FORTRAN	24K Fortran IV compiler with library manager, library package and utilities.	MK78158
LIBRARY	Collection of FLP-80DOS utilities and other programs includes SORT, 8080 to Z80 source code translator, word processor printer, LLL basic (6K), 23 total programs including source, object and binary files.	MK78164

TO BE AVAILABLE SOON PASCAL; PLM.

SUPPORT

Mostek has in each office one or more application engineers ready to discuss detailed microprocessor application questions with your engineers.

Additionally, Mostek representatives in European countries have application engineers who are well trained in Mostek's product line.

Each office has demonstration equipment for your introduction to the Mostek family.

These offices also stock equipment for quick delivery.

HARDWARE

Name	Description	Part No.
SYS-80FT	Complete dual disk system with 58K Ram, standard software and 30 x 24 display terminal.	MK78042
SYS-80F	Complete dual disk system with 58K Ram, standard software no terminal.	MK78134
AIM-80E	In application debug module for the MK3880 (Z80) microprocessor.	MK78106
AIM-72E	In application debug module for the 3870, 3872 and 3876 single chip microprocessors.	MK79077
PPG-8/16	2708, 2758, 2716 PROM programmer.	MK79082
Line Printer	120CPS matrix printer and cable for connection to the SYS-80F(T).	MK78150 MK78098
A/D-80	Family of A/D converter cards.	MK78172-7
VDI-P	CRT interface for additional displays to the SYS-80F.	MK78035

TO BE AVAILABLE SOON

AIM/80-4	Enhanced 3880 (Z80) AIM.
AIM-73	3870, 72, 73, 76 AIM with enhanced trace facilities.
HDC-80	Hard disk interface and software for improved system troughout in high level language applications, and more on line data in business applications.

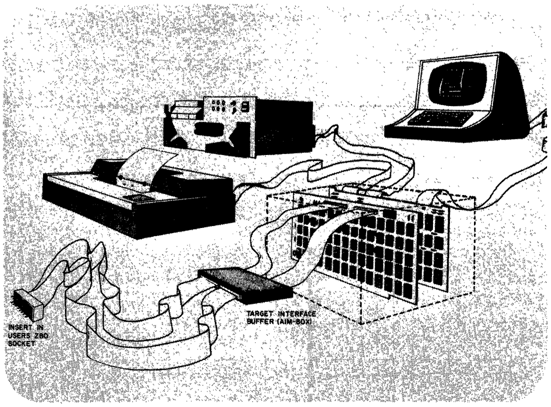
LITERATURE

Contact your local Mostek office for more detailed literature about MOSTEK'S development systems.

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Application Interface Module (AIM*-80E)



HARDWARE FEATURES

□ Direct Interface with SDB-80E

□ Single step/multistep with register trace

□ Execution intercept (breakpoint) intercepts on memory access, port access, external trigger, event counter, or delay counter

□ Push button execution intercept

□ 256x32 history memory which samples Data Bus, Address Bus, $\overline{\text{M1}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{IORQ}}$, and four external probes

□ History memory clock selectable from $\overline{\text{M1}}$, $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, or INTERRUPT ACKNOWLEDGE

□ Selectable history memory clock conditions: read only, write only, DMA only, or external probe only (high or low)

□ 8Kx8 ROM memory (firmware)

SOFTWARE FEATURES

□ ROM resident mnemonic dis-assembler

□ ROM resident RAM test for SDB or target RAM.

GENERAL DESCRIPTION

AIM-80E provides Z80 system debug assistance for both software and hardware via in-circuit emulation. (See Block Diagram)

Single step/multistep allows the programmer to trace through a program and display the CPU registers after each instruction. The execution intercept feature allows suspending program execution on the nth occurrence of an address or other specified condition. If the program has begun an unknown sequence, the intercept pushbutton will return the system to the single step mode. Single step and execution intercept (breakpoint) operate in RAM or ROM/PROM.

Hardware debugging is aided greatly by use of the 256x32 history memory which monitors bus transactions for a specified period. This information may then be displayed on the console. The data bus, address bus, $\overline{\text{M1}}$, $\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{IORQ}}$, and inputs from four probes are sampled and stored in the history memory upon every occurrance of the user specified clock ($\overline{\text{M1}}$, $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, or interrupt acknowledge) qualified by the user specified conditions (read only, write only, DMA only, probe High only, or probe Low only). Upon the occurrence of the selected intercept, AIM-80E returns control to the system debug (DDT-80). The history memory may then be displayed (See Figure 1) with or without mnemonic dis-assembly.

USING THE AIM-80E

AIM-80E may be added directly to any SDB-80E system. All system bus signals are wired one to one between SDB-80E and AIM-80E. Voltage requirements for the AIM-80E are the same as for the SDB-80E. Programs may be debugged in SDB-80E memory space or with the target interface buffer box (AIM-80X) may be debugged in the target environment. Dynamic memory mapping allows target memory to be simulated using SDB-80E system RAM. All peripheral devices of the SDB-80E are still functional with the AIM-80E.

* Trademark of Mostek Corporation

SYSTEM FIRMWARE

To minimize the impact of the AIM-80E firmware is resident in one MK36000, 8Kx8 ROM. This firmware is completely compatible with DDT-80 firmware and includes five new commands for control of the AIM-80E. The interactive nature of the commands makes operation simple and avoids operator errors. The ROM resident dis-assembler makes correlation with the user's source listing easier and reduces the necessity of memorizing op codes.

ELECTRICAL SPECIFICATIONS

Operating Temperature Range 0°C to +50°C
Power Supply Requirements (Typical)
+12V + 5% @ 12mA
+ 5V ± 5% @ 1.0 Amp
Interface - SDB-80E compatible
Operating Frequency - 1-2.5 MHz (with SDB-80E)

MECHANICAL SPECIFICATIONS

Board Size:	250mm x 233.4mm x 18mm
Bottom Connector:	Dual 64 pin Eurocard Connector DIN 41612 form D; A and C Pinned
Top Connectors:	One 40 pin 3M ribbon One 50 pin 3M ribbon

AIM-80E PRINT OUT EXAMPLE
(User entries underlined)

```
.I 2,T 2

TRIG ON (MREQ/IORQ/+/-) M
EVENT CNT (1-FF) 2
DELAY CNT (0-FF) 3
CLOCK ON (M1,MREQ,IORQ,INTA) MR IO
ONLY IF (RD/WR/DMA/H/L)
.E 0 2
0005 2421

.
```

Set intercept at address 0002H with trigger option

Trigger on $\overline{\text{MREQ}}$
After 2 occurrences
Delay 1 clock after trigger
Clock history (sample) memory on $\overline{\text{MREQ}}$ or $\overline{\text{IORQ}}$
No qualifying conditions selected
Begin execution at address 0002H
Intercept occurs at second occurrence of address 2 with a delay of one.

At this point the history memory contains the bus transactions which occurred before the intercept.

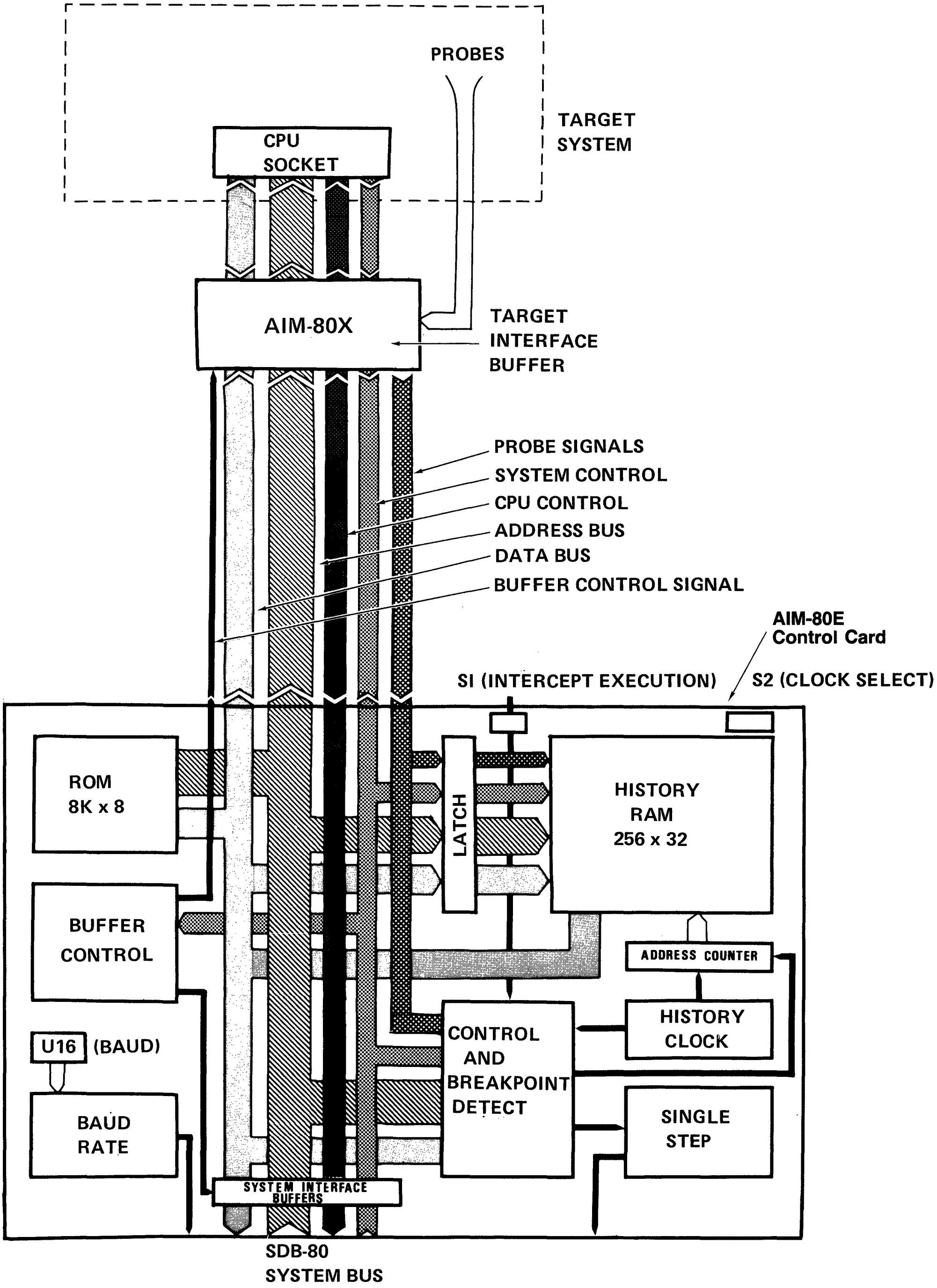
```
.T F,-B 2

                                          Control Bus        Probes (red, blue, green, yellow)
 OS  ADDR DB                             /M1 /MR /RD /IO  R   B   G   Y
-0B  0000 3E   LD    A,20H                0   0   0   1   1   1   1   1
-0A  0001 20                              1   0   0   1   1   1   1   1
-09  0002 D3   OUT   (0),A                0   0   0   1   1   1   1   1
-08  0003 00                              1   0   0   1   1   1   1   1
-07  2000 20                              1   1   1   0   1   1   1   1
-06  0004 3C   INC   A                    0   0   0   1   1   1   1   1
-05  0005 3C   INC   A                    0   0   0   1   1   1   1   1
-04  0006 3C   INC   A                    0   0   0   1   1   1   1   1
-03  0007 C3   JP    0002H                0   0   0   1   1   1   1   1
-02  0008 02                              1   0   0   1   1   1   1   1
-01  0009 00                              1   0   0   1   1   1   1   1
+00  0002 D3   OUT   (0),A                0   0   0   1   1   1   1   1
+01  0003 00                              1   0   0   1   1   1   1   1
+02  2300 23                              1   1   1   0   1   1   1   1
+03  0004 3C   INC   A                    0   0   0   1   1   1   1   1

-FC ?.
.
```

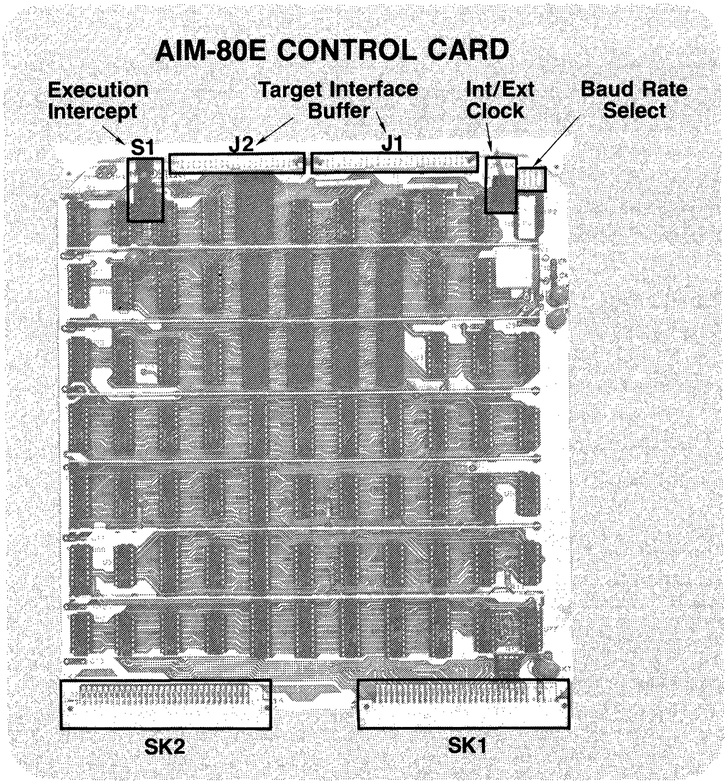
Trace for 15 lines - starting at
offset -11 before trigger.
Offset from trigger
Memory Address Bus contents
Data Bus contents
Dis-assembled instructions

First occurrence of address 0002H
Trigger (OS = 0) (2nd occurrence of address 0)
Delay 1 count after trigger
No more data in history module


PROBES
TARGET SYSTEM
CPU SOCKET
AIM-80X
TARGET INTERFACE BUFFER
PROBE SIGNALS
SYSTEM CONTROL
CPU CONTROL
ADDRESS BUS
DATA BUS
BUFFER CONTROL SIGNAL
AIM-80E Control Card
SI (INTERCEPT EXECUTION)
S2 (CLOCK SELECT)
ROM 8K x 8
LATCH
HISTORY RAM 256 x 32
BUFFER CONTROL
ADDRESS COUNTER
U16 (BAUD)
CONTROL AND BREAKPOINT DETECT
HISTORY CLOCK
BAUD RATE
SINGLE STEP
SYSTEM INTERFACE BUFFERS
SDB-80 SYSTEM BUS

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
AIM-80E	AIM-80E Control Card includes AIM-80E Firmware AIM-80E Operations Manual Target Interface Buffer (AIM-80X) includes: cables connectors 4 probe clips	MK78106
	AIM-80E Operations Manual	MK78559



AIM-80E CONTROL CARD

MOSTEK®

3870 MICROCOMPUTER SYSTEMS

Application Interface Module (AIM-72E)

FEATURES

□ Real time in-circuit emulation of Mostek's 3870 family of microcomputers, including MK3870, MK3872, and MK3876,

□ Direct interface to Mostek's SYS-80F Dual Floppy Disk Microcomputer with ZAIM-72 software supplied on floppy disk.

□ Standard Features Include:

● Breakpoint insertion, memory display and modification, port display and modification, and single step.

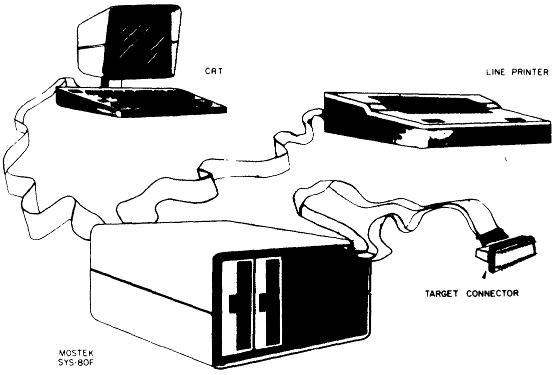
● Execution intercept from user keyboard with the ESCAPE key.

● Debugging or emulation mode is selectable from the user's console.

● Debugging of 3870 and F8 programs up to 8K long.

DESCRIPTION

AIM-72E (Application Interface Module) is a unique development aid for debugging MK3870 Series Microcomputer applications in the actual hardware and software configuration of the user's final system (referred to as the 'Target'.) To accomplish this, it is first necessary to emulate the Target ROM with RAM. This RAM must appear as ROM to the application while retaining the ability to be loaded, debugged, and modified using peripherals independent of the Target. It is the purpose of AIM, used in conjunction with the SYS-80F Disk Based Microcomputer to provide these capabilities. With AIM-72E, all of the peripheral and debugging capabilities of the user's development system may be applied directly to either the prototype or final production configuration of any MK3870, MK3872, or MK3876 application; no modifications to the user's hardware, software, or mechanical package are required.



USING AIM

The pictorial diagram above shows how AIM-72E would typically be used during system development. Because the AIM-72E is an exact functional emulation of the MK3870 family, it may be directly inserted into the 3870, 3872, or 3876 socket in the target system. Also, since the Target can be a production version of the user's application, product revisions and enhancements may be easily implemented. As shown in the diagram, the AIM Board is mounted in the card cage of the user's development system. It is the purpose of the SYS-80F to provide the user with the means for accessing and controlling the target system (via the AIM Board) during the program development phase. This provides access to all the debugging software and peripherals of the development system without having to introduce any perturbations to the Target system environment. AIM-72E does not affect the peripheral expansion capabilities of the development system.

SPECIFICATIONS

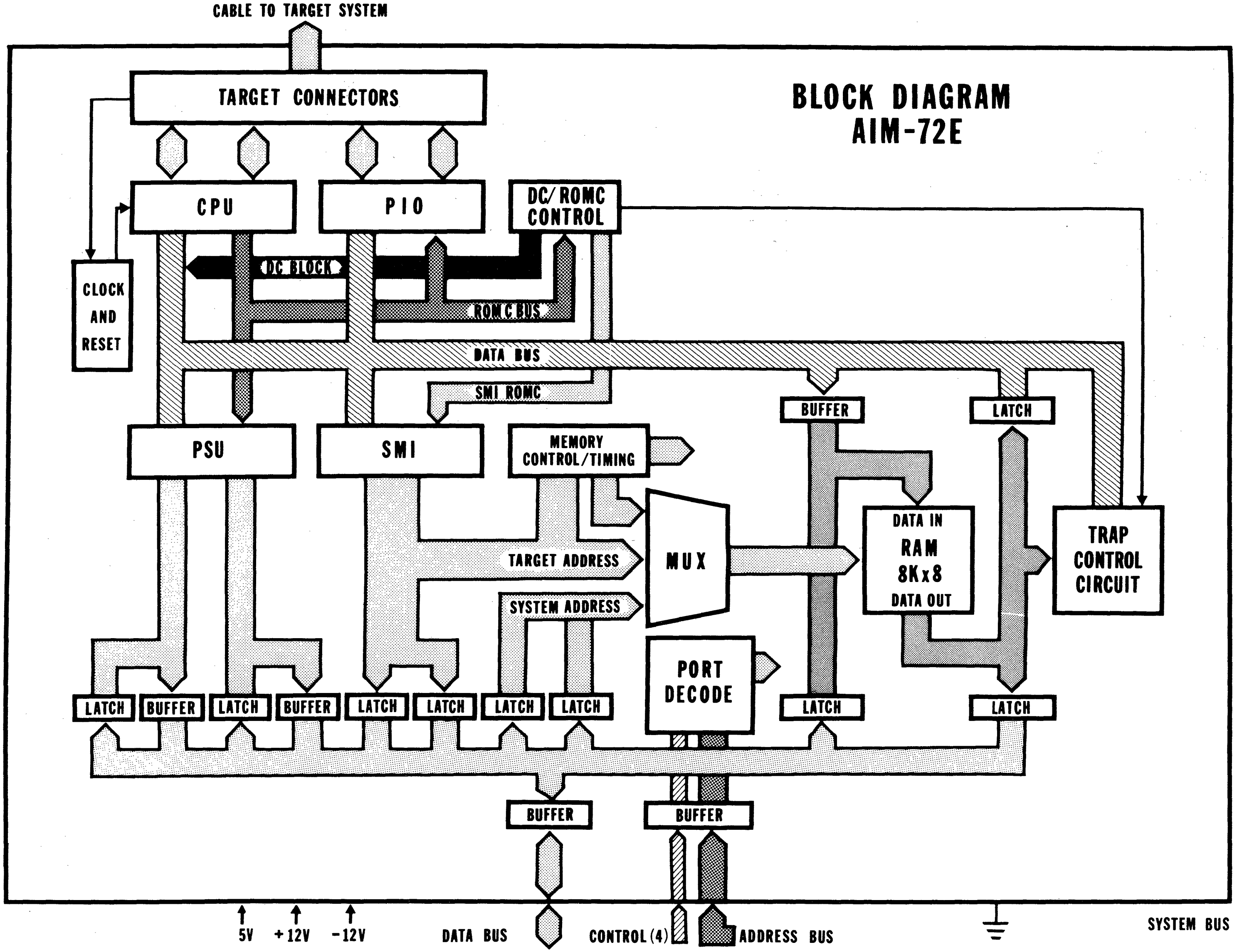
Operating Temperature Range. . . . . 0° C to 50° C

Power Supply Requirements
+5V ± 5% @ 1.5A max.
+12V ± 5% @ 100mA max.
-12V ± 5% @ 30mA max.

Board Size. . . . . .233.4 mm x 257.62 x 25 mm

Connectors/Cables:
40-Pin Ribbon Cable (24" long)

Develop
Systems
Europe



BLOCK DIAGRAM AIM-72E

BLOCK DIAGRAM DESCRIPTION

As shown in the block diagram, the AIM-72E contains all the functional elements necessary to emulate MK3870 Series Microcomputers. Target Ports are emulated with the CPU and PIO Ports. Target ROM and RAM are emulated with the 8K x 8 RAM which can also be accessed directly by the control system via the system bus connector. System memory accesses are transparent to the Target system execution. Thus, there is no impact on Target execution timing. The Target memory map can be controlled from the system allowing 2K, 4K or 8K Bytes of memory to be available in the Target System. Debug firmware in a PSU on the AIM-72E interfaces with the system to implement the breakpoint, single step and other functions. Trap control circuitry allows the use of a single byte breakpoint, providing complete flexibility when using break points in tight programming loops. Execution is at full speed, determined only by the user's crystal frequency - no speed reduction is introduced by the AIM's operating system. The AIM-72E clock may be emulated for the Target system from an onboard crystal oscillator or from the SYS-80F clock.

MULTI 3870 SERIES APPLICATIONS

Up to eight AIM-72E boards may be installed in one control system with each AIM-72E used to emulate a different MK3870 Series Microcomputer. The debug functions on each AIM-72E may be enabled one at a time and each program developed until all Target programs are functional. Only one AIM-72E may be in the active debug mode at a time; other AIM-72E's will be in the Emulator mode.

ZAIM-72 SOFTWARE DESCRIPTION

ZAIM-72 is the software designed to operate the AIM-72E board on Mostek's SYS-80F Dual Floppy Disk Microcomputer. It is supplied on a standard FLP-80DOS diskette. The software has the same command structure as other Mostek debuggers. The commands available with ZAIM-72 are summarized below. Designations s, f, and d stand for operands.

,A s, f Assign data byte f to target memory location s.

,B s Set a breakpoint at target memory location s. Up to 8 breakpoints can be set at once.

,C s, f, d Copy the target memory block s to f to target memory starting at d.

,E s Execute target program at location s.

,F s, f, d Fill target memory locations s through f with data d.

,G s Get binary file s and load it into Target memory.

,H Hexadecimal arithmetic.

,I Reinitialize target system.

,L s, f, d Locate data d in target memory at location f.

,M s Display and update target memory at location s.

,M s, f, d Tabulate target memory locations s through f. Option d specifies additional printout of ASCII characters or disassembly.

,O s Set relative offset equal to s for all address operands.

,P s Display and update target port number s.

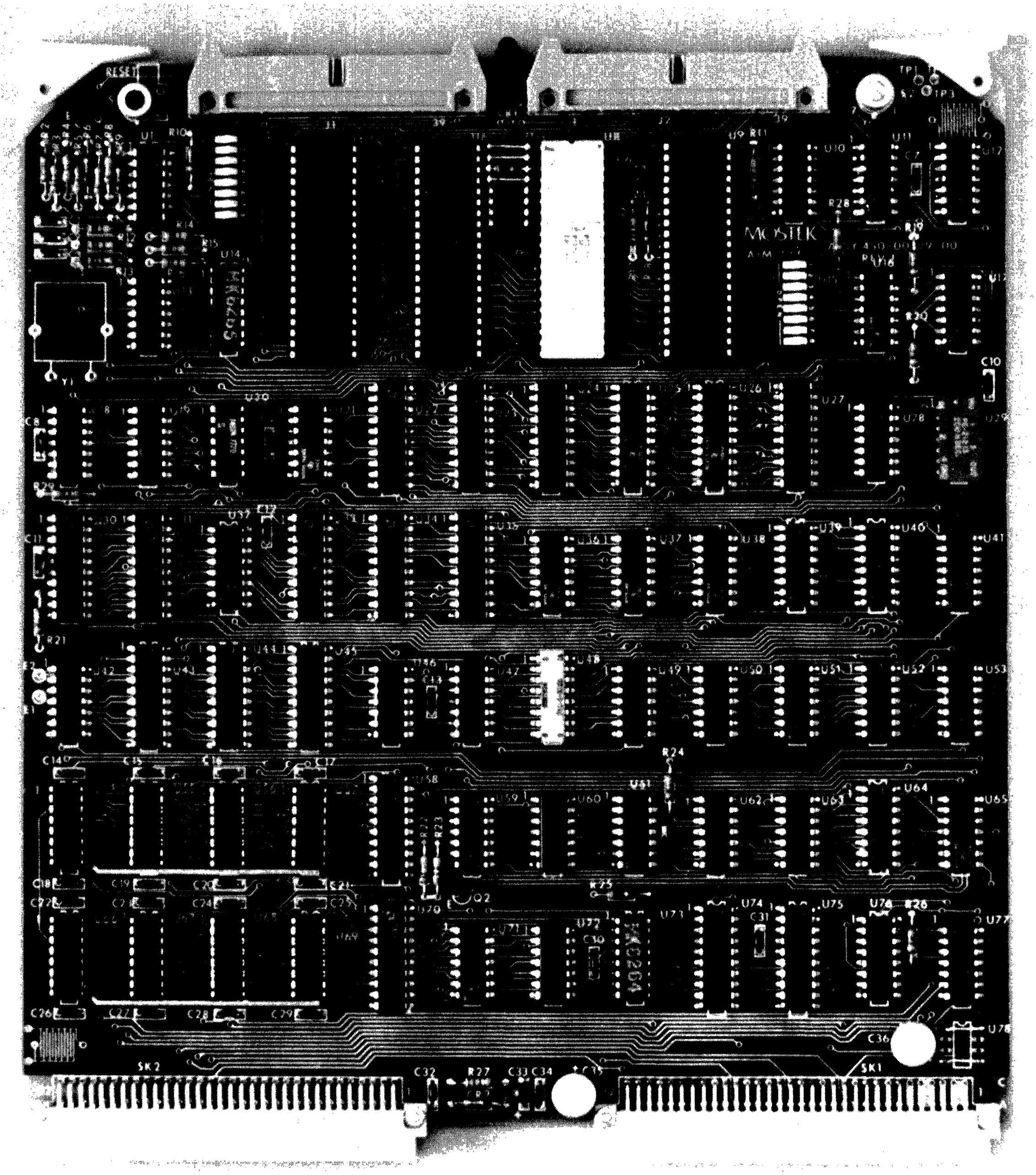
,Q Quit and return to FLP-80DOS Monitor.

,R s, f Display target registers, Option s allows a heading to be printed and option f specifies the number of scratchpad registers to be displayed.

,S s, f Single step starting at target location s for f number of steps.

,V s, f, d Verify target memory block s through f against target memory block starting at location d.

Target system programs are developed using the Mostek SYS-80F Cross Assembler for 3870/F8 Micocomputers (FZCASM-MK79075). Then ZAIM-72 is used to debug the completed program on the user's Target system. The software features multiple breakpoints, single step, and in-line disassembly. Target system memory, ports, and registers may be displayed and updated.


RESET
MOSTEK
U1
U14
Y1
C8
C10
C11
R21
R24
R25
R26
R27
R28
C14
C36
Q2
SK2
SK1
C32
C33
C34
C35
MKC264

ORDERING INFORMATION		
DESIGNATOR	DESCRIPTION	PART NO
AIM-72E Operations Manual	Contains a complete description of the use and operation of AIM-72E.	MK79579
AIM-72E	Includes the AIM-72E circuit board and the AIM-72E Operations Manual and the ZAIM-72 (on diskette) software for developing 3870 series applications.	MK79077
SYS-80F	A complete dual floppy disk development system (less terminal and line printer) to provide 3870 family assembly capability. Order AIM-72E to provide 3870 family debug capability.	MK78134
FZCASM	SYS-80F Cross Assembler for 3870/F8 Microcomputers. Provides disk-based assembly for 3870 assemby language programs on the Mostek SYS-80F Microcomputer.	MK79079

NOTE: A SDB-50/70 compatible version of the AIM-72E is also available. It is the AIM-72 and it includes FAIM-72 which is a control program equivalent to FAIM-72.

Prom Programmer (PPG-08)

FEATURES

□ Programs, reads, and verifies MK 2708 PROMS

□ Directly interfaces to SDB-50/70 and SDB-80

□ Driver software included

□ Zero insertion force socket

□ Power and programming indicators

GENERAL DESCRIPTION

The MK 2708 PROM Programmer (PPG-08) is a peripheral which provides a low-cost means of programming MK 2708 UV erasable PROMs. The PPG-08 has a generalized computer interface (two 8-bit I/O ports) allowing it to be controlled by most types of host computers with user-generated driver software. It is directly compatible with MOSTEK's F8 Software Development Board (SDB-50/70) and Z80 Software Development Board (SDB-80). Driver software in paper-tape form and source listings for the SDB-50/70 and SDB-80 are included with the purchase of the PPG-08. A complete set of documentation is also provided with the PPG-08 which describes the internal operation and details user's operating procedures. Interface cables for the SDB-50/70 and SDB-80 may be purchased separately. Another optional accessory is a TI Silent 700 compatible cassette tape containing control software for the SDB-50/70 and SDB-80.

SPECIFICATIONS

Interface

40 pin control connector (.1" centers card edge)
12 pin power connector (.156" centers card edge)
All control signals are TTL compatible.

Power requirements

+12 VDC @ 250 mA typical
+ 5 VDC @ 100 mA typical
−12 VDC @ 50 mA typical

Operating Temperature	0° to 50°C
Programming time (maximum)	2.5 minutes
Physical Dimensions	5" x 7" x 2"

ORDER INFORMATION

NAME	DESCRIPTION	PART NO.
PPG-08	MK 2708 PROM Programmer	MK 79033
XAID-805	Cable for interface to SDB-80	MK 79041
XAID-705	Cable for interface to SDB-50/70	MK 79046
SWD-1	Driver software on TI Silent 700 compatible cassette tape for SDB-50/70 and SDB-80	MK 79051

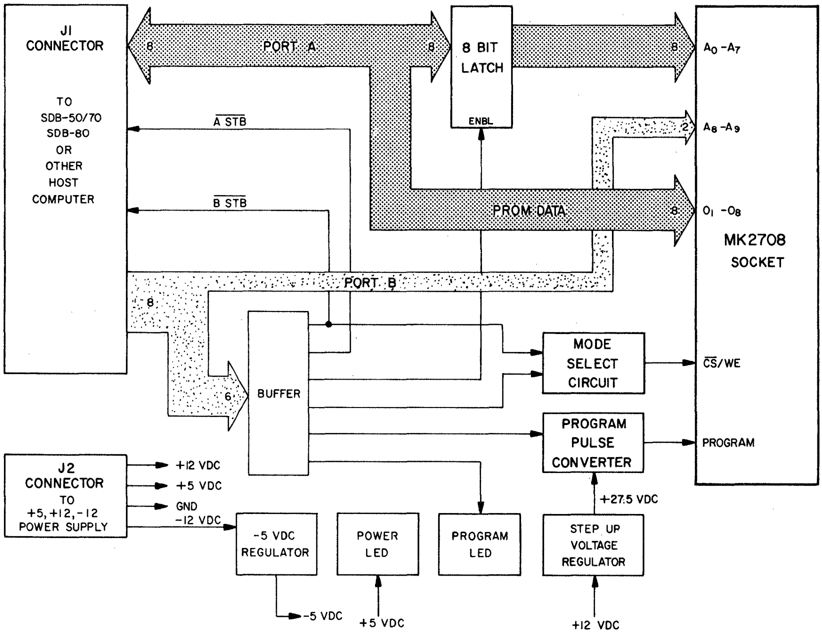
CONTROL CONNECTOR (J1) PIN-OUT

PIN #	Signal Name	Direction	Description
All Odd Pins (1-39)	GND		Logic Ground
J1-2	ASTB	Output	"LOW"when Port A (PA0-PA7) is in output mode
	PA0 - PA7	Bidirectional	PORT A (PA0-PA7) is used to output the lower 8 bits of PROM address to latch, output PROM data during programming and input PROM data during read sequence.
J1-24	$\overline{\text{BSTB}}$	Output	"LOW" when Port A (PA0-PA7) is in input mode.
J1-26	PB0/ADDR8	Input	PROM address bit 8
J1-28	PB1/ADDR9	Input	PROM address bit 9
J1-30	PB2/PAIN	Input	"HIGH" when Port A (PA0-PA7) is in input mode and PROM is in read mode.
J1-32	PB3/PROG MODE	Input	"HIGH" during program mode.
J1-34	PB4/PROG PULSE	Input	Programming Pulse
J1-36	PB5/PA OUT	Input	"HIGH" when Port A (PA0-PA7) is in output mode.
J1-38	PB6/CLK LATCH	Input	Clock to strobe address bits 0-7 into latch
J1-40	PB7/PROG LED	Input	Control line for programming indicator

POWER CONNECTOR (J2) PIN-OUT

J2-1,A	$+5V_{DC}$	J2-4, 5, D, E	$+12V_{DC}$
J2-2, 3, B, C	GND	J2-6, F	$-12V_{DC}$

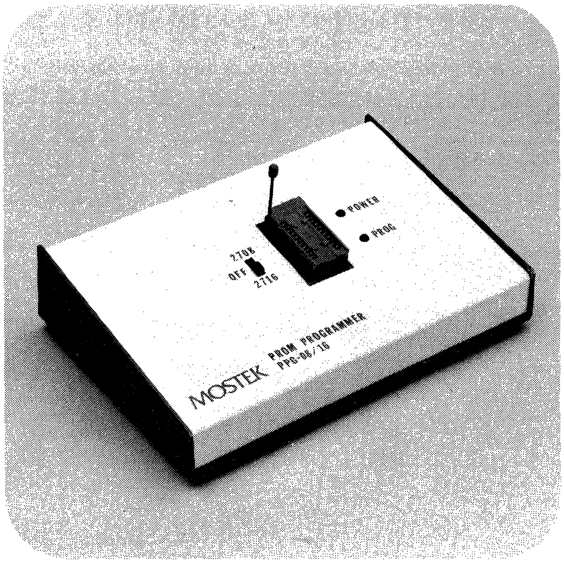
BLOCK DIAGRAM



MOSTEK®

MICROCOMPUTER SYSTEM

PROM Programmer (PPG-8/16)



FEATURES

□ Programs, reads, and verifies 2708, 2758, and 2716 type PROMs (2758 and 2716 PROMS must be 5 Volt only type)

□ Directly interfaces to SDB-50/70, SDB-80, AID-80F and SYS-80F

□ Driver software included

□ Zero insertion force socket

□ Power and programming indicators

DESCRIPTION

The PPG-8/16 PROM Programmer is a peripheral which provides a low-cost means of programming 2708, 2758, or 2716 PROMs. It is directly compatible with MOSTEK's F8 Software Development Board (SDB-50/70), Z80 Software Development Board (SDB-80), Z80 MDX system and Z80 Microcomputer Development System (AID-80F or SYS-80F). The PPG-8/16 has a generalized computer interface (two 8-bit I/O ports) allowing it to also be controlled by other types of host computers with user-generated driver software. A complete set of documentation is provided with the PPG-8/16 which describes the internal operation and details user's operating procedures.

The PPG-8/16 is available in two packaging configurations: (1) In a metal enclosure for use with the SDB-50/70, SDB-80 or SYS-80F, or alternatively (2) fully integrated into the AID-80F. Interface cables for either the SDB-80 or SDB-50/70 must be purchased separately.

SOFTWARE DESCRIPTION

The driver software available for the SDB-80, SDB-50/70, AID-80F, and SYS-80F accomplishes four basic operations. These are: (1) loading data (object tapes for SDB-80 and SDB-50/70 or binary files for the AID-80F or SYS-80F) into host computer memory, (2) reading the contents of a PROM into host computer memory, (3) programming a PROM from the contents the host computer memory, and (4) verifying the contents of a PROM with the contents of the host computer memory.

The driver software is provided in the form of paper-tape for both the SDB-50/70 and the SDB-80. An optional accessory is a TI Silent 700 compatible cassette object tape containing control software for the SDB-50/70 and SDB-80. Users of MOSTEK's AID-80F or SYS-80F who wish to upgrade their systems with a PPG-8/16 will find the driver software on their system diskette (version 2.0 or later). The user documentation provided with PPG-8/16 fully explains programming procedures which enable a user to develop a software driver on a different host computer.

SPECIFICATIONS

INTERFACE

25 Pin control connector (D type) for MDX or SYS-80F
40 Pin control connector (.1 in. centers card edge) for AID-80F, SDB-80, or SDB-50/70
12 pin power connector (.156 in. centers card edge)
All control signals are TTL compatible.

POWER REQUIREMENTS

+12VDC @ 250 mA typical
+5VDC @ 100 mA typical (Supplied by AID-80F or SYS-80F)
-12VDC @ 50 mA typical

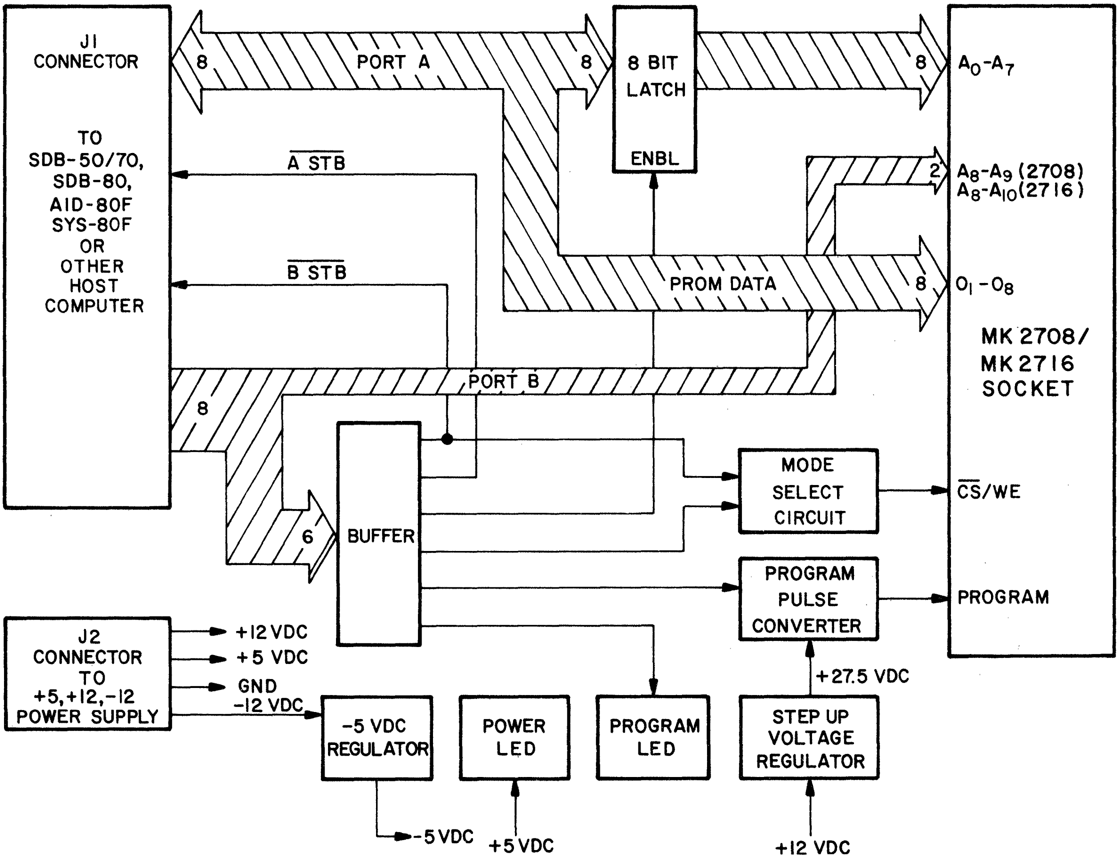
OPERATING TEMPERATURE

0°C - 50°C

PROGRAMMING TIME

2708	2.5 minutes
2758	0.9 minutes
2716	1.8 minutes

PPG 8/16 BLOCK DIAGRAM



ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
PPG-8/16*	PROM Programmer for 2708/2758/2716 PROMs with Operations Manual and paper tape drivers for SDB-50/70 and SDB-80. Does not include cables.	MK79081
	PROM Programmer for 2708/2758/2716 PROMs with Operations Manual and paper tape drivers for SDB-50/70 and SDB-80. Includes cable for interface to SYS-80F. (Europe only).	MK79082
XAID-805	PPG-8/16 Interface Cable for SDB-80	MK79041
XAID-705	PPG-8/16 Interface Cable forSDB-50/70	MK79046
MD-PPG	PPG-8/16 Interface Cable for MDX-PIO	MK77957
SWD-2	PPG-8/16 Object programs for SDB-50/70 & SDB-80 on Silent 700 cassette tape.	MK79084
PPG-RETRO	Installation package for installing PPG 8/16 into the AID-80F (also can be used for PPG-08 integration into AID-80F). Includes cable.	MK78154
	PPG-8/16 Operations Manual	MK79603

*NOTE: The PPG-8/16 will only program the 2708, 2758, and 2716 PROMs. The 2758 and 2716 are 5 Volt only type PROMs. **THE PPG-8/16 WILL NOT PROGRAM THE TI 2716 MULTIPLE VOLTAGE 2K x 8 PROM.**

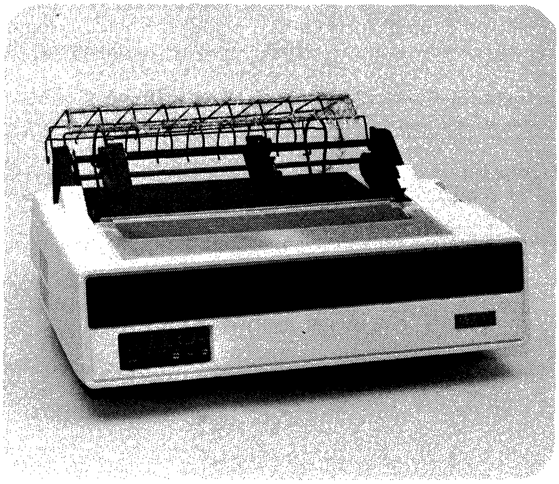
MOSTEK®

MICROCOMPUTER SUPPORT

Line Printer

FEATURES

- □ Interfaces directly to SYS-80F
- □ Prints 120 characters per second
- □ Up to 132 characters per line
- □ Prints original plus five copies
- □ Character elongation
- □ 8 inches per second paper slew rate
- □ Ribbon cartridge
- □ 7x7 dot matrix, 64 character ASCII
- □ Tractor feed/Pin feed platen
- □ Parallel interface



DESCRIPTION

The MOSTEK line printer is a state-of-the-art microprocessor controlled, dot-matrix line printer that prints at the rate of 120 characters per second. The printer has a maximum print width of 132 characters with a horizontal format of 10 characters per inch and 6 lines per inch vertical. Elogated (double width) characters are software selectable.

The Mostek line printer interfaces directly to the AID-80F or SYS-80F Microcomputer Systems and can be interfaced easily to other computer systems supporting parallel I/O.

SPECIFICATIONS

Print Performance - Minimum Throughout

Printer Model	Print Speed (cps)	Max. Print Width	10Char/ Line (lpm)	80Char/ Line (lpm)	132Char/ Line (lpm)
702	120	13.2 in. (335mm)	260	74	47

Characters
7x7 dot matrix, 64 character U.S. ASCII

Format
10 Characters per inch horizontal
6 Lines per inch vertical
Elongated (double width) characters software selectable

Forms Handling
Tractor feed, for rear or bottom feed forms

8 ips slew rate
Usable paper 4 in. (102mm) to 17.3 in. (439mm) width
Paper tension adjustment

Ribbon System
Ribbon cartridge
Continuous ribbon 9/16 in. (14mm) wide, 20 yards (18.3 meters) long.
Mobius loop allows printing on upper and lower portion on alternate passes.

Panel Indicators
Power On: Indicates AC power is applied to printer.
Select: Indicates printer can receive data.
Alert: Indicates operator-correctable error condition.

Operator Controls
Select/deselect
Forms thickness

Top of form
Horizontal forms positioning
Vertical forms positioning
Power ON/OFF
Single line feed
Paper empty override
Self-test

Internal Controls

Auto motor control: turns stepping motors off when no data is received.
Electronic top of form: allows paper to space to top of form when command is received.
Preset for 11 in. (279mm) or 12 in. (305mm) forms
Opt. VFU must be used for other form lengths.

Data Input

7 or 8 bit ASCII parallel; microprocessor electronics; TTL levels with strobe.
Acknowledge pulse indicates that data was received.

Electrical Requirements

50/60 Hz, 115/230 VAC;+10%/-15% of Nominal Tappable Transformer (100, 110, 115, 120, 200, 220, 230, 240 VAC).

Physical Dimensions

	Model 702
Weight:	60 lbs. (27 Kg)
Width:	24.5 in. (622mm)
Height:	8 in. (203mm)
Depth:	18 in. (457mm)

Temperature

Operating:	40° to 100°F (4.4° to 37.7°C)
Storage:	-40° to 160°F (-40° to 71.1°C)

Humidity

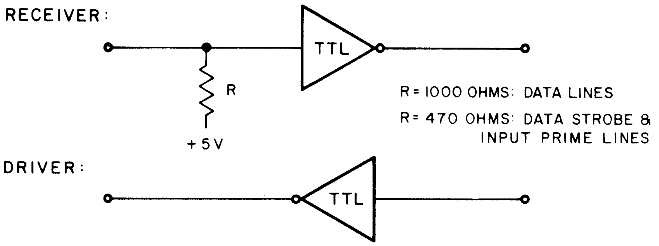
Operating:	20% to 90% (No condensation)
Storage:	5% to 95% (No condensation)

INTERFACING

INTERFACE DRIVERS AND RECEIVERS

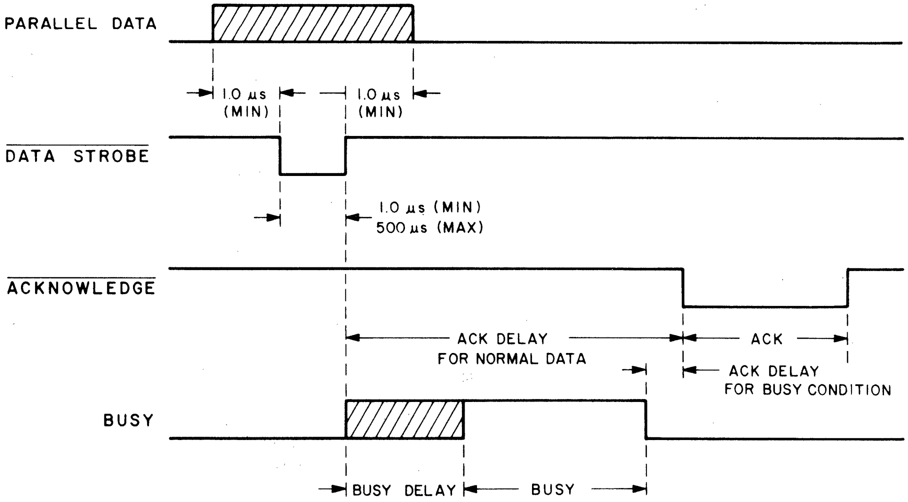
ALL INPUT/OUTPUT SIGNALS ARE TTL COMPATIBLE

LO: 0.4 VOLTS HI: 2.4 VOLTS



CONNECTOR: AMPHENOL 57 . 40360 SERIES, 36-PIN
(CENTRONICS 31310019)

INTERFACING TIMING



INTERFACE TIMING CONT'D

NORMAL DATA INPUT TIMING	ACK DELAY	2 - 6 µsec.
	ACK	4 µsec.
BUSY CONDITION TIMING	BUSY DELAY	0 - 1.5 µsec.
	ACK DELAY	1 - 6 µsec.
	ACK	4 µsec.
	BUSY DURATION:	
	Line Feed	350 - 500 µsec.
	Vertical Tab (1-in.)	135 - 145 msec.
	Form Feed (11-in.)	1.48 - 1.50 sec.
	Delete	160 - 400 µsec.
	Bell	0
	Select*	0 - 1.5 µsec.
	Deselect	Until Printer is selected
	Printer	8.33 msec/char.; plus 148 msec. non-printing time/line.

*No busy if inhibit prime on select option is used.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO.
MOSTEK LP	Mostek line printer featuring 120cps operation, 7x7 dot matrix, 10 cpi, and paper slew rate of 8 ips. Includes SYS-80F cable.	MK78150
	SYS-80F parallel interface cable for line printer (Centronics 702).	MK78098

Develop
Systems
Europe

MOSTEK®

MICROCOMPUTER SYSTEMS

AID-80F Cross Assembler for 3870/F8 (FZCASM)

FEATURES

□ Assembles all standard 3870/F8 family source statements

□ Object output in industry standard hexadecimal format extended for relocatable and linkable programs

□ Allows the following pseudo-ops:

ORG	-	program origin
EQU	-	equate label
DC	-	define constant
DEFL	-	define label
DEFM	-	define message
DEFB	-	define byte
DEFW	-	define word
DEFS	-	define storage
END	-	end statement
IF	-	conditional assembly
ENDIF	-	end of conditional assembly
INCLUDE	-	include another dataset within current assembly
NAME	-	program name definition
PSECT	-	program section definition
GLOBAL	-	global symbol definition

□ Supports the following assembler directive pseudo-ops:

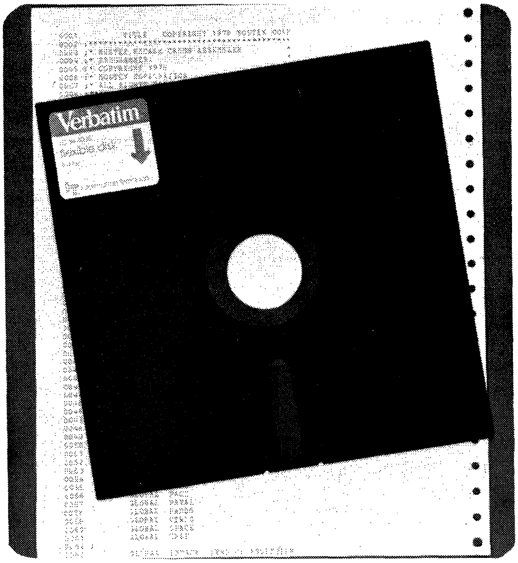
EJECT	-	eject a page of listing
TITLE	-	place heading at top of each page of listing
LIST	-	turn listing on
NLIST	-	turn listing off

□ Complete assembly in two passes with second pass repeatable

□ Size of program to be assembled limited only by memory available for symbol table

□ Supports conditional assembly, relocatable and linkable modules, symbol table and cross reference listings

□ Supplied on a standard FLP-80DOS diskette for use with the MOSTEK AID-80F floppy disk based development system



DESCRIPTION

The purpose of the 3870/F8 Cross Assembler is to assemble source language programs for the MOSTEK 3870 Series and F8 microcomputers. The Cross Assembler is designed to run on the MOSTEK AID-80F Dual Disk Development System with the FLP-80DOS operating system. The Cross Assembler is supplied on flexible diskette. The Assembler reads F8 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The assembly listing shows address, machine code, statement number, and source statement. The object code is in industry standard hexadecimal format modified for relocatable, linkable assemblies. A conversion utility (F8DUMP) is supplied to produce object code in F8 format for users of the MOSTEK SDB-50/70. The Assembler supports conditional assemblies, global symbols, relocatable programs, a printed symbol table and cross reference listing. It can assemble any length program limited only by a symbol table size of over 400 symbols. Expressions involving mathematical and logical operations are allowed. Conditional assembly allows the user to suspend assembly for a portion of the program depending upon the result of an expression. A global symbol is catagorized as "internal" if it appears as a label in the program; otherwise it is an "external" symbol. The printed symbol table and cross reference listing show which symbols are internal and which are external. The Cross Assembler allows the user to select relocatable or non-relocatable assembly via the "PSECT" pseudo-op. Relocation records are placed in the object output for relocatable assemblies.

The Assembler can be run as a single pass assembler or as a learning tool. (In this mode, global symbols and forward references are not allowed).

In conjunction with the FLP-80DOS Text Editor and Linker, FZCASM provides the means for editing, assembling and linking F8 or 3870 family programs.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NUMBER
FZCASM Cross Assembler	Includes the 3870/F8 Cross Assembler on a FLP-80 DOS system diskette, and the FZCASM Operations Manual.	MK79079
FZCASM Operations Manual	Describes in detail the operation of the 3870/F8 Cross Assembler	MK78582
AID-80F Data Sheet	Describes the MOSTEK AID-80F Dual Disk Development System	MK78568
FLP-80DOS Data Sheet	Describes the operating system used on the AID-80F System	MK78556
FLP-80DOS Operations Manual	Describes in detail the software and operating system used to run FZCASM on the AID-80F System	MK78557
AIM-72 Data Sheet	Describes the MOSTEK AIM-72 Application Interface Module used to provide in-circuit emulation capability for 3870 series Microcomputers on the AID-80F.	MK79576

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold
on condition that the Purchaser agrees to
the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.

4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject To This Agreement:

Order Number	Description	Price*

Ship To: ______________________

Bill To: ______________________

Method of Shipment: ______________________

Customer P.O. Number: ______________________

Agreed To:

PURCHASER	MOSTEK CORPORATION
By: ______________	By: ______________
Title: ______________	Title: ______________
Date: ______________	Date: ______________

* Prices Subject To Change Without Notice

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

BASIC Software Interpreter

FEATURES

□ Direct access to CPU I/O Ports

□ Ability to read or write any memory location (PEEK, POKE)

□ Arrays with up to 255 dimensions

□ Dynamic allocation and de-allocation of arrays

□ IF...THEN...ELSE and nested IF...THEN...ELSE

□ Direct (immediate) execution of statements

□ Error trapping, with error messages in English

□ Four variable types: Integer, string, real and double precision real.

□ **Full PRINT USING capabilities for formatted output**

□ Extensive program editing facilities

□ Trace facilities

□ Can call up to 10 assembly language subroutines

□ Boolean (logical) operations

□ **Supports up to 6 sequential and random access files on floppy disk.**

□ **Complete set of file manipulation statements**

□ Occupies only 19K bytes, not including operating system

□ Supports console and line printer I/O

□ Allows console output to be redirected to the line printer

DESCRIPTION

MOSTEK BASIC is an extensive implementation of Microsoft BASIC for the Z80 microprocessor. Its features are comparable to those BASICs found on minicomputers and large mainframes. Mostek BASIC is among the fastest microprocessor BASICs available.

Designed to operate on the Mostek Dual Disk Development System with FLP-80DOS and 32K bytes or more memory. Mostek BASIC provides a sophisticated software development tool.

Mostek BASIC is implemented as an interpreter and is highly suitable for user interactive processing. In a 32K byte system, about 7K bytes of free storage area are available to the user. Programs and data are stored in a compressed internal format to maximize memory utilization. By adding more memory to the system, the user's program and data storage area may be increased to as much as 31K bytes.

Unique features include long variable names, substring assignments and hexadecimal and octal constants. Many other features ease the task of programming complex functions. The Programmer is seldom limited by array size (up to 255 dimensions, with run time allocation and deallocation) or I/O restrictions. Full PRINT USING capabilities allow formatted output, while both input and output may be performed with multiple sequential and random files on floppy disk as well as with the CPU I/O ports. Editing, error trapping, and trace facilities greatly simplify program debugging.

Commands:

AUTO	CLEAR	CONTST	DELETE	EDIT
FILES	LIST	LLIST	LOAD	MERGE
NEW	NULL	RENUM	RESET	RUN
SAVE	SYSTEM	TRON	TROFF	WIDTH

Program Statement:

DEFNx	DEFDBL	DEFINT	DEFSNG	DRFSTR
DIM	END	ERASE	ERROR	FOR
GOSUB	GOTO	IF...THEN(ELSE)	LET	NEXT
ON...ERROR	ON...GO SUB	ON...GOTO	OUT	POKE
REM	RESUME	RETURN	STOP	SWAP
WAIT				

Input/Output Statements:

CLOSE	DATA	FIELD	GET	INPUT
KILL	LINEINPUT	LSET	NAME	OPEN
PRINT	PUT	READ	RESTORE	RESET

Operators

=	-	+	*	/
↑	\	MOD	NOT	AND
OR	XOR	IMP	EQV	<
>	<=	>=	<>	

Arithmetic Functions

ABS	ATN	CDBL	CINT	COS
CSNG	ERL	ERR	EXP	FRE
INP	INT	LOG	LPOS	PEEK
POS	RND	SGN	SIN	SPC
SQR	TAB	USRn	VARPTR	

String Functions

ASC	CHR$	FRE	HEX$	INSTR
LEFT$	LEN	MID$	OCT$	RIGHT$
SPACE$	STRING$	STR$	VAL	

Input/Output Functions

CVD	CVI	CVS	EOF	LOC
LOF	MKD$	MKI$	MKS$	

In order to receive Mostek BASIC, the attached Mostek BASIC Non-disclosure agreement should be signed and returned with each purchase order.

ORDERING INFORMATION		
Designator	**Description**	**Part No.**
MOSTEK BASIC	BASIC INTERPRETER high-level Language to run on FLP-80DOS. Requires 32K bytes of memory.	MK78157
AID-80F	Floppy disk development system for Z80 and and 3870/F8 systems. Includes FLP-80DOS and 32K bytes of RAM	MK78125

MOSTEK BASIC NON-DISCLOSURE AGREEMENT

The party below agrees that it is receiving a copy of Mostek BASIC for use on a single computer only, as designated on its registration card. The party agrees to fill out and mail in the registration card before making use of Mostek BASIC. The party agrees that all copies will be strictly safeguarded against disclosure to or use by persons not authorized by Mostek to use Mostek BASIC, and that the location of all copies will be reported to Mostek at Mostek's request. The party agrees that is agreement shall insure to the benefit of any third party holding any right, title or interest in the Mostek BASIC or any software from which it was derived.

"Party"__________________(Date)____________________
Company:______________________________________
Address:______________________________________
Phone:_______________________________________

Return to:

Mostek Corp.
Microcomputer Dept.
Software Librarian
MS 501
1215 W. Crosby Road
Carrollton, Texas 75006

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

FORTRAN IV Compiler

FEATURES

□ All of ANSI standard FORTRAN IV (X3.9-1966) except complex data type.

□ Generates relocatable linkable object code.

□ Subroutines may be compiled separately and stored in a system library.

□ Compiles several hundred statements per minute in a single pass.

□ Enhancements include

1. LOGICAL variables which can be used as integer quantities
2. LOGICAL DO loops for tighter, faster execution of small valued integer loops.
3. Mixed mode arithmetic.
4. Hexadecimal constants.
5. Literals and Holleriths allowed in expressions.
6. Logical operations on integer data. .AND., .OR., .NOT. and .XOR. can be used for 16-bit or 8-bit Boolean operations.
7. READ/WRITE End-of-File or Error Condition transfer. END=n and ERR=n (where n is the statement number) can be included in READ or WRITE statements to transfer control to the specified statement on detection of an error or end-of-file condition.
8. ENCODE/DECODE for FORMAT operations to memory.

□ Long descriptive error messages.

□ Extended optimizations

□ Z80 assembly language subprograms may be called from FORTRAN programs

DESCRIPTION

Mostek's FORTRAN IV Compiler package provides new capabilities for users of Z80–based microcomputer systems. Mostek FORTRAN is comparable to FORTRAN compilers on large mainframes and minicomputers. All of ANSI Standard FORTRAN X3.9-1966 is included except the COMPLEX data type. Therefore, users may take advantage of the many applications programs already written in FORTRAN.

Mostek FORTRAN IV is unique in that it provides a microprocessor FORTRAN development package that generates relocatable object modules. This means that only the subroutines and system routines required to run FORTRAN programs are loaded before execution. Subroutines can be placed in a system library so that users can develop a common set of subroutines that are used in their programs. Also, if only one module of a program is changed, it is necessary to re-compile only that module.

The standard library of subroutines supplied with FORTRAN includes:

ABS	IABS	DABS	AINT
INT	IDINT	AMOD	MOD
AMAXO	AMAX1	MAXO	MAX1
DMAX1	AMINO	AMIN1	MINO
MIN1	DMIN1	FLOAT	IFIX
SIGN	ISIGN	DSIGN	DIM
IDIM	SNGL	DBLE	EXP
DEXP	ALOG	DLOG	ALOG10
DLOG1O	SIN	DSIN	COS
DCOS	TANH	SQRT	DSQRT
ATAN	DATAN	ATAN2	DATAN2
DMOD	PEEK	POKE	INP
OUT			

The library also contains routines for 32-bit and 64-bit floating point additon, subtraction, multiplication, division, etc. These routines are among the fastest available for performing these functions on the Z80.

Develop Systems Europe

A minimum system size of 48K bytes (including FLP-80DOS) is required to provide efficient optimization. The Mostek FORTRAN compiler optimizes the generated object code in several ways:

1. Common subexpression elimination. Common subexpressions are evaluated once, and the value is substituted in later occurrences of the subexpression.

2. Peephole Optimization. Small sections of code are replaced by more compact, faster code in special cases.

3. Constant folding. Integer constant expressions are evaluated at compile time.

4. Branch Optimizations. The number of conditional jumps in arithmetic and logical IFs is minimized.

Long descriptive error messages are another feature of the compiler. For instance:

?Statement unrecognizable

is printed if the compiler scans a statement that is not an assignment or other FORTRAN statement. The last twenty characters scanned before the detected error are also printed.

As an option, the compiler generates a fully symbolic listing of the machine language to be generated. At the end of the listing, the compiler produces an error summary and tables showing the addresses assigned to labels, variables and constants.

LINKER

A relocating linking loader (LINK-80) and a library manager (LIB-80) are included in the Mostek FORTRAN package.

LINK-80 resolves internal and external references between the object modules loaded and also performs library searches for system subroutines and generates a load map of memory showing the locations of the main program, subroutines and common areas.

LIBRARY MANAGER

LIB-80 allows users to customize libraries of object modules. LIB-80 can be used to insert, replace or delete object modules within a library, or create a new library from scratch. Library modules and the symbol definitions they contain may also be listed.

CP/M UTILITY

A utility program (XCPM) is included which allows the user to copy FORTRAN source programs from CP/M diskettes to FLP-80DOS diskettes. At this point the programs can be compiled using the Mostek FORTRAN compiler.

FTRANS UTILITY

FTRANS allows the user to convert object programs produced by the Mostek Z80 assembler to a form that is linkable to FORTRAN programs.

DESIGNATOR	DESCRIPTION	PART NO.
MOSTEK FORTRAN IV	FORTRAN IV high level compiler to run on FLP-80DOS. Requires 48K bytes of RAM. Includes Operations Manual.	MK78158
Mostek FORTRAN IV User's Manual	Operations Manual only	MK79643

MOSTEK®

Z80 MICROCOMPUTER SYSTEM

FLP-80DOS Software Library Vol. 1 (LIB-80-V1)

FEATURES

- □ Includes 23 useful subroutines and programs for the Z80, including:
  - • Lawrence Livermore Lab's Basic
  - • Generalized sort program for up to 8 fields per record
  - • 8080 - Z80 source code converter
  - • Fast disk-to-disk copy utility
  - • Hexadecimal Dump Utility to dump memory on files
  - • Assembly Language Formatter Utility to format Z80 source into columns
  - • Word Processor Program Version 2.0, used to format documents
  - • Disk Recovery Utility used to recover bad disk files
- □ All programs are supplied in source, object, and binary format with complete documentation on a standard FLP-80DOS diskette.
- □ Requires FLP-80DOS Version 2.0 or higher.

DESCRIPTION

The Mostek FLP-80DOS Software Library is a collection of programs of general utility that run under FLP-80DOS Version 2.0 or higher. These programs are used quite extensively at Mostek. They are being offered in source format on diskette so that the user may not only use them as supplied, but may use them as a base for individually tailored software.

This software library differs from other libraries in that all programs in the library have been developed or modified in-house. All programs in the library are in use at Mostek and all have some utility.

The FLP-80DOS Software Library Volume 1 consists of a User's Guide and two diskettes containing the source and binary (or object for subroutines) forms for each one of the twenty-three included programs. In order to reduce the cost of the library, printed source listings are not supplied. The user can obtain a source listing easily by assembling the required source program. A brief User's Guide is a part of each program source.

The FLP-80DOS Software Library is a "Level 2" product. "Level 2" software products are supplied by Mostek but are not supported in the areas of technical assistance or updates.

ORDERING INFORMATION

PART NO.	DESCRIPTION
MK78164	LIB-80 Volume 1 - FLP-80DOS Software Library, including source, object, and binary formats on diskette, and a printed user's guide.

Develop Systems Europe

1979 MICROCOMPUTER DATA BOOK

MOSTEK®

3870/F8 MICROCOMPUTER SOFTWARE SUPPORT

Fortran IV Cross Assembler (XFOR-50/70)

FEATURES

□ ANSI-Fortran IV Source

□ Executes on 16 bit word length machine

□ Cross Assembler is machine independent for:

Character representation (ASCII or BCD)

Numerical representation (1's or 2's complement)

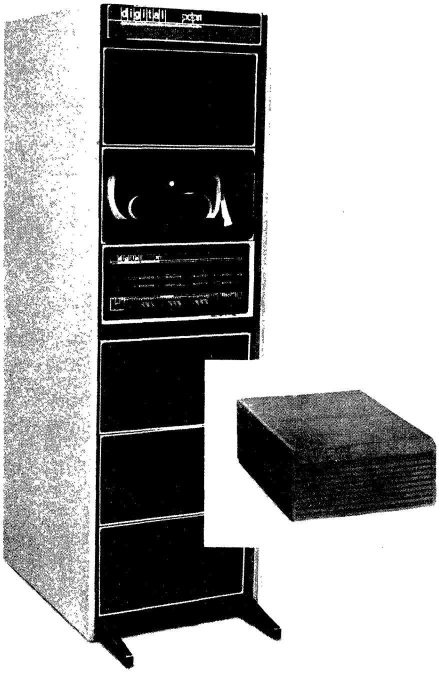
□ I/O logical device assignments are user definable

□ 2 pass assembly easily accomodated if no secondary storage available

□ Memory required: 13K words (typical)

□ Assembler directives

- TITLE 'Set page title'
- EJECT 'Page'
- EQU 'Values'
- ORG 'Beginning address'
- PUNCH 'Create load tape F8 loader format'
- PRINT 'Off and On enable for output listing
- DC 'Define constants'
- END



DESCRIPTION

The MOSTEK 3870/F8 Cross Assembler XFOR-50/70 is written in ANSI FORTRAN IV. It may be compiled and executed on any computer system which has at least a 16 bit word length for integer storage and 13K of memory for program storage. The Cross Assembler is independent of machine character representation (ASCII, BCD, etc.) and numerical representation (2's complement, 1's complement, etc.) Logical device assignments are set up in the source of the main program module, and may be easily changed to suit the installation. Also, if no secondary storage is available the main program may be changed to accommodate re-reading of the user input for the second pass of the assembly. Output is in F8 loader format.

ORDERING INFORMATION

The XFOR-50/70 is available directly from Mostek by filling out a copy of the Software Licensing Agreement printed on the back of this data sheet and returning it with the appropriate payment or Customer Purchase Order to:

MOSTEK CORPORATION
Microcomputer Systems Dept.
1215 West Crosby Road
Carrollton, Texas 75006

DESIGNATOR	DESCRIPTION	PART NO.
XFOR-50/70	3870/F8 Cross Assembler written in ANSI Fortran IV is supplied as a source card deck with Operations Manual.	MK79012

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold
on condition that the Purchaser agrees to
the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.

4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTIC—ULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject To This Agreement:

Order Number Description Price*

____

____

Ship To: ____________________ Bill To: ____________________

____

Method of Shipment: ____________________ Customer P.O. Number: ____________________

Agreed To:

PURCHASER	MOSTEK CORPORATION
By: ____________________	By: ____________________
Title: ____________________	Title: ____________________
Date: ____________________	Date: ____________________

*Prices Subject to change Without Notice

MOSTEK®

Z80 MICROCOMPUTER SOFTWARE SUPPORT

FORTRAN IV Cross Assembler (XFOR-80)

FEATURES

□ ANSI-FORTRAN IV Source

□ Executes on 16-bit word length machine

□ Cross Assembler is machine independent for:

Character representation (ASCII or BCD)

Numerical representation (1's or 2's complement)

□ I/O logical device assignments are user definable

□ 2 pass assembly easily accomodated if no secondary storage is available

□ Memory required: 20K words (typical)

□ Assembles all standard Z80 source statements and MACROs

□ Object output in Intel compatible hex format

□ Size of program to be assembled is limited only by memory available for symbol table.

□ Includes the following pseudo-ops:

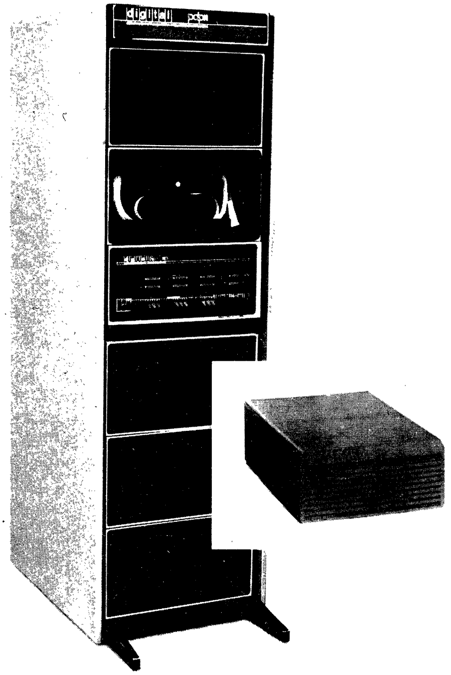
● ORG	– Program Origin
● EQU	– Equate
● DEFL	– Define Label ('Set')
● DEFM	– Define Message (ASCII Text)
● DEFB	– Define Byte
● DEFW	– Define Word
● DEFS	– Define Storage
● END	– End Statement
● MACR	– MACRO Definition
● ENDM	– End MACRO Definition

□ Supplied as a Source Card Deck with fully documented installation procedures.

DESCRIPTION

The XFOR-80 is a Cross Assembler for assembling Z80 source programs into the corresponding machine code for the Z80 microprocessor.

The XFOR-80 Cross Assembler is written in ANSI FORTRAN IV. It may be compiled and executed on any computer system which has at least a 20K memory for program storage. The Cross Assembler is independent of machine character representation (ASCII, BCD, etc.) and numerical representation (2's complement, 1's complement, etc.) Logical device assignments are set up in the source of the main program module, and may be easily changed to suit the installation. Also, if no secondary storage is available the main program may be changed to accomodate reading of the user input for the second phase of the assembly.



The assembled object output of the program is generated in Intel standard hexadecimal format and may be loaded into any compatible Z80 based microcomputer (such as the MOSTEK SDB-80) for execution and debug.

ORDERING INFORMATION

The XFOR-80 is available directly from MOSTEK by filling out a copy of the Software Licensing Agreement printed on the back of this data sheet and returning it with the appropriate payment or Customer Purchase Order to:

MOSTEK CORPORATION
Microprocessor Systems Dept.
1215 West Crosby Road
Carrollton, Texas 75006

Order Number	Description
MK 78117	X FOR-80

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold
on condition that the Purchaser agrees to
the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.

4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject To this Agreement:

Order Number Description Price*

Ship To: Bill To:

Method of Shipment: Customer P.O. Number

Agreed To:

PURCHASER MOSTEK CORPORATION

By: By:

Title: Title:

Date: Date:

*Prices Subject To Change Without Notice

MOSTEK®

MICROCOMPUTER HARDWARE SUPPORT

MK3870 Emulator (EMU-70)

FEATURES

□ Completely emulates the MK3870 single chip F8

□ Utilizes MK2708 PROMs

□ Connects directly to user's MK3870 socket

□ Provides exact program verification

The MK3870 Emulator (EMU-70) is a development aid for designing and field testing F8 microprocessor systems which utilize the MK3870 single-chip F8.

The Emulator is electrically equivalent to the MK 3870 but is field programmable instead of mask programmable. This enables a user to obtain final software verification prior to ordering an MK3870. Also, since the Emulator "plugs in" like an MK3870 (via a male, 40-pin connector or a 40-conductor cable), prototype systems can be converted to final production status by simply unplugging the Emulator and plugging in the corresponding custom MK3870.

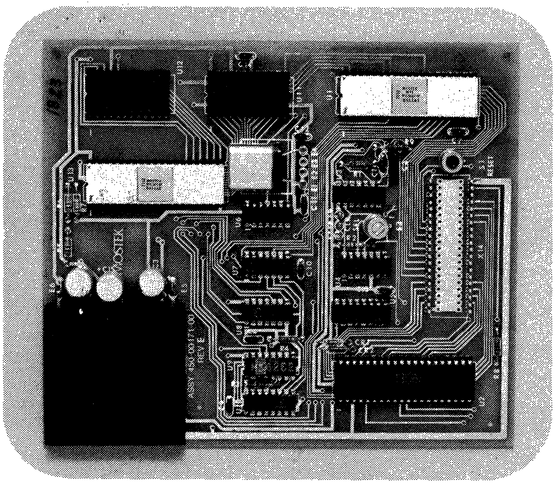
The MK3870 is a 5 volt only, 40-pin integrated circuit that provides 2K bytes of ROM, 64 bytes of RAM, four 8-bit latched I/O ports, a software programmable timer, and interrupt control circuitry.

EMU-70 DESCRIPTION

The Emulator performs all the functions of the MK3870'

CPU
ROM/RAM
INPUT/OUTPUT PORTS
VECTORED INTERRUPT
TIMER

The CPU functions, plus two I/O ports and scratchpad RAM, are implemented using an MK3850 on the Emulator board.

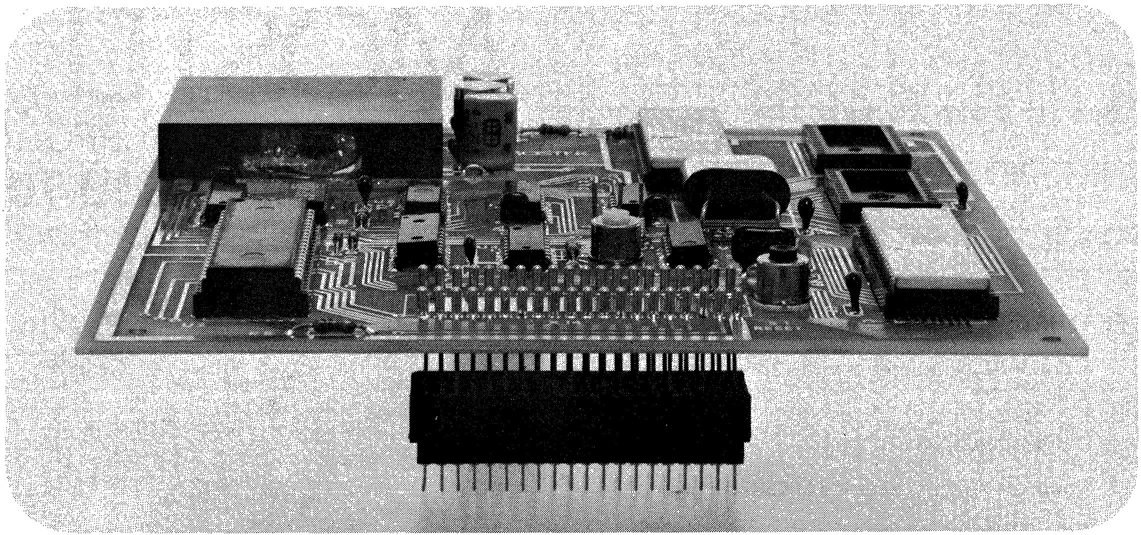


The ROM, Data Counter and Program Counter functions are implemented with an MK3853 SMI and two 1K x 8-bit UV Erasable PROMs to provide non-volatile storage of the user's program. The PROMs are programmed using a PROM programmer and then installed on the Emulator board.

Two I/O ports, interrupt and timer logic are implemented using an MK3871/90071 on the Emulator Board. The Emulator may be converted from standard TTL I/O ports to either open drain or direct drive I/O ports by ordering the appropriate PIO listed in the order information.

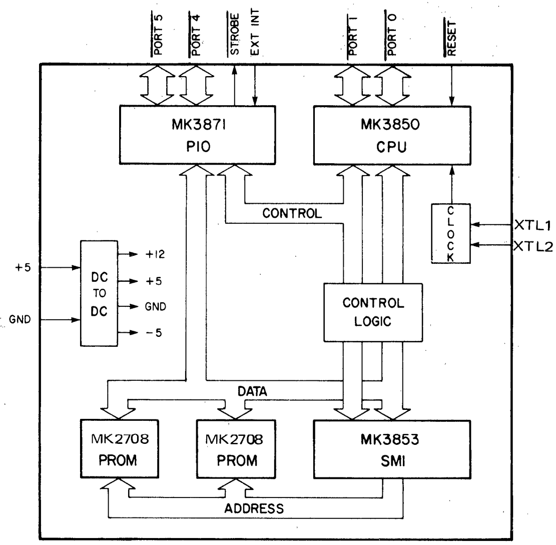
DOCUMENTATION

A complete set of documentation is provided with the Emulator to describe both the internal operation of the circuit board and the techniques for using it in system development. Also included are detailed instructions for ordering the MK3870 directly from the verified data contained in the Emulator (i.e. the contents of the UV PROMs).



EMU-70 showing 40-pin connector implementation.

FUNCTIONAL DIAGRAM



SPECIFICATIONS

Operating Temperature Range . . . . .0°C to 50°C

Power Supply Requirements (max.)
$+5V_{DC}\pm5\%$@ 1.5A

Board Size . . . 6.0" x 7.0"

ORDER INFORMATION

NAME	DESCRIPTION	PART NO.
EMU-70 Operations Manual	Contains a detailed technical description with schematic diagrams.	MK79550
EMU-70 (Less PROMs)	Circuit Board with documentation. Less PROMs.	MK79030
EMU-70 (With PROMs)	Circuit Board with documentation. Includes 2-MK2708 PROMs.	MK79032
XAID-706	Auxiliary 2ft interface cable for 'non-rigid' connection to the target system.	MK79050
Peripheral Input/Output	Direct Drive PIO	MK3871/ 90070
Peripheral Input/Output	Open drain PIO	MK3871/ 90072

*All prices subject to change without notice, and apply only within the U.S. and Canada.

† Contact the factory for current pricing.

3870 Series Microcomputer Emulator (EMU-72)

FEATURES

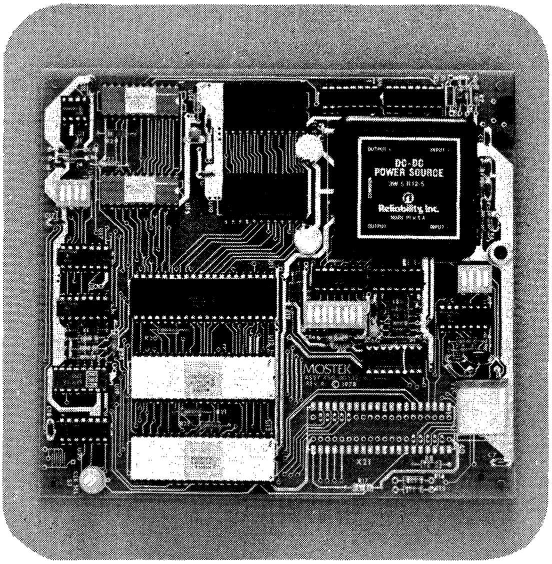
□ Completely emulates 3870 Series single chip Microcomputers (MK3870, MK3872, and MK3876)

□ Utilizes MK2716 PROMs

□ Connects directly to user's 3870 Series socket

□ Provides exact program verification

□ The 3870 Series Microcomputer Emulator (EMU-72) is a development aid for designing and field testing microcomputer systems which utilize 3870 Series Microcomputers. The Emulator is electrically equivalent to a 3870 Series Microcomputer (3870, 3872, or 3876) but is field programmable instead of mask programmable. This enables a user to obtain final software verification prior to ordering the mask programmable 3870 Series Microcomputer. Also, since the Emulator "plugs into" a 3870 socket (via a male, 40-pin connector or a 40-conductor cable), prototype systems can be converted to final production status by simply unplugging the Emulator and plugging in the corresponding 3870 Series Chip.



DESCRIPTION

The Emulator performs all the functions of the 3870 Series Microcomputers:

CPU
ROM/RAM
INPUT/OUTPUT PORTS
VECTORED INTERRUPT
TIMER

The CPU functions, plus two I/O ports and scratchpad RAM, are implemented using an MK3850 on the Emulator board.

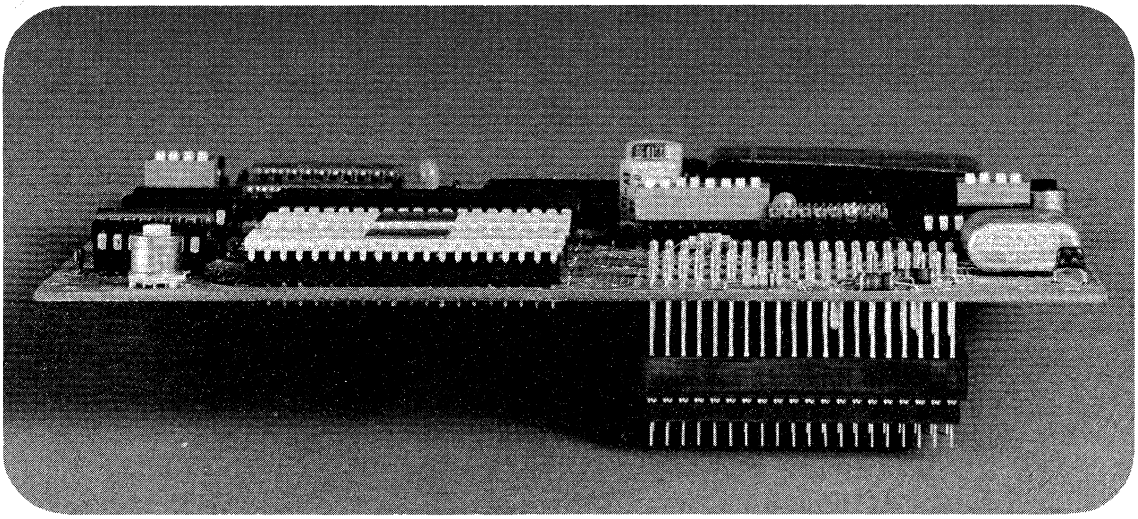
The ROM, Data Counter and Program Counter functions are implemented with an MK3853 SMI and two 2K x 8-bit UV Erasable PROMS to provide non-volatile storage of the user's program. The PROMs are programmed using a PROM programmer and then installed on the Emulator board.

The executable RAM of the MK3872 and MK3876 is emulated with two 5101 Static CMOS RAMs. Standby current and battery trickle charge current at the VSB terminal have been adjusted to emulate those functions on the MK3872 and MK3876.

Two I/O ports, interrupt logic, and timer logic are implemented using an MK3871 on the Emulator Board. The Emulator may be converted from standard TTL I/O ports to either open drain or direct drive I/O ports by ordering the appropriate PIO listed in the ordering information.

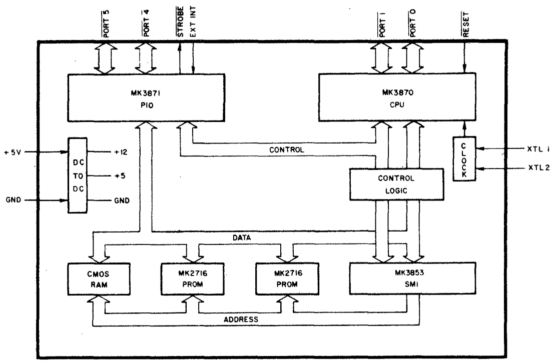
DOCUMENTATION

A complete set of documentation is provided with the Emulator to describe both the internal operation of the circuit board and the techniques for using it in system development.



EMU-72 showing 40-pin connector implementation

FUNCTIONAL DIAGRAM



SPECIFICATIONS

Operating Temperature Range. . . . . 0° C to 50° C

Power Supply Requirements. . .+5V ±5%@ 1.2A max. (700mAtyp.)

Board Size. . . . . 6.25 x 7.0 in.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NUMBER
EMU-72	3870 Series Microcomputer with Operations Manual. Includes a 40 pin adapter plug to interface to the user's 40 pin 3870 socket. Does not include 2716 PROMs.	MK79078
EMU-72 Operations Manual	Contains a complete technical description of the operation and use of the EMU-72 Schematic diagram included.	MK79581
XAID-706	Auxilary 2 foot interface cable for 'non-rigid' connection to the target system	MK79050
Peripheral Input/Output	Direct Drive PIO	MK3871N/ 90070 MK3871P/ 90070
Peripheral Input/Output	Open Drain PIO	MK3871N/ 90072 MK3871P/ 90072

* All prices subject to change without notice, and apply only within the U.S. and Canada.

Develop Aids

MOSTEK®

PRODUCT BRIEF

P-PROM™ Microcomputer MK3874

FEATURES

□ EPROM version of the MK3870, MK3872 and MK3876

□ Accepts 24-pin, industry-standard EPROMs or bipolar PROMs

□ PROM capacity: 1K, 2K, 4K bytes

□ Completely pin compatible with 3870 family of single-chip microcomputers

□ Software compatibility with 3870

□ Use as prototyping tool or for low volume production

□ 64 x 8 scratchpad RAM

□ 64 x 8 of executable RAM addressable by program or data counter

□ Standby power mode option for executable RAM which includes

- Low standby power, less than 8.2 mW
- Minimum 2.2V standby supply voltage
- No external components required to trickle charge battery

□ 32 bits (4 eight-bit ports) TTL compatible I/O (30 with Standby Option)

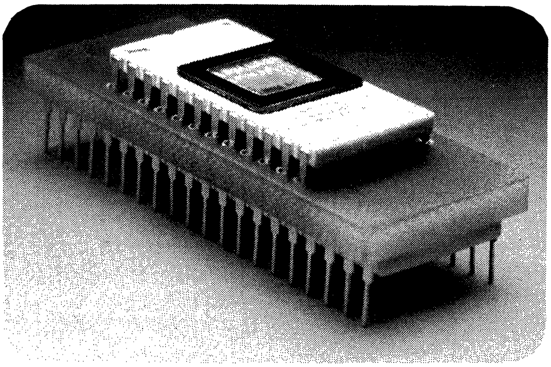
□ Programmable binary timer which includes:

- Interval timer mode
- Pulse width measurement mode
- Event counter mode

□ External interrupt

□ Crystal, LC, RC, or external time base

□ Single +5 volt supply

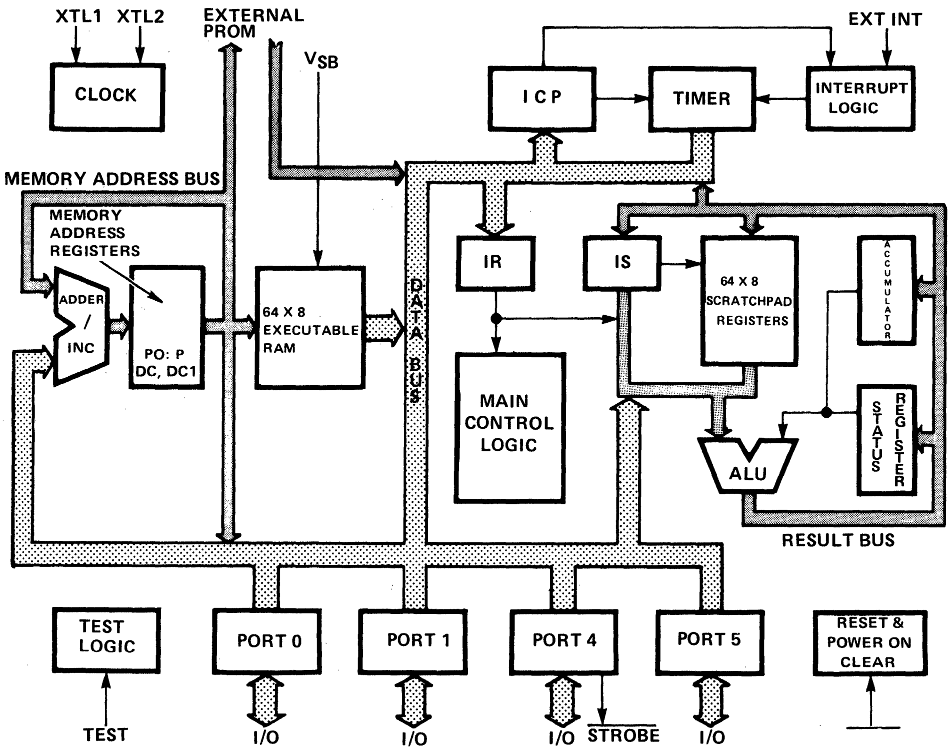


INTRODUCTION

The new MK3874 microprocessor is the PROM based version of the industry-standard 3870 family of single-chip microprocessors. The MK3874 is called the Piggyback PROM (P-PROM)TM because of a new Double-DipTM packaging concept. This concept allows a standard 24-pin PROM to be mounted directly on top of the microprocessor. This allows a standard EPROM to be easily removed for reprogramming and then reinserted as many times as desired. The MK3874 retains exactly the same pinout and architectural features as other members of the 3870 family. There are 32 lines (or 30 with the standby power RAM option) of bi-directional input/output, a sophisticated timer, vectored interrupts, executable and scratchpad RAM and an 8-bit CPU. Thus the 3874 P-PROMTM has the same functional capability and pinout as its 3870 masked-ROM counterpart while being able to support a standard PROM plugged into the top of the package.

Industry standard 24-pin, 5 volt PROMs are used with the MK3874. Presently six PROMs are compatible with the MK3874. They are the 2716 (2K x 8) 5 volt only, 2516 (2K x 8) 5 volt only, 2758 (1K x 8), 2532 (4K x 8), 2732 (4K x 8) and 82S2708 (1K x 8). The 2716 EPROM with its 2K of storage will allow the 3874 to emulate the 3870 and 3876 while a 2732 or 2532 EPROM containing 4K bytes of memory will allow emulation of the 3872. The 1K x 8 PROMs can be used for developing shorter programs. The standby power option is also available with the MK3874.

MK3874 BLOCK DIAGRAM



Supporting the 3874 is a complete line of development equipment including the low-cost Software Development Board (SDB-50/70) and an Application Interface Module (AIM-72). A fully integrated 3870/F8 development capability is provided by the AID-80F Disk Based Development System. Coupled with the AIM-72 and F8 Cross Assembler, it provides software generation and in-circuit emulation capabilities for the 3870 family of microcomputers.

PROM programming capability is provided through the use of the PPG 8/16 programming module available for either of the above systems.

Six different versions of the MK3874 are available and are designated MK974XX. The available versions of the MK3874 and their relevant features are listed in the table below. The different versions are provided so as to offer an option for low-power standby mode for the executable RAM and for different PROM pinouts. The MK97401 and MK97404 are supplied with MK2716 EPROMs. Otherwise the MK97401 is identical to the MK97400 and the MK97404 is identical to the MK97403.

All MK3874 versions have no internal pull-up resistor for the external interrupt and reset inputs. All are configured with the standard TTL port option for Ports 4 and 5. An open-drain and direct-drive version will be available in the second quarter of 1979.

ORDERING INFORMATION

MK3874 VERSION	PROM INCLUDED	STANDBY POWER OPTION	COMPATIBLE 5 - VOLT PROM's	3870 FAMILY DEVICE EMULATED
MK97400	NO	NO	2758 (1K x 8) 82S2708 (1K x 8) 2516 (2K x 8) 2716 (2K x 8) 2532 (4K x 8)	Partial 3870 Partial 3870 3870, 3876 3870, 3876 3872
MK97401	Yes (MK 2716)	NO	Same as MK97400	Same as MK 97400
MK97402	NO	NO	82S2708 (1K x 8) 2732 (4K x 8)	Partial 3870 3872
MK97403	NO	YES	2758 (1K x 8) 82S2708 (1K x 8) 2716 (2K x 8) 2516 (2K x 8) 2532 (4K x 8)	Partial 3870 Partial 3870 3876 3876 3872
MK97404	YES (MK 2716)	YES	Same as MK97403	Same as MK97403
MK97405	NO	YES	82S2708 (1K x 8) 2732 (4K x 8)	Partial 3870 3872

MOSTEK®

3870/F8 MICROCOMPUTER HARDWARE SUPPORT

Evaluation Kit (MCK 50/70)

FEATURES

The MOSTEK F8 Evaluation Kit is a basic F8 evaluation/development microcomputer with these features.

□ 24 bits of I/O arranged in three 8 bit ports

□ 1024 bits of Static Random Access Memory (MK4102)

□ Full duplex TTY Interface (20mA loop)

□ Crystal control clock

□ Non-volatile operating system in MK3851 Program Storage Unit. Firmware called designer development tool L1 (DDT—1)

DESCRIPTION

The F8 Evaluation Kit comes with complete documentation including a detailed application note, programming guide, and a listing of the DDT—1 program.

Purchasers of the Mostek F8 Evaluation Kit will receive free the F8/ANSI Fortran IV Cross Assembler.

The Evaluation Kit may be ordered as an assembled and tested unit (MK79002), or as an unassembled kit (MK79001) containing all necessary components for assembly including a 72-pin edge connector. A power supply box (MK79003) that provides an edge card connector, all necessary power, switch selectable BAUD rate and a TTY cable is also available.

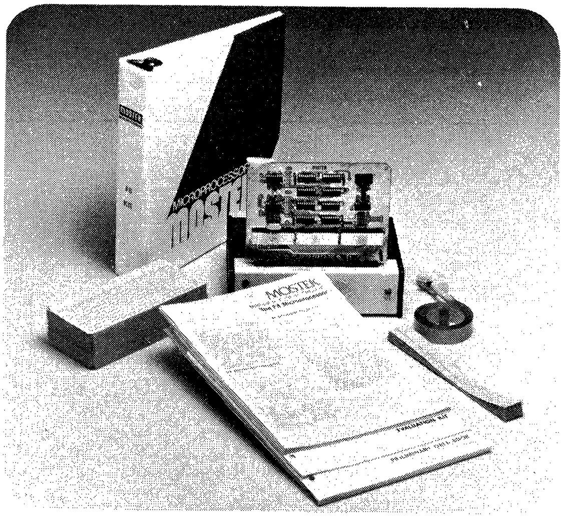
OPERATION

To operate, you simply attach a 110 or 300 BAUD ASCII terminal (such as a teletype or CRT monitor system) and +5 and +12V power supply. Using DDT—1, you can load, debug and modify your software in the 1K byte of RAM provided in the kit.

DDT—1 provides these features that can be accessed from the ASCII terminal to write and execute your own software.

□ Load command-loads memory from paper tape

□ Dump command-formats data and output to paper tape punch



Assembled F8 Evaluation Kit (79002) and Power Supply (79003)

□ Type command-examines blocks of memory

□ Memory Display and Modify command-examines and modifies memory one byte at a time

□ Copy command-moves blocks of memory from one location to another

□ Port commands-displays and modifies the 24 I/O lines

□ Hexadecimal Arithmetic commands-performs hexadecimal arithmetic

□ Execute command-executes programs at a specific location

□ Breakpoint command-debugs users software

ORDERING INFORMATION

Unassembled Evaluation Kit. Order number MK-79001.

Assembled and Tested Evaluation Kit. Order number MK79002.

Power Supply for Evaluation Kit. Order number MK79003.

MOSTEK®

F8 MICROCOMPUTER SUPPORT

Software Development Board (SDB-50/70)

SOFTWARE FEATURES

- □ 2K x 8 Operating System In ROM (DDT-2)
- □ 4K x 8 Resident Assembler In ROM
- □ Resident Text Editor Loadable In RAM

HARDWARE FEATURES

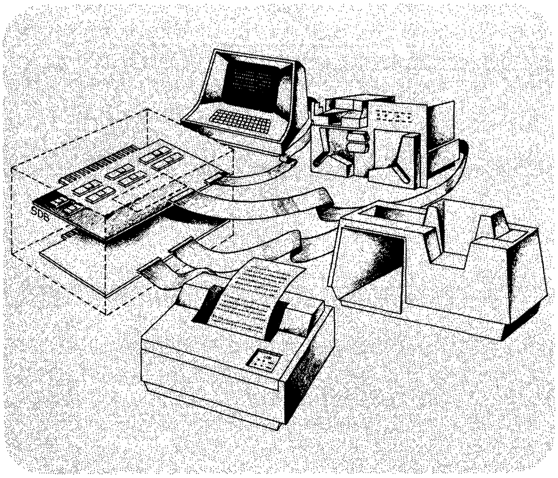
- □ 8K x 8 RAM Memory
- □ Four 8 bit I/O Ports
- □ Serial ASCII Interface (110-9600 Baud)
- □ Parallel Interface For High-Speed Reader/Punch
- □ Optional "Application Interface Module" (AIM)

GENERAL DESCRIPTION

The Software Development Board is a complete F8 Microprocessor System designed to aid in developing software for the F8. When combined with power supply, card cage and an ASCII terminal (such as a teletype), it will enable the user to develop the software for all types of F8 applications. This not only includes the ability to execute and debug user software, but also the ability to create and edit "source" listings (using the resident text editor) and assemble them into corresponding "object" code (using the resident assembler). Its other features include 8K x 8 of RAM (expandable with additional memory boards), a variable speed ASCII interface, and resident console and debugging routines. The SDB also includes an interface to an optional high speed paper tape reader/punch. Other peripherals such as a card reader and line printer may be added using an Auxiliary Interface Board.

USING THE SDB

The SDB may be used in two ways. First, as a stand-alone microcomputer, the SDB may be used to both generate (edit and assemble) and debug F8 Software using the 8K bytes of RAM and 32 bits of I/O available on the board. In many F8 applications, the SDB will thus provide all of the development capability the user will require. Other users, however, may prefer to emulate their application software in the circuit configuration required for their final system. This procedure can significantly reduce the development time for many types of applications. To support these users, an option is available for the SDB called AIM (Application Interface Module). With AIM, the user may apply all of the debug capabilities of the SDB operating system (DDT-2) directly to his final application configuration. As explained in the AIM descriptive literature, this is accomplished without any modifications to the hardware, software, or mechanical packaging of the users final system. The reader is referred to the AIM literature for further information on the use and operation of the SDB with the AIM option.



SPECIFICATIONS

Operating Temperature Range . . . 10°C to 40°C

Power Supply Requirements
+12V ± 5% @ 150mA
+5V ± 5% @ 1.2A
−12 5% @ 50mA
Board Size . . . 8.0"x 12.0" x 1.5"

Connector . . .100 pin edge connector (included)

DDT-2 COMMAND SUMMARY

The DDT-2 Operating system uses 10 basic commands:

.M s	Display and update memory at s
.M s,f	Tabulate memory block s,f
.P s	Display and update port s
.P s,f	Tabulate port block s,f
.E s	Execute program at s
.B s	Set breakpoint to exit program at s
.S s	Step single instruction at s in program
.L	Load tape into memory
.D s,f	Dump tape from memory block s,f
.C s,f,d	Copy memory block s,f to d

The s,f and d represent operands which may be hexadecimal constants, Literals (ASCII Equivalents), predefined mnemonics, or simple arithmetic expressions involving any combination of these. Allowable expressions are of the form ± n1 (=hhhh) ± n2 (=hhhh). . ., where the optional "=" may be used to display the four digit hexadecimal result. Expressions may be utilized in any of the DDT-2 commands, including a 'Dummy' command, 'H', which is provided to permit hexadecimal expression evaluation without performing any other operation.

MEMORY AND PORT COMMANDS (M,P)

The M and P commands provide the user with the means for sequentially accessing F8 I/O ports and memory. Both commands will accept either one or two operands (or operand expressions). With one operand, the contents of the memory or port locations indicated will be displayed and may be optionally modified. Typing carriage returns will automatically display the next successive locations which may also be modified. Typing a '∧ ' will either display the previous location or, if contents of the current location are being changed, display the new contents of the current location. This process will continue until a 'period' is typed to return to the command mode. A 'period' may also be used to abort improperly entered commands. In the example on the adjacent page note the ease with which relative branch offsets may be calculated (at 4106).

With two operands, the M and P commands provide a compact listing of memory or I/O ports. The contents of the addresses specified (inclusively) by the two operands are typed sixteen bytes per line as shown on the adjacent page.

EXECUTE, BREAKPOINT, SINGLE STEP (E,B,S)

The E command is used to execute all programs, including design aids such as the Assembler and Text Editor. The B command may be used to set a Breakpoint to exit from a program at some predetermined location for debugging purposes. At the instant of Breakpoint exit, the contents of all system registers (Scratchpad, Status, Accumulator, etc.) are transferred to a designated 115 byte area of the SDB RAM where they may be examined or modified. This portion of the SDB memory is called the 'Register Map'. It is also used to initialize system registers whenever execution is initiated (or resumed). Each register image in the Register Map may be accessed using the 'M' command followed by the predefined register mnemonic (or absolute address) of the storage location for that register (example :AC, :IS, :00, ..., :3F, etc). The E and B commands can thus be used together to initialize, execute and examine the results of individual program segments.

When a breakpoint is encountered, the address and accumulator are typed in the stepping format, and the user may continue stepping as above. The breakpoint is cleared automatically to prevent old breakpoints from cluttering up the program.

For a 'Trace' of the execution details of a routine, the programmer may use the S command to step one instruction at a time. With each step, the registers are loaded from the Register Map; the instruction executed; and the registers dumped back to the Register Map. After each step the Register Map may be examined or modified prior to executing the next instruction. The accumulator contents and the address of the next instruction to be executed are always typed after each Step. The programmer continues Stepping by typing carriage returns. In the example a short program has been loaded into memory locations 4100→4106 which will multiply Scratchpad Register R0 times R1 (MOD 256) and place the result in R2.

LOAD, DUMP, COPY (L,D,C)

The L and D commands load and dump object tapes thru the Object channel in standard F8 loader format. Checksums are used for error detection, and the addresses of questionable blocks are typed automatically while loading.

The C command will transfer the contents of the memory block specified by the first two operands to the memory block starting at the location specified by the third operand.

DDT-2 I/O CAPABILITIES

The SDB has 3 I/O channels, designated 'Console', 'Object', and 'Source', to which any suitable devices may be assigned. The Channel Assignment table is located in RAM where it may be updated using the M command. Where mnemonic designations have been predefined, they are automatically substituted for the Table Addresses and the dual byte contents of the table. The Table Addresses correspond to the I/O channels, with the Table Contents corresponding to the addresses of the peripheral driver routines. All the mnemonics used in the example are predefined in DDT-2 Firmware.

When a device is first assigned to a channel, the driver is automatically initialized as required. The user may write his own drivers, define mnemonics for them, and then use those mnemonics to assign them to channels as above. The user may also define mnemonics for any other addresses, such as starting points of programs or subroutines.

SAMPLE PROGRAM EXECUTION

```
. M : 00
:00 00 3
:01 01 17
:02 00
```

Set R0 = 3, R1 = 17 in Register Map

Ititialize Register Map

```
. M 4100
4100 90 70
4101 37 42
4102 03 C1
4103 0E 52
4104 2A 30
4105 5F 94
4106 BE 4101-*=FFFB^
4106 FB
```

```
        LIS   H'0'    R2 = 0
        LR    2,A
Loop    LR    A,2     R2 = R2+R1
        AS    1
        LR    2,A
        DS    0       R0 = R0-1
        BNZ   Loop    (R0 = 0?)
        (Calculate Branch Offset)
```

Load Program
(Solves R0 x R1
MOD 256)

```
 S 4100
*4101 00
*4102 00
*4103 17
*4104 17
*4105 17
*4101 17
*4102 17
*4103 2E
*4104 2E
*4105 2E
*4101 2E
*4102 2E
*4103 45
*4104 45
*4105 45
*4107 45.
```

```
ACC =   00; Next Instruction at 4101
        00                      4102
        17                      4103
        17                      4104
        17                      4105
        17                      4101
        17                      4102
        2E                      4103
        2E                      4104
        2E                      4105
        2E                      4101
        2E                      4102
        45                      4103
        45                      4104
        45                      4105
        45                      4107
```

Execute
(Single Step)

```
. M : 00
:00 00
:01 17
:02 45
```

R0 = 0
R1 = 17
R2 = 45 (= 3 x 17) MOD 256

Example Register Map

BLOCK MEMORY OPERATIONS

```
. M 4100, 4146
4100   00 00 00 00   11 11 11 11   22 22 22 22   33 33 33 33
4110   44 44 44 44   55 55 55 55   66 66 66 66   77 77 77 77
4120   88 88 88 88   99 99 99 99   AA AA AA AA   BB BB BB BB
4130   CC CC CC CC   DD DD DD DD   EE EE EE EE   FF FF FF FF
4140   AE 45 33 28   07 66 CC
```

List Memory Block
4100 thru 4146

```
. C 4100, 4117, 4118
. M 4100, 4146
4100   00 00 00 00   11 11 11 11   22 22 22 22   33 33 33 33
4110   44 44 44 44   55 55 55 55   00 00 00 00   11 11 11 11
4120   22 22 22 22   33 33 33 33   44 44 44 44   55 55 55 55
4130   CC CC CC CC   DD DD DD DD   EE EE EE EE   FF FF FF FF
4140   AE 45 33 28   07 66 CC
```

Copy Memory Block
4100 thru 4117 to location
4118 thru 412F and list.

```
. D 4100, 4146
 ******************
S4100
X00000000111111118
X22222222333333338
X44444444555555558
X00000000111111118
X22222222333333338
X44444444555555558
XCCCCCCCCDDDDDDDD8
XEEEEEEEEFFFFFFFF8
XAE4533280766CCB29
******************
```

Dump Memory Block 4100
thru 4146 in F8 Loader Format

Develop Aids

RESIDENT ASSEMBLER

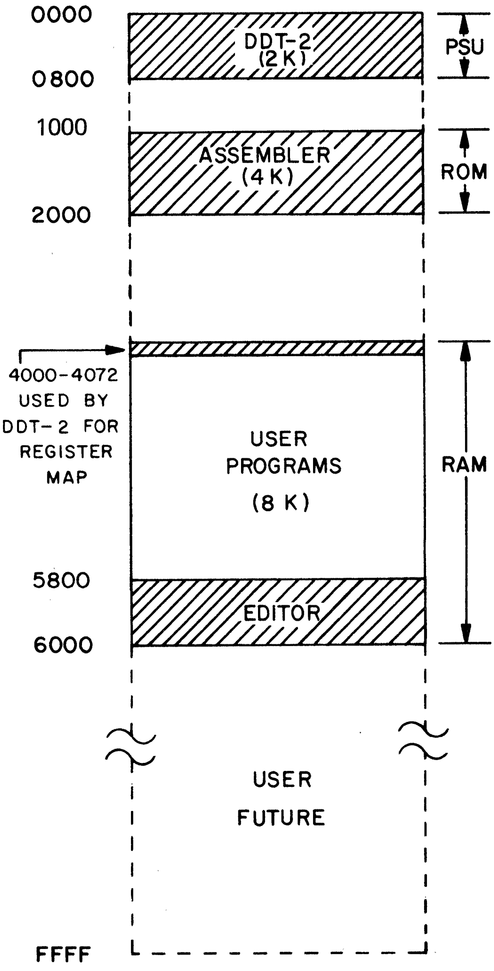
The Resident Assembler in the SDB is a program which translates F8 Assembly Languare Source Statements into Machine Language. The Machine Language produced by the Assembler (called an Object Module) is output in standard F8 Loader Format which may be loaded directly into RAM and Executed. Two Passes are required over the Source input for a complete assembly. The user also has the option of having an 'Assembled Source Listing' produced in addition to the Object Module. The Assembled Source Listing is printed (if desired) during Pass 2, while the Object Module is being buffered in memory. The Object Module is punched after an END statement is encountered or the object buffer has been filled. Buffering the Object Module eliminates the need for the third Pass required with many other Assemblers. This also permits the use of a single peripheral (such as a TTY) for outputting both the Assembled Source Listing and the Object Module without conflict. The only restriction on using the Assembler is that programs having more than 420 Labels must be assembled in sections:

All I/O for the Assembler is handled through the 'Console', 'Object', and 'Source' Channels provided by DDT–2. The Assembler receives Control characters (and responds) via the Console Channel, while the Source Channel is used for the Assembly Language input and the optional Assembled Source Listing output. The Object Channel is used to output the Object Module. All Channels are assigned to the serial ASCII Port when a teletype is the only available peripheral.

TEXT EDITOR

The Text Editor supplied with the SDB is in the form of a Paper Tape which may be loaded into RAM Memory (5800 thru 5FFF) and Executed. The various commands recognized by the Text Editor permit random access editing of ASCII characters strings (as would be stored on magnetic or paper tape). The data to be edited is read into memory where individual characters may be located by position or context. Approximately 5000 characters may be stored in the buffer area from 4100 thru 57FF. Character strings longer than this are edited in blocks with all loading and dumping of the buffer being performed automatically by the Text Editor. The Text Editor and Resident Assembler share the same buffer space and may be used alternately without reloading the Text Editor. While the primary application for the Text Editor is in the editing of Assembly Language Source Statements, it may be applied to any arbitrary ASCII character strings which are partitioned by 'Carriage Returns' into records of not more than 80 characters.

SDB MEMORY MAP



BLOCK DIAGRAM DESCRIPTION

Each of the major circuits shown on the block diagram is described below:

CPU – The Central Processing Unit for the SDB is the MK 3850. There are two eight bit I/O ports on the CPU. They are designated P0 and P1 and are available on the 100 pin edge connector. An eight bit bidirectional data bus is used for data transfer between the CPU and all other blocks in the system. The CPU generated control and timing signals for interface to the other blocks.

PSU – There are two 3851 Program Storage Units on the SDB. The ROM portion of these devices contain the DDT-2 operating system. They also provide four eight bit I/O ports (designated P10, P11, P14, and P15). Two of these ports (P11 and P14) are reserved for use by DDT–2, but the other two ports, as well as the timer and interrupt features or both PSUs, are available to the user.

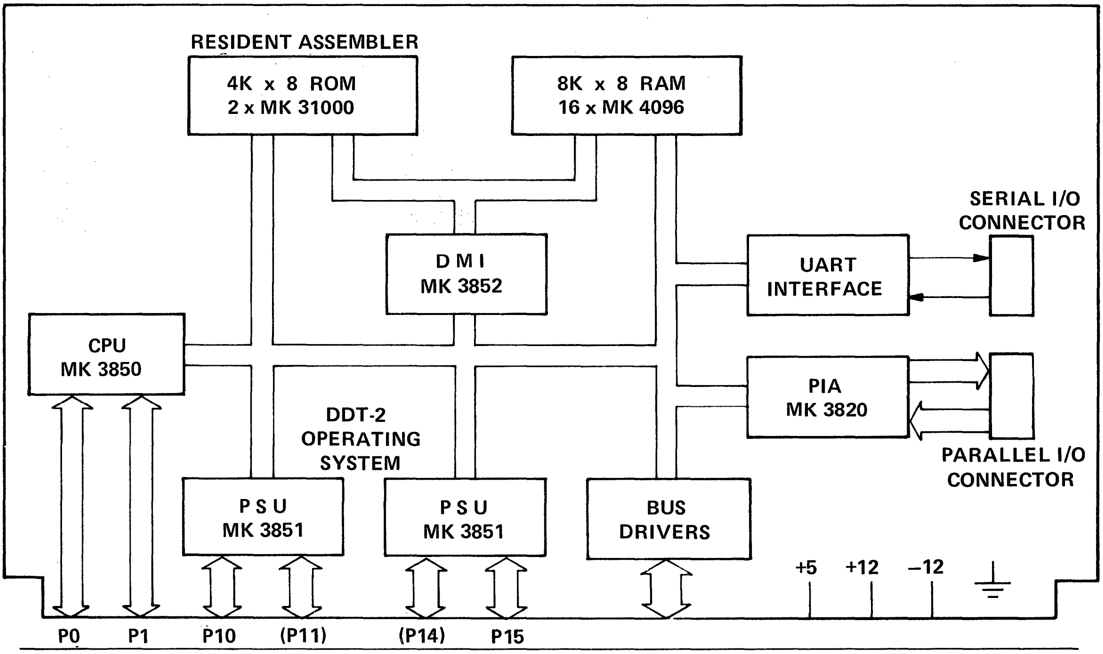
UART – The SDB uses a UART (Universal Asynchronous Receiver Transmitter) device to generate the variable speed serial ASCII interface. The BAUD rate is switch selectable from 110 to 9600 BAUD to be compatible with the various types of teletype and CRT terminals available.

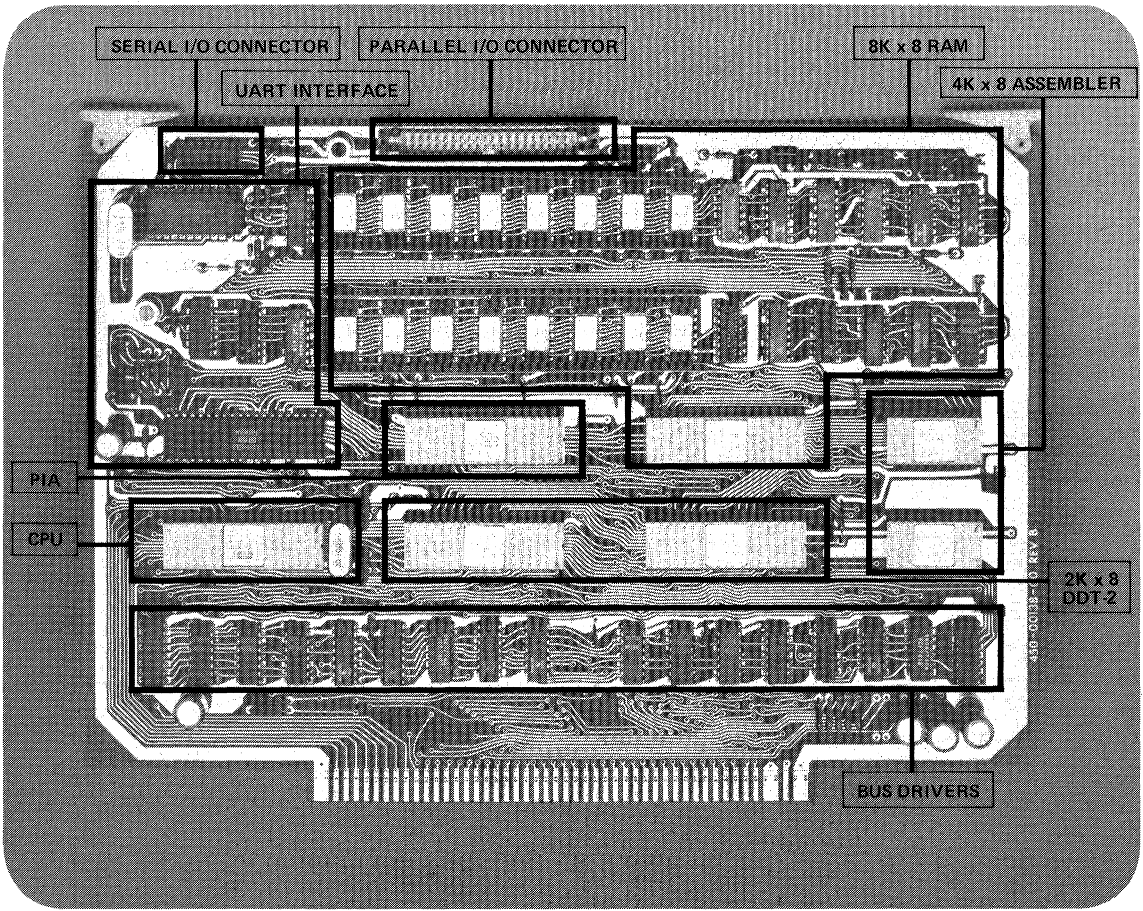
PIA – The MK 3820 Peripheral Interface Adapter provides the two eight bit I/O ports required for the optional high-speed reader/punch interface. This interface is available on the 40 pin 3M connector on the front edge of the SDB.

BUS DRIVERS – All F8 data bus, timing, and control signals are buffered and available on the 100 pin edge connector for expansion.

DMI – The MK 3852 Dynamic Memory Interface generates the timing and address signals for the sixteen MK 4096 (8K bytes) RAM and the two MK 31000 (4K bytes) ROMS.

FUNCTIONAL BLOCK DIAGRAM





ORDERING INFORMATION

PART NUMBER MK 79019

MOSTEK®

F8 MICROCOMPUTER HARDWARE SUPPORT

Application Interface Module (AIM-70)

FEATURES

□ Real time in-circuit emulation

□ Breakpoint insertion

□ Single step operation

□ Direct interface with Mostek's Software Development Board (SDB-50/70)

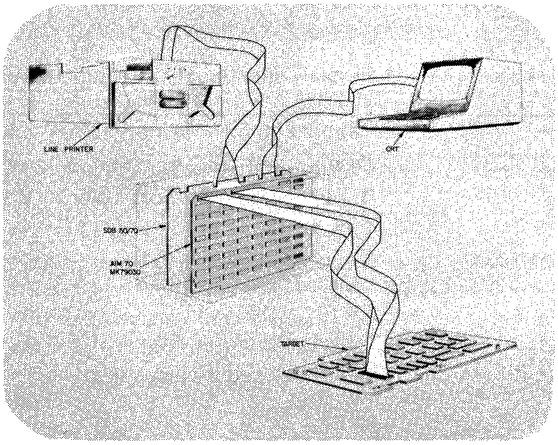
□ 3K bytes of RAM available during program development

GENERAL DESCRIPTION

AIM-70 (Application Interface Module) is a unique development aid for debugging MK3870 applications in the actual hardware and software configuration of the user's final system (referred to as the 'Target'.) To accomplish this, it is first necessary to emulate the Target ROM with RAM. This RAM must appear as ROM to the application, while retaining the ability to be loaded, debugged, and modified using peripherals independent of the Target. It is the purpose of AIM, used in conjunction with the Software Development Board (SDB-50/70) to provide these capabilities. With AIM-70, all of the peripheral and debugging capabilities of the SDB-50/70 may be applied directly to either the prototype or final production configuration of any MK3870 application; no modifications to the user's hardware, software, or mechanical packaging are required.

USING AIM

The pictorial diagram on the right shows how AIM-70 would typically be used during system development. Because the AIM-70 is an exact functional emulation of the MK3870, it may be directly inserted in the '3870 socket in the target system. Also, since the Target can be a production version of the user's application, product revisions and enhancements may be easily implemented.



As shown in the diagram, the AIM Board is usually mounted in a card cage with the Software Development Board (SDB). It is the purpose of the SDB to provide the user with the means for accessing and controlling the target system (via the AIM Board) during the program development phase. This provides access to all the development software and peripherals of the SDB without having to introduce any perturbations to the target system environment. AIM does not affect the peripheral expansion capabilities of the SDB.

SPECIFICATIONS

Operating Temperature Range.	0°C to 50°C
Power Supply Requirements	+5V ±5% @ 1.5A max. +12V ± 5% @ 100mA max.
Board Size.	8.5" x 12.0" x 1.5"
Connectors/Cables: 40-Pin Ribbon Cable (24" long)	

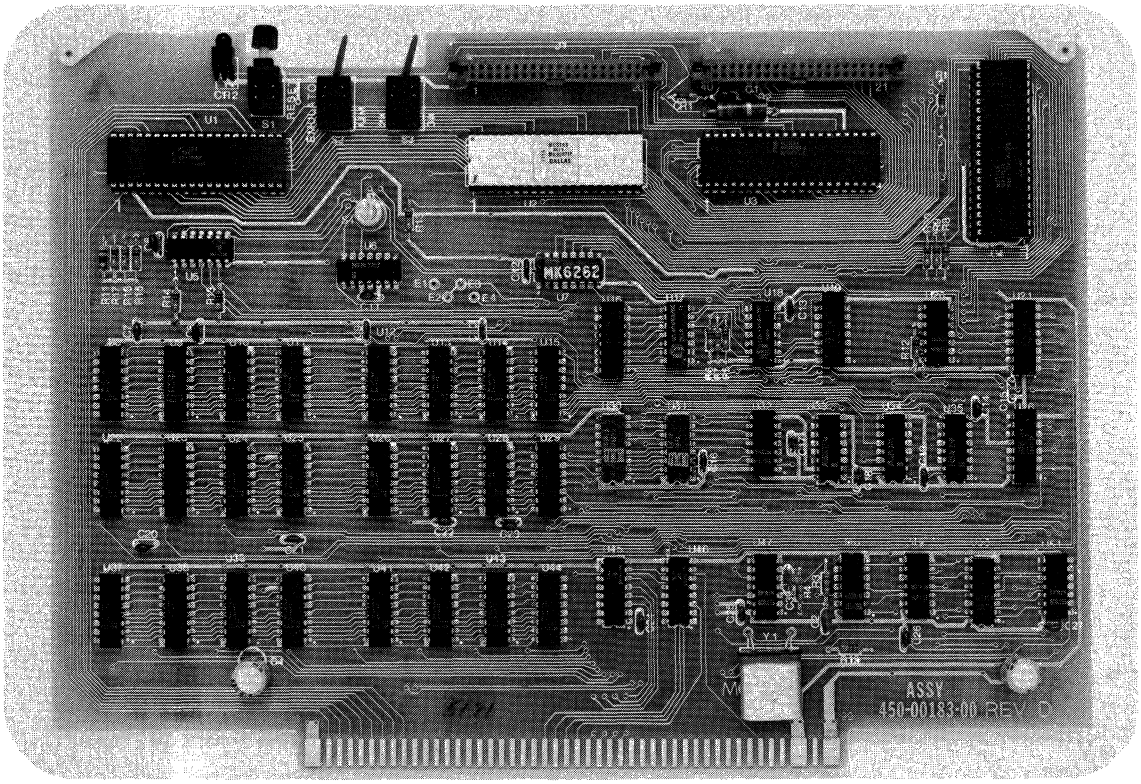
OPERATION DESCRIPTION

The hardware and software associated with AIM have been designed to retain the same command structure as the SDB. The only difference is that all operands (Memory Addresses or Port Addresses) which correspond to the 'Target' system must be preceded by the letter 'T'. The commands available with AIM are summarized below. Designations s, f and d stand for operands.

.M Ts	Display and update target memory at s
.M Ts, Tf	Tabulate target memory block s,f
.P Ts	Display and update target port s
.P Ts,Tf	Tabulate target port block s,f
.E Ts	Execute target program at s
.B Ts	Set breakpoint to exit target program at s
.S Ts	Begin single step execution at s in target program
.L T	Load formated tape into target memory
.D Ts,Tf	Dump formated tape from target memory block s,f
.C Ts,Tf,Td .C s,f, Td .C Ts,Tf,d	Copy memory block s,f in the SDB or target to the memory block location starting at address d in the SDB or target

Each of these SDB commands may be applied to any portion of the target system's port or memory map. This is accomplished by means of a 'handshaking' procedure between the CPU on the AIM and the CPU in the SDB. Handshaking is initiated whenever a target system breakpoint has been encountered, or the single step execution of a target instruction has been completed. Also, whenever handshaking is initiated, the contents of all target system registers (Scratchpad, Status, Accumulator, etc.) are transferred to a designated portion of the SDB memory map where they may be examined or modified. This portion of the SDB memory is called the 'Register Map' and is also used to initialize the target system register whenever execution is initiated (or resumed) in the target system.

AIM–70 PHOTO



BLOCK DIAGRAM DESCRIPTION

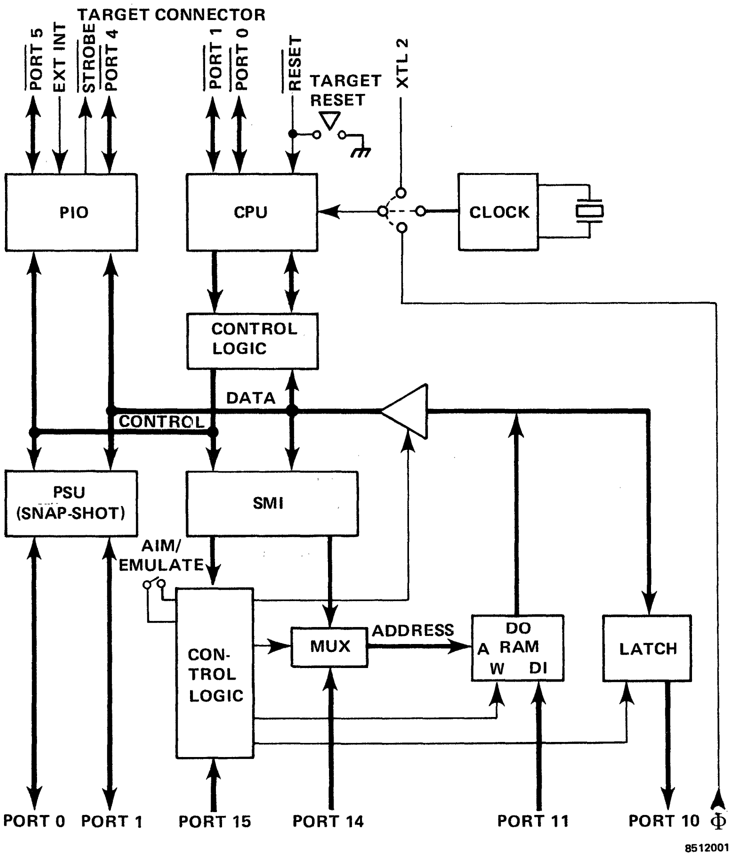
As shown in the block diagram, the AIM-70 contains all the functional elements necessary to emulate the MK3870. The portion of the handshaking software (called 'Snapshot') which resides in the target memory map is located in the PSU on the AIM Board. An MK3871 (PIO), is used to emulate the I/O timer, and interrupt features of the MK3870. Note that the AIM-70 contains 3K bytes of RAM memory-1K bytes more than required to emulate the MK3870's 2K bytes of ROM. The extra 1Kx8 of RAM is provided for use during program development for 'Patches' and to allow execution of the user's program prior to final optimization and code reduction. The AIM/EMULATE switch is provided to disable the extra 1K of memory and the upper 5 bits of the program and data counters. When the AIM/EMULATE switch is in the Emulate position, the AIM-70 is an exact RAM based equivalent of the MK3870. When the switch is in the AIM position, the expanded memory and handshaking are available for use during program development.

The AIM-70 clock may be from either the Target system, from an on-board crystal oscillator, or from the SDB-50/70 clock.

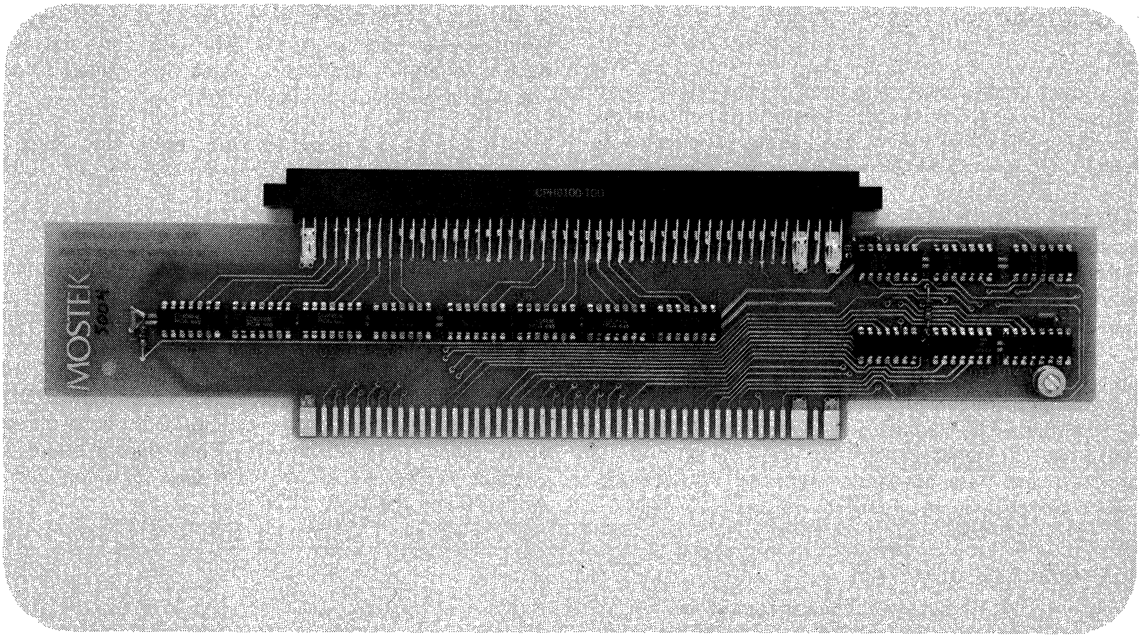
MULTI–3870 APPLICATIONS

For debugging applications incorporating more than one MK3870, multiple AIM–70s may be used in a single SDB/AIM development system. For these systems one AIM–70 plus an adaptor board is required for each MK3870 being emulated in the system. The adaptor board (designated AIM-70X) permits the use of a single SDB-50/70 for controlling up to seven AIM–70 boards. This adaptor board is physically inserted in the card cage between each AIM–70 and the SDB-50/70 bus.

AIM 70 BLOCK DIAGRAM



AIM–70X PHOTO



ORDER INFORMATION

Name	Description	Part No.
AIM–70 Operations Manual	Contains a complete description of the use and operation of AIM–70 and AIM–70X for developing software for 3870 applications.	MK79549
AIM–70	Includes the complete AIM–70 circuit board with the above described documentation.	MK79031
AIM–70X	Includes the AIM–70X circuit board with the AIM–70 operations manual.	MK79053
SDB-50/70	Includes the SDB-50/70 circuit board with complete documentation. The SDB-50/70 is used both with the AIM-70 and as a stand-alone microcomputer with resident firmware for F8 program assembly and text editing.	MK79019

*All prices are subject to change without notice and apply only within the U.S. and Canada

MOSTEK®

MICROPROCESSOR HARDWARE SUPPORT

F8 PSU Emulator (EMU–51)

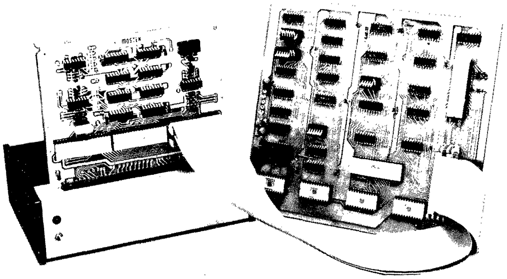
FEATURES

□ Completely emulates the MK 3851 Program Storage Unit (PSU)

□ Utilizes either MK 3702/1702A or 2708 PROMS

□ 2MHz operation

□ 40-pin adapter cable for simple fast interconnect



The F8 PSU Emulator is a development aid for designing and field testing F8 microprocessor systems which utilize one or more MK 3851 Program Storage Units (PSU). The Emulator is electrically equivalent to the PSU but is field programmable instead of mask programmable. This enables a user to obtain final hardware verification of all PSU programming prior to ordering custom PSUs. Also, since the Emulator "plugs in" like a PSU (via a male, 40 pin connector on the end of an "umbilical cord"), prototype systems can be converted to final production status by simply unplugging the Emulator(s) and plugging in the corresponding custom PSU(s).

The MK 3851 is a 40-pin integrated circuit that provides 1K bytes of ROM, two 8-bit latched I/O ports, a software programmable timer, and interrupt circuitry for vectored addressing and priority control. Multiple MK 3851 PSU chips can be used in a single system.

USING THE EMULATOR

The Emulator performs all the functions of the PSU:

ROM
INPUT/OUTPUT PORTS
INTERRUPT VECTOR
TIMER

The ROM section of the Emulator uses either four 256 x 8 bit ultraviolet erasable PROMs or a single 1K x 8 bit ultraviolet erasable PROM to provide non-volatile storage of the users' program. The PROM(s) should be programmed using a PROM programmer and then installed on the Emulator. The six ROM address select switches can then be used to establish the location of the PROM in the system memory map.

The input/output ports, interrupt vector, and timer functions of the Emulator, are implemented using an MK 3851/12001, which is provided on the Emulator Board.

SPECIFICATIONS

Operating Temperature Range . . . 10°C to 40°C

Power Supply Requirements (max.)

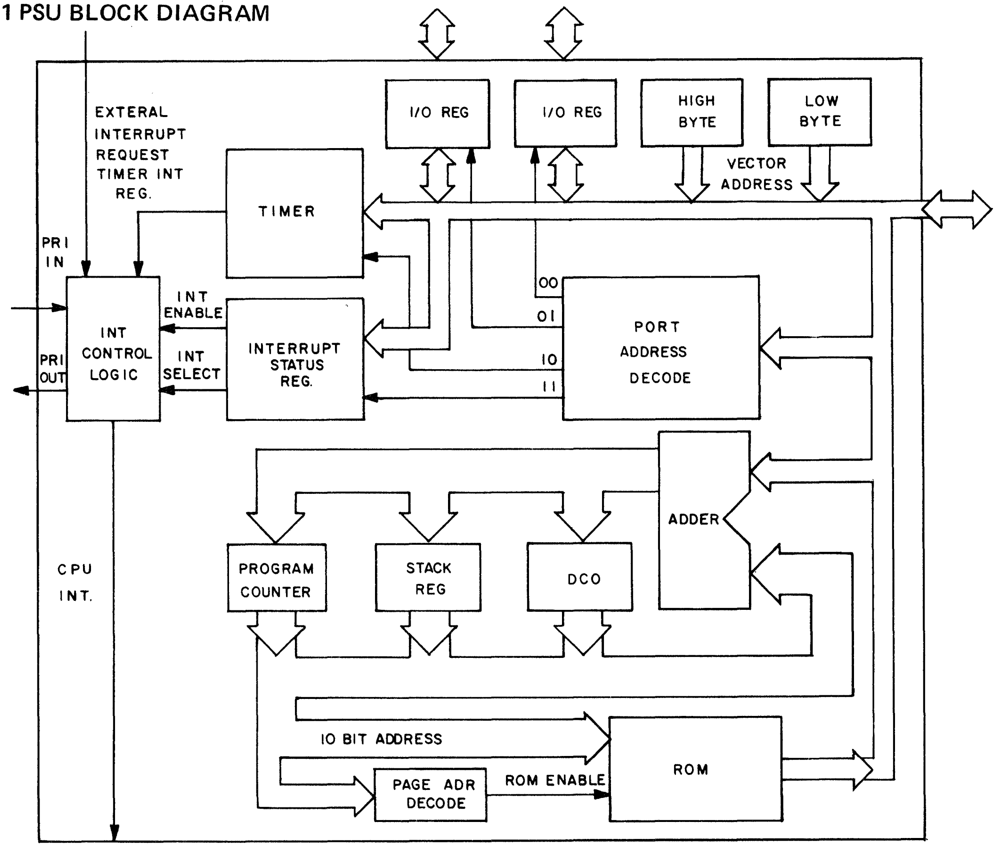
with 4, MK 3702s	with 1, 2708
+12V ± 5% @ 75mA	+12V ±5% @ 75mA
+5V ± 5% @ 500mA	+5V ±5% @ 350mA
–12V ± 5% @ 200mA	–12V ±5% – 100mA

Board Size . . . 8.2 in. x 9.19 in. x 1.0 in.

Connectors/Cables: (supplied with board)

- 5-Pin Power Connector
- 40-Pin Ribbon Cable (18 in. long)

MK 3851 PSU BLOCK DIAGRAM



EMULATOR BLOCK DIAGRAM DESCRIPTION

The Emulator block diagram shows how the PSU Emulator functions. Six ROM page select switches allow the user to place the 1K x 8 PROM memory at the desired location in the system memory map. Selection logic compares the most significant six bits of memory address from the Static Memory Interface (SMI) circuit with the six ROM page select switches, and causes the control logic to enable the PROM data output driver when the address is within range. Communication between the SMI and the CPU takes place on the Data Bus and the ROM Control Bus in the conventional manner. Note that either four 256 x 8 bit PROMS or a single 1024 x 8 bit PROM may be used to implement the PROM memory.

An MK 3851 PSU is used to provide timing and I/O port interface. The Data Bus is not connected directly to this PSU. Instead, port address translator logic modifies the contents of the Data Bus to allow the user to select the I/O port address desired.

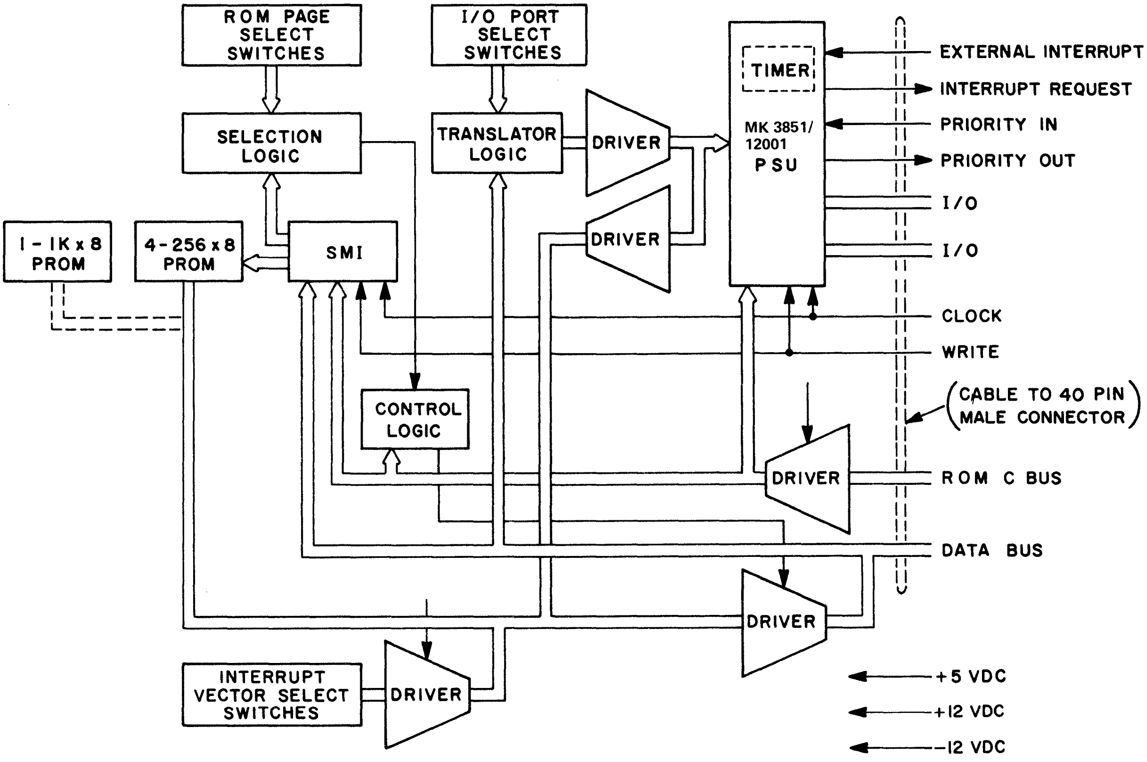
The I/O port addresses are determined by the position of the six I/O port select switches on the Emulator. The external interrupt line and the interrupt request line of the PSU on the Emulator provide interrupt control that allows the Emulator to perform in the system exactly as a production PSU.

The Interrupt Vector address on the Emulator is determined by the fifteen interrupt vector select switches, allowing the user to simulate the mask programmable vector address on the PSU. Since a PSU is used in the Emulator to provide the interrupt control logic, the interrupt control port status can still be modified normally.

The timer contained in the on board PSU circuit provides the timer function for the Emulator.

After the PROMs have been programmed and the switches have been correctly positioned the Emulator

EMULATOR BLOCK DIAGRAM



may be connected to the user's system by simply plugging the 40-pin connector into the corresponding PSU socket in the production/prototype circuit board.

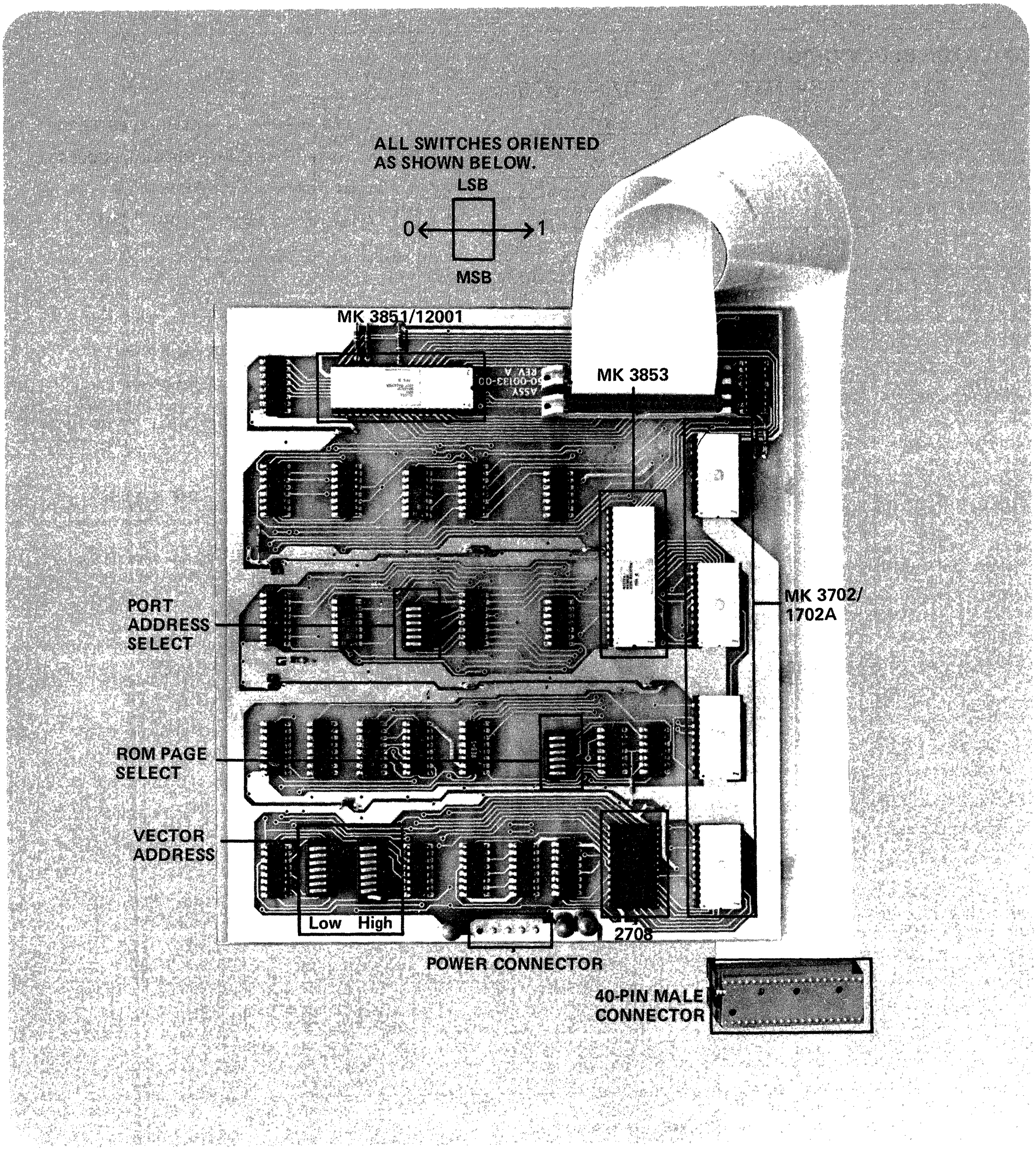
DOCUMENTATION

A complete set of documentation is provided with the Emulator to describe both the internal operation of the circuit board and the techniques for using it in system development. Also included are detailed instructions for ordering MK 3851 PSU's directly from the already verified data contained in the corresponding Emulator (i.e. the contents of the UV PROMs and the various switch positions).

ORDER INFORMATION

NAME	DESCRIPTION	PART NO.
EMU-51	PROM Emulator for the MK 3851. Includes power cable and 40-pin interface cable. PROMs not included.	MK 79018

*Prices are subject to change without notice and apply only in U.S. and Canada.


ALL SWITCHES ORIENTED
AS SHOWN BELOW.
LSB
0
1
MSB
MK 3851/12001
MK 3853
PORT ADDRESS SELECT
MK 3702/ 1702A
ROM PAGE SELECT
VECTOR ADDRESS
Low
High
2708
POWER CONNECTOR
40-PIN MALE CONNECTOR

MOSTEK®

MICROCOMPUTER SYSTEMS

Video Adaptor Board (VAB-2)

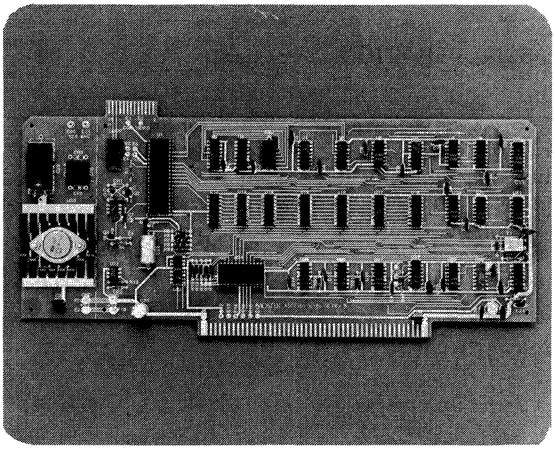
FEATURES

- □ Complete video interface system on one board
- □ Single supply (+5VDC or 12.6VAC) operation
- □ On board rectifier and regulator for 12.6VAC operation
- □ 16 lines of 64 characters
- □ Full ASCII character set - 128 symbols including upper/lower case letters
- □ Full cursor controls: ↑ ↓ ← → home, screen clear, carriage return, erase to end of line/screen; plus direct X-Y addressing
- □ 8 bit ASCII or 5 bit Baudot operation

DESCRIPTION

The VAB-2 is a single board video terminal based on the MOSTEK MK3870 single chip microcomputer. It functions as an interface between a 20mA full duplex serial data loop, an ASCII encoded keyboard, and an EIA standard video monitor. The only other external component required is a 12.6 volt transformer.

The P.C. board 'form factor' facilitates installation within most standard keyboard housings. Alternatively, the 2 inch power supply section may be cut off the P.C. board allowing the board to be inserted into a standard 12" card rack (such as Mostek's XAID-100 MK79034) for system use.



SPECIFICATIONS

Operating Temperature 0°C – 50°C

Power Supply Requirements
5VDC±5% @ 0.75A max.
or
8 – 14 VAC rms @ 0.75A rms max.

Board size (with power supply) 14" x 6.5" x 1"
(without power supply) 12" x 6.5" x 1"

Video output 1.5Vp-p into 75Ω (EIA RS-170)

Current loop input/output 20mA nominal opto-isolated 240V max loop to ground

Keyboard inputs – standard TTL compatible

CUSTOMER SUPPLIED EQUIPMENT

Keyboard – Cherry B70-4753 or equivalent

Monitor – SC Electronics, Inc. 10M915 or equivalent

Transformer – Stancor P8384 or equivalent

MICROCOMPUTER BASED

The heart of the VAB-2 is the MK3870 single chip microcomputer. The MK3870 provides the following functions:

Serial data link interface
Control character decoding
Cursor positioning
Keyboard interface

ASCII OPERATION

In ASCII mode, the VAB-2 receives and transmits an 8 bit code (parity bit = 0 on transmit, ignored on receive). Two stop bits are transmitted by the VAB-2, but only one stop bit is required by the VAB-2 receiver. The VAB-2 works equally well with external systems transmitting one, two, or more stop bits. Available Baud rates for ASCII are 300 and 110.

See also Figure 1 — ASCII character set, and Table 1 — ASCII control characters.

BAUDOT OPERATION

In Baudot mode, the VAB-2 receives and transmits a 5 bit code (compatible with Model 15, Model 28, or similar Teletypes™). Two stop bits are transmitted, but only one stop bit is required by the VAB-2 receiver. The VAB-2 works equally well with external systems transmitting one, 1.5, or more stop bits. Available Baud rates for Baudot are 74.2 and 45.45. In Baudot mode, the only control codes available are carriage return and line feed. The Baudot "Letters" and "Figures" shift characters are generated automatically as required. Keys on the ASCII keyboard which generate codes having no equivalent Baudot code are ignored. ASCII code "Rubout" ($7F_{16}$ or 177_8) generates a "Letters" shift to facilitate synchronization of the distant end receiver.

ASCII CHARACTER SET

α β γ δ ϵ θ ι λ μ ν π Σ φ ψ ω Ω 0 1 2 3$^{0\ 2}$ _ ÷ ≏ √ ∫ | ← ↑↓
! " # $ % & ' ( ) * + , – ; / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ] ^
` a b c d e f g h i j k l m n o p q r s t u v w x y z { ¦ } ~ ▒

Figure 1

BAUDOT CHARACTER SET

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
– ? : * 3 $ & # 8 ( ) . , 9 0 1 4 ! 5 7 ; 2 / 6 "

Figure 2

FUNCTIONAL DIAGRAM

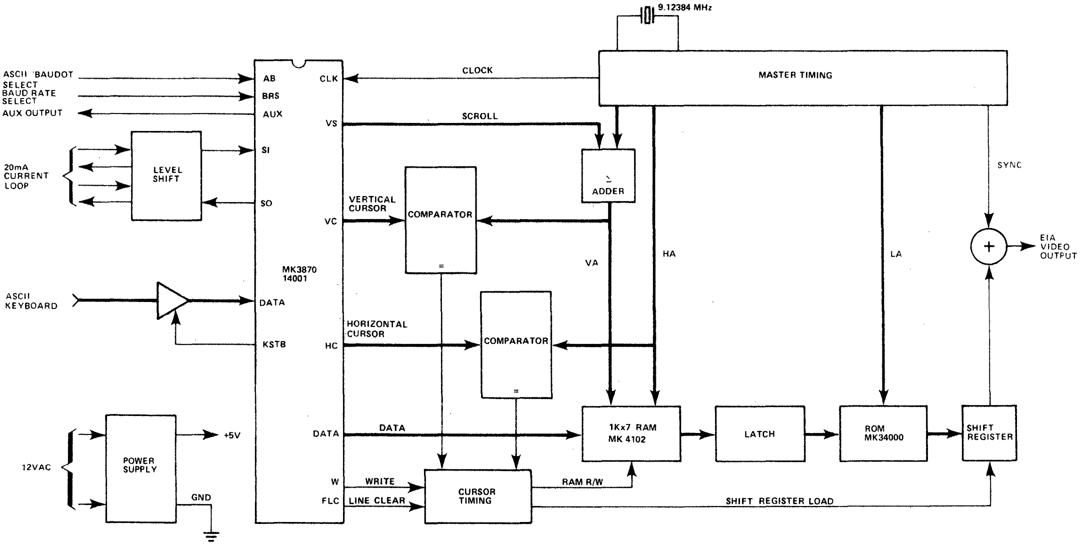


Figure 3

OCTAL	HEX	CNTL	FUNCTION	
004	04	D	HOM	Home – moves cursor to upper left corner of screen
005	05	E	EOL	Erase end of line – erases current line from right margin to current cursor position (1600mS max)
006	06 1	F	EOS	Erase end of screen – erases lines from bottom of screen to, but not including, current line (400mS max)
010	08	H	BS	Back space – move cursor left one column unless already in left most column
011	09	I	HT	Horizontal tab – moves cursor right one column unless already in right most column
012	0A	J	LF	Line feed – moves cursor down one line, scrolls screen up if already on bottom line
013	0B	K	VT	Vertical tab – moves cursor up one line, scrolls screen down if already on top line
014	0C	L	FF	Form feed – clears screen and homes cursor (400mS)
015	0D	M	CR	Carriage return – moves cursor to left margin
020	10	P	DS	Down shift sequence – causes character following DS to be interpreted as printable rather than control. Required for lower 32 symbols (Greek and math), but may be used with any characters.
021	11	Q	DC1	Device control – sets AUX bit
023	13	S	DC3	Device control – clears AUX bit
033	1B		ESC	Start cursor sequence – ESC + ΔV ΔH adds ΔV modulo 16 to vertical cursor address ΔH modulo 64 to horizontal cursor address ESC = ΔV ΔH sets vertical cursor address to ΔV modulo 16 horizontal cursor address to ΔH modulo 64
177	7F		DEL	Delete – moves cursor left one column, unless cursor was already on leftmost column; erases new position

TABLE 1. – ASCII CONTROL CHARACTERS

CHARACTER GENERATOR

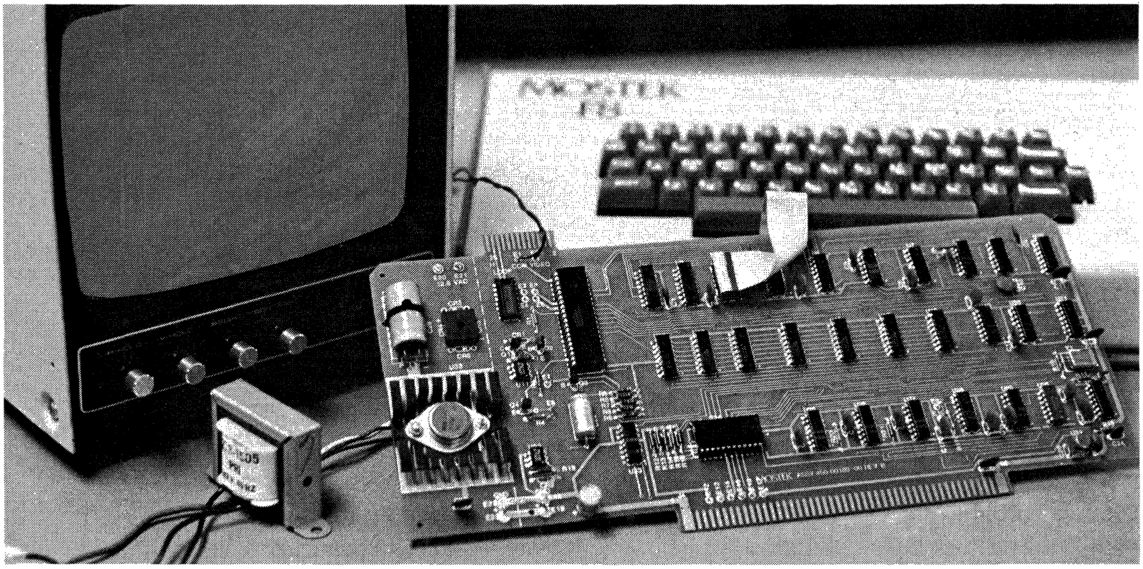
The VAB-2 is shipped with an MK34073 (2K x 8) character generator ROM, providing 128 displayable characters (see Figure 1 – ASCII character set). For custom applications, the MK34073 ROM may be removed and an MK2708 type PROM (1K x 8) installed, programmed with the user's custom font (external +12V and -5V or -12V supplies required for some PROMs). Alternatively, for high volume applications, a new ROM mask may be ordered. The MK34000 series can provide two complete 128 character sets per ROM. Provision is made for wiring the AUX bit to the ROM for program-selectable character font.

AUXILLARY BIT OUTPUT

A special output (AUX) is provided for custom control applications. AUX is capable of driving one TTL load, and is brought out to the P.C. edge connector. AUX is cleared upon power up and each time a DC3 character is recieved. AUX is set upon receipt of a DC1 character.

KEYBOARD

The VAB-2 interfaces directly with standard ASCII encoded keyboards. Although normally used with active high data and strobe keyboards, provision is made for active low keyboards.



CUSTOMER SELECTABLE OPTIONS

- □ 50/60 Hz (Strap option)
- □ 110/300 Baud ASCII (strap option)
- □ 74.2/45.45 Baud Baudot (strap option)
- □ MK34000 series ROM or MK2708 type PROM character generator (strap and population option; MK34073 standard)
- □ 5VDC or 12VAC operation (strap and population option; 12 VAC standard)
- □ Serial loop connector — 16 pin DIP socket or 26 pin edge connector
- □ Active high or active low keyboard input (population option; active high standard)
- □ Custom features and/or character generator for high volume OEM applications (one-time mask charge applicable)

ORDER INFORMATION

NAME	DESCRIPTION	PART NO.	PRICE
VAB-2 Operations Manual	Detailed description of the use and operation of VAB-2	MK79560	$ 1.50
VAB-2 Source Listing	Source Listing of the 3870 Firmware used in VAB-2	MK79561	$ 15.00
MK3870/ 14001 Firmware Package	Pre-programmed 3870 used with VAB-2 plus the Operations Manual and Source Listing described above	MK79056	$ 50.00
VAB-2	Assembled and tested VAB-2 Circuit Board plus the Operations Manual and Program Source Listing	MK79052	$195.00

*Prices are subject to change without notice and apply only to U.S. and Canada.

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Software Development Board (SDB-80)

HARDWARE FEATURES

□ Available with choice of either 4K or 16K bytes of RAM

□ Four 8-bit I/O ports with handshake lines

□ Serial ASCII interface (110-9600 BAUD)

□ Fully buffered for system expandability

□ Four counter/timer channels

□ On board capacity for 5K bytes of PROM or 20K bytes of ROM

SOFTWARE FEATURES

□ 2K x 8 Operating System in ROM (DDT-80)

□ 8K x 8 assembler/editor in ROM (ASMB-80)

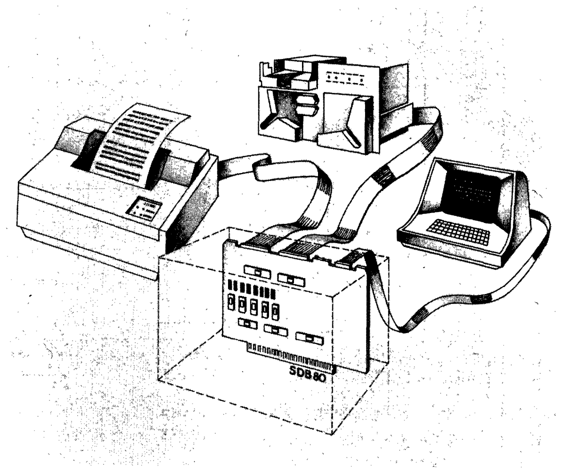
□ Channeled I/O for user convenience

GENERAL DESCRIPTION

The SDB-80 is a stand-alone microcomputer designed by MOSTEK around the advanced Z80 microprocessor family. It contains more on-board firmware and RAM memory than any previously offered single board microcomputer, plus all the features of the industries' most sophisticated microprocessor. This board represents the very latest in state-of-the-art technology by utilizing MOSTEK's new 16K Dynamic RAM memories. The SDB-80 also is the first single board microcomputer to offer a complete package of software development aids in ROM. This 10K byte firmware package is included with the SDB-80 and provides the ability to generate, edit, assemble, load, execute, and debug Z80 programs for all types of applications.

USING THE SDB-80

In addition to functioning as a stand-alone development aid, the SDB-80 is fully expandable through the addition of optional add-on circuit boards. It may also be utilized directly in OEM applications by inserting custom programmed ROM or PROM memories into the sockets provided on the board. For these OEM applications, partially populated versions of the SDB-80 (designated OEM-80) are available without the standard system firmware, and with quantity discounts.



SYSTEM FIRMWARE

A standard feature of the SDB-80 is a complete package of development software aids which are resident in the five MK34000, 2K x 8 ROM memories located on the board. This firmware includes a sophisticated operating system, debug package, assembler, and text editor. Among the many features provided are execute and breakpoint commands, console routines for examining and/or modifying memory and port locations, object load capability for both absolute and relocatable object modules, I/O driver routines for a variety of standard peripheral devices, and channeled I/O for user defined peripheral drivers. The presence of this software in ROM provides instant access to these development aids, eliminating the time-consuming requirement of loading the software from some peripheral device into RAM. Another key feature of having the development aid software in ROM is that the entire RAM space is available for the user's programs.

ELECTRICAL SPECIFICATIONS

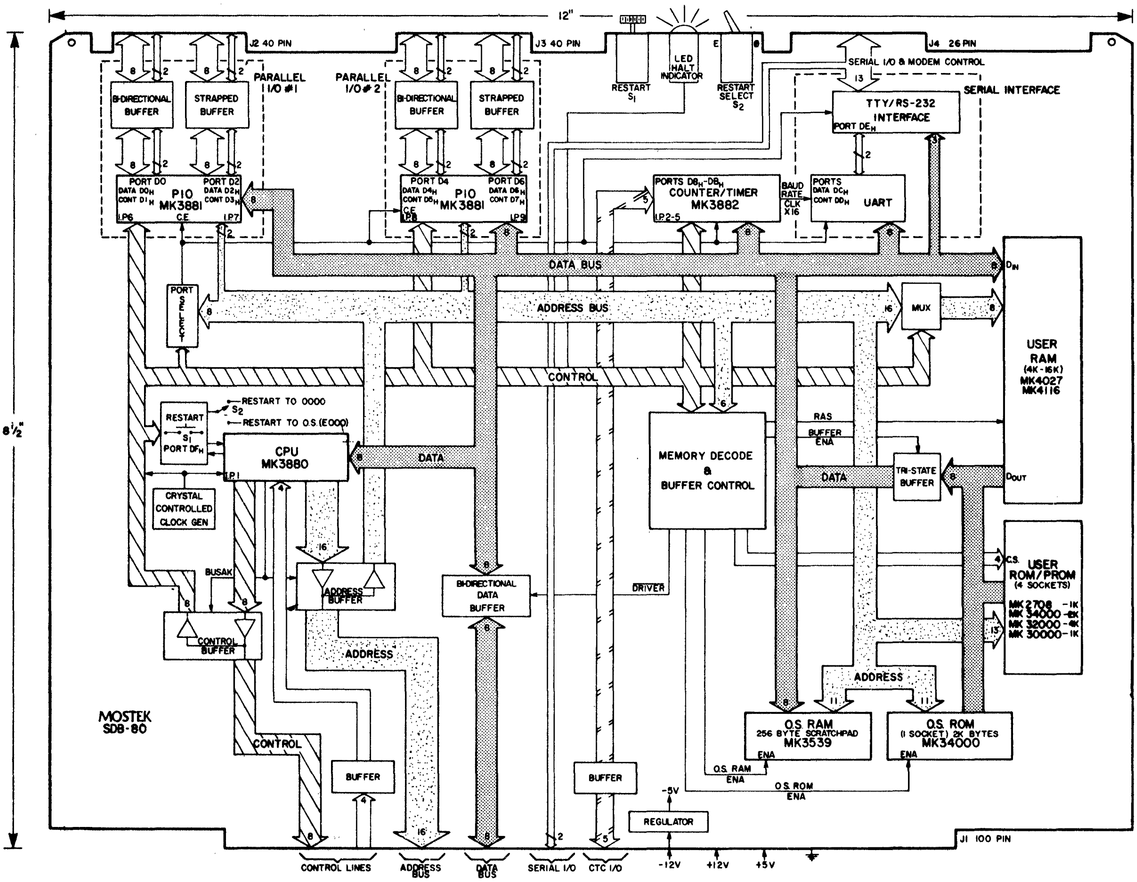
Operating Temperature Range . . .0°C to 50°C

Power Supply Requirements (Typical)

+12V ± 5% @ 175 mA
+ 5V ± 5% @ 1.5 A
−12V ± 5% @ 100 mA

Interface Levels . . . TTL Compatible

SDB-80 FUNCTIONAL BLOCK DIAGRAM



NON-RESIDENT SOFTWARE AVAILABLE

XFOR-80 Fortran IV Cross Assembler. Assembles Z80 programs but is written in Fortran IV. It is useful for persons desiring to perform Z80 assembly on mini-computers such as the PDP-11. It is furnished as a Fortran IV source deck. (MK78117)

XMDS-80 8080A Cross Assembler. Performs the same function as the Fortran IV Cross Assembler, except that it is designed to be used with an Intel MDS system. It is furnished as an object tape in Intel compatible Hex format. (MK78115)

XMDS-80D This is identical to the XMDS-80 except that it is compatible with Intel MDS systems which use floppy disks. It is furnished as object code on an MDS compatible floppy diskette. (MK78116)

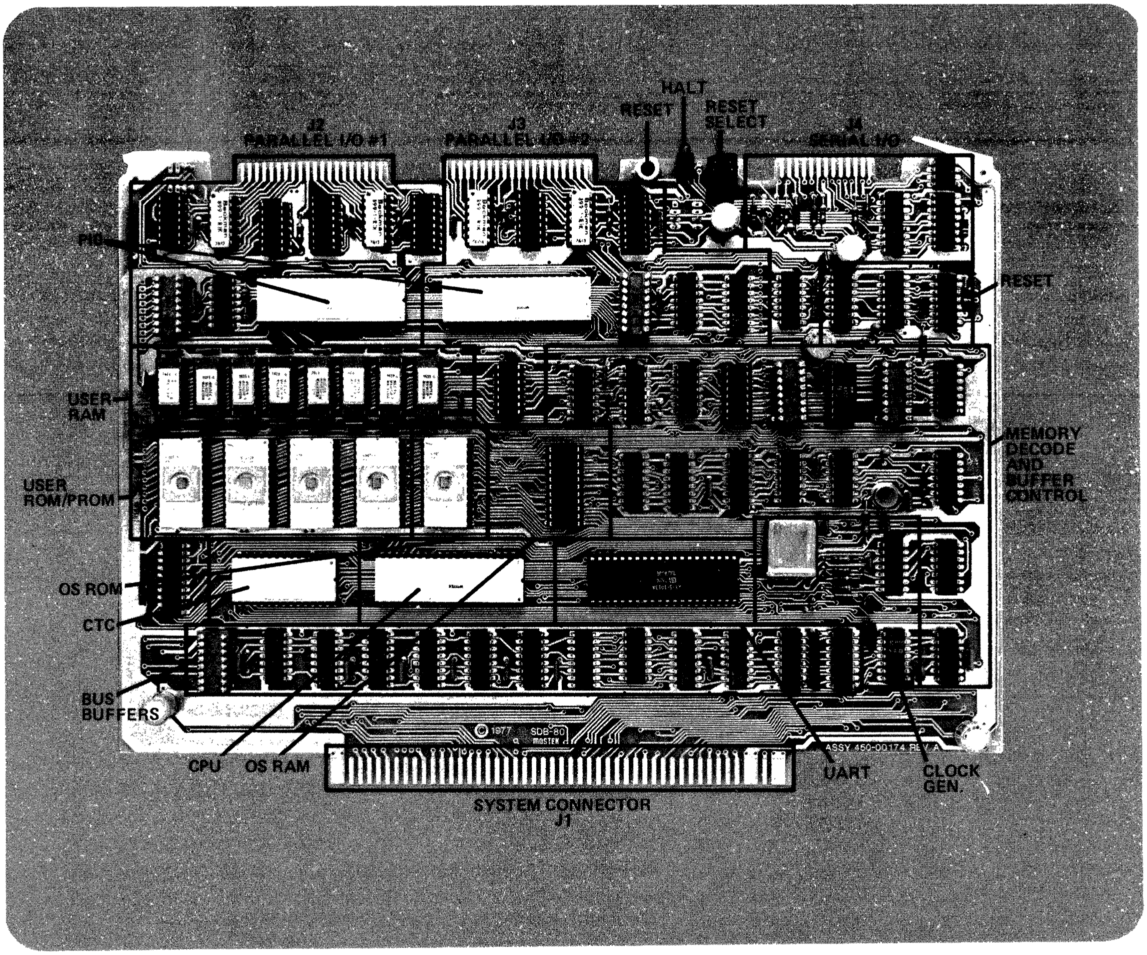
COMPATIBLE ADD-ON BOARDS

RAM-80A A 16K byte RAM board for users requiring the most economical means for expanding memory. (MK78107)

RAM-80B Combination memory and I/O expansion board. The memory may be configured to have a capacity of 16K, 32K, 48K, or 65K bytes of RAM. The board also provides four 8-bit I/O ports from two Z80 PIO circuits. (MK78108)

AIM-80 In-circuit emulation capability is added to the SDB-80 by using the AIM-80 board (Application Interface Module). This board also provides other debugging capabilities such as TRACE and SINGLE STEP, and a DISASSEMBLER. (MK78132)

AIM-72 Provides in-circuit emulation capability for emulating the MK3870 family of single chip microcomputers. Compatible with SDB-80 only in AID-80F floppy disk system environment.



FLP-80 The FLP-80 interfaces the SDB-80 to two floppy disk drives. Software drivers are included with the board. (MK78111)

OTHER ACCESSORIES AVAILABLE

PPG-08 PROM Programmer module for programming MK 2708 UV Erasable PROM memories. Interfaces directly with the SDB-80. (MK79033)

XAID-100 System package which includes a 13-slot card cage, enclosure and power supply. (MK79034)

AID-80F Complete Z80 Microcomputer system which includes enclosure with 6 slot card cage, power supply, cooling fan, OEM-80, RAM-80B, FLP-80, two Floppy Disk drives, and FLP-80DOS software package. (MK78125)

XAID-102 Three-slot card cage. (MK79028)

XAID-103 Wire wrap card (MK79023)

XAID-104 Extender card (MK79024)

MECHANICAL SPECIFICATIONS

Domestic Version

Board Size: 8.5" x 12.0" x 0.65"
Bottom Connector: 100 pin, 125 mil centers
Top Serial Connector: 26 pin, 100 mil centers
Top Parallel Connectors: 40 pin, 100 mil centers

Double Eurocard Version Available

Board Size: 250mm x 233.4mm x 18mm
Connector: Dual 64 pin Eurocard Conn.

ORDER INFORMATION FOR THE SDB-80 AND OEM-80

Z80 Microcomputer System Components		SDB-80 Package 'A' MK78101 includes:	SDB-80 Package 'B' MK78102 includes:
DESCRIPTION	PART NO.		
OEM-80* with 256 bytes of static RAM, 4K bytes of dynamic RAM, and sockets for ROM and PROM.	78121	X	
OEM-80* with 256 bytes of static RAM, 16K bytes of dynamic RAM and sockets for ROM and PROM.	78123		X
DDT-80 operating system in 1-MK34000 2K x 8 ROM.	78118	X	X
ASMB-80 Resident Assembler and Text Editor in 4 MK34000 2K x 8 ROMs.	78119	X	X
TTY Interface Cable (XAID-800).	79036	X	X
EIA/RS-232 Interface Cable (XAID-802).	79038	X	X

* The Circuit Board for the SDB-80 and the OEM-80 are identical and include 2 MK3881 PIOs, 1 MK3882 CTC, and 1 UART, plus the associated circuitry for control and buffering of all bus and I/O signals. Sockets are provided for expansion of on-board system RAM and ROM/PROM.

SYSTEM DATA SHEETS

MK78519	RAM-80A/B
MK78537	AIM-80
MK78538	FLP-80
MK79576	AIM-72

MK79081	PPG-8/16
MK78568	AID-80F
MK79081	PPG-8/16
MK79552	XAID-103
MK79552	XAID-104

MOSTEK®

MICROPROCESSOR SUPPORT

Z80 Software Development Board (SDB-80E)

HARDWARE FEATURES

- □ Available as board or complete system
- □ 4K bytes of RAM, expandable on board to 16K Bytes
- □ Four 8-bit I/O ports with handshake lines
- □ Serial ASCII interface (110-9600 BAUD)
- □ Fully buffered for system expandability
- □ Four counter/timer channels
- □ On board capacity from 5K bytes of PROM to 20K bytes of ROM
- □ Double euro-card format

SOFTWARE FEATURES

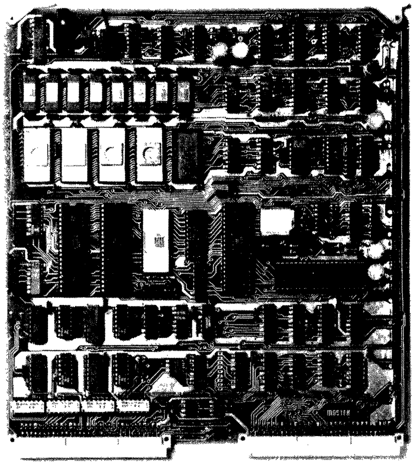
- □ 2K x 8 Operating System in ROM (DDT-80)
- □ 8K x 8 assembler/editor in ROM (ASMB-80)
- □ Channeled I/O for user convenience
- □ Double euro-card format

GENERAL DESCRIPTION

The SDB-80 is a stand-alone microcomputer designed by MOSTEK around the advanced Z80 microprocessor familiy. It contains more on-board firmware and RAM memory than any previously offered single board microcomputer, plus all the features of the industries most sophisticated microprocessor. This board represents the very latest in state-of-the-art technology by utilizing MOSTEK's new 16K Dynamic RAM memories. The SDB-80 also is the first single board microcomputer to offer a complete package of software development aids in ROM. This 10K byte firmware package is included with the SDB-80 and provides the ability to generate, edit, assemble, load, execute, and debug Z80 programs for all types of applications.

USING THE SDB-80

In addition to functioning as a stand-alone development aid, the SDB-80 is fully expandable through the addition of optional add-on circuit boards. It may also be utilized directly in OEM applications by inserting custom programmed ROM or PROM memories into the sockets provided on the board. For these OEM applications, partially populated versions of the SDB-80 (designated OEM-80) are available without the standard system firmware, and with quantity discounts.



SYSTEM FIRMWARE

A standard feature of the SDB-80 is a complete package of development software aids which are resident in the five MK 34000, 2k x 8 ROM memories located on the board. This firmware includes a sophisticated operating system, debug package, assembler, and text editor. The presence of this software in ROM provides instant access to these development aids, eliminating the time-consuming requirement of loading the software from some perpheral device into RAM.

Another key feature of having the development aid software in ROM is that entire RAM space is available for the user's programs.

Debug (DDT-80) includes:

- □ object program Load/Dump
- □ Memory or Port Examine/Change
- □ Breakpoint/Execute
- □ Logical/Physical I/O mapping (with user expandable drivers)
- □ Drivers for Standard Peripherals

Z80 SYSTEM SUPPORT

SYS-80E	SDB-80E with MK 78039	
	4k byte	MK 78040
	16k byte	MK 78041
MOSTEK TERMINAL	Complete Video Display Unit	MK 78037

Z80 PROCESSOR ELEMENTS

OEM-80E	with 4k bytes RAM with 16k bytes RAM	MK 78122 MK 78124
DDT-80	Debug 2k Byte ROM	MK 78118
ASMB-80 EDIT-80	Assembler four 2k Byte ROM's (including the Editor)	MK 78119
SDB-80E	with OEM-80 + DDT-80 + ASMB-80 + EDIT-80 + documentation	
	4k byte RAM	MK 78103
	16k byte RAM	MK 78104

Z80 HARDWARE SUPPORT

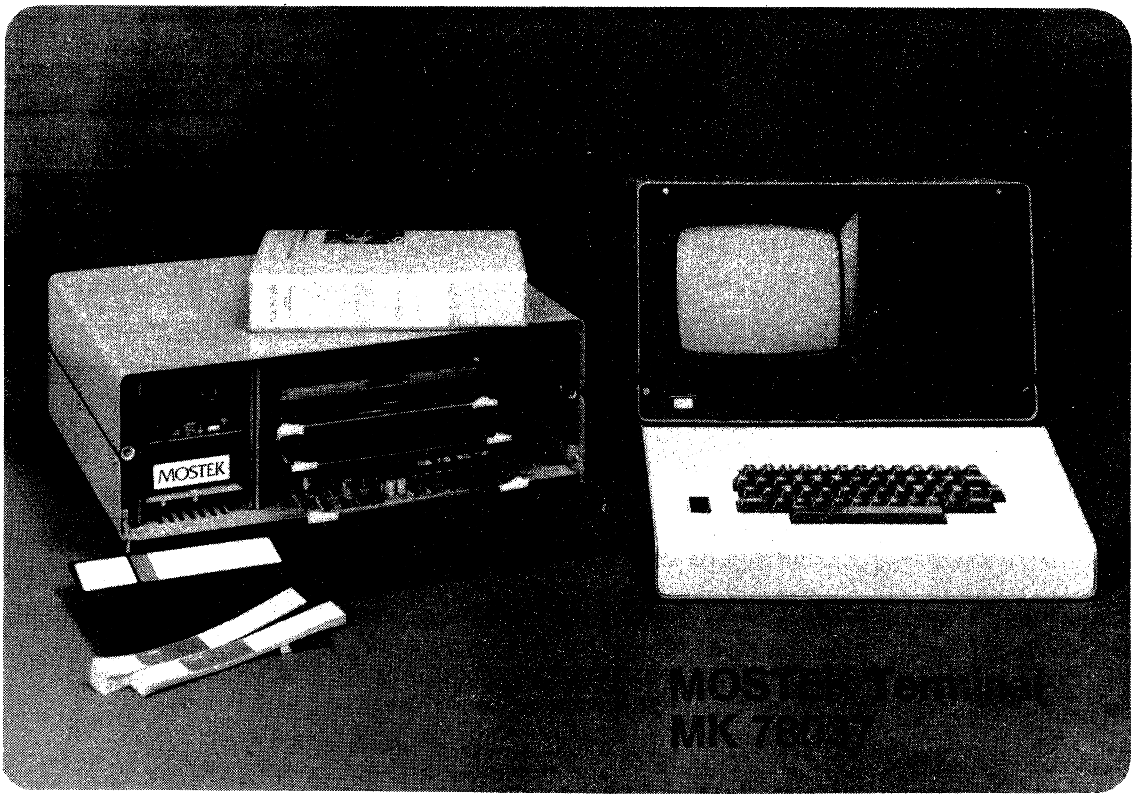
RAM-80AE	16k RAM	MK 78109
RAM-80BE	16k RAM, 2 PIO	MK 78110
XRAM-80	16k expander for RAM-80BE	MK 78126
AIM-80E	I.C.E. (In-Circuit-Emulation)	MK 78106
FLP-80E	Floppy Interface	MK 78112
RIO-80E	16k PROM, 2 PIO, 1-CTC, UART	MK 78128
Universal Display Interface	Serial UDI-S Parallel UDI-P Screen read option	MK 78033 MK 78035 MK 78036
	EIA Interface Cable For SDB-80E	MK 79058
	TTY Cable for SYS-80E	MK 79059
PPG-08	PROM Programmer for MK 2708 1kx8 UV PROM's with enclosure (requires MK 79060)	MK 79033
	PPG-08 Cable for SYS-80E	MK 79060
	Wire Wrap Card	MK 79063
	Extender Card	MK 79062
BACK-80E	Backplane Card, 6 slot	MK 79054
	Development Station Z80 6 total slots, power supply, no cards	MK 78039

Z80 SOFTWARE SUPPORT

XFOR-80	Fortran IV Cross Assembler requires 20k, 16 bit words	
	Card Deck	MK 78117C
	Paper Tape	MK 78117P
XMDS-80	8080 MDS Cross Assembler Paper Tape	MK 78115
XMDS-80D	8080 MDS Cross Assembler Soft sectored diskette	MK 78116
	Listing for DDT-80	MK 78534
	Listing for ASMB-80	MK 78536

DOCUMENTATION

Z80 CPU	Manual	MK 78070
Z80 PIO	Manual	MK 78071
SDB-80E	Manual	MK 78548
RAM-80E	Manual	MK 78545
AIM-80E	Manual	MK 78546
SDB-80E	Literature Package includes CPU, PIO, SDB-80 manuals plus data sheets	MK 78549
PPG-08	Manual	MK 78532
Z80	Pocket reference manual	MK 78516
Z80	Programming manual	MK 78515



MOSTEK TERMINAL MK 78037

FEATURES

- □ Self contained visual terminal
- □ 24 line, 80 character per line display
- □ Baud rate selection 110–9600
- □ Current loop or V24
- □ Comprehensive commands

GENERAL

A keyboard and a monitor provide together a “teletype” replacement video terminal for MOSTEK development systems, that can also be used in other applications. The terminal is completely self contained with its own power supply and electronics, requiring only the serial communication lines to the computer.

KEYBOARD

The keyboard obtains its power from the display unit, the coded keyboard information and power connections are made over a 25 pin type D connector.

DISPLAY ELECTRONICS

The display electronics uses the MOSTEK universal display interface board (MK 78033), power being provided within the terminal itself. The set of available functions is fully described in the MK 78033 data sheet; the key features are:

- □ 24 lines with 80 characters per line
- □ Cursor movements, absolute and relative
- □ Serial communication, 110–9600 baud
- □ Upper and lower case characters
- □ Clear screen, clear line etc.
- □ Tabulate
- □ A 9” diagonal display is used.

MECHANICAL

The display and keyboard are separate units connected by a cable. The display dimensions are:

B 43 cm   H 26 cm   T 32 cm

The keyboard dimensions are:

B 43 cm   H 4,5 cm   T 24 cm
H 9,0 cm

The Assembler (ASMB-80) includes:

- □ 1, 2 or 3 pass operation
- □ conditional Assembly
- □ Relocatable object module generation
- □ Relocatable linking loader
- □ Drivers for Silent 700 Cassette

The Text Editor (EDIT-80) includes:

- □ Line or character operation
- □ Macro commands

ELECTRICAL SPECIFICATIONS

Operating Temperature Range ...0 °C to 50 °C

Power Supply requirements (Typical)

+12V ± 5%	175 mA
+ 5V ± 5%	1.5 A
−12V ± 5%	100 mA

Interface Levels ... TTL Compatible

MECHANICAL SPECIFICATIONS

Extended double Eurocard

Board Size: 250 mm x 233.4 mm x 18 mm

Connector: Dual 64 pin Eurocard Connector
DIN 41612 form D; A and C pinned.

COMPATIBLE ADD-ON BOARDS

RAM-80AE Add-on RAM card for the SDB-80E. This card supplies 16k bytes of MK 4027 dynamic RAM Memory.

RAM-80BE Add-on RAM/IO card for the SDB-80E. This card supplies 16k (expandable to 64k) bytes of MK 4116 dynamic RAM Memory, plus 4 fully buffered I/O ports using 2 MK 3881 PIO's. On-card bank switching allows expansion of SDB-80E memory space beyond 64k bytes.

XRAM-80 Expansion Kit for RAM-80BE. Consists of 8-MK 4116 RAMs.

AIM-80E In-circuit-emulation capability is added to the SDB-80 by using the AIM-80 board also provides other debugging capabilities such as TRACE and SINGLE STEP.

FLP-80E The FLP-80 interfaces the SDB-80 to two soft-sectored floppy disk drives. Full file handling software and firmware is provided with the card.

RIO-80E The RIO-80E includes 2-buffered PIO's, I-UART, I-CTC, and sockets for 16k bytes of MK 2708 PROM.

NON RESIDENT SOFTWARE AVAILABLE

XFOR-80 Fortran IV Cross Assembler. Assembles Z80 programs but is written in Fortran IV. It is useful for persons desiring to perform Z80 assembly in mini-computers such as the PDP-11. It is furnished in Fortran IV source as a card deck or paper tape.

XMDS-80 8080A Cross Assembler. Performs the same function as the Fortran IV Cross Assembler, except that it is designed to be used with an Intel MDS system. It is furnished as an object tape in Intel compatible Hex format.

XMDS-80D This is identical to the XMDS-80 except that it is compatible with Intel MDS systems which use floppy disk. It is furnished as object code on an MDS compatible floppy diskette.

OTHER ACCESSORIES AVAILABLE

PPG-08 PROM Programmer module for programming MK 2708 UV erasable PROM memories. Interfaces directly with the SDB-80. Enclosure included.

OEM USERS CARDS AVAILABLE

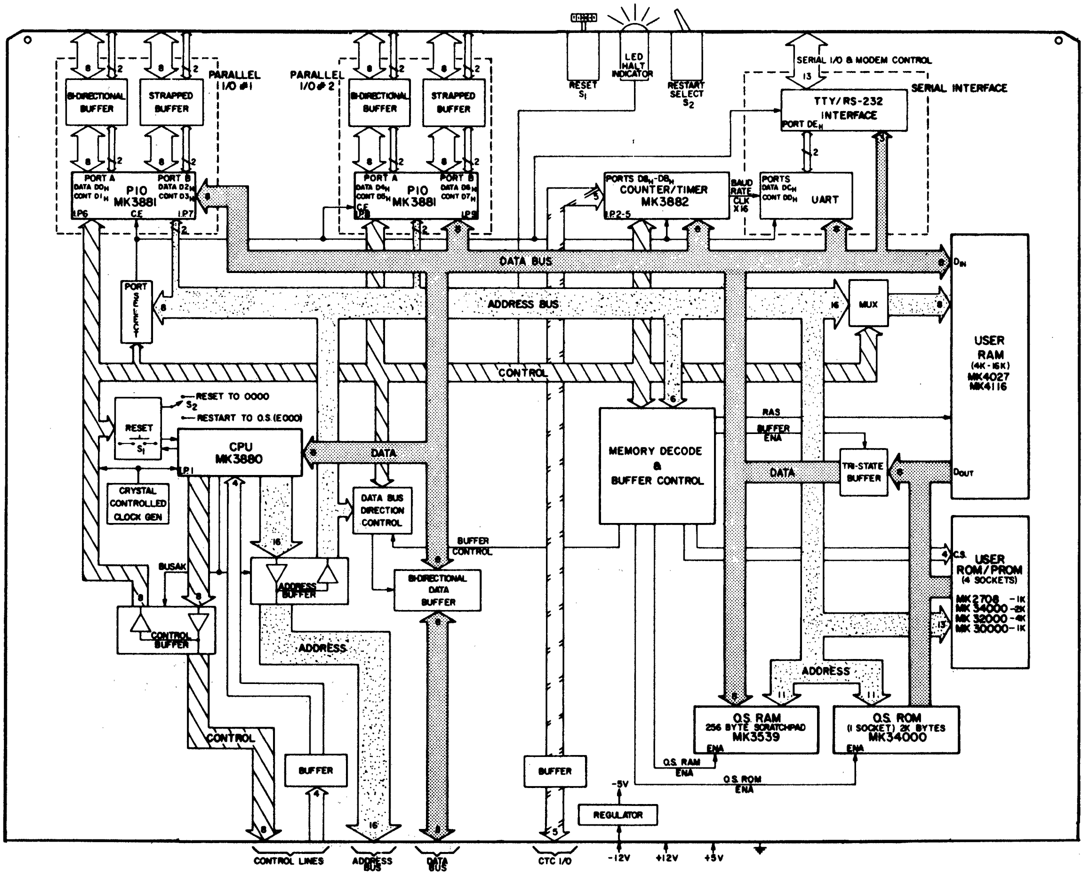
OEM-80E SDB-80E without Software. Available with 4 or 16k RAM Memory. 5 PROM/ROM sockets are free for user programs.

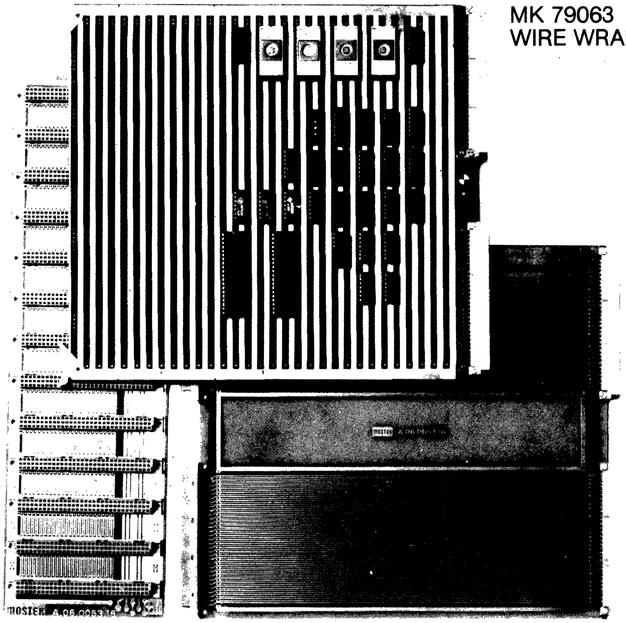
Parallel Universal Display Interface UDI-P This double Eurocard CRT/Keyboard Interface is bus compatible with the SDB-80E. A MK 3881 PIO on the card allows writing to the CRT Display at up to 3.300 characters per second.The CRT-80E provides 24 lines of 80 characters. The standard ASCII 96 character font is provided, other fonts may be programmed using MK 2708 PROM's. The command set includes TAB and cursor control.

Serial Universal Dislplay Interface UDI-S This is identical to the parallel except it operates over a 4 wire serial current loop connection. Useful in remote terminal applications.

BACK-80E 12 slot prewired printed circuit backplane for the SDB-80E family. This card greatly simplifies system construction.

SDB-80 FUNCTIONAL BLOCK DIAGRAM





MK 79063
WIRE WRAP CARD

EXTENDER CARD
MK 79062

BACKPLANE CARD
MK 79054

DEVELOPMENT STATION Z80 MK 78039

FEATURES

- □ Accepts upto 6 total boards
- □ Protected power supplies
- □ 11 I/O connectors for Peripheral equipment
- □ Double Europaformat boards

GENERAL

The MOSTEK development station has been designed to house and provide power for all MOSTEK boards with the double european format as in the detailed description of the SDB-80E. When used in conjunction with the MOSTEK terminal it forms a powerful development system.

INPUT/OUTPUT

11 connectors, 25 pin type D, are available for peripheral equipment. For the SDB 80E some of these are already commited as listed here below:

Connector	**SDB 80 E function**
1	Terminal
2	C T C
3	Floppy disc controller (1)
4	Uncommitted.
5	Uncommitted.
6	Paper tape reader
7	Paper tape punch
8	Line printer (2)
9	Uncommitted.
10	PROM programmer-PPG08
11	Uncommitted.

(1) with FLP 80E
(3) with RAM 80BE (or RIO80E or use PROM prog. Connector)

POWER SUPPLIES

Plug-in power supplies are used, the supplies being one card with +5 volt 7 ampere and one card with ±12 volt 1 ampere. All supplies have overvoltage protection and current limiting.

MECHANICAL

The housing has the following dimensions:

B 52 cm H 18 cm T 36 cm

The front panel has quick release fastners to give free access to the boards.

MOSTEK Z80 DEVELOPMENT SYSTEM

Includes:

MK 78103	SDB-80 Package A or	(4k byte RAM)
MK 78104	SDB-80 Package B with 256 byte static RAM DDT 80 Operating System ASMB 80 Resident Assembler and Text Editor and documentation	(16k byte RAM)
MK 78037	MOSTEK Terminal	
MK 78039	Development Station Z80	

MOSTEK®

Z80 MICROCOMPUTER SOFTWARE SUPPORT

Operating System (DDT-80)

FEATURES

- □ Program debug capability
- □ Channeled I/O for user convenience
- □ A set of I/O peripheral drivers is supplied
- □ Interactive hexadecimal addition and subtraction is in force when entering commands
- □ User expandable operating system

DESCRIPTION

DDT-80 is the Operating System for the Z80 Software Development Board (SDB-80). It resides in a 2K ROM (MK34000 series) resident on the SDB-80. It provides the necessary tools and techniques to operate the system, i.e., to efficiently and conveniently perform the tasks necessary to develop microcomputer software. DDT-80 is designed to support the user from initial design through production testing. It allows the user to display and update memory, registers, and ports, load and dump object files, set breakpoints, copy blocks of memory, and execute programs.

DDT-80 COMMAND SUMMARY

M s — Display and/or update the content of memory location s.

M s,f — Tabulate the contents to memory locations s through f.

P s — Display and/or update the content of I/O port s.

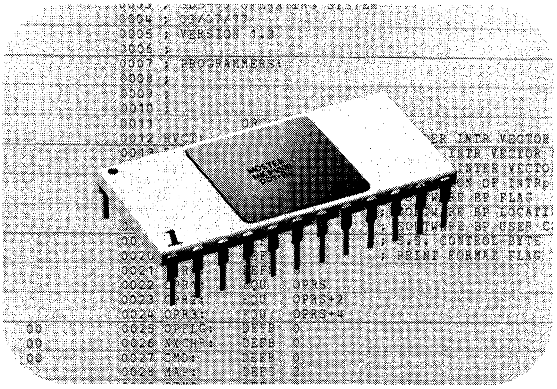
D s,f — Dump the contents of memory locations s through f in a format suitable to be read by the L command.

L — Load, into memory, data which is in the appropriate format.

E s — Transfer control from DDT-80 to a user's program starting at location s.

H — Perform 16 bit hexadecimal addition and/or subtraction.

C s,f,d — Copy the contents of memory locations s through f to another location in memory starting at location d.



B s — Insert a breakpoint in the user's program (must be in RAM) at location s which transfers control back to DDT-80. This allows the user to intercept his program at a specific point (location s) and examine memory and CPU registers to determine if his program is working correctly.

R — Display the contents of the user registers.

The s, f, and d represent start, finish, and destinations operands required for each command.

MEMORY, PORT AND REGISTER COMMANDS (M, P, R)

The M, P, and R commands provide the means for displaying the contents of specified memory locations, port addresses, or CPU registers. The M and P commands sequentially access memory locations or ports and display their contents. The user has the option of updating the content of the memory location or port. (Note some ports are output only and their contents cannot be displayed). The M command also gives the user access to the CPU registers through an area in RAM called the Register Map (discussed in the Execute, Breakpoint section below).

The M and R commands are used to tabulate blocks of memory locations (M) or the CPU registers (R). The M command will accept two operands, the starting and ending address of the memory block to be tabulated. The R command will accept either no operand or one. If no operand is specified, the CPU registers will be displayed without a heading. If an operand is specified then a heading which labels the register contents will be displayed as well.

EXECUTE AND BREAKPOINT (E, B)

The E command is used to execute all programs, including aids such as the Assembler. The B command is used to set a breakpoint to exit from a program at some predetermined location for debugging purposes. At the instant of a breakpoint exit, the contents of all CPU registers are saved in a designated area of SDB-80 RAM called the Register Map. In the Register Map, the register contents may be examined or modified using the M command and a predefined mnemonic (or absolute address) of the storage location for that register (example :PC, :A, . . . , :SP). The Register Map is also used to initialize the CPU registers whenever execution is initiated or resumed. Thus the E and B commands can be used together to initialize, execute, and examine the results of individual program segments.

The B command gives the user the option of having all CPU registers displayed when the breakpoint is encountered. This is done by entering a second operand to the B command. Otherwise DDT-80 defaults to displaying the PC and AF registers. When all CPU registers are displayed, the format is the same as for the R command previously discussed.

LOAD, DUMP, AND COPY (L, D, C)

The L and D commands load and dump object files through the object I/O channel in standard Intel Hex format. Checksums are used for error detection, and the addresses of questionable blocks are typed automatically while loading.

The C command will copy the contents of the memory block specified to another block of memory. There are no restrictions on the direction of the copy or on whether the blocks overlap.

HEXADECIMAL ARITHMETIC (H)

The H command is a dummy command used to allow hexadecimal addition and subtraction for expression evaluation without performing any other operation.

DDT-80 I/O CAPABILITIES

DDT-80 specifies three I/O channels, designated 'Console', 'Object', and 'Source', to which any suitable devices may be assigned. The Channel Assignment Table is located in RAM where it may be examined or modified using the M command. The table addresses correspond to the I/O channels and the table contents correspond to the addresses of the peripheral driver routines. A channel which has a device assignment may have that device assignment changed using the M command. This is accomplished by merely modifying the table contents of that channel's table address to correspond to the address of the new peripheral driver routine. A set of peripheral driver routines is supplied and listed below. This scheme also allows the user to write a driver routine for his own peripheral, load it into memory, and easily configure that peripheral into the system.

DDT-80 I/O PERIPHERAL DRIVERS

1. A serial input driver (usually a keyboard).
2. A serial output driver (usually a CRT or teletype typehead).
3. A serial input driver which sends out a reader step signal (usually a teletype reader).
4. A serial output driver which forces a delay after a carriage return (usually a Silent 700 typehead).
5. A parallel input driver (usually for high speed paper tape input).
6. A parallel output driver (usually for high speed paper tape output).
7. A parallel output driver (usually for a line printer).

DDT-80 USER EXPANDABILITY

In its operation, DDT-80 will perform a jump indirect to itself using the contents of 2 designated RAM locations as the address jumped to. Usually this jump will be to a location in DDT-80 and on power-up and reset the 2 RAM locations are loaded with the correct address in DDT-80 for the jump. However, using the M command, the 2 RAM locations may be modified to correspond to a different address. DDT-80 will collect the command (single letter) and save it and will also scan for operands (up to 3), evaluating expressions to 4 hex digits. It is at this point that DDT-80 will perform the indirect jump to the address specified in the 2 RAM locations. Therefore, the user can supply an additional set of commands to enhance the operating system if desired.

ORDERING INFORMATION

NAME	DESCRIPTION	PART NO.
DDT-80 Operations Manual	Detailed description of the use and operation of DDT-80	MK 78522
DDT-80 Source Listing	Complete assembly language source listing with comments	MK 78534
DDT-80 Firmware Package	Includes the operation manual and source listing plus the MK34000 series ROM containing DDT-80 firmware	MK 78118

The DDT-80 source listing and firmware packages are available directly from MOSTEK by filling out a copy of the Software Licensing Agreement printed on the opposite page of this data sheet and returning it with the appropriate payment or Customer Purchase Order to:

MOSTEK CORPORATION
Microcomputer Systems Div.
1215 West Crosby Road
Carrollton, Texas 75006

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold
on condition that the Purchaser agrees to
the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.

4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject To this Agreement:

Order Number Description Price*

____

____

____

Ship To: ______________________ Bill To: ______________________

______________________________ ______________________________

______________________________ ______________________________

Method of Shipment: ______________ Customer.P.O. Number ______________

Agreed To:

PURCHASER MOSTEK CORPORATION

By: ______________________ By: ______________________

Title: ______________________ Title: ______________________

Date: ______________________ Date: ______________________

MOSTEK®

Z80 MICROCOMPUTER SOFTWARE SUPPORT

Assembler/Editor/Loader (ASMB-80)

The MOSTEK ASMB-80 is a software package which consists of a Text Editor, Z80 Assembler, and Relocating Linking Loader. The software is supplied in ROM for the MOSTEK SDB-80 (Z80 Software Development Board). The programs make extensive use of the Z80 Designer's Development Tool (DDT-80) which is also supplied in ROM for the SDB 80 to provide the user with state-of-the-art software for developing Z80 programs. All I/O is done via the SDB-80 channels which can be directed to any software driver. The DDT-80 contains drivers for paper tape, Silent 700, TTY, CRT, and line printer devices. The ASMB-80 contains drivers for Silent 700 digital cassette and for RAM-based operation. RAM-based operation allows editing, assembling and loading of programs using RAM instead of external media for intermediate storage.

FEATURES—ASMB-80 TEXT EDITOR

□ Allows input and modification of ASCII text on the MOSTEK SDB-80

□ Allows line and character editing

□ Has two alternate command buffers for pseudo-macro command capability

□ Allows the following commands:

An - Advance record pointer n records

Bn - Backup record pointer n records

Cn dSldS2d - change string S1 to string S2 for n occurrences

Dn - delete next n records

E - exchange current record with records to be inserted

I - insert records

Ln - Go to line number n

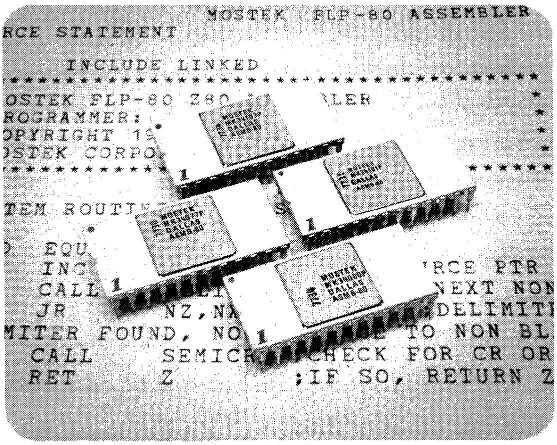
Mn - Enter commands into one of two alternate command buffers (pseudo-macro)

N - Print top, bottom, and current line number

Pn - Punch n records from buffer

R - Read source records into buffer

Sn dSld - Search for nth occurrence of string S1



Vn - Output n records to console output channel

Wn - Output n records to source output channel

Xn - Execute alternate command buffer n

DESCRIPTION

The Text Editor permits random access editing of ASCII character strings. It can be used as a line or character oriented editor. Individual characters may be located by position or context. The Editor works on blocks of characters which are typically read into memory from magnetic tape or paper tape. Each edited block can be output to magnetic tape or paper tape after editing is completed. While the primary application for the Text Editor is in editing assembly language source statements, it may be applied to any ASCII text delimited by "carriage returns".

The Editor has a macro command processing option. Up to two sets of commands may be stored and processed at any time during the editing process.

All I/O is done via the SDB-80 channels. The Editor can be used with the MOSTEK ASMB-80 Assembler and Loader to edit, assemble, and load programs in memory without the need for external media for intermediate storage.

FEATURES – ASMB-80 ASSEMBLER

□ Assembles all standard Z80 source statements

□ Object output in industry standard hexadecimal format extended for relocatable and linkable programs

□ Allows the following pseudo-ops:

ORG	–	program origin
EQU	–	equate label
DEFL	–	define label
DEFM	–	define message
DEFB	–	define byte
DEFW	–	define word
DEFS	–	define storage
END	–	end statement
IF	–	conditional assembly
ENDIF	–	end of conditional assembly
NAME	–	program name definition
PSECT	–	program section definition
GLOBAL	–	global symbol definition

Supports the following assembler directive pseudo-ops:

EJECT	–	eject a page of listing
TITLE	–	place heading at top of each page of listing
LIST	–	turn listing on
NLIST	–	turn listing off

□ Complete assembly in two passes with second pass repeatable for outputting object tape and assembly listing on a single peripheral (such as TTY)

□ Size of program to be assembled limited only by memory available for a symbol table, symbol table size selectable by user

□ Channeled I/O adaptable to any device and to other systems

□ Applicable as a single pass assembler or as a learning tool if no forward symbol references exist in the program

DESCRIPTION

The Assembler reads Z80 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The assembly listing shows address, machine code, statement number, and source statement. The object code is in industry standard hexadecimal format modified for relocatable, linkable assemblies.

The Assembler supports conditional assemblies, global symbols, relocatable programs, and a printed symbol table. It can assemble any length program, limited only by a symbol table size which is user selectable. Expressions involving addition and subtraction are allowed. Conditional assembly allows the user to suspend assembly for a portion of the program depending upon the result of an expression. A global symbol is catagorized as "internal" if it appears as a label in the program; otherwise it is an "external" symbol. The printed symbol table shows which symbols are internal and which are external. The Assembler allows the user to select relocatable or non-relocatable assembly via the "PSECT" pseudo-op. Relocation records are placed in the object output for relocatable assemblies (The MOSTEK object format is defined below). The Assembler can be run as a single pass assembler or as a learning tool. (In this mode, global symbols and forward references are not allowed).

All I/O is done via the SDB-80 channels. Assemblies can be done from source statements stored in memory (by the Editor). The object output can be directed to a memory buffer rather than to an external device. Thus, assembly and loading can be done without external storage media.

FEATURES – ASMB-80 RELOCATING LINKING LOADER

□ Loads into memory both relocatable and non-relocatable object output of the ASMB-80 Assembler

□ Allows loading of relocatable modules anywhere in memory

□ Automatically provides linkage of global symbols between object modules as they are loaded

□ Prints beginning and ending addresses of each module loaded and provides printing of global symbol addresses to aid in program debugging

□ Allows the following commands:

L offset - load object module at address "offset" plus program origin address

E - execute loaded program at transfer address of first module

T - print global symbol table

□ Also loads industry standard non-relocatable, non-linkable object output

DESCRIPTION

The MOSTEK ASMB-80 Relocating Linking Loader firmware provides state-of-the-art capability for loading programs into memory by allowing loading and linking of any number of relocatable and non-relocatable object modules. Non-relocatable modules are always loaded at their starting address as defined by the ORG pseudo-op during assembly. Relocatable object modules can be positioned anywhere in memory at an offset address.

The MOSTEK ASMB-80 Loader automatically links and relocates global symbols which are used to provide communication or linkage between program modules. As object programs are loaded, a table containing global symbol references and definitions is built up. At the end of each module, the loader resolves all references to global symbols which are defined by the current or a previously loaded module. It also prints on the console device the number of defined global symbols that have been referenced. The symbol table can be printed to list all global symbols and their load addresses. The number of object modules which can be loaded by the Loader is limited only by the amount of RAM available for the modules and the symbol table. Space for the symbol table is allocated dynamically downward in memory from either the top of memory or from a specified address entered as an operand of the load command.

The Loader prints the beginning and ending address of each module as it is loaded. The transfer address as defined by the END pseudo-op is printed for the first module loaded. The Loader execute command (E) can be used to automatically start execution at the transfer address.

MOSTEK OBJECT OUTPUT DEFINITION

Each record of an object module begins with a delimiter (colon or dollar sign) and ends with carriage return and line feed. A colon (:) is used for data records and end-of-file record. A dollar sign ($) is used for records containing relocation information and linking information. All information is in ASCII. Each record is identified by "type". The type is determined by the 8th and 9th bytes of the record which can take the following values:

00	-	data
01	-	end-of-file
02	-	internal symbol
03	-	external symbol
04	-	relocation information
05	-	module definition

DELIMITER			RECORD TYPE			
1	2 3	4 5 6 7	8	9	10	
:	# of BINARY DATA BYTES ④	START ADDRESS OF DATA	0	0	. . .DATA. . .	CHECK SUM ①
:	0 0	TRANSFER ADDRESS OF MODULE	0	1	CHECK SUM ①	
$	INTERNAL SYMBOL NAME		0	2	ADDRESS	CHECK SUM ①
$	EXTERNAL SYMBOL NAME		0	3	LINK ADDRESS ②	CHECK SUM ①
$	# of BINARY BYTES ④	0 0 0 0	0	4	ADDRESSES WHICH . . . REQUIRE RELOCATION . . .	CHECK SUM ①
$	MODULE NAME		0	5	FLAGS ③	CHECK SUM ①

NOTES:

1. Check Sum is negative of the binary sum of all bytes except delimiter and carriage return/line feed.

2. Link Address points to last address in the data which uses the external symbol. This starts a backward link list through the data records for that external symbol. The list terminates with OFFFFH.

3. The flags are one binary byte. Bit 0 is defined as:

   0 - absolute module
   1 - relocatable module

4. Maximum of 64 ASCII bytes.

ORDERING INFORMATION

NAME	DESCRIPTION	PART NO.
Operations Manual	Describes in detail the operation and use of the ASMB-80 firmware package. The operation and use of DDT-80 firmware is also covered (130 pages).	MK78522
Program Source Listing	Complete assembly language source listing of the ASMB-80 firmware package, including all comments and other notations. (190 pages).	MK78536*
Firmware Package	Four MK34000 (2Kx8) ROM's which include the ASMB-80 firmware package. This includes the Operations Manual and Program Source Listing.	MK78119*

*The ASMB-80 source listing and firmware packages are available directly from MOSTEK by filling out a copy of the Software Licensing Agreement printed on the opposite page of this data sheet and returning it with the appropriate payment or Customer Purchase Order to:

MOSTEK CORPORATION
Microcomputer Systems Div.
1215 West Crosby Road
Carrollton, Texas 75006

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold
on condition that the Purchaser agrees to
the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.

4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject To This Agreement:

Order Number Description Price*

____

____

Ship To:____________________ Bill To:____________________

____

Method of Shipment:____________________ Customer P.O. Number____________________

Agreed To:

PURCHASER

By:____________________

Title:____________________

Date:____________________

MOSTEK CORPORATION

By:____________________

Title:____________________

Date:____________________

*Prices Subject To Change Without Notice

MOSTEK®

MICROPROCESSOR HARDWARE SUPPORT

Video Display Interface

FEATURES

□ Complete video/keyboard interface

□ 24 line x 80 character display

□ Inverse video by character programmable

□ 5 x 8 dot matrix

□ Serial UART port : 110-9600 baud

□ Parallel port : up to 3200 characters/second

□ Upper and lower case display : 96 character ASCII +32 special characters

□ Auto repeat on keyboard interface

□ Direct cursor addressing and bidirectional scrolling

□ 48 character FIFO

□ 50/60 Hz operation

DESCRIPTION

The VDI card is a self-contained interface for interconnection of a video monitor and an ASCII keyboard to a computer over a serial link. Both 20 mA. current loop and RS232 (V24) compatible voltage loop interconnection techniques are supported.

The card can be used as a teletype replacement or as an intelligent terminal in many applications. The interface is constructed on a standard width double Eurocard. The power connections (and bus) are directly compatible to the MOSTEK standard SD series bus.

The present cursor position is indicated by a blinking white rectangle which inverts any data that may be covered by the cursor. Besides the normal up/down/left/right cursor motions, the cursor may be directly addressed and positioned anywhere on the screen. The terminal may also be used in local mode; screen data can be read back by a computer.

The card supports up to 80 user-defined tab locations. The 48 character FIFO allows continuous high speed data transmission even when time-consuming commands are given; for example, clear screen.

SPECIFICATIONS

Operating temperature 0 to 50°C

Power supply requirements
+ 5V @ 2.0 A
+12V @ 0.1 A
–5V (for use in optional RS232 or 2708 PROM)

Board format	: 233.4 x 250 mm (9.19 x 9.84 inches) : DIN 41612 connectors : extended double Eurocard
Video output	: RS170 1V into 75 Ohm : 525 line 60Hz or 625 line 50Hz
Keyboard inputs	: standard TTL, ASCII encoded : active high or low strobe 300 microseconds minimum pulse. : active high data
Serial link	: 110,300,600,1200,2400, 4800,9600 baud generator on board : external baud rate may be used (max 19200 baud) : programmable 1/2 stops bits, even/odd/no parity : completely opto isolated current loop (20 mA.)
Parallel interface	: SD bus compatable with on board Z80 PIO : 3200 char per second transfer rate : 2 interrupts : 64 possible port addresses
Character generator	: MK34073 2K x 8 ROM : customer defined 2708/58 PROM can also be used
Processing time	: typical (character data) 320 microseconds. : worst case (clear display) 9.33 ms.
Bell output	: direct drive to 50 Ohm speaker

POWER ON CONDITION

At power on, the following conditions are set: home, clear screen, line, all tabs clear and normal video.

COMMAND SET

Shift means the shift key must also be depressed on most keyboards. Esc means that the command must be preceded by the escape character

HEX	CONTROL	FUNCTION	COMMENT
07	G	BELL	300 ms. 700Hz tone
08	H	BACKSPACE	Cursor moves left once unless at left margin
09	I	TAB	Cursor moves to next tab location
0A	J	LINE FEED	Cursor moves down one line. If at bottom then display scrolls up one line
0B	K	VT	Cursor moves up one line, if at top, then display scrolls down one line
0C	L	FF	Cursor moves right once unless at right margin
0D	M	RETURN	Cursor moves to left margin.
0E	N	RESET TAB	Tab is cleared from this position
0F	O	SET TAB	Tab is set at this position
10	P	DOWNSHIFT	Following character is displayed as special non ASCII character
1B	SHIFT-K	ESCAPE	Enables escape sequence, following character is the command.
3D	ESC =	MOVE CURSOR	Command to move cursor. Next two characters are binary y and x addresses of new cursor position. Lower left is 0 : 0
2B	ESC +	MOVE CURSOR	Command to move cursor. Next two characters are binary y and x two's complement offsets from current position.
3F	ESC ?	READ CURSOR	Read back cursor address. VDI sends STX then y,x address of cursor. 30 hex is added to avoid control character conflicts.
7D	ESC †	LOCAL	Sets VDI to local mode. Keyboard data is sent directly to the display without being output to the serial/parallel port.
29	ESC)	READ ONE	Read back to the port the contents of the cursor postion. Cursor does FF.
3C	ESC <	READ LINE	Line from cursor to last space is sent to the port. Data is preceded by a STX and followed by a RETURN and ETX. Cursor does not move.
3E	ESC >	READ PAGE	Entire page from cursor to end is sent to the port. Data is preceded by a STX, trailing spaces are deleted, and lines are separated by returns. At the end an ETX is sent.
17	W	INVERT	All following characters will be written in inverted video. (black on white)
19	Y	NORMAL	All following characters will be written in normal video. (white on black)
1C	®	LINE	Set VDI to line mode, send ACK to port. All following keyboard inputs go directly to the port.
1D	]	CLEAR SCREEN	Clear display from cursor to end
1E	©	HOME	Move cursor to upper left corner
1F	—	CLEAR LINE	Clear line from cursor to end
7F		DEL	Nop

OPTIONS

All options including port address, baud rate, parity, stop bits, 50/60 Hz and keyboard strobe active edge are programmable on connector #2.

SOFTWARE SUPPORT

A screen editor for the Z80 which uses the VDI card as a "window" into a Z80 system memory is also available from MOSTEK.

ORDERING INFORMATION

NAME	DESCRIPTION	PART NUMBER
VDI-P	Parallel port version board with complete manual	MK78035
VDI-S	Serial port version board with complete manual only	MK78033
VDI		MK78586

MOSTEK®

1215 W. Crosby Rd. • Carrollton, Texas 75006 • 214/242-0444
In Europe: Contact, MOSTEK Brussels
150 Chaussee de la Hulpe, B1170, Belgium;
Telephone: (32) 02/660.69.24

June 1979
PRINTED IN USA
MK79707