

DMC-11

DMC-11 CROM & JUMP TEST
CZDMGDO

AH-8562D-MC

MAR 1978

COPYRIGHT © 76-78

digital

FICHE 1 OF 2

MADE IN USA

The microfiche card displays a grid of 24 frames, arranged in 4 rows and 6 columns. Each frame contains a small table with multiple columns and rows of text, likely representing test results or program data for the DMC-11 system. The text is too small to read clearly but appears to be organized into structured data tables.

DMC-11

DMC-11 CROM & JUMP TEST
CZDMGDO

AH-8562D-MC

MAR 1978

COPYRIGHT © 76-78

digital

FICHE 2 OF 2

MADE IN USA

[Faint, illegible text visible through the dark background, likely bleed-through from the reverse side of the page.]

EOF1CVKAFCSAQ
PDF10 PAGE: 0001

00010000

780223

PDP10 411

H0R1CZDMGDSEQ

00010000

780223

SEQ 0001

IDENTIFICATION

PRODUCT CODE: AC-85600-MC
PRODUCT NAME: CZDMGDD DMC-11 CROM & JUMP TST
DATE: FEB 1978
MAINTAINER: DIAGNOSTICS
AUTHOR: FAY BASHAW

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

Copyright (C) 1976, 1978 by Digital Equipment Corporation

1. ABSTRACT

The function of the DMC11 diagnostics is to verify that the option operates according to specifications. The diagnostics verify that there are no malfunctions and that all operations of the DMC11 are correct in its environment.

Parameters must be set up to alert the diagnostics to the DMC11 configuration. These parameters are contained in the STATUS TABLE and are generated in two ways: 1) Manual Input - the operator answers questions. 2) Autosizing - the program determines the parameters automatically.

CZDMGD tests the DMC11-AR and DMC11-AL micro-processors (MB200-YA MB200-YB). It performs jump tests on the micro-processor and verifies the control ROM of the MB200. This diagnostic will not run on a KMC (MB204), however it is possible to load the KMC CRAM with the DMC micro-code. See test 2 for details.

Currently there are five off line diagnostics that are to be run in sequence to insure that if an error should occur it will be detected at an early stage.

NOTE: Additional diagnostics may be added in the future.

The five diagnostics are:

1. DZDMC [REV] Basic W/R and Micro-processor tests
2. DZDME [REV] DDCMP Line unit tests
3. DZDMF [REV] BITSTUFF Line unit tests
4. CZDMGD [REV] Jump and CROM tests
5. DZDMH [REV] Free-running tests (Heat test tape)

2. REQUIREMENTS

2.1 EQUIPMENT

Any PDP11 family CPU (except an LSI-11) with minimum 8k memory
ASR 33 (or equivalent)
DMC11-AR (MB200-YA) or an DMC11-AL (MB200-YB)

2.2 STORAGE

Program will use all BK of memory except where AEL and BOOTSTRAP LOADER reside. Locations 1500 thru 1640; contain the "STATUS TABLE" information which is generated at start of diagnostics by manual input (questions) or automatically (auto-sizing). This area is an overlay area and should not be altered by the operator.

3. LOADING PROCEEDURE

3.1 METHOD

All programs are in absolute format and are loaded using the ABSOLUTE LOADER. NOTE: if the diagnostics are on a media such as DISK, MAGTAPE, DECTAPE, or CASSETTE; follow instructions for the monitor which has been provided on that specific media.

ABSOLUTE LOADER starting address *500

MEMORY * SIZE

4k	17
8k	37
12k	57
16k	77
20k	117
24k	137
28k	157

- 3.1.1 Place address of ABS loader into switch register.
(also place 'HALT' SW up)
- 3.1.2 Depress 'LOAD ADDRESS' key on console and release.
- 3.1.3 Depress 'START KEY' on console and release (program should now be loading into CPU)

4. STARTING PROCEDURE

- a. Set switch register to 000200
- b. Depress 'LOAD ADDRESS' key and release
- c. Set SWR to zero for 'AUTO SIZING' or SWR bit0=1 for manual input (questions) or SWR bit7=1 to use existing parameters set up by a previous start or a previously run DMC11 diagnostic.
- d. Depress 'START KEY' and release. The program will type Maindec Name and program name (if this was the first start up of the program) and also the following:

MAP OF DMC11 STATUS

PC	CSR	STAT1	STAT2	STAT3
001500	160010	145310	177777	000000
001510	160020	145320	177777	000000

The program will type 'R' and proceed to run the diagnostic. The above is only an example. This would indicate the status table starting at add. 1500 in the program. In this example the table contains the information and status of two DMC11'S. THE STATUS TABLE MUST BE VERIFIED BY THE USER IF AUTO SIZING IS DONE. For information of status table see section 8.4 for help.

If the diagnostic was started with SW00=1 indicating manual parameter input then the following shows an example of the questions asked and some example answers:

HOW MANY DMC11'S TO BE TESTED?1

D1
 CSR ADDRESS?160010
 VECTOR ADDRESS?310
 BR PRIORITY LEVEL? (4,5,6,7)?5
 DOES MICRO-PROCESSOR HAVE CRAM? (Y OR N)N
 WHICH LINE UNIT? IF NONE TYPE "N", IF MB201 TYPE "1", IF MB202 TYPE "2"?1
 IS THE LOOP BACK CONNECTOR ON?Y
 SWITCH PAC#1 (DDCMP LINE#)?377
 SWITCH PAC#2 (BM873 BOOT ADD)?377

Following the questions the status map is printed out as described above, the information in the map reflects the answers to the questions. If the diagnostic was started with SW00=0 and SW07=0 (AUTO-SIZING) then no questions are asked and only the status-map is printed out. If AUTO-SIZING is used the status information must be verified to be correct (match the hardware). If it does not match the hardware the diagnostic must be restarted with SW00=1 and the questions answered.

4.1 CONTROL SWITCH SETTINGS

SW 15 Set: Halt on error
SW 14 Set: Loop on current test
SW 13 Set: Inhibit error print out
SW 12 Set: Inhibit type out/abell on error.
SW 11 Set: Inhibit iterations. (quick pass)
SW 10 Set: Escape to next test on error
SW 09 Set: Loop with current data
SW 08 Set: Catch error and loop on it
SW 07 Set: Use previous status table.
SW 06 Set: Halt in ROMCLK routine before clocking
micro-processor
SW 05 Set: Reserved
SW 04 Set: Reserved
SW 03 Set: Reselect DMC11's desired active
SW 02 Set: Lock on selected test
SW 01 Set: Restart program at selected test
SW 00 Set: Build new status table from questions. (If SW07=0
and SW00=0 a new status table is built by
auto-sizing)

Switch 06 and 08-15 are dynamic and can be changed as needed while the diagnostic is running. Switches 00-03 and switch 07 are static, and are used only on starting or restarting the diagnostic.

4.1.2 SWITCH REGISTER OPTIONS (at start up)

SW 01 RESTART PROGRAM AT SELECTED TEST. It is strongly suggested that at least one pass has been made before trying to select a test, the reason being is that the program has to clear areas and set up parameters. When this switch is used the diagnostic will ask TEST NO.? Answer by typing the number of the test desired and carriage return to begin execution at the selected test.

SW 02 LOCK ON SELECTED TEST. This switch when used with SW01 will cause the program to constantly loop on the selected test. Hitting any key on the console will let it advance to the next test and loop until a key is hit again. If SW02=0 when SW01 is used. The program will begin at the selected test and continue normal operations.

SW 03 RESELECT DMC11'S DESIRED ACTIVE. Please note that a message is typed out for setting the switch register equal to DMC11's active. this means if the system has four DMC11s; bits 00,01,02,03 will be set in loc 'DMACTV' from the switch register. Using this switch(SW00) alters that location; therefore if four DMC11s are in the system ***DO NOT*** set switches greater than SW 03 in the up position. this would be a fatal error. do not select more active DMC11s than there is information on in the status table.

METHOD: A: Load address 200
 B: Start with SW 00=1
 C: Program will type message
 D: Set a switch for each DMC desired active.
 EXAMPLE: If you have 4 DMC's but only want to run the first and the last set SWR bits 0 and 3 = 1. PRESS CONTINUE
 E: Number (IF VALID) will be in data lights (excluding 11/05)
 F: Set with any other switch settings desired. PRESS CONTINUE.

4.1.3 DYNAMIC SWITCHES

ERROR SWITCHES

1. SW 12 Delete print out/bell on error.
2. SW 13 Delete error printout.
3. SW 15 Halt on the error.
4. SW 08 Goto beginning of the test(on error).
5. SW 10 Goto next test(on error).

SCOPE SWITCHES

1. SW06 Halt in ROMCLK routine before clocking micro-processor instruction. This allows the operator to scope a micro-processor instruction in the static state before it is clocked. Hit continue to resume running.
2. SW09 (if enabled by 'SCOPI') on an error; If an '*' is printed in front of the test no. (ex. *TEST NO. 10) SW09 is incorporated in that test and therefore SW09 is usually the best switch for the scope loop (SW14=0, SW10=0, SW09=1, SW08=0). If SW09 is not enabled; and there is a HARD error (constant); SW08 is best. (SW14=1,0, SW10=0, SW09=0, SW08=1). for intermittent errors; SW14=1 will loop on test regardless of error or not error. (SW14=1, SW10=0, SW09=0, SW08=1,0)
3. SW11 Inhibit iterations.
4. SW14 Loop on current test.

4.2 STARTING ADDRESS

Starting address is at 000200 there are no other starting addresses for the DMC11 diagnostics. (See Section 4.0)

NOTE: If address 000042 is non-zero the program assumes it is under ACT11 or XXDP control and will act accordingly after all available DMC11's are tested the program will return to 'XXDP' or 'ACT-11'.

5. OPERATING PROCEDURE

When program is initially started messages as described in section 4.0 will be printed, and program will begin running the diagnostic

5.2 PROGRAM AND/OR OPERATOR ACTION

The typical approach should be

1. Halt on error (via SW 15=1) when ever an error occurs.
2. Clear SW 15.
3. Set SW 14: (loop on this test)
4. Set SW 13: (inhibit error print out)

The TEST NUMBER and PC will be typed out and possibly an error message (this depends on the test) to give the operator an idea as to the source of the problem. If it is necessary to know more information concerning the error report; LOOK IN THE LISTING for that TEST NUMBER which was typed out and then NOTE THE PC of the ERROR REPORT this way the EXACT FUNCTION of the test CAN BE DETERMINED.

6. ERRORS

As described previously there will always be a TEST NUMBER and PC typed out at the time of an error (providing SW 13=0 and SW 12=0). in most cases additional information will be supplied in the the error message to give the operator an indication of the error.

6.2 ERROR RECOVERY

If for some reason the DMC11 should 'HANG THE BUS' (gain control of bus so that console manual functions are inhibited) an init or power down/up is necessary for operator to regain control of cpu. If this should happen; look in location 'TSTNO' (address 1226) for the number of the test that was running at the time of the catastrophic error. In this way the operator will have an idea as to what the DMC11 was doing at the time of the error.

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

See section 4. (PLEASE)
Status table should be verified regardless of how program was started. Also it is important to use this listing along with the information printed on the TTY to completely isolate problems.

7.2 OPERATING RESTRICTIONS

The first time a DMC11 diagnostic is loaded into core and run the STATUS TABLE must be set up. This is done by manual input (SW00=1) or by autosizing (SW00=0 and SW07=0). Thereafter however the status table need not be setup by subsequent restarts or even loading the next DMC diagnostic because the STATUS TABLE is overlaid. The current parameters in the STATUS TABLE are used when SW07=1 on start up.

7.3 HARDWARE CONFIGURATION RESTRICTIONS

DMC11(MB200)- Jumper W1 must be in, and switch 7 of E76 must be in the OFF position.

KMC(MB204)- Jumper W1 must be in.

8. MISCELLANEOUS

8.1 EXECUTION TIME

All DMC11 device diagnostics will give an 'END PASS' message (providing no errors and sw12=0) within 4 mins. This is assuming SW11=1 (DELETE ITERATIONS) is set to give the fastest possible execution. The actual execution time depends greatly on the PDP11 CPU configuration and the amount of memory in the system.

8.2 PASS COMPLETE

NOTE: EVERY time the program is started; the tests will run as if SW11 (delete iterations) was up (=1). This is to 'VERIFY NO HARD ERRORS' as soon as possible. Therefore the first pass -EACH TIME PROGRAM IS STARTED- will be a 'QUICK PASS' until all DMC11's in system are tested. When the diagnostic has completed a pass the following is an example of the print out to be expected.

```
END PASS CZDMGD CSR: 175000 VEC: 0300 PASSES: 000001  
ERRORS: 000000
```

NOTE: The pass count and error counts are cumulative for each DMC11 that is running, and are set to zero only when the diagnostic is started. Therefore after an overnight run for example, the total passes and errors for each DMC11 since the diagnostic was started are reflected in PASSES: and ERRORS:.

8.4 KEY LOCATIONS

- RETURN (1214) Contains the address where program will return when iteration count is reached or if loop on test is asserted.
- NEXT (1216) Contains the address of the next test to be performed.
- TSTNO (1226) Contains the number of the test now being performed.
- RUN (1316) The bit in 'RUN' always points to the DMC11 currently being tested. EXAMPLE: (RUN) 1302/0000000001000000 Means that DMC11 no.06 is the DMC11 now running.

DMC000-DMC17
DMST00-DMST17
(1500)-(1640)

These locations contain the information needed to test up to 16 (decimal) DMC11s sequentially. they contain the CSR VECTOR and STATUS concerning the configuration of each DMC11.

DMACTV (1306) Each bit set in this location indicates that the associated DMC11 will be tested in turn. EXAMPLE: (DMACTV) 1276/0000000000011111 means that DMC11 no. 00,01,02,03,04 will be tested. EXAMPLE: (DMACTV) 1276/0000000000010001 Means that DMC11 no. 00,04 will be tested.

DMCSR (1404) Contains the CSR of the current DMC11 under test.

8.4A 'STATUS TABLE' (1500-1640)

The table is filled by AUTO SIZING or by the manual parameter input (questions) as described previously. Also if desired by user; the locations may be altered by hand (toggled in) to suit the specific configuration.

The example status map shown below contains information for two DMC11'S. the table can contain up to 16 DMC11'S. Following the map is a description of the bits for each map entry

MAP OF DMC11 STATUS

PC	CSR	STAT1	STAT2	STAT3
001500	160010	145310	177777	000000
001510	160020	016320	000000	000000

Each map entry contains 4 words which contain the status information for 1 DMC11. The PC shows where in core memory the first of the 4 words is. In the example above the first DMC'S status is in locations, 1500, 1502, 1504, and 1506. The second DMC status is located at 1510, 1512, 1514, and 1516. The information contained in each 4 word entry is defined as follows:

CSR: Contains DMC1' CSR address

STAT1: BITS 00-08 IS DMC11 VECTOR ADDRESS
BIT15=1 MICRO-PROCESSOR HAS CROM
BIT15=0 MICRO-PROCESSOR HAS CROM
BIT14=1 TURNAROUND CONNECTOR IS ON
BIT14=0 NO TURNAROUND CONNECTOR
BIT13=0 LINE UNIT IS AN M8201
BIT13=1 LINE UNIT IS AN M8202
BIT12=1 NO LINE UNIT
BITS 09-11 IS DMC11 BR PRIORITY LEVEL

STAT2: LOW BYTE IS SWITCH PAC#1 (DDCMP LINE NUMBER)
HIGH BYTE IS SWITCH PAC#2 (BM873 BOOT ADD)

STAT3: BIT0=1 PERFORM FREE RUNNING TESTS ON KMC
BIT1=0 DMC11-AR (LOW SPEED)
BIT1=1 DMC11-AL (HIGH SPEED)

8.5 METHOD OF AUTO SIZING

8.5.1 FINDING THE CONTROL STATUS REGISTER.

The auto-sizing routine finds a DMC11 as follows: It starts at address 160000 and tests all address in increments of 10 up to and including address 167760. If the address does not time out, the following is done, the first CROM address is written to a 125252 then it is read back. If it contains a -1 or 125252 or 626 or 16520 a DMC11 or KMC11 has been found, if not, the address is updated by 10 and the search continues. A -1 indicates a DMC11 with no CROM, a 125252 indicates a KMC11 with CROM, a 626 indicates a DMC11-AL, and a 16520 indicates a DMC11-AL. Further tests are performed at this point to determine which line unit, if any, is installed, if a loop-back connector is installed and various switch settings on the line unit. THIS IS WHY THE STATUS TABLE MUST BE VERIFIED BY THE USER AND IF ANY OF THE INFORMATION DOES NOT AGREE WITH THE HARDWARE THE DIAGNOSTIC MUST BE RESTARTED AND THE QUESTIONS MUST BE ANSWERED. All DMC11's in the system will be found by the auto-sizer. If it does not find a DMC11 the diagnostic must be restarted and the questions answered.

8.5.2 FINDING THE VECTOR AND BR LEVEL

The vector area (address 300-776) is filled with the instruction IOT and '+2' (next address). The processor status is started at 7 and the DMC is programmed to interrupt. The PS is lowered by 1 until the DMC interrupts, a delay is made and if no interrupt occurs at PS level 3 (because of a bad DMC11) the program assumes vector address 300 at BR level 5 and the problem should be fixed in the diagnostic. Once the problem is fixed; the program should be re-setup again to get correct vector. If an interrupt occurred; the address to which the DMC11 interrupted to is picked up and reported as the vector. NOTE: if the vector reported is not the vector set up by you; there is a problem and AUTO SIZING should not be done.

8.6 SOFTWARE SWITCH REGISTER

If the diagnostic is run on an 11/04 or other CPU without a switch register then a software switch register is used to allow user the same switch options as described previously. If the hardware switch register does not exist or if one does and it contains all ones (177777) this software switch register is used.

Control:

To obtain control at any allowable time during execution of the diagnostic the operator types a CTRL G on the console terminal keyboard. As soon as the CTRL G is recognized, by the diagnostic, the following message will be displayed:

SWR=XXXXXX NEW?

Where XXXXXX is the current contents of the software switch register in octal. The software control routine will then await operator action. At which time the operator is required to type one or more of the legal characters: 1) 0 - 7, 2) line feed(<LF>), 3) carriage return(<CR>), or 4) control-U (CTRL U). No check is made for legality. If the input character is not a <LF>, <CR>, or CTRL U it is assumed to be an octal digit.

To change the contents of the SSR the operator simply types the new desired value in octal - leading zeros need not be typed. And terminates the input string with a <CR> or <LF> depending on the program action desired as described below. The input value will be truncated to the last 6 digits typed. At least one digit must be typed on any given input string prior to the terminator before a change to the SSR will occur.

When the input string is terminated with a <CR> the diagnostic will continue execution from the point at which it was interrupted. If a <CR> is the only thing typed the program will continue without changing the SSR. The <LF> differs from the <CR> by restarting the program as if it were restarted at address 200.

If a CTRL U is typed at any point in the input string prior to the terminator the input value will be disregarded and the prompt displayed (SWR = XXXXXX NEW?).

To set the SSR for the starting switches, first load the diagnostic, then hit CTRL G, then start the diagnostic.

DOCUMENT

CZDMGD LST

COPYRIGHT 1978
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

6 MAINDEC-11-CZDMG-D DMC11 CROM AND JUMP TESTS
 COPYRIGHT 1976, DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754

1675 ***** TEST 1 *****
 THIS IS A SPECIAL TEST WHICH IS ONLY EXECUTED ONE TIME,
 THE FIRST PASS AFTER THE DIAGNOSTIC IS LOADED. IT TYPES ON
 THE CONSOLE THE PART NUMBERS OF THE CROMS WHICH THIS
 REVISION SUPPORTS. TO FORCE A TYPE OUT PATCH LOCATION
 ROMNUM: TO A ZERO.

1696 ***** TEST 2 *****
 THIS IS A SPECIAL TEST WHICH WILL RUN ON A KMC (DMC WITH
 WRITABLE CONTROL STORE) TO LOAD THE CROM WITH THE DDCMP
 MICRO-CODE. FIRST BE SURE BIT1 OF STAT3 IS SET UP AS FOLLOWS
 1=LOCAL HIGH SPEED CODE, 0=REMOTE LOW SPEED CODE THE STATUS
 OF STAT3 BIT1 DETERMINES WHICH MICRO-CODE WILL
 BE LOADED IN THE KMC. LOOP ON THIS TEST FOR A FEW SECONDS
 TO LOAD THE KMC.

1727 ***** TEST 3 *****
 TEST OF BR RIGHT SHIFT
 VERIFY THAT A DEST OF BR RSH (011) OF A MICRO-INSTRUCTION
 SHIFTS THE RESULTING BR DATA RIGHT ONCE.

1768 ***** TEST 4 *****
 CROM READ TEST
 THIS TEST READS EACH ROM LOCATION AND COMPARES
 IT TO A SOFTWARE DUPLICATE OF THE CROM. THIS TEST
 ALSO TESTS THE JUMP(I) MICRO-PROCESSOR INSTRUCTION.

1773 IF THIS TEST FAILS CHECK YOUR CROM PART NUMBERS.
 CZDMG-D SUPPORTS THE FOLLOWING PART NUMBERS:

DMC11-AR (M8200-YA)
 23-630A9
 23-631A9
 23-632A9
 23-633A9
 23-634A9
 23-635A9
 23-636A9
 23-637A9

DMC11-AL (M8200-YB)
 23-622A9
 23-623A9
 23-624A9
 23-625A9
 23-626A9
 23-627A9
 23-628A9
 23-629A9

1840 ***** TEST 5 *****
CROM TEST OF JUMP(I) NEVER MICRO-PROCESSOR INSTRUCTION.
PERFORM THE JUMP INSTRUCTION
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

1898 ***** TEST 6 *****
CROM TEST OF JUMP(I) ALWAYS MICRO-PROCESSOR INSTRUCTION.
PERFORM THE JUMP INSTRUCTION
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

1952 ***** TEST 7 *****
CROM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
SET THE C BIT, PERFORM THE JUMP INSTRUCTION
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2009 ***** TEST 10 *****
CROM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
SET THE Z BIT, PERFORM THE JUMP INSTRUCTION
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2066 ***** TEST 11 *****
CROM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
SET THE BRO BIT, PERFORM THE JUMP INSTRUCTION
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2123 ***** TEST 12 *****
CROM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR1 BIT, PERFORM THE JUMP INSTRUCTION
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2180 ***** TEST 13 *****
CROM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR4 BIT, PERFORM THE JUMP INSTRUCTION
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2237 ***** TEST 14 *****
CROM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR7 BIT, PERFORM THE JUMP INSTRUCTION
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2294 ***** TEST 15 *****
CROM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE C BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

2352 ***** TEST 16 *****
CROM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE Z BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

2410 ***** TEST 17 *****
CROM TEST OF JUMP(I) ON BR0 SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE BR0 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

2468 ***** TEST 20 *****
CROM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE BR1 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

2526 ***** TEST 21 *****
CROM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE BR4 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

2584 ***** TEST 22 *****
CROM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE BR7 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

: *MAINDEC-11-CZDMG-D DMC11 CROM AND JUMP TESTS
: *COPYRIGHT 1976, DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754
: *

: STARTING PROCEDURE
: LOAD PROGRAM
: LOAD ADDRESS 000200
: SWR=0 AUTOSIZE DMC11
: SW07=1 USE CURRENT DMC11 PARAMETERS
: SW00=1 INPUT NEW DMC11 PARAMETERS
: PRESS START
: PROGRAM WILL TYPE "MAINDEC-11-CZDMG-D DMC11 CROM AND JUMP TESTS"
: PROGRAM WILL TYPE STATUS MAP
: PROGRAM WILL TYPE "R" TO INDICATE THAT TESTING HAS STARTED
: AT THE END OF A PASS, PROGRAM WILL TYPE PASS COMPLETE MESSAGE
: AND THEN RESUME TESTING
: SUBSEQUENT RESTARTS WILL NOT TYPE PROGRAM TITLE

: SWITCH REGISTER OPTIONS
: -----

100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001

SW15=100000 ;=1, HALT ON ERROR
SW14=40000 ;=1, LOOP ON CURRENT TEST
SW13=20000 ;=1, INHIBIT ERROR TYPEOUT
SW12=10000 ;=1, DELETE TYPEOUT/BELL ON ERROR.
SW11=4000 ;=1, INHIBIT ITERATIONS
SW10=2000 ;=1, ESCAPE TO NEXT TEST ON ERROR
SW09=1000 ;=1, LOOP WITH CURRENT DATA
SW08=400 ;=1, LOOP ON ERROR
SW07=200 ;=1, USE CURRENT DMC11 PARAMETERS, =0, AUTOSIZE DMC11
SW06=100 ;=1, HALT BEFORE CLOCKING MICRO-PROCESSOR INSTRUCTION

: RESELECT DMC11'S TO BE TESTED (ACTIVE)
: LOCK ON TEST SELECT
: RESTART PROGRAM AT SELECTED TEST
: INPUT DMC11 PARAMETERS

GENERAL DEFINATIONS AND EQUIVALENCIES

46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
00

; REGISTER DEFINITIONS
;-----

000000	R0=%0	; GENERAL REGISTER
000001	R1=%1	; GENERAL REGISTER
000002	R2=%2	; GENERAL REGISTER
000003	R3=%3	; GENERAL REGISTER
000004	R4=%4	; GENERAL REGISTER
000005	R5=%5	; GENERAL REGISTER
000006	SP=%6	; PROCESSOR STACK POINTER
000007	PC=%7	; PROGRAM COUNTER

; LOCATION EQUIVALENCIES
;-----

177776	PS=177776	; PROCESSOR STATUS WORD
001200	STACK=1200	; START OF PROCESSOR STACK

; INSTRUCTION DEFINITIONS
;-----

005746	PUSH1SP=5746	; DECREMENT PROCESSOR STACK 1 WORD
005726	POP1SP=5726	; INCREMENT PROCESSOR STACK 1 WORD
010046	PUSHR0=10046	; SAVE R0 ON STACK
012600	POP R0=12600	; RESTORE R0 FROM STACK
024646	PUSH2SP=24646	; DECREMENT STACK TWICE
022626	POP2SP=22626	; INCREMENT STACK TWICE
	.EQUIV EMT,HLT	; BASIC DEFINITION OF ERROR CALL

; BIT DEFINITIONS
;-----

100000	BIT15=100000
040000	BIT14=40000
020000	BIT13=20000
010000	BIT12=10000
004000	BIT11=4000
002000	BIT10=2000
001000	BIT9=1000
000400	BIT8=400
000200	BIT7=200
000100	BIT6=100
000040	BIT5=40
000020	BIT4=20
000010	BIT3=10
000004	BIT2=4
000002	BIT1=2
000001	BIT0=1

H02

CZDMGD.P11 05-JAN-78 11:08

TRAPCATCHER FOR UNEXPECTED INTERRUPTS

```

98
99
100
101
102
103
104
105
106
107
108      000000
109
110
111
112      000024
113 000024 005336
114 000026 000340
115 000030 004750
116 000032 000340
117 000034 004716
118 000036 000340
119      000040
120 000040 000000
121 000042 000000
122 000044 000000
123 000046 003522
124      000052
125 000052 000000
126
127      000174
128 000174 000000
129 000176 000000
130
131      000200
132 000200 000137 002002
133
134
135      001000
136 001000 005377 040515 047111
(2) 001025 104 041515 030461
(2)
137      001200
138
139
140
141
142 001200 177570
143 001202 177570

```

```

:*****
:-----
: TRAPCATCHER FOR ILLEGAL INTERRUPTS
: THE STANDARD "TRAP CATCHER" IS PLACED
: BETWEEN ADDRESS 0 TO ADDRESS 776.
: IT LOOKS LIKE "PC+2 HALT".
:-----
:*****
.=0
: STANDARD INTERRUPT VECTORS
:-----
.=24
      .PFAIL          ; POWER FAIL HANDLER
      340             ; SERVICE AT LEVEL 7
      .HLT            ; ERROR HANDLER
      340             ; SERVICE AT LEVEL 7
      .TRPSRV         ; GENERAL HANDLER DISPATCH SERVICE
      340             ; SERVICE AT LEVEL 7
.=40
      0               ; SAVE FOR ACT-11 OR XXDP
      0               ; RETURN ADDRESS IF UNDER ACT-11 OR XXDP
      0               ; SAVE FOR ACT-11 OR XXDP
      $ENDAD          ; FOR USE WITH ACT-11 OR XXDP
.=52
      0               ; ACT-11 PROGRAM CHARACTERISTICS
.=174
DISPREG: 0           ; SOFTWARE DISPLAY REGISTER
SWREG: 0            ; SOFTWARE SWITCH REGISTER
.=200
      JMP      .START ; GO TO START OF PROGRAM
.=1000
MTITLE: .ASCII <377><12>/MAINDEC-11-CZDMG-D/<377>
        .ASCIIZ /DMC11 CROM AND JUMP TESTS/<377>
.=1200
: INDIRECT POINTERS TO SWITCH REGISTER AND LIGHT DISPLAY
:-----
DISPLAY: 177570
SWR:     177570

```



```

144
145 ;INDIRECT POINTERS TO TELETYPE VECTORS AND REGISTERS
146 -----
147
148 001204 177560 TKCSR: 177560 ;TELETYPE KEYBOARD CONTROL REGISTER
149 001206 177562 TKDBR: 177562 ;TELETYPE KEYBOARD DATA BUFFER
150 001210 177564 TPCSR: 177564 ;TELEPRINTER CONTROL REGISTER
151 001212 177566 TPDBR: 177566 ;TELEPRINTER DATA BUFFER
152
153 ;PROGRAM CONTROL PARAMETERS
154 -----
155
156 001214 000000 RETURN: 0 ;SCOPE ADDRESS FOR LOOP ON TEST
157 001216 000000 NEXT: 0 ;ADDRESS OF NEXT TEST TO BE EXECUTED
158 001220 000000 LOCK: 0 ;ADDRESS FOR LOCK ON CURRENT DATA
159 001222 000003 ICOUNT: 3 ;NUMBER OF ITERATIONS THAT CURRENT TEST WILL BE EXECUTED
160 001224 000000 LPCNT: 0 ;NUMBER OF ITERATIONS COMPLETED
161 001226 000000 TSTNO: 0 ;NUMBER OF TEST IN PROGRESS
162 001230 000000 PASCNT: 0 ;NUMBER OF PASSES COMPLETED
163 001232 000000 ERRCNT: 0 ;TOTAL NUMBER OF ERRORS
164 001234 000000 LSTERR: 0 ;PC OF LAST ERROR CALL
165
166 ;PROGRAM VARIABLES
167 -----
168
169 001236 000000 STRTSW: 0 ;SWITCHES AT START OF PROGRAM
170 001240 000000 STAT: 0 ;DM STATUS WORD STORAGE
171 001242 000000 CLKX: 0
172 001244 000000 MASKX: 0
173 001246 000000 TEMP1: 0 ;TEMPORARY STORAGE
174 001250 000000 TEMP2: 0 ;TEMPORARY STORAGE
175 001252 000000 TEMP3: 0 ;TEMPORARY STORAGE
176 001254 000000 TEMP4: 0 ;TEMPORARY STORAGE
177 001256 000000 TEMPS: 0 ;TEMPORARY STORAGE
178 001260 000000 SAVR0: 0 ;R0 STORAGE
179 001262 000000 SAVR1: 0 ;R1 STORAGE
180 001264 000000 SAVR2: 0 ;R2 STORAGE
181 001266 000000 SAVR3: 0 ;R3 STORAGE
182 001270 000000 SAVR4: 0 ;R4 STORAGE
183 001272 000000 SAVR5: 0 ;R5 STORAGE
184 001274 000000 SAVSP: 0 ;STACK PCINTER STORAGE
185 001276 000000 SAVPC: 0 ;PROGRAM COUNTER STORAGE
186 001300 000000 ZERO: 0
187 001302 000001 ONE: 1
188 001304 000000 MEMLIM: 0 ;HIGHEST LOCATION FOR NPR'S
189 001306 000001 DMACTV: .BLKW 1 ;DMC11'S SELECTED ACTIVE.
190 001310 000001 DMNUM: .BLKW 1 ;OCTAL NUMBER OF DMC11'S.
191 001312 000001 SAVACT: .BLKW 1 ;ORIGINAL ACTV DEVICES
192 001314 000001 SAVNUM: .BLKW 1 ;WORKABLE NUMBER
193 001316 000000 RUN: 0 ;POINTER TO RUNNING DEVICE.
194
195 001320 001472 CREAM: DM.MAP-6 ;TABLE POINTER.
196 001322 001676 MILK: CNT.MAP-4 ;TABLE POINTER

```

CZDMGD.P11 05-JAN-78 11:08

PROGRAM PARAMETERS, VARIABLES, AND TRAP CALLS.

197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247

001324 000
001325 00C
001326 000
001327 000

001330 104400
001330 003576
001332 104401
001332 003736
001334 104402
001334 003766
001336 104403
001336 004050
001340 104404
001340 004154
001342 104405
001342 004174
001344 104406
001344 004374
001346 104407
001346 004434
001350 104410
001350 004466
001352 104411
001352 004472
001354 104412
001354 005466
001356 104413
001356 005436
001360 104414
001360 005504
001362 104415
001362 005552
001364 104416
001364 005616

;PROGRAM CONTROL FLAGS
;-----

INIFLG: .BYTE 0
ERRFLG: .BYTE 0
LOKFLG: .BYTE 0
QV.FLG: .BYTE 0

;PROGRAM INITIALIZATION FLAG
;ERROR OCCURED FLAG
;LOCK ON CURRENT TEST FLAG
;QUICK VERIFY FLAG.
;ON FIRST PASS OF EACH DMC11 ITERATIONS WILL BE SUPPRESS

.EVEN

;DEFINITIONS FOR TRAP SUBROUTINE CALLS
;POINTERS TO SUBROUTINES CAN BE FOUND
;IN THE TABLE IMMEDIATLY FOLLOWING THE DEFINITIONS

;*****

;-----
;TRPTAB:

SCOPE=TRAP+0 ;CALL TO SCOPE LOOP AND ITERATION HANDLER
SCOPE ;SCOPE
SCOPI=TRAP+1 ;CALL TO LOOP ON CURRENT DATA HANDLER
SCOPI ;SCOPI
TYPE=TRAP+2 ;CALL TO TELETYPE OUTPUT ROUTINE
TYPE ;TYPE
INSTR=TRAP+3 ;CALL TO ASCII STRING INPUT ROUTINE
INSTR ;INSTR
INSTER=TRAP+4 ;CALL TO INPUT ERROR HANDLER
INSTER ;INSTER
PARAM=TRAP+5 ;CALL TO NUMERICAL DATA INPUT ROUTINE
PARAM ;PARAM
SAVOS=TRAP+6 ;CALL TO REGISTER SAVE ROUTINE
SAVOS ;SAVOS
RESOS=TRAP+7 ;CALL TO REGISTER RESTORE ROUTINE
RESOS ;RESOS
CONVRT=TRAP+10 ;CALL TO DATA OUTPUT ROUTINE
CONVRT ;CONVRT
CNVRT=TRAP+11 ;CALL TO DATA OUTPUT ROUTINE WITHOUT CR LF.
CNVRT ;CNVRT
MSTCLR=TRAP+12 ;CALL TO ISUE A MASTER CLEAR
MSTCLR ;MSTCLR
DELAY=TRAP+13 ;CALL TO DELAY
DELAY ;DELAY
ROMCLK=TRAP+14 ;CALL TO CLOCK ROM ONCE
ROMCLK ;ROMCLK
DATACLK=TRAP+15 ;CALL TO CLK DATA
DATACLK ;DATACLK
TIMER=TRAP+16 ;CALL TO DELAY A CLOCK TICK
TIMER ;TIMER

;-----
;*****

```

248 ;DMC11 CONTROL INDICATORS FOR CURRENT DMC11 UNDER TEST
249 ;-----
250
251 001366 000000 STAT1: 0
252 001370 000000 STAT2: 0
253 001372 000000 STAT3: 0
254
255 ;DMC11 VECTOR AND REGISTER INDIRECT POINTERS
256 ;-----
257
258 001374 000000 DMRVEC: 0 ; POINTER TO DMC11 RECEIVER INTERRUPT VECTOR
259 001376 000000 DMRLVL: 0 ; POINTER TO DMC11 RECEIVER INTERRUPT SERVICE PS
260 001400 000000 DMTVEC: 0 ; POINTER TO DMC11 TRANSMITTER INTERRUPT VECTOR
261 001402 000000 DMTLVL: 0 ; POINTER TO DMC11 TRANSMITTER INTERRUPT SERVICE PS
262 001404 000000 DMCSR: 0 ; POINTER TO DMC11 CONTROL STATUS REGISTER
263 001406 000000 DMCSRH: 0 ; POINTER TO DMC11 CONTROL STATUS REGISTER HIGH BYTE.
264 001410 000000 DMCTL: 0 ; POINTER TO DMC11 CONTROL OUT REGISTER
265 001412 000000 DMP04: 0 ; POINTER TO DMC11 PORT REGISTER(SEL 4)
266 001414 000000 DMP06: 0 ; POINTER TO DMC11 PORT REGISTER(SEL 6)
267
268 ;TEMP STORAGE
269 ;-----
270
271 001416 000000 TEMP: 0
272 001460 .=. +40
273
274 ;DMC11 STATUS TABLE AND ADDRESS ASSIGNMENTS
275 ;-----
276
277 . =1500
278 001500 DM.MAP:
279 001500 000001 DMCRO0: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 00
280 001502 000001 DMS100: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 00
281 001504 000001 DMS200: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 00
282 001506 000001 DMS300: .BLKW 1 ; 3RD STATUS WORD
283
284 001510 000001 DMCRO1: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 01
285 001512 000001 DMS101: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 01
286 001514 000001 DMS201: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 01
287 001516 000001 DMS301: .BLKW 1 ; 3RD STATUS WORD
288
289 001520 000001 DMCRO2: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 02
290 001522 000001 DMS102: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 02
291 001524 000001 DMS202: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 02
292 001526 000001 DMS302: .BLKW 1 ; 3RD STATUS WORD
293
294 001530 000001 DMCRO3: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 03
295 001532 000001 DMS103: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 03
296 001534 000001 DMS203: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 03
297 001536 000001 DMS303: .BLKW 1 ; 3RD STATUS WORD
298
299 001540 000001 DMCRO4: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 04
300 001542 000001 DMS104: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 04
301 001544 000001 DMS204: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 04
302 001546 000001 DMS304: .BLKW 1 ; 3RD STATUS WORD
303

```

CZDMGO.P11 05-JAN-78 11:08

PROGRAM PARAMETERS, VARIABLES, AND TRAP CALLS.

304	001550	000001	DMCR05: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 05
305	001552	000001	DMS105: .BLKW	1	;VECTOR FOR DMC11 NUMBER 05
306	001554	000001	DMS205: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 05
307	001556	000001	DMS305: .BLKW	1	;3RD STATUS WORD
308					
309	001560	000001	DMCR06: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 06
310	001562	000001	DMS106: .BLKW	1	;VECTOR FOR DMC11 NUMBER 06
311	001564	000001	DMS206: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 06
312	001566	000001	DMS306: .BLKW	1	;3RD STATUS WORD
313					
314	001570	000001	DMCR07: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 07
315	001572	000001	DMS107: .BLKW	1	;VECTOR FOR DMC11 NUMBER 07
316	001574	000001	DMS207: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 07
317	001576	000001	DMS307: .BLKW	1	;3RD STATUS WORD
318					
319	001600	000001	DMCR10: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 10
320	001602	000001	DMS110: .BLKW	1	;VECTOR FOR DMC11 NUMBER 10
321	001604	000001	DMS210: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 10
322	001606	000001	DMS310: .BLKW	1	;3RD STATUS WORD
323					
324	001610	000001	DMCR11: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 11
325	001612	000001	DMS111: .BLKW	1	;VECTOR FOR DMC11 NUMBER 11
326	001614	000001	DMS211: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 11
327	001616	000001	DMS311: .BLKW	1	;3RD STATUS WORD
328					
329	001620	000001	DMCR12: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 12
330	001622	000001	DMS112: .BLKW	1	;VECTOR FOR DMC11 NUMBER 12
331	001624	000001	DMS212: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 12
332	001626	000001	DMS312: .BLKW	1	;3RD STATUS WORD
333					
334	001630	000001	DMCR13: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 13
335	001632	000001	DMS113: .BLKW	1	;VECTOR FOR DMC11 NUMBER 13
336	001634	000001	DMS213: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 13
337	001636	000001	DMS313: .BLKW	1	;3RD STATUS WORD
338					
339	001640	000001	DMCR14: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 14
340	001642	000001	DMS114: .BLKW	1	;VECTOR FOR DMC11 NUMBER 14
341	001644	000001	DMS214: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 14
342	001646	000001	DMS314: .BLKW	1	;3RD STATUS WORD
343					
344	001650	000001	DMCR15: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 15
345	001652	000001	DMS115: .BLKW	1	;VECTOR FOR DMC11 NUMBER 15
346	001654	000001	DMS215: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 15
347	001656	000001	DMS315: .BLKW	1	;3RD STATUS WORD
348					
349	001660	000001	DMCR16: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 16
350	001662	000001	DMS116: .BLKW	1	;VECTOR FOR DMC11 NUMBER 16
351	001664	000001	DMS216: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 16
352	001666	000001	DMS316: .BLKW	1	;3RD STATUS WORD
353					
354	001670	000001	DMCR17: .BLKW	1	;CONTROL STATUS REGISTER FOR DMC11 NUMBER 17
355	001672	000001	DMS117: .BLKW	1	;VECTOR FOR DMC11 NUMBER 17
356	001674	000001	DMS217: .BLKW	1	;DDCMP LINE# FOR DMC11 NUMBER 17
357	001676	000001	DMS317: .BLKW	1	;3RD STATUS WORD
358					
359	001700	000000	DM.END: 000000		

M02

CZDMGD.P11 05-JAN-78 11:08

PROGRAM PARAMETERS, VARIABLES, AND TRAP CALLS.

PAGE: 0025CZ
SEQ 0025

```

360
361 ;DMC11 PASS COUNT AND ERROR COUNT TABLE
362 ;-----
363
364 CNT.MAP:
365 001702 000000 PACT00: 0 ;PASS COUNT FOR DMC11 NUMBER 00
366 001704 000000 ERCT00: 0 ;ERROR COUNT FOR DMC11 NUMBER 00
367
368 001706 000000 PACT01: 0 ;PASS COUNT FOR DMC11 NUMBER 01
369 001710 000000 ERCT01: 0 ;ERROR COUNT FOR DMC11 NUMBER 01
370
371 001712 000000 PACT02: 0 ;PASS COUNT FOR DMC11 NUMBER 02
372 001714 000000 ERCT02: 0 ;ERROR COUNT FOR DMC11 NUMBER 02
373
374 001716 000000 PACT03: 0 ;PASS COUNT FOR DMC11 NUMBER 03
375 001720 000000 ERCT03: 0 ;ERROR COUNT FOR DMC11 NUMBER 03
376
377 001722 000000 PACT04: 0 ;PASS COUNT FOR DMC11 NUMBER 04
378 001724 000000 ERCT04: 0 ;ERROR COUNT FOR DMC11 NUMBER 04
379
380 001726 000000 PACT05: 0 ;PASS COUNT FOR DMC11 NUMBER 05
381 001730 000000 ERCT05: 0 ;ERROR COUNT FOR DMC11 NUMBER 05
382
383 001732 000000 PACT06: 0 ;PASS COUNT FOR DMC11 NUMBER 06
384 001734 000000 ERCT06: 0 ;ERROR COUNT FOR DMC11 NUMBER 06
385
386 001736 000000 PACT07: 0 ;PASS COUNT FOR DMC11 NUMBER 07
387 001740 000000 ERCT07: 0 ;ERROR COUNT FOR DMC11 NUMBER 07
388
389 001742 000000 PACT10: 0 ;PASS COUNT FOR DMC11 NUMBER 10
390 001744 000000 ERCT10: 0 ;ERROR COUNT FOR DMC11 NUMBER 10
391
392 001746 000000 PACT11: 0 ;PASS COUNT FOR DMC11 NUMBER 11
393 001750 000000 ERCT11: 0 ;ERROR COUNT FOR DMC11 NUMBER 11
394
395 001752 000000 PACT12: 0 ;PASS COUNT FOR DMC11 NUMBER 12
396 001754 000000 ERCT12: 0 ;ERROR COUNT FOR DMC11 NUMBER 12
397
398 001756 000000 PACT13: 0 ;PASS COUNT FOR DMC11 NUMBER 13
399 001760 000000 ERCT13: 0 ;ERROR COUNT FOR DMC11 NUMBER 13
400
401 001762 000000 PACT14: 0 ;PASS COUNT FOR DMC11 NUMBER 14
402 001764 000000 ERCT14: 0 ;ERROR COUNT FOR DMC11 NUMBER 14
403
404 001766 000000 PACT15: 0 ;PASS COUNT FOR DMC11 NUMBER 15
405 001770 000000 ERCT15: 0 ;ERROR COUNT FOR DMC11 NUMBER 15
406
407 001772 000000 PACT16: 0 ;PASS COUNT FOR DMC11 NUMBER 16
408 001774 000000 ERCT16: 0 ;ERROR COUNT FOR DMC11 NUMBER 16
409
410 001776 000000 PACT17: 0 ;PASS COUNT FOR DMC11 NUMBER 17
411 002000 000000 ERCT17: 0 ;ERROR COUNT FOR DMC11 NUMBER 17
412

```

FORMAT OF STATUS TABLE

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	CSR
I	C	O	N	T	R	O	L	R	E	G	I	S	T	E	R	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	
I	*	I	*	I	*	I	*	I	*	I	*	I	*	I	*	STAT1
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	
I	*	I	B	I	M	I	A	I	D	I	D	*	I	*	I	STAT2
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	STAT3
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	

DEFINITION OF FORMAT

- CSR: CONTAINS DMC11 CSR ADDRESS
- STAT1: BITS 00-08 IS DMC11 VECTOR ADDRESS
 BIT15=1 MICRO-PROCESSOR HAS CROM
 BIT15=0 MICRO-PROCESSOR HAS CROM
 BIT14=1 ??? TURNAROUND CONNECTOR IS ON
 BIT14=0 NO TURNAROUND CONNECTOR
 BIT13=0 LINE UNIT IS AN M8201
 BIT13=1 LINE UNIT IS AN M8202
 BIT12=1 NO LINE UNIT
 BITS 09-11 IS DMC11 BR PRIORITY LEVEL
- STAT2: LOW BYTE IS SWITCH PAC#1 (DDCMP LINE NUMBER)
 HIGH BYTE IS SWITCH PAC#2 (BM873 BOOT ADD)
- STAT3: BIT0=1 DO FREE RUNNING TESTS ON KMC
 (MUST BE SET TO A ONE MANUALLY (PROGRAM DZDMI ONLY))
 KMC MUST HAVE MICRO-CODE WRITTEN FROM RUNNING
 DZDMG TEST 2 FIRST
 BIT1=1 DMC11-AL LOCAL HIGH SPEED MICRO-CODE
 BIT1=0 DMC11-AR REMOTE LOW SPEED MICRO-CODE

CZDMGD.P11 05-JAN-78 11:08

PROGRAM INITIALIZATION AND START UP.

```

468 ;PROGRAM INITIALIZATION
469 ;LOCK OUT INTERRUPTS
470 ;SET UP PROCESSOR STACK
471 ;SET UP POWER FAIL VECTOR
472 ;CLEAR PROGRAM CONTROL FLAGS AND COUNTS
473 ;TYPE TITLE MESSAGE
474
475
476 0020C2 012737 000340 177776 .START: MOV #340,PS ;LOCK OUT INTERRUPTS
477 002010 012706 001200 MOV #STACK,SP ;SET UP STACK
478 002014 012737 005336 000024 MOV #PFAIL,#24 ;SET UP POWER FAIL VECTOR
479 002022 013737 001310 001314 MOV DMNUM,SANUM ;SAVE NUMBER OF DEVICES IN SYSTEM.
480 002030 005037 010020 CLR SWFLG ;CLEAR SOFT TIMEOUT FLAG
481 002034 105037 001325 CLRB ERRFLG ;CLEAR ERROR FLAG
482 002040 105037 001327 CLRB QV.FLG ;ZERO QUICK VERIFY FLAG
483 002044 012737 001470 001320 MOV #DM.MAP-10,CREAM ;GET MAP POINTER.
484 002052 012737 001676 001322 MOV #CNT.MAP-4,MILK ;GET PASS COUNT MAP POINTER
485 002060 012737 100000 001316 MOV #BIT15,RUN ;POINT POINTER TO FIRST DEVICE.
486 002066 012700 001702 MOV #CNT.MAP,RO ;PASS COUNT POINTER TO RO
487 002072 005020 23$: CLR (RO)+ ;CLEAR TABLE
488 002074 022700 002002 CMP #CNT.MAP+100,RO ;DONE YET?
489 002100 001374 BNE 23$ ;KEEP GOING
490 002102 005037 001234 CLR LSTERR ;CLEAR LAST ERROR POINTER
491 002106 012737 000001 0012cb MOV #1,TSTNO ;SET UP FOR TEST 1
492 002114 012737 002002 001214 MOV #.START,RETURN ;SET UP FOR POWER FAIL BEFORE
493 ;TESTING STARTS
494 002122 013746 000006 MOV #6,-(SP) ;SAVE CURRENT VECTORS
495 002126 013746 000004 MOV #4,-(SP)
496 002132 012737 002166 000004 MOV #6$,#4 ;SET UP FOR TIMEOUT
497 002140 012737 177570 001202 MOV #177570,SWR ;SET SWR TO HARD SWR ADDRESS
498 002146 012737 177570 001200 MOV #177570,DISPLAY ;SET DISPLAY TO HARD SWR ADDRESS
499 002154 022777 177777 177020 CMP #-1,#SWR ;REFERENCE HARDWARE SWITCH REGISTER
500 002162 001402 BEQ 6$+2 ;IF = -1 USE SOFT SWR ANYWAY
501 002164 000407 BR 7$ ;IF IT EXISTS AND NOT = -1 USE HARD SWR
502 002166 022626 6$: CMP (SP)+(SP)+ ;ADJUST STACK
503 002170 012737 000176 001202 MOV #SWREG,SWR ;POINTER TO SOFT SWR
504 002176 012737 000174 001200 MOV #DISPREG,DISPLAY ;POINTER TO SOFT DISPLAY REG
505 002204 012637 000004 7$: MOV (SP)+,#4 ;RESTORE VECTORS
506 002210 012637 000006 MOV (SP)+,#6
507 002214 105737 001324 TSTB INIFLG ;HAS INITIALIZATION BEEN PERFORMED
508 002220 001006 BNE 20$ ;BR IF YES
509 002222 022737 003522 000042 CMP #SENDAD,#42 ;IF ACT-11 AUTOMATIC MODE, DON'T TYPE ID
510 002230 001402 BEQ 20$
511 002232 104402 001000 TYPE #MTITLE ;TYPE TITLE MESSAGE
512 002236 004737 007610 JSR PC,CKSWR ;CHECK FOR SOFT SWR
513 002242 017737 176734 001236 MOV #SWR,STRTSW ;STORE STARTING SWITCHES
514 002250 005737 000042 TST #42 ;IS IT RUNNING IN AUTO MODE?
515 002254 001402 BEQ .+6 ;BR IF NO
516 002256 005037 001236 CLR STRTSW ;IF YES, CLEAR SWITCHES
517 002262 032737 000001 001236 BIT #SW00,STRTSW ;IF SW00=1, QUESTIONS ARE ASKED.
518 002270 001012 BNE 17$ ;BR IF SW00=1
519 002272 105737 001236 TSTB STRTSW ;BIT7=1??
520 002276 100007 BPL 17$ ;BR IF SW07=0
521 002300 005737 001306 TST DMACTV ;ARE ANY DEVICES SELECTED?
522 002304 001006 BNE 16$ ;BR IF YES
523 002306 104402 007155 TYPE, NOACT ;NO DEVICES SELECTED.

```

CZDMGD.P11 05-JAN-78 11:08

PROGRAM INITIALIZATION AND START UP.

```

524 002312 000000          HALT                ;STOP THE SHOW
525 002314 000776          BR                ;DISQUALIFY CONTINUE SWITCH
526 002316 004737 010514   17$: JSR          PC,AUTO.SIZE ;GO DO THE AUTO SIZE
527 002322 105737 001324   16$: TSTB         INIFLG        ;FIRST TIME?
528 002326 001410          BEQ          21$          ;BR IF YES
529 002330 105737 001236   TSTB         STRTSW        ;IF USING SAME PARAMETERS DONT TYPE MAP
530 002334 100431          BMI          1$
531 002336 032737 000006 001236  BIT          #BIT1!BIT2,STRTSW;IS TEST NO. OR LOCK SELECTED
532 002344 001403          BEQ          24$          ;IF NO THEN TYPE STATUS
533 002346 000424          BR                ;IF YES DO NOT TYPE STATUS
534 002350 005137 001324   21$: COM          INIFLG        ;SET FLAG
535 002354 104402 006225   24$: TYPE         XHEAD        ;TYPE HEADER
536 002360 012704 001500   MOV          #DM.MAP,R4    ;SET POINTER
537 002364 010437 001246   5$:  MOV          R4,TEMP1    ;SET ADDRESS
538 002370 012437 001250   MOV          (R4)+,TEMP2   ;SET CSR
539 002374 001411          BEQ          1$
540 002376 012437 001252   MOV          (R4)+,TEMP3   ;SET STAT1
541 002402 012437 001254   MOV          (R4)+,TEMP4   ;SET STAT2
542 002406 012437 001256   MOV          (R4)+,TEMP5   ;SET STAT3
543 002412 104410          CONVRT         ;TYPE OUT STATUS MAP
544 002414 007456          XSTATQ
545 002416 000762          BR                ;
546 002420 012700 001500   1$:  MOV          #DM.MAP,RO  ;RO POINTS TO STATUS TABLE

```

```

547
548 ::*****
549 ::*AUTO SIZE TEST
550 ::*THIS TEST VERIFYS THAT THE DMC11S AND/OR KMC11S ARE AT THE CORRECT FLOATING
551 ::*ADDRESSES FOR YOUR SYSTEM. IF THIS TEST FAILS, IT IS NOT A HARDWARE ERROR.
552 ::*CHECK THE ADDRESSES OF ALL FLOATING DEVICES (DJ,DH,DA,DU,DUP,LK,DMC,DZ,KMC).
553 ::*IF THERE ARE NO OTHER FLOATING DEVICES BEFORE THE DMC11, THE FIRST
554 ::*DMC11 ADDRESS IS 760070, KMC11 IS 760110. NO DEVICE SHOULD EVER BE AT
555 ::*ADDRESS 760000. THIS TEST MAY REQUIRE 2 OR MORE ATTEMPTS TO GET THE
556 ::*RIGHT ADDRESSES. AFTER YOU HAVE CHANGED THE ADDRESS TO WHAT IT TOLD
557 ::*YOU THE FIRST TIME, IT MAY COME BACK AND TELL YOU A DIFFERENT ADDRESS
558 ::*THE NEXT TIME YOU RUN IT. PLEASE HAVE PATIENCE, THE FINAL ADDRESS
559 ::*WILL BE CORRECT (AS LONG AS ALL DEVICES IN FRONT OF THE DMC'S ARE
560 ::*CORRECT).
561 ::*****

```

```

562
563 002424 013746 000004          MOV          2#4,-(SP)    ;SAVE LOC 4
564 002430 013746 000006          MOV          2#6,-(SP)    ;SAVE LOC 6
565 002434 005037 000006          CLR          2#6          ;CLEAR VEC+2
566 002440 005037 001252          CLR          TEMP3       ;CLEAR FLAG
567 002444 005005          CLR          R5          ;R5=0=DMC, R5=-1=KMC
568 002446 011C37 001404   AUSTRT: MOV          (RO),DMCSR ;GET NEXT DMC CSR
569 002452 001564          BEQ          AUDONE      ;BR IF DONE
570 002454 005705          TST          R5          ;DMC OR KMC?
571 002456 001005          BNE          1$          ;BR IF KMC
572 002460 032760 100000 000002  BIT          #BIT15,2(RO) ;CHECK FOR DMC CSR
573 002466 001061          BNE          SKIP        ;SKIP IF NOT DMC
574 002470 000404          BR          2$          ;ITS A DMC SO CONTINUE
575 002472 032760 100000 000002  1$:  BIT          #BIT15,2(RO) ;CHECK FOR KMC CSR
576 002500 001454          BEQ          SKIP        ;SKIP IF NOT KMC
577 002502 012737 002674 000004  2$:  MOV          #NODEV,2#4  ;SET UP FOR TIMEOUT
578 002510 005705          TST          R5          ;DMC OR KMC
579 002512 001003          BNE          3$          ;BR IF KMC

```


CZDMGD.P11 05-JAN-78 11:08

PROGRAM INITIALIZATION AND START UP.

580	002514	012703	000006		MOV	#6,R3	;R3 IS COUNT OF DEVICES BEFORE DMC
581	002520	000402			BR	4\$;GO ON
582	002522	012703	000010		3\$: MOV	#10,R3	;R3 IS COUNT OF DEVICES BEFORE KMC
583	002526	012702	003010		4\$: MOV	#DEVTAB,R2	;R2 IS DEVICE TABLE POINTER
584	002532	012701	160010		MOV	#160010,R1	;START WITH ADDRESS 160010
585	002536	005711			FLOAT: TST	(R1)	;CHECK ADDRESS IN R1
586	002540	111204			MOV#B	(R2),R4	;IF NO TIMEOUT, GET NEXT ADDRESS
587	002542	060401			ADD	R4,R1	;IN R1
588	002544	005201			INC	R1	
589	002546	040401			BIC	R4,R1	
590	002550	005703			TST	R3	;ANY MORE DEVICES TO CHECK FOR?
591	002552	001371			BNE	FLOAT	;BR IF YES
592	002554	012737	002700	000004	MOV	#ERR,2#4	;OK ONLY DMC'S ARE LEFT, SET UP FOR TIMEOUT
593	002562	010137	003022		MOV	R1,XLOC	;SAVE FIRST DMC/KMC ADDRESS
594	002566	005705			FY: TST	R5	;DMC OR KMC?
595	002570	001005			BNE	1\$;BR IF KMC
596	002572	032760	100000	000002	BIT	#BIT15,2(R0)	;CHECK FOR DMC CSR
597	002600	001014			BNE	SKIP	;SKIP IF NOT DMC
598	002602	000404			BR	2\$;ITS A DMC SO CONTINUE
599	002604	032760	100000	000002	1\$: BIT	#BIT15,2(R0)	;CHECK FOR KMC CSR
600	002612	001407			BEQ	SKIP	;SKIP IF NOT KMC
601	002614	005711			2\$: TST	(R1)	;CHECK DMC ADDRESS
602	002616	020137	001404		CMP	R1,DMCSR	;DOES IT MATCH
603	002622	001411			BEQ	OK	;BR IF YES
604	002624	062701	000010		ADD	#10,R1	;GET NEXT DMC ADDRESS
605	002630	000756			BR	FY	;DO IT AGAIN
606	002632	062700	000010		SKIP: ADD	#10,R0	;SKIP TO NEXT CSR IN TABLE
607	002636	011037	001404		MOV	(R0),DMCSR	;GET NEXT CSR
608	002642	001470			BEQ	AUDONE	;BR IF DONE
609	002644	000750			BR	FY	;ELSE CONTINUE
610	002646	062700	000010		OK: ADD	#10,R0	;SKIP TO NEXT DMC CSR
611	002652	062737	000010	003022	ADD	#10,XLOC	;UPDATE EXPECTED DMC/KMC ADDRESS
612	002660	011037	001404		MOV	(R0),DMCSR	;GET NEXT DMC/KMC CSR
613	002664	001457			BEQ	AUDONE	;BR IF DONE
614	002666	013701	003022		MOV	XLOC,R1	;GET EXPECTED DMC/KMC ADDRESS
615	002672	000735			BR	FY	;CONTINUE
616	002674	122243			NODEV: CMP#B	(R2)+,-(R3)	;ON TIMEOUT, INC R2, DEC R3
617	002676	000002			RTI		;RETURN
618	002700	005737	001252		ERR: TST	TEMP3	;CHECK FLAG IF = 0 TYPE HEADER
619	002704	001014			BNE	1\$;SKIP HEADER
620	002706	104402			TYPE		;TYPEOUT HEADER MESSAGE
621	002710	007224			CONERR		;CONFIGURATION ERROR!!!!
622	002712	012737	002700	001276	MOV	#ERR,SAVPC	;SAVE PC FOR TYPEOUT
623	002720	104411			CONVRT		;TYPE OUT ERROR PC
624	002722	002770			ERRPC		
625	002724	104402			TYPE		;TYPE REST OF HEADER
626	002726	007300			CNERR		
627	002730	012737	177777	001252	MOV	#-1,TEMP3	;SET FLAG SO IT ONLY GETS TYPED ONCE
628	002736	010137	001262		1\$: MOV	R1,SAVR1	;SAVE R1 FOR TYPEOUT
629	002742	104410			CONVRT		
630	002744	002776			CONTAB		;TYPE CSR VALUES
631	002746	005705			TST	R5	;DMC OR KMC ?
632	002750	001003			BNE	3\$;BR IF KMC
633	002752	104402			TYPE		
634	002754	007321			DMCM		
635	002756	000402			BR	4\$;CONTINUE

CZDMGD.P11 05-JAN-78 11:08

PROGRAM INITIALIZATION AND START UP.

636	002760	104402			3\$:	TYPE		
637	002762	007331				KMCM		
638	002764	022626			4\$:	CMP	(SP)+,(SP)+	;ADJUST STACK
639	002766	000727				BR	OK	;BR TO GET OUT
640	002770	000001			ERRPC:	1		
641	002772	006	002			.BYTE	6,2	
642	002774	001276				SAVPC		
643	002776	000002			CONTAB:	2		
644	003000	006	004			.BYTE	6,4	
645	003002	003022				XLOC		
646	003004	006	002			.BYTE	6,2	
647	003006	001404				DMCSR		
648	003010	007			DEVTAB:	.BYTE	7	;DJ
649	003011	017				.BYTE	17	;DH
650	003012	007				.BYTE	7	;DQ
651	003013	007				.BYTE	7	;DU
652	003014	007				.BYTE	7	;DUP
653	003015	007				.BYTE	7	;LK
654	003016	007				.BYTE	7	;DMC
655	003017	007				.BYTE	7	;DZ
656	003020	007				.BYTE	7	;KMC
657		003022			.EVEN			
658	003022	000000			XLOC:	D		
659	003024	005705			AUDONE:	TST	R5	;DMC?
660	003026	001005				BNE	1\$;BR IF KMC AND ALL DONE
661	003030	012705	177777			MOV	#-1,R5	;SET R5 TO -1 (KMC)
662	003034	012700	001500			MOV	#DM.MAP,RO	;RESET RO TO START OF TABLE
663	003040	000602				BR	AUSTRT	;GO DO KMC'S
664	003042	012637	000006		1\$:	MOV	(SP)+,Q#6	;RESTORE LOC 6
665	003046	012637	000004			MOV	(SP)+,Q#4	;RESTORE LOC 4
666	003052	032737	000010	001236		BIT	#SW03,STRTSW	;SELECT SPECIFIC DEVICES??
667	003060	001422				BEQ	3\$;BR IF NO.
668	003062	104402	006145			TYPE	MNEW	;TYPE THE MESSAGE.
669	003066	005000				CLR	RO	;ZERO DATA LIGHTS
670	003070	000000				HALT		;WAIT FOR USER TO TELL WHAT DEVICES TO RUN
671	003072	027737	176104	001312		CMP	QSWR,SAVACT	;IS THE NUMBER VALID?
672	003100	101404				BLOS	2\$;BR IF NUMBER IS OK.
673	003102	104402	006006			TYPE	,MERR3	;TELL USER OF INVALID NUMBER.
674	003106	000000				HALT		;STOP EVERY THING.
675	003110	000776				BR	-2	;RESTART THE PROGRAM AGAIN.
676	003112	017737	176064	001306	2\$:	MOV	QSWR,DMACTV	;GET NEW DEVICE PATTERN
677	003120	013700	001306			MOV	DMACTV,RO	;SHOW THE USER WHAT HE SELECTED.
678	003124	000000				HALT		;CONTINUE DYNAMIC SWITCHES.
679	003126	012700	000300		3\$:	MOV	#300,RO	;PREPARE TO CLEAR THE FLOATING
680	003132	012701	000302			MOV	#302,R1	;VECTOR AREA. 300-776
681	003136	010120			4\$:	MOV	R1,(R0)+	;START PUTTING "PC+2 - HALT"
682	003140	005021				CLR	(R1)+	;IN VECTOR AREA.
683	003142	022021				CMP	(R0)+,(R1)+	;POP POINTERS
684	003144	022700	001000			CMP	#1000,RO	;ALL DONE??
685	003150	001372				BNE	4\$;BR IF NO.
686								
687								
688								
689								
690	003152	012706	001200		.BEGIN:	MOV	#STACK,SP	;SET UP STACK
691	003156	013746	000006			MOV	Q#6,-(SP)	;SAVE LOC 6

;TEST START AND RESTART

F03

CZDMGD.P11 05-JAN-78 11:08

PROGRAM INITIALIZATION AND START UP.

```

692 003162 013746 000004      MOV      2#4,-(SP)      ;SAVE LOC 4
693 003166 005000      CLR      RO           ;START AT 0
694 003170 012737 003234 000004      MOV      #2$,2#4      ;SET UP FOR TIME OUT
695 003176 005037 000006      CLR      2#6         ;TO AUTOSIZE MEMORY
696 003202 005720      6$:      TST      (RO)+       ;CHECK ADDRESS IN RO
697 003204 022700 157776      CMP      #157776,RO   ;IS IT AT LEAST 28K
698 003210 001374      BNE      6$         ;BR IF NO
699 003212 162700 007776      SUB      #7776,RO     ;SAVE 2K FOR MONITORS
700 003216 010037 001304      7$:      MOV      RO,MEMLIM  ;STORE MEMORY LIMIT
701 003222 012637 000004      MOV      (SP)+,2#4   ;RESTORE LOC 4
702 003226 012637 000006      MOV      (SP)+,2#6   ;RESTORE LOC 6
703 003232 000413      BR       10$        ;CONTINUE
704 003234 022626      2$:      CMP      (SP)+,(SP)+ ;ADJUST STACK
705 003236 162700 000004      SUB      #4,RO       ;GET LAST GOOD ADDRESS
706 003242 162700 007776      SUB      #7776,RO    ;SAVE 2K FOR MONITORS
707 003246 022700 030000      CMP      #30000,RO   ;IS IT 8K?
708 003252 001361      BNE      7$         ;BR IF NO
709 003254 012700 037400      MOV      #37400,RO   ;IF 8K DON'T SAVE 2K
710 003260 000756      BR       7$
711 003262 012737 000340 177776 10$:      MOV      #340,PS     ;LOCK OUT INTERRUPTS
712 003270 032737 000004 001236      BIT      #BIT2,STRTSW ;CHECK FOR LOCK ON TEST
713 003276 001411      BEQ      1$         ;BR IF NO LOCK DESIRED.
714 003300 104402 006044      TYPE    ,MLOCK      ;TYPE LOCK SELECTED.
715 003304 012737 000240 003612      MOV      #NOP,TTST   ;ADJUST SCOPE ROUTINE.
716 003312 012737 000240 003614      MOV      #NOP,TTST+2 ;SET UP TO LOCK
717 003320 000406      BR       3$         ;CONTINUE ALONG.
718 003322 013737 003730 003612 1$:      MOV      BRW,TTST    ;PREPARE NORMAL SCOPE ROUTINE
719 003330 013737 003732 003614      MOV      BRX,TTST+2  ;LOCK NOT SELECTED. SET UP FOR NORMAL SCOPE LOOP
720 003336 012737 010062 001214 3$:      MOV      #CYCLE,RETURN ;START AT "CYCLE" FIND WHICH DEVICE TO TEST
721 003344 032737 000002 001236 4$:      BIT      #SW01,STRTSW ;IS TEST NO. SELECTED?
722 003352 001002      BNE      5$         ;BR IF YES
723 003354 104402 005756      TYPE    ,MR         ;TYPE R
724 003360 000177 175630      5$:      JMP      @RETURN     ;START TESTING

```

CZDMGC.P11 05-JAN-78 11:08

END OF PASS ROUTINE

```

725                                     :END OF PASS
726                                     :TYPE NAME OF TEST
727                                     :UPDATE PASS COUNT
728                                     :CHECK FOR EXIT TO ACT-11
729                                     :RESTART TEST
730
731 003364 000005 .EOP: RESET ;MAKE THE WORLD CLEAN AGAIN.
732 003366 005037 CLR LSTERR ;CLEAR LAST ERROR PC
733 003372 105037 CLR8 ERRFLG ;CLEAR ERROR FLAG
734 003376 005237 INC PASCNT ;UPDATE PASS COUNT
735 003402 013777 MOV PASCNT, @DISPLAY ;DISPLAY PASS COUNT
736 003410 104402 TYPE ,MEPASS ;TYPE END PASS
737 003414 104402 TYPE ,MCSR ;TYPE CSR
738 003420 104411 CNVRT ,XCSR ;SHOW IT
739 003424 104402 TYPE ,MVECX ;TYPE VECTOR
740 003430 104411 CNVRT ,XVEC ;SHOW IT
741 003434 104402 TYPE ,MPASSX ;TYPE PASSES
742 003440 104411 CNVRT ,XPASS ;SHOW IT
743 003444 104402 TYPE ,MERRX ;TYPE ERRORS
744 003450 104411 CNVRT ,XERR ;SHOW IT
745 003454 013700 MOV MILK, RO ;GET POINTER TO PASS COUNT
746 003460 013720 MOV PASCNT, (RO)+ ;STORE PASS COUNT FOR THIS DMC11
747 003464 013720 MOV ERRCNT, (RO)+ ;STORE ERROR COUNT FOR THIS DMC11
748 003470 005337 DEC SAVNUM ;ARE ALL DEVICES TESTED?
749 003474 001017 BNE RESTR ;BR IF NO.
750 003476 112737 MOV8 #377, QV.FLG ;SET THE QUICK VERIFY FLAG.
751 003504 013737 MOV DMNUM, SAVNUM ;RESTORE THE COUNT
752 003512 013701 MOV @#42, R1 ;CHECK FOR ACT-11 OR DDP
753 003516 001406 BEQ RESTR ;IF NOT, CONTINUE TESTING
754 003520 000005 RESET ;STOP THE SHOW--CLEAR THE WORLD
755 003522
756 003522 004711 $ENDAD: JSR PC, (R1)
757 003524 000240 NOP
758 003526 000240 NOP
759 003530 000240 NOP
760 003532 000240 NOP
761 003534 012737 010062 001214 RESTR: MOV #CYCLE, RETURN
762 003542 000137 010062 JMP CYCLE
763 003546 000001 XCSR: 1
764 003550 006 002 .BYTE 6, 2
765 003552 001404 DMCSR
766 003554 000001 XVEC: 1
767 003556 004 002 .BYTE 4, 2
768 003560 001374 DMRVEC
769 003562 000001 XPASS: 1
770 003564 006 002 .BYTE 6, 2
771 003566 001230 XERR: 1
772 003570 000001 .BYTE 6, 2
773 003572 006 002 ERRCNT
774 003574 001232
775
776 ;SCOPE LOOP AND INTERATION HANDLER
777 ;-----
778
779 003576 004737 007610 .SCOPE: JSR PC, CKSWR ;CHECK FOR SOFT SWR
780 003602 010016 MOV RO, (SP) ;SAVE RO ON THE STACK

```

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

781 003604 032777 040000 175370
782 003612 001407
783 003614 000437
784 003616 005737 003734
785 003622 001434
786 003624 005037 003734
787 003630 000415
788 003632 032777 004000 175342 1$:
789 003640 001011
790 003642 105737 001327
791 003646 001406
792 003650 005237 001224
793 003654 023737 001224 001222
794 003662 101414
795 003664 105037 001325 2$:
796 003670 005037 001224
797 003674 005037 001220
798 003700 012737 000020 001222
799 003706 013737 001216 001214
800 003714 011600 3$:
801 003716 022626
802 003720 013701 001404
803 003724 000177 175264
804 003730 001407
805 003732 000437
806 003734 000000
807
808
809
810
811 003736 004737 007610
812 003742 032777 001000 175232 .SCOPE:
813 003750 001405
814 003752 005737 001220
815 003756 001402
816 003760 013716 001220 1$:
817 003764 000002
818
819
820
821
822 003766 010546 .TYPE:
823 003770 017605 000002
824 003774 062766 000002 000002
825 004002 005737 010020 4$:
826 004006 001004
827 004010 032777 010000 175164
828 004016 001012
829 004020 105715 1$:
830 004022 100002
831 004024 104402 005672
832 004030 105777 175154 2$:
833 004034 100375
834 004036 112577 175150
835 004042 001357
836 004044 012605 3$:

```

```

BIT #BIT14,DSWR ;"LOOP ON THIS TEST"?
BEQ 1$ ;BR IF NO. (IF LOCK SW01=1; THIS LOC =240)
BR 3$ ;GOTO 3$ (IF LOCK SW01=1; THIS LOC =240)
TST DONE ;WAS TKCSR DONE SET?
BEQ 3$ ;BR IF NO (LOCKED ON TEST)
CLR DONE ;YES, CLEAR FLAG
BR 2$ ;GO TO NEXT TEST
BIT #SW11,DSWR ;DELETE ITERATION? (QUICK PASS)
BNE 2$ ;BR IF YES
TSTB QV.FLG ;HAVE PASSES BEECOMPLETED?
BEQ 2$ ;BR IF QUICK PASS.
INC LPCNT ;UPDATE ITERATION COUNTER
CMP LPCNT,ICOUNT ;ARE ALL ITERATIONS DONE??
BLOS 3$ ;BR IF NOT YET
CLRB EFRFLG ;PREPARE FOR NEW TEST
CLR LPCNT ;START ICOUNTER AT 0
CLR LOCK
MOV #20,ICOUNT ;RESET ITERATIONS
MOV NEXT,RETURN ;GET NEXT TEST
MOV (SP),RO ;POP RO OFF OF THE STACK
POP2SP ;FAKE AN "RTI"
MOV DMCSR,R1 ;R1 CONTAINS BASE DMC ADDRESS
JMP JRETURN ;GO DO THE TEST
BRW: 1407
BRX: 437
DONE: 0

```

```

;CHECK FOR FREEZE ON CURRENT DATA
-----
JSR PC,CKSWR ;CHECK FOR SOFT SWR
BIT #SW09,DSWR ;IS SW09=1(SET)?
BEQ 1$ ;BR IF NOT SET.
TST LOCK
BEQ 1$
MOV LOCK,(SP) ;GOTO THE ADDRESS IN LOCK.
RTI ;GO BACK.

```

```

;TELETYPE OUTPUT ROUTINE
-----
MOV R5, -(SP) ;SAVE R5 ON THE STACK.
MOV @2(SP),R5 ;GET ADDRESS OF MESSAGE.
ADD #2,2(SP) ;POP OVER ADDRESS.
TST SWFLG ;SOFT SWR MESSAGE?
BNE 1$ ;IF YES TYPE IT OUT REGARDLESS OF SW12
BIT #SW12,DSWR ;INHIBIT ALL PRINT OUT??
BNE 3$ ;BR IF NO PRINT OUT WANTED (SW12=1.
TSTB (R5) ;IS NUMBER MINUS? (MSB=1(BIT?))
BPL 2$ ;BR IF NUMBER IS PLUS
TYPE MCRLF ;TYPE A CR/LF!
TSTB JTPCSR ;TTY READY?
BPL 2$ ;BR IF NO.
MOVB (R5)+,@TPDBR ;PRINT CURRENT CHAR.
BNE 4$ ;IF NOT ZERO KEEP PRINTING!
MOV (SP)+,R5 ;ENC OF OUTPUT. RESTORE R5

```


020MGD.F11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

837 004046 000002 RTI ;GO HOME
838 ;-----
839
840 004050 010346 .INSTR: MOV R3, -(SP) ;SAVE R3 ON STACK
841 004052 010446 MOV R4, -(SP) ;SAVE R4 ON STACK
842 004054 017637 000004 004072 MOV @4(SP), .MSG
843 004062 062766 000002 000004 ADD #2, 4(SP)
844 004070 104402 .INST1: TYPE
845 004072 000000 .MSG: 0
846 004074 012704 007504 MOV #INBUF, R4
847 004100 012703 000007 MOV #7, R3
848 004104 105777 175074 1$: TSTB @TKCSR
849 004110 100375 BPL 1$
850 004112 117714 175070 MOVB @TKOBR, (R4)
851 004116 142714 000200 BICB #200, (R4)
852 004122 122427 000015 CMPB (R4)+, #15
853 004126 001417 BEQ INSTR2
854 004130 105777 175054 2$: TSTB @TPCSR
855 004134 100375 BPL 2$
856 004136 017777 175044 175046 MOV @TKOBR, @TPOBR
857 004144 005303 DEC R3
858 004146 001356 BNE 1$
859 004150 012604 MOV (SP)+, R4
860 004152 012603 MOV (SP)+, R3
861 004154 104402 005666 .INSTE: TYPE
862 004160 010346 MOV R3, -(SP)
863 004162 010446 MOV R4, -(SP)
864 004164 000741 BR .INST1
865 004166 012604 INSTR2: MOV (SP)+, R4 ;RESTORE R4
866 004170 012603 MOV (SP)+, R3 ;RESTORE R3
867 004172 000002 RTI
868
869 ;CONVERT ASCII STRING TO OCTAL
870 ;-----
871
872 004174 010546 .PARAM: MOV R5, -(SP)
873 004176 010446 MOV R4, -(SP)
874 004200 016605 000004 MOV 4(SP), R5
875 004204 012537 004364 MOV (R5)+, LOLIM
876 004210 012537 004366 MOV (R5)+, HILIM
877 004214 012537 004370 MOV (R5)+, DEVADR
878 004220 112537 004372 MOVB (R5)+, LOBITS
879 004224 112537 004373 MOVB (R5)+, ADRCNT
880 004230 010566 000004 MOV R5, 4(SP)
881 004234 005005 PARAM1: CLR R5
882 004236 012704 007504 MOV #INBUF, R4
883 004242 122714 000015 CMPB #15, (R4)
884 004246 001420 BEQ PARERR
885 004250 121427 000060 1$: CMPB (R4), #60
886 004254 002415 BLT PARERR
887 004256 121427 000067 CMPB (R4), #67
888 004262 003012 BGT PARERR
889 004264 142714 000060 BICB #60, (R4)
890 004270 152405 BISB (R4)+, R5
891 004272 122714 000015 CMPB #15, (R4)
892 004276 001406 BEG LIMITS
    
```

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

893 004300 006305
 894 004302 006305
 895 004304 006305
 896 004306 000760
 897 004310 104404
 898 004312 000750
 899
 900
 901
 902
 903 004314 020537 004366
 904 004320 101373
 905 004322 020537 004364
 906 004326 103770
 907 004330 133705 004372
 908 004334 001365
 909
 910
 911
 912 004336 013704 004370
 913 004342 010524
 914 004344 062705 000002
 915 004350 105337 004373
 916 004354 001372
 917 004356 012604
 918 004360 012605
 919 004362 000002
 920 004364 000000
 921 004366 000000
 922 004370 000000
 923 004372 000000
 924 004373
 925
 926
 927
 928
 929 004374 016637 000004 001276
 930
 931
 932
 933 004402 010537 001272
 934 004406 010437 001270
 935 004412 010337 001266
 936 004416 010237 001264
 937 004422 010137 001262
 938 004426 010037 001260
 939 004432 000002
 940
 941
 942
 943 004434 013700 001260
 944 004440 013701 001262
 945 004444 013702 001264
 946 004450 013703 001266
 947 004454 013704 001270
 948 004460 013705 001272

```

ASL      R5
ASL      R5
ASL      R5
BR       1$
PARERR:  INSTER
BR       PARAM1

;TEST TO SEE IF NUMBER IS WITHIN LIMITS
-----
LIMITS:  CMP      R5,HILIM
        BHI      PARERR
        CMP      R5,LOLIM
        BLO      PARERR
        BITB     LOBITS,R5
        BNE      PARERR

;STORE NUMBER AT SPECIFIED ADDRESS
1$:     MOV      DEVADR,R4
        MOV      R5,(R4)+
        ADD      #2,R5
        DECB     ADCNT
        BNE     1$
        MOV      (SP)+,R4
        MOV      (SP)+,R5
        RTI

LOLIM:  0
HILIM:  0
DEVADR: 0
LOBITS: 0
ADCNT=LLOBITS+1

;SAVE PC OF TEST THAT FAILED AND R0-R5
-----
.SAVOS: MOV      4(SP),SAVPC      ;SAVE R7 (PC)

;SAVE R0-R5
SVOS:   MOV      R5,SAVR5      ;SAVE R5
        MOV      R4,SAVR4      ;SAVE R4
        MOV      R3,SAVR3      ;SAVE R3
        MOV      R2,SAVR2      ;SAVE R2
        MOV      R1,SAVR1      ;SAVE R1
        MOV      R0,SAVR0      ;SAVE R0
        RTI                    ;LEAVE

;RESTORE R0-R5
.RESOS: MOV      SAVR0,R0      ;RESTORE R0
        MOV      SAVR1,R1      ;RESTORE R1
        MOV      SAVR2,R2      ;RESTORE R2
        MOV      SAVR3,R3      ;RESTORE R3
        MOV      SAVR4,R4      ;RESTORE R4
        MOV      SAVR5,R5      ;RESTORE R5
    
```

K03

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

949 004464 000002          RTI          ;LEAVE
950
951                          ;CONVERT OCTAL NUMBER TO ASCII AND OUTPUT TO TELEPRINTER
952                          ;-----
953
954 004466 104402 005672    .CONVR: TYPE      MCRLF
955 004472 010046    .CNVRT: MOV        R0,-(SP)
956 004474 010146      MOV        R1,-(SP)
957 004476 010346      MOV        R3,-(SP)
958 004500 010446      MOV        R4,-(SP)
959 004502 010546      MOV        R5,-(SP)
960 004504 017601 000012    MOV        @12(SP),R1
961 004510 062766 000002 000012    ADD        #2,12(SP)
962 004516 012137 004710      MOV        (R1)+,WRDCNT
963 004522 112137 004712    1$:      MOVB      (R1)+,CHRCNT
964 004526 112137 004713      MOVB      (R1)+,SPACNT
965 004532 013137 004714      MOV        @2(R1)+,BINWRD
966 004536 122737 000003 004712    CMPB      #3,CHRCNT
967 004544 001003      BNE        2$
968 004546 042737 177400 004714    BIC        #177400,BINWRD
969 004554 013704 004714    2$:      MOV        BINWRD,R4
970 004560 113705 004712      MOVB      CHRCNT,R5
971 004564 012700 001416      MOV        #TEMP,R0
972 004570 010403    3$:      MOV        R4,R3
973 004572 042703 177770      BIC        #177770,R3
974 004576 062703 000060      ADD        #060,R3
975 004602 110320      MOVB      R3,(R0)+
976 004604 000241      CLC
977 004606 006004      ROR        R4
978 004610 000241      CLC
979 004612 006004      ROR        R4
980 004614 000241      CLC
981 004616 006004      ROR        R4
982 004620 005305      DEC        R5
983 004622 001362      BNE        3$
984 004624 012703 007546      MOV        #MDATA,R3
985 004630 114023    4$:      MOVB      -(R0),(R3)+
986 004632 105337 004712      DECB      CHRCNT
987 004636 001374      BNE        4$
988 004640 105737 004713      TSTB      SPACNT
989 004644 001405      BEQ        6$
990 004646 112723 000040    5$:      MOVB      #040,(R3)+
991 004652 105337 004713      DECB      SPACNT
992 004656 001373      BNE        5$
993 004660 105013    6$:      CLRB      (R3)
994 004662 104402 007546      TYPE      ,MDATA
995 004666 005337 004710      DEC        WRDCNT
996 004672 001313      BNE        1$
997 004674 012605      MOV        (SP)+,R5
998 004676 012604      MOV        (SP)+,R4
999 004700 012603      MOV        (SP)+,R3
1000 004702 012601      MOV        (SP)+,R1
1001 004704 012600      MOV        (SP)+,R0
1002 004706 000002          RTI
1003 004710 000000          WRDCNT: 0
1004 004712 000000          CHRCNT: 0

```

CZDMGO.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

SPACNT=CHRCNT+1
BINWRD: 0

1005 004713
1006 004714 000C00
1007
1008
1009
1010
1011
1012
1013
1014 004716 011646
1015 004720 162716 000002
1016 004724 017616 000000
1017 004730 006316
1018 004732 042716 177001
1019 004736 062716 001330
1020 004742 017616 000000
1021 004746 000136
1022
1023
1024
1025
1026 004750 004737 007610
1027 004754 032777 010000 174220
1028 004762 001406
1029 004764 105777 174220
1030 004770 100003
1031 004772 112777 000207 174212
1032 005000 032777 020000 174174
1033 005006 001105
1034 005010 021637 001234
1035 005014 001404
1036 005016 011637 001234
1037 005022 105037 001325
1038 005026 104406
1039 005030 011605
1040 005032 162705 000002
1041 005036 011504
1042 005040 006304
1043 005042 061504
1044 005044 006304
1045 005046 042704 177001
1046 005052 062704 027604
1047 005056 012437 005172
1048 005062 012437 005204
1049 005066 011437 005216
1050 005072 105737 001325
1051 005076 001403
1052 005100 005737 005216
1053 005104 001040
1054 005106 104402 005672
1055 005112 104402 005672
1056 005116 005737 001220
1057 005122 001402
1058 005124 104402 006143
1059 005130 104402 006131
1060 005134 104411 005330

```

; TRAP DISPATCH SERVICE
; ARGUMENT OF TRAP IS EXTRACTED
; AND USED AS OFFSET TO OBTAIN POINTER
; TO SELECTED SUBROUTINE

.TRPSR: MOV      (SP) -(SP)      ; GET PC OF RETURN
        SUB      #2, (SP)      ; =PC OF TRAP
        MOV      @ (SP), (SP)  ; GET TRP
TRPOK:  ASL      (SP)          ; MULTIPLY TRAP ARG BY 2
        BIC      #177001, (SP) ; CLEAR UNWANTED BITS
        ADD      #.TRPTAB, (SP) ; POINTER TO SUBROUTINE ADDRESS
        MOV      @ (SP), (SP)  ; SUBROUTINE ADDRESS
        JMP      @ (SP)+      ; GO TO SUBROUTINE

; ERROR HANDLER
;-----
.HLT:   JSR      PC, CKSWR      ; CHECK FOR SOFT SWR
        BIT      #SW12, @SWR   ; BELL ON ERROR?
        BEQ      XBX          ; BR IF NO BELL
        TSTB    @TPCSR        ; TTY READY.
        BPL      XBX          ; DON'T WAIT IF TTY NOT READY.
        MOVB    #207, @TPDBR   ; PUSH A BELL AT THE TTY.
        BIT      #SW13, @SWR   ; DELETE ERROR PRINT OUT?
        BNE      HALTS        ; BR IF NO PRINT OUT WANTED.
        CMP     (SP), LSTERR    ; WAS THIS ERROR FOUND LAST TIME?
        BEQ     1$            ; BR IF YES
        MOV     (SP), LSTERR    ; RECORD BEING HERE
        CLRB    ERRFLG        ; PREPARE HEADER
        SAVOS   SAVOS         ; SAVE ALL PROC REGISTERS
        MOV     (SP), R5       ; GET THE PC OF ERROR
        SUB     #2, R5         ; GET ADDRESS OF TRAP CALL
        MOV     (R5), R4       ; GET HLT INSTRUCTION
        ASL     R4            ; MULT BY TWO
        ADD     (R5), R4       ; DOUBLE IT
        ASL     R4            ; MULT AGAIN
        BIC     #177001, R4     ; CLEAR JUNK
        ADD     #.ERRTAB, R4    ; GET POINTER
        MOV     (R4)+, ERRMSG   ; GET ERROR MESSAGE
        MOV     (R4)+, DATAHD  ; GET DATA HEADRER
        MOV     (R4), DATABP    ; GET DATA TABLE
        TSTB   ERRFLG         ; TYPE HEADREER
        BEQ     1$            ; BR IF YES
        TST    DATABP         ; DOES DATA TABLE EXIST?
        BNE    TYPDAT         ; BR IF YES.
TYPMSG: TYPE    , MCRLF
        TYPE    , MCRLF
        TST    LOCK
        BEQ    1$
        TYPE    , MASTEK
        TYPE    , MTSTN
1$:      CNVRT    , XTSTN      ; SHOW IT
    
```

M03

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1061	005140	104402	006220			TYPE	,MERRPC	;TYPE PC.
1062	005144	104411	005322			CONVRT	,ERTAB0	;SHOW IT
1063	005150	104402	005672			TYPE	,MCRLF	;GIVE A CR/LF
1064	005154	112737	177777	001325		MOV8	#-1,ERRFLG	;NO MORE HEADER UNLESS NO DATA TABLE.
1065	005162	005737	005172			TST	ERRMSG	;IS THERE AN ERROR MESSAGE?
1066	005166	001402				BEQ	WPKO.FM	;BR IF NO.
1067	005170	104402				TYPE		;TYPE
1068	005172	000000				ERRMSG: 0		;ERROR MESSAGE
1069	005174					WRKO.FM:		
1070	005174	005737	005204			TST	DATAHD	;DATA HEADER?
1071	005200	001402				BEQ	TYPDAT	;BR IF NO
1072	005202	104402				TYPE		;TYPE
1073	005204	000000				DATAHD: 0		;DATA HEADER
1074	005206	005737	005216			TYPDAT: TST	DATABP	;DATA TABLE?
1075	005212	001402				BEQ	RESREG	;BR IF NO.
1076	005214	104410				CONVRT		;SHOW
1077	005216	000000				DATABP: 0		;DATA TABLE
1078	005220	104407				RESREG: RES05		;RESTORE PROC REGISTERS
1079	005222	022737	003522	000042		HALTS: CMP	#SENDAD,2#42	;IF ACT-11 AUTOMATIC MODE, HALT!!
1080	005230	001403				BEQ	1\$	
1081	005232	005777	173744			TST	2\$SWR	;HALT ON ERROR?
1082	005236	100005				bPL	EXITER	;BR IF NO HALT ON ERROR
1083	005240	010046				1\$: PUSHRO		;SAVE RO
1084	005242	016600	000002			MOV	2(SP),RO	;SHOW ERROR PC IN DATA LIGHTS
1085	005246	000000				HALT		;HALT
1086	005250	012600				POPPO		;GET RO
1087	005252	005237	001232			EXITER: INC	ERRCNT	;UPDATE ERROR COUNT
1088	005256	032777	000400	173716		BIT	#SW08,2\$SWR	;GOTO TOP OF TEST?
1089	005264	001007				BNE	1\$;BR IF YES
1090	005266	032777	002000	173706		BIT	#SW10,2\$SWR	;GOTO NEXT TEST?
1091	005274	001411				BEQ	2\$;BR IF NO
1092	005276	013737	001216	001214		MOV	NEXT,RETURN	;SET FOR NEXT TEST
1093	005304	012706	001200			1\$: MOV	#STACK,SP	;RESET SP
1094	005310	013701	001404			MOV	DMCSR,R1	;SET UP R1
1095	005314	000177	173674			JMP	2\$RETURN	;GOTO SPECIFIED TEST
1096	005320	000002				2\$: RTI		;RETURN
1097	005322	000001				ERTAB0: 1		
1098	005324	006	002			.BYTE	6,2	
1099	005326	001276				SAVPC		
1100	005330	000001				XTSTN: 1		
1101	005332	003	002			.BYTE	3,2	
1102	005334	001226				TSTNO		
1103						;ENTER HERE ON POWER FAILURE		
1104						;-----		
1105								
1106								
1107	005336					.PFAIL:		
1108	005336	012737	005350	000024		MOV	#RESTART,24	;SET UP FOR POWER UP TRAP
1109	005344	000000				HALT		;HALT ON POWER DOWN NORMAL
1110	005346	000777				BR	.	
1111						;PROCESSOR WILL TRAP HERE WHEN POWER IS RESTORED		
1112								
1113								
1114	005350					RESTAR:		
1115	005350	012737	005336	000024		MOV	#.PFAIL,24	;SET UP FOR POWER FAILURE
1116	005356	012706	001200			MOV	#STACK,SP	;RESET THE STACK POINTER

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1117 005362 013701 001404      MOV      DMCSR,R1      ;RESTORE R1
1118 005366 005037 001416      CLR      TEMP          ;READY FOR TIMMER
1119 005372 005237 001416      INC      TEMP          ;PLUS ONE TO THE TIMER!
1120 005376 001375          BNE      #-4           ;BR IF MORE TO GO
1121 005400 104402 005675      TYPE    ,MPFAIL       ;TYPE THE MESSAGE
1122 005404 104411 005430      CNVRT   ,PFTAB        ;TELL WHAT TEST TO RETURN TO.
1123 005410 105037 001325      CLRB    ERRFLG        ;START CLEAN
1124 005414 005037 001234      CLR     LSTERR        ;.....
1125 005420 005011          CLR     (R1)          ;CLEAR MAINT BITS
1126 005422 104412          MSTCLR          ;START CLEAN UP OF DEVICE
1127 005424 000177 173564      JMP     @RETURN       ;START DOING THAT TEST AGAIN.
1128 005430 000001          PFTAB: 1
1129 005432 003      002      .BYTE  3,2
1130 005434 001226          TSTNO
1131
1132 005436          .DELAY:
1133 005436 012777 000020 173746      MOV     #20,@DMP04
1134 005444 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1135 005446 121111          121111          ;POKE CLOCK DELAY BIT
1136 005450          1$:
1137 005450 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1138 005452 121224          121224          ;PORT4+IBUS*11
1139 005454 032777 000020 173730      BIT     #BIT4,@DMP04 ;IS CLOCK BIT SET?
1140 005462 001772          BEQ     1$        ;BR IF NO
1141 005464 000002          RTI
1142
1143 005466          .MSTCLR:
1144 005466 152777 000100 173712      BISB   #BIT6,@DMCSRH ;SET MASTER CLEAR
1145 005474 142777 000300 173704      BICB   #BIT6!BIT7,@DMCSRH ;CLEAR MASTER CLEAR AND RUN
1146 005502 000002          RTI              ;RETURN
1147
1148 005504          .ROMCLK:
1149 005504 152777 000002 173674      BISB   #BIT1,@DMCSRH ;SET ROMI
1150 005512 013677 173676      MOV    @2(SP)+,@DMP06 ;LOAD INSTRUCTION IN SEL6
1151 005516 062746 000002          ADD    #2,-(SP)      ;ADJUST STACK
1152 005522 032777 000100 173452      BIT    #SW06,@SWR    ;HALT IF SW06 =1
1153 005530 001401          BEQ    1$           ;BR IF SW06 =0
1154 005532 000000          HALT          ;HALT BEFORE CLOCKING INSTRUCTION
1155 005534 152777 000003 173644      1$:  BISB   #BIT1!BIT0,@DMCSRH ;CLOCK INSTRUCTION
1156 005542 142777 000007 173636      BICB   #BIT2!BIT1!BIT0,@DMCSRH ;CLEAR ROM0, ROMI, STEP
1157 005550 000002          RTI
1158
1159 005552          .DATACLK:
1160 005552 013637 001416      MOV    @2(SP)+,TEMP   ;PUT TICK COUNT IN TEMP
1161 005556 062746 000002          ADD    #2,-(SP)      ;ADJUST STACK
1162 005562 152777 000020 173616      1$:  BISB   #BIT4,@DMCSRH ;SET STEP LU
1163 005570 027777 173610 173606      CMP    @DMCSR,@DMCSR ;WASTE TIME
1164 005576 142777 000020 173602      BICB   #BIT4,@DMCSRH ;CLEAR STEP LU
1165 005604 005337 001416      DEC    TEMP          ;DEC TICK COUNT
1166 005610 001364          BNE    1$           ;BR IF NOT DONE
1167 005612 000002          RTI              ;RETURN
1168 005614 000001          3$:  .BLKW 1
1169
1170 005616          .TIMER:
1171 005616 013637 001416      MOV    @2(SP)+,TEMP   ;MOVE COUNT TO TEMP
1172 005622 062746 000002          ADD    #2,-(SP)      ;ADJUST STACK

```

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1173 005626
1174 005626 104414
1175 005630 021364
1176 005632 032777 000002 173552
1177 005640 001772
1178 005642
1179 005642 104414
1180 005644 021364
1181 005646 032777 000002 173536
1182 005654 001372
1183 005656 005337 001416
1184 005662 001361
1185 005664 000002
1186
1187 005666 020040 000077
(2) 005672 005015 000
(2) 005675 377 053520 020122
(2) 005733 377 047105 020104
(2) 005756 051377 000
(2) 005761 377 047516 042040
(2) 006006 044777 051516 043125
(2) 006032 052377 051505 020124
(2) 006044 046377 041517 020113
(2) 006073 103 051123 020072
(2) 006101 126 041505 020072
(2) 006107 120 051501 042523
(2) 006120 051105 047522 051522
(2) 006131 124 051505 020124
(2) 006143 052 000
(2) 006145 377 042523 020124
(2) 006220 041520 020072 000
(2) 006225 212 020040 020040
(2) 006264 020377 020040 020040
(2) 006323 212 020040 041520
(2) 006375 377 026455 026455
(2) 006451 377 047510 020127
(2) 006511 377 051503 020122
(2) 006527 377 042526 052103
(2) 006550 041377 020122 051120
(2) 006607 377 043111 042040
(2) 006705 377 044127 041511
(2) 007017 377 053523 052111
(2) 007055 377 053523 052111
(2) 007115 377 051511 052040
(2) 007155 377 047516 042040
(2) 007206 005377 053523 036522
(2) 007216 042516 037527 000040
(2) 007224 177777 046504 030503
(2) 007300 042777 050130 041505
(2) 007321 040 042050 041515
(2) 007331 040 045450 041515
(2) 007341 377 046504 030503
(2) 007456 007456
(2) 007456 000005 003
1188 007460 006
1189 007462 001246

```

```

1$: ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021364 ;PORT4+IBUS* REG11
BIT #2, @DMP04 ;IS PGM CLOCK BIT CLEAR?
BEQ 1$ ;BR IF YES

2$: ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021364 ;PORT4+IBUS* REG11
BIT #2, @DMP04 ;IS PGM CLOCK BIT SET?
BNE 2$ ;BR IF YES
TEMP 2$ ;DEC COUNT
DEC 1$ ;DEC COUNT
BNE 1$ ;BR IF NOT DONE
RTI ;RETURN

MQM: .ASCIZ / ?/
MCRLF: .ASCIZ <15><12>
MPFAIL: .ASCIZ <377>/PWR FAILED. RESTART AT TEST /
MEPASS: .ASCIZ <377>/END PASS CZDMGD /
MR: .ASCIZ <377>/R/
MERR2: .ASCIZ <377>/NO DEVICES PRESENT./
MERR3: .ASCIZ <377>/INSUFFICIENT DATA!/
MTSTPC: .ASCIZ <377>/TEST PC-/
MLOCK: .ASCIZ <377>/LOCK ON SELECTED TEST/
MCSR: .ASCIZ /CSR: /
MVEC: .ASCIZ /VEC: /
MPASSX: .ASCIZ /PASSES: /
MERRX: .ASCIZ /ERRORS: /
MTSTN: .ASCIZ /TEST NO: /
MASTEK: .ASCIZ /*/
MNEW: .ASCIZ <377>/SET SWITCH REG TO DMC11'S DESIRED ACTIVE./
MERRPC: .ASCIZ /PC: /
XHEAD: .ASCII <212>/ MAP OF DMC11 STATUS/
.ASCII <377>/-----/
.ASCII <212>/ PC CSR STAT1 STAT2 STAT3/
.ASCII <377>/-----/
NUM: .ASCIZ <377>/HOW MANY DMC11'S TO BE TESTED?/
CSR: .ASCIZ <377>/CSR ADDRESS?/
VEC: .ASCIZ <377>/VECTOR ADDRESS?/
PRIC: .ASCIZ <377>/BR PRIORITY LEVEL? (4,5,6,7)?/
CRAM: .ASCIZ <377>/IF DMC HAS CRAM (M8204) TYPE "Y", IF CROM (M8200) TYPE "N" ?/
MODU: .ASCIZ <377>/WHICH LINE UNIT? IF NONE TYPE "N", IF M8201 TYPE "1", IF M8202 TYP
LINE: .ASCIZ <377>/SWITCH PAC#1 (DDCMP LINE #)?/
BM: .ASCIZ <377>/SWITCH PAC#2 (BM873 BOOT ADD)?/
CONN: .ASCIZ <377>/IS THE LOOP BACK CONNECTOR ON?/
NOACT: .ASCIZ <377>/NO DEVICES ARE SELECTED/
SUMES: .ASCIZ <377><12>/SWR= /
SUMES1: .ASCIZ /NEW? /
CONERR: .ASCIZ <377><377>/DMC11 FOUND AT NON-STANDARD ADDRESS PC: /
CNERR: .ASCIZ <377>/EXPECTED FOUND/
DMCM: .ASCIZ / (DMC) /
KMCM: .ASCIZ / (KMC) /
SPFED: .ASCIZ <377>/DMC11-AR(REMOTE,LOW SPEED) OR DMC11-AL(LOCAL,HIGH SPEED) TYPE "R"
.EVEN
XSTATQ: S
.BYTE 6.3
TEMP1

```

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1190 007464 006 003
 1191 007466 001250
 1192 007470 006 003
 1193 007472 001252
 1194 007474 006 003
 1195 007476 001254
 1196 007500 006 002
 1197 007502 001256
 1198
 1199
 1200
 1201
 1202 007504 000000
 1203 007546
 1204 007546 000000
 1205 007610
 1206
 1207
 1208
 1209
 1210
 1211
 1212 007610 022737 000176 001202
 1213 007616 001077
 1214 007620 105777 171360
 1215 007624 100003
 1216 007626 012737 177777 003734
 1217 007634 022777 000007 171344
 1218 007642 001404
 1219 007644 022777 000207 171334
 1220 007652 001061
 1221 007654 010246
 1222 007656 010346
 1223 007660 010446
 1224 007662 012737 177777 010020
 1225 007670 005002
 1226 007672 012704 177777
 1227 007676 104402 007206
 1228 007702 104411
 1229 007704 010054
 1230 007706 104402 007216
 1231 007712 004737 010022
 1232 007716 022703 000015
 1233 007722 001424
 1234 007724 022703 000012
 1235 007730 001416
 1236 007732 022703 000025
 1237 007736 001754
 1238 007740 022703 000007
 1239 007744 001762
 1240 007746 005004
 1241 007750 042703 177770
 1242 007754 006302
 1243 007756 006302
 1244 007760 006302
 1245 007762 050302

```

      .BYTE 6,3
      TEMP2
      .BYTE 6,3
      TEMP3
      .BYTE 6,3
      TEMP4
      .BYTE 6,2
      TEMPS

      .EVEN

      ;BUFFERS FOR INPUT-OUTPUT

      INBUF. 0
      .=. +40
      MDATA: 0
      .=. +40

      ;ROUTINE USED TO CHANGE SOFTWARE SWITCH
      ;REGISTER USING THE CONSOLE TERMINAL
      -----

      CKSWR:  CMP      #SWREG,SWR      ; IS THE SOFT SWR BEING USED?
              BNE      CKSWR5          ; BR IF NO
              TSTB     @TKCSR          ; IS DONE SET?
              BPL      2$              ; GO ON IF NOT SET
              MOV      #-1,DONE        ; IF DONE SET SET FLAG
              CMP      #7,@TKDBR      ; WAS CTRL G TYPED? (7 BIT ASCII)
              BEQ      1$              ; BR IF YES
              CMP      #207,@TKDBR    ; WAS CTRL G TYPED? (8 BIT ASCII)
              BNE      CKSWR5          ; BR IF NO
              MOV      R2,-(SP)        ; STORE R2
              MOV      R3,-(SP)        ; STORE R3
              MOV      R4,-(SP)        ; STORE R4
              MOV      #-1,SWFLG      ; SET SOFT TYPE OUT FLAG
              CLR      R2              ; CLEAR NEW SWR CONTENTS
              MOV      #-1,R4          ; SET FLAG TO ALL ONES
              TYPE     ,SWMES          ; TYPE "SWR="
              CNVRT    SOFTSW          ; TYPE OUT PRESENT CONTENTS
              SOFTSW  ; OF SOFT SWITCH REGISTER
              TYPE     "NEW"          ; TYPE "NEW"
              JSR     PC,INCHAR        ; GET RESPONSE
              CMP      #15,R3          ; WAS IT A CR?
              BEQ      5$              ; BR IF YES
              CMP      #12,R3          ; WAS IT A LF?
              BEQ      4$              ; BR IF YES
              CMP      #25,R3          ; WAS IT CTRL U?
              BEQ      CKSWR1          ; BR IF YES(START OVER)
              CMP      #7,R3           ; IF CNTL G GET NEXT CHAR
              BEQ      CKSWR4          ;
              CLR      R4              ; IT MUST BE A DIGIT SO CLR FLAG
              BIC      #177770,R3      ; ONLY 0-7 ARE LEGAL SO MASK OFF BITS
              ASL      R2              ; SHIFT R2 3 TIMES
              ASL      R2
              ASL      R2
              BIS      R3,R2          ; ADD LAST DIGIT

```

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1246	007764	000752			BR	CKSWR4		:GET NEXT CHARACTER
1247	007766	012766	002002	000006	4\$: MOV	*.START,6(SP)		:LF WAS TYPED SO GO TO START
1248	007774	005704			5\$: TST	R4		:IS FLAG CLEAR?
1249	007776	001002			BNE	6\$:IF NOT DON'T CHANGE SOFT SWR
1250	010000	010277	171176		MOV	R2,@SWR		:IF YES THEN WRITE NEW CONTENTS TO SOFT SWR
1251	010004	005037	010020		6\$: CLR	SWFLG		:CLEAR TYPEOUT FLAG
1252	010010	012604			MOV	(SP)+,R4		:RESTORE R4
1253	010012	012603			MOV	(SP)+,R3		:RESTORE R3
1254	010014	012602			MOV	(SP)+,R2		:RESTORE R2
1255	010016	000207			CKSWR5: RTS	PC		:RETURN
1256								
1257	010020	000000			SWFLG: 0			
1258								
1259	010022	105777	171156		INCHAR: TSTB	@TKCSR		
1260	010026	100375			BPL	-4		
1261	010030	017703	171152		MOV	@TKOBR,R3		
1262	010034	105777	171150		TSTB	@TPCSR		
1263	010040	100375			BPL	-4		
1264	010042	010377	171144		MOV	R3,@TPOBR		
1265	010046	042703	000200		BIC	#BIT7,R3		
1266	010052	000207			RTS	PC		
1267								
1268	010054	000001			SOFTSW: 1			
1269	010056	006	002		:BYTE	6.2		
1270	010060	000176			:SWREG			

E04

```

1271
1272
1273
1274
1275
1276
1277
1278
1279
1280 010062 005737 001306
1281 010066 001004
1282 010070 104402 007155
1283 010074 000000
1284 010076 000776
1285 010100 000241
1286 010102 006137 001316
1287 010106 005537 001316
1288 010112 062737 000004 001322
1289 010120 062737 000010 001320
1290 010126 022737 001700 001320
1291 010134 001006
1292 010136 012737 001500 001320
1293 010144 012737 001702 001322
1294 010152 033737 001316 001306
1295 010160 001747
1296 010162 013700 001320
1297 010166 013702 001322
1298 010172 012037 001404
1299 010176 011037 001374
1300 010202 042737 177000 001374
1301 010210 012037 001366
1302 010214 012037 001370
1303 010220 012037 001372
1304 010224 012237 001230
1305 010230 012237 001232
1306 010234 012700 000002
1307 010240 013737 001404 001406
1308 010246 005237 001406
1309 010252 013737 001406 001410
1310 010260 005237 001410
1311 010264 013737 001410 001412
1312 010272 060037 001412
1313 010276 013737 001412 001414
1314 010304 060037 001414
1315
1316 010310 013737 001374 001376
1317 010316 060037 001376
1318 010322 013737 001376 001400
1319 010330 060037 001400
1320 010334 013737 001400 001402
1321 010342 060037 001402
1322
1323 010346 032737 000002 001236
1324 010354 001450
1325 010356
1326 010356 005737 000042

```

CYCLE: TST DMACTV ; ARE ANY DMC11'S TO BE TESTED?
BNE 1\$; BR IF OK
TYPE ,NOACT ; NO DMC11'S SELECTED!!
HALT ; STOP THE SHOW.
BR .-2 ; DISQUALIFY CONT. SW.
1\$: CLC ; CLEAR PROC. CARRY BIT.
ROL RUN ; UPDATE POINTER
ADC RUN ; CATCH CARRY FROM RUN
ADD #4,MILK ; UPDATE POINTER
ADD #10,CREAM ; UPDATE ADDRESS POINTER.
CMP #DM.MAP+200,CREAM
BNE 2\$; KEEP GOING; NOT ALL TESTED FOR.
MOV #DM.MAP,CREAM ; RESET ADDRESS POINTER.
MOV #CNT.MAP,MILK ; RESET PASS COUNT POINTER
2\$: BIT RUN,DMACTV ; IS THIS ONE ACTIVE?
BEQ 1\$; BR IF NO
MOV CREAM,R0 ; GET ADDRESS POINTER
MOV MILK,R2 ; GET PASS COUNT POINTER
MOV (R0)+,DMCSR ; LOAD SYSTEM CTRL. REG
MOV (R0),DMRVEC ; LOAD VECTOR
BIC #177000,DMRVEC ; CLEAR UNWANTED BITS
MOV (R0)+,STAT1 ; LOAD STAT1
MOV (R0)+,STAT2 ; LOAD STAT2
MOV (R0)+,STAT3 ; LOAD STAT3
MOV (R2)+,PASCNT ; LOAD PASS COUNT
MOV (R2)+,ERRCNT ; LOAD ERROR COUNT
MOV #2,R0 ; SAVE CORE THIS WAY!
MOV DMCSR,DMCSRH
INC DMCSRH
MOV DMCSRH,DMCTL
INC DMCTL
MOV DMCTL,DMP04
ADD R0,DMP04
MOV DMP04,DMP06
ADD R0,DMP06
MOV DMRVEC,DMRLVL ; PTY LVL
ADD R0,DMRLVL
MOV DMRLVL,DMTVEC ; TX VEC
ADD R0,DMTVEC
MOV DMTVEC,DMTLVL ; TX LVL
ADD R0,DMTLVL
4\$: BIT #SW01,STRTSW ; IS TEST NO. SELECTED
BEQ 7\$; BR IF NO
TST #42 ; RUNNING IN AUTO MODE?

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1327 010362 001045          BNE      7$          ;BR IF YES
1328 010364 104402 005672  TYPE      ,MCRLF
1329 010370 104403          INSTR
1330 010372 006131          MTSTN
1331 010374 104405          PARAM
1332 010376 000001          1
1333 010400 001000          1000
1334 010402 001226          TSTNO
1335 010404          000      .BYTE
1336 010405          001      .BYTE
1337 010406 012700 022324  MOV      #TST1,R0
1338 010412 022710 5$:      CMP      (PC)+,(R0)      ;CMP FIRST WORD TO 12737
1339 010414 012737          MOV      (PC)+,2(PC)+
1340 010416 001020          BNE      6$          ;BR IF NOT SAME
1341 010420 023760 001226 000002  CMP      TSTNO,2(R0)      ;DOES TSTNO MATCH?
1342 010426 001014          BNE      6$          ;BR IF NO
1343 010430 022760 001226 000004  CMP      #TSTNO,4(R0)      ;IS LAST WORD OK?
1344 010436 001010          BNE      6$          ;BR IF NO
1345 010440 010037 001214  MOV      R0,RETURN      ;IT IS A LEGAL TEST SO DO IT
1346 010444 104402 005756  TYPE      MR
1347 010450 042737 000002 001236  BIC      #SW01,STRTSW
1348 010456 000412          BR      8$
1349 010460 005720 6$:      TST      (R0)+          ;POP R0
1350 010462 020027 026474  CMP      R0,#TLAST+10      ;AT END YET?
1351 010466 001351          BNE      5$          ;BR IF NO
1352 010470 104402 005666  TYPE      MQM          ;YES ILLEGAL TEST NO.
1353 010474 000730          BR      4$          ;TRY AGAIN
1354
1355 010476 012737 022324 001214 7$:      MOV      #TST1,RETURN      ;PREPARE RETURN ADDRESS
1356 010504 013701 001404 8$:      MOV      DMCSR,R1          ;R1 = BASE DMC11 ADDRESS
1357 010510 000177 170500  JMP      @RETURN          ;GO START TESTING.
1358
1359
1360          ;ROUTINE USED TO "AUTO SIZE" THE DMC11
1361          ;CSR AND VECTOR.
1362          ;NOTE: THE CSR MAY BE ANY WHERE IN THE FLOATING
1363          ;ADDRESS RANGE (160000:164000)
1364          ;AND THE VECTOR MAY BE ANY WHERE IN THE
1365          ;FLOATING VECTOR RANGE (300:770)
1366          ;
1367          ;
1368          AUTO.SIZE:
1369 010514 000005          RESET
1370 010516 012702 001500  CSRMAP: MOV      #DM.MAP,R2      ;INSURE A BUS INIT.
1371 010522 005022 1$:      CLR      (R2)+          ;LOAD MAP POINTER.
1372 010524 022702 001700  CMP      #DM.END,R2      ;ZERO ENTIRE MAP
1373 010530 001374          BNE      1$          ;ALL DONE?
1374 010532 005037 001310  CLR      DMNUM          ;BR IF NO
1375 010536 012702 001500  MOV      #DM.MAP,R2      ;SET OCTAL NUMBER OF DMC11'S TO 0
1376 010542 005037 001306  CLR      DMACTV          ;R2 POINTS TO DMC MAP
1377 010546 032737 000001 001236  BIT      #SW00,STRTSW      ;CLEAR ACTIVE
1378 010554 001002          BNE      +6          ;QUESTIONS?
1379 010556 000137 011254  JMP      7$          ;BR IF YES
1380 010562 012737 000001 001256  MOV      #1,TEMPS          ;IF NO SKIP QUESTIONS
1381 010570 104403          INSTR
1382 010572 006451          NUM

```

02DMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1383	010574	104405				PARAM			
1384	010576	000001				1			
1385	010600	000020				16.			
1386	010602	001252				TEMP3			
1387	010604	000				.BYTE	0		
1388	010605	001				.BYTE	1		
1389	010606	013737	001252	001310	12\$:	MOV	TEMP3,DMNUM	;DMNUM = HOW MANY	
1390	010614	104402	005672			TYPE	,MCRLF		
1391	010620	104410				CONVRT		;TYPE WHICH DMC IS BEING DONE	
1392	010622	012004				WHICH		;TEMPS IS WHICH DMC	
1393	010624	005237	001256			INC	TEMPS		
1394	010630	104403				INSTR			
1395	010632	006511				CSR			
1396	010634	104405				PARAM			
1397	010636	160000				160000			
1398	010640	164000				164000			
1399	010642	001254				TEMP4			
1400	010644	000				.BYTE	0		
1401	010645	001				.BYTE	1		
1402	010646	013722	001254			MOV	TEMP4,(R2)+	;STORE CSR IN MAP	
1403	010652	104403				INSTR			
1404	010654	006527				VEC			
1405	010656	104405				PARAM			
1406	010660	000000				0			
1407	010662	000776				776			
1408	010664	001254				TEMP4			
1409	010666	000				.BYTE	0		
1410	010667	001				.BYTE	1		
1411	010670	013712	001254			MOV	TEMP4,(R2)	;STORE VECTOR IN MAP	
1412	010674	104402			10\$:	TYPE			
1413	010676	006550				PRI0		;ASK WHAT BR LEVEL	
1414	010700	004737	012270			JSR	PC,INTTY	;GET RESPONSE	
1415	010704	022703	000024			CMP	#24,R3		
1416	010710	101014				BHI	50\$;BR IF LESS THAN 4	
1417	010712	022703	000027			CMP	#27,R3		
1418	010716	103411				BLO	50\$;BR IF GREATER THAN 7	
1419	010720	012704	000011			MOV	#11,R4	;R4 = NUMBER OF SHIFTS	
1420	010724	006303				ASL	R3	;SHIFT R3 LEFT	
1421	010726	005304				DEC	R4	;DEC SHIFT COUNT	
1422	010730	001375				BNE	-4	;BR IF NOT DONE	
1423	010732	042703	170777			BIC	#170777,R3	;BIC UNWANTED BITS	
1424	010736	050312				BIS	R3,(R2)	;PUT BR LEVEL IN STATUS MAP	
1425	010740	000403				BR	8\$;CONTINUE	
1426	010742	104402			50\$:	TYPE		;RESPONSE IS OUT OF LIMITS	
1427	010744	005666				MQM		;TRY AGAIN	
1428	010746	000752				BR	10\$		
1429	010750	104402			8\$:	TYPE			
1430	010752	000607				CRAM		;DOES DMC HAVE CRAM?	
1431	010754	004737	012270			JSR	PC,INTTY	;GET REPLY	
1432	010760	022703	000131			CMP	#131,R3		
1433	010764	001427				BEQ	9\$;YES	
1434	010766	022703	000116			CMP	#116,R3	;NO	
1435	010772	001403				BEQ	40\$;NOT A Y OR N	
1436	010774	104402				TYPE			
1437	010776	005666				MQM		;TYPE ""	
1438	011000	000763				BR	8\$;ASK AGAIN	

02DMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1439 011002 104402
1440 011004 007341
1441 011006 004737 012270
1442 011012 022703 000122
1443 011016 001414
1444 011020 022703 000114
1445 011024 001403
1446 011026 104402
1447 011030 005666
1448 011032 000763
1449 011034 052762 000002 000004 41$:
1450 011042 000402
1451 011044 052712 100000 9$:
1452 011050 104402 16$:
1453 011052 006705
1454 011054 004737 012270
1455 011060 022703 000021
1456 011064 001417
1457 011066 022703 000022
1458 011072 001412
1459 011074 022703 000116
1460 011100 001403
1461 011102 104402
1462 011104 005666
1463 011106 000760
1464 011110 052722 010000 32$:
1465 011114 022222
1466 011116 000447
1467 011120 052712 020000 31$:
1468 011124 104402 30$:
1469 011126 007115
1470 011130 004737 012270
1471 011134 022703 000131
1472 011140 001406
1473 011142 022703 000116
1474 011146 001406
1475 011150 104402
1476 011152 005666
1477 011154 000763
1478 011156 052722 040000 17$:
1479 011162 000402
1480 011164 042722 040000 18$:
1481 011170 19$:
1482 011170 104403
1483 011172 007017
1484 011174 104405
1485 011176 000000
1486 011200 000377
1487 011202 001254
1488 011204 000
1489 011205 001
1490 011206 113722 001254
1491 011212 104403
1492 011214 007055
1493 011216 104405
1494 011220 000000

```

```

40$: TYPE
SPEED ;DMC11-AR OR DMC11-AL?
JSR PC,INTTY ;GET RESPONSE
CMP #122,R3 ;IS IT R
BEQ 16$ ;BR IF REMOTE
CMP #114,R3 ;IS IT L
BEQ 41$ ;BR IF LOCAL

40$ ;TRY AGAIN
41$: BIS #BIT1,4(R2) ;SET BIT1 IN STAT3
BR 16$ ;CONTINUE
9$: BIS #BIT15,(R2) ;SET BIT 15 IF CRAM
16$: TYPE
MODU ;ASK WHICH LINE UNIT
JSR PC,INTTY ;GET REPLY
CMP #21,R3 ;"1"
BEQ 30$ ;"2"
CMP #22,R3 ;"2"
BEQ 31$ ;"N"
CMP #116,R3 ;"N"
BEQ 32$

16$ ; IF NOT A 1.2 OR N TYPE ""
BR ;TRY AGIAN
32$: BIS #BIT12,(R2)+ ;SET BIT 12 IN STAT2 IF NO LU
CMP (R2)+,(R2)+ ;POP OVER STAT2 AND STAT3
BR 33$
31$: BIS #BIT13,(R2) ;SET BIT 13 IN STAT2 IF M8202
30$: TYPE
CONN ;ASK IF LOOP-BACK IS ON
JSR PC,INTTY ;GET REPLY
CMP #131,R3 ;Y
BEQ 17$ ;N
CMP #116,R3 ;N
BEQ 18$

30$ ; IF NOT Y OR N TYPE ""
BR ;TRY AGAIN
17$: BIS #BIT14,(R2)+ ;TURNAROUND IS CONNECTED
BR 19$
18$: BIC #BIT14,(R2)+ ;NO TURNAROUND
19$:

INSTR
LINE
PARAM
0
377
TEMP4
.BYTE 0
.BYTE 1
MOV8 TEMP4,(R2)+ ;STORE SWITCH PAC IN MAP
INSTR
BM
PARAM
0

```

0ZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1495	011222	000377			377		
1496	011224	001254			TEMP4		
1497	011226	000			.BYTE	0	
1498	011227	001			.BYTE	1	
1499	011230	113722	001254		MOVB	TEMP4,(R2)+	;STORE SWITCH PAC IN MAP
1500	011234	005722			TST	(R2)+	;POP OVER STAT3
1501	011236	005337	001252	33\$:	DEC	TEMP3	;DEC DMC COUNT
1502	011242	001402			BEQ	34\$;BR IF DONE
1503	011244	000137	010614		JMP	12\$;JUMP IF NOT
1504	011250	000137	011704	34\$:	JMP	13\$;CONTINUE
1505	011254	012701	160000	7\$:	MOV	#160000,R1	;SET FOR FIRST ADDRESS TO BE TESTED
1506	011260	012737	011776	000004	MOV	#6\$,2#4	;SET FOR NON-EXISTANT DEVICE TIME OUT
1507	011266	005011		2\$:	CLR	(R1)	;CLEAR SEL0
1508	011270	005711			TST	(R1)	;IF DMC11 DMCSR S/B 0
1509	011272	001172			BNE	3\$;IF NO DEV; TRAP TO 4. IF NO BIT 8 THEN NO DMC11
1510	011274	005061	000006		CLR	6(R1)	;CLEAR SEL6
1511	011300	005761	000006		TST	6(R1)	;IF DMC11 THEN DMRIC S/B =0!
1512	011304	001165			BNE	3\$;BR IF NOT DMC11
1513	011306	012711	002000		MOV	#BIT10,(R1)	;SET ROM0
1514	011312	005061	000004		CLR	4(R1)	;CLEAR SEL4
1515	011316	012761	125252	000006	MOV	#125252,6(R1)	;WRITE THIS TO SEL6
1516	011324	052711	020000		BIS	#BIT13,(R1)	;WRITE IT!
1517	011330	022761	125252	000004	CMP	#125252,4(R1)	;WAS IT WRITTEN?
1518	011336	001004			BNE	21\$;IF NO IT IS NOT CRAM
1519	011340	052762	100000	000002	BIS	#BIT15,2(R2)	;SET BIT15 IF CRAM
1520	011346	000431			BR	22\$	
1521	011350	012711	001000	21\$:	MOV	#BIT9,(R1)	;SET ROM1
1522	011354	012761	100430	000006	MOV	#100430,6(R1)	;PUT INSTRUCTION IN SEL6
1523	011362	012711	001400		MOV	#BIT9!BIT8,(R1)	;CLOCK INSTRUCTION (MICRO PROC PC TO 0)
1524	011366	012711	002000		MOV	#BIT10,(R1)	;SET ROM0
1525	011372	022761	016472	000006	CMP	#016472,6(R1)	;IS IT LOCAL CROM?
1526	011400	001411			BEQ	23\$;BR IF YES
1527	011402	022761	016461	000006	CMP	#016461,6(R1)	;IS IT REMOTE CROM?
1528	011410	001410			BEQ	22\$;BR IF YES
1529	011412	022761	177777	000006	CMP	#-1,6(R1)	;NO CROM?
1530	011420	001404			BEQ	22\$;BR IF YES
1531	011422	000516			BR	3\$;NOT A DMC
1532	011424	052762	000002	000006	23\$:	BIS	#BIT1,6(R2)
1533					15\$:	MOV	R1,(R2)+
1534	011432	010122			22\$:	MOV	#BIT9,(R1)
1535	011434	012711	001000		15\$:	CLR	4(R1)
1536	011440	005061	000004			MOV	#122113,6(R1)
1537	011444	012761	122113	000006		BIS	#BIT8,(R1)
1538	011452	052711	000400			MOV	#021264,6(R1)
1539	011456	012761	021264	000006		BIS	#BIT8,(R1)
1540	011464	052711	000400			CMPB	#377,4(R1)
1541	011470	122761	000377	000004		BNE	+.10
1542	011476	001003				BIS	#BIT12,(R2)
1543	011500	052712	010000			BR	20\$
1544	011504	000436				BIT	#BIT1,4(R1)
1545	011506	032761	000002	000004		BEQ	+.10
1546	011514	001403				BIS	#BIT13!BIT14,(R2)
1547	011516	052712	060000			BR	20\$
1548	011522	000427				BIT	#BIT3,4(R1)
1549	011524	032761	000010	000004		BNE	20\$
1550	011532	001023					;BR IF M8201 NO CONNECTOR (CN LINE)

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1551	011534	012761	000100	000004		MOV	#BIT6,4(R1)	;	LOAD PORT4
1552	011542	012761	122113	000006		MOV	#122113,6(R1)	;	LOAD INSTRUCTION
1553	011550	052711	000400			BIS	#BIT8,(R1)	;	CLOCK INSTRUCTION(SET DTR)
1554	011554	012761	021264	000006		MOV	#021264,6(R1)	;	LOAD INSTRUCTION
1555	011562	052711	000400			BIS	#BIT8,(R1)	;	CLOCK INSTRUCTION(READ MODEM REG)
1556	011566	032761	000010	000004		BIT	#BIT3,4(R1)	;	IS MRDY SET NOW?
1557	011574	001402				BEQ	20\$;	BR IF NO CONNECTOR
1558	011576	052712	040000			BIS	#BIT14,(R2)	;	SET STATUS BIT FOR CONNECTOR
1559	011602	005722			20\$:	TST	(R2)+	;	POP POINTER
1560	011604	012761	021324	000006		MOV	#021324,6(R1)	;	PUT INSTRUCTION IN PORT6
1561	011612	012711	001400			MOV	#BIT9:BIT8,(R1)	;	PORT4←LU 15
1562	011616	156122	000004			BISB	4(R1),(R2)+	;	STORE DDCMP LINE # IN TABLE
1563	011622	012761	021344	000006		MOV	#021344,6(R1)	;	PORT6←INSTRUCTION
1564	011630	012711	001400			MOV	#BIT8:BIT9,(R1)	;	CLOCK INSTR.
1565	011634	156122	000004			BISB	4(R1),(R2)+	;	STORE BM873 ADD IN TABLE
1566	011640	005722				TST	(R2)+	;	POP OVER STAT3
1567	011642	005011				CLR	(R1)	;	CLEAR ROMI
1568	011644	005237	001310			INC	DMNUM	;	UPDATE DEVICE COUNTER
1569	011650	022737	000020	001310		CMP	#20,DMNUM	;	ARE MAX. NO. OF DEV FOUND?
1570	011656	001412				BEQ	13\$;	YES DON'T LOOK FOR ANY MORE.
1571	011660	005011			3\$:	CLR	(R1)	;	CLEAR BIT 10
1572	011662	005061	000006			CLR	6(R1)	;	CLEAR SEL 6
1573	011666	062701	000010		14\$:	ADD	#10,R1	;	UPDATE CSR POINTER ADDRESS
1574	011672	022701	164000			CMP	#164000,R1		
1575	011676	001402				BEQ	13\$;	BR IF DONE
1576	011700	000137	011266			JMP	2\$;	JUMP IF NOT
1577	011704	005037	001306		13\$:	CLR	DMACTV		
1578	011710	005737	001310			TST	DMNUM	;	WERE ANY DMC11'S FOUND AT ALL?
1579	011714	001423				BEQ	5\$;	ERROR AUTO SIZER FOUND NO DMC11'S IN THIS SYS.
1580	011716	013701	001310			MOV	DMNUM,R1		
1581	011722	010137	001314			MOV	R1,SAVNUM	;	SAVE NUMBER OF DEVICES
1582	011726	000241			4\$:	CLC			
1583	011730	006137	001306			ROL	DMACTV	;	GENERATE ACTIVE REGISTER OF DEVICES.
1584	011734	005237	001306			INC	DMACTV	;	SET THE BIT
1585	011740	005301				DEC	R1		
1586	011742	001371				BNE	4\$;	BR IF MORE TO GENERATE
1587	011744	012737	000006	000004		MOV	#6,2#4	;	RESTORE TRAP VECTOR
1588	011752	013737	001306	001312		MOV	DMACTV,SAVACT	;	SAVE ACTIVE REGISTER
1589	011760	000137	012012			JMP	VECMAP	;	GO FIND THE VECTOR NOW.
1590	011764	104402	005761		5\$:	TYPE	MERR2	;	NOTIFY OPR THAT NO DMC11'S FOUND.
1591	011770	005000				CLR	RD	;	MAKE DATA LIGHTS ZERO
1592	011772	000000				HALT		;	STOP THE SHOW
1593	011774	000776				BR	.-2	;	DISABLE CONT. SW.
1594	011776	012716	011666		6\$:	MOV	#14\$,(SP)	;	ENTERED BY NON-EXISTANT TIME-OUT.
1595	012002	000002				RTI		;	RETURN TO MAINSTREAM
1596									
1597	012004	000001			WHICH:	1			
1598	012006	002	002			.BYTE	2,2		
1599	012010	001256				TEMPS			
1600									
1601	012012	032737	000001	001236	VECMAP:	BIT	#SW00,STRTSW		
1602	012020	001114				BNE	5\$		
1603	012022	012737	000340	000022		MOV	#340,2#22	;	SET IOT TRAP PRIO TO 7
1604	012030	012737	012204	000020		MOV	:4\$,2#20	;	SET IOT TRAP VECTOR
1605	012036	012702	001500			MOV	#DM.MAP,R2	;	SET SOFTWARE POINTER
1606	012042	012700	000300			MOV	#300,RO	;	FLOATING VECTORS START HERE.

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1607	012046	012701	000302		MOV	#302,R1	;PC OF IOT INSTR.	
1608	012052	010120		1\$:	MOV	R1,(R0)+	;START FILLING VECTOR AREA	
1609	012054	012721	000004		MOV	#4,(R1)+	;WITH +2; IOT	
1610	012060	022021			CMP	(R0)+(R1)+	;ADD 2 TO R0 +R1	
1611	012062	020127	001000		CMP	R1,#1000		
1612	012066	101771			BLOS	1\$;BR IF MORE TO FILL	
1613	012070	013737	001306	001246	MOV	DMACTV,TEMP1	;STORE TEMPORALLY	
1614	012076	006037	001246	2\$:	ROR	TEMP1	;BRING OUT A BIT	
1615	012102	103063			BCC	5\$;BR IF ALL DONE	
1616	012104	012704	000012		MOV	#12,R4	;R4 IS INDEX REGISTER	
1617	012110	016437	012254	177776	MOV	BRLVL(R4),PS	;SET PS TO 7	
1618	012115	011201			MOV	(R2),R1		
1619	012120	012761	000200	000004	MOV	#200,4(R1)		
1620	012126	012711	001000		MOV	#BIT9,(R1)	;SET ROMI	
1621	012132	012761	121111	000006	MOV	#121111,6(R1)	;PUT INSTRUCTION IN PORT6	
1622	012140	012711	001400		MOV	#BIT9:BIT8,(R1)	;FORCE AN INTERRUPT	
1623	012144	105200		7\$:	INCB	R0	;STALL	
1624	012146	001376			BNE	.-2	;FOR TIME TO INTERRUPT	
1625	012150	162704	000002		SUB	#2,R4	;GET NEXT LOWEST PS LEVEL	
1626	012154	001404			BEQ	6\$;BR IF R4 = 0	
1627	012156	016437	012254	177776	MOV	BRLVL(R4),PS	;MOVE NEXT LOWER LEVEL IN PS	
1628	012164	000767			BR	7\$;BR TO DELAY	
1629	012166	052762	005300	000002	6\$:	BIS	#5300,2(R2)	;NO INTERRUPT ASSUME 300 AT LEVEL 5 AND FIX DMC11 LATER
1630	012174	005011		3\$:	CLR	(R1)	;CLEAR ROMI	
1631	012176	062702	000010		ADD	#10,R2	;POP SOFTWARE POINTER	
1632	012202	000735			BR	2\$;KEEP GOING	
1633	012204	051662	000002		4\$:	BIS	(SP),2(R2)	;GET VECTOR ADDRESS
1634	012210	042762	000007	000002	BIC	#7,2(R2)	;CLEAR JUNK	
1635	012216	016405	012256		MOV	BRLVL+2(R4),R5	;GET BR LEVEL OF DMC11	
1636	012222	006305			ASL	R5	;SHIFT LEVEL 4 PLACES	
1637	012224	006305			ASL	R5	;TO THE LEFT FOR THE	
1638	012226	006305			ASL	R5	;STATUS TABLE	
1639	012230	006305			ASL	R5		
1640	012232	042705	170777		BIC	#170777,R5	;CLEAR UNWANTED BITS	
1641	012236	050562	000002		BIS	R5,2(R2)	;PUT BR LEVEL IN STATUS TABLE	
1642	012242	022626			CMP	(SP)+(SP)+	;POP IOT JUNK OFF STACK	
1643	012244	012716	012174		MOV	#3\$,(SP)	;SET FOR RETURN	
1644	012250	000002			RTI			
1645	012252	000207		5\$:	RTS	PC	;ALL DONE WITH "AUTO SIZING"	
1646								
1647	012254	000000		BRLVL:	0	;LEVEL 0		
1648	012256	000000			0	;LEVEL 0		
1649	012260	000200			200	;LEVEL 4		
1650	012262	000240			240	;LEVEL 5		
1651	012264	000300			300	;LEVEL 6		
1652	012266	000340			340	;LEVEL 7		
1653								
1654								
1655	012270	105777	166710	INTTY:	TSTB	@TKCSR	;WAIT FOR DONE	
1656	012274	100375			BPL	.-4		
1657	012276	017703	166704		MOV	@TKDBR,R3	;PUT CHAR IN R3	
1658	012302	105777	166702		TSTB	@TPCSR	;WAIT UNTIL PRINTER IS READY	
1659	012306	100375			BPL	.-4		
1660	012310	010377	166676		MOV	R3,@TPDBR	;ECHO CHAR	
1661	012314	042703	000240		BIC	#BIT7:BIT5,R3	;MASK OFF LOWER CASE	
1662	012320	000207			RTS	PC	;RETURN	

LO4

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

PAGE: 0050CZ
SEQ 0050

1663
1664
!S65 012322 000000
1666
1667 012324

ROMMAP: 0 ; POINTER TO HI OR LO SPEED MICRO-CODE
LOMAP: ; LOW SPEED (REMOTE) MICRO-CODE

DMCNEW.MAC 26-OCT-77 13:33 TABLE OF CONTENTS

7	MACRO DEFINITIONS
9	REVISION 00
10	FEBRUARY 25, 1975
11	
12	REVISION 01
13	MARCH 18, 1975
14	NEW CSR BOARD CHANGES
15	
16	HARVEY M. SCHLESINGER
18	COPYRIGHT 1975 DIGITAL EQUIPMENT CORPORATION
70	MICRO INSTRUCTION DEFINITIONS
71	BRANCH INSTRUCTIONS
114	INDEXED BRANCH INSTRUCTIONS
150	MOVE INSTRUCTIONS
258	INPUT/OUTPUT ASSIGNMENTS
311	PROTOCOL DEPENDANT MACROS
354	DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT
386	VERSION 00A FEBRUARY 26, 1975
387	
388	HARVEY M. SCHLESINGER
389	
390	
391	VERSION 00B MARCH 17, 1975
392	CSR AND MICROPROCESSOR CHANGES
393	
394	VERSION 00C NOVEMBER 6, 1975
395	RETRANSMISSION CHANGES
396	
397	VERSION 00D DECEMBER 3, 1975
398	TRANSMIT DONE CHANGES
399	
400	VERSION 01A 6-MAY-77
401	ACKNOWLEDGEMENT CHANGES - ACK A MESSAGE EVEN IF NO DATA TO RETURN
402	
403	VERSION 01B 01-DEC-77
404	MANY CHANGES
405	
406	THE LATEST MODIFICATIONS WERE ADDED ON:
407	01-DEC-77
409	MICROPROCESSOR MAIN MEMORY ASSIGNMENTS
474	SCRATCH PAD ASSIGNMENTS
509	INIT--INITIALIZATION ROUTINE
561	IDLE--PROGRAM IDLE LOOP
594	NIDLE2---NO CSR ACTIVITY STATE
673	BASSRV---- BASE SERVICE ROUTINE
716	OUTINT---SET UP OUTPUT INTERRUPT [RDYO]
764	OUTWAI---WAIT FOR RDYO TO GO AWAY
775	CTLSRV--CNTL I SERVICE
799	TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE
818	RBASRV--RECEIVE BUFFER ADDRESS SERVICE
876	RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER
904	RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD
938	RCVC--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD, SELECT AND FINAL
956	RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES
984	RCVE--ROUTINE TO HANDLE N FIELD OF NUMBERED MESSAGE
996	RCVF--ROUTINE TO IGNORE ADDRESS

DMCGAL.MAC

18-JAN-78 16:05

TABLE OF CONTENTS

1004	RCVG--ROUTINE TO IGNORE CRC1
1009	RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYPES
1070	RCVK01--ROUTINE TO HANDLE FIRST BYTE ODD RECEIVE
1078	RCVK0--PROCESS ODD CHARACTER
1102	RCVKE--HANDLE EVEN BYTES
1154	RCVI--STORE UNNUMBERED MESSAGE TYPE
1159	RCVJ--ROUTINE TO HANDLE SUBTYPE FIELD, SELECT AND FINAL
1172	RCVR--UNNUMBERED MESSAGE RESPONSE FIELD
1181	RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD
1187	RCVL--PROCESS CRC3
1206	RCVM--PROCESS CRC4--END OF DATA MESSAGE
1225	EM2--PROCESS RLD MESSAGE
1252	SELQSY--ROUTINE TO CHECK SELECT AND QSYNC AND DIDDLE LINE STATUS WORD
1288	TMTA--FIRST CHARACTER OF HEADER
1355	TMTB--OUTPUT FIRST CHAR OF COUNT
1398	TMTC--OUTPUT SECOND CHAR OF COUNT
1408	TMTD--RESPONSE FIELD-NUMBERED MESSAGE
1416	TMTE--NUMBER FIELD--NUMBERED MESSAGE
1423	TMTF--NUMBERED MSG ADDRESS FIELD
1435	TF1--NUMBERED MSG HEADER EOM
1445	TMTH--ROUTINE TO OUTPUT DATA CHARACTERS
1502	TMTI--SEND UNNUMBERED TYPE FIELD
1508	TMTJ--SEND SUB-TYPE FIELD
1512	TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG)
1520	TMTL--UNNUMB MSG NUMBER FIELD
1537	TMTM--UNNUMB MSG--STATION ADDRESS
1556	TIMSRV--TIMEOUT ROUTINE--SENDS REP
1630	SNDACK--ROUTINE TO SEND AN ACK
1679	REP HANDLER
1734	NXMERR ---NON EXISTANT MEMORY HANDLER
1776	START HANDLER

5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

.TITLE DMC-11 MICROPROCESSOR INSTRUCTIONS
.SBTTL MACRO DEFINITIONS
:
.SBTTL REVISION 00
.SBTTL FEBRUARY 25, 1975
.SBTTL
.SBTTL REVISION 01
.SBTTL MARCH 18, 1975
.SBTTL NEW CSR BOARD CHANGES
.SBTTL
.SBTTL HARVEY M. SCHLESINGER
:
.SBTTL COPYRIGHT 1975 DIGITAL EQUIPMENT CORPORATION
:

DMCNEW.MAC

26-OCT-77 13:33

COPYRIGHT 1975 DIGITAL EQUIPMENT CORPORATION

21 000000
22 000000
23 100000
24 020000
25 000000
26 040000
27 060000
28 060000
29 060000
30 060000
31 060000
32 010000
33 014000
34 000400
35 001000
36 001400
37 002000
38 002400
39 003000
40 003400
41 000200
42 000220
43 000240
44 000260
45 000300
46 000320
47 000340
48 000360
49 000000
50 000020
51 000040
52 000060
53 000100
54 000120
55 000140
56 000160
57 004000
58 010000
59 014000
60 001000
61 001400
62 000400
63 002000
64 002400
65 003000
66 003400

```

NEW-0
; MICROPROCESSOR INSTRUCTION WORD DEFINITIONS
MOVE=0 ; OPCODE MOVE
JUMP=100000 ; OPCODE JUMP
IBUS=20000 ; SOURCE IBUS
IMM=0 ; SOURCE IMMEDIATE
MEMX=40000 ; SOURCE MEMORY
BRX=60000 ; SOURCE BR
BR=60000 ; SOURCE BR

DP=60000 ; SOURCE BR
LDMAR=10000 ; MA-LOAD MAR LO
INCMAR=14000 ; MA-INCREMENT MAR
WRITEBR=400 ; DEST-WRITE BR
WROUTX=1000 ; DEST-EXTENDED IBUS
SHFTBR=1400 ; DEST-SHIFT BR LEFT
WROUT=2000 ; DEST-WRITE OUTPUT
WRMEM=2400 ; DEST-WRITE MEMORY
SPX=3000 ; DEST-WRITE SP
SPBRX=3400 ; DEST-WRITE SP AND BR

; FUNCTIONS
SELA=200 ; FUNCTION-SELECT A
SELB=220 ; FUNCTION-SELECT B
AORNB=240 ; FUNCTION-A OR NOT B
AANDB=260 ; FUNCTION A AND B
AORB=300 ; FUNCTION-A OR B
AXORB=320 ; FUNCTION A XOR B
SUB=340 ; SUBTRACT
SUBTC=360 ; FUNCTION- TWOS COMPLEMENT SUBTRACT
ADD=0 ; ADD A+B
ADDC=20 ; A+B+CARRY
SUBC=40 ; A-B-C
INCA=60 ; INCREMENT A
AC=100 ; A PLUS CARRY
AA=120 ; A PLUS A
AAC=140 ; A PLUS A PLUS C
DECA=160 ; DECREMENT A

; END FUNCTIONS
PAGE1=4000
PAGE2=10000
PAGE3=14000

CCOND=1000 ; CONDITION C
ZCOND=1400 ; CONDITION Z
ALCOND=400 ; ALWAYS
BROCON=2000 ; CONDITION BRO
BR1CON=2400 ; CONDITION BR1
BR4CON=3000 ; CONDITION BR4
BR7CON=3400 ; CONDITION BR7
    
```


DMCNEW.MAC

26-OCT-77 13:33

INPUT/OUTPUT ASSIGNMENTS

```

258      .SBTTL  INPUT/OUTPUT ASSIGNMENTS
259      ;IBUS  ASSIGNMENTS
260      100000      INCON=0+100000      ; IN CONTROL CSR
261      100020      MAIN=20+100000     ; MAINTAINENCE REGISTER
262      100040      OCON=40+100000     ; OUT CONTROL CSR
263      100060      UBADDR=60+100000   ; UNUSED
264      100100      PORT1=100+100000   ; CSR4
265      100120      PORT2=120+100000   ; CSR5
266      100140      PORT3=140+100000   ; CSR6
267      100160      PORT4=160+100000   ; CSR7
268      100200      NPR=200+100000     ; NPR CONTROL
269      100220      UBBR=220+100000    ; BR(INTERRUPT)CONTROL
270      000000      INDAT1=0           ; INPUT DATA LOW BYTE
271      000020      INDAT2=20          ; INPUT DATA HIGH BYTE
272      000140      IOBA1=140          ; OUTPUT BA LOW BYTE
273      000160      IOBA2=160          ; OUTPUT BA HIGH BYTE
274      000100      IIBA1=100          ; INPUT BA LOW BYTE
275      000120      IIBA2=120          ; INPUT BA HIGH BYTE
276      000200      RCVDAT=200         ; RECEIVE DATA
277      000220      TMTCON=220         ; TMR CONTROL
278      000240      RCVCON=240         ; RCVR CONTROL
279      000260      MODEM=260          ; MODEM CONTROL
280      000300      SYNREG=300          ; SYN REGISTER
281      000320      LNOSW=320          ; LINE NUMBER SWITCH
282      000340      BM873=340          ; BM873 ADDRESS
283      000360      LUMAIN=360         ; LINE UNIT MAINTAINENCE
284      ;OBUS  ASSIGNMENTS
285      ;EXTENDED OBUS
286      000000      OINCON=0           ; IN CONTROL CSR
287      000001      OMAIN=1            ; MAINT
288      000002      OCON=2             ; OUT CONTROL CSR
289      000003      OUBADD=3           ; UNUSED
290      000004      OPORT1=4          ; CSR4
291      000005      OPORT2=5          ; CSR5
292      000006      OPORT3=6          ; CSR6
293      000007      OPORT4=7          ; CSR7
294      000010      ONPR=10           ; NPR CONTROL
295      000011      OBR=11            ; BR CONTROL
296      ;UNEXTENDED OBUS
297      000000      OINDAT1=0          ; OUTPUT OF
298      000002      OUTDA1=2           ; OUTPUT DATA LOW BYTE
299      000003      OUTDA2=3           ; OUTPUT DATA HIGH BYTE
300      000006      OBA1=6            ; OUTPUT BA LOW BYTE
301      000007      OBA2=7            ; OUTPUT BA HIGH BYTE
302      000004      IBA1=4            ; INPUT BA LOW BYTE
303      000005      IBA2=5            ; INPUT BA HIGH BYTE
304      000010      TMTDAT=10         ; TMR DATA
305      000011      OTMTCO=11         ; TMR CONTROL
306      000012      ORCVCO=12         ; RCVR CONTROL
307      000013      OMODEM=13        ; MODEM CONTROL
308      000014      SYNC=14           ; SYN REGISTER
309      000017      OLUMAN=17         ; LINE UNIT MAINT.

```

DMCNEW.MAC

26-OCT-77 13:33

PROTOCOL DEPENDANT MACROS

.SBTTL PROTOCOL DEPENDANT MACROS

.....

177777

MICPC=177777 ;INIT MICRO PC

311
318
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
995

G05

LOW1.MAC

22-AUG-77 17:54

DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT

PAGE: 0058DM
SEQ 0058

354
355
356

000000
000000

.SBTTL DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT
LOW=0
SLOW=0


```

409 .SBTTL MICROPROCESSOR MAIN MEMORY ASSIGNMENTS
410 ;ALLOCATION OF MICROPROCESSOR MAIN MEMORY
411 000000 NAKSR=0 ;NAKS RECD--DYNAMIC
412 000001 NAKST=NAKSR+1 ;NAKS TMTD--DYNAMIC
413 000002 REPSR=NAKST+1 ;REPS RECD--DYNAMIC
414 000003 REPST=REPSR+1 ;REPS TMTD--DYNAMIC
415 000006 NP=REPST+3 ;CONSTANT 0
416 000007 NTLR=NP+1 ;NAKS-MSG NO BUFFERS CUMUL.
417 000010 NHDR=NTLR+1 ;NAKS-MSG HEADER BAD
418 000011 NDATA=NHDR+1 ;NAKS-DATA BAD
419 000012 NTLR=NTLR+1 ;NAK SENT --NO BUFFERS
420 000013 NHDS=NTLS+1 ;NAK SENT BAD HEADER
421 000014 NDATS=NHDS+1 ;NAK SENT BAD DATA
422 000015 REPCS=NDATS+1 ;REPS SENT CUMUL
423 000016 REPCR=REPCS+1 ;REPS RECD CUMUL
424 000017 BASE=REPCR+1 ;CORE TABLE BASE ADDRESS
425 000022 SRC=BASE+3 ;START OF INPUT CHAIN--NEXT RECV DONE
426 000023 ERC=SRC+1 ;END OF INPUT CHAIN
427 000024 RCL1=ERC+1 ;RECEIVE LINK #1
428 000031 RCL2=RCL1+5 ;" " #2
429 000036 RCL3=RCL2+5 ;" " #3
430 000043 RCL4=RCL3+5
431 000050 RCL5=RCL4+5
432 000055 RCL6=RCL5+5
433 000062 RCL7=RCL6+5
434 000067 STC=RCL7+5 ;START OF OUTPUT CHAIN---NEXT TMT DONE
435 000070 ETC=STC+1 ;END OF TRANSMIT CHAIN
436 000071 TML1=ETC+1 ;TRANSMIT LINK #1
437 000077 TML2=TML1+6 ;" " #2
438 000105 TML3=TML2+6 ;" " #3
439 000113 TML4=TML3+6
440 000121 TML5=TML4+6
441 000127 TML6=TML5+6
442 000135 TML7=TML6+6
443 000143 TML8=TML7+6
444 000151 T=TML8+6 ;TYPE FIELD
445 000152 ST=T+1 ;SUBTYPE FIELD
446 000153 ISP17=ST+1 ;MSG ACKED IMAGE
447 000154 IMG10=ISP17+1 ;IMAGE OF BIT 1 OF SP10
448 000155 IMG11=IMG10+1 ;IMAGE OF SP11
449 000156 IMG12=IMG11+1 ;IMAGE OF SP12
450 000157 IMG14=IMG12+1 ;IMAGE OF SP14
451 000160 IMG16=IMG14+1 ;IMAGE OF SP16
452 000161 IMG17=IMG16+1 ;IMAGE OF SP17
453 000162 TYPTAB=IMG17+1 ;TYPE TABLE---
454 ;72 TYPE TABLE REP
455 ;73 " " NAK
456 000164 TYPSTT=TYPTAB+2 ;74 " " START
457 ;75 " " STACK
458
459
460 000167 BC=TYPSTT+3 ;RECEIVE BYTE COUNT
461 000171 ISP11=BC+2 ;SP11 IMAGE
462 000172 ISP12=ISP11+1 ;SP12 IMAGE
463 000173 INCONS=ISP12+1 ;IN CONTROL CSR IMAGE
464 000174 RTHRS=INCONS+1 ;RCV THRESHOLD LINK

```


J05

DMCGAL.MAC

18-JAN-78 16:05

MICROPROCESSOR MAIN MEMORY ASSIGNMENTS

PAGE: 0061DM
SEQ 0061

;ALL LOCATIONS FROM 200 ON ARE NOT WRITTEN OUT DURING A TABLE UPDATE

465
466
467 000210
468 000211
469 000240
470 000241
471 000242
472 000400

TABST=210 ;TABLE UPDATE STATE
PRST=TABST+1 ;PORT STATE
NXTINT=240 ;NEXT INTERRUPT POSITION
NXTSP=NXTINT+1 ;END OF INTERRUPT CHAIN
INTSTK=NXTSF+1 ;STACK OF INTERRUPTS
MMEND=400 ;MAIN MEMORY END

DMCGAL.MAC

18-JAN-78 16:05

SCRATCH PAD ASSIGNMENTS

```

474          .SBTTL  SCRATCH PAD ASSIGNMENTS
475          000000  ;SP0---SCRATCH REGISTER
476          000001  ;SP1---PORT STATUS WORD
477          ;BIT ASSIGNMENTS
478          ;BIT0--INIT MODE
479          ;BIT1--NO BUFFER ASSIGNED IN BOOT MODE
480          ;BIT2--UNUSED
481          ;BIT3--OLE RECEIVED WHILE NOT IN MAINT MODE
482          ;BIT4--INTERRUPT PENDING
483          ;BIT6--DISCONNECT ERROR
484          ;BIT7--BOOT MODE
485          000002  SP2=2  ;SP2---TRANSMIT STATE POINTER
486          000003  SP3=3  ;SP3---RECEIVE STATE POINTER
487          000004  SP4=4  ;SP4---END RECV ADDRESS
488          000005  SP5=5  ;SP5---END RECEIVE ADDRESS
489          000006  SP6=6  ;SP6---END TRANSMIT ADDRESS
490          000007  SP7=7  ;SP7---END TRANSMIT ADDRESS
491          000010  SP10=10 ;SP10---LINE STATUS WORD
492          ;BIT ASSIGNMENTS
493          ;BIT0--UNNUMB PENDING
494          ;BIT1--MESSAGE IN PROGRESS
495          ;BIT2--LINE HAS GONE IDLE
496          ;BIT3--START RECEIVED
497          ;BIT4--CLEAR ACTIVE ON END
498          ;BIT5--START MODE
499          ;BIT6--HALF DUPLEX
500          ;BIT7--OK TO SEND
501          000011  SP11=11 ;SP11---R FIELD
502          000012  SP12=12 ;SP12---N FIELD
503          000013  SP13=13 ;SP13---TYPE
504          000014  SP14=14 ;SP14---TIMER ENTRY---NUMBER OF ONE SECOND TICKS
505          000015  SP15=15 ;SP15---RECEIVE LINK IMAGE
506          000016  SP16=16 ;SP16---POINTER TO TMT LINK COPY IN MAIN MEM
507          000017  SP17=17 ;SP17---LAST MESSAGE ACKNOWLEDGED

```

DMCGAL.MAC 18-JAN-78 16:05

INIT--INITIALIZATION ROUTINE

```

509          .SBTTL  INIT--INITIALIZATION ROUTINE
510          ;ZEROS MAIN MEMORY
511          ;LOOPS WAITING FOR RECEIVE DATA(BOOT?)
512          ;OR FOR RQI TO BE SET
513          ;WILL ACCEPT ONLY BASE FORMAT. ALL OTHERS WILL RETURN A PROCEDURE ERROR
514          ;
515          ;AT INITIALIZATION --- THE HARDWARE CLEARS THE BR AND MAR
516          =27654
517 027654 027654  INIT:  SP      BR SELB, SPO          ;CLEAR SPO
(1)      000000  MICPC=MICPC+1
(1) 027654 063220  <MOVE!SPX!BR!SELB!SPO>
518 027656 027656  SP      BR SELB, SP3          ;PAGE ONE TRANSFER ADDRESS
(1)      000001  MICPC=MICPC+1
(1) 027656 063223  <MOVE!SPX!BR!SELB!SP3>
519 027660 027660  SP      BR SELB, SP17        ;CLEAR SP17
(1)      000002  MICPC=MICPC+1
(1) 027660 063237  <MOVE!SPX!BR!SELB!SP17>
520 027662 027662  SP      BR SELB, SP12        ;CLEAR SP12
(1)      000003  MICPC=MICPC+1
(1) 027662 063232  <MOVE!SPX!BR!SELB!SP12>
521 027664 027664  OUT     BF <SELA!OINCON>      ;ZERO THE IN CONTROL CSR
(1)      000004  MICPC=MICPC+1
(1) 027664 061200  <MOVE!WROUTX!BR!<SELA!OINCON>>
522 027666 027666  OUT     BR <SELA!OOCON>      ;ZERO THE OUT CONTROL CSR
(1)      000005  MICPC=MICPC+1
(1) 027666 061202  <MOVE!WROUTX!BR!<SELA!OOCON>>
523 027670 027670  SP      IMM 370, SP10        ;WRITE 5 ONE BITS TO THE HIGH ORDER
(1)      000006  MICPC=MICPC+1
(1) 027670 003370  <MOVE!SPX!IMM!370!SP10>
524          ;BITS OF SP10
525 027672 027672  S$:  SP      BR AA, SP10        ;SHIFT SP10 LEFT SETTING CARRY THE
(1)      000007  MICPC=MICPC+1
(1) 027672 063130  <MOVE!SPX!BR!AA!SP10>
526          ;FIRST 5 TIMES THRU THE LOOP
527 027674 027674  MEMINC BR ADDC!SP3        ;WRITE A ONE TO THE FIRST 5 MEMORY
(1)      000010  MICPC=MICPC+1
(1) 027674 076423  <MOVE!WRMEM!INCMAR!BR!<ADDC!SP3>>
528          ;LOCATIONS AND ZERO THE REST
529 027676 027676  SP      BR INCA, SPO        ;INCREMENT COUNTER
(1)      000011  MICPC=MICPC+1
(1) 027676 063060  <MOVE!SPX!BR!INCA!SPO>
530 027700 027700  Z      10$          ;ALL DONE
(1)      000012  MICPC=MICPC+1
(1) 027700 101414  <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
531 027702 027702  ALWAYS S$          ;KEEP GOING
(1)      000013  MICPC=MICPC+1
(1) 027702 100407  <JUMP!ALCOND!<5$-INIT&3000*4>!<5$-INIT&777/2>>
532 027704 027704  SPBR   IMM 1, SP1        ;WRITE A 1 TO THE BR AND SP1
(1)      000014  MICPC=MICPC+1
(1) 027704 003401  <MOVE!SPBRX!IMM!1!SP1>
533 027706 027706  SP      BR SELB, SP11        ;WRITE A 1 TO SP11
(1)      000015  MICPC=MICPC+1
(1) 027706 063231  <MOVE!SPX!BR!SELB!SP11>
534 027710 027710  LDMA   IMM TYPTAB          ;POINT MAR TO TYPE TABLE
(1)      000016  MICPC=MICPC+1
(1) 027710 010162  <MOVE!LDMAR!IMM!<TYPTAB&377>>

```

DMCGAL.MAC 18-JAN-78 16:05

INIT--INITIALIZATION ROUTINE

```

535 027712 000017 BRWRT IMM,226 ;WRITE SYNC TO MEMORY
(1) (1) 027712 000626 <MOVE!WRTEBR!IMM!<226>>
536 027714 000020 OUTPUT BR,SELB,SYNC ;LOAD THE SYNC REGISTER
(1) (1) 027714 062234 <MOVE!WROUT!BR!<SELB!SYNC>>
537 027716 000021 MEMINC IMM,3 ;REP
(1) (1) 027716 016403 <MOVE!WRMEM!INCMAR!IMM!<3>>
538 027720 000022 MEMINC IMM,2 ;NAK
(1) (1) 027720 016402 <MOVE!WRMEM!INCMAR!IMM!<2>>
539 027722 000023 MEMINC IMM,6 ;START
(1) (1) 027722 016406 <MOVE!WRMEM!INCMAR!IMM!<6>>
540 027724 000024 MEMINC IMM,7 ;STACK
(1) (1) 027724 016407 <MOVE!WRMEM!INCMAR!IMM!<7>>
541 027726 000025 MEMINC IMM,1 ;ACK
(1) (1) 027726 016401 <MOVE!WRMEM!INCMAR!IMM!<1>>
542 027730 000026 LDMA IMM,TABST ;POINT TO TABLE UPDATE STATE
(1) (1) 027730 010210 <MOVE!LDMAR!IMM!<TABST&377>>
543 027732 000027 PSTATI I3 ;INITIALIZE IT
(1) (2) 027732 016455 MEMINC IMM,<<I3-INIT&777/2>>
(2) (2) 027732 016455 <MOVE!WRMEM!INCMAR!IMM!<<I3-INIT&777/2>>>
544 027734 000027 PSTATI NIDLE2 ;INITIALIZE PORT STATUS
(1) (2) 027734 016461 MEMINC IMM,<<NIDLE2-INIT&777/2>>
(2) (2) 027734 016461 <MOVE!WRMEM!INCMAR!IMM!<<NIDLE2-INIT&777/2>>>
545 027736 000031 LDMA IMM,STC ;LOAD ADDRESS OF LAST TMT CHAIN
(1) (1) 027736 010067 <MOVE!LDMAR!IMM!<STC&377>>
546 027740 000032 MEMINC IMM,TML1 ;STORE ADDRESS OF FIRST TMT LINK
(1) (1) 027740 016471 <MOVE!WRMEM!INCMAR!IMM!<TML1>>
547 027742 000033 MEM IMM,TML1
(1) (1) 027742 002471 <MOVE!WRMEM!IMM!<TML1>>
548 027744 000034 SP MEMX,SELB,SP16 ;INITIALIZE LAST XMIT POINTER
(1) (1) 027744 043236 <MOVE!SPX!MEMX!SELB!SP16>
549 027746 000035 LDMA IMM,SRC ;LOAD ADDRESS OF LAST RECV CHAIN
(1) (1) 027746 010022 <MOVE!LDMAR!IMM!<SRC&377>>
550 027750 000036 MEMINC IMM,RCL1 ;SET UP ADDRESS OF FIRST RECV LINK
(1) (1) 027750 016424 <MOVE!WRMEM!INCMAR!IMM!<RCL1>>
551 027752 000037 MEM IMM,RCL1
(1) (1) 027752 002424 <MOVE!WRMEM!IMM!<RCL1>>
552 027754 000040 SP MEMX,SELB,SP15
(1) (1) 027754 043235 <MOVE!SPX!MEMX!SELB!SP15>

```

DMCGAL.MAC 18-JAN-78 16:05

INIT--INITIALIZATION ROUTINE

```

553 027756          LDMA   IMM,NXTINT          ;ADDRESS OF NEXT INTERRUPT POINTER TO MAR
      (1)           000041
      (1) 027756    010240          MICPC=MICPC+1
      (1)           000042          <MOVE!LDMAR!IMM!<NXTINT&377>>
554 027760          MEMINC IMM,INTSTK          ;INITIALIZE NEXT INTERRUPT POINTER
      (1)           000043          MICPC=MICPC+1
      (1) 027760    016642          <MOVE!WRMEM!INCMAR!IMM!<INTSTK>>
555 027762          MEM    IMM,INTSTK          ;INITIALIZE INSERTION POINTER
      (1)           000044          MICPC=MICPC+1
      (1) 027762    002642          <MOVE!WRMEM!IMM!<INTSTK>>
556 027764          BRWRT  IMM,200            ;MASK TO SET RUN BIT
      (1)           000044          MICPC=MICPC+1
      (1) 027764    000600          <MOVE!WRTEBR!IMM!<200>>
557 027766          OUT    BR,<SELB!OMAIN>      ;WRITE THE RUN BIT TO MAINT CSR
      (1)           000045          MICPC=MICPC+1
      (1) 027766    061221          <MOVE!WROUTX!BR!<SELB!OMAIN>>
558                    ;FALL INTO IDLE LOOP
559 027770          ALWAYS TEOM2
      (1)           000046          MICPC=MICPC+1
      (1) 027770    110642          <JUMP!ALCOND!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>

```

DMCGAL.MAC 18-JAN-78 16:05

IDLE--PROGRAM IDLE LOOP

```

561          .SBTTL  IDLE--PROGRAM IDLE LOOP
562          ;PROGRAM IDLE LOOP
563          ;DISPATCHES TO APPROPRIATE SERVICE ROUTINES
564          ;USES STATE POINTERS FOR TMT,RCV,CSR ACTIVITY
565          ;
566 027772  IDLE:  BRWRT  IBUS,TMTCON
567 027772          MICPC=MICPC+1
(1)          000047          <MOVE!WRTBR!IBUS!<TMTCON>>
(1) 027772 020620          .BR4  DP,SELA,<2!PAGE2>
568 027774          MICPC=MICPC+1
(1)          000050          <JUMP!BR4CON!DP!SELA!2!PAGE2>
(1) 027774 173202  I1:  BRWRT  IBUS,RCVCON          ;READ LINE UNIT RECEIVE CONTROL WORD
(1)          000051          MICPC=MICPC+1
(1) 027776 020640          <MOVE!WRTBR!IBUS!<RCVCON>>
570 030000          .BR4  BR,SELA,SP3!PAGE1          ;BRANCH BASED UPON RCV STATE
(1)          000052          MICPC=MICPC+1
(1) 030000 167203          <JUMP!BR4CON!BR!SELA!SP3!PAGE1>
571 030002  I2:  LDMA  IMM,TABST          ;POINT TO TABLE UPDATE STATE
(1)          000053          MICPC=MICPC+1
(1) 030002 010210          <MOVE!LDMAR!IMM!<TABST&377>>
572 030004          .ALWAY MEMX,SELB,0
(1)          000054          MICPC=MICPC+1
(1) 030004 140620          <JUMP!ALCOND!MEMX!SELB!0>
573 030006  I3:  IDLE0: SPBR  IBUS,UBBR,SPO          ;TIMER EXPIRES?
582 030006          MICPC=MICPC+1
(1)          000055          <MOVE!SPBRX!IBUS!UBBR!SPO>
(1) 030006 123620          BR4  TIMSRV
583 030010          MICPC=MICPC+1
(1)          000056          <JUMP!BR4CON!<TIMSRV-INIT&3000*4>!<TIMSRV-INIT&777/2>>
(1) 030010 113246  IDLE1: LDMA  IMM,PRTST          ;ADDRESS PORT STATE
584 030012          MICPC=MICPC+1
(1)          000057          <MOVE!LDMAR!IMM!<PRTST&377>>
(1) 030012 010211          .ALWAY MEMX,SELB,0          ;INDEX
590 030012          MICPC=MICPC+1
(1)          000060          <JUMP!ALCOND!MEMX!SELB!0>
(1) 030014 140620

```

DMCGAL.MAC 18-JAN-78 16:05

NIDLE2---NO CSR ACTIVITY STATE

```

594
595 030016 000061 .SBTTL NIDLE2---NO CSR ACTIVITY STATE
(1) (1) 030016 060601 NIDLE2: BRWRITE BR, SELA!SP1 ;READ PORT STATUS WORD
600 030020 000062 MICPC=MICPC+1
(1) (1) 030020 103214 <MOVE!WRTEBR!BR!<SELA!SP1>>
602 030022 000063 BR4 OUTINT
(1) (1) 030022 123400 MICPC=MICPC+1
603 030024 000064 <JUMP!BR4CON!<OUTINT-INIT&3000*4>!<OUTINT-INIT&777/2>>
(1) (1) 030024 001620 SPBR IBUS, INCON, SPO ;READ INPUT CONTROL CSR
604 030026 000065 <MOVE!SPBRX!IBUS!INCON!SPO>
(1) (1) 030026 117044 BRSHFT ;SHIFT IT RIGHT
605 MICPC=MICPC+1
606 030030 000066 <MOVE!SHFTBR!WRTEBR!SELB>
610 030030 023660 BR4 INWAT1 ;IF RQI SET -- BRANCH
612 030032 000067 MICPC=MICPC+1
613 030032 060520 <MOVE!WRTEBR!BR!<AA!SPO>>
614 030034 000070 BR4 SETDSR ;IF DSR SET, CLEAR FLAG
615 030036 000071 MICPC=MICPC+1
616 030040 000072 <JUMP!BR4CON!<SETDSR-INIT&3000*4>!<SETDSR-INIT&777/2>>
617 030042 000073 BRWRITE BR, AA!SPO ;READ LINE STATUS WORD
618 030044 000074 MICPC=MICPC+1
619 030046 000075 <MOVE!WRTEBR!BR!<SELA!SP10>>
620 030050 000076 BRSHFT ;START MODE
621 030052 000077 MICPC=MICPC+1
622 030054 000100 <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
623 030056 000101 BRWRITE BR, AA!SP1 ;READ PORT STATUS WORD
624 030060 000102 MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<AA!SP1>>
        BR1 IDLE ;INIT MODE
        MICPC=MICPC+1
        <JUMP!BR1CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        BR7 IDLE ;DISCONNECT ERROR ALREADY SENT
        MICPC=MICPC+1
        <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        SPBR IBUS, MAIN, SPO ;READ THE MAINT REGISTER
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!MAIN!SPO>
        BRWRITE BR, ADD!SPO ;SHIFT LEFT
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<ADD!SPO>>
        BR4 IDLE ;LU LOOP -- EXIT
        MICPC=MICPC+1
        <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        BRWRITE IMM, 100 ;WRITE DISCONNECT ERROR

```


CMC GAL.MAC 18-JAN-78 16:05

NIDLE2---NO CSR ACTIVITY STATE

```

(1) 000102
(1) 030060 000500
625 030062
(1) 000103
(1) 030062 063301
626 030064
(1) 000104
(1) 030064 114657
627 030066
(1) 000105
(1) 030066 000677
628 030070
(1) 000106
(1) 030070 100646
    
```

```

MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<100>>
SP BR AORB,SP1 ;FLAG ERROR RECORDED
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP1>
ALWAYS ERXX ;MAKE A CONTROL OUT
MICPC=MICPC+1
<JUMP!ALCOND!<ERRXX-INIT&3000*4>!<ERRXX-INIT&777/2>>
SETDSR: BRWTE IMM,277 ;CLEAR DISCONNECT ERROR FLAG
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<277>>
ALWAYS CLRIDL
MICPC=MICPC+1
<JUMP!ALCOND!<CLRIDL-INIT&3000*4>!<CLRIDL-INIT&777/2>>
    
```

E06

DMCGAL.MAC 18-JAN-78 16:05

NIDLE2---NO CSR ACTIVITY STATE

```

635 030072 000107
(1) 030072 123400
636 030074 000110
(1) 030074 103512
637 030076 000111
(1) 030076 114444
638 030100 000112
(1) 030100 060520
639 030102 000113
(1) 030102 103606
640 030104 000114
(1) 030104 120400
641 030106 000115
(1) 030106 001620
643 030110 000116
(1) 030110 103066
650 030112 000117
(1) 030112 123400
651 030114 000120
(1) 030114 102532
652 030116 000121
(2) 030116 002655
653 030120 000122
(1) 030120 102127
654 030122 000123
(1) 030122 001620
655 030124 000124
(2) 030124 002723
656 030126 000125
(1) 030126 102527
657 030130 000126
(2) 030130 002702
658 030132 000127
(1) 030132 060601
659 030134 000130
(1)

```

```

IEIWAT: SPBR IBUS INCON, SPO ;READ INPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!INCON!SPO>
BR7 10$ ;RDYI STILL SET
MICPC=MICPC+1
<JUMP!BR7CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
ALWAYS INWAT1 ;GOT CLEARED BY 11 RE-SET IT
MICPC=MICPC+1
<JUMP!ALCOND!<INWAT1-INIT&3000*4>!<INWAT1-INIT&777/2>>
10$: BRWRT BR, <AA!SPO> ;CHECK IEI
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AA!SPO>>
BR7 ININT ;IEI SET, GENERATE INTERRUPT
MICPC=MICPC+1
<JUMP!BR7CON!<ININT-INIT&3000*4>!<ININT-INIT&777/2>>
INWAT2: BRWRT IBUS INCON ;READ INPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!WRTEBR!IBUS!<INCON>>
BRSHFT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR4 NIDLE6
MICPC=MICPC+1
<JUMP!BR4CON!<NIDLE6-INIT&3000*4>!<NIDLE6-INIT&777/2>>
INSRV: SPBR IBUS INCON, SPO ;READ THE INPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!INCON!SPO>
BR1 30$ ;--SENSE OR BASE
MICPC=MICPC+1
<JUMP!BR1CON!<30$-INIT&3000*4>!<30$-INIT&777/2>>
PSTATE CTLSRV ;ASSUME CNTL I
MEM IMM, <<CTLSRV-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<CTLSRV-INIT&777/2>>>
BR0 20$ ;CNTL I
MICPC=MICPC+1
<JUMP!BR0CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>
BRSHFT ;MUST BE BA/CC
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
PSTATE RBASRV ;ASSUME IN
MEM IMM, <<RBASRV-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<RBASRV-INIT&777/2>>>
BR1 20$ ;REC BA/CC
MICPC=MICPC+1
<JUMP!BR1CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>
PSTATE TBASRV ;XMIT BA/CC
MEM IMM, <<TBASRV-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<TBASRV-INIT&777/2>>>
20$: BRWRT BR, SELA!SP1 ;INIT MODE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
BR0 PROCER ;IF INIT MODE--ERROR
MICPC=MICPC+1

```

DMCGAL.MAC 18-JAN-78 16:05

NIDLE2---NO CSR ACTIVITY STATE

```
(1) 030134 102133
660 030136
(1) 030136 000131
(1) 030136 100447
661 030140
(1) 030140 000132
(1) 030140 102143
662 030142
(1) 030142
(2) 030142 000133
(2) 030142 002461
663 030144
(1) 030144 000134
(1) 030144 000500
664 030146
(1) 030146 000135
(1) 030146 061260
665 030150
(1) 030150 000136
(1) 030150 010177
666 030152
(1) 030152 000137
(1) 030152 016402
667 030154
(1) 030154 000140
(1) 030154 002400
668 030156
(1) 030156 000141
(1) 030156 042233
669 030160
(1) 030160 000142
(1) 030160 114511
670 030162
(1) 030162 000143
(1) 030162 060721
671 030164
(1) 030164 000144
(1) 030164 102133
```

```
<JUMP!BROCON!<PROCEP-INIT&3000*4>!<PROCEP-INIT&777/2>>
ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
30$: BRO INSRV1 ;IF BASE---PROCESS
MICPC=MICPC+1
<JUMP!BROCON!<INSRV1-INIT&3000*4>!<INSRV1-INIT&777/2>>
PROCER: PSTATE NIDLE2 ;RESET PORT STATUS
MEM IMM <<NIDLE2-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>
BRWRT IMM 100 ;CLEAR INPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<100>>
OUT BR AANDB!OINCON ;;
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AANDB!OINCON>>
LDMA IMM <<RTHRS+3>> ;ADDRESS ERROR LINK
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<<RTHRS+3>&377>>
MEMINC IMM 2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<2>>
MEM IMM 0
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<0>>
OUTPUT MEMX SELB!OMODEM ;CLEAR DATA TERMINAL READY
MICPC=MICPC+1
<MOVE!WROUT!MEMX!<SELB!OMODEM>>
ALWAYS RCEXX ;POST THE ERROR - FATAL
MICPC=MICPC+1
<JUMP!ALCOND!<RCEXX-INIT&3000*4>!<RCEXX-INIT&777/2>>
INSRV1: BRWRT BR AXORB!SP1 ;INIT MODE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AXORB!SP1>>
BRO PROCER
MICPC=MICPC+1
<JUMP!BROCON!<PROCEP-INIT&3000*4>!<PROCEP-INIT&777/2>>
```

DMCGAL.MAC 18-JAN-78 16:05

BASSRV---- BASE SERVICE ROUTINE

673
674 030166
(1) 030166
(2) 000145
(2) 030166 002461
675 030170
(1) 000146
(1) 030170 010017
676 030172
(1) 000147
(1) 030172 136500
677 030174
(1) 000150
(1) 030174 136520
678 030176
(1) 000151
(1) 030176 122560
679 030200
(1) 000152
(1) 030200 123000
680 030202
(1) 000153
(1) 030202 000500
681 030204
(1) 000154
(1) 030204 061260
682 030206
(1) 000155
(1) 030206 002133
683 030210
(1) 000156
(1) 030210 040620
684 030212
(1) 000157
(1) 030212 103167
685 030214
(1) 000160
(1) 030214 010151
686 030216
(1) 000161
(1) 030216 016406
687 030220
(1) 000162
(1) 030220 002700
688 030222
(1) 000163
(1) 030222 063161
689 030224
(1) 000164
(1) 030224 000641
690 030226
(1) 000165
(1) 030226 003374
691 030230
(1) 000166
(1) 030230 110743

```

.SBTTL BASSRV---- BASE SERVICE ROUTINE
BASSRV: PSTATE NIDLE2
MEM IMM, <<NIDLE2-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>
LDMA IMM, BASE ;CLEAR TO MAR SO IT POINTS TO BASE POINT
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<BASE&377>>
MEMINC IBUS, PORT1 ;READ CSR4
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT1>>
MEMINC IBUS, PORT2 ;READ CSRS
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT2>>
MEM IBUS, PORT4
MICPC=MICPC+1
<MOVE!WRMEM!IBUS!<PORT4>>
SP IBUS, INCON, SPD ;READ INPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPX!IBUS!INCON!SPD>
BRWRT IMM, 100 ;CLEAR THE BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<100>>
OUT BR, <AANDB!OINCON> ;CLEAR THE INCONTROL CSR
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AANDB!OINCON>>
OUTPUT IMM, <120!OMODEM> ;MASK FOR HDX AND DTR
MICPC=MICPC+1
<MOVE!WROUT!IMM!<120!OMODEM>>
BRWRT MEMX, SELB ;READ SEL6
MICPC=MICPC+1
<MOVE!WRTEBR!MEMX!<SELB>>
BR4 RESUME ;IF SET RESUME
MICPC=MICPC+1
<JUMP!BR4CON!<RESUME INIT&3000*4>!<RESUME-INIT&777/2>>
LDMA IMM, T ;LOAD ADDRESS OF TYPE FIELD
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<T&377>>
MEMINC IMM, 6 ;WRITE START TYPE TO MEMORY
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<6>>
MEM IMM, 300 ;WRITE SELECT AND FINAL TO MEMORY
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<300>>
SP BR, DECA, SP1 ;TURN OFF INIT MODE
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP1>
BS1: BRWRT IMM, 241 ;SET OK TO SEND, STARTMODE AND UNNUM PENDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<241>>
BS0: SP IMM, 374, SP14 ;RESET REP COUNTER
MICPC=MICPC+1
<MOVE!SPX!IMM!374!SP14>
ALWAYS SA3
MICPC=MICPC+1
<JUMP!ALCOND!<SA3-INIT&3000*4 !<SA3-INIT&777/2>>

```

DMOGAL.MAC 18-JAN-78 16:05

BASSRV---- BASE SERVICE ROUTINE

692 030232 000167
(1) 030232 003004
693 030234 000170
(1) 030234 063070
694 030236 000171
(1) 030236 010017
696 030240 000172
(1) 030240 000745
697 030242 000173
(1) 030242 110463
698 030244 000174
(1) 030244 010154
699 030246 000175
(1) 030246 057310
700 030250 000176
(1) 030250 057231
701 030252 000177
(1) 030252 057235
702 030254 000200
(1) 030254 043237
703 030256 000201
(1) 030256 043232
704 030260 000202
(1) 030260 063170
705 030262 000203
(1) 030262 063161
707 030264 000204
(1) 030264 000606
708 030266 000205
(1) 030266 114434
709 030270 000206
(1) 030270 002514
710 030272 000207
(1) 030272 000415
711 030274 000210

```
RESUME: SP      IMM,SP4,4           ;SET UP SP4 FOR COUNTING NPRS
          MICPC=MICPC+1
          <MOVE!SPX!IMM!SP4!4>
          SP      BR,INCA,SP10      ;SET UNNUMB MESSAGE PENDING TO
          MICPC=MICPC+1
          <MOVE!SPX!BR!INCA!SP10>

          LDMA    IMM,BASE          ;TRICK TRANSMITTER CODE
          MICPC=MICPC+1              ;ADDRESS BASE TABLE ADDRESS
          <MOVE!LDMAR!IMM!<BASE&377>>
          STATE   FUDGE
          MICPC=MICPC+1              ;SET TMTR STATE TO ENTER TABLE UPDATE
          <MOVE!WRTEBR!IMM!<FUDGE-INIT&777/2>>
          ALWAYS  TBO
          MICPC=MICPC+1              ;GO SET UP MXT BITS AND ADRESS OF BASE FOR NPRS
          <MOVE!ALCOND!<TBO-INIT&3000*4>!<TBO-INIT&777/2>>
BS2:      LDMA    IMM,IMG10
          MICPC=MICPC+1
          <MOVE!LDMAR!IMM!<IMG10&377>>
          SP      MEMX!INCMAR,AORB,SP10 ;RESTORE BIT 1 OF SP10
          MICPC=MICPC+1
          <MOVE!SPX!MEMX!INCMAR!AORB!SP10>
          SP      MEMX!INCMAR,SELB,SP11 ;RESTORE SP11
          MICPC=MICPC+1
          <MOVE!SPX!MEMX!INCMAR!SELB!SP11>
          SP      MEMX!INCMAR,SELB,SP15 ;RESTORE SP15
          MICPC=MICPC+1
          <MOVE!SPX!MEMX!INCMAR!SELB!SP15>
          SP      MEMX,SELB,SP17      ;RESTORE          SP17
          MICPC=MICPC+1
          <MOVE!SPX!MEMX!SELB!SP17>
          SP      MEMX,SELB,SP12      ;RESTORE          SP12
          MICPC=MICPC+1
          <MOVE!SPX!MEMX!SELB!SP12>
          SP      BR,DECA,SP10        ;TURN OFF UNNUM MESSAGE PENDING AND
          MICPC=MICPC+1
          <MOVE!SPX!BR!DECA!SP10>
          ;ZERO THE BRG
          ;CLEAR INIT MODE
          SP      BR,DECA,SP1
          MICPC=MICPC+1
          <MOVE!SPX!BR!DECA!SP1>
          BRWRT  IMM,206
          MICPC=MICPC+1
          <MOVE!WRTEBR!IMM!<206>>
          ALWAYS  NAK1
          MICPC=MICPC+1
          <MOVE!ALCOND!<NAK1-INIT&3000*4>!<NAK1-INIT&777/2>>
ININT:    PSTATE  INWAT2
          MEM      IMM,<<INWAT2-INIT&777/2>>
          MICPC=MICPC+1
          <MOVE!WRMEM!IMM!<<INWAT2-INIT&777/2>>>
          BRWRT  IMM,15                ;MASK FOR TURN OFF ALL BUT EXT MEM BITS + NXM
          MICPC=MICPC+1
          <MOVE!WRTEBR!IMM!<15>>
          SP      IBUS,UBBR,SPO
          MICPC=MICPC+1
          ;READ BR CONTROL REG
```

DMCGAL.MAC 18-JAN-78 16:05

BASSRV---- BASE SERVICE ROUTINE

PAGE: 0073DM
SEQ 0073

(1)	030274	123220	<MOVE!SPX!IBUS!UBBR!SPO>	
712	030276		SP BR, AANDB, SPO	:MASK OFF VECTOR TO X04
(1)		000211	MICPC=MICPC+1	
(1)	030276	063260	<MOVE!SPX!BR!AANDB!SPO>	
713	030300		BRWRT IMM, 200	:MASK FOR INTERRUPT
(1)		000212	MICPC=MICPC+1	
(1)	030300	000600	<MOVE!WRTEBR!IMM!<200>>	
714	030302		ALWAYS RB4	
(1)		000213	MICPC=MICPC+1	
(1)	030302	104457	<JUMP!ALCOND!<RB4-INIT&3000*4>!<RB4-INIT&777 2>>	

DMCGAL.MAC 18-JAN-78 16:05

OUTINT---SET UP OUTPUT INTERRUPT (RDY0)

716
717 030304
722 030304
(1) 030304
(2) 000214
(2) 030304 002650
724
725 030306
(1) 000215
(1) 030306 010240
726 030310
(1) 000216
(1) 030310 050220
727 030312
(1) 000217
(1) 030312 123040
728 030314
(1) 000220
(1) 030314 055302
729 030316
(1) 000221
(1) 030316 050220
730 030320
(1) 000222
(1) 030320 074520
731
732
733 030322
(1) 000223
(1) 030322 055224
734 030324
(1) 000224
(1) 030324 055225
735 030326
(1) 000225
(1) 030326 055227
736 030330
(1) 000226
(1) 030330 055226
737
738 030332
(1) 000227
(1) 030332 103752
739
745 030334
747 030334
(1) 000230
(1) 030334 010240
748 030336
(1) 000231
(1) 030336 043220
749 030340
(1) 000232
(1) 030340 002642
750 030342
(1) 000233

```
.SBTTL OUTINT---SET UP OUTPUT INTERRUPT (RDY0)
OUTINT: PSTATE OUTWAIT ;PORT STATUS TO WAITING FOR OUT
MEM IMM, <<OUTWAIT-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<OUTWAIT-INIT&777/2>>>
;COMPLETION
LDMA IMM, NXTINT ;ADDRESS OF NEXT INTERRUPT POINTER
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<NXTINT&377>>
LDMA MEMX, SELB ;NEXT INTERRUPT
MICPC=MICPC+1
<MOVE!LDMAR!MEMX!<SELB>>
SP IBUS, OCON, SPO ;READ THE OUTPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPX!IBUS!OCON!SPO>
OUT <MEMX!INCMAR>, <AORB!OOCON> ;WRITE THE OUT CONTROL CSR
MICPC=MICPC+1
<MOVE!WROUTX!MEMX!INCMAR!<AORB!OOCON>>
LDMA MEMX, SELB ;ADDRESS LINK
MICPC=MICPC+1
<MOVE!LDMAR!MEMX!<SELB>>
BRWRT <BR!INCMAR>, <AA!SPO> ;KICK PAST LINK STATUS BYTE
MICPC=MICPC+1
<MOVE!WRTBR!BR!INCMAR!<AA!SPO>>
;SHIFT CSRO IMAGE LEFT
OUT <MEMX!INCMAR>, <SELB!OPOK!1> ;***DO NOT CHANGE BR UNTIL BR7***
;WRITE LOW BYTE OF BA TO CSR
MICPC=MICPC+1
<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPOK!1>>
OUT <MEMX!INCMAR>, <SELB!OPOK!2> ;WRITE HIGH BYTE OF BA TO CSR
MICPC=MICPC+1
<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPOK!2>>
OUT <MEMX!INCMAR>, <SELB!OPOK!4> ;WRITE HIGH BYTE OF COUNT TO CSR
MICPC=MICPC+1
<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPOK!4>>
OUT <MEMX!INCMAR>, <SELB!OPOK!3> ;WRITE THE LOW BYTE OF COUNT
MICPC=MICPC+1
<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPOK!3>>
;***HERE IS BR7***
BR7 PE1 ;INTERRUPT ENABLE IS SET
MICPC=MICPC+1
<JUMP!BR7CON!<PE1-INIT&3000*4>!<PE1-INIT&777/2>>>
;GENERATE AN INTERRUPT
PINT2: LDMA IMM, NXTINT ;ADDRESS NEXT INTERRUPT QUEUE
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<NXTINT&377>>
SP MEMX, SELB, SPO ;COPY ADDRESS FOR NEXT INT TO SPO
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SPO>
MEM IMM, INTSTK ;ASSUME WRAP AROUND CASE
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<INTSTK>>
BRWRT IMM, <<MMEND-2>> ;ADDRESS OF LAST INT IN STACK
MICPC=MICPC+1
```

DMCGAL.MAC 18-JAN-78 16:05

OUTINT---SET UP OUTPUT INTERRUPT (RDY0)

```

(1) 030342 000776      <MOVE!WRTEBR!IMM!<<MMEND-2>>>
751 030344          CMP      BR, SPO          ; SHOULD WE WRAP
(1) 030344 000234      MICPC=MICPC+1
(1) 030344 060360      <SUBTC!BR!SPO>
752 030346          Z          ; YES--BRANCH
(1) 030346 000235      MICPC=MICPC+1
(1) 030346 101640      <JUMP!ZCOND!<5$-INIT&3000*4>!<5$-INIT&777/2>>
753 030350          BRWRT  IMM, 2      ; OFFSET FOR NEXT POINTER
(1) 030350 000236      MICPC=MICPC+1
(1) 030350 000402      <MOVE!WRTEBR!IMM!<2>>
754 030352          MEM      BR, ADD!SPO      ; UPDATE POINTER
(1) 030352 000237      MICPC=MICPC+1
(1) 030352 062400      <MOVE!WRMEM!BR!<ADD!SPO>>
755 030354          5$: SP      MEMX, SELB, SPO      ; COPY POINTER TO SPO
(1) 030354 000240      MICPC=MICPC+1
(1) 030354 043220      <MOVE!SPX!MEMX!SELB!SPO>
756 030356          LDMA   IMM, NXTSP      ; PICK UP START OF IN QUEUE
(1) 030356 000241      MICPC=MICPC+1
(1) 030356 010241      <MOVE!LDMA!IMM!<NXTSP&377>>
757 030360          CMP      MEMX, SPO      ; COMPARE TO END
(1) 030360 000242      MICPC=MICPC+1
(1) 030360 040360      <SUBTC!MEMX!SPO>
758 030362          Z          ; IF EQUAL--CLEAR INT PENDING
(1) 030362 000243      MICPC=MICPC+1
(1) 030362 101645      <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
759 030364          ALWAYS  IDLE
(1) 030364 000244      MICPC=MICPC+1
(1) 030364 100447      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
760 030366          10$: BRWRT  IMM, 357      ; MASK TO CLEAR INT PENDING
(1) 030366 000245      MICPC=MICPC+1
(1) 030366 000757      <MOVE!WRTEBR!IMM!<357>>
761 030370          CLRIDL: SP      BR, AANDB, SP1
(1) 030370 000246      MICPC=MICPC+1
(1) 030370 063261      <MOVE!SPX!BR!AANDB!SP1>
762 030372          ALWAYS  IDLE
(1) 030372 000247      MICPC=MICPC+1
(1) 030372 100447      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

```


DMCGAL.MAC 18-JAN-78 16:05

OUTWAI--WAIT FOR RDYO TO GO AWAY

764
765 030374 000250
(1) 123440
(1) 030374 000251
767 030376 103466
(1) 000252
(1) 030400 000500
772 030402 061262
(1) 000253
(1) 030402 061262
774 030404 000254
(1) 100667
775
776 030406 000255
(1) 123560
(1) 030406 000256
777 030410 001620
(1) 000257
(1) 030412 102677
779 030414 000260
(1) 002113
780 030416 000261
(1) 060600
(1) 030416 000262
781 030420 116351
(1) 030420 000263
782 030422 123000
(1) 000264
(1) 030422 030500
783 030424 000265
(1) 061260
784 030426 000266
(1) 010211
(1) 030430 000267
786 030432 002461
(2) 030432 000270
787 030434 100447
(1) 030434 100447
788
789 030436

```

.SBTTL OUTWAI--WAIT FOR RDYO TO GO AWAY
OUTWAI: SPBR IBUS, OCON, SPO ;READ OUTPUT CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!OCON!SPO>
        BR7 NIDLE6 ;RDYO SET --GET OUT
        MICPC=MICPC+1
        <JUMP!BR7CON!<NIDLE6-INIT&3000*4>!<NIDLE6-INIT&777/2>>
        BRWRITE IMM,100 ;CLEAR CONTROL BITS
        MICPC=MICPC+1
        <MOVE!WRITEBR!IMM!<100>>
        OUT BR, OCON!AANDB
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<OCON!AANDB>>
        ALWAYS INS13
        MICPC=MICPC+1
        <JUMP!ALCOND!<INS13-INIT&3000*4>!<INS13-INIT&777/2>>
.SBTTL CTLSRV--CNTL I SERVICE
CTLSRV: SPBR IBUS, PORT4, SPO ;TO SPO
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!PORT4!SPO>
        BRSHFT
        MICPC=MICPC+1
        <MOVE!SHFTBR!WRITEBR!SELB>
        BR1 HDSEL ;IF SET IS HALF DUPLEX
        MICPC=MICPC+1
        <JUMP!BR1CON!<HDSEL-INIT&3000*4>!<HDSEL-INIT&777/2>>
        OUTPUT IMM, <100!OMODEM> ;MASK DTR, TURN OFF HDX
        MICPC=MICPC+1
        <MOVE!WROUT!IMM!<100!OMODEM>>
INS11: BRWRITE DP, <SELA!SPO> ;RESTORE THE CNTL WORD
        MICPC=MICPC+1
        <MOVE!WRITEBR!DP!<SELA!SPO>>
        BRO CBOOT ;IF SET IS BOOT
        MICPC=MICPC+1
        <JUMP!BROCON!<CBOOT-INIT&3000*4>!<CBOOT-INIT&777/2>>
INS12: SP IBUS, INCON, SPO ;READ THE INPUT CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!INCON!SPO>
        BRWRITE IMM,100 ;ZERO THE BR REGISTER EXCEPT INT ENABLE
        MICPC=MICPC+1
        <MOVE!WRITEBR!IMM!<100>>
        OUT BR, <AANDB!OINCON> ;CLEAR IN CONTROL CSR
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<AANDB!OINCON>>
        LDMA IMM, PRST ;ADDRESS PORT STATE
        MICPC=MICPC+1
        <MOVE!LDMA!IMM!<PRST&377>>
INS13: PSTATE NIDLE2
        MEM IMM, <<NIDLE2-INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>
        ALWAYS IDLE
        MICPC=MICPC+1
        <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
TABMXT: SP IBUS, UBR, SPO ;READ BR CONTROL
    
```

(1)		000271	MICPC=MICPC+1	
(1)	030436	123220	<MOVE!SPX!IBUS!UBBR!SPO>	
790	030440		BRWRT IMM,115	:MASK TO CLEAR BR REQ
(1)		000272	MICPC=MICPC+1	
(1)	030440	000515	<MOVE!WRTEBR!IMM!<115>>	
791	030442		SP BR,AANDB,SPO	
(1)		000273	MICPC=MICPC+1	
(1)	030442	063260	<MOVE!SPX!BR!AANDB!SPO>	
792	030444		BRWRT IMM,4	:INCREMENT MXT
(1)		000274	MICPC=MICPC+1	
(1)	030444	000404	<MOVE!WRTEBR!IMM!<4>>	
793	030446		OUT BR,ADD!OBR	
(1)		000275	MICPC=MICPC+1	
(1)	030446	061011	<MOVE!WROUTX!BR!<ADD!OBR>>	
794	030450		ALWAYS ECX	
(1)		000276	MICPC=MICPC+1	
(1)	030450	114734	<JUMP!ALCOND!<ECX-INIT&3000*4>!<ECX-INIT&777/2>>	
795	030452		BRWRT IMM,100	;HD MASK TO BR
(1)		000277	MICPC=MICPC+1	
(1)	030452	000500	<MOVE!WRTEBR!IMM!<100>>	
796	030454		SP BR,AORB,SP10	;UPDATE PORT STATUS WORD
(1)		000300	MICPC=MICPC+1	
(1)	030454	063310	<MOVE!SPX!BR!AORB!SP10>	
797	030456		ALWAYS INS11	
(1)		000301	MICPC=MICPC+1	
(1)	030456	100661	<JUMP!ALCOND!<INS11-INIT&3000*4>!<INS11-INIT&777/2>>	

HDSEL:

TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE

```

799
800 030460
(1) 000302
(1) 030460 010070
801 030462
(1) 000303
(1) 030462 053220
802 030464
(1) 000304
(1) 030464 016401
803 030466
(1) 000305
(1) 030466 014543
804
805 030470
(1) 000306
(1) 030470 136500
806 030472
(1) 000307
(1) 030472 136520
807 030474
(1) 000310
(1) 030474 136560
808 030476
(1) 000311
(1) 030476 136540
809 030500
(1) 000312
(1) 030500 010070
810 030502
(1) 000313
(1) 030502 002471
811 030504
(1) 000314
(1) 030504 060360
812 030506
(1) 000315
(1) 030506 101720
813 030510
(1) 000316
(1) 030510 000406
814 030512
(1) 000317
(1) 030512 062400
815 030514
(1) 000320
(1) 030514 000402
816 030516
(1) 000321
(1) 030516 063310
817 030520
(1) 000322
(1) 030520 100663
818
819 030522
(1) 000323
    
```

```

.SBTTL TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE
TBASRV: LDMA IMM,ETC ;GET POINTER TO END OF TMT CHAIN
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<ETC&377>>
LDMA MEMX<SELB!SPX!SPO> ;FIND THE LINK
MICPC=MICPC+1
<MOVE!LDMAR!MEMX!<SELB!SPX!SPO>>
MEMINC IMM,1 ;BUFFER ASSIGNED IN IN LINK FLAGS
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<1>>
BRWRT <IMM!INCMAR>,TMLB ;POINT PAST NUMBER FIELD
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!INCMAR!<TMLB>> ;SET BR FOR ADDITION TO SPO
MEMINC IBUS,PORT1
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT1>>
MEMINC IBUS,PORT2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT2>>
MEMINC IBUS,PORT4
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT4>>
MEMINC IBUS,PORT3
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT3>>
LDMA IMM,ETC
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<ETC&377>>
MEM IMM,TML1 ;ASSUME QUEUE WRAP AROUND
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<TML1>> ;END OF CHAIN?
CMP BR,SPO
MICPC=MICPC+1
<SUBTC!BR!SPO> ;IF YES--BRANCH
Z 10$
MICPC=MICPC+1
<JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
BRWRT IMM,6 ;QUEUE ENTRY LENGTH
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<6>>
MEM BR,ADD!SPO ;UPDATE THE END POINTER IN MEMORY
MICPC=MICPC+1
<MOVE!WRMEM!BR!<ADD!SPO>>
10$: BRWRT IMM,2 ;NUMBERED MSG PENDING MASK
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>>
SP BR,AORB,SP10 ;UPDATE LINE STATUS
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP10>
ALWAYS INS12
MICPC=MICPC+1
<JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
.SBTTL RBASRV--RECEIVE BUFFER ADDRESS SERVICE
RBASRV: LDMA IMM,ERC ;ADDRES END OF RECEIVE CHAIN
MICPC=MICPC+1
    
```

DMCGAL.MAC 18-JAN-78 16:05

RBASRV--RECEIVE BUFFER ADDRESS SERVICE

```

(1) 030522 010023 <MOVE!LDMAR!IMM!<ERC&377>>
820 030524 LDMA MEMX,<SELB!SPX!SPO> ;GET THE POINTER TO LINK
(1) 030524 000324 MICPC=MICPC+1
(1) 030524 053220 <MOVE!LDMAR!MEMX!<SELB!SPX!SPO>>
821 030526 MEMINC IMM,1
(1) 030526 000325 MICPC=MICPC+1
(1) 030526 016401 <MOVE!WRMEM!INCMAR!IMM!<1>>
822 030530 MEMINC IBUS,PORT1
(1) 030530 000326 MICPC=MICPC+1
(1) 030530 136500 <MOVE!WRMEM!INCMAR!IBUS!<PORT1>>
823 030532 MEMINC IBUS,PORT2
(1) 030532 000327 MICPC=MICPC+1
(1) 030532 136520 <MOVE!WRMEM!INCMAR!IBUS!<PORT2>>
824 030534 MEMINC IBUS,PORT4
(1) 030534 000330 MICPC=MICPC+1
(1) 030534 136560 <MOVE!WRMEM!INCMAR!IBUS!<PORT4>>
825 030536 MEMINC IBUS,PORT3
(1) 030536 000331 MICPC=MICPC+1
(1) 030536 136540 <MOVE!WRMEM!INCMAR!IBUS!<PORT3>>
826 ;...NOTE INVERTED ORDER OF PORT 3 AND PORT4
827 030540 LDMA IMM,ERC
(1) 030540 000332 MICPC=MICPC+1
(1) 030540 010023 <MOVE!LDMAR!IMM!<ERC&377>>
828 030542 MEM IMM,RCL1 ;ASSUME WRAP AROUND CASE
(1) 030542 000333 MICPC=MICPC+1
(1) 030542 002424 <MOVE!WRMEM!IMM!<RCL1>>
829 030544 BRWRT IMM,RCL7 ;GET ADDRESS OF END OF CAHIN AREA
(1) 030544 000334 MICPC=MICPC+1
(1) 030544 000462 <MOVE!WRTEBR!IMM!<RCL7>>
830 030546 CMP BR,SPO ;CHECK FOR END
(1) 030546 000335 MICPC=MICPC+1
(1) 030546 060360 <SUBTC!BR!SPO>
831 030550 Z INS12 ;IF EQUAL BRANCH
(1) 030550 000336 MICPC=MICPC+1
(1) 030550 101663 <JUMP!ZCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
832 030552 BRWRT IMM,5 ;CALCULATE ADDRESS OF NEXT LINK
(1) 030552 000337 MICPC=MICPC+1
(1) 030552 000405 <MOVE!WRTEBR!IMM!<5>>
833 030554 MEM BR,ADD!SPO ;...
(1) 030554 000340 MICPC=MICPC+1
(1) 030554 062400 <MOVE!WRMEM!BR!<ADD!SPO>>
834 030556 ALWAYS INS12 ;EXIT
(1) 030556 000341 MICPC=MICPC+1
(1) 030556 100663 <JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
835 030560 BRWRT IMM,357 ;MASK TO CLEAR CLR ACTIVE
(1) 030560 000342 MICPC=MICPC+1
(1) 030560 000757 <MOVE!WRTEBR!IMM!<357>>
836 030562 SPBR BR,AANDB,SP10 ;CLEAR BIT IN LINE STATUS WORD
(1) 030562 000343 MICPC=MICPC+1
(1) 030562 063670 <MOVE!SPBRX!BR!AANDB!SP10>
837 030564 BRSHFT
(1) 030564 000344 MICPC=MICPC+1
(1) 030564 001620 <MOVE!SHFTBR!WRTEBR!SELB>
838 030566 BR4 FLUSH
(1) 030566 000345 MICPC=MICPC+1
(1) 030566 107015 <JUMP!BR4CON!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>

```

RA1:

DMCGAL.MAC 18-JAN-78 16:05

RBASRV--RECEIVE BUFFER ADDRESS SERVICE

PAGE: 00800M
SEQ 0080

```

839 030570
(1) 000346
(1) 030570 000400
840 030572
(1) 000347
(1) 030572 063233
841 030574
(1) 000350
(1) 030574 000427
842 030576
(1) 000351
(1) 030576 104425
843

```

```

RA3:  BRWRT  IMM,0           ;CLEAR BR
      MICPC=MICPC+1
      <MOVE!WRTBR!IMM!<0>>
      SP BR,SELB,SP13       ;SET NUMB MESSAGE TYPE IN SP13
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP13>
      STATE RCVB           ;CHANGE RECEIVE STATE POINTER TO STATE B
      MICPC=MICPC+1
      <MOVE!WRTBR!IMM!<RCVB-INIT&777/2>>
      ALWAYS REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
      ;

```

DMCGAL.MAC 18-JAN-78 16:05

RBASRV--RECEIVE BUFFER ADDRESS SERVICE

```

845 030600 000352
(1) 030600 000700
846 030602 000353
(1) 030602 123220
847 030604 000354
(1) 030604 061311
852 030606 000355
(1) 030606 100630
854
855
856 030610 000356
(1) 030610 120600
857 030612 000357
(1) 030612 102047
858 030614 000360
(1) 030614 114725
859
860
861 030616 000361
(1) 030616 000404
862 030620 000362
(1) 030620 114657
863 030622 000363
(1) 030622 000720
864 030624 000364
(1) 030624 110554
865 030626 000365
(1) 030626 000727
866 030630 000366
(1) 030630 063270
867 030632 000367
(1) 030632 104504
872

```

```

PE1:  BRWRT  IMM,300          ;MASK FOR INTERRUPT AND VECTOR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<300>>
      SF  IBUS,UBBR,SPO      ;READ BR CONTROL REG
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!UBBR!SPO>
      OUT BR,<AORB!OBR>      ;INTERRUPT
      MICPC=MICPC+1
      <MOVE!WROUTX!BR!<AORB!OBR>>
      ALWAYS PINT2
      MICPC=MICPC+1
      <JUMP!ALCOND!<PINT2-INIT&3000*4>!<PINT2-INIT&777/2>>

;
†BU1: BRWRT  IBUS,NPR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<NPR>>
      BRO  IDLE
      MICPC=MICPC+1
      <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      ALWAYS EC2
      MICPC=MICPC+1
      <JUMP!ALCOND!<EC2-INIT&3000*4>!<EC2-INIT&777/2>>

;
OVRRUN: BRWRT  IMM,4
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<4>>
        ALWAYS NTRSO
        MICPC=MICPC+1
        <JUMP!ALCOND!<NTRSO-INIT&3000*4>!<NTRSO-INIT&777/2>>

HEH1:  STATE  TEOM
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<TEOM-INIT&777/2>>
        ALWAYS XEXIT
        MICPC=MICPC+1
        <JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>

ACK1:  BRWRT  IMM,327
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<327>>
        SP  BR,AANDB,SP10
        MICPC=MICPC+1
        <MOVE!SPX!BR!AANDB!SP10>
        ALWAYS RDS
        MICPC=MICPC+1
        <JUMP!ALCOND!<RDS-INIT&3000*4>!<RDS-INIT&777/2>>

```

```

874          030654          . =INIT+1000
875          000377          MICPC=377
876          .SBTTL RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER
877          ;ENTERED FROM IDLE LOOP
878          ;DETERMINES IF MESSAGE TYPE IS NUMBERED, UNNUMBERED OR BOOT
879          ;SETS UP APPROPRIATE STATES FOR REST OF MESSAGE.
880 030654    RCVA:  $P          IBUS,RCVDAT,SPO          ;READ RECEIVE CHARACTER TO SPO
(1)          000400          MICPC=MICPC+1
(1) 030654    023200          <MOVE!SPX!IBUS!RCVDAT!SPO>
881 030656    BRWRT BR SELA!SP1          ;READ PORT STATUS WORD
(1)          000401          MICPC=MICPC+1
(1) 030656    060601          <MOVE!WRTBR!BR!<SELA!SP1>>
882 030660    BRO          SS          ;IF INIT MODE---ONLY BOOT OK
(1)          000402          MICPC=MICPC+1
(1) 030660    106012          <JUMP!BROCON!<SS-INIT&3000*4>!<SS-INIT&777/2>>
883 030662    BR7          SS          ;IF BOOT MODE---ONLY BOOT OK
(1)          000403          MICPC=MICPC+1
(1) 030662    107412          <JUMP!BR7CON!<SS-INIT&3000*4>!<SS-INIT&777/2>>
884 030664    BRWRT IMM,201          ;SOH TO BR
(1)          000404          MICPC=MICPC+1
(1) 030664    000601          <MOVE!WRTBR!IMM!<201>>
885 030666    CMP          BR,SPO          ;COMPARE BR TO SPO
(1)          000405          MICPC=MICPC+1
(1) 030666    060360          <SUBTC!BR!SPO>
886 030670    Z          RA1          ;IF EQUAL-IS NUMBERED MESSAGE
(1)          000406          MICPC=MICPC+1
(1) 030670    101742          <JUMP!ZCOND!<RA1-INIT&3000*4>!<RA1-INIT&777/2>>
887 030672    BRWRT IMM,S          ;ENQ TO BR
(1)          000407          MICPC=MICPC+1
(1) 030672    000405          <MOVE!WRTBR!IMM!<S>>
888 030674    CMP          BR,SPO          ;COMPARE ENQ TO SPO
(1)          000410          MICPC=MICPC+1
(1) 030674    060360          <SUBTC!BR!SPO>
889 030676    Z          RA2          ;IF EQUAL-IS UNNUMBERED MESSAGE
(1)          000411          MICPC=MICPC+1
(1) 030676    105424          <JUMP!ZCOND!<RA2-INIT&3000*4>!<RA2-INIT&777/2>>
890 030700    SS:  BRWRT IMM,220          ;DLE TO BR
(1)          000412          MICPC=MICPC+1
(1) 030700    000620          <MOVE!WRTBR!IMM!<220>>
891 030702    CMP          BR,SPO          ;COMPARE DLE TO SPO
(1)          000413          MICPC=MICPC+1
(1) 030702    060360          <SUBTC!BR!SPO>
892 030704    Z          BOOT          ;IF EQUAL IS BOOT
(1)          000414          MICPC=MICPC+1
(1) 030704    115760          <JUMP!ZCOND!<BOOT-INIT&3000*4>!<BOOT-INIT&777/2>>
893 030706    FLUSH:  OUTPUT IMM,<200!ORCVCO>          ;FLUSH INPUT SILO
(1)          000415          MICPC=MICPC+1
(1) 030706    002212          <MOVE!WROUT!IMM!<200!ORCVCO>>
894          ;(LOW ORDER BITS READ ONL.)
895 030710    RSTATE RCVA          ;SET STATE TO RCVA
(1)          000416          MICPC=MICPC+1
(1) 030710    000400          <MOVE!WRTBR!IMM!<RCVA-INIT&777/2>>
(1)          000417          MICPC=MICPC+1
(1) 030712    063223          <MOVE!SPX!BR!SELB!SP3>
896 030714    BRWRT IMM,357          ;MASK TO CLEAR--CLEAR ACTIVE
(1)          000420          MICPC=MICPC+1

```

F07

DMCGAL.MAC 18-JAN-78 16:05

RCVA--ROUTINE TO HANDLE FIRST JDCMP CHARACTER

```
(1) 030714 000757
897 030716 000421
(1) 030716 063270
898 030720 000422
(1) 030720 060070
899 030722 000423
(1) 030722 110665
900 030724 000424
(1) 030724 000703
901 030726 000425
(1) 030726 063223
902 030730 000426
(1) 030730 100447
```

```

      <MOVE!WRTEBR!IM!<357>>
      SP      BR,AANDB,SP10      ;IN LINE STATUS WORD
      MICPC=MICPC+1
      <MOVE!SPX!BR!AANDB!SP10>
FLUSHA: NOP      BR,INCA,SP10      ;CLEAR "C" BIT FOR TIMEO
      MICPC=MICPC+1
      <BR!INCA!SP10>
      ALWAYS TIMEO      ;SEE IF WE CAN SEND AN ACK
      MICPC=MICPC+1
      <JUMP!ALCOND!<TIMEO-INIT&3000*4>!<TIMEO-INIT&777/2>>
RA2:  STATE RCVI      ;CHANGE RECEIVE STATE TO I
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVI-INIT&777/2>>
REXIT: SP      BR,SELB,SP3
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP3>
      ALWAYS IDLE
      MICPC=MICPC+1
      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
```


DMCGAL.MAC 18-JAN-78 16:05

RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD

```

904
905
906
907
908 030732
909 030732
(1) 000427
(1) 030732 023204
910 030734
(1) 000430
(1) 030734 070215
911 030736
(1) 000431
(1) 030736 054620
912 030740
(1) 000432
(1) 030740 106042
913 030742
(1) 000433
(1) 030742 060601
914 030744
(1) 000434
(1) 030744 107440
915 030746
(1) 000435
(1) 030746 000710
916 030750
(1) 000436
(1) 030750 010012
917 030752
(1) 000437
(1) 030752 104560
918 030754
(1) 000440
(1) 030754 000402
919 030756
(1) 000441
(1) 030756 063301
920 030760
(1) 000442
(1) 030760 000462
921 030762
(1) 000443
(1) 030762 063223
922 030764
(1) 000444
(1) 030764 056226
923 030766
(1) 000445
(1) 030766 056227
924 030770
(1) 000446
(1) 030770 123220
925 030772
(1) 000447
(1) 030772 000501

```

```

.SBTTL RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD
:ENTERED FROM IDLE LOOP
:STORES COUNT FIELD AND SETS UP RCVC AS NEXT STATE
:

```

```

RCVB:
SP      IBUS,RCVDAT,SP4           ;READ CHARACTER TO SP4
MICPC=MICPC+1
<MOVE!SPX!IBUS!RCVDAT!SP4>
LDMA   BR <SELA!SP15>           ;LOAD MAR WITH ADDRESS OF CURRENT BA
MICPC=MICPC+1
<MOVE!LDMA!BR!<SELA!SP15>>
BRWRT  MEMX!INCMAR,SELB       ;READ THE FLAGS BYTE
MICPC=MICPC+1
<MOVE!WRTBR!MEMX!INCMAR!<SELB>>
BR0    RB1                     ;BUFFER IS ASSIGNED
MICPC=MICPC+1
<JUMP!BR0CON!<RB1-INIT&3000*4>!<RB1-INIT&777/2>>
BRWRT  BR,SELA!SP1            ;NO BUFFER ASSIGNED - MAINT MODE?
MICPC=MICPC+1
<MOVE!WRTBR!BR!<SELA!SP1>>
BR7    RB3                     ;YES
MICPC=MICPC+1
<JUMP!BR7CON!<RB3-INIT&3000*4>!<RB3-INIT&777/2>>
BRWRT  IMM,310                 ;SET NAK REASON CODE
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<310>>
LDMA   IMM,NTLS                ;ADDRESS CUMULATIVE COUNTER
MICPC=MICPC+1
<MOVE!LDMA!IMM!<NTLS&377>>
ALWAYS RHS                     ;SEND A NAK
MICPC=MICPC+1
<JUMP!ALCOND!<RHS-INIT&3000*4>!<RHS-INIT&777/2>>
RB3:  BRWRT  IMM,2              ;FLAG FOR NO BUF ASSIGNED IN MAINT MODE
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<2>>
SP      BR,AORB,SP1           ;SET THE FLAG
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP1>
RB1:  STATE  RCVC
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<RCVC-INIT&777/2>>
RB0:  SP      BR,SELB,SP3
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>
OUTPUT <MEMX!INCMAR>,<SELB!OBA1> ;OUTPUT LOW ORDER BYTE OF ADDRESS
MICPC=MICPC+1
<MOVE!WROUT!MEMX!INCMAR!<SELB!OBA1>>
OUTPUT MEMX!INCMAR,<SELB!OBA2> ;OUTPUT HIGH BYTE OF ADDRESS
MICPC=MICPC+1
<MOVE!WROUT!MEMX!INCMAR!<SELB!OBA2>>
SP      IBUS,UBBR,SPO         ;READ THE BUS REQ REGISTER
MICPC=MICPC+1
<MOVE!SPX!IBUS!UBBR!SPO>
BRWRT  IMM,101                 ;MASK OFF ALL BUT NXM AND VEC4 BITS
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<101>

```

H07

DMCGAL.MAC 18-JAN-78 16:05

RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD

926	030774	000450	SP BR AANDB, SPO ;AND SAVE IN SPO
(1)		063260	MICPC=MICPC+1
927	030774		<MOVE!SPX!BR!AANDB!SPO>
(1)	030776	000451	SP IMM, 300, SP5 ;MASK TO ISOLATE EX. MEM BITS
(1)		003305	MICPC=MICPC+1
928			<MOVE!SPX!IMM!300!SP5>
929			;NOTE THIS REALLY WRITES A 305 BUT THE
930	031000	000452	BRWRT MEMX, AANDB!SP5 ;S GETS SHIFTED OUT
(1)		040665	MICPC=MICPC+1 ;MASK ALL BUT EX. MEM BITS
(1)	031000		<MOVE!WRTEBR!MEMX!<AANDB!SP5>
931	031002		BRSHFT ;SHIFT THEM INTO THE CORRECT POSITION
(1)		000453	MICPC=MICPC+1
(1)	031002	001620	<MOVE!SHFTBR!WRTEBR!SELB>
932	031004		BRSHFT
(1)		000454	MICPC=MICPC+1
(1)	031004	001620	<MOVE!SHFTBR!WRTEBR!SELB>
933	031006		BRSHFT
(1)		000455	MICPC=MICPC+1
(1)	031006	001620	<MOVE!SHFTBR!WRTEBR!SELB>
934	031010		BRSHFT
(1)		000456	MICPC=MICPC+1
(1)	031010	001620	<MOVE!SHFTBR!WRTEBR!SELB>
935	031012		OUT BR AORB!OBR ;WRITE EX MEM BITS OUT
(1)		000457	MICPC=MICPC+1
(1)	031012	061311	<MOVE!WROUTX!BR!<AORB!OBR>>
936	031014		ALWAYS IDLE
(1)		000460	MICPC=MICPC+1
(1)	031014	100447	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
937	031016		ALWAYS I2
(1)		000461	MICPC=MICPC+1
(1)	031016	100453	<JUMP!ALCOND!<I2-INIT&3000*4>!<I2-INIT&777/2>>
938			.SBTTL RCVB--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD. SELECT AND FINA
939			;ENTERED FROM IDLE LOOP
940			;INTERPRETS SELECT AND FINAL
941			;CHECKS FOR COUNT TOO LARGE
942			;
943	031020		RCVC: ALWAYS SELQSY ;"CALL" SELECT/OSYNC SUBROUTINE
945	031020		MICPC=MICPC+1
(1)		000462	<JUMP!ALCOND!<SELQSY-INIT&3000*4>!<SELQSY-INIT&777/2>>
(1)	031020	104756	LDMA IMM, BC ;LOAD MAR TO BYTE COUNT
950	031022		MICPC=MICPC+1
(1)		000463	<MOVE!LDMAR!IMM!<BC&377>>
(1)	031022	010167	MEMINC BR SELA!SP4 ;SAVE LOW BYTE
951	031024		MICPC=MICPC+1
(1)		000464	<MOVE!WRMEM!INCMAR!BR!<SELA!SP4>>
(1)	031024	076604	MEMINC BR SELA!SP5 ;AND NOW HIGH BYTE
952	031026		MICPC=MICPC+1
(1)		000465	<MOVE!WRMEM!INCMAR!BR!<SELA!SP5>>
(1)	031026	076605	STATE RCVD ;SET NEXT STATE TO D
953	031030		MICPC=MICPC+1
(1)		000466	<MOVE!WRTEBR!IMM!<RCVD-INIT&777/2>>
(1)	031030	000470	ALWAYS REXIT
954	031032		MICPC=MICPC+1
(1)		000467	<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
(1)	031032	104425	

DMCGAL.MAC 18-JAN-78 16:05

RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES

```

956
957
958 031034 000470
(1) 031034 000522
959 031036 000471
(1) 031036 063223
960 031040 000472
(1) 031040 023600
961 031042 000473
(1) 031042 060757
962 031044 000474
(1) 031044 060641
963 031046 000475
(1) 031046 107514
964 031050 000476
(1) 031050 060610
965 031052 000477
(1) 031052 001620
966 031054 000500
(1) 031054 103047
967 031056 000501
(1) 031056 106110
968 031060 000502
(1) 031060 010153
969 031062 000503
(1) 031062 062600
970 031064 000504
(1) 031064 010003
971 031066 000505
(1) 031066 002401
972 031070 000506
(1) 031070 003374
973 031072 000507
(1) 031072 100447
974 031074 000510
(1) 031074 060600
975 031076 000511
(1) 031076 060772

```

```

.SBTTL RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES
RCVD: STATE RCVE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVE-INIT&777/2>>
RD2: SP BP SELB,SP3 ;SAVE THE STATE
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>
SPBR IBUS RCVDAT,SP0 ;INPUT THE CHARACTER
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!RCVDAT!SP0>
BRWRT BR SUB!SP17 ;COMPARE NEW R TO LAST R
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SUB!SP17>>
BRWRT BR AORNB!SP1 ;IF NEW IS THE SAME OR LESS OR
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AORNB!SP1>>
BR7 RD7 ;MAINT MODE - GET OUT
MICPC=MICPC+1
<JUMP!BR7CON!<RD7-INIT&3000*4>!<RD7-INIT&777/2>>
BRWRT BR SELA!SP10
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP10>>
BRSHFT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR4 IDLE
MICPC=MICPC+1
<JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
BR0 RD4 ;MESSAGE IN PROGRESS
MICPC=MICPC+1
<JUMP!BR0CON!<RD4-INIT&3000*4>!<RD4-INIT&777/2>>
RD3: LDMA IMM,ISP17 ;ADDRESS LAST ACKED IMAGE
MICPC=MICPC+1
<MOVE!LDMA!IMM!<ISP17&377>>
MEM BR SELA!SP0 ;COPY THE CHAR
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELA!SP0>>
RD5: LDMA IMM,REPST ;POINT TO REP THRESHOLD COUNTER
MICPC=MICPC+1
<MOVE!LDMA!IMM!<REPST&377>>
MEM IMM,1 ;RESET REP THRESHOLD
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<1>>
RD6: SP IMM,374,SP14 ;RESET THE COUNT
MICPC=MICPC+1
<MOVE!SPX!IMM!374!SP14>
ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
RD4: BRWRT BR SELA!SP0
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP0>>
BRWRT BR SUBTC!SP12
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SUBTC!SP12>>

```

DMUJHL.MAC 18-JAN-78 16:05

RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES

976	031100	000512	BR7 IDLE
(1)		103447	MICPC=MICPC+1
(1)	031100		<JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
977	031102	000513	ALWAYS RD3
(1)		104502	MICPC=MICPC+1
(1)	031102		<JUMP!ALCOND!<RD3-INIT&3000*4>!<RD3-INIT&777/2>>
978	031104	000514	BRWRITE BR SELA!SPO
(1)		060600	MICPC=MICPC+1
(1)	031104		<MOVE!WRITEBR!BR!<SELA!SPO>>
979	031106	000515	CMP BR SP12
(1)		060372	MICPC=MICPC+1
(1)	031106		<SUBTC!BR!SP12>
980	031110	000516	Z RD6
(1)		105506	MICPC=MICPC+1
(1)	031110		<JUMP!ZCOND!<RD6-INIT&3000*4>!<RD6-INIT&777/2>>
981	031112	000517	BRWRITE BR AA!SP10
(1)		060530	MICPC=MICPC+1 ;IF HDX
(1)	031112		<MOVE!WRITEBR!BR!<AA!SP10>>
982	031114	000520	BR7 TIME2
(1)		113705	MICPC=MICPC+1 ;QUEUE UP REP
(1)	031114		<JUMP!BR7CON!<TIME2-INIT&3000*4>!<TIME2-INIT&777/2>>
983	031116	000521	ALWAYS IDLE
(1)		100447	MICPC=MICPC+1
(1)	031116		<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
984			.SBTTL RCVE--ROUTINE TO HANDLE N FIELD OF NUMBERED MESSAGE
985			.
986	031120	000522	BRWRITE BR SELA!SP1 ;READ THE STATUS BYTE
(1)		060601	MICPC=MICPC+1
(1)	031120		<MOVE!WRITEBR!BR!<SELA!SP1>>
987	031122	000523	BR7 RCVQ
(1)		107716	MICPC=MICPC+1
(1)	031122		<JUMP!BR7CON!<RCVQ-INIT&3000*4>!<RCVQ-INIT&777/2>>
988	031124	000524	BRWRITE IBUS RCVDAT ;INPUT THE CHARACTER
(1)		020600	MICPC=MICPC+1
(1)	031124		<MOVE!WRITEBR!IBUS!<RCVDAT>>
989	031126	000525	CMP BR SP11
(1)		060371	MICPC=MICPC+1
(1)	031126		<SUBTC!BR!SP11>
990	031130	000526	Z SS
(1)		105531	MICPC=MICPC+1
(1)	031130		<JUMP!ZCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
991	031132	000527	SP BR DECA,SP13 ;FORCE MSG TYPE TO -1
(1)		063173	MICPC=MICPC+1
(1)	031132		<MOVE!SPX!BR!DECA!SP13>
992	031134	000530	ALWAYS RE2
(1)		104532	MICPC=MICPC+1
(1)	031134		<JUMP!ALCOND!<RE2-INIT&3000*4>!<RE2-INIT&777/2>>
993	031136	000531	SP BR INCA,SP11 ;UPDATE R FIELD
(1)		063071	MICPC=MICPC+1
(1)	031136		<MOVE!SPX!BR!INCA!SP11>
994	031140	000532	STATE RCVF ;NEXT RECEIVE STATE IS F
(1)		000534	MICPC=MICPC+1
(1)	031140		<MOVE!WRITEBR!IMM!<RCVF-INIT&777/2>>
995	031142	000533	ALWAYS REXIT
(1)		104425	MICPC=MICPC+1
(1)	031142		<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>

```

996
997 031144 000534
(1) 031144 063164
998 031146 000535
(1) 031146 105137
999 031150 000536
(1) 031150 063165
1000 031152 000537
(1) 031152 000542
1001 031154 000540
(1) 031154 020200
1002 031156 000541
(1) 031156 104425
1003
1004
1005
1006 031160 000542
(1) 031160 000544
1007 031162 000543
(1) 031162 104540

```

RCVF--ROUTINE TO IGNORE ADDRESS

```

.SBTTL RCVF--ROUTINE TO IGNORE ADDRESS
RCVF:  SP      BR,DECA,SP4          ;DECREMENTLOW BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP4>
      C      RCVFO
      MICPC=MICPC+1
      <JUMP!CCOND!<RCVFO-INIT&3000*4>!<RCVFO-INIT&777/2>>
      SP      BR,DECA,SP5          ;OVERFLOW - DECREMENT HIGH BYTE
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP5>
RCVFO:  STATE  RCVG
      MICPC=MICPC+1
      <MOVE!WTEBR!IMM!<RCVG-INIT&777/2>>
RCVF1:  NOP    IBUS,RCVDAT,0        ;INPUT CHARACTER - AND DISCARD
      MICPC=MICPC+1
      <IBUS!RCVDAT!0>
      ALWAYS  REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
      .SBTTL RCVG--ROUTINE TO IGNORE CRC1
RCVG:  STATE  RCVH
      MICPC=MICPC+1
      <MOVE!WTEBR!IMM!<RCVH-INIT&777/2>>
      ALWAYS  RCVF1
      MICPC=MICPC+1
      <JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>

```

```

1009
1010
1011 031164
1012 031164
(1) 000544
(1) 031164 023200
1013 031166
(1) 000545
(1) 031166 020640
1014 031170
(1) 000546
(1) 031170 116157
1015 031172
(1) 000547
(1) 031172 060601
1016 031174
(1) 000550
(1) 031174 107746
1017 031176
(1) 000551
(1) 031176 060610
1018 031200
(1) 000552
(1) 031200 001620
1019 031202
(1) 000553
(1) 031202 117303
1020 031204
(1) 000554
(1) 031204 010153
1021 031206
(1) 000555
(1) 031206 062617
1022 031210
(1) 000556
(1) 031210 000701
1023 031212
(1) 000557
(1) 031212 010013
1024 031214
(1) 000560
(1) 031214 043220
1025 031216
(1) 000561
(1) 031216 062460
1026 031220
(1) 000562
(1) 031220 010151
1027 031222
(1) 000563
(1) 031222 016402
1028 031224
(1) 000564
(1) 031224 062620
1029 031226
(1) 000565
    
```

```

.SBTTL RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYP
;
RCVH:
SP      IBUS,RCVDAT,SPO                ;GET CHAR IN SPO
MICPC=MICPC+1
<MOVE!SPX!IBUS!RCVDAT!SPO>
BRWRT  IBUS,RCVCON                    ;READ RECVR CONTROL REGISTER
MICPC=MICPC+1
<MOVE!WRTBR!IBUS!<RCVCON>>
BR0     TDON1                          ;IF BCC MATCH SET CRC IS GOOD
MICPC=MICPC+1
<JUMP!BR0CON!<TDON1-INIT&3000*4>!<TDON1-INIT&777/2>>
BRWRT  BR,SELA!SP1                    ;READ STATUS BYTE
MICPC=MICPC+1
<MOVE!WRTBR!BR!<SELA!SP1>>
BR7     RHX                             ;MAINT MODE
MICPC=MICPC+1
<JUMP!BR7CON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
BRWRT  DP,<SELA!SP10>                 ;READ PORT STATUS WORD TO BR
MICPC=MICPC+1
<MOVE!WRTBR!DP!<SELA!SP10>>
BRSHFT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTBR!SELB>
BR4     SNAK1                          ;IF START MODE--PROCEED TO RESEND START
MICPC=MICPC+1
<JUMP!BR4CON!<SNAK1-INIT&3000*4>!<SNAK1-INIT&777/2>>
LDMA   IMM,ISP17                       ;ELSE BCC ERROR--POINT TO IMAGE OF SP17 IN RAM
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<ISP17&377>>
MEM     BR,SELA!SP17                   ;RESTORE LAST ACKED IMAGE
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELA!SP17>>
BRWRT  IMM,301                         ;WRITE HEADER BCC ERROR SUBTYPE
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<301>>
LDMA   IMM,NHDS                         ;ADDRESS CUM ERROR COUNTER
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<NHDS&377>>
RHS:  SP      MEMX,SELB,SPO            ;WRITE IT TO SPO
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SPO>
MEM     BR,INCA!SPO                    ;INCREMENT IT
MICPC=MICPC+1
<MOVE!WRMEM!BR!<INCA!SPO>>
LDMA   IMM,T                            ;LOAD ADDRESS OF TYPE FIELD
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<T&377>>
MEMINC IMM,2                            ;WRITE NAK TYPE
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!\2>>
MEM     BR,SELB                         ;WRITE NAK SUBTYPE FIELD SUBTYPE
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELB>>
LDMA   IMM,NAKST                        ;ADDRESS NAKS TMTED DYNAMIC
MICPC=MICPC+1
    
```

DMCGAL.MAC 18-JAN-78 16:05

RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYPES

(1)	031226	010001	<MOVE!LDMAR!IMM!<NAKST&377>>	
1030	031230		BRWRT MEMX,SELB	;WRITE IT TO BR
(1)		000566	MICPC=MICPC+1	
(1)	031230	040620	<MOVE!WRTEBR!MEMX!<SELB>>	
1031	031232		BSHFTB	;SHIFT IT RIGHT
(1)		000567	MICPC=MICPC+1	
(1)	031232	061620	<MOVE!SHFTBR!SELB!BR>	
1032	031234		MEM BR,SELB	;UPDATE IT
(1)		000570	MICPC=MICPC+1	
(1)	031234	062620	<MOVE!WRMEM!BR!<SELB>>	
1033	031236		BRO NTHRES	;BRANCH IF THRESHOLD EXCEEDED
(1)		000571	MICPC=MICPC+1	
(1)	031236	116252	<JUMP!BROCON!<NTHRES-INIT&3000*4>!<NTHRES-INIT&777/2>>	
1034	031240		ALWAYS SNAK	
(1)		000572	MICPC=MICPC+1	
(1)	031240	114700	<JUMP!ALCOND!<SNAK-INIT&3000*4>!<SNAK-INIT&777/2>>	
1035	031242		BRWRT DP,<DECA!SP13>	;LOAD TYPE RECEIVED--DECREMENTING
(1)		000573	MICPC=MICPC+1	
(1)	031242	060573	<MOVE!WRTEBR!DP!<DECA!SP13>>	
1036	031244		Z RH1	;IF ALUOUT IS ALL ONES IS NUMBERED MSG
(1)		000574	MICPC=MICPC+1	
(1)	031244	115456	<JUMP!ZCOND!<RH1-INIT&3000*4>!<RH1-INIT&777/2>>	
1037	031246		RSTATE RCVA	
(1)		000575	MICPC=MICPC+1	
(1)	031246	000400	<MOVE!WRTEBR!IMM!<RCVA-INIT&777/2>>	
(1)		000576	MICPC=MICPC+1	
(1)	031250	063223	<MOVE!SPX!BR!SELB!SP3>	
1038	031252		BRWRT DP,<SELA!SP10>	;LOAD LINE STATUS WORD IN BR
(1)		000577	MICPC=MICPC+1	
(1)	031252	060610	<MOVE!WRTEBR!DP!<SELA!SP10>>	
1040	031254		BR4 FLUSH1	
(1)		000600	MICPC=MICPC+1	
(1)	031254	117024	<JUMP!BR4CON!<FLUSH1-INIT&3000*4>!<FLUSH1-INIT&777/2>>	
1041	031256			
1046	031256		CG1: BRSHFT	;SHIFT RIGHT
(1)		000601	MICPC=MICPC+1	
(1)	031256	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
1047	031260		BR4 10\$	
(1)		000602	MICPC=MICPC+1	
(1)	031260	107210	<JUMP!BR4CON!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>	
1048	031262		LDMA IMM,TYPTAB	;ADDRESS TYPE TABLE
(1)		000603	MICPC=MICPC+1	
(1)	031262	010162	<MOVE!LDMAR!IMM!<TYPTAB&377>>	
1049	031264		CMP <MEMX!INCMAR>,SP13	
(1)		000604	MICPC=MICPC+1	
(1)	031264	054373	<SUBTC!MEMX!INCMAR!SP13>	
1050	031266		Z REP	
(1)		000605	MICPC=MICPC+1	
(1)	031266	115402	<JUMP!ZCOND!<REP-INIT&3000*4>!<REP-INIT&777/2>>	
1051	031270		CMP <MEMX!INCMAR>,SP13	
(1)		000606	MICPC=MICPC+1	
(1)	031270	054373	<SUBTC!MEMX!INCMAR!SP13>	
1052	031272		Z NAK	
(1)		000607	MICPC=MICPC+1	
(1)	031272	115426	<JUMP!ZCOND!<NAK-INIT&3000*4>!<NAK-INIT&777/2>>	
1053	031274		LDMA IMM,TYPSTT	;SET POINTER TO START TYPE

(1)		000610	MICPC=MICPC+1
(1)	031274	010164	<MOVE!LDMAR!IMM!<TYPSTT&377>>
1054	031276		CMP <MEMX!INCMAR>,SP13
(1)		000611	MICPC=MICPC+1
(1)	031276	054373	<SUBTC!MEMX!INCMAR!SP13>
1055	031300		Z START
(1)		000612	MICPC=MICPC+1
(1)	031300	115516	<JUMP!ZCOND!<START-INIT&3000*4>!<START-INIT&777/2>>
1056			;STACK TYPE
1057	031302		CMP <MEMX!INCMAR>,SP13
(1)		000613	MICPC=MICPC+1
(1)	031302	054373	<SUBTC!MEMX!INCMAR!SP13>
1058	031304		Z STACK
(1)		000614	MICPC=MICPC+1
(1)	031304	115411	<JUMP!ZCOND!<STACK-INIT&3000*4>!<STACK-INIT&777/2>>
1059	031306		CMP <MEMX!INCMAR>,SP13
(1)		000615	;ACK TYPE
(1)	031306	054373	MICPC=MICPC+1
1060	031310		<SUBTC!MEMX!INCMAR!SP13>
(1)		000616	Z ACK
(1)	031310	105773	MICPC=MICPC+1
1061	031312		<JUMP!ZCOND!<ACK-INIT&3000*4>!<ACK-INIT&777/2>>
(1)		000617	ALWAYS IDLE
(1)	031312	100447	MICPC=MICPC+1
			<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
			;OTHERWISE IGNORE--MUST BE OBS MSG

DMCGAL.MAC 18-JAN-78 16:05

RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYPES

1064 031314 000620
(1)
(1) 031314 023640
1065 031316 000621
(1)
(1) 031316 060400
1066 031320 000622
(1)
(1) 031320 103451
1067 031322 000623
(1)
(1) 031322 110402
1069
1070
1071 031324 000624
(1)
(1) 031324 123600
1072 031326 000625
(1)
(1) 031326 102047
1073 031330 000626
(1)
(1) 031330 022203
1074 031332 000627
(1)
(1) 031332 000672
1075 031334 000630
(1)
(1) 031334 063223
1076 031336 000631
(1)
(1) 031336 000621
1077 031340 000632
(1)
(1) 031340 104641
1078
1079 031342 000633
(1)
(1) 031342 123600
1084 031344 000634
(1)
(1) 031344 102047
1086 031346 000635
(1)
(1) 031346 000646
1087 031350 000636
(1)
(1) 031350 063223
1088 031352 000637
(1)
(1) 031352 022203
1089 031354 000640
(1)
(1) 031354 000421
1090 031356 000641
(1)

```

RCVCK:  SPBR  IBUS,RCVCON,SP0          ;READ RCVR CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!RCVCON!SP0>
        BRWRT  BR,ADD!SP0             ;SHIFT LEFT
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<ADD!SP0>>
        BR7    I1
        MICPC=MICPC+1
        <JUMP!BR7CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
        ALWAYS  TAI
        MICPC=MICPC+1
        <JUMP!ALCOND!<TAI-INIT&3000*4>!<TAI-INIT&777/2>>
:*****TIME CRITICAL CODE-- CHANGE WITH GREAT CARE*****
RCVK01: SBTTL  RCVK01--ROUTINE TO HANDLE FIRST BYTE ODD RECEIVE
        SPBR  IBUS,NPR,SP0          ;READ NPR REGISTER
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!NPR!SP0>
        BRO    IDLE
        MICPC=MICPC+1
        <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        OUTPUT IBUS,RCV DAT!OUTDA2  ;OUTPUT A CHAR
        MICPC=MICPC+1
        <MOVE!WROUT!IBUS!<RCV DAT!OUTDA2>>
        STATE  RKE1
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<RKE1-INIT&777/2>>
RCVK02: SP    BR,SELB,SP3           ;SET STATE
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP3>
        BRWRT  IMM,221
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<221>>
        ALWAYS  RK7
        MICPC=MICPC+1
        <JUMP!ALCOND!<RK7-INIT&3000*4>!<RK7-INIT&777/2>>
        SBTTL  RCVK0--PROCESS ODD CHARACTER
RCVK0:  SPBR  IBUS,NPR,SP0          ;IS AN NPR GOING
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!NPR!SP0>
        BRO    IDLE                 ;IF SO, GO BACK TO IDLE LOOP
        MICPC=MICPC+1
        <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        STATE  RCVKE
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<RCVKE-INIT&777/2>>
        SP    BR,SELB,SP3           ;SET STATE
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP3>
        OUTPUT IBUS,RCV DAT!OUTDA2  ;OUTPUT A CHAR
        MICPC=MICPC+1
        <MOVE!WROUT!IBUS!<RCV DAT!OUTDA2>>
RKB:    BRWRT  IMM,21                ;SET OUT NPR (C1) AND NPR REQ
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<21>>
RK7:    OUT    BR,<AORB!ONPR>        ;WRITE NPR REGISTER
        MICPC=MICPC+1
    
```

DMCGAL.MAC 18-JAN-78 16:05

RCVK0--PROCESS ODD CHARACTER

(1)	031356	061310
1091	031360	
(1)		000642
(1)	031360	100447
1092	031362	
1096	031362	
(1)		000643
(1)	031362	123600
1100	031364	
(1)		000644
(1)	031364	000720
1101	031366	
(1)		000645
(1)	031366	104630
1102		
1103	031370	
(1)		000646
(1)	031370	120600
1108	031372	
(1)		000647
(1)	031372	102047
1110	031374	
(1)		000650
(1)	031374	023140
1111	031376	
(1)		000651
(1)	031376	062066
1112	031400	
(1)		000652
(1)	031400	063164
1113	031402	
(1)		000653
(1)	031402	105256
1114	031404	
(1)		000654
(1)	031404	063165
1115	031406	
(1)		000655
(1)	031406	105724
1116	031410	
(1)		000656
(1)	031410	022202
1117	031412	
(1)		000657
(1)	031412	023140
1118	031414	
(1)		000660
(1)	031414	062066
1120	031416	
(1)		000661
(1)	031416	023160
1121	031420	
(1)		000662
(1)	031420	062107
1123	031422	
(1)		000663

```

<MOVE!WROUTX!CR!<AORB!ONPR>>
ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
RL4:
SPBR IBUS,NPR,SPO
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!NPR!SPO>
STATE RCVL
MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<RCVL-INIT&777/2>>
PWAYS RCVK02
MICPC=MICPC+1
<JUMP!ALCOND!<RCVK02-INIT&3000*4>!<RCVK02-INIT&777/2>>
SBTTL RCVKE--HANDLE EVEN BYTES
RCVKF: BRWRITE IBUS,NPR ;READ NPR CONTROL REGISTER
MICPC=MICPC+1
<MOVE!WRITEBR!IBUS!<NPR>>
BRO IDLE
MICPC=MICPC+1
<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
RKS: SP IBUS,IOBA1,SPO ;READ LOW BYTE OF BA TO SP
MICPC=MICPC+1
<MOVE!SPX!IBUS!IOBA1!SPO>
OUTPUT DP<INCA!OBA1> ;WRITE INCREMENTED BA
MICPC=MICPC+1
<MOVE!WROUT!DP!<INCA!OBA1>>
RK50: SP BR,DECA,SP4 ;DECREMENT CHARACTER COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP4>
C 10$ ;NO OVERFLOW
MICPC=MICPC+1
<JUMP!CCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
SP BR,DECA,SP5 ;OVERFLOW - DECREMENT HIGH BYTE
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP5>
Z RL3 ;BYTE COUNT ZERO
MICPC=MICPC+1
<JUMP!ZCOND!<RL3-INIT&3000*4>!<RL3-INIT&777/2>>
10$: OUTPUT IBUS,<RCVDAT!OUTDA1> ;READ CHARACTER AND WRITE IT
MICPC=MICPC+1
<MOVE!WROUT!IBUS!<RCVDAT!OUTDA1>>
SP IBUS,IOBA1,SPO ;READ INCREMENTED BA
MICPC=MICPC+1
<MOVE!SPX!IBUS!IOBA1!SPO>
OUTPUT DP<INCA!OBA1> ;WRITE INCREMENTED BA
MICPC=MICPC+1
<MOVE!WROUT!DP!<INCA!OBA1>>
SP IBUS,IOBA2,SPO ;GET BA HIGH
MICPC=MICPC+1
<MOVE!SPX!IBUS!IOBA2!SPO>
OUTPUT DP<AC!OBA2> ;INCREMENT IF CARRY SET
MICPC=MICPC+1
<MOVE!WROUT!DP!<AC!OBA2>>
C ICBA23 ;OVER FLOW
MICPC=MICPC+1

```

DMCGAL.MAC 18-JAN-78 16:05

RCVKE--HANDLE EVEN BYTES

1 031422 115016
 1124 031424 000664
 (1) 031424 063164
 1125 031426 000665
 (1) 031426 105270
 1126 031430 000666
 (1) 031430 063165
 1127 031432 000667
 (1) 031432 105643
 1135 031434 000670
 (1) 031434 000633
 1136 031436 000671
 (1) 031436 104425
 1138
 1139 031440 000672
 (1) 031440 123200
 1143 031442 000673
 (1) 031442 000577
 1144 031444 000674
 (1) 031444 061270
 1145 031446 000675
 (1) 031446 104652
 1146
 1147
 1148 031450 000676
 (1) 031450 023200
 1149 031452 000677
 (1) 031452 062202
 1150 031454 000700
 (1) 031454 060601
 1151 031456 000701
 (1) 031456 117571
 1152 031460 000702
 (1) 031460 104664

```

    < JUMP!CCOND!<ICBA23-INIT&3000*4>!<ICBA23-INIT&777/2>>
RK3:  SP      BR DECA,SP4          ;DECREMENT THE COUNT OF BYTES
      MICPC=MICPC+1
      < MOVE!SPX!BR!DECA!SP4>
      C      RK6                    ;NO OVERFLOW
      MICPC=MICPC+1
      < JUMP!CCOND!<RK6-INIT&3000*4>!<RK6-INIT&777/2>>
      SP      BR DECA,SP5          ;DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      < MOVE!SPX!BR!DECA!SP5>
      Z      RL4                    ;BYTE COUNT ZERO
      MICPC=MICPC+1
      < JUMP!ZCOND!<RL4-INIT&3000*4>!<RL4-INIT&777/2>>
RK6:  STATE  RCVKO
      MICPC=MICPC+1
      < MOVE!WRTBR!IMM!<RCVKO-INIT&777/2>>
      ALWAYS REXIT
      MICPC=MICPC+1
      < JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
RK61: SP      IBUS NPR,SP0         ;READ NPR REGISTER
      MICPC=MICPC+1
      < MOVE!SPX!IBUS!NPR!SP0>
      BRWRT  IMM,177              ;MASK FOR ALL BUT CO
      MICPC=MICPC+1
      < MOVE!WRTBR!IMM!<177>>
      OUT    BR <AANDB!ONPR>      ;TURN OFF ALL BUT CO
      MICPC=MICPC+1
      < MOVE!WROUT!BR!<AANDB!ONPR>>
      ALWAYS RK50
      MICPC=MICPC+1
      < JUMP!ALCOND!<RK50-INIT&3000*4>!<RK50-INIT&777/2>>
;*****END OF TIME CRITICAL PATH*****
RCVKE0: SP      IBUS RCVDAT,SP0    ;READ CHARACTER AND SAVE IN SPO
      MICPC=MICPC+1
      < MOVE!SPX!IBUS!RCVDAT!SPO>
      OUTPUT BR <SELA!OUTDA1>    ;SEND NONSENSE CHARACTER
      MICPC=MICPC+1
      < MOVE!WROUT!BR!<SELA!OUTDA1>>
      BRWRT  BR SELA!SP1         ;READ STATUS BYTE
      MICPC=MICPC+1
      < MOVE!WRTBR!BR!<SELA!SP1>>
      BR7    PASWRD              ;MAINT MODE - SEE IF RLD MESSAGE
      MICPC=MICPC+1
      < JUMP!BR7CON!<PASWRD-INIT&3000*4>!<PASWRD-INIT&777/2>>
      ALWAYS RK3                 ;OTHERWISE PROCESS NORMALLY
      MICPC=MICPC+1
      < JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>
    
```

DMCGAL.MAC 18-JAN-78 16:05

RCVI--STORE UNNUMBERED MESSAGE TYPE

1154
1155 031462 000703
(1) 031462 023213
1156 031464 000704
(1) 031464 000706
1157 031466 000705
(1) 031466 104425
1158
1159
1160 031470
1162 031470
(1) 000706
(1) 031470 104756
1170 031472 000707
(1) 031472 000711
1171 031474 000710
(1) 031474 104425
1172
1173
1174
1175 031476 000711
(1) 031476 000403
1176 031500 000712
(1) 031500 060353
1177 031502 000713
(1) 031502 000716
1178
1179 031504 000714
(1) 031504 105140
1180 031506 000715
(1) 031506 104471
1181
1182
1183
1184 031510 000716
(1) 031510 000534
1185 031512 000717
(1) 031512 104540

```

.SBTTL RCVI--STORE UNNUMBERED MESSAGE TYPE
RCVI:  SP      IBUS RCV DAT, SP13      ;STORE UNNUMBERED TYPE
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!RCV DAT!SP13>
      STATE   RCVJ      ;NEXT STATE IS J
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVJ-INIT&777/2>>
      ALWAYS  REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
      ;
.SBTTL RCVJ--ROUTINE TO HANDLE SUBTYPE FIELD,SELECT AND FINAL
RCVJ:  ALWAYS  SELQSY      ;"CALL" SELECT AND QSYNC SUBROUTINE
      MICPC=MICPC+1
      <JUMP!ALCOND!<SELQSY-INIT&3000*4>!<SELQSY-INIT&777/2>>
      STATE   RCVR      ;NEXT STATE IS N
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVR-INIT&777/2>>
      ALWAYS  REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
      .SBTTL RCVR--UNNUMBERED MESSAGE RESPONSE FIELD
      ;ENTERED FROM IDLE LOOP
      ;
RCVR:  BRWRITE IMM,3      ;REP MESSAGE TYPE TO BR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<3>>
      NOP      BR SUB,SP13      ;IS TYPE ACK OR NAK
      MICPC=MICPC+1
      <BR!SUB!SP13>
      STATE   RCVQ      ;NEXT STATE IS RCVQ
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVQ-INIT&777/2>>
      ;***NOTE THIS INSTR DOES NOT CLOCK "C"
      ;IF NOT IGNORE
      C      RCVF1
      MICPC=MICPC+1
      <JUMP!CCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>
      ALWAYS  RD2      ;DO RANGE CHECKS
      MICPC=MICPC+1
      <JUMP!ALCOND!<RD2-INIT&3000*4>!<RD2-INIT&777/2>>
      .SBTTL RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD
      ;ENTER FROM IDLE
      ;
RCVQ:  STATE   RCVF      ;NEXT STATE IS ADDRESS
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVF-INIT&777/2>>
      ALWAYS  RCVF1
      MICPC=MICPC+1
      <JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>

```

DMCGAL.MAC 18-JAN-78 16:05

RCVL--PROCESS CRC3

1187			.SBTTL RCVL--PROCESS CRC3
1188			:ENTERED FROM IDLE LOOP
1189	031514		RCVL: SPBR IBUS,NPP,SPO ;READ NPR CONTROL
(1)		000720	MICPC=MICPC+1
(1)	031514	123600	<MOVE!SPBRX!IBUS!NPR!SPO>
1194	031516		BRO IDLE
(1)		000721	MICPC=MICPC+1
(1)	031516	102047	<JUMP!BROCON!<IDLE-INIT&3000*4!<IDLE-INIT&777/2>>
1196	031520		RL2: BRWRT IMM,176 ;MASK TO TURN OFF CO
(1)		000722	MICPC=MICPC+1
(1)	031520	000576	<MOVE!WRTEBR!IMM!<176>>
1197	031522		OUT BR,AANDB!ONPR
(1)		000723	MICPC=MICPC+1
(1)	031522	061270	<MOVE!WROUTX!BR!<AANDB!ONPR>>
1198			.
1199	031524		RL3: STATE RCVM
(1)		000724	MICPC=MICPC+1
(1)	031524	000726	<MOVE!WRTEBR!IMM!<RCVM-INIT&777/2>>
1200	031526		ALWAYS RCVF1
(1)		000725	MICPC=MICPC+1
(1)	031526	104540	<JUMP!ALCOND!<RCVF1-INIT&3000*4!<RCVF1-INIT&777/2>>
1201			.
1206			.SBTTL RCVM--PROCESS CRC4--END OF DATA MESSAGE
1207			:ENTERED FROM IDLE LOOP
1208			:IF CRC CORRECT -- QUEUE INTERRUPT AND UPDATE RESPONSE
1209			.
1210			:IF CRC WRONG SEND NAK
1211	031530		RCVM: BRWRT IBUS,UBBR ;READ UNIBUS BR REGISTER
(1)		000726	MICPC=MICPC+1
(1)	031530	120620	<MOVE!WRTEBR!IBUS!<UBBR>>
1212	031532		BRO NXMERR ;NON-EXISTANT MEMORY
(1)		000727	MICPC=MICPC+1
(1)	031532	116051	<JUMP!BROCON!<NXMERR-INIT&3000*4!<NXMERR-INIT&777/2>>
1213	031534		SP IBUS,RCVDAT,SPO ;READ CRC CHARACTER
(1)		000730	MICPC=MICPC+1
(1)	031534	023200	<MOVE!SPX!IBUS!RCVDAT!SPO>
1214	031536		BRWRT IBUS,RCVCON ;READ RECEIVER CONTROL REGISTER
(1)		000731	MICPC=MICPC+1
(1)	031536	020640	<MOVE!WRTEBR!IBUS!<RCVCON>>
1215	031540		BRO RCVM1 ;IF CRC GOOD -- PROCESS
(1)		000732	MICPC=MICPC+1
(1)	031540	116206	<JUMP!BROCON!<RCVM1-INIT&3000*4!<RCVM1-INIT&777/2>>
1216	031542		BRWRT BR,SELA!SP1 ;READ STATUS BYTE
(1)		000733	MICPC=MICPC+1
(1)	031542	060601	<MOVE!WRTEBR!BR!<SELA!SP1>>
1217	031544		BR7 RHX ;CRC ERROR IN BOOT MODE - FLUSH
(1)		000734	MICPC=MICPC+1
(1)	031544	107746	<JUMP!BR7CON!<RHX-INIT&3000*4!<RHX-INIT&777/2>>
1218	031546		BRWRT IMM,302 ;DATA ERROR SUBTYPE
(1)		000735	MICPC=MICPC+1
(1)	031546	000702	<MOVE!WRTEBR!IMM!<302>>
1219	031550		LDMA IMM,NOATS ;ADDRESS CUM ERROR COUNTER
(1)		000736	MICPC=MICPC+1
(1)	031550	010014	<MOVE!LDMAR!IMM!<NOATS&377>>
1220	031552		ALWAYS RHS ;SEND NAK
(1)		000737	MICPC=MICPC+1

DMCGAL.MAC 18-JAN-78 16:05

RCVM--PROCESS CRC4--END OF DATA MESSAGE

```

(1) 031552 104560
1221
1222 031554 000740
(1) 031554 000410
1223 031556 000741
(1) 031556 010177
1224 031560 000742
(1) 031560 114507
1225
1226
1227
1228
1229 031562 000743
(1) 031562 020600
1230 031564 000744
(1) 031564 060373
1231 031566 000745
(1) 031566 105754
1232
1233 031570 000746
(1) 031570 060521
1234 031572 000747
(1) 031572 107351
1235 031574 000750
(1) 031574 104415
1236 031576 000751
(1) 031576 000565
1237 031600 000752
(1) 031600 063261
1238 031602 000753
(1) 031602 104415
1239 031604 000754
(1) 031604 063164
1240 031606 000755
(1) 031606 100447

<JUMP!ALCOND!<RHS-INIT&3000*4>!<RHS-INIT&777/2>>
RCVM0: BRWRT IMM,10 ;MAINT MESS ERROR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<10>>
RCVM2: LDMA IMM,<<RTHRS+3>> ;POINT TO ERROR WORD
MICPC=MICPC+1
<MOVE!LDMA!IMM!<<RTHRS+3>&377>>
ALWAYS RCEXY ;GIVE FATAL ERROR
MICPC=MICPC+1
<JUMP!ALCOND!<RCEXY-INIT&3000*4>!<RCEXY-INIT&777/2>>
.SBTTL EM2--PROCESS RLD MESSAGE
;ENTERED FROM IDLE LOOP
;IF RLD PASSWORD CHECKS TRIGGER THE BOOT ROM
EM2: BRWRT IBUS,RCV DAT ;READ THE CHAR
MICPC=MICPC+1
<MOVE!WRTEBR!IBUS!<RCV DAT>>
CMP BR,SP13 ;IS IT A MATCH
MICPC=MICPC+1
<SUBTC!BR!SP13>
Z EM3
MICPC=MICPC+1
<JUMP!ZCOND!<EM3-INIT&3000*4>!<EM3-INIT&777/2>>
;FALL INTO RHX
RHX: BRWRT BR,AA!SP1 ;READ STATUS BYTE SHIFTED LEFT
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AA!SP1>>
BR4 10$ ;DLE RECEIVED IN NORMAL MODE
MICPC=MICPC+1
<JUMP!BR4CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
ALWAYS FLUSH ;ALREADY IN MAINT MODE
MICPC=MICPC+1
<JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
10$: BRWRT IMM,165 ;MASK TO CLEAR ALL MAINT RELATED BITS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<165>>
SP BR,AANDB,SP1 ;CLEAR THEM
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SP1>
ALWAYS FLUSH
MICPC=MICPC+1
<JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
EM3: SP BR,DECA,SP4
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP4>
ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

```

DMCGAL.MAC 18-JAN-78 16:05

SELQSY--ROUTINETOCHECK SELECT AND QSYNC AND DIDDLE LINE STATUS WORD

1252
1253
1254 031610 000756
(1) 031610 023605
1255 031612 000757
(1) 031612 107763
1256 031614 000760
(1) 031614 060525
1257 031616 000761
(1) 031616 107766
1258 031620 000762
(1) 031620 164463
1259
1260 031622 000763
(1) 031622 000600
1261 031624 000764
(1) 031624 063310
1262 031626 000765
(1) 031626 104760
1263 031630 000766
(1) 031630 000420
1264 031632 000767
(1) 031632 063310
1265 031634 000770
(1) 031634 164463
1267
1275 031636 000771
(1) 031636 000402
1276 031640 000772
(1) 031640 114657
1277
1279 031642 000773
(1) 031642 060530
1280 031644 000774
(1) 031644 103365
1281 031646 000775
(1) 031646 107422
1282 031650 000776
(1) 031650 100447

```
.SBTTL SELQSY--ROUTINETOCHECK SELECT AND QSYNC AND DIDDLE LINE STATUS WORD
:USES SPS. ALWAYS CALLED BY FIRST INSTR IN A RSTATE
SELQSY: SPBR IBUS,RCVDAT,SPS ;READCHARACTERINTO SPS AND THE BR
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!RCVDAT!SPS>
BR7 155 ;SELECT SET?--BRANCH
MICPC=MICPC+1
<JUMP!BR7CON!<155-INIT&3000*4>!<155-INIT&777/2>>
SS: BRWRT BR,AA!SPS ;SHIFTBR LEFT
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AA!SPS>>
BR7 205 ;FINAL SET?
MICPC=MICPC+1
<JUMP!BR7CON!<205-INIT&3000*4>!<205-INIT&777/2>>
.ALWAY BR,INCA,SP3!PAGE1
MICPC=MICPC+1
<JUMP!ALCOND!BR!INCA!SP3!PAGE1>
ISS: BRWRT IMM,200 ;SET OK TO SEND
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<200>>
SP BR,AORB,SP10 ;IN LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP10>
.ALWAY SS
MICPC=MICPC+1
<JUMP!ALCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
205: BRWRT IMM,20 ;SETCLEARACTIVE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<20>>
SP BR,AORB,SP10 ;IN LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP10>
.ALWAY BR,INCA,SP3!PAGE1
MICPC=MICPC+1
<JUMP!ALCOND!BR!INCA!SP3!PAGE1>
RTHRES: BRWRT IMM,2
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>>
.ALWAY ERRXX
MICPC=MICPC+1
<JUMP!ALCOND!<ERRXX-INIT&3000*4>!<ERRXX-INIT&777/2>>
ACK: BRWRT BR,AA!SP10
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AA!SP10>>
BR4 ACK1
MICPC=MICPC+1
<JUMP!BR4CON!<ACK1-INIT&3000*4>!<ACK1-INIT&777/2>>
BR7 FLUSHA
MICPC=MICPC+1
<JUMP!BR7CON!<FLUSHA-INIT&3000*4>!<FLUSHA-INIT&777/2>>
.ALWAY IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
```

I08

DMCGAL.MAC

18-JAN-78 16:05

SELQSY--ROUTINETOCHECK SELECT AND QSYNC AND DIDDLE LINE STATUS WORD

PAGE: 0099DM
SEQ 0099

1284

:

JOB

DMCGAL.MAC 18-JAN-78 16:05

SELQSY--ROUTINETOCHECK SELECT AND QSYNC AND DIDDLE LINE STATUS WORD

PAGE: 0100DM
SEQ 0100

1286	031654	. =INIT+2000
1287	000777	MICPC=777
1288		.SBTTL TMTA--FIRST CHARACTER OF HEADER
1289		;
1290	031654	TMTA: BRWRT BR,AA!SP10 ;SHIFT LEFT
1291	031654	MICPC=MICPC+1
(1)	001000	<MOVE!WRTEBR!BR!<AA!SP10>>
(1)	031654	060530
1292	031656	BR7 RCVCK
(1)	001001	MICPC=MICPC+1
(1)	031656	107620
1293	031660	TA1: <JUMP!BR7CON!<RCVCK-INIT&3000*4>!<RCVCK-INIT&777/2>>
1294	031660	BRWRT BR,SELA!SP10 ;REREAD STATUS
(1)	001002	MICPC=MICPC+1
(1)	031660	060610
1295	031662	<MOVE!WRTEBR!BR!<SELA!SP10>>
(1)	001003	BR7 TA2 ;OK TO SEND?
(1)	031662	113405
1296	031664	MICPC=MICPC+1
(1)	001004	<JUMP!BR7CON!<TA2-INIT&3000*4>!<TA2-INIT&777/2>>
(1)	031664	100451
1297	031666	TMTA6: ALWAYS I1 ;NO
(1)	001005	MICPC=MICPC+1
(1)	031666	112032
1298	031670	<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
(1)	001006	TA2: BRO NUMSYN ;IF UNNUMBPENDING -- SEND IT
(1)	031670	112410
1299	031672	MICPC=MICPC+1
(1)	001007	<JUMP!BROCON!<NUMSYN-INIT&3000*4>!<NUMSYN-INIT&777/2>>
(1)	031672	100451
1300	031674	BR1 10\$
1301	031674	MICPC=MICPC+1
(1)	001010	<JUMP!BR1CON!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>
(1)	031674	001620
1302	031676	ALWAYS I1
(1)	001011	MICPC=MICPC+1
(1)	031676	103051
1304	031700	<JUMP!BR4CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
(1)	001012	BR1 NUMSYN ;IF LINE HAS GONE IDLE SEND SYN
(1)	031700	112432
1305		MICPC=MICPC+1
1316	031702	<JUMP!BR1CON!<NUMSYN-INIT&3000*4>!<NUMSYN-INIT&777/2>>
(1)	001013	TMTEXT: TSTATE TMTB ;ELSE--START TO SEND MESSAGE
(1)	031702	000450
(1)	001014	<STATE TO SFND COUNT
(1)	031704	063222
1317	031706	MICPC=MICPC+1
(1)	001015	<MOVE!SPX!BR!SELB!SP2>
(1)	031706	060521
1318	031710	BRWRT BR,AA!SP1 ;IN PORT STATUS
(1)	001016	MICPC=MICPC+1
(1)	031710	113021
1319	031712	<MOVE!WRTEBR!BR!<AA!SP1>>
(1)	001017	BR4 10\$;DLE RCV'D SET-IGNORE MAINT MODE BIT
(1)	031712	000620
1320	031714	MICPC=MICPC+1
		<JUMP!BR4CON!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>
		BRWRT IMM,220 ;WRITE DLE TO BR
		<MOVE!WRTEBR!IMM!<220>>
		C TMTAS ;MAINT MODE - SEND DLE

DMCGAL.MAC 18-JAN-78 16:05

TMTA--FIRST CHARACTER OF HEADER

(1)		001020	MICPC=MICPC+1
(1)	031714	111024	<JUMP!CCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>
1321	031716		BRWRTE BR, SELA!SP10 ;NUMBERED MESSAGE?
(1)		001021	MICPC=MICPC+1
(1)	031716	060610	<MOVE!WRTEBR!BR!<SELA!SP10>>
1322	031720		BRD TMTUN ;NO - BRANCH
(1)		001022	MICPC=MICPC+1
(1)	031720	112026	<JUMP!BROCON!<TMTUN-INIT&3000*4>!<TMTUN-INIT&777/2>>
1323	031722		BRWRTE IMM,201 ;OTHERWISE SEND AN SOH
(1)		001023	MICPC=MICPC+1
(1)	031722	000601	<MOVE!WRTEBR!IMM!<201>>
1324	031724		TMTAS: OUTPUT BR, <SELB!TMTDAT> ;IN TMT SILO
(1)		001024	MICPC=MICPC+1
(1)	031724	062230	<MOVE!WROUT!BR!<SELB!TMTDAT>>
1325	031726		ALWAYS I1
(1)		001025	MICPC=MICPC+1
(1)	031726	100451	<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
1326	031730		TMTUN: TSTATE TMTI
(1)		001026	MICPC=MICPC+1
(1)	031730	000601	<MOVE!WRTEBR!IMM!<TMTI-INIT&777/2>>
(1)		001027	MICPC=MICPC+1
(1)	031732	063222	<MOVE!SPX!BR!SELB!SP2>
1327	031734		BRWRTE IMM,5 ;ENQ TO BR
(1)		001030	MICPC=MICPC+1
(1)	031734	000405	<MOVE!WRTEBR!IMM!<5>>
1328	031736		ALWAYS TMTAS
(1)		001031	MICPC=MICPC+1
(1)	031736	110424	<JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>

DMCGAL.MAC 18-JAN-78 16:05

TMTA--FIRST CHARACTER OF HEADER

1331 031740 001032
(1)
(1) 031740 020660
1332 031742 001033
(1)
(1) 031742 001620
1333 031744 001034
(1)
(1) 031744 103051
1334 031746 001035
(1)
(1) 031746 000773
1335 031750 001036
(1)
(1) 031750 063270
1336 031752 001037
(1)
(1) 031752 000443
(1) 031754 001040
(1) 031754 063222
1337 031756 001041
(1)
(1) 031756 000410
1338 031760 001042
(1)
(1) 031760 063226
1339 031762 001043
(1)
(1) 031762 063166
1340 031764 001044
(1)
(1) 031764 111413
1341 031766 001045
(1)
(1) 031766 002011
1342 031770 001046
(1)
(1) 031770 000626
1343 031772 001047
(1)
(1) 031772 110424

```

NUMSYN: BRWRT  IBUS,MODEM           ;ARE WE STILL SENDING?
        MICPC=MICPC+1
        <MOVE!WRTEBR!IBUS!<MODEM>>
        BRSHFT
        MICPC=MICPC+1
        <MOVE!SHFTBR!WRTEBR!SELB>
        BR4      I1                   ;RTS SET? IF SO WE ARE--STALL
        MICPC=MICPC+1
        <JUMP!BR4CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
        BRWRT  IMM,373                 ;MASK TO TURN OFFLINE IDLE
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<373>>
        SP      BR  AANDB,SP10         ;IN LINE STATUS WORD
        MICPC=MICPC+1
        <MOVE!SPX!BR!AANDB!SP10>
        TSTATE TMTA1
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<TMTA1 INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP2>
        BRWRT  IMM,10
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<10>>
        SP      BR  SELB,SP6           ;STORE IN SP6
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP6>
TMTA1:  SP      BR  DECA,SP6           ;DECREMENT SYN COUNT
        MICPC=MICPC+1
        <MOVE!SPX!BR!DECA!SP6>
        Z      TMTXT
        MICPC=MICPC+1
        <JUMP!ZCOND!<TMTXT-INIT&3000*4>!<TMTXT-INIT&777/2>>
        OUTPUT IMM,<1!OTMTCO>         ;WRITE SOM TO TMTR CONTRL
        MICPC=MICPC+1
        <MOVE!WROUT!IMM!<1!OTMTCO>>
        BRWRT  IMM,226                 ;SYNC CHAR
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<226>>
        ALWAYS TMTA5
        MICPC=MICPC+1
        <JUMP!ALCOND!<TMTA5 INIT&3000*4>!<TMTA5-INIT&777/2>>
    
```

M08

DMCGAL.MAC
1355

18-JAN-78 16:05

TMTB--OUTPUT FIRST CHAR OF COUNT

.SBTTL TMTB--OUTPUT FIRST CHAR OF COUNT

PAGE: 0103DM
SEQ 0103

N08

DMCGAL.MAC 18-JAN-78 16:05

TMTB--OUTPUT FIRST CHAR OF COUNT

PAGE: 01040M
SEQ 0104

1357
1358 031774
(1) 001050
(1) 031774 063472

TMTB: SPBR BR,INCA,SP12
MICPC=MICPC+1
<MOVE!SPBRX!BR!INCA!SP12>

: INCREMENT MSG NO

DMCGAL.MAC 18-JAN-78 16:05

TMTB--OUTPUT FIRST CHAR OF COUNT

1360	031776	001051	OUTPUT BR SELB!OINDAT1	
(1)		062220	MICPC=MICPC+1	
(1)	031776		<MOVE!WROUT!BR!<SELB!OINDAT1>>	
1361	032000		LDMA BR SELA!SP16	;GETPOINTER TO NEXT TMT LINK
(1)		001052	MICPC=MICPC+1	
(1)	032000	07021E	<MOVE!LDMAR!BR!<SELA!SP16>>	
1362	032002		MEMINC IMM 3	;WRITE MSG TMTED TO FLAGS
(1)		001053	MICPC=MICPC+1	
(1)	032002	016403	<MOVE!WRMEM!INCMAR!IMM!<3>>	
1363	032004		MEMINC BR SELA!SP12	;PICK UP MSGNO
(1)		001054	MICPC=MICPC+1	
(1)	032004	076612	<MOVE!WRMEM!INCMAR!BR!<SELA!SP12>>	
1364	032006		BRWTE IMM TML8	;GET WRAPAROUND ADDRESS
(1)		001055	MICPC=MICPC+1	
(1)	032006	000543	<MOVE!WRTEBR!IMM!<TML8>>	
1365	032010		CMP BR SP16	;WRAPAROUND
(1)		001056	MICPC=MICPC+1	
(1)	032010	060376	<SUBTC!BR!SP16>	
1366	032012		Z TB2	
(1)		001057	MICPC=MICPC+1	
(1)	032012	111504	<JUMP!ZCOND!<TB2-INIT&3000*4>!<TB2-INIT&777/2>>	

DMCCGL.MAC 18-JAN-78 16:05

TMTB--OUTPUT FIRST CHAR OF COUNT

1368	032014		BRWRT IMM,6	:OFFSET TO NEXT LINK
(1)		001060	MICPC=MICPC+1	
(1)	032014	000406	<MOVE!WRTEBR!IMM!<6>>	
1369	032016		TB1: SP BR,ADD,SP16	:UPDATE THE POINTER
(1)		001061	MICPC=MICPC+1	
(1)	032016	063016	<MOVE!SPX!BR!ADD!SP16>	
1370	032020		STATE TMT	:ADDRESS TMT STATE
(1)		001062	MICPC=MICPC+1	
(1)	032020	000506	<MOVE!WRTEBR!IMM!<TMT-INIT&777/2>>	
1371	032022		TB0: SP BR,SELB,SP2	:UPDATE IT
(1)		001063	MICPC=MICPC+1	
(1)	032022	063222	<MOVE!SPX!BR!SELB!SP2>	
1372	032024		OUTPUT <MEMX!INCMAR>,SELB!IBA1	:WRITE LOW BYTE OF ADDRESS
(1)		001064	MICPC=MICPC+1	
(1)	032024	056224	<MOVE!WROUT!MEMX!INCMAR!<SELB!IBA1>>	
1373	032026		OUTPUT <MEMX!INCMAR>,SELB!IBA2	:WRITE HIGH BYTE OF ADDRESS
(1)		001065	MICPC=MICPC+1	
(1)	032026	056225	<MOVE!WROUT!MEMX!INCMAR!<SELB!IBA2>>	
1374	032030		SP MEMX,SELB,SP7 ;HIGH BYTE OF COUNT TO SP7	
(1)		001066	MICPC=MICPC+1	
(1)	032030	043227	<MOVE!SPX!MEMX!SELB!SP7>	
1375				:WAIT TO MASK OFF MEM EXT. BITS
1376	032032		SP IBUS,NPR,SP0	
(1)		001067	MICPC=MICPC+1	
(1)	032032	123200	<MOVE!SPX!IBUS!NPR!SP0>	
1377	032034		BRWRT IMM,220	
(1)		001070	MICPC=MICPC+1	
(1)	032034	000620	<MOVE!WRTEBR!IMM!<220>>	
1378	032036		SP BR,AANDB,SP0	
(1)		001071	MICPC=MICPC+1	
(1)	032036	063260	<MOVE!SPX!BR!AANDB!SP0>	
1379	032040		SP IMM,300,SP6	:MASK FOR MXT
(1)		001072	MICPC=MICPC+1	
(1)	032040	003306	<MOVE!SPX!IMM!300!SP6>	
1380	032042		BRWRT MEMX!INCMAR,AANDB!SP6	:TURN OFF CC2
(1)		001073	MICPC=MICPC+1	
(1)	032042	054666	<MOVE!WRTEBR!MEMX!INCMAR!<AANDB!SP6>>	
1381	032044		OUTPUT MEMX,SELB!TMTDAT	:ALSO WRITE COUNT TO TMT SILO
(1)		001074	MICPC=MICPC+1	
(1)	032044	042230	<MOVE!WROUT!MEMX!<SELB!TMTDAT>>	
1382	032046		BRSHFT	:SHIFT BITS INTO CORRECT POSITION
(1)		001075	MICPC=MICPC+1	
(1)	032046	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
1383	032050		BRSHFT	
(1)		001076	MICPC=MICPC+1	
(1)	032050	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
1384	032052		BRSHFT	
(1)		001077	MICPC=MICPC+1	
(1)	032052	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
1385	032054		BRSHFT	
(1)		001100	MICPC=MICPC+1	
(1)	032054	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
1386	032056		OUT BR,AORB!ONPR	
(1)		001101	MICPC=MICPC+1	
(1)	032056	061310	<MOVE!WROUTX!BR!<AORB!ONPR>>	
1387	032060		SPBR MEMX,SELB,SP6	:LOWBYTE OF COUNT TO SP6

CMC3AL.MAC 18-JAN-78 16:05

TMTB--OUTPUT FIRST CHAR OF COUNT

1		001102
(1)	032060	043626
1389	032062	
(1)		001103
(1)	032062	100447
1394	032064	
(1)		001104
(1)	032064	000726
1395	032066	
(1)		001105
(1)	032066	110461
1396		

```

MICPC=MICPC+1
<MOVE!SPBRX!MEMX!SELB!SP6>
ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
TB2: BRWRT IMM,<<TML1-TML8>&377> ;GO BACK TOFIRST LINK
MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<<TML1-TML8>&377>>
ALWAYS TB1
MICPC=MICPC+1
<JUMP!ALCOND!<TB1-INIT&3000*4>!<TB1-INIT&777/2>>
;

```


DMCGAL.MAC 18-JAN-78 16:05

TMTD--OUTPUT SECOND CHAR OF COUNT

```

1398
1399
1400 032070 001106
(1) 032070 000477
1401 032072 001107
(1) 032072 063667
1402 032074 001110
(1) 032074 062230
1404 032076 001111
(1) 032076 000513
1405 032100 001112
(1) 032100 110554
1407
1408
1409 032102 001113
(1) 032102 000522
1410 032104 001114
(1) 032104 063166
1411 032106 001115
(1) 032106 111117
1412 032110 001116
(1) 032110 063167
1413 032112 001117
(1) 032112 010171
1414 032114 001120
(1) 032114 042230
1415 032116 001121
(1) 032: 6 110554
1416
1417 032120
1418 032120 001122
(1) 032120 123600
1419 032122 001123
(1) 032122 102051
1420 032124 001124
(1) 032124 022010
1421 032126 001125
(1) 032126 000527
1422 032130 001126

```

```

.SBTTL TMTD--OUTPUT SECOND CHAR OF COUNT
TMTD: BRWRT IMM,77 ;MASK TO CLEAR MXT BITS
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<77>>
      SPBR BP,AANDB,SP7 ;CLEAR THEM
      MICPC=MICPC+1
      <MOVE!SPBRX!BR!AANDB!SP7>
      OUTPUT DP<SELB!TMTDAT> ;WRITE TO TMT SILO
      MICPC=MICPC+1
      <MOVE!WROUT!DP!<SELB!TMTDAT>>
      STATE TMTD
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTD-INIT&777/2>>
      ALWAYS XEXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
      ;
.SBTTL TMTD--RESPONSE FIELD-NUMBERED MESSAGE
TMTD: STATE TMTE
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTE-INIT&777/2>>
      SP BR,DECA,SP6 ;ADJUST COUNT FOR TWO'S COMPLEMENT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP6>
      C TD2 ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<TD2-INIT&3000*4>!<TD2-INIT&777/2>>
      SP BR,DECA,SP7 ;DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP7>
      LDMA IMM,ISP11 ;RESP FIELD ADDR TO MAR
      MICPC=MICPC+1
      <MOVE!LDMA!IMM!<ISP11&377>>
      OUTPUT MEMX,SELB!TMTDAT ;WRITE IT TO SILO
      MICPC=MICPC+1
      <MOVE!WROUT!MEMX!<SELB!TMTDAT>>
      ALWAYS XEXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
      .SBTTL TMTE--NUMBER FIELD--NUMBERED MESSAGE
TMTE: SPBR IBUS,NPR,SPO ;READ NPR CONTROL REGISTER
      MICPC=MICPC+1
      <MOVE!SPBRX!IBUS!NPR!SPO>
      BRO I1 ;BUSY - GET OUT
      MICPC=MICPC+1
      <JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
      OUTPUT IBUS,<INDAT1!TMTDAT> ;WRITE IT TO THE SILO
      MICPC=MICPC+1
      <MOVE!WROUT!IBUS!<INDAT1!TMTDAT>>
      STATE TMTF
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTF-INIT&777/2>>
      ALWAYS TH3
      MICPC=MICPC+1

```

DMCGAL.MAC 18-JAN-78 16:05

TMTE--NUMBER FIELD--NUMBERED MESSAGE

```

1) 032130 110574
1423
1424
1425 032132 001127
(1) 032132 000532
1426 032134 001130
(1) 032134 063222
1427 032136 001131
(1) 032136 000401
1433 032140 001132
(1) 032140 110424
1435
1436 032142 001133
(1) 032142 000402
1437 032144 001134
(1) 032144 062231
1438 032146 001135
(1) 032146 062230
1439 032150 001136
(1) 032150 020500
1440 032152 001137
(1) 032152 112153
1441 032154 001140
(1) 032154 000542
1442 032156 001141
(1) 032156 110554
    
```

```

< JUMP!ALCOND!<TH3-INIT&3000*4>!<TH3-INIT&777/2>>
.SBTTL TMTE--NUMBERED MSG ADDRESS FIELD
TMTE:
STATE TF1
MICPC=MICPC+1
< MOVE!WRTEBR!IMM!<TF1-INIT&777/2>>
TF2: SP BR,SELB,SP2
MICPC=MICPC+1
< MOVE!SPX!BR!SEL9!SP2>
BRWRT IMM,1 ;LOAD ADDRESS
MICPC=MICPC+1
< MOVE!WRTEBR!IMM!<1>>
ALWAYS TMTAS
MICPC=MICPC+1
< JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>
.SBTTL TF1-NUMBERED MSG HEADER EOM
TF1: BRWRT IMM,2 ;EOM MASK TO BR
MICPC=MICPC+1
< MOVE!WRTEBR!IMM!<2>>
OUTPUT BR,<SELB!OTMTCO> ;UPDATE TMTR CONTROL REGISTER
MICPC=MICPC+1
< MOVE!WROUT!BR!<SELB!OTMTCO>>
OUTPUT BR,<SELB!TMTDAT> ;OUTPUT A GARBAGE CHAR
MICPC=MICPC+1
< MOVE!WROUT!BR!<SELB!TMTDAT>>
BRWRT IBUS,IIBA1 ;READ LOW ORDER FROM INBA
MICPC=MICPC+1
< MOVE!WRTEBR!IBUS!<IIBA1>>
BRO TMTF1 ;IF ODD BYTE--BRANCH
MICPC=MICPC+1
< JUMP!BROCON!<TMTF1-INIT&3000*4>!<TMTF1-INIT&777/2>>
STATE TMTH
MICPC=MICPC+1
< MOVE!WRTEBR!IMM!<TMTH-INIT&777/2>>
ALWAYS XEXIT
MICPC=MICPC+1
< JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
    
```

SMOJAL MAC 18-JAN-78 16:05

TF1-NUMBERED MSG HEADER EOM

```

:*****TIME CRITICAL PATH--MODIFY WITH GREAT CARE
:SBTTL TMTM--ROUTINE TO OUTPUT DATA CHARACTERS
TMTM: SPBR IBUS,NPR,SPO ;READ NPR CONTROL
      MI PC=MICPC+1
      <MOVE!SPBRX!IBUS!NPR!SPO>
      BRO I1 ;IF NPR IN PROGRESS --BRANCH
      MICPC=MICPC+1
      <JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
S3: OUTPUT IBUS<INDAT1!TMTDAT> ;WRITE THE EVEN CHAR TO TMT SILO
      MICPC=MICPC+1
      <MOVE!WROUT!IBUS!<INDAT1!TMTDAT>>
      SP IBUS,IIBA1,SPO ;READ LOW BYTE OF BA TO SP
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IIBA1!SPO>
      OUTPUT BR,<INCA!IBA1> ;OUTPUT INCREMENTED BA
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<INCA!IBA1>>
      SP BR,DECA,SP6 ;DECREMENT CHARACTER COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP6>
      C TH6 ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<TH6-INIT&3000*4>!<TH6-INIT&777/2>>
      SP BR,DECA,SP7 ;DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP7>
      Z HEH1 ;BYTE COUNT ZERO
      MICPC=MICPC+1
      <JUMP!ZCOND!<HEH1-INIT&3000*4>!<HEH1-INIT&777/2>>
TH6:
TMTF1: STATE TMTM0
      MICPC=MICPC+1
      <MOVE!WTEBR!IMM!<TMTM0-INIT&777/2>>
XEXIT: SP BR,SELB,SP2 ;STORE NEW TRANSMIT STATE
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      ALWAYS I1
      MICPC=MICPC+1
      <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
TMTM0: SPBR IBUS,NPR,SPO ;NPR BUSY
      MICPC=MICPC+1
      <MOVE!SPBRX!IBUS!NPR!SPO>
      BRO I1
      MICPC=MICPC+1
      <JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
TH9: OUTPUT IBUS<INDAT2!TMTDAT> ;ODD CHAR TO SILO
      MICPC=MICPC+1
      <MOVE!WROUT!IBUS!<INDAT2!TMTDAT>>
      SP IBUS,IIBA1,SPO ;READ LOW BYTE TO BA
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IIBA1!SPO>
      OUTPUT BR,<INCA!IBA1> ;OUTPUT THE INCREMENTED BA
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<INCA!IBA1>>
      SP IBUS,IIBA2,SPO ;GET HIGH BYTE

```

DMCGAL.MAC 18-JAN-78 16:05

TMTH--ROUTINE TO OUTPUT DATA CHARACTERS

```

(1) 032222 001163
1476 032224 023120
(1) 032224 001164
(1) 032224 062105
1478 032226
(1) 032226 001165
(1) 032230 111376
1479 032230
(1) 032230 001166
(1) 032232 063166
1480 032232
(1) 032232 001167
(1) 032234 111172
1481 032234
(1) 032234 001170
(1) 032236 063167
1482 032236
(1) 032236 001171
(1) 032240 101763
1483 032240
(1) 032240 001172
(1) 032242 123600
1487 032242
(1) 032242 001173
(1) 032244 000542
1488 032244
(1) 032244 001174
(1) 032246 063222
1489 032246
(1) 032246 001175
(1) 032250 000556
1490 032250
(1) 032250 001176
(1) 032252 063260
1491 032252
(1) 032252 001177
(1) 032254 061070
1493 032254
(1) 032254 001200
(1) 032254 104506
1500
    
```

```

MICPC=MICPC+1
<MOVE!SPX!IBUS!IBA2!SPO>
OUTPUT BR <AC!IBA2> ;ADD CARRY
MICPC=MICPC+1
<MOVE!WROUT!BR!<AC!IBA2>>
C HOINCH
MICPC=MICPC+1
<JUMP!CCOND!<HOINCH-INIT&3000*4>!<HOINCH-INIT&777/2>>
TH8: SP BR DECA,SP6 ;DECREMENT CHARACTERCOUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP6>
C TH7 ;NO OVERFLOW
MICPC=MICPC+1
<JUMP!CCOND!<TH7-INIT&3000*4>!<TH7-INIT&777/2>>
SP BR DECA,SP7 ;DECREMENT HIGH BYTE OF COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP7>
Z HEH1 ;BYTE COUNT ZERO
MICPC=MICPC+1
<JUMP!ZCOND!<HEH1-INIT&3000*4>!<HEH1-INIT&777/2>>
TH7: SPBR IBUS,NPR,SPO ;READ NPR REGISTER
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!NPR!SPO>
STATE TMTH
MICPC=MICPC+1
<MOVE!WTEBR!IMM!<TMTH-INIT&777/2>>
TH3: SP BR SELB,SP2 ;SAVE TSTATE
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
TH3X: BRWTE IMM,156 ;CLEAR C0 AND C1
MICPC=MICPC+1
<MOVE!WTEBR!IMM!<156>>
SP BR AANDB,SPO ;CLEAR THE BITS
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SPO>
OUT BR <INCA!ONPR>
MICPC=MICPC+1
<MOVE!WROUTX!BR!<INCA!ONPR>>
ALWAYS RD6
MICPC=MICPC+1
<JUMP!ALCOND!<RD6-INIT&3000*4>!<RD6-INIT&777/2>>
:*****END TIME CRITICAL PATH*****
    
```

DMCGAL.MAC 18-JAN-78 16:05

TMTI--SEND UNNUMBERED TYPE FIELD

1502
1503 032256
(1) 001201
(1) 032256 010151
1504 032260
(1) 001202
(1) 032260 043226
1505 032262
(1) 001203
(1) 032262 000605
1506 032264
(1) 001204
(1) 032264 110520
1507
1508
1509 032266
(1) 001205
(1) 032266 010152
1510 032270
(1) 001206
(1) 032270 000610
1511 032272
(1) 001207
(1) 032272 110520
1512
1513
1514 032274
(1) 001210
(1) 032274 000403
1515 032276
(1) 001211
(1) 032276 060346
1516 032300
(1) 001212
(1) 032300 000616
(1) 001213
(1) 032302 063222
1517 032304
(1) 001214
(1) 032304 111223
1518 032306
(1) 001215
(1) 032306 110517
1519
1520
1521 032310
(1) 001216
(1) 032310 000627
(1) 001217
(1) 032312 063222
1522 032314
(1) 001220
(1) 032314 000403
1523 032316
(1) 001221
(1) 032316 060366

```

TMTI: .SBTTL TMTI--SEND UNNUMBERED TYPE FIELD
LDMA IMM T ;ADDRESS OF TYPE FIELD TO MAR
MICPC=MICPC+1
<MOVE!LDMA!IMM!<T&377>>
SP MEMX,SELB,SP6 ;COPY IT TO SP6
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SP6>
STATE TMTJ
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTJ-INIT&777/2>>
ALWAYS TD3
MICPC=MICPC+1
<JUMP!ALCOND!<TD3-INIT&3000*4>!<TD3-INIT&777/2>>
;
TMTJ: .SBTTL TMTJ--SEND SUB-TYPE FIELD
LDMA IMM ST ;ADDRESS OF SUB-TYPE FIELD TO MAR
MICPC=MICPC+1
<MOVE!LDMA!IMM!<ST&377>>
STATE TMTK
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTK-INIT&777/2>>
ALWAYS TD3
MICPC=MICPC+1
<JUMP!ALCOND!<TD3-INIT&3000*4>!<TD3-INIT&777/2>>
.SBTTL TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG)
;
TMTK: BRWRT IMM 3 ;WRITE A 3 TO BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<3>>
NOP BR,SUB,SP6 ;IF TYPE LESS THAN 3
MICPC=MICPC+1
<BR!SUB!SP6>
TSTATE TMTL
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTL-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
C TMTLO
MICPC=MICPC+1
<JUMP!CCOND!<TMTLO-INIT&3000*4>!<TMTLO-INIT&777/2>>
ALWAYS TD2
MICPC=MICPC+1
<JUMP!ALCOND!<TD2-INIT&3000*4>!<TD2-INIT&777/2>>
;
TMTL: .SBTTL TMTL--UNNUMB MSG NUMBER FIELD
TSTATE TMTM
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTM-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
BRWRT IMM 3
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<3>>
CMP BR,SP6 ;IS MESSAGE REP
MICPC=MICPC+1
<SUBTC!BR!SP6>

```

DMCGAL.MAC 18-JAN-78 16:05

TMTL--UNNUMB MSG NUMBER FIELD

1524 032320 001222
 (1) 032320 111625
 1525 032322 001223
 (1) 032322 000400
 1527 032324 001224
 (1) 032324 110424
 1533 032326 001225
 (1) 032326 060612
 1535 032330 001226
 (1) 032330 110424
 1536 032332 001227
 (1) 032332 000631
 1539 032334 001230
 (1) 032334 110530
 1540 032336 001231
 (1) 032336 000402
 1541 032340 001232
 (1) 032340 062231
 1542 032342 001233
 (1) 032342 062230
 1544 032344 001234
 (1) 032344 000404
 1545 032346 001235
 (1) 032346 063710
 1547 032350 001236
 (1) 032350 060530
 1548 032352 001237
 (1) 032352 113644
 1549 032354 001240
 (1) 032354 000776
 1550 032356 001241
 (1) 032356 063270
 1551 032360 001242
 (1) 032360 000400
 1552 032362 001243
 (1)

```

Z          TMTL1          ;YES
MICPC=MICPC+1
<JUMP!ZCOND!<TMTL1-INIT&3000*4>!<TMTL1-INIT&777/2>>
TMTLO:    BRWRT  IMM 0          ;ADDRESS CONTNAT OF ZERO
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<0>>
ALWAYS  TMTAS
MICPC=MICPC+1
<JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>

TMTL1:    BRWRT  BR SELA!SP12      ;WRITE A RESPONSE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP12>>
ALWAYS  TMTAS
MICPC=MICPC+1
<JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>

TMTM:    ;SBTTL TMTM--UNNUMB MSG--STATION ADDRESS
STATE  TNEOM
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TNEOM-INIT&777/2>>
ALWAYS  TF2
MICPC=MICPC+1
<JUMP!ALCOND!<TF2-INIT&3000*4>!<TF2-INIT&777/2>>
TNEOM:    BRWRT  IMM 2          ;END OF MESSAGE TO BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>>
OUTPUT  BR <SELB!OTMTCO>
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELB!OTMTCO>>
OUTPUT  BR <SELB!TMTDAT>          ;OUTPUT A GARBAGE CHARACTER
MICPC=MICPC+1
<MOVE!WROUT!BR! SELB!TMTDAT>
BRWRT  IMM 4          ;SET UP LINE HAS GONE IDLE MASK
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<4>>
SPBR    BR AORB,SP10          ;UPDATE LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPBRX!BR!AORB!SP10>
BRWRT  BR AA!SP10          ;SHIFT STATUS LEFT
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AA!SP10>>
BR7     TNEOM2          ;IF HDX SET---BRANCH TO CLEAR OK TO SEND
MICPC=MICPC+1
<JUMP!BR7CON!<TNEOM2-INIT&3000*4>!<TNEOM2-INIT&777/2>>
BRWRT  IMM 376          ;MASK TO TURN OFF UNNUMB PENDDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<376>>
TNEOM1:  SP          BR AANDB,SP10          ;MASK TO LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SP10>
TEOM2:  STATE  TMTA
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTA-INIT&777/2>>
ALWAYS  XEXIT
MICPC=MICPC+1

```

K09

DMCGAL.MAC 18-JAN-78 16:05

TMTM--UNNUMB MSG--STATION ADDRESS

(1)	032362	110554
1553	032364	
(1)		001244
(1)	032364	000576
1554	032366	
(1)		001245
(1)		110641
1	032366	

```

TNEOM2: <JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
BRWRT IMM,176 ;CLEAR OK TO SEND AND UNNUMB PENPENDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<176>>
ALWAYS TNEOM1
MICPC=MICPC+1
<JUMP!ALCOND!<TNEOM1-INIT&3000*4>!<TNEOM1-INIT&777/2>>

```

DMCGAL.MAC 18-JAN-78 16:05

TIMSRV--TIMEOUT ROUTINE--SENDS REP

1556
1557
1558
1559 032370 001246
(1) 032370 000577
1560
1561
1562
1563
1564 032372 001247
(1) 032372 061271
1565 032374 001250
(1) 032374 010236
1566 032376 001251
(1) 032376 043220
1567 032400 001252
(1) 032400 063460
1568 032402 001253
(1) 032402 076620
1569 032404 001254
(1) 032404 111256
1570 032406 001255
(1) 032406 100457
1571 032410 001256
(1) 032410 043220
1572 032412 001257
(1) 032412 063460
1573 032414 001260
(1) 032414 062620
1574 032416 001261
(1) 032416 111263
1575 032420 001262
(1) 032420 100457
1576 032422 001263
(1) 032422 002650
1577 032424 001264
(1) 032424 063074
1578
1579
1580
1581 032426

```

.SBTTL TIMSRV--TIMEOUT ROUTINE--SENDS REP
:
:ENABLE LSB
TIMSRV: BRWRT IMM,177 ;MASK OFF BR REQ
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<177>>

;RESET TIMER---SLICK MOVE
;SINCE TIMER IS RESET BY WRITING
;A 1 AND THE EXPIRATION LOOKS
;LIKE 1--VOILA
;AND THE BIT ON

OUT BR <AANDB!OBR>
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AANDB!OBR>>
LDMA IMM,236
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<236&377>>
SP MEMX,SELB,SPO
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SPO>
SPBR BR,INCA,SPO
MICPC=MICPC+1
<MOVE!SPBRX!BR!INCA!SPO>
MEMINC BR,SELB
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<SELB>>
C 5$
MICPC=MICPC+1
<JUMP!CCOND!<5$-INIT&3000*4>!<5$-INIT&777/2>>
ALWAYS IDLE1
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE1-INIT&3000*4>!<IDLE1-INIT&777/2>>
5$: SP MEMX,SELB,SPO
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SPO>
SPBR BR,INCA,SPO
MICPC=MICPC+1
<MOVE!SPBRX!BR!INCA!SPO>
MEM BR,SELB
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELB>>
C 10$
MICPC=MICPC+1
<JUMP!CCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
ALWAYS IDLE1
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE1-INIT&3000*4>!<IDLE1-INIT&777/2>>
10$: MEM IMM,250
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<250>>
SP BR,INCA,SP14 ;INCREMENT THE COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!INCA!SP14>

:NOTE: STATE OF "C" BIT
:MUST BE PRESERVED FROM THIS
:POINT UNTIL IT IS TESTED

TIMEO:

```


DMCAL.MAC 18 JAN-78 16:05

TIMSRV--TIMEOUT ROUTINE--SENDS REP

```

1582 032426 001265 BRWRT BR SELA!SP1 ;READ STATUS BYTE
(1) (1) 032426 060601 MICPC=MICPC+1
1583 032430 (1) 001266 <MOVE!WRTEBR!BR!<SELA!SP1>>
(1) 032430 102047 BRO IDLE
1584 032432 (1) 001267 MICPC=MICPC+1 ;IF IN MAINT. MODE DISABLE TIMER
(1) 032432 103447 <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1585 032434 (1) 001270 BRWRT BR SELA!SP10 ;READ LINE STATUS REGISTER
(1) 032434 060610 MICPC=MICPC+1
1586 032436 (1) 001271 BSHFTB ;ROTATE IT RIGHT ONE BIT
(1) 032436 061620 MICPC=MICPC+1
1587 032440 (1) 001272 <MOVE!SHFTBR!SELB!BR> ;START MODE - RESEND A START
(1) 032440 103164 BR4 BS1
1588 032442 (1) 001273 MICPC=MICPC+1 ;TEST STATE OF C BIT FROM INC SP14
1589 032442 (1) 032442 111277 <JUMP!BR4CON!<BS1-INIT&3000*4>!<BS1-INIT&777/2>> ;IF CARRY SET COUNT HAS EXPIRED
1590 032444 (1) 001274 C 20$ ;NUMBERED MESSAGE IN PROGRESS
(1) 032444 116306 MICPC=MICPC+1
1591 032446 (1) 001275 <JUMP!CCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>> ;UNNUMBMSGIN PROGRESS
(1) 032446 117706 BRO TABUPD
1592 032450 (1) 001276 MICPC=MICPC+1 ;ELSE TRY TO SEND AN ACK
(1) 032450 110737 <JUMP!BROCON!<TABUPD-INIT&3000*4>!<TABUPD-INIT&777/2>>
1593 032452 (1) 032452 001277 ALWAYS SNDACK
1594 032452 (1) 032452 000605 MICPC=MICPC+1 ;SET OK TO SEND,UNNUMB PEND,AND
1595 032452 (1) 032452 000605 <MOVE!WRTEBR!IMM!<20$>> ;LINE GONE IDLE
1596 032454 (1) 001300 SP BR AORB,SP10
1597 032454 (1) 032454 063310 MICPC=MICPC+1 ;RESET THE TIMER TICK COUNT
(1) 032454 001301 <MOVE!SPX!BR!AORB!SP10>
1598 032456 (1) 032456 003374 SP IMM,374,SP14 ;GET LAST NUMBER SENT
(1) 032456 001302 BRWRT BR SELA!SP12
1600 032462 (1) 032462 060612 MICPC=MICPC+1 ;COMPARE TO LAST ACKED
(1) 032462 060377 <MOVE!WRTEBR!BR!<SELA!SP12>>
1601 032464 (1) 001303 CMP BR,SP17 ;IF EQ --SEND ACK
(1) 032464 111737 Z SNDACK
1602 032466 (1) 001305 MICPC=MICPC+1 ;CUMULATIVE REPS SENT
(1) 032466 010015 <JUMP!ZCOND!<SNDACK-INIT&3000*4>!<SNDACK-INIT&777/2>>
1603 032470 (1) 032470 010015 LDMA IMM,REPCS ;COPY IT TO SPO
(1) 032470 010015 <MOVE!LDMAR!IMM!<REPCS&377>>
SP MEMX,SELB,SPO

```

TIME1:
20\$:

TIME2:

(1)		001306	MICPC=MICPC+1	
(1)	032470	043220	<MOVE!SPX!MEMX!SELB!SPO>	
1604	032472		MEM BR,INCA!SPO	; INCREMENT IT
(1)		001307	MICPC=MICPC+1	
(1)	032472	062460	<MOVE!WRMEM!BR!<INCA!SPO>>	
1605	032474		LDMA IMM,REPST	; ADDRESS DYNAMIC REP COUNTER
(1)		001310	MICPC=MICPC+1	
(1)	032474	010003	<MOVE!LDMAR!IMM!<REPST&377>>	
1606	032476		BRWRT MEMX,SELB	; COPY IT TO THE BR
(1)		001311	MICPC=MICPC+1	
(1)	032476	040620	<MOVE!WRTEBR!MEMX!<SELB>>	
1607	032500		BSHFTB	
(1)		001312	MICPC=MICPC+1	
(1)	032500	061620	<MOVE!SHFTBR!SELB!BR>	
1608	032502		MEM BR,SELB	
(1)		001313	MICPC=MICPC+1	
(1)	032502	062620	<MOVE!WRMEM!BR!<SELB>>	
1609	032504		BRO RTHRES	
(1)		001314	MICPC=MICPC+1	
(1)	032504	!06371	<JUMP!BROCON!<RTHRES-INIT&3000*4>!<RTHRES-INIT&777/2>>	
1610	032506		LDMA IMM,T	; LOAD ADDRESS OF TYPE FIELD IN UNNUMB SK
(1)		001315	MICPC=MICPC+1	
(1)	032506	010151	<MOVE!LDMAR!IMM!<T&377>>	
1611	032510		MEMINC IMM,3	; LOAD REP TYPE
(1)		001316	MICPC=MICPC+1	
(1)	032510	016403	<MOVE!WRMEM!INCMAR!IMM!<3>>	
1612	032512		ALWAYS SA1	
(1)		001317	MICPC=MICPC+1	
(1)	032512	110741	<JUMP!ALCOND!<SA1-INIT&3000*4>!<SA1-INIT&777/2>>	
1613			.DSABLE LSB	
1614			;	

DMCGAL.MAC 18-JAN-78 16:05

TIMSRV--TIMEOUT ROUTINE--SENDS REP

```

1616 032514 001320
(1) 032514 120620
1617 032516 001321
(1) 032516 116051
1618 032520 001322
(1) 032520 000402
1619 032522 001323
(1) 032522 062231
1620 032524 001324
(1) 032524 062230
1621 032526 001325
(1) 032526 060601
1622 032530 001326
(1) 032530 113764
1623 032532 001327
(1) 032532 070216
1624 032534 001330
(1) 032534 040620
1625 032536 001331
(1) 032536 112242
1626 032540 001332
(1) 032540 000775
1627 032542 001333
(1) 032542 063670
1628 032544 001334
(1) 032544 112242
1629 032546 001335
(1) 032546 000400
(1) 032550 001336
(1) 032550 063222
1630
1631 032552 001337
(1) 032552 010151
1632 032554 001340
(1) 032554 016401
1633 032556 001341
(1) 032556 000405
1634 032560 001342
(1)

```

```

TEOM:  BRWRT  IBUS,UBBR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<UBBR>>
      BR0      NXMERR ;NON-EXISTANT MEMOR
      MICPC=MICPC+1
      <JUMP!BROCON!<NXMERR-INIT&3000*4>!<NXMERR-INIT&777/2>>
      BRWRT  IMM,2 ;EOM TO BP
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<2>>
      OUTPUT BR,<SELB!OTMTCO> ;WRITE TMTR CONTROL
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!OTMTCO>>
      OUTPUT BR,<SELB!TMTDAT> ;WRITE GARBAGE DATA
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!TMTDAT>>
      BRWRT  BR,SELA!SP1 ;CHECK FOR BOOT MODE
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
      BR7      BTEOM ;---IF SET IS MAINT MSG
      MICPC=MICPC+1
      <JUMP!BR7CON!<BTEOM-INIT&3000*4>!<BTEOM-INIT&777/2>>
TEOM1: LDMA  BR,SELA!SP16 ;ADDRESS LAST TMT LINK
      MICPC=MICPC+1
      <MOVE!LDMAR!BR!<SELA!SP16>>
      BRWRT  MEMX,SELB
      MICPC=MICPC+1
      <MOVE!WRTEBR!MEMX!<SELB>>
      BR0      TEOM2
      MICPC=MICPC+1
      <JUMP!BROCON!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>
TEOM3: BRWRT  IMM,375 ;TURN OFF MESSAGE PENDING
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<375>>
      SPBR  BR,AANDB,SP10 ;
      MICPC=MICPC+1
      <MOVE!SPBRX!BR!AANDB!SP10>
      BR0      TEOM2 ;IF UNNUMB PENDING--GO AWAY
      MICPC=MICPC+1
      <JUMP!BROCON!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>
      TSTATE TMTA
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTA-INIT&777/2>>
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      .SBTTL SNDACK--ROUTINE TO SEND AN ACK
SNDACK: LDMA  IMM,T
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<T&377>>
      MEMINC IMM,1
      MICPC=MICPC+1
      <MOVE!WRMEM!INCMAR!IMM!<1>>
SA1:  BRWRT  IMM,5
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<5>>
SA2:  MEMINC IMM,300
      MICPC=MICPC+1

```

DMCGAL.MAC 18-JAN-78 16:05

SNDACK--ROUTINE TO SEND AN ACK

```

(1) 032560 016700
1635 032562 001343
(1) 032562 063310
1636 032564 001344
(1) 032564 100447
1637
1638 032566 001345
(1) 032566 120600
1639 032570 001346
(1) 032570 102047
1640 032572 001347
(1) 032572 070604
1641 032574 001350
(1) 032574 103574
1642 032576 001351
(1) 032576 036400
1643 032600 001352
(1) 032600 036420
1644 032602 001353
(1) 032602 000402
1645 032604 001354
(1) 032604 063004
1646 032606 001355
(1) 032606 023100
1647 032610 001356
(1) 032610 062004
1648 032612 001357
(1) 032612 023120
1649 032614 001360
(1) 032614 062105
1650 032616 001361
(1) 032616 123200
1651 032620 001362
(1) 032620 115166
1652 032622 001363
(1) 032622 110575

```

```

SA3: <MOVE!WRMEM!INCMAR!IMM!<300>>
      SP BR AORB,SP10
      MICPC=MICPC+1
      <MOVE!SPX!BR!AORB!SP10>
      ALWAYS IDLE
      MICPC=MICPC+1
      <JUMP!ALCONC!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

FUDGE: BRWRT IBUS,NPR ;READ NPR CONTROL
        MICPC=MICPC+1
        <MOVE!WRTEBR!IBUS!<NPR>>
        BRD IDLE ;IF NPR GOING---LEAVE
        MICPC=MICPC+1
        <JUMP!BRCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        BRWRT BR!LDMAR,SELA!SP4 ;LOAD THE MAR
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>
        BR7 BS2 ;IF SET - READ BACK ALL 200
        MICPC=MICPC+1
        <JUMP!BR7CON!<BS2-INIT&3000*4>!<BS2-INIT&777/2>>
        MEMINC IBUS,INDAT1 ;OTHERWISE RESTORE TWO BYTES
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IBUS!<INDAT1>>
        MEMINC IBUS,INDAT2 ;..
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IBUS!<INDAT2>>
        BRWRT IMM,2 ;UPDATE---UNIBUS ADDRESS
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<2>>
        SP BR,ADD,SP4 ;UPDATE NPR COUNTER
        MICPC=MICPC+1
        <MOVE!SPX!BR!ADD!SP4>
        SP IBUS,IIBA1,SPO ;UPDATE ADDRESS LOW
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!IIBA1!SPO>
        OUTPUT BR,ADD!IBA1
        MICPC=MICPC+1
        <MOVE!WROUT!BR!<ADD!IBA1>>
        SP IBUS,IIBA2,SPO ;READ HIGH ADDRESS
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!IIBA2!SPO>
        OUTPUT BR,AC!IBA2 ;UPDATE HIGH
        MICPC=MICPC+1
        <MOVE!WROUT!BR!<AC!IBA2>>
        SP IBUS,NPR,SPO ;READ NPR REGISTER
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!NPR!SPO>
        C RESEXT ;IF CARRY---UPDATE MXT
        MICPC=MICPC+1
        <JUMP!CCOND!<RESEXT-INIT&3000*4>!<RESEXT-INIT&777/2>>
        ALWAYS TH3X ;GO DO ANOTHER NPR
        MICPC=MICPC+1
        <JUMP!ALCOND!<TH3X-INIT&3000*4>!<TH3X-INIT&777/2>>

```

DMCGAL.MAC 18-JAN-78 16:05

SNDACK--ROUTINE TO SEND AN ACK

1654 032624 001364
(1)
(1) 032624 000774
1655 032626 001365
(1)
(1) 032626 063270
1656 032630 001366
(1)
(1) 032630 063233
1657
1658 032632 001367
(1)
(1) 032632 010067
1659 032634 001370
(1)
(1) 032634 043220
1661 032636 001371
(1)
(1) 032636 000404
1662 032640 001372
(1)
(1) 032640 063310
1664 032642 001373
(1)
(1) 032642 000400
(1) 001374
(1) 032644 063222
1665 032646 001375
(1)
(1) 032646 114531
1666
1667 032650
1675 032650 001376
(1)
(1) 032650 123200
1676 032652 001377
(1)
(1) 032652 000404
1677 032654 001400
(1)
(1) 032654 061010
1678 032656 001401
(1)
(1) 032656 110566
1679
1680 032660 001402
(1)
(1) 032660 010016
1681 032662 001403
(1)
(1) 032662 043220
1682 032664 001404
(1)
(1) 032664 062460
1683 032666 001405
(1)

```

BTEOM: BRWRITE IMM,374 ;MASK FOR CLEAR MSG PENDING
        MICPC=MICPC+1
        <MOVE!WRITEBR!IMM!<374>>
        SP BR, AANDB, SP10 ;TURN THEM OFF IN LINE STATUS WORD
        MICPC=MICPC+1
        <MOVE!SPX!BR!AANDB!SP10>
        SP BR, SELB, SP13 ;STORE UNRECOGNIZABLE VALUE INTO SP13
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP13>
        ;SO "RH3" WILL EXIT BACK TO IDLE LOOP
        ;ADDRESS START OF TMT CHAIN

        LDMA IMM,STC
        MICPC=MICPC+1
        <MOVE!LDMAR!IMM!<STC&377>>
        SP MEMX, SELB, SPO ;COPY LINK ADDRESS
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!SELB!SPO>
        BRWRITE IMM,4
        MICPC=MICPC+1
        <MOVE!WRITEBR!IMM!<4>>
        SP BR, AORB, SP10
        MICPC=MICPC+1
        <MOVE!SPX!BR!AORB!SP10>
        TSTATE TMTA ;CHANGE XMIT STATE TO LINE IS IDLE
        MICPC=MICPC+1
        <MOVE!WRITEBR!IMM!<TMTA-INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP2>
        ALWAYS TDON2 ;POST A DONE
        MICPC=MICPC+1
        <JUMP!ALCOND!<TDON2-INIT&3000*4>!<TDON2-INIT&777/2>>
        ;INCREMENT MXT BITS

HOINCH: SP IBUS, NPR, SPO ;READ NPR REG IWTH CURRENT MXT BITS
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!NPR!SPO>
        BRWRITE IMM,4 ;WRITE BIT TO ADD
        MICPC=MICPC+1
        <MOVE!WRITEBR!IMM!<4>>
        OUT BR, <ADD!ONPR> ;TURN ON PROPER MXT BITS
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<ADD!ONPR>
        ALWAYS TH8
        MICPC=MICPC+1
        <JUMP!ALCOND!<TH8-INIT&3000*4>!<TH8-INIT&777/2>>
        .SBTTL REP HANDLER

REP: LDMA IMM,REPCR ;LOAD MAR ADDRESS WITH POINTER TO REPS RECD
        MICPC=MICPC+1
        <MOVE!LDMAR!IMM!<REPCR&377>>
        SP MEMX, SELB, SPO ;READ NUMBER OF REPS RECD
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!SELB!SPO>
        MEM DP, <INCA!SPO> ;INCREMNT REPS RECD
        MICPC=MICPC+1
        <MOVE!WRMEM!DP!<INCA!SPO>>
        LDMA IMM,T ;LOAD ADDRESS OF TYPE FIELD
        MICPC=MICPC+1
    
```

DMCGAL.MAC 18-JAN-78 16:05

REP HANDLER

(1)	032666	010151	<MOVE!LDMAR!IMM!<T&377>>	
1684	032670		MEMINC IMM,2	: LOAD NAK TYPE
(1)		001406	MICPC=MICPC+1	
(1)	032670	016402	<MOVE!WRMEM!INCMAR!IMM!<2>>	
1685	032672		MEMINC IMM,303	:LOAD REP RESPONSE SUB-TYPE
(1)		001407	MICPC=MICPC+1	
(1)	032672	016703	<MOVE!WRMEM!INCMAR!IMM!<303>>	
1686	032674		ALWAYS SNAK	:SEND AN UNNUMB MSG
(1)		001410	MICPC=MICPC+1	
(1)	032674	114700	<JUMP!ALCOND!<SNAK-INIT&3000*4>!<SNAK-INIT&777/2>>	
1687			.	
1688	032676		BR4 SS	:IF START MODE ACCEPT STACK
(1)		001411	MICPC=MICPC+1	
(1)	032676	117013	<JUMP!BR4CON!<SS-INIT&3000*4>!<SS-INIT&777/2>>	
1689	032700		ALWAYS IDLE	:IGNORE STACK
(1)		001412	MICPC=MICPC+1	
(1)	032700	100447	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
1690	032702		BRWRT IMM,327	:MASK TO CLEAR START
(1)		001413	MICPC=MICPC+1	
(1)	032702	000727	<MOVE!WRTEBR!IMM!<327>>	
1691	032704		SP BR, AANDB, SP10	:CLEAR START MODE
(1)		001414	MICPC=MICPC+1	
(1)	032704	063270	<MOVE!SPX!BR!AANDB!SP10>	
1692	032706		ALWAYS SNDACK	:RESET TIMER AND IDLE
(1)		001415	MICPC=MICPC+1	
(1)	032706	110737	<JUMP!ALCOND!<SNDACK-INIT&3000*4>!<SNDACK-INIT&777/2>>	

DMCGAL.MAC 18-JAN-78 16:05

REP HANDLER

1694 032710
1702 032710
(1) 001416
(1) 032710 123220
1703 032712
(1) 001417
(1) 032712 000515
1704 032714
(1) 001420
(1) 032714 063260
1705 032716
(1) 001421
(1) 032716 000404
1706 032720
(1) 001422
(1) 032720 061011
1707 032722
(1) 001423
(1) 032722 104664
1709 032724
(1) 001424
(1) 032724 002212
1710 032726
(1) 001425
(1) 032726 104601
1712 032730
(1) 001426
(1) 032730 010011
1713 032732
(1) 001427
(1) 032732 043220
1714 032734
(1) 001430
(1) 032734 042460
1715 032736
(1) 001431
(1) 032736 060617
1716 032740
(1) 001432
(1) 032740 063232
1717 032742
(1) 001433
(1) 032742 000406
1718
1719 032744
(1) 001434
(1) 032744 010067
1720 032746
(1) 001435
(1) 032746 053236
1721 032750
(1) 001436
(1) 032750 063310
1722 032752
(1) 001437
(1) 032752 003374

```

ICBA23:
SP      IBUS,UBBR,SPO
MICPC=MICPC+1
<MOVE!SPX!IBUS!UBBR!SPO
BRWRT  IMM,115
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<115>>
SP      BR,AANDB,SPO
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SPO>
BRWRT  IMM,4
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<4>>
OUT    BR,<ADD!OBR>
MICPC=MICPC+1
<MOVE!WROUTX!BR!<ADD!OBR>>
ALWAYS RK3
MICPC=MICPC+1
<JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777,2>>
FLUSH1: OUTPUT IMM,<200!ORCVCO>          ;FLUSH THE RECVR
MICPC=MICPC+1
<MOVE!WROUT!IMM!<200!ORCVCO>>
ALWAYS CG1
MICPC=MICPC+1
<JUMP!ALCOND!<CG1-INIT&3000*4>!<CG1-INIT&777,2>>
NAK:    LDMA  IMM,NDATR          ;CUMMULATIVE NAK COUNTER
MICPC=MICPC+1
<MOVE!LDMA!IMM!<NDATR&377>>
SP      MEMX,SELB,SPO          ;READ IT
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SPO>
MEM     MEMX,INCA!SPO          ;INCREMENT THE COUNTER
MICPC=MICPC+1
<MOVE!WRMEM!MEMX!<INCA!SPO>>
BRWRT  BR,SELA!SP17          ;GETLASTMESSAGE ACKED
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP17>>
SP      BR,SELB,SP12          ;COPY TO CURRENT NUMBER
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP12>
BRWRT  IMM,6                  ;WRITE NUMBERED MSG PENDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<6>>
;AND LINE HAS GONE IDLE
;ADDRESS START OF XMIT CHAIN
NAK1:   LDMA  IMM,STC
MICPC=MICPC+1
<MOVE!LDMA!IMM!<STC&377>>
SP      MEMX!LDMA,SELB,SP16    ;COPY START OF CHAIN TO LAST XMIT
MICPC=MICPC+1
<MOVE!SPX!MEMX!LDMA!SELB!SP16>
SP      BR,AORB,SP10          ;SET IT IN LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP10>
SP      IMM,374,SP14          ;RESET TIMER COUNT
MIC C=MICPC+1
<MOVE!SPX!IMM!374!SP14
    
```

DMCGAL.MAC 18-JAN-78 16:05

REP HANDLER

1723	032754	001440	BR7 TEOM1
(1)		113727	MICPC=MICPC+1
(1)	032754		<JUMP!BR7CON!<TEOM1-INIT&3000*4>!<TEOM1-INIT&777/2>>
1724	032756	001441	BRWRTE MEMX,SELB
(1)		040620	MICPC=MICPC+1
(1)	032756		<MOVE!WRTEBR!MEMX!<SELB>>
1725	032760	001442	BRO I1
(1)		102051	MICPC=MICPC+1
(1)	032760		<JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
1726	032762	001443	ALWAYS TEOM3
(1)		110732	MICPC=MICPC+1
(1)	032762		<JUMP!ALCOND!<TEOM3-INIT&3000*4>!<TEOM3-INIT&777/2>>
1727	032764	001444	INWAT1: BRWRTE IMM,200
(1)		000600	MICPC=MICPC+1
(1)	032764		<MOVE!WRTEBR!IMM!<200>>
1728	032766	001445	SP IBUS,INCON,SPO
(1)		123000	MICPC=MICPC+1
(1)	032766		<MOVE!SPX!IBUS!INCON!SPO>
1729	032770	001446	OUT BR,AORB!OINCON
(1)		061300	MICPC=MICPC+1
(1)	032770		<MOVE!WROUTX!BR!<AORB!OINCON>>
1730	032772		PSTATE IEIWAT
(1)			MEM IMM, <<IEIWAT-INIT&777/2>>
(2)		001447	MICPC=MICPC+1
(2)	032772	002507	<MOVE!WRMEM!IMM!<<IEIWAT-INIT&777/2>>>
1731	032774	001450	ALWAYS IDLE
(1)		100447	MICPC=MICPC+1
(1)	032774		<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1732			;
1734			.SBTTL NXMERR ---NON EXISTANT MEMORY HANDLER
1735	032776	001451	NXMERR: LDMA IMM, <<RTHRS+3>>
(1)		010177	MICPC=MICPC+1
(1)	032776		<MOVE!LDMA!IMM!<<RTHRS+3>&377>>
1736	033000	001452	MEMINC IMM,1
(1)		016401	MICPC=MICPC+1
(1)	033000		<MOVE!WRMEM!INCMAR!IMM!<1>>
1737	033002	001453	MEM IMM,0
(1)		002400	MICPC=MICPC+1
(1)	033002		<MOVE!WRMEM!IMM!<0>>
1738	033004	001454	SP MEMX,SELB,SP10
(1)		043230	MICPC=MICPC+1
(1)	033004		<MOVE!SPX!MEMX!SELB!SP10>
1739	033006	001455	ALWAYS RCEXX
(1)		114511	MICPC=MICPC+1
(1)	033006		<JUMP!ALCOND!<RCEXX-INIT&3000*4>!<RCEXX-INIT&777/2>>

H10

DMCGAL.MAC 18-JAN-78 16:05

NXMERR ---NON EXISTANT MEMORY HANDLER

PAGE: 01240M
SEQ 0124

1742					
1743	033010	001456		:FUGITIVE RECEIVE ROUTINES---DON'T FIT IN PAGE	
(1)		070075		LDMA BR <INCA!SP15>	:LD MAR WITH ADDR OF BA
(1)	033010			MICPC=MICPC+1	
1744	033012			<MOVE!LDMAR!BR!<INCA!SP15>>	
(1)		001457		BRWRT BR SELA!SP1	:READ STATUS BYTE
(1)	033012	060601		MICPC=MICPC+1	
1745	033014			<MOVE!WRTEBR!BR!<SELA!SP1>>	
(1)		001460		BRI RH4	:NO BUFF ASSIGNED IN MAINT MODE
(1)	033014	116472		MICPC=MICPC+1	
1746	033016			<JUMP!BR!CON!<RH4-INIT&3000*4>!<RH4-INIT&777/2>>	
(1)		001461		BRWRT IMM!INCMAR,77	
(1)	033016	014477		MICPC=MICPC+1	
1747	033020			<MOVE!WRTEBR!IMM!INCMAR!<77>>	
(1)		001462		SP BR AANDB,SP5	
(1)	033020	063265		MICPC=MICPC+1	
1748	033022			<MOVE!SPX!BR!INCMAR!SELB!SPD>	
(1)		001463		SP BR!INCMAR,SELB,SPD	:SAVE BYTE COUNT MASK
(1)	033022	077220		MICPC=MICPC+1	
1749	033024			<MOVE!SPX!BR!INCMAR!SELB!SPD>	
(1)		001464		BRWRT MEMX!INCMAR,AANDB!SPD	:GET HIGH BYTE COUNT BITS
(1)	033024	054660		MICPC=MICPC+1	
1750	033026			<MOVE!WRTEBR!MEMX!INCMAR!<AANDB!SPD>>	
(1)		001465		CMP BR,SP5	:COMPARE HIGH ORDER BITS OF COUNT
(1)	033026	060365		MICPC=MICPC+1	
1751	033030			<SUBTC!BR!SP5>	
(1)		001466		Z RCLW	:IF EQUAL COMPARE LOW ORDER BITS OF COUNT
(1)	033030	115476		MICPC=MICPC+1	
1752	033032			<JUMP!ZCOND!<RCLW-INIT&3000*4>!<RCLW-INIT&777/2>>	
(1)		001467		C RCFATL	:IF CARRY--TOO BIG ERROR
(1)	033032	115100		MICPC=MICPC+1	
1753	033034			<JUMP!CCOND!<RCFATL-INIT&3000*4>!<RCFATL-INIT&777/2>>	
(1)		001470		BRWRT IBUS,I0BA1	:READ LOW BYTE OF IN BA
(1)	033034	020540		MICPC=MICPC+1	
1754	033036			<MOVE!WRTEBR!IBUS!<I0BA1>>	
(1)		001471		BRD RCVODD	:IF SET IS ODD TRANSFER
(1)	033036	116074		MICPC=MICPC+1	
1755	033040			<JUMP!BRCOND!<RCVODD-INIT&3000*4>!<RCVODD-INIT&777/2>>	
(1)		001472		STATE RCVKED	
(1)	033040	000676		MICPC=MICPC+1	
1756	033042			<MOVE!WRTEBR!IMM!<RCVKED-INIT&777/2>>	
(1)		001473		ALWAYS REXIT	
(1)	033042	104425		MICPC=MICPC+1	
1757	033044			<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>	
(1)		001474		STATE RCVK01	
(1)	033044	000624		MICPC=MICPC+1	
1758	033046			<MOVE!WRTEBR!IMM!<RCVK01-INIT&777/2>>	
(1)		001475		ALWAYS REXIT	
(1)	033046	104425		MICPC=MICPC+1	
1759				<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>	
1760	033050			RCVODD: STATE RCVK01	
(1)		001476		MICPC=MICPC+1	
(1)	033050	040364		<SUBTC!MEMX!SP4>	:COMPARE LOW ORDER BITS OF COUNT
1761	033052			ALWAYS RH6	
(1)		001477		MICPC=MICPC+1	
(1)	033052	114467		<JUMP!ALCOND!<RH6-INIT&3000*4>!<RH6-INIT&777/2>>	

DMCGAL.MAC 18-JAN-78 16:05

NXMERR ---NON EXISTANT MEMORY HANDLER

```

1762 033054 001500
(1) 033054 010151
1763 033056 001501
(1) 033056 016402
1764 033060 001502
(1) 033060 002711
1765 033062 001503
(1) 033062 010175
1766 033064 001504
(1) 033064 036540
1767 033066 001505
(1) 033066 036560
1768 033070 001506
(1) 033070 000420
1769 033072 001507
(1) 033072 016400
1770 033074 001510
(1) 033074 062620
1771 033076 001511
(1) 033076 002212
1772 033100 001512
(1) 033100 000404
(1) 033102 001513
(1) 033102 063222
1773 033104 001514
(1) 033104 003001
1774 033106 001515
(1) 033106 114662
1775
1776
1777 033110 001516
(1) 033110 060610
1778 033112 001517
(1) 033112 001620
1779 033114 001520
(1) 033114 117123
1780
1781 033116 001521
(1) 033116 000600

```

```

RCFATL: LDMA IMM,T
MICPC=MICPC+1
<MOVE!LDMAR!IMM!(<T&377>>
MEMINC IMM,2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!(<2>>
MEM IMM,311
MICPC=MICPC+1
<MOVE!WRMEM!IMM!(<311>>
LDMA IMM,<<RTHRS+1>> ;ADDRESS ERROR LINK
MICPC=MICPC+1
<MOVE!LDMAR!IMM!(<<RTHRS+1>&377>>
MEMINC IBUS IOBA1
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!(<IOBA1>>
MEMINC IBUS IOBA2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!(<IOBA2>>
BRWRT IMM,20
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!(<20>>
RCEXY: MEMINC IMM,0
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!(<0>>
MEM BR SELB
MICPC=MICPC+1
<MOVE!WRMEM!BR!(<SELB>>
RCEXX: OUTPUT IMM,<200!ORCVCO> ;FLUSH INPUT SILO
MICPC=MICPC+1
<MOVE!WROUT!IMM!(<200!ORCVCO>>
TSTATE TMTA6
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!(<TMTA6-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
SP IMM,1,SP1 ;SET INIT MODE IN PORT STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!IMM!1!SP1>
ALWAYS NTRS1
MICPC=MICPC+1
<JUMP!ALCOND!(<NTRS1-INIT&3000*4>!(<NTRS1-INIT&777/2>>
:
START: .SBTTL START HANDLER
BRWRT DP,<SELA!SP10> ;READ LINE STATUS WORD
MICPC=MICPC+1
<MOVE!WRTEBR!DP!(<SELA!SP10>>
BRSHFT ;GET START MODE BIT IN POSITION
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR4 10$ ;IF IN START MODE SET STACK
MICPC=MICPC+1
<JUMP!BR4CON!(<10$-INIT&3000*4>!(<10$-INIT&777/2>>
;ELSE SET UP START ERROR
BRWRT IMM,200
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!(<200>>

```

DMCGAL.MAC 18-JAN-78 16:05

START HANDLER

1782	033120	001522	ALWAYS RCVM2
(1)			MICPC=MICPC+1
(1)	033120	104741	<JUMP!ALCOND!<RCVM2-INIT&3000*4>!<RCVM2-INIT&777/2>>
1783	033122		LDMA IMM,T ;SET UP ADDR OF TYPE FIELD
(1)		001523	MICPC=MICPC+1
(1)	033122	010151	<MOVE!LDMAR!IMM!<T&377>>
1784	033124		MEMINC IMM,7 ;WRITE STACK TYPE
(1)		001524	MICPC=MICPC+1
(1)	033124	016407	<MOVE!WRMEM!INCMAR!IMM!<7>>
1785	033126		BRWTE IMM,11 ;SEND THE UNNUMB MESS
(1)		001525	MICPC=MICPC+1
(1)	033126	000411	<MOVE!WRTEBR!IMM!<11>>
1786	033130		ALWAYS SA2
(1)		001526	MICPC=MICPC+1
(1)	033130	110742	<JUMP!ALCOND!<SA2-INIT&3000*4>!<SA2-INIT&777/2>>

DMCGAL.MAC 18-JAN-78 16:05

START HANDLER

```

1788 033132 001527 TDON3: BRWRT MEMX,SUB!SP17 ;COMPARE RESPONSE TO MSG NO
(1) (1) 033132 040757 MICPC=MICPC+1
1789 033134 001530 <MOVE!WRTEBR!MEMX!<SUB!SP17>>
(1) (1) 033134 107572 BR7 RH3 ;IF NEGATIVE EXIT
MICPC=MICPC+1
1790 033136 001531 <JUMP!BR7CON!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>
(1) (1) 033136 070200 TDON2: LDMA BR,SELA!SPO ;ADDRESS THE TRANSMITLINK
MICPC=MICPC+1
1791 033140 001532 <MOVE!LDMAR!BR!<SELA!SPO>>
(1) (1) 033140 002400 MEM IMM,0 ;TURN OF ASSIGNEDAND TMTED BITS IN FLAG
MICPC=MICPC+1
1792 033142 001533 LDMA IMM,STC
(1) (1) 033142 010067 MICPC=MICPC+1
1793 033144 001534 <MOVE!LDMAR!IMM!<STC&377>>
(1) (1) 033144 002471 MEM IMM,TML1 ;ASSUME WRAPAROUND
MICPC=MICPC+1
1794 033146 001535 <MOVE!WRMEM!IMM!<TML1>>
(1) (1) 033146 000543 BRWRT IMM,TMLB ;WRAPAROUND?
MICPC=MICPC+1
1795 033150 001536 <MOVE!WRTEBR!IMM!<TMLB>>
(1) (1) 033150 060360 CMP BR,SPO
MICPC=MICPC+1
1796 033152 001537 <SUBTC!BR!SPO>
(1) (1) 033152 115542 Z TDON4 ;YES
MICPC=MICPC+1
1797 033154 001540 <JUMP!ZCOND!<TDON4-INIT&3000*4>!<TDON4-INIT&777/2>>
(1) (1) 033154 000406 BRWRT IMM,6 ;OFFSET FOR NEXT TMT LINK
MICPC=MICPC+1
1798 033156 001541 <MOVE!WRTEBR!IMM!<6>>
(1) (1) 033156 062400 MEM BR,ADD!SPO ;UPDATE THE POINTER
MICPC=MICPC+1
1799 033160 001542 <MOVE!WRMEM!BR!<ADD!SPO>>
(1) (1) 033160 010241 TDON4: LDMA IMM,NXTSP ;ADDRESS DONE LINK
MICPC=MICPC+1
1800 033162 001543 <MOVE!LDMAR!IMM!<NXTSP&377>>
(1) (1) 033162 053223 LDMA MEMX,SELB!SPX!SP3 ;ADDRESS THE LINK,COPYING
MICPC=MICPC+1
1801 033164 001544 <MOVE!LDMAR!MEMX!<SELB!SPX!SP3>>
(1) (1) 033164 016600 MEMINC IMM,200 ;ITS ADDRESS TO SPO
MICPC=MICPC+1 ;WRITE THE INTERRUPT TYPE
1803 033166 001545 <MOVE!WRMEM!INCMAR!IMM!<200>>
(1) (1) 033166 062460 MEM BR,INCA!SPO ;COPY ACTUAL LINK ADDRESS
MICPC=MICPC+1
1804 033170 001546 <MOVE!WRMEM!BR!<INCA!SPO>>
(1) (1) 033170 010241 LDMA IMM,NXTSP ;ADDRESS PTR INT STACK
MICPC=MICPC+1
1805 033172 001547 <MOVE!LDMAR!IMM!<NXTSP&377>>
(1) (1) 033172 002642 MEM IMM,INTSTK ;ASSUME WRAP AROUND
MICPC=MICPC+1
1806 033174 001550 <MOVE!WRMEM!IMM!<INTSTK>>
(1) (1) 033174 000776 BRWRT IMM,<<MMEND-2>> ;ADDRESS ENDOFINT STACK
MICPC=MICPC+1
1807 033176 000776 <MOVE!WRTEBR!IMM!<<MMEND-2>>>
(1) (1) 033176 000776 CMP BR,SP3 ;WRAPAROUND?

```

DMCGAL.MAC 18-JAN-78 16:05

START HANDLER

```

(1)          001551
(1)          060363
1808 033200 001552
(1)          115555
(1)          001553
(1)          000402
1810 033202 001554
(1)          062403
(1)          001555
(1)          000420
1812 033206 001556
(1)          063301
1813 033210 001557
1814 033212 010153
(1)          001560
(1)          043237
1816 033214 001561
(1)          010067
(1)          001562
(1)          053620
1818 033220 001563
(1)          054620
(1)          001564
(1)          116527
1820 033222 001565
(1)          104573
1821 033224 001566
1822 033226 000404
1823 033230 001567
(1)          063000
(1)          001570
1825 033232 110575
(1)          063000
(1)          110575

MICPC=MICPC+1
<SUBTC!BR!SP3>
Z          TDON40          ;YES---BRANCH
MICPC=MICPC+1
<JUMP!ZCOND!<TDON40-INIT&3000*4>!<TDON40-INIT&777/2>>
BRWRT     IMM,2          ;OFFSET TO NEXT PAIR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>>
MEM      BR,ADD!SP3      ;UPDATE POINTER
MICPC=MICPC+1
<MOVE!WRMEM!BR!<ADD!SP3>>
TDON40: BRWRT     IMM,20  ;WRITE INTERRUPT PENDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<20>>
SP      BR,AORB,SP1      ;IN PORT STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP1>

TDON1:  LDMA     IMM,ISP17  ;GET LAST ACKED
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<ISP17&377>>
SP      MEMX,SELB,SP17    ;STORE IT IN SP17
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SP17>
LDMA     IMM,STC          ;GET START OF TMT CHAIN
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<STC&377>>
LCMA     MEMX,SELB!SPBRX!SPO ;ADDRESS THE LINK
MICPC=MICPC+1
<MOVE!LDMAR!MEMX!<SELB!SPBRX!SPO>>
BRWRT     MEMX!INCMAR,SELB ;GET THE FLAGS
MICPC=MICPC+1
<MOVE!WRTEBR!MEMX!INCMAR!<SELB>>
BR1      TDON3          ;IF BUFFER ASSIGNED PROCEED
MICPC=MICPC+1
<JUMP!BR1CON!<TDON3-INIT&3000*4>!<TDON3-INIT&777/2>>
ALWAYS   RH3          ;ELSE---EXIT
MICPC=MICPC+1
<JUMP!ALCOND!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>

RESEXT: BRWRT     IMM,4    ;ADD TO MXT BITS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<4>>
SP      BR,ADD,SPO
MICPC=MICPC+1
<MOVE!SPX!BR!ADD!SPO>
ALWAYS   TH3X
MICPC=MICPC+1
<JUMP!ALCOND!<TH3X-INIT&3000*4>!<TH3X-INIT&777/2>>

```

DMCGAL.MAC 18-JAN-78 16:05

START HANDLER

```

1827
1828
1829
1830
1831 033236 001571
(1) 033236 023333
1832 033240 001572
(1) 033240 115576
1833 033242 001573
(1) 033242 000406
1834 033244 001574
(1) 033244 060360
1835 033246 001575
(1) 033246 115603
1836 033250 001576
(1) 033250 060601
1837 033252 001577
(1) 033252 106746
1838 033254 001600
(1) 033254 060521
1839 033256 001601
(1) 033256 107340
1840 033260 001602
(1) 033260 104664
1841 033262 001603
(1) 033262 063164
1842 033264 001604
(1) 033264 000743
1843 033266 001605
(1) 033266 104425
1844
1845
1846
1847 033270 001606
(1) 033270 010001
1848 033272 001607
(1) 033272 002401
1849 033274 001610
(1) 033274 010167
1850 033276

```

```

: INPUTS:
: SPO = RECEIVE CHARACTER
PASWRD: SP IBUS LNOSW,SP13 ;READ PASSWD SWITCH
MICPC=MICPC+1
<MOVE!SPX!IBUS!LNOSW!SP13>
Z 10$ ;IF ALL ONES NO RLD ENABLED
MICPC=MICPC+1
<JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
BRWRT IMM,6 ;CHECK FOR ENTER MOP MODE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<6>>
CMP BR,SPO
MICPC=MICPC+1
<SUBTC!BR!SPO>
Z 20$ ;IF EQUAL ENTER MOP
MICPC=MICPC+1
<JUMP!ZCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
10$: BRWRT BR,SELA!SP1 ;READ STATUS BYTE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
BR1 RHX ;MESSAGE WITH NO BUFFER ASSIGNED
MICPC=MICPC+1
<JUMP!BR1CON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
BRWRT BR,AA!SP1
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AA!SP1>>
BR4 RCVMO ;DLE RECEIVED IN NORMAL MODE
MICPC=MICPC+1
<JUMP!BR4CON!<RCVMO-INIT&3000*4>!<RCVMO-INIT&777/2>>
ALWAYS RK3 ;HANDLE MAINT MODE MESSAGE
MICPC=MICPC+1
<JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>
20$: SP BR,DECA,SP4 ;COUNT FOR NUMB OF COMPARES
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP4>
STATE EM2
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<EM2-INIT&777/2>>
ALWAYS REXIT
MICPC=MICPC+1
<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
;
; .ENABL LSB
:
RCVM1: LDMA IMM,NAKST ;RESET NAKS SENT
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<NAKST&377>>
MEM IMM,1 ;...
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<1>>
LDMA IMM,BC ;ADDRESS ORIGINAL RECV BYTE COUNT
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<BC&377>>
SP MEMX!INCMAR,SELB,SP4 ;MOVE BYTE COUNT TO SP4

```

DMCGAL.MAC

18-JAN-78 16:05

START HANDLER

```

(1)      001611      MICPC=MICPC+1
(1)      033276      057224      <MOVE!SPX!MEMX!INCMAR!SELB!SP4>
1851     033300      MEMX!INCMAR,SELB,SP5      ;AND SP5
(1)      001612      MICPC=MICPC+1
(1)      033300      057225      <MOVE!SPX!MEMX!INCMAR!SELB!SP5>
1852     033302      MEM      BP,DECA!SP11      ;COPY SP11 TO MEMORY
(1)      001613      MICPC=MICPC+1
(1)      033302      062571      <MOVE!WRMEM!BR!<DECA!SP11>>
1853     033304      LDMA      IMM,NXTSP
(1)      001614      MICPC=MICPC+1
(1)      033304      010241      <MOVE!LDMAR!IMM!'NXTSP&377'\,
1854     033306      SP      MEMX!LDMAR,SELB,SP3      ;COPY TO SP3
(1)      001615      MICPC=MICPC+1
(1)      033306      053223      <MOVE!SPX!MEMX!LDMAR!SELB!SP3>
1855     033310      MEMINC      IMM,204      ;RECEIVE DONE IMAGE
(1)      001616      MICPC=MICPC+1
(1)      033310      016604      <MOVE!WRMEM!INCMAR!IMM!<204>>
1856     033312      MEM      BR!LDMAR,SELA!SP15      ;COPY LINK ADDRESS TO NEXT INTER
(1)      001617      MICPC=MICPC+1
(1)      033312      072615      <MOVE!WRMEM!BR!LDMAR!<SELA!SP15>>
1857     033314      MEMINC      IMM,0      ;ZERO THE FLAGS
(1)      001620      MICPC=MICPC+1
(1)      033314      016400      <MOVE!WRMEM!INCMAR!IMM!<0>>
1858     033316      SP      IMM!INCMAR,SPO,300      ;WRITE A 300 TO SPO
(1)      001621      MICPC=MICPC+1
(1)      033316      017300      <MOVE!SPX!IMM!INCMAR!SPO!300>
1859     033320      BRWRT      IMM!INCMAR,2      ;PREPARE TO ADDRESS NEXT
(1)      001622      MICPC=MICPC+1
(1)      033320      014402      <MOVE!WRTEBR!IMM!INCMAR!<2>>
1860      ;INTERRUPT STACK AND INCREMENT
1861      ;THE MAR
1862     033322      MEM      MEMX,AANDB!SPO      ;MASK OFF ORIGINAL HIGH BYTE
(1)      001623      MICPC=MICPC+1
(1)      033322      042660      <MOVE!WRMEM!MEMX!<AANDB!SPO>>
1863      ;OF COUNT SAVING EXTENDED MEM BITS
1864     033324      MEMINC      MEMX,AORB!SP5      ;COPY TO MEMORY LINK
(1)      001624      MICPC=MICPC+1
(1)      033324      056705      <MOVE!WRMEM!INCMAR!MEMX!<AORB!SP5>>
1865     033326      MEMINC      BR,SELA!SP4
(1)      001625      MICPC=MICPC+1
(1)      033326      076604      <MOVE!WRMEM!INCMAR!BR!<SELA!SP4>>
1866     033330      LDMA      IMM,NXTSP      ;ADDRESS NEXT INT STACK
(1)      001626      MICPC=MICPC+1
(1)      033330      010241      <MOVE!LDMAR!IMM!<NXTSP&377>>
1867     033332      MEM      BR,ADD!SP3
(1)      001627      MICPC=MICPC+1
(1)      033332      062403      <MOVE!WRMEM!BR!<ADD!SP3>>
1868     033334      BRWRT      IMM,<<MMEND-2>>      ;ADDRESEND OF INT STACK
(1)      001630      MICPC=MICPC+1
(1)      033334      000776      <MOVE!WRTEBR!IMM!<<MMEND-2>>>
1869     033336      CMP      BR,SP3      ;WRAP AROUND
(1)      001631      MICPC=MICPC+1
(1)      033336      060363      <SUBTC!BR!SP3>
1870     033340      Z      40$      ;IF YES-- BRANCH
(1)      001632      MICPC=MICPC+1
(1)      033340      115646      <JUMP!ZCOND!<40$-INIT&3000*4>!<40$-INIT&777/2>>

```

```

1871 033342
1872 033342
(1) 001633
(1) 033342 000462
1873 033344
(1) 001634
(1) 033344 060375
1874 033346
(1) 001635
(1) 033346 115650
1875 033350
(1) 001636
(1) 033350 000405
1876 033352
(1) 001637
(1) 033352 063015
1877 033354
(1) 001640
(1) 033354 000420
1878 033356
(1) 001641
(1) 033356 063301
1880 033360
(1) 001642
(1) 033360 060610
1881 033362
(1) 001643
(1) 033362 107015

```

20\$:

```

BRWRT IMM,RCL7
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCL7>>
CMP BR,SP15
MICPC=MICPC+1
<SUBTC!BR!SF15>
Z SOS
MICPC=MICPC+1
<JUMP!ZCOND!<SOS-INIT&3000*4>!<SOS-INIT&777/2>>

```

30\$:

```

BRWRT IMM,5
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<5>>
SP BR,ADD,SP15
MICPC=MICPC+1
<MOVE!SPX!BR!ADD!SP15>
BRWRT IMM,20 ;MASK FOR INTERUP* PENDING
MICPC=MICPC+1
SP DP,AORB,SP1 ;UPDATE PORT STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!DP!AORB!SP1>
BRWRT DP,<SELA!SP10> ;READ LINE STATUS WORD
MICPC=MICPC+1
<MOVE!WRTEBR!DP!<SELA!SP10>>
BR4 FLUSH ;IF CLEAR ACTIVE SET---FLUSH
MICPC=MICPC+1
<JUMP!BR4CON!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>

```


DMCGAL.MAC 18-JAN-78 16:05

START HANDLER

1883	033364	001644	RM1:	STATE RCVA
(1)	(1)	000400		MICPC=MICPC+1
1884	033364			<MOVE!WRTEBR!IMM!<RCVA-INIT&777/2>>
(1)	(1)	001645		ALWAYS REXIT
(1)	033366	104425		MICPC=MICPC+1
1889	033370	001646	40\$:	<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
(1)	(1)	002642		MEM IMM INTSTK ;POINT TO START OF INTERRUPT STACK
(1)	033370			MICPC=MICPC+1
1890	033372	001647		<MOVE!WRMEM!IMM!<INTSTK>>
(1)	(1)	114633		ALWAYS 20\$
1891	033374	001650	50\$:	MICPC=MICPC+1
(1)	(1)	000742		<JUMP!ALCOND!<20\$-INIT&3000*4>!<20\$-INIT&777/2>>
(1)	033374			BRWRT IMM <<RCL1-RCL7>&377> ;POINT TO START OF QUEUE
1892	033376	001651		MICPC=MICPC+1
(1)	(1)	114637		<MOVE!WRTEBR!IMM!<<RCL1-RCL7>&377>>
1893	033400	001652		ALWAYS 30\$
(1)	(1)	010152		MICPC=MICPC+1
1895	033402	001653	NTHRES:	LDMA IMM ST
(1)	(1)	043620		MICPC=MICPC+1
1896	033404	001654		<MOVE!LDMAR!IMM!<ST&377>>
(1)	(1)	060400		SPBR MEMX SELB SPO
1897	033406	001655		MICPC=MICPC+1
(1)	(1)	103361		<MOVE!SPBRX!MEMX!SELB!SPO>
1898	033410	001656		BRWRT BR ADD!SPO ;SHIFT LEFT
(1)	(1)	000401		MICPC=MICPC+1
1899	033412	001657		<MOVE!WRTEBR!BR!ADD!SPO>>
(1)	(1)	010177		BR4 OVRUN
1901	033414	001660		MICPC=MICPC+1
(1)	(1)	016400		<JUMP!BR4CON!<OVRUN-INIT&3000*4>!<OVRUN-INIT&777/2>>
1902	033416	001661		BRWRT IMM 1
(1)	(1)	062620		MICPC=MICPC+1
1903	033420	001662		<MOVE!WRTEBR!IMM!<1>>
(1)	(1)	010241	ERRXX:	LDMA IMM <<RTHRS+3>>
1904	033422	001663	NTRSO:	MICPC=MICPC+1
(1)	(1)	053220		<MOVE!LDMAR!IMM!<<RTHRS+3>&377>>
1905	033424	001664		MEMINC IMM 0
(1)	(1)	016601		MICPC=MICPC+1
1906	033426	001665		<MOVE!WRMEM!INCMAR!IMM!<0>>
(1)	(1)	002574		MEM BR SELB
				MICPC=MICPC+1
				<MOVE!WRMEM!BR!<SELB>>
			NTRS1:	LDMA IMM NXTSP
				MICPC=MICPC+1
				<MOVE!LDMAR!IMM!<NXTSP&377>>
				LDMA MEMX SELB!SPX!SPO
				MICPC=MICPC+1
				<MOVE!LDMAR!MEMX!<SELB!SPX!SPO>>
				MEMINC IMM 201
				MICPC=MICPC+1
				<MOVE!WRMEM!INCMAR!IMM!<201>>
				MEM IMM <<RTHRS>>
				MICPC=MICPC+1
				<MOVE!WRMEM!IMM!<<RTHRS>>

DMCGAL.MAC 18-JAN-78 16:05

START HANDLER

1907	033430		LDMA IMM,NXTSP
(1)		001666	MICPC=MICPC+1
(1)	033430	010241	<MOVE!LDMAR!IMM!<NXTSP&377>>
1908	033432		MEM IMM,INTSTK ;ASSUME QUEUE WRAP AROUND
(1)		001667	MICPC=MICPC+1
(1)	033432	002642	<MOVE!WRMEM!IMM!<INTSTK>
1909	033434		BRWRT IMM,<<MMEND-2>>
(1)		001670	MICPC=MICPC+1
(1)	033434	000776	<MOVE!WRTEBR!IMM!<<MMEND-2>>
1910	033436		CMP BR,SPO
(1)		001671	MICPC=MICPC+1
(1)	033436	060360	<SUBTC!BR!SPO>
1911	033440		Z NTRS2 ;IT DID WRAP AROUND
(1)		001672	MICPC=MICPC+1
(1)	033440	115675	<JUMP!ZCOND!<NTRS2-INIT&3000*4>!<NTRS2-INIT&777/2>>
1912	033442		BRWRT IMM,2 ;OFFSET TO NEXT PAIR
(1)		001673	MICPC=MICPC+1
(1)	033442	000402	<MOVE!WRTEBR!IMM!<2>>
1913	033444		MEM BR,ADD!SPO ;UPDATE QUEUE POINTER
(1)		001674	MICPC=MICPC+1
(1)	033444	062400	<MOVE!WRMEM!BR!<ADD!SPO>
1914	033446		NTRS2: BRWRT IMM,20
(1)		001675	MICPC=MICPC+1
(1)	033446	000420	<MOVE!WRTEBR!IMM!<20>>
1915	033450		SPBR BR,AORB,SP1
(1)		001676	MICPC=MICPC+1
(1)	033450	063701	<MOVE!SPBR!BR!AORB!SP1>
1916	033452		BRD TAB1 ;FLAGGED BY ERROR TYPE
(1)		001677	MICPC=MICPC+1
(1)	033452	116311	<JUMP!BRCON!<TAB1-INIT&3000*4>!<TAB1-INIT&777/2>>
1917	033454		SNAK: LDMA IMM,ISP11
(1)		001700	MICPC=MICPC+1
(1)	033454	010171	<MOVE!LDMAR!IMM!<ISP11&377>>
1918	033456		SP MEMX,SELB,SP11
(1)		001701	MICPC=MICPC+1
(1)	033456	043231	<MOVE!SPX!MEMX!SELB!SP11>
1919	033460		SP BR,INCA,SP11 ;INCREMENT MSG EXPECTED
(1)		001702	MICPC=MICPC+1
(1)	033460	063071	<MOVE!SPX!BR!INCA!SP11>
1920	033462		SNAK1: BRWRT IMM,1 ;UNNUMB PENING BIT TO BR
(1)		001703	MICPC=MICPC+1
(1)	033462	000401	<MOVE!WRTEBR!IMM!<1>>
1921	033464		SNAK2: SP BR,AORB,SP10 ;UPDATE LINE STATUS WORD
(1)		001704	MICPC=MICPC+1
(1)	033464	063310	<MOVE!SPX!BR!AORB!SP10>
1922	033466		ALWAYS FLUSH
(1)		001705	MICPC=MICPC+1
(1)	033466	104415	<JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>

DMCGAL.MAC 18-JAN-78 16:05

START HANDLER

```

1924 033470 001706
(1) 033470 023640
1925 033472 001707
(1) 033472 06040C
1926 033474 001710
(1) 033474 103447
1927 033476 001711
(1) 033476 010154
1928 033500 001712
(1) 033500 000442
1929 033502 001713
(1) 033502 076670
1930 033504 001714
(1) 033504 076611
1931 033506 001715
(1) 033506 076615
1932 033510 001716
(1) 033510 076617
1933 033512 001717
(1) 033512 000461
1935 033514 001720
(1) 033514 010210
1936 033516 001721
(2) 033516 002756
1937 033520 001722
(1) 033520 003004
1938 033522 001723
(1) 033522 010017
1941 033524 001724
(1) 033524 104443

```

```

TABUPD: SPBR IBUS,RCVCON,SP0 ;READ RECEIVER CONTROL REG
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!RCVCON!SP0>
BRWRT BR,ADD!SP0 ;SHIFT LEFT
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<ADD!SP0>>
BR7 IDLE ;RECEIVE ACTIVE--IDLE
MICPC=MICPC+1
<JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
TAB1: LDMA IMM,IMG10
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<IMG10&377>>
BRWRT IMM,42
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<42>>
MEMINC BR,AANDB!SP10 ;SAVE BITS 1 AND 5 OF SP10
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<AANDB!SP10>>
MEMINC BR,SELA!SP11 ;SAVE SP11
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<SELA!SP11>>
MEMINC BR,SELA!SP15 ;SAVE SP15
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<SELA!SP15>>
MEMINC BR,SELA!SP17 ;SAVE SP17
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<SELA!SP17>>
STATE RB2 ;MAR NOW POINTS TO BASE
MICPC=MICPC+1 ;DO NOT CHANGE BR UNTIL RBO
<MOVE!WRTEBR!IMM!<RB2-INIT&777/2>>
LDMA IMM,TABST ;POINT TO TABLE UPDATE STATE
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<TABST&377>>
PSTATE TBU1 ;NEW PORT STATE ADDRESS
MEM IMM,<<TBU1-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<TBU1-INIT&777/2>>>
SP IMM,4,SP4 ;INITIALIZE COUNT
MICPC=MICPC+1
<MOVE!SPX!IMM!4!SP4>
;NOTE: FIRST 6 RAM LOCATIONS ARE NOT WRITTEN
;TO CORE TABLE.
LDMA IMM,BASE
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<BASE&377>>
ALWAYS RBO
MICPC=MICPC+1
<JUMP!ALCOND!<RBO-INIT&3000*4>!<RBO-INIT&777/2> \

```

DMCGAL.MAC 18-JAN-78 16:05

START HANDLER

1943 033526 001725
(1) 033526 000402
1944 033530 001726
(1) 033530 063004
1945 033532 001727
(1) 033532 023140
1946 033534 001730
(1) 033534 062006
1947 033536 001731
(1) 033536 023160
1948 033540 001732
(1) 033540 062107
1949 033542 001733
(1) 033542 101271
1950 033544 001734
(1) 033544 060601
1951 033546 001735
(1) 033546 116340
1952
1953 033550 001736
(1) 033550 070604
1954 033552 001737
(1) 033552 117346
1955 033554 001740
(1) 033554 070604
1956 033556 001741
(1) 033556 117746
1957 033560 001742
(1) 033560 056222
1958 033562 001743
(1) 033562 056223
1959 033564 001744
(1) 033564 123200
1960 033566 001745
(1) 033566 104640
1961 033570 001746
(1) 033570 010210
1962 033572

```

EC2:  BRWRT  IMM,2          ;INCREMENT COUNT TEST
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<2>>
      SP BR,ADD,SP4
      MICPC=MICPC+1
      <MOVE!SPX!BR!ADD!SP4>
      SP IBUS,I0BA1,SPO          ;POINT TO NEXT ADDRESS
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!I0BA1!SPO>
      OUTPUT BR,ADD!OBA1
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<ADD!OBA1>>
      SP IBUS,I0BA2,SPO
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!I0BA2!SPO>
      OUTPUT BR,AC!OBA2
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<AC!OBA2>>
      C TABMXT
      MICPC=MICPC+1
      <JUMP!CCOND!<TABMXT-INIT&3000*4>!<TABMXT-INIT&777/2>>
ECX:  BRWRT BR,SELA!SP1          ;READ PORT STATUS
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
      BR0 30$          ;INIT MODE, WRITE OUT 200 BYTES
      MICPC=MICPC+1
      <JUMP!BR0CON!<30$-INIT&3000*4>!<30$-INIT&777/2>>
      ;OTHERWISE ONLY WRITE OUT ERROR COUNTERS
      BRWRT BR!LDMAR,SELA!SP4          ;READ COUNTER
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>
      BR4 20$          ;ALL DONE
      MICPC=MICPC+1
      <JUMP!BR4CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>
30$:  BRWRT BR!LDMAR,SELA!SP4          ;READ CTR
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>
      BR7 20$          ;ALL DONE
      MICPC=MICPC+1
      <JUMP!BR7CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>
      OUTPUT MEMX!INCMAR,SELB!OUTDA1          ;STORE COUNTS OF ERRORS
      MICPC=MICPC+1
      <MOVE!WROUT!MEMX!INCMAR!<SELB!OUTDA1>>
      OUTPUT MEMX!INCMAR,SELB!OUTDA2
      MICPC=MICPC+1
      <MOVE!WROUT!MEMX!INCMAR!<SELB!OUTDA2>>
      SP IBUS,NPR,SPO
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!NPR!SPO>
      ALWAYS RKB
      MICPC=MICPC+1
      <JUMP!ALCOND!<RKB-INIT&3000*4>!<RKB-INIT&777/2>>
20$:  LDMA IMM,TABST
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<TABST&377>>
      PSTATE I3
    
```

DMCGAL.MAC 18-JAN-78 16:05

START HANDLER

1	033572	001747	MEM IMM, << I3-INIT&777 2 >>
2		002455	MICPC=MICPC+1
1964	033572		< MOVE!WRMEM! IMM! << I3-INIT&777 2 >>
(1)	033574	001750	ALWAYS RM1
(1)		114644	MICPC=MICPC+1
1976	033574		< JUMP!ALCOND! < RM1-INIT&3000*4 >! < RM1-INIT&777/2 >>
(1)	033576	001751	CB00T: BRWRT IMM, 200 ; MASK FOR BOOT MODE
(1)		000600	MICPC=MICPC+1
1977	033600		< MOVE!WRTEBR! IMM! < 200 >>
(1)		001752	SP BR, AORB, SP1 ; IN PORT STATUS WORD
(1)	033600	063301	MICPC=MICPC+1
1978	033602		< MOVE!SPX! BR! AORB! SP1 >
(1)		001753	BRWRT IMM, 204 ; MASK FOR OK TO SEND AND LINE IDLE
(1)	033602	000604	MICPC=MICPC+1
1979	033604		< MOVE!WRTEBR! IMM! < 204 >>
(1)		001754	SP BR, SELB, SP10 ; IN LINE STATUS
(1)	033604	063230	MICPC=MICPC+1
1980	033606		< MOVE!SPX! BR! SELB! SP10 >
(1)		001755	TSTATE TMTA
(1)	033606	000400	MICPC=MICPC+1
(1)		001756	< MOVE!WRTEBR! IMM! < TMTA-INIT&777 2 >>
(1)	033610	063222	MICPC=MICPC+1
1981	033612		< MOVE!SPX! BR! SELB! SP2 >
(1)		001757	ALWAYS INS12
(1)	033612	100663	MICPC=MICPC+1
1983	033614		< JUMP!ALCOND! < INS12-INIT&3000*4 >! < INS12-INIT&777/2 >>
(1)		001760	BOOT: BRWRT BR, SELA! SP1
(1)	033614	060601	MICPC=MICPC+1
1984	033616		< MOVE!WRTEBR! BR! < SELA! SP1 >>
(1)		001761	BR7 RA3
(1)	033616	103746	MICPC=MICPC+1
1985	033620		< JUMP!BR7CON! < RA3-INIT&3000*4 >! < RA3-INIT&777/2 >>
(1)		001762	BRWRT IMM, 210
(1)	033620	000610	MICPC=MICPC+1
1986	033622		< MOVE!WRTEBR! IMM! < 210 >>
(1)		001763	SP BR, AORB, SP1
(1)	033622	063301	MICPC=MICPC+1
1987	033624		< MOVE!SPX! BR! AORB! SP1 >
(1)		001764	ALWAYS RA3
(1)	033624	100746	MICPC=MICPC+1
1995		000001	< JUMP!ALCOND! < RA3-INIT&3000*4 >! < RA3-INIT&777/2 >>
			.END
ABS.	033626	000	

ERRORS DETECTED: 0

DDCMP/CRF/DS: CRF+DMCNEW, LOW1, DMCAL
RUN-TIME: 6 11 0 SECONDS
RUN-TIME RATIO: 366/17=21.1
CORE USED: 6K (11 PAGES)

1668

H11

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

PAGE: 01370Z
SEQ 0137

1669
1670 C1E324

HIMAP:

:HIGH SPEED (LOCAL) MICRO-CODE

DMCNEW.MAC

26-OCT-77 13:33

TABLE OF CONTENTS

7	MACRO DEFINITIONS
9	REVISION 00
10	FEBRUARY 25, 1975
11	
12	REVISION 01
13	MARCH 18, 1975
14	NEW CSR BOARD CHANGES
15	
16	HARVEY M. SCHLESINGER
18	COPYRIGHT 1975 DIGITAL EQUIPMENT CORPORATION
70	MICRO INSTRUCTION DEFINITIONS
71	BRANCH INSTRUCTIONS
114	INDEXED BRANCH INSTRUCTIONS
150	MOVE INSTRUCTIONS
258	INPUT/OUTPUT ASSIGNMENTS
311	PROTOCOL DEPENDANT MACROS
354	DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT
382	VERSION 00A FEBRUARY 26, 1975
383	
384	HARVEY M. SCHLESINGER
385	
386	
387	VERSION 00B MARCH 17, 1975
388	CSR AND MICROPROCESSOR CHANGES
389	
390	VERSION 00C NOVEMBER 6, 1975
391	RETRANSMISSION CHANGES
392	
393	VERSION 00D DECEMBER 3, 1975
394	TRANSMIT DONE CHANGES
395	
396	VERSION 01A 6-MAY-77
397	ACKNOWLEDGEMENT CHANGES - ACK A MESSAGE EVEN IF NO DATA TO RETURN
398	
399	VERSION 01B 01-DEC-77
400	MANY CHANGES
401	
402	THE LATEST MODIFICATIONS WERE ADDED ON:
403	01-DEC-77
405	MICROPROCESSOR MAIN MEMORY ASSIGNMENTS
470	SCRATCH PAD ASSIGNMENTS
505	INIT--INITIALIZATION ROUTINE
557	IDLE--PROGRAM IDLE LOOP
569	IDLE2---NO CSR ACTIVITY STATE
668	BASSRV---- BASE SERVICE ROUTINE
711	OUTINT---SET UP OUTPUT INTERRUPT (RDYO)
759	OUTWAI---WAIT FOR RDYO TO GO AWAY
770	CTLSRV--CNTL I SERVICE
784	TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE
813	RBASRV--RECEIVE BUFFER ADDRESS SERVICE
879	RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER
907	RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD
941	RCVC--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD, SELECT AND FINAL
959	RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES
987	RCVE--ROUTINE TO HANDLE N FIELD OF NUMBERED MESSAGE
999	RCVF--ROUTINE TO IGNORE ADDRESS

DDCGAH.MAC

14-DEC-77 15:07

TABLE OF CONTENTS

1007	RCVG--ROUTINE TO IGNORE CRC1
1012	RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYPES
1073	RCVK01--ROUTINE TO HANDLE FIRST BYTE ODD RECEIVE
1081	RCVKE--PROCESS ODD CHARACTER
1105	RCVKE--HANDLE EVEN BYTES
1157	RCVI--STORE UNNUMBERED MESSAGE TYPE
1162	RCVJ--ROUTINE TO HANDLE SUBTYPE FIELD, SELECT AND FINAL
1175	RCVR--UNNUMBERED MESSAGE RESPONSE FIELD
1184	RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD
1190	RCVL--PROCESS CRC3
1209	RCVM--PROCESS CRC4--END OF DATA MESSAGE
1228	EM2--PROCESS RLD MESSAGE
1247	NXMERR ---NON EXISTANT MEMORY HANDLER
1292	TMTA--FIRST CHARACTER OF HEADER
1359	TMTB--OUTPUT FIRST CHAR OF COUNT
1400	TMTC--OUTPUT SECOND CHAR OF COUNT
1410	TMTD--RESPONSE FIELD-NUMBERED MESSAGE
1418	TMTE--NUMBER FIELD--NUMBERED MESSAGE
1425	TMTF--NUMBERED MSG ADDRESS FIELD
1437	TF1-NUMBERED MSG HEADER EOM
1447	TMTH--ROUTINE TO OUTPUT DATA CHARACTERS
1504	TMTI--SEND UNNUMBERED TYPE FIELD
1510	TMTJ--SEND SUB-TYPE FIELD
1514	TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG)
1522	TMTL--UNNUMB MSG NUMBER FIELD
1539	TMTM--UNNUMB MSG--STATION ADDRESS
1558	TIMSRV--TIMEOUT ROUTINE--SENDS REP
1620	SNDACK--ROUTINE TO SEND AN ACK
1669	REP HANDLER
1766	START HANDLER

K11

DMCNEW.MAC

26-OCT-77 13:33

PAGE: 0140DM
SEQ 0140

5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

.TITLE DMC-11 MICROPROCESSOR INSTRUCTIONS
.SBTTL MACRO DEFINITIONS
:
.SBTTL REVISION 00
.SBTTL FEBRUARY 25, 1975
.SBTTL
.SBTTL REVISION 01
.SBTTL MARCH 18, 1975
.SBTTL NEW CSR BOARD CHANGES
.SBTTL
.SBTTL HARVEY M. SCHLESINGER
:
.SBTTL COPYRIGHT 1975 DIGITAL EQUIPMENT CORPORATION
:

```

21 000000
22 000000
23 100000
24 020000
25 000000
26 040000
27 060000
28 060000
29 060000
30 060000
31 010000
32 014000
33 000400
34 001000
35 001400
36 002000
37 002400
38 003000
39 003400
40 000200
41 000220
42 000240
43 000260
44 000300
45 000320
46 000340
47 000360
48 000000
49 000020
50 000040
51 000060
52 000100
53 000120
54 000140
55 000160
56 004000
57 010000
58 014000
59 001000
60 001400
61 000400
62 002000
63 002400
64 003000
65 003400

```

```

NEW=0
; MICROPROCESSOR INSTRUCTION WORD DEFINITIONS
MOVE=0 ; OPCODE MOVE
JUMP=100000 ; OPCODE JUMP
IBUS=20000 ; SOURCE IBUS
IMM=0 ; SOURCE IMMEDIATE
MEMX=40000 ; SOURCE MEMORY
BRX=50000 ; SOURCE BR
BR=50000 ; SOURCE BR

DP=60000 ; SOURCE BR
LDMAR=10000 ; MA-LOAD MAR LO
INCMAR=14000 ; MA-INCREMENT MAR
WRTEBR=400 ; DEST-WRITE BR
WROUTX=1000 ; DEST-EXTENDED IBUS
SHFTBR=1400 ; DEST-SHIFT BR LEFT
WROUT=2000 ; DEST-WRITE OUTPUT
WRMEM=2400 ; DEST-WRITE MEMORY
SPX=3000 ; DEST-WRITE SP
SPBRX=3400 ; DEST-WRITE SP AND BR

; FUNCTIONS
SELA=200 ; FUNCTION-SELECT A
SELB=220 ; FUNCTION-SELECT B
AORNB=240 ; FUNCTION-A OR NOT B
AANDB=260 ; FUNCTION A AND B
AORB=300 ; FUNCTION-A OR B
AXORB=320 ; FUNCTION A XOR B
SUB=340 ; SUBTRACT
SUBTC=360 ; FUNCTION- TWOS COMPLEMENT SUBTRACT
ADD=0 ; ADD A+B
ADDC=20 ; A+B+CARRY
SUBC=40 ; A-B-C
INCA=60 ; INCREMENT A
AC=100 ; A PLUS CARRY
AA=120 ; A PLUS A
AAC=140 ; A PLUS A PLUS C
DECA=160 ; DECREMENT A

; END FUNCTIONS
PAGE1=4000
PAGE2=10000
PAGE3=14000

CCOND=1000 ; CONDITION C
ZCOND=1400 ; CONDITION Z
ALCOND=400 ; ALWAYS
BROCON=2000 ; CONDITION BRO
BR1CON=2400 ; CONDITION BR1
BR4CON=3000 ; CONDITION BR4
BR7CON=3400 ; CONDITION BR7

```

M11

DMCNEW.MAC

26-OCT-77 13:33

MICRO INSTRUCTION DEFINITIONS

PAGE: 0142DM
SEQ 0142

70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250

100000

JUMP=100000

; JUMP OP CODE

.SBTTL

INDEXED BRANCH INSTRUCTIONS

.SBTTL

MOVE INSTRUCTIONS

000000

MOVE=0

; MOVE OP CODE

258		.SBTTL INPUT/OUTPUT ASSIGNMENTS	
259		:IBUS ASSIGNMENTS	
260	100000	INCON=0+100000	: IN CONTROL CSR
261	100020	MAIN=20+100000	: MAINTAINENCE REGISTER
262	100040	OCON=40+100000	: OUT CONTROL CSR
263	100060	UBADDR=60+100000	: UNUSED
264	100100	PORT1=100+100000	: CSR4
265	100120	PORT2=120+100000	: CSR5
266	100140	PORT3=140+100000	: CSR6
267	100160	PORT4=160+100000	: CSR7
268	100200	NPR=200+100000	: NPR CONTROL
269	100220	UBBR=220+100000	: BR (INTERRUPT) CONTROL
270	000000	INDAT1=0	: INPUT DATA LOW BYTE
271	000020	INDAT2=20	: INPUT DATA HIGH BYTE
272	000140	IOBA1=140	: OUTPUT BA LOW BYTE
273	000160	IOBA2=160	: OUTPUT BA HIGH BYTE
274	000100	IIBA1=100	: INPUT BA LOW BYTE
275	000120	IIBA2=120	: INPUT BA HIGH BYTE
276	000200	RCVDAT=200	: RECEIVE DATA
277	000220	TMTCON=220	: TMR CONTROL
278	000240	RCVCON=240	: RCVR CONTROL
279	000260	MODEM=260	: MODEM CONTROL
280	000300	SYNREG=300	: SYN REGISTER
281	000320	LNOSW=320	: LINE NUMBER SWITCH
282	000340	BM873=340	: BM873 ADDRESS
283	000360	LUMAIN=360	: LINE UNIT MAINTAINENCE
284		:OBUS ASSIGNMENTS	
285		:EXTENDED OBUS	
286	000000	OINCON=0	: IN CONTROL CSR
287	000001	OMAIN=1	: MAINT
288	000002	OCON=2	: OUT CONTROL CSR
289	000003	OUBADD=3	: UNUSED
290	000004	OPORT1=4	: CSR4
291	000005	OPORT2=5	: CSR5
292	000006	OPORT3=6	: CSR6
293	000007	OPORT4=7	: CSR7
294	000010	ONPR=10	: NPR CONTROL
295	000011	OBR=11	: BR CONTROL
296		:UNEXTENDED OBUS	
297	000000	OINDAT1=0	: OUTPUT OF
298	000002	OUTDA1=2	: OUTPUT DATA LOW BYTE
299	000003	OUTDA2=3	: OUTPUT DATA HIGH BYTE
300	000006	OBA1=6	: OUTPUT BA LOW BYTE
301	000007	OBA2=7	: OUTPUT BA HIGH BYTE
302	000004	IIBA1=4	: INPUT BA LOW BYTE
303	000005	IIBA2=5	: INPUT BA HIGH BYTE
304	000010	TMTDAT=10	: TMR DATA
305	000011	OTMTCO=11	: TMR CONTROL
306	000012	ORCVCO=12	: RCVR CONTROL
307	000013	OMODEM=13	: MODEM CONTROL
308	000014	SYNC=14	: SYN REGISTER
309	000017	OLUMAN=17	: LINE UNIT MAINT.

DMCNEW.MAC

26-OCT-77 13:33

PROTOCOL DEPENDANT MACROS

PAGE: 0144DM
SEQ 0144

.SBTTL PROTOCOL DEPENDANT MACROS

.....

177777

MICPC=177777 :INIT MICRO PC

311
310
309
308
307
306
305
304
303
302
301
300
299
298
297
296
295
294
293
292
291
290
289
288
287
286
285
284
283
282
281
280
279
278
277
276
275
274
273
272
271
270
269
268
267
266
265
264
263
262
261
260
259
258
257
256
255
254
253
252
251
250
249
248
247
246
245
244
243
242
241
240
239
238
237
236
235
234
233
232
231
230
229
228
227
226
225
224
223
222
221
220
219
218
217
216
215
214
213
212
211
210
209
208
207
206
205
204
203
202
201
200
199
198
197
196
195
194
193
192
191
190
189
188
187
186
185
184
183
182
181
180
179
178
177
176
175
174
173
172
171
170
169
168
167
166
165
164
163
162
161
160
159
158
157
156
155
154
153
152
151
150
149
148
147
146
145
144
143
142
141
140
139
138
137
136
135
134
133
132
131
130
129
128
127
126
125
124
123
122
121
120
119
118
117
116
115
114
113
112
111
110
109
108
107
106
105
104
103
102
101
100
99
98
97
96
95
94
93
92
91
90
89
88
87
86
85
84
83
82
81
80
79
78
77
76
75
74
73
72
71
70
69
68
67
66
65
64
63
62
61
60
59
58
57
56
55
54
53
52
51
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

C12

HI1.MAC 22-AUG-77 17:55

DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT

PAGE: 01450M
SEQ 0195

354
355

000000

.SBTTL DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT
LOW=0

DDCGAH.MAC

14-DEC-77 15:07

DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT

.TITLE DMC11 DDCMP PROTOCOL IMPLEMENTATION

.....
 COPYRIGHT (C) 1977
 DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

.....
 THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A
 SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE
 INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR
 ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE
 MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH
 SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE
 TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN
 IN DEC.

.....
 THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT
 NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
 EQUIPMENT CORPORATION.

.....
 DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF
 ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

.....
 .SBTTL VERSION 00A FEBRUARY 26, 1975

.....
 .SBTTL

.....
 .SBTTL HARVEY M. SCHLESINGER

.....
 .SBTTL

.....
 .SBTTL

.....
 .SBTTL VERSION 00B MARCH 17, 1975
 CSR AND MICROPROCESSOR CHANGES

.....
 .SBTTL

.....
 .SBTTL VERSION 00C NOVEMBER 6, 1975
 RETRANSMISSION CHANGES

.....
 .SBTTL

.....
 .SBTTL VERSION 00D DECEMBER 3, 1975
 TRANSMIT DONE CHANGES

.....
 .SBTTL

.....
 .SBTTL VERSION 01A 6-MAY-77
 ACKNOWLEDGEMENT CHANGES - ACK A MESSAGE EVEN IF NO DATA TO RETURN

.....
 .SBTTL

.....
 .SBTTL VERSION 01B 01-DEC-77
 MANY CHANGES

.....
 .SBTTL

.....
 .SBTTL THE LATEST MODIFICATIONS WERE ADDED ON:
 01-DEC-77

.....
 .SBTTL

360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376
 377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403

DDCGAH.MAC

14-DEC-77 15:07

MICROPROCESSOR MAIN MEMORY ASSIGNMENTS

```

405
406
407 000000
408 000001
409 000002
410 000003
411 000006
412 000007
413 000010
414 000011
415 000012
416 000013
417 000014
418 000015
419 000016
420 000017
421 000022
422 000023
423 000024
424 000031
425 000036
426 000043
427 000050
428 000055
429 000062
430 000067
431 000070
432 000071
433 000077
434 000105
435 000113
436 000121
437 000127
438 000135
439 000143
440 000151
441 000152
442 000153
443 000154
444 000155
445 000156
446 000157
447 000160
448 000161
449 000162
450
451
452 000164
453
454
455
456 000167
457 000171
458 000172
459 000173
460 000174

```

```

.SBTTL MICROPROCESSOR MAIN MEMORY ASSIGNMENTS
;ALLOCATION OF MICROPROCESSOR MAIN MEMORY
NAKSR=0 ;NAKS RECD--DYNAMIC
NAKST=NAKSR+1 ;NAKS TMTD--DYNAMIC
REPSR=NAKST+1 ;REPS RECD--DYNAMIC
REPST=REPSR+1 ;REPS TMTD--DYNAMIC
NP=REPST+3 ;CONSTANT 0
NHLR=NP+1 ;NAKS-MSG NO BUFFERS CUMUL.
NHDR=NLHR+1 ;NAKS-MSG HEADER BAD
NDATR=NHDR+1 ;NAKS-DATA BAD
NTLS=NDATR+1 ;NAK SENT --NO BUFFERS
NHDS=NTLS+1 ;NAK SENT BAD HEADER
NDATS=NHDS+1 ;NAK SENT BAD DATA
REPCS=NDATS+1 ;REPS SENT CUMUL
REPCR=REPCS+1 ;REPS RECD CUMUL
BASE=REPCR+1 ;CORE TABLE BASE ADDRESS
SRC=BASE+3 ;START OF INPUT CHAIN--NEXT RECV DONE
ERC=SRC+1 ;END OF INPUT CHAIN
RCL1=ERC+1 ;RECEIVE LINK #1
RCL2=RCL1+5 ;" " #2
RCL3=RCL2+5 ;" " #3
RCL4=RCL3+5
RCL5=RCL4+5
RCL6=RCL5+5
RCL7=RCL6+5
STC=RCL7+5 ;START OF OUTPUT CHAIN---NEXT TMT DONE
ETC=STC+1 ;END OF TRANSMIT CHAIN
TML1=ETC+1 ;TRANSMIT LINK #1
TML2=TML1+6 ;" " #2
TML3=TML2+6 ;" " #3
TML4=TML3+6
TML5=TML4+6
TML6=TML5+6
TML7=TML6+6
TML8=TML7+6
T=TML8+6
ST=T+1 ;SUBTYPE FIELD
;SP17=ST+1 ;MSG ACKED IMAGE
IMG10=ISP17+1 ;IMAGE OF BIT 1 OF SP10
IMG11=IMG10+1 ;IMAGE OF SP11
IMG12=IMG11+1 ;IMAGE OF SP12
IMG14=IMG12+1 ;IMAGE OF SP14
IMG16=IMG14+1 ;IMAGE OF SP16
IMG17=IMG16+1 ;IMAGE OF SP17
TYPTAB=IMG17+1 ;TYPE TABLE---
;72 TYPE TABLE REP
;73 " " NAK
;74 " " START
;75 " " STACK
BC=TYPTAB+3 ;RECEIVE BYTE COUNT
ISP11=BC+2 ;SP11 IMAGE
ISP12=ISP11+1 ;SP12 IMAGE
INCONS=ISP12+1 ;IN CONTROL CSR IMAGE
RTHRS=INCONS+1 ;RECV THRESHOLD LINK

```


F12

DDCGAH.MAC

14-DEC-77 15:07

MICROPROCESSOR MAIN MEMORY ASSIGNMENTS

:ALL LOCATIONS FROM 200 ON ARE NOT WRITTEN OUT DURING A TABLE UPDATE

461	000210
462	000211
463	000240
464	000241
465	000242
466	000400
467	
468	

TABST=210	:TABLE UPDATE STATE
PRIST=TABST+1	:PORT STATE
NXTINT=240	:NEXT INTERRUPT POSITION
NXTSP=NXTINT+1	:END OF INTERRUPT CHAIN
INTSTK=NXTSF+1	:STACK OF INTERRUPTS
MMEND=400	:MAIN MEMORY ENC

DCCGH.MAC

14-DEC-77 15:07

SCRATCH PAD ASSIGNMENTS

```

4800 000000 .SBTTL SCRATCH PAD ASSIGNMENTS
4801 000001 SP0=0 :SP0---SCRATCH REGISTER
4802 SP1=1 :SP1---PORT STATUS WORD
4803 :BIT ASSIGNMENTS
4804 :BIT0--INIT MODE
4805 :BIT1--NO BUFFER ASSIGNED IN BOOT MODE
4806 :BIT2--UNUSED
4807 :BIT3--DLE RECEIVED WHILE NOT IN MAINT MODE
4808 :BIT4--INTERRUPT PENDING
4809 :BIT6--DISCONNECT ERROR
4810 :BIT7--BOOT MODE
4811 000002 SP2=2 :SP2---TRANSMIT STATE POINTER
4812 000003 SP3=3 :SP3---RECEIVE STATE POINTER
4813 000004 SP4=4 :SP4---END RECV ADDRESS
4814 000005 SP5=5 :SP5---END RECEIVE ADDRESS
4815 000006 SP6=6 :SP6---END TRANSMIT ADDRESS
4816 000007 SP7=7 :SP7---END TRANSMIT ADDRESS
4817 000010 SP10=10 :SP10---LINE STATUS WORD
4818 :BIT ASSIGNMENTS
4819 :BIT0--UNNUMB PENDING
4820 :BIT1--MESSAGE IN PROGRESS
4821 :BIT2--LINE HAS GONE IDLE
4822 :BIT3--START RECEIVED
4823 :BIT4--CLEAR ACTIVE ON END
4824 :BIT5--START MODE
4825 :BIT6--HALF DUPLEX
4826 :BIT7--OK TO SEND
4827 000011 SP11=11 :SP11---R FIELD
4828 000012 SP12=12 :SP12---N FIELD
4829 000013 SP13=13 :SP13---TYPE
4830 000014 SP14=14 :SP14---TIMER ENTRY---NUMBER OF ONE SECOND TICKS
4831 000015 SP15=15 :SP15---RECEIVE LINK IMAGE
4832 000016 SP16=16 :SP16---POINTER TO TMT LINK COPY IN MAIN MEM
4833 000017 SP17=17 :SP17---LAST MESSAGE ACKNOWLEDGED

```

CDCGAM.MAC 14-DEC-77 15:07

INIT--INITIALIZATION ROUTINE

```

505 .SBTTL INIT--INITIALIZATION ROUTINE
506 :ZEROS MAIN MEMORY
507 :LOOPS WAITING FOR RECEIVE DATA(BOOT?)
508 :OR FOR RQI TO BE SET
509 :WILL ACCEPT ONLY BASE FORMAT. ALL OTHERS WILL RETURN A PROCEDURE ERRJR
510 :
511 :AT INITIALIZATION --- THE HARDWARE CLEARS THE BR AND MAR
512 :=16322
513 INIT: SP BR,SELB,SP0 ;CLEAR SP0
514 (1) 016322 000000 MICPC=MICPC+1
515 (1) 016322 063220 <MOVE!SPX!BR!SELB!SP0>
516 (1) 016324 063224 SP BR,SELB,SP3 ;PAGE ONE TRANSFER ADDRESS
517 (1) 016324 000001 MICPC=MICPC+1
518 (1) 016324 063223 <MOVE!SPX!BR!SELB!SP3>
519 (1) 016326 000002 SP BR,SELB,SP17 ;CLEAR SP17
520 (1) 016326 063237 MICPC=MICPC+1
521 (1) 016330 000003 <MOVE!SPX!BR!SELB!SP17>
522 (1) 016330 063232 SP BR,SELB,SP12 ;CLEAR SP12
523 (1) 016332 000004 OUT BR,<SELA!OINCON> ;ZERO THE IN CONTROL CSR
524 (1) 016332 061200 MICPC=MICPC+1
525 (1) 016334 061200 <MOVE!WROUT!BR!<SELA!OINCON>> ;ZERO THE OUT CONTROL CSR
526 (1) 016334 000005 OUT F,SELA!OCON
527 (1) 016334 061202 MICPC=MICPC+1
528 (1) 016336 061202 <MOVE!WF,X!BR!<SELA!OCON>> ;WRITE 5 ONE BITS TO THE HIGH ORDER
529 (1) 016336 003370 SP 1,370,SP10
530 (1) 016336 003370 MICPC=MICPC+1
531 (1) 016336 003370 <MOVE!SPX!IMM!370!SP10> ;BITS OF SP10
532 (1) 016340 000007 SS: SP BR,AA,SP10 ;SHIFT SP10 LEFT SETTING CARRY THE
533 (1) 016340 063130 MICPC=MICPC+1
534 (1) 016340 063130 <MOVE!SPX!BR!AA!SP10> ;FIRST 5 TIMES THRU THE LOOP
535 (1) 016342 000010 MEMINC BR,ADDC!SP3 ;WRITE A ONE TO THE FIRST 5 MEMORY
536 (1) 016342 076423 MICPC=MICPC+1
537 (1) 016342 076423 <MOVE!WRMEM!INCMAR!BR!<ADDC!SP3>> ;LOCATIONS AND ZERO THE REST
538 (1) 016344 000011 SP BR,INCA,SP0 ;INCREMENT COUNTER
539 (1) 016344 063060 MICPC=MICPC+1
540 (1) 016344 063060 <MOVE!SPX!BR!INCA!SP0>
541 (1) 016346 000012 Z 10$ ;ALL DONE
542 (1) 016346 101414 MICPC=MICPC+1
543 (1) 016346 101414 <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
544 (1) 016350 000013 ALWAYS SS ;KEEP GOING
545 (1) 016350 100407 MICPC=MICPC+1
546 (1) 016350 100407 <JUMP!ALCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
547 (1) 016352 000014 10$: SPBR IMM,1,SP1 ;WRITE A 1 TO THE BR AND SP1
548 (1) 016352 003471 MICPC=MICPC+1
549 (1) 016352 003471 <MOVE!SPBRX!IMM!1!SP1>
550 (1) 016354 000015 SP BR,SELB,SP11 ;WRITE A 1 TO SP11
551 (1) 016354 063231 MICPC=MICPC+1
552 (1) 016354 063231 <MOVE!SPX!BR!SELB!SP11>
553 (1) 016356 000016 LDMA IMM,TYPTAB ;POINT MAR TO TYPE TABLE
554 (1) 016356 010162 MICPC=MICPC+1
555 (1) 016356 010162 <MOVE!LDMAR!IMM!<TYPTAB&377>>

```

DDCGAM.MAC 14-DEC-77 15:07

INIT--INITIALIZATION ROUTINE

```

531 016360 000017 BRWRT IMM,226 ;WRITE SYNC TO MEMORY
(1) 016360 000626 MICPC=MICPC+1
532 016362 000020 <MOVE!WRTEBR!IMM!<226>>
(1) 016362 062234 OUTPUT BR SELB!SYNC ;LOAD THE SYNC REGISTER
533 016364 000021 MICPC=MICPC+1
(1) 016364 016403 <MOVE!WROUT!BR!<SELB!SYNC>> ;REP
534 016366 000022 MEMINC IMM,3
(1) 016366 016402 MICPC=MICPC+1 <MOVE!WRMEM!INCMAR!IMM!<3>> ;NAK
535 016370 000022 MEMINC IMM,2
(1) 016370 016406 MICPC=MICPC+1 <MOVE!WRMEM!INCMAR!IMM!<2>> ;START
536 016372 000023 MEMINC IMM,6
(1) 016372 016406 MICPC=MICPC+1 <MOVE!WRMEM!INCMAR!IMM!<6>> ;STACK
537 016374 000024 MEMINC IMM,7
(1) 016374 016401 MICPC=MICPC+1 <MOVE!WRMEM!INCMAR!IMM!<7>> ;ACK
538 016376 000025 MEMINC IMM,1
(1) 016376 010210 LDMA IMM,TABST ;POINT TO TABLE UPDATE STATE
539 016400 000026 MICPC=MICPC+1 <MOVE!LDMAR!IMM!<TABST&377>> ;INITIALIZE IT
(1) 016400 016455 PSTATI I3
(2) 016400 000027 MEMINC IMM,<<I3-INIT&777/2>>
(2) 016400 016455 MICPC=MICPC+1 <MOVE!WRMEM!INCMAR!IMM!<<I3-INIT&777/2>>> ;INITIALIZE PORT STATUS
540 016402 000030 PSTATI NIDLE2
(1) 016402 016472 MEMINC IMM,<<NIDLE2-INIT&777/2>>
(2) 016402 000031 MICPC=MICPC+1
(2) 016402 016472 <MOVE!WRMEM!INCMAR!IMM!<<NIDLE2-INIT&777/2>>>
541 016404 000031 LDMA IMM,STC ;LOAD ADDRESS OF LAST TMT CHAIN
(1) 016404 010067 MICPC=MICPC+1
542 016406 000032 <MOVE!LDMAR!IMM!<STC&377>> ;STORE ADDRESS OF FIRST TMT LINK
(1) 016406 016471 MEMINC IMM,TML1
543 016410 000033 MICPC=MICPC+1
(1) 016410 002471 <MOVE!WRMEM!INCMAR!IMM!<TML1>>
544 016412 000034 MEM IMM,TML1
(1) 016412 043236 MICPC=MICPC+1 SP MEMX,SELB,SP16 ;INITIALIZE LAST XMIT POINTER
545 016414 000035 <MOVE!SPX!MEMX!SELB!SP16>
(1) 016414 010022 LDMA IMM,SRC ;LOAD ADDRESS OF LAST RECV CHAIN
546 016416 000036 MICPC=MICPC+1
(1) 016416 016424 <MOVE!LDMAR!IMM!<SRC&377>> ;SET UP ADDRESS OF FIRST RECV LINK
547 016420 000037 MEMINC IMM,RCL1
(1) 016420 002424 MICPC=MICPC+1 <MOVE!WRMEM!INCMAR!IMM!<RCL1>>
548 016422 000040 MEM IMM,RCL1
(1) 016422 043235 SP MEMX,SELB,SP15
(1) 016422 043235 MICPC=MICPC+1
(1) 016422 043235 <MOVE!SPX!MEMX!SELB!SP15

```

```

549 016424          LDMA IMM,NXTINT          ;ADDRESS OF NEXT INTERRUPT POINTER TO MAR
(1) (1) 016424 000041 MICPC=MICPC+1
(1) (1) 016424 010240 <MOVE!LDMAR!IMM!<NXTINT&377>>
550 016426          MEMINC IMM,INTSTK        ;INITIALIZE NEXT INTERRUPT POINTER
(1) (1) 016426 000042 MICPC=MICPC+1
(1) (1) 016426 016642 <MOVE!WRMEM!INCMAR!IMM!<INTSTK>>
551 016430          MEM IMM,INTSTK          ;INITIALIZE INSERTION POINTER
(1) (1) 016430 000043 MICPC=MICPC+1
(1) (1) 016430 002642 <MOVE!WRMEM!IMM!<INTSTK>>
552 016432          BRWRT IMM,200           ;MASK TO SET RUN BIT
(1) (1) 016432 000044 MICPC=MICPC+1
(1) (1) 016432 000600 <MOVE!WRTEBR!IMM!<200>>
553 016434          OUT BR,<SELB!OMAIN>      ;WRITE THE RUN BIT TO MAINT CSR
(1) (1) 016434 000045 MICPC=MICPC+1
(1) (1) 016434 061221 <MOVE!WROUTX!BR!<SELB!OMAIN>>
554 016436          ALWAS TEOM2            ;FALL INTO IDLE LOOP
(1) (1) 016436 000046 MICPC=MICPC+1
(1) (1) 016436 110642 <JUMP!ALCOND!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>

```

IDLE--PROGRAM IDLE LOOP

```

557      .SBTTL IDLE--PROGRAM IDLE LOOP
558      :PROGRAM IDLE LOOP
559      :DISPATCHES TO APPROPRIATE SERVICE ROUTINES
560      :USES STATE POINTERS FOR TMT,RCV,CSR ACTIVITY
561      ;
562 016440 IDLE: BRWRT  IBUS,TMTCON
563 016440      MICPC=MICPC+1
(1)      000047      <MOVE!WRTEBR!IBUS!<TMTCON>>
(1) 016440 020620      .BR4  DP,SELA,<2!PAGE2>
564 016442      MICPC=MICPC+1
(1)      000050      <JUMP!BR4CON!DP!SELA!2!PAGE2>
(1) 016442 173202      I1: BRWRT  IBUS,RCVCON      ;READ LINE UNIT RECEIVE CONTROL WORD
565 016444      MICPC=MICPC+1
(1)      000051      <MOVE!WRTEBR!IBUS!<RCVCON>>
(1) 016444 020640      .BR4  BR,SELA,SP3!PAGE1      ;BRANCH BASED UPON RCV STATE
566 016446      MICPC=MICPC+1
(1)      000052      <JUMP!BR4CON!BR!SELA!SP3!PAGE1>
(1) 016446 167203      I2: LDMA  IMM,TABST      ;POINT TO TABLE UPDATE STATE
567 016450      MICPC=MICPC+1
(1)      000053      <MOVE!LDMAR!IMM!<TABST&377>>
(1) 016450 010210      .ALWAY MEMX,SELB,0
568 016452      MICPC=MICPC+1
(1)      000054      <JUMP!ALCOND!MEMX!SELB!0>
569 016454      I3: BRWRT  IBUS,TMTCON
571 016454      MICPC=MICPC+1
(1)      000055      <MOVE!WRTEBR!IBUS!<TMTCON>>
(1) 016454 020620      BR4  I4
572 016456      MICPC=MICPC+1
(1)      000056      <JUMP!BR4CON!<I4-INIT&3000*4>!<I4-INIT&777/2>>
(1) 016456 103060      ALWAYS IDLEO
573 C16460      MICPC=MICPC+1
(1)      000057      <JUMP!ALCOND!<IDLEO-INIT&3000*4>!<IDLEO-INIT&777/2>>
(1) 016460 100463      I4: STATE  TMTAS
574 016462      MICPC=MICPC+1
(1)      000060      <MOVE!WRTEBR!IMM!<TMTAS-INIT&777/2>>
(1) 016462 000433      NOP  BR,SUB,SP2
575 016464      MICPC=MICPC+1
(1)      000061      <BR!SUB!SP2>
(1) 016464 060342      C  IDLE
576 016466      MICPC=MICPC+1
(1)      000062      <JUMP!CCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
578 016470 IDLEO: SPBR  IBUS,UBR,SPO      ;TIMER EXPIRES?
(1)      000063      MICPC=MICPC+1
(1) 016470 123620      <MOVE!SPBRX!IBUS!UBR!SPO>
579 016472      BR4  TIMSRV
(1)      000064      MICPC=MICPC+1
(1) 016472 113246      <JUMP!BR4CON!<TIMSRV-INIT&3000*4>!<TIMSRV-INIT&777/2>>
581 016474      SP  IBUS,RCVCON,SPO      ;READ RCV CONTROL REGISTER
(1)      000065      MICPC=MICPC+1
(1) 016474 023240      <MOVE!SPX!IBUS!RCVCON!SPO>
582 016476      BRWRT  BR,AA!SPO      ;SHIFT LEFT
(1)      000066      MICPC=MICPC+1
(1) 016476 060520      <MOVE!WRTEBR!BR!<AA!SPO>>
583 016500      BR7  I1      ;RCV ACTIVE, DON'T DO PORT STATUS

```

DDCGAH.MAC 14-DEC-77 15:07

IDLE--PROGRAM IDLE LOOP

PAGE: 0154DM
SEQ 0154

```
(1) 000067  
(1) 016500 103451  
585 016502  
(1) 000070  
(1) 016502 010211  
586 016504  
(1) 000071  
(1) 016504 140620
```

```
MICPC=MICPC+1  
<JUMP!BR7CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>  
LDMA IMM, PRTST ;ADDRESS PORT STATE  
MICPC=MICPC+1  
<MOVE!LDMAR!IMM!<PRTST&377>>  
.ALWAY MEMX, SELB, 0 : INDEX  
MICPC=MICPC+1  
<JUMP!ALCOND!MEMX!SELB!0>
```

DDCGAH.MAC 14-DEC-77 15:07

NIDLE2---NO CSR ACTIVITY STATE

```

589 .SBTTL NIDLE2---NO CSR ACTIVITY STATE
590 016506 BRWRITE BR, SELA!SP1 ;READ PORT STATUS WORD
(1) 000072
(1) 016506 060601
592 016510 BR4 NIDLE5 ; INTERRUPT PENDING?---BRANCH
(1) 000073
(1) 016510 103100
597 016512 <MOVE!WRTEBR!BR!<SELA!SP1>>
(1) 000074
(1) 016512 123400
598 016514 SPBR IBUS INCON,SPO ;READ INPUT CONTROL CSR
(1) 000075
(1) 016514 001620
599 016516 <MOVE!SPBRX!IBUS!INCON!SPO>
(1) 000076
(1) 016516 117034
600 BRSHFT ;SHIFT IT RIGHT
601
603 016520 ALWAYS IDLE
(1) 000077
(1) 016520 100447
605 016522 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
626 016522 NIDLE6: PSTATE OUTINT ;SET STATE FOR INTERRUPT PROCESSING
(1) 016522 MEM IMM,<<OUTINT-INIT&777/2>>
(2) 000100
(2) 016522 MICPC=MICPC+1
627 016524 <MOVE!WRMEM!IMM!<<OUTINT-INIT&777/2>>>
(1) 000101
(1) 016524 ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
;TO RE-READ THE IN CNTRL REGISTER TO AVOID
;A RACE IN MICRO-P READ/UNIBUS WRITE

```


DDCGAH.MAC 14-DEC-77 15:07

NIDLE2---NO CSR ACTIVITY STATE

630	016526	000102	IEIWAT: SPBR IBUS, INCON, SPD	; READ INPUT CONTROL CSR
(1)		123400	MICPC=MICPC+1	
(1)	016526		<MOVE!SPBRX!IBUS!INCON!SPD>	
631	016530		BR7 10\$; ROYI STILL SET
(1)		000103	MICPC=MICPC+1	
(1)	016530	103505	<JUMP!BR7CON!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>	
632	016532		ALWAYS INWAT1	; GOT CLEARED BY 11 RE-SET IT
(1)		000104	MICPC=MICPC+1	
(1)	016532	114434	<JUMP!ALCOND!<INWAT1-INIT&3000*4>!<INWAT1-INIT&777/2>>	
633	016534		10\$: BRWRT BR <AA!SPD>	; CHECK IEI
(1)		000105	MICPC=MICPC+1	
(1)	016534	060520	<MOVE!WRTEBR!BR!<AA!SPD>>	
634	016536		BR7 ININT	; IEI SET, GENERATE INTERRUPT
(1)		000106	MICPC=MICPC+1	
(1)	016536	103603	<JUMP!BR7CON!<ININT-INIT&3000*4>!<ININT-INIT&777/2>>	
635	016540		INWAT2: BRWRT IBUS, INCON	; READ INPUT CONTROL CSR
(1)		000107	MICPC=MICPC+1	
(1)	016540	120400	<MOVE!WRTEBR!IBUS!<INCON>>	
636	016542		BRSHFT	
(1)		000110	MICPC=MICPC+1	
(1)	016542	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
641	016544		BR4 IDLE	; RQI STILL SET... GO AWAY
(1)		000111	MICPC=MICPC+1	
(1)	016544	103047	<JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
642	016546		PSTATE INSRV	; SET NEXT STATE TO INPUT SERVICE
(1)	016546		MEM IMM, <<INSRV-INIT&777/2>>	
(2)		000112	MICPC=MICPC+1	
(2)	016546	002514	<MOVE!WRMEM!IMM!<<INSRV-INIT&777/2>>>	
643	016550		ALWAYS IDLE	
(1)		000113	MICPC=MICPC+1	
(1)	016550	100447	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
645	016552		INSRV: SPBR IBUS, INCON, SPD	; READ THE INPUT CONTROL CSR
(1)		000114	MICPC=MICPC+1	
(1)	016552	123400	<MOVE!SPBRX!IBUS!INCON!SPD>	
646	016554		BR1 30\$; --SENSE OR BASE
(1)		000115	MICPC=MICPC+1	
(1)	016554	102527	<JUMP!BR1CON!<30\$-INIT&3000*4>!<30\$-INIT&777/2>>	
647	016556		PSTATE CTLSRV	; ASSUME CNTL I
(1)	016556		MEM IMM, <<CTLSRV-INIT&777/2>>	
(2)		000116	MICPC=MICPC+1	
(2)	016556	002654	<MOVE!WRMEM!IMM!<<CTLSRV-INIT&777/2>>>	
648	016560		BR0 20\$; CNTL I
(1)		000117	MICPC=MICPC+1	
(1)	016560	102124	<JUMP!BR0CON!<20\$-INIT&3000*4>!<20\$-INIT&777/2>>	
649	016562		BRSHFT	; MUST BE BA/CC
(1)		000120	MICPC=MICPC+1	
(1)	016562	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
650	016564		PSTATE RBASRV	; ASSUME IN
(1)	016564		MEM IMM, <<RBASRV-INIT&777/2>>	
(2)		000121	MICPC=MICPC+1	
(2)	016564	002722	<MOVE!WRMEM!IMM!<<RBASRV-INIT&777/2>>>	
651	016566		BR1 20\$; REC BA/CC
(1)		000122	MICPC=MICPC+1	
(1)	016566	102524	<JUMP!BR1CON!<20\$-INIT&3000*4>!<20\$-INIT&777/2>>	
652	016570		PSTATE TBASRV	; XMIT BA/CC
(1)	016570		MEM IMM, <<TBASRV-INIT&777/2>>	

DDCGAH.MAC 14-DEC-77 15:07

NIDLE2---NO CSR ACTIVITY STATE

```

2) 000123
(2) 016570 002701
653 016572
(1) 000124
(1) 016572 060601
654 016574
(1) 000125
(1) 016574 102130
655 016576
(1) 000126
(1) 016576 100447
656 016600
(1) 000127
(1) 016600 102140
657 016602
(1) 016602
(2) 000130
(2) 016602 002472
658 016604
(1) 000131
(1) 016604 000500
659 016606
(1) 000132
(1) 016606 061260
660 016610
(1) 000133
(1) 016610 010177
661 016612
(1) 000134
(1) 016612 016402
662 016614
(1) 000135
(1) 016614 002400
663 016616
(1) 000136
(1) 016616 042233
664 016620
(1) 000137
(1) 016620 114474
665 016622
(1) 000140
(1) 016622 060721
666 016624
(1) 000141
(1) 016624 102130

```

```

MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<TBASRV-INIT&777/2>>>
20$: BRWRTÉ BR SELA!SP1 ;INIT MODE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
BRD PPOCER ;IF INIT MODE--ERROR
MICPC=MICPC+1
<JUMP!BROCON!<PROCER-INIT&3000*4>!<PROCER-INIT&777/2>>
ALWAYS IDLE
MICPC=MICPC+1
30$: <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
BRD INSRV1 ;IF BASE---PROCESS
MICPC=MICPC+1
<JUMP!BROCON!<INSRV1-INIT&3000*4>!<INSRV1-INIT&777/2>>
PROCER: PSTATE NIDLE2 ;RESET PORT STATUS
MEM IMM,<<NIDLE2-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>
BRWRTÉ IMM,100 ;CLEAR INPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<100>>
OUT BR AANDB!OINCON ;
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AANDB!OINCON>>
LDMÁ IMM,<<RTHRS+3>> ;ADDRESS ERROR LINK
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<<RTHRS+3 &377>>
MEMINC IMM,2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<2>>
MEM IMM,0
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<0>>
OUTPUT MEMX SELB!OMODEM ;CLEAR DATA TERMINAL READY
MICPC=MICPC+1
<MOVE!WROUT!MEMX!<SELB!OMODEM>>
ALWAYS RCEXX ;POST THE ERROR - FATAL
MICPC=MICPC+1
<JUMP!ALCOND!<RCEXX-INIT&3000*4>!<RCEXX-INIT&777/2>>
INSRV1: BRWRTÉ BR AXORB!SP1 ;INIT MODE?
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AXORB!SP1>>
BRD PROCER
MICPC=MICPC+1
<JUMP!BROCON!<PROCER-INIT&3000*4>!<PROCER-INIT&777/2>>

```

```

668      .SBTTL BASSRV---- BASE SERVICE ROUTINE
669      PSTATE NIDLE2
670      MEM IMM, <<NIDLE2-INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>
671      LDMA IMM, BASE ;CLEAR TO MAR SO IT POINTS TO BASE POINT
        MICPC=MICPC+1
        <MOVE!LDMAR!IMM!<BASE&377>>
672      MEMINC IBUS, PORT1 ;READ CSR4
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IBUS!<PORT1>>
673      MEMINC IBUS, PORT2 ;READ CSRS
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IBUS!<PORT2>>
674      MEM IBUS, PORT4
        MICPC=MICPC+1
        <MOVE!WRMEM!IBUS!<PORT4>>
675      SP IBUS, INCON, SPO ;READ INPUT CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!INCON!SPO>
        BRWRT IMM, 100 ;CLEAR THE BR
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<100>>
676      OUT BR, <AANDB!OINCON> ;CLEAR THE INCONTROL CSR
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<AANDB!OINCON>>
677      OUTPUT IMM, <120!OMODEM> ;MASK FOR HDX AND LTR
        MICPC=MICPC+1
        <MOVE!WROUT!IMM!<120!OMODEM>>
678      BRWRT MEMX, SELB ;READ SEL6
        MICPC=MICPC+1
        <MOVE!WRTEBR!MEMX!<SELB>>
679      BR4 RESUME ;IF SET RESUME
        MICPC=MICPC+1
        <JUMP!BR4CON!<RESUME-INIT&3000*4>!<RESUME-INIT&777/2>>
680      LDMA IMM, T ;LOAD ADDRESS OF TYPE FIELD
        MICPC=MICPC+1
        <MOVE!LDMAR!IMM!<T&377>>
681      MEMINC IMM, 6 ;WRITE START TYPE TO MEMORY
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IMM!<6>>
682      MEM IMM, 30C ;WRITE SELECT AND FINAL TO MEMORY
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<300>>
683      SP BR, DECA, SP1 ;TURN OFF INIT MODE
        MICPC=MICPC+1
        <MOVE!SPX!BR!DECA!SP1>
684      BRWRT IMM, 241 ;SET OK TO SEND, STARTMODE AND JNNUM PENDING
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<241>>
685      SP IMM, 374, SP14 ;RESET REP COUNTER
        MICPC=MICPC+1
        <MOVE!SPX!IMM!374!SP14>
        ALWAYS SA3
        MICPC=MICPC+1
        <JUMP!ALCOND!<SA3-INIT&3000*4>!<SA3-INIT&777/2>>
        1) 016626 000142
        1) 016626 002472
        1) 016630 000143
        1) 016630 010017
        1) 016632 000144
        1) 016632 136500
        1) 016634 000145
        1) 016634 136520
        1) 016636 000146
        1) 016636 122560
        1) 016640 000147
        1) 016640 123000
        1) 016642 000150
        1) 016642 000500
        1) 016644 000151
        1) 016644 061260
        1) 016646 000152
        1) 016646 002133
        1) 016650 000153
        1) 016650 040620
        1) 016652 000154
        1) 016652 103164
        1) 016654 000155
        1) 016654 010151
        1) 016656 000156
        1) 016656 016406
        1) 016660 000157
        1) 016660 002700
        1) 016662 000160
        1) 016662 063161
        1) 016664 000161
        1) 016664 000641
        1) 016666 000162
        1) 016666 003374
        1) 016670 000163
        1) 016670 110727

```

DDCGAM.MAC 14-DEC-77 15:07

BASSRV---- BASE SERVICE ROUTINE

687 016672 000164
 (1) 016672 003004
 688 016674 000165
 (1) 016674 063070
 689 016676 000166
 (1) 016676 010017
 691 016700 000167
 (1) 016700 000731
 692 016702 000170
 (1) 016702 110463
 693 016704 000171
 (1) 016704 010154
 694 016706 000172
 (1) 016706 057310
 695 016710 000173
 (1) 016710 057231
 696 016712 000174
 (1) 016712 057235
 697 016714 000175
 (1) 016714 043237
 698 016716 000176
 (1) 016716 043232
 699 016720 000177
 (1) 016720 063170
 700 016722 000200
 (1) 016722 063161
 702 016724 000201
 (1) 016724 000606
 703 016726 000202
 (1) 016726 114424
 704 016730 000203
 (1) 016730 002507
 705 016732 000204
 (1) 016732 000415
 706 016734 000205

```

RESUME: SP      IMM,SP4,4           ;SET UP SP4 FOR COUNTING NPRS
        MICPC=MICPC+1
        <MOVE!SPX!IMM!SP4!4>
        SP      BR INCA,SP10       ;SET UNNUMB MESSAGE PENDING TO
        MICPC=MICPC+1
        <MOVE!SPX!BR!INCA!SP10>

        LDMA    IMM,BASE           ;TRICK TRANSMITTER CODE
        MICPC=MICPC+1              ;ADDRESS BASE TABLE ADDRESS
        <MOVE!LDMAR!IMM!<BASE&377>>
        STATE   FUDGE              ;SET TMTR STATE TO ENTER TABLE UPDATE
        MICPC=MICPC+1
        <MOVE!WTEBR!IMM!<FUDGE-INIT&777/2>>
        ALWAYS  TBO                 ;GO SET UP MXT BITS AND ADRESS OF BASE FOR NPRS
        MICPC=MICPC+1
        <JUMP!ALCOND!<TBO-INIT&3000*4>!<TBO-INIT&777/2>>

BS2:    LDMA    IMM,IMG10
        MICPC=MICPC+1
        <MOVE!LDMAR!IMM!<IMG10&377>>
        SP      MEMX!INCMAR,AORB,SP10 ;RESTORE BIT 1 OF SP10
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!INCMAR!AORB!SP10>
        SP      MEMX!INCMAR,SELB,SP11 ;RESTORE SP11
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!INCMAR!SELB!SP11>
        SP      MEMX!INCMAR,SELB,SP15 ;RESTORE SP15
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!INCMAR!SELB!SP15>
        SP      MEMX,SELB,SP17       ;RESTORE      SP17
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!SELB!SP17>
        SP      MEMX,SELB,SP12       ;RESTORE      SP12
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!SELB!SP12>
        SP      BR DECA,SP10         ;TURN OFF UNNUM MESSAGE PENDING AND
        MICPC=MICPC+1
        <MOVE!SPX!BR!DECA!SP10>
        ;ZERO THE BRG
        ;CLEAR INIT MODE

        SP      BR DECA,SP1
        MICPC=MICPC+1
        <MOVE!SPX!BR!DECA!SP1>
        BRWATE  IMM,206
        MICPC=MICPC+1
        <MOVE!WTEBR!IMM!<206>>
        ALWAYS  NAK1
        MICPC=MICPC+1
        <JUMP!ALCOND!<NAK1-INIT&3000*4>!<NAK1-INIT&777/2>>

ININT:  PSTATE  INWAT2
        MEM      IMM,<<INWAT2-INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<<INWAT2-INIT&777/2>>>
        BRWATE  IMM,15               ;MASK FOR TURN OFF ALL BUT EXT MEM BITS + NXM
        MICPC=MICPC+1
        <MOVE!WTEBR!IMM!<15>>
        SP      IBUS,UBBR,SPO       ;READ BR CONTROL REG
        MICPC=MICPC+1
    
```

DDCGAM.MAC 14-DEC-77 15:07

BASSRV---- BASE SERVICE ROUTINE

(1)	016734	123220
707	016736	
(1)		000206
(1)	016736	063260
708	016740	
(1)		000207
(1)	016740	000600
709	016742	
(1)		000210
(1)	016742	104457

```

<MOVE!SPX!IBUS!UBBR!SPO
SP BR,AANDB,SPO ;MASK OFF VECTOR TO X04
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SPO>
BRWRT IMM,200 ;MASK FOR INTERRUPT
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<200>>
ALWAYS RB4
MICPC=MICPC+1
<JUMP!ALCOND!<RB4-INIT&3000*4>! RB4-INIT&777 2.>

```

DCCGAM.MAC 14-DEC-77 15:07

OUTINT---SET UP OUTPUT INTERRUPT (RDY0)

```

711
712 016744
714 016744
(1) 016744
(2) 000211
(2) 016744 00262E
719 016746
(1) 000212
(1) 016746 010240
721 016750
(1) 000213
(1) 016750 050220
722 016752
(1) 000214
(1) 016752 123040
723 016754
(1) 000215
(1) 016754 055302
724 016756
(1) 000216
(1) 016756 050220
725 016760
(1) 000217
(1) 016760 074520
726
727
728 016762
(1) 000220
(1) 016762 055224
729 016764
(1) 000221
(1) 016764 055225
730 016766
(1) 000222
(1) 016766 055227
731 016770
(1) 000223
(1) 016770 055226
732
733 016772
(1) 000224
(1) 016772 103751
734
736 016774
(1) 000225
(1) 016774 100447
737 016776
(1) 016776
(1) 016776
(2) 000226
(2) 016776 002647
742 017000
(1) 000227
(1) 017000 010240
743 017002
(1) 000230

```

```

.SBTTL OUTINT---SET UP OUTPUT INTERRUPT (RDY0)
OUTINT: PSTATE PINT2
MEM IMM <<PINT2-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<PINT2-INIT&777/2>>>
;COMPLETION
LDMA IMM,NXTINT ;ADDRESS OF NEXT INTERRUPT POINTER
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<NXTINT&377>>
LDMA MEMX,SELB ;NEXT INTERRUPT
MICPC=MICPC+1
<MOVE!LDMAR!MEMX!<SELB>>
SP IBUS,OCON,SPO ;READ THE OUTPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPX!IBUS!OCON!SPO>
OUT <MEMX!INCMAR>,<AORB!OCON> ;WRITE THE OUT CONTROL CSR
MICPC=MICPC+1
<MOVE!WROUTX!MEMX!INCMAR!<AORB!OCON>>
LDMA MEMX,SELB ;ADDRESS LINK
MICPC=MICPC+1
<MOVE!LDMAR!MEMX!<SELB>>
BRWRT <BR!INCMAR>,<AA!SPO> ;KICK PAST LINK STATUS BYTE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!INCMAR!<AA!SPO>>
;SHIFT CSRO IMAGE LEFT
;***DO NOT CHANGE BR UNTIL BR7***
OUT <MEMX!INCMAR>,<SELB!OPORT1> ;WRITE LOW BYTE OF BA TO CSR
MICPC=MICPC+1
<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT1>>
OUT <MEMX!INCMAR>,<SELB!OPORT2> ;WRITE HIGH BYTE OF BA TO CSR
MICPC=MICPC+1
<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT2>>
OUT <MEMX!INCMAR>,<SELB!OPORT4> ;WRITE HIGH BYTE OF COUNT TO CSR
MICPC=MICPC+1
<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT4>>
OUT <MEMX!INCMAR>,<SELB!OPORT3> ;WRITE THE LOW BYTE OF COUNT
MICPC=MICPC+1
<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT3>>
;***HERE IS BR7***
BR7 PE1 ;INTERRUPT ENABLE IS SET
MICPC=MICPC+1
<JUMP!BR7CON!<PE1-INIT&3000*4>!<PE1-INIT&777/2>>
;GENERATE AN INTERRUPT
ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
PINT2: PSTATE OUTWAIT
MEM IMM <<OUTWAIT-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<OUTWAIT-INIT&777/2>>>
LDMA IMM,NXTINT ;ADDRESS NEXT INTERRUPT QUEUE
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<NXTINT&377>>
SP MEMX,SELB,SPO ;COPY ADDRESS FOR NEXT INT TO SPO
MICPC=MICPC+1

```

DCCGAH.MAC 14-DEC-77 15:07

OUTINT---SET UP OUTPUT INTERRUPT (RDY0)

```

(1) 017002 043220 <MOVE!SPX!MEMX!SELB!SPO>
744 017004 000231 MEM IMM,INTSTK ;ASSUME WRAP AROUND CASE
(1) 017004 002642 MICPC=MICPC+1
745 017006 000232 <MOVE!WRMEM!IMM!<INTSTK>>
(1) 017006 000776 BRWRT IMM,<<MMEND-2>> ;ADDRESS OF LAST INT IN STACK
746 017010 000232 MICPC=MICPC+1
(1) 017010 060360 <MOVE!WRTEBR!IMM!<<MMEND-2>>>
747 017012 000233 CMP BR,SPO ;SHOULD WE WRAP
(1) 017012 060360 MICPC=MICPC+1
748 017014 000234 <SUBTC!BR!SPO>
(1) 017014 101637 Z SS ;YES--BRANCH
749 017016 000235 MICPC=MICPC+1
(1) 017016 062400 <JUMP!ZCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
750 017020 000236 BRWRT IMM,2 ;OFFSET FOR NEXT POINTER
(1) 017020 043220 MICPC=MICPC+1
751 017022 000235 <MOVE!WRTEBR!IMM!<2>>
(1) 017022 010241 MEM BR,ADD!SPO ;UPDATE POINTER
752 017024 000236 MICPC=MICPC+1
(1) 017024 040360 <MOVE!WRMEM!BR!<ADD!SPO>>
753 017026 000237 SS: SP MEMX,SELB,SPO ;COPY POINTER TO SPO
(1) 017026 101644 MICPC=MICPC+1
754 017030 000240 <MOVE!SPX!MEMX!SELB!SPO>
(1) 017030 100447 LDMA IMM,NXTSP ;PICK UP START OF IN QUEUE
755 017032 000240 MICPC=MICPC+1
(1) 017032 000757 <MOVE!LDMA!IMM!<NXTSP&377>>
756 017034 000241 CMP MEMX,SPO ;COMPARE TO END
(1) 017034 063261 MICPC=MICPC+1
757 017036 000242 <SUBTC!MEMX!SPO>
(1) 017036 100447 Z 10$ ;IF EQUAL--CLEAR INT PENDING
758 017038 000242 MICPC=MICPC+1
(1) 017038 101644 <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
759 017040 000243 ALWAYS IDLE
(1) 017040 100447 MICPC=MICPC+1
760 017042 000244 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
761 017044 000757 10$: BRWRT IMM,357 ;MASK TO CLEAR INT PENDING
(1) 017044 000757 MICPC=MICPC+1
762 017046 000244 <MOVE!WRTEBR!IMM!<357>>
(1) 017046 000757 CLRIDL: SP BR,AANDB,SP1
763 017048 000245 MICPC=MICPC+1
(1) 017048 063261 <MOVE!SPX!BR!AANDB!SP1>
764 017050 000246 ALWAYS IDLE
(1) 017050 100447 MICPC=MICPC+1
765 017052 000246 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1) 017052 100447

```

DDCGAM MAC 14-DEC-77 15:07

OUTWAI--WAIT FOR RDYO TO GO AWAY

759
760 017040 00024
(1) 017040 123440
765 017042 000250
(1) 017042 103447
767 017044 000251
(1) 017044 000500
768 017046 000252
(1) 017046 061262
769 017050 000253
(1) 017050 100666
770
771 017052 000254
(1) 017052 123560
772 017054 000255
(1) 017054 001620
773 017056 000256
(1) 017056 102676
774 017060 000257
(1) 017060 002113
775 017062 000260
(1) 017062 060600
776 017064 000261
(1) 017064 116365
777 017066 000262
(1) 017066 123000
778 017070 000263
(1) 017070 000500
779 017072 000264
(1) 017072 061260
780 017074 000265
(1) 017074 010211
781 017076 000266
(1) 017076 002472
782 017100 000267
(1) 017100 100447
783
784 017102

```
.SBTTL OUTWAI--WAIT FOR RDYO TO GO AWAY
OUTWAI: SPBR IBUS, OCON, SPC ;READ OUTPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!OCON!SPO>
BR7 IDLE
MICPC=MICPC+1
<JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
BRWRT IMM, 100 ;CLEAR CONTROL BITS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<100>>
OUT BR, OCON!AANDB
MICPC=MICPC+1
<MOVE!WROUTX!BR!<OCON!AANDB>>
ALWAYS INS13
MICPC=MICPC+1
<JUMP!ALCOND!<INS13-INIT&3000*4>!<INS13-INIT&777/2>>
.SBTTL CTLSRV--CNTL I SERVICE
CTLSRV: SPBR IBUS, PORT4, SPO ;TO SPO
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!PORT4!SPO>
BRSHFT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR1 HOSEL ;IF SET IS HALF DUPLEX
MICPC=MICPC+1
<JUMP!BR1CON!<HOSEL-INIT&3000*4>!<HOSEL-INIT&777/2>>
OUTPUT IMM, <100!OMODEM> ;MASK DTR, TURN OFF HDX
MICPC=MICPC+1
<MOVE!WROUT!IMM!<100!OMODEM>>
INS11: BRWRT DP, <SELA!SPO> ;RESTORE THE CNTL WORD
MICPC=MICPC+1
<MOVE!WRTEBR!DP!<SELA!SPO>>
BR0 CBOOT ;IF SET IS BOOT
MICPC=MICPC+1
<JUMP!BR0CON!<CBOOT-INIT&3000*4>!<CBOOT-INIT&777/2>>
INS12: SP IBUS, INCON, SPO ;READ THE INPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPX!IBUS!INCON!SPO>
BRWRT IMM, 100 ;ZERO THE BR REGISTER EXCEPT INT ENABLE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<100>>
OUT BR, <AANDB!OINCON> ;CLEAR IN CONTROL CSR
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AANDB!OINCON>>
LDMA IMM, PRST ;ADDRESS PORT STATE
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<PRST&377>>
INS13: PSTATE NIDLE2
MEM IMM, <<NIDLE2-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>
ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
TABMXT: SP IBUS, UBBR, SPO ;READ BR CONTROL
```


DCCGM.MAC 14-DEC-77 15:07

CTLSRV--CNTL I SERVICE

```

1          000270          MICPC=MICPC+1
(1) 017102 123220      <MOVE!SPX!IBUS!UBBR!SPO
785 017104          BRWRT IMM,115          :MASK TO CLEAR BP REQ
(1)          000271          MICPC=MICPC+1
(1) 017104 000515      <MOVE!WRTEBR!IMM!<115>>
786 017106          SP BP,AANDB,SPO
(1)          000272          MICPC=MICPC+1
(1) 017106 063260      <MOVE!SPX!BR!AANDB!SPO>
787 017110          BRWRT IMM,4          :INCREMENT MXT
(1)          000273          MICPC=MICPC+1
(1) 017110 000404      <MOVE!WRTEBR!IMM!<4>>
788 017112          OUT BR,ADU,OBR
(1)          000274          MICPC=MICPC+1
(1) 017112 061011      <MOVE!WROUTX!BR!<ADD!OBR>>
789 017114          ALWAYS ECX
(1)          000275          MICPC=MICPC+1
(1) 017114 114743      <JUMP!ALCOND!<ECX-INIT&3000*4>!<ECX-INIT&777/2>>
790 017116          HDSEL: BRWRT IMM,100          ;HD MASK TO BR
(1)          000276          MICPC=MICPC+1
(1) 017116 000500      <MOVE!WRTEBR!IMM!<100>>
791 017120          SP BR,AORB,SP10          ;UPDATE PORT STATUS WORD
(1)          000277          MICPC=MICPC+1
(1) 017120 063310      <MOVE!SPX!BR!AORB!SP10>
792 017122          ALWAYS INS11
(1)          000300          MICPC=MICPC+1
(1) 017122 100660      <JUMP!ALCOND!<INS11-INIT&3000*4>!<INS11-INIT&777/2>>

```

```

794
795 017124 000301
(1) 017124 010070
796 017126 000302
(1) 017126 053220
797 017130 000303
(1) 017130 016401
798 017132 000304
(1) 017132 014543
799
800 017134 000305
(1) 017134 136500
801 017136 000306
(1) 017136 136520
802 017140 000307
(1) 017140 136560
803 017142 000310
(1) 017142 136540
804 017144 000311
(1) 017144 010070
805 017146 000312
(1) 017146 002471
806 017150 000313
(1) 017150 060360
807 017152 000314
(1) 017152 101717
808 017154 000315
(1) 017154 000406
809 017156 000316
(1) 017156 062400
810 017160 000317
(1) 017160 000402
811 017162 000320
(1) 017162 063310
812 017164 000321
(1) 017164 100662
813
814 017166 000322
(1)

```

```

.SBTTL TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE
TBASRV: LDMA IMM,ETC ;GET POINTER TO END OF TMT CHAIN
MICPC=MICPC+1
<MOVE!LDMA!IMM!<ETC&377>>
LDMA MEMX,<SELB!SPX!SPO> ;FIND THE LINK
MICPC=MICPC+1
<MOVE!LDMA!MEMX!<SELB!SPX!SPO>>
MEMINC IMM,1 ;BUFFER ASSIGNED IN IN LINK FLAGS
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<1>>
BRWRT <IMM!INCMAR>,TMLB ;POINT PAST NUMBER FIELD
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!INCMAR!<TMLB>> ;SET BR FOR ADDITION TO SPO
MEMINC IBUS,PORT1
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT1>>
MEMINC IBUS,PORT2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT2>>
MEMINC IBUS,PORT4
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT4>>
MEMINC IBUS,PORT3
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT3>>
LDMA IMM,ETC
MICPC=MICPC+1
<MOVE!LDMA!IMM!<ETC&377>>
MEM IMM,TML1 ;ASSUME QUEUE WRAP AROUND
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<TML1>> ;END OF CHAIN?
CMP BR,SPO
MICPC=MICPC+1
<SUBTC!BR!SPO> ;IF YES--BRANCH
Z 10$
MICPC=MICPC+1
<JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>> ;QUEUE ENTRY LENGTH
BRWRT IMM,6
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<6>> ;UPDATE THE END POINTER IN MEMORY
MEM BR,ADD!SPO
MICPC=MICPC+1
<MOVE!WRMEM!BR!<ADD!SPO>> ;NUMBERED MSG PENDING MASK
10$: BRWRT IMM,2
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>> ;UPDATE LINE STATUS
SP BR,AORB,SP10
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP10>
ALWAYS INS12
MICPC=MICPC+1
<JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
.SBTTL RBASRV--RECEIVE BUFFER ADDRESS SERVICE
RBASRV: LDMA IMM,ERC ;ADDRES END OF RECEIVE CHAIN
MICPC=MICPC+1

```

```

(1) 017166 010023 <MOVE!LDMAR!IMM!<ERC&377>>
815 017170 000323 LDMA MEMX,<SELB!SPX!SPO> ;GET THE POINTER TO LINK
(1) 017170 053220 MICPC=MICPC+1
(1) 017170 053220 <MOVE!LDMAR!MEMX!<SELB!SPX!SPO>>
816 017172 000324 MEMINC IMM,1
(1) 017172 016401 MICPC=MICPC+1
817 017174 000325 <MOVE!WRMEM!INCMAR!IMM!<1>>
(1) 017174 136500 MEMINC IBUS,PORT1
(1) 017174 136500 MICPC=MICPC+1
818 017176 000326 <MOVE!WRMEM!INCMAR!IBUS!<PORT1>>
(1) 017176 136520 MEMINC IBUS,PORT2
(1) 017176 136520 MICPC=MICPC+1
819 017200 000327 <MOVE!WRMEM!INCMAR!IBUS!<PORT2>>
(1) 017200 136560 MEMINC IBUS,PORT4
(1) 017200 136560 MICPC=MICPC+1
820 017202 000330 <MOVE!WRMEM!INCMAR!IBUS!<PORT4>>
(1) 017202 136540 MEMINC IBUS,PORT3
(1) 017202 136540 MICPC=MICPC+1
821 017204 ;...NOTE INVERTED ORDER OF PORT 3 AND PORT4
822 017204 LDMA IMM,ERC
(1) 017204 000331 MICPC=MICPC+1
(1) 017204 010023 <MOVE!LDMAR!IMM!<ERC&377>>
823 017206 000332 MEM IMM,RCL1 ;ASSUME WRAP AROUND CASE
(1) 017206 002424 MICPC=MICPC+1
(1) 017206 002424 <MOVE!WRMEM!IMM!<RCL1>>
824 017210 000333 BRWRTE IMM,RCL7 ;GET ADDRESS OF END OF CAHIN AREA
(1) 017210 000462 MICPC=MICPC+1
(1) 017210 000462 <MOVE!WRTEBR!IMM!<RCL7>>
825 017212 000334 CMP BR,SPO ;CHECK FOR END
(1) 017212 060360 MICPC=MICPC+1
(1) 017212 060360 <SUBTC!BR!SPO>
826 017214 000335 Z INS12 ;IF EQUAL BRANCH
(1) 017214 101662 MICPC=MICPC+1
(1) 017214 101662 <JUMP!ZCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
827 017216 000336 BRWRTE IMM,5 ;CALCULATE ADDRESS OF NEXT LINK
(1) 017216 000405 MICPC=MICPC+1
(1) 017216 000405 <MOVE!WRTEBR!IMM!<5>>
828 017220 000337 MEM BR,ADD!SPO ;...
(1) 017220 062400 MICPC=MICPC+1
(1) 017220 062400 <MOVE!WRMEM!BR!<ADD!SPO>>
829 017222 000340 ALWAYS INS12 ;EXIT
(1) 017222 100662 MICPC=MICPC+1
(1) 017222 100662 <JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
830 017224 000341 BRWRTE IMM,357 ;MASK TO CLEAR CLR ACTIVE
(1) 017224 000757 MICPC=MICPC+1
(1) 017224 000757 <MOVE!WRTEBR!IMM!<357>>
831 017226 000342 SPBR BR,AANDB,SP10 ;CLEAR BIT IN LINE STATUS WORD
(1) 017226 063670 MICPC=MICPC+1
(1) 017226 063670 <MOVE!SPBRX!BR!AANDB!SP10>
832 017230 000343 BRSHFT
(1) 017230 001620 MICPC=MICPC+1
(1) 017230 001620 <MOVE!SHFTBR!WRTEBR!SELB>
833 017232 000344 BR4 FLUSH
(1) 017232 107015 MICPC=MICPC+1
(1) 017232 107015 <JUMP!BR4CON!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>

```

RA1:

834 017234
 (1) 000345
 (1) 017234 000400
 835 017236
 (1) 000346
 (1) 017236 063233
 836 017240
 (1) 000347
 (1) 017240 000427
 837 017242
 (1) 000350
 (1) 017242 104425
 838

```

RA3:  BRWRTE IMM,0           :CLEAR BR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<0>>
      SP BR,SELB,SP13       :SET NUMB MESSAGE TYPE IN SP13
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP13>
      STATE RCVB           :CHANGE RECEIVE STATE POINTER TO STATE B
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVB-INIT&777/2>>
      ALWAYS REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777.2>>
      ;
  
```

DDCGAM.MAC 14-DEC-77 15:07

RBASRV--RECEIVE BUFFER ADDRESS SERVICE

840 017244 000351
 (1) 017244 000700
 841 017246 000352
 (1) 017246 123220
 842 017250 000353
 (1) 017250 061311
 844 017252 000354
 (1) 017252 100447
 849
 850 017254 000355
 (1) 017254 000676
 851
 852 017256 000356
 (1) 017256 063223
 853 017260 000357
 (1) 017260 000402
 854 017262 000360
 (1) 017262 061231
 855 017264 000361
 (1) 017264 120620
 856 017266 000362
 (1) 017266 102761
 857 017270 000363
 (1) 017270 174603
 858
 859 017272 000364
 (1) 017272 120600
 860 017274 000365
 (1) 017274 102047
 861 017276 000366
 (1) 017276 114734
 862
 863
 864 017300 000367
 (1) 017300 000404
 865 017302 000370
 (1) 017302 114637
 866 017304 000371
 (1) 017304 000704

PE1: BRWRT IMM,300 ;MASK FOR INTERRUPT AND VECTOR
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<300>>
 SP IBUS UBBR,SPO ;READ BR CONTROL REG
 MICPC=MICPC+1
 <MOVE!SPX!IBUS!UBBR!SPO>
 OUT BR,AORB,OBR ;INTERRUPT
 MICPC=MICPC+1
 <MOVE!WROUTX!BR!<AORB!OBR>>
 ALWAYS IDLE
 MICPC=MICPC+1
 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
 HALTED: BRADDR EM6
 MICPC=MICPC+1
 <MOVE!WRTEBR!<EM6-INIT&777/2>>
 ACLOW: SP BR,SELB,SP3 ;FALL INTO ACLOW
 MICPC=MICPC+1 ;CHANGE RECEIVE STATE POINTER
 <MOVE!SPX!BR!SELB!SP3>
 BRWRT IMM,2 ;CAUSE AN AC LOW
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<2>>
 OUT BR,SELB,OBR
 MICPC=MICPC+1
 <MOVE!WROUTX!BR!<SELB!OBR>>
 SS: BRWRT IBUS,UBBR ;WAIT FOR IT TO COMPLETE
 MICPC=MICPC+1
 <MOVE!WRTEBR!IBUS!<UBBR>>
 BR1 SS
 MICPC=MICPC+1
 <JUMP!BRCON!<SS-INIT&3000*4>!<SS-INIT&777/2>>
 .ALWAY BR,SELA,SP3,PAGE3
 MICPC=MICPC+1
 <JUMP!ALCOND!BR!SELA!SP3!PAGE3>
 IBU1: BRWRT IBUS,NPR
 MICPC=MICPC+1
 <MOVE!WRTEBR!IBUS!<NPR>>
 BRO IDLE
 MICPC=MICPC+1
 <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
 ALWAYS EC2
 MICPC=MICPC+1
 <JUMP!ALCOND!<EC2-INIT&3000*4>!<EC2-INIT&777/2>>
 OVRUN: BRWRT IMM,4
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<4>>
 ALWAYS NTRSO
 MICPC=MICPC+1
 <JUMP!ALCOND!<NTRSO-INIT&3000*4>!<NTRSO-INIT&777/2>>
 HEH1: STATE TEOM
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<TEOM-INIT&777/2>>

DDCGAH.MAC 14-DEC-77 15:07

RBASRV--RECEIVE BUFFER ADDRESS SERVICE

867 017306
(1) 000372
(1) 017306 110556
868 017310
(1) 000373
(1) 017310 00072:
869 017312
(1) 000374
(1) 017312 063270
870 017314
(1) 000375
(1) 017314 104504
872 017316
(1) 000376
(1) 017316 102047
873 017320
(1) 000377
(1) 017320 104647
875

ACK1:
ALWAYS XEXIT
MICPC=MICPC+1
<JUMP:ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
BRWRT IMM 327
MICPC=MICPC+1
<MOVE:WRTBR!IMM!<327>>
SP BR AANDB,SP10
MICPC=MICPC+1
<MOVE:SPX!BR!AANDB!SP10>
ALWAYS RDS
MICPC=MICPC+1
<JUMP:ALCOND!<RDS-INIT&3000*4>!<RDS-INIT&777/2>>
Rk4:
BRO IDLE
MICPC=MICPC+1
<JUMP:BRCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
ALWAYS RKS
MICPC=MICPC+1
<JUMP:ALCOND!<RKS-INIT&3000*4>!<RKS-INIT&777/2>>

DDCGAH.MAC 14-DEC-77 15:07

RBASRV--RECEIVE BUFFER ADDRESS SERVICE

877 017322
878 000377
879
880
881
882
883 017322 000400
(1) 017322 023200
884 017324 000401
(1) 017324 060601
885 017326 000402
(1) 017326 106012
886 017330 000403
(1) 017330 107412
887 017332 000404
(1) 017332 000601
888 017334 000405
(1) 017334 060360
889 017336 000406
(1) 017336 101741
890 017340 000407
(1) 017340 000405
891 017342 000410
(1) 017342 060360
892 017344 000411
(1) 017344 105424
893 017346 000412
(1) 017346 000620
894 017350 000413
(1) 017350 060360
895 017352 000414
(1) 017352 105771
896 017354 000415
(1) 017354 002212
897
898 017356 000416
(1) 017356 000400
(1) 017360 000417
(1) 017360 063223
899 017362 000420
(1)

```

.=INIT+1000
MICPC=377
.SBTTL RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER
:ENTERED FROM IDLE LOOP
:DETERMINES IF MESSAGE TYPE IS NUMBERED, UNNUMBERED OR BOOT
:SETS UP APPROPRIATE STATES FOR REST OF MESSAGE.
PCVA: SP IBUS RCV DAT, SPO ;READ RECEIVE CHARACTER TO SPO
MICPC=MICPC+1
<MOVE!SPX!IBUS!RCV DAT!SPO>
BRWRT BR SELA!SP1 ;READ PORT STAT'IS WORD
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
BR0 SS ;IF INIT MODE---ONLY BOOT OK
MICPC=MICPC+1
<JUMP!BR0CON!<SS-INIT&3000*4>!<SS-INIT&777/2>>
BR7 SS ;IF BOOT MODE---ONLY BOOT OK
MICPC=MICPC+1
<JUMP!BR7CON!<SS-INIT&3000*4>!<SS-INIT&777/2>>
BRWRT IMM,201 ;SOH TO BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<201>>
CMP BR SPO ;COMPARE BR TO SPO
MICPC=MICPC+1
<SUBTC!BR!SPO>
Z RA1 ;IF EQUAL-IS NUMBERED MESSAGE
MICPC=MICPC+1
<JUMP!ZCOND!<RA1-INIT&3000*4>!<RA1-INIT&777/2>>
BRWRT IMM,5 ;ENQ TO BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<5>>
CMP BR SPO ;COMPARE ENQ TO SPO
MICPC=MICPC+1
<SUBTC!BR!SPO>
Z RA2 ;IF EQUAL-IS UNNUMBERED MESSAGE
MICPC=MICPC+1
<JUMP!ZCOND!<RA2-INIT&3000*4>!<RA2-INIT&777/2>>
SS: BRWRT IMM,220 ;DLE TO BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<220>>
CMP BR SPO ;COMPARE DLE TO SPO
MICPC=MICPC+1
<SUBTC!BR!SPO>
Z BOOT ;IF EQUAL IS BOOT
MICPC=MICPC+1
<JUMP!ZCOND!<BOOT-INIT&3000*4>!<BOOT-INIT&777/2>>
FLUSH: OUTPUT IMM,<200!ORCVCO> ;FLUSH INPUT SILO
MICPC=MICPC+1
<MOVE!WROUT!IMM!<200!ORCVCO>>
; (LOW ORDER BITS READ ONLY)
RSTATE RCVA ;SET STATE TO RCVA
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVA-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>
BRWRT IMM,357 ;MASK TO CLEAR--CLEAR ACTIVE
MICPC=MICPC+1

```

DDCGAH.MAC 14-DEC-77 15:07

RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER

1	017362	000757	<MOVE!WRTEBR!IMM!<357>>	
900	017364		SP BR, AANDB, SP10	: IN LINE STATUS WORD
(1)		000421	MICPC=MICPC+1	
(1)	017364	063270	<MOVE!SPX!BR!AANDB!SP10>	
901	017366		FLUSHA: NOP BR, INCA, SP10	: CLEAR "C" BIT FOR TIMEO
(1)		000422	MICPC=MICPC+1	
(1)	017366	060070	<BR!INCA!SP10>	
902	017370		ALWAYS TIMEO	: SEE IF WE CAN SEND AN ACK
(1)		000423	MICPC=MICPC+1	
(1)	017370	110651	<JUMP!ALCOND!<TIMEO-INIT&3000*4>!<TIMEO-INIT&777/2>>	
903	017372		RA2: STATE RCVI	: CHANGE RECEIVE STATE TO I
(1)		000424	MICPC=MICPC+1	
(1)	017372	000703	<MOVE!WRTEBR!IMM!<RCVI-INIT&777/2>>	
904	017374		REXIT: SP BR, SELB, SP3	
(1)		000425	MICPC=MICPC+1	
(1)	017374	063223	<MOVE!SPX!BR!SELB!SP3>	
905	017376		ALWAYS IDLE	
(1)		000426	MICPC=MICPC+1	
(1)	017376	100447	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	

DDCGAM.MAC 14-DEC-77 15:07

RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD

907
908
909
910
911 017400
912 017400
(1) 000427
(1) 017400 023204
913 017402
(1) 000430
(1) 017402 070215
914 017404
(1) 000431
(1) 017404 054620
915 017406
(1) 000432
(1) 017406 106042
916 017410
(1) 000433
(1) 017410 060601
917 017412
(1) 000434
(1) 017412 107440
918 017414
(1) 000435
(1) 017414 000710
919 017416
(1) 000436
(1) 017416 010012
920 017420
(1) 000437
(1) 017420 104560
921 017422
(1) 000440
(1) 017422 000402
922 017424
(1) 000441
(1) 017424 063301
923 017426
(1) 000442
(1) 017426 000462
924 017430
(1) 000443
(1) 017430 063223
925 017432
(1) 000444
(1) 017432 056226
926 017434
(1) 000445
(1) 017434 056227
927 017436
(1) 000446
(1) 017436 123220
928 017440
(1) 000447
(1) 017440 000501

```

.SBTTL RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD
:ENTERED FROM IDLE LOOP
:STORES COUNT FIELD AND SETS UP RCVC AS NEXT STATE
:
RCVB:
SP      IBUS,RCVDAT,SP4      ;READ CHARACTER TO SP4
MICPC=MICPC+1
<MOVE!SPX!IBUS!RCVDAT!SP4>
LDMA    BR,<SELA!SP15>      ;LOAD MAR WITH ADDRESS OF CURRENT BA
MICPC=MICPC+1
<MOVE!LDMA!BR!<SELA!SP15>>
BRWRT  MEMX!INCMAR,SELB    ;READ THE FLAGS BYTE
MICPC=MICPC+1
<MOVE!WRTEBR!MEMX!INCMAR!<SELB>>
BR0     RB1                ;BUFFER IS ASSIGNED
MICPC=MICPC+1
<JUMP!BR0CON!<RB1-INIT&3000*4>!<RB1-INIT&777/2>>
BRWRT  BR,SELA!SP1        ;NO BUFFER ASSIGNED - MAINT MODE?
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
BR7     RB3                ;YES
MICPC=MICPC+1
<JUMP!BR7CON!<RB3-INIT&3000*4>!<RB3-INIT&777/2>>
BRWRT  IMM,310            ;SET NAK REASON CODE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<310>>
LDMA    IMM,NTLS          ;ADDRESS CUMULATIVE COUNTER
MICPC=MICPC+1
<MOVE!LDMA!IMM!<NTLS&377>>
ALWAYS  RHS              ;SEND A NAK
MICPC=MICPC+1
<JUMP!ALCOND!<RHS-INIT&3000*4>!<RHS-INIT&777/2>>
RB3:   BRWRT  IMM,2        ;FLAG FOR NO BUF ASSIGNED IN MAINT MODE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>>
SP      BR,AORB,SP1      ;SET THE FLAG
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP1>
RB1:   STATE  RCVC
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVC-INIT&777/2>>
RBO:   SP      BR,SELB,SP3
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>
OUTPUT <MEMX!INCMAR>,<SELB!OBA1> ;OUTPUT LOW ORDER BYTE OF ADDRESS
MICPC=MICPC+1
<MOVE!WROUT!MEMX!INCMAR!<SELB!OBA1>>
OUTPUT MEMX!INCMAR,<SELB!OBA2> ;OUTPUT HIGH BYTE OF ADDRESS
MICPC=MICPC+1
<MOVE!WROUT!MEMX!INCMAR!<SELB!OBA2>>
SP      IBUS,UBBR,SPO    ;READ THE BUS REQ REGISTER
MICPC=MICPC+1
<MOVE!SPX!IBUS!UBBR!SPO>
BRWRT  IMM,101          ;MASK OFF ALL BUT NXM AND VEC4 BITS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<101>

```

```

929 017442          SP      BR, AANDB, SPO          ;AND SAVE IN SPO
(1) 017442 000450  MICPC=MICPC+1
(1) 017442 063260  <MOVE!SPX!BR!AANDB!SPO>
930 017444          SP      IMM, 300, SP5         ;MASK TO ISOLATE EX. MEM BITS
(1) 017444 000451  MICPC=MICPC+1
(1) 017444 003305  <MOVE!SPX!IMM!300!SP5>
931 017446          BRWRT  MEMX, AANDB!SP5        ;NOTE THIS REALLY WRITES A 305 BUT THE
932 017446 000452  MICPC=MICPC+1                 ;5 GETS SHIFTED OUT
933 017446 040665  <MOVE!WRTEBR!MEMX!<AANDB!SP5>    ;MASK AL! BUT EX. MEM BITS
934 017450          BRSHFT  MICPC=MICPC+1         ;SHIFT THEM INTO THE CORRECT POSITION
(1) 017450 000453  <MOVE!SHFTBR!WRTEBR!SELB>
935 017452          BRSHFT  MICPC=MICPC+1
(1) 017452 000454  <MOVE!SHFTBR!WRTEBR!SELB>
936 017454          BRSHFT  MICPC=MICPC+1
(1) 017454 000455  <MOVE!SHFTBR!WRTEBR!SELB>
937 017456          BRSHFT  MICPC=MICPC+1
(1) 017456 000456  <MOVE!SHFTBR!WRTEBR!SELB>
938 017460          RB4:   OUT      BR, AORB!OBR          ;WRITE EX MEM BITS OUT
(1) 017460 000457  MICPC=MICPC+1
(1) 017460 061311  <MOVE!WROUTX!BR!<AORB!OBR>>
939 017462          ALWAYS  IDLE
(1) 017462 000460  MICPC=MICPC+1
(1) 017462 100447  <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
940 017464          RB2:   ALWAYS  I2
(1) 017464 000461  MICPC=MICPC+1
(1) 017464 100453  <JUMP!ALCOND!<I2-INIT&3000*4>!<I2-INIT&777/2>>
941 017466          .SBTTL  RCVB--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD. SELECT AND FINA
942 017466 000462  ;ENTERED FROM IDLE LOOP
943 017466 023205  ;INTERPRETS SELECT AND FINAL
944 017470          ;CHECKS FOR COUNT TOO LARGE
945 017470 000463  ;
946 017470 010167  ;
951 017472          RCVC:  SP      IBUS, RCVDAT, SP5         ;GET CHARACTER
(1) 017472 000464  MICPC=MICPC+1
(1) 017472 023205  <MOVE!SPX!IBUS!RCVDAT!SP5>
953 017474          LDMA   IMM, BC                    ;LOAD MAR TO BYTE COUNT
(1) 017474 000463  MICPC=MICPC+1
(1) 017474 010167  <MOVE!LDMAR!IMM!<BC&377>>
954 017476          MEMINC  BR, SELA!SP4             ;SAVE LOW BYTE
(1) 017476 000464  MICPC=MICPC+1
(1) 017476 076604  <MOVE!WRMEM!INCMAR!BR!<SELA!SP4>>
955 017478          MEMINC  BR, SELA!SP5             ;AND NOW HIGH BYTE
(1) 017478 000465  MICPC=MICPC+1
(1) 017478 076605  <MOVE!WRMEM!INCMAR!BR!<SELA!SP5>>
956 017480          STATE  RCVD                      ;SET NEXT STATE TO D
(1) 017480 000466  MICPC=MICPC+1
(1) 017480 000470  <MOVE!WRTEBR!IMM!<RCVD-INIT&777/2>>
957 017500          ALWAYS  REXIT
(1) 017500 000467  MICPC=MICPC+1
(1) 017500 104425  <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>

```

DDCGAM.MAC 14-DEC-77 15:07

RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES

```

959
960
961 017502 000470
(1) 017502 000522
962 017504 000471
(1) 017504 063223
963 017506 000472
(1) 017506 023600
964 017510 000473
(1) 017510 060757
965 017512 000474
(1) 017512 060641
966 017514 000475
(1) 017514 107514
967 017516 000476
(1) 017516 060610
968 017520 000477
(1) 017520 C01620
969 017522 000500
(1) 017522 103047
970 017524 000501
(1) 017524 106110
971 017526 000502
(1) 017526 010153
972 017530 000503
(1) 017530 062600
973 017532 000504
(1) 017532 010003
974 017534 000505
(1) 017534 002401
975 017536 000506
(1) 017536 003374
976 017540 000507
(1) 017540 100447
977 017542 000510
(1) 017542 060600
978 017544 000511
(1) 017544 060772
    
```

```

.SBTTL RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES
RCVD: STATE RCVE
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<RCVE-INIT&777/2>>
RD2:  SP BR SELB,SP3 ;SAVE THE STATE
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP3>
      SPBR IBUS,RCVDAT,SPO ;INPUT THE CHARACTER
      MICPC=MICPC+1
      <MOVE!SPBRX!IBUS!RCVDAT!SPO>
      BRWRT BR,SUB!SP17 ;COMPARE NEW R TO LAST R
      MICPC=MICPC+1
      <MOVE!WRITEBR!BR!<SUB!SP17>>
      BRWRN BR,AORNB!SP1 ;IF NEW IS THE SAME OR LESS OR
      MICPC=MICPC+1
      <MOVE!WRITEBR!BR!<AORNB!SP1>>
      BR7 RD7 ;MAINT MODE - GET OUT
      MICPC=MICPC+1
      <JUMP!BR7CON!<RD7-INIT&3000*4>!<RD7-INIT&777/2>>
      BRWRT BR,SELA!SP10
      MICPC=MICPC+1
      <MOVE!WRITEBR!BR!<SELA!SP10>>
      BRSHFT
      MICPC=MICPC+1
      <MOVE!SHFTBR!WRITEBR!SELB>
      BR4 IDLE
      MICPC=MICPC+1
      <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      BRO RD4 ;MESSAGE IN PROGRESS
      MICPC=MICPC+1
      <JUMP!BROCON!<RD4-INIT&3000*4>!<RD4-INIT&777/2>>
RD3: LDMA IMM,ISP17 ;ADDRESS LAST ACKED IMAGE
      MICPC=MICPC+1
      <MOVE!LDMA!IMM!<ISP17&377>>
      MEM BR,SELA!SPO ;COPY THE CHAR
      MICPC=MICPC+1
      <MOVE!WRMEM!BR!<SELA!SPO>>
RD5: LDMA IMM,REPST ;POINT TO REP THRESHOLD COUNTER
      MICPC=MICPC+1
      <MOVE!LDMA!IMM!<REPST&377>>
      MEM IMM,1 ;RESET REP THRESHOLD
      MICPC=MICPC+1
      <MOVE!WRMEM!IMM!<1>>
RD6: SP IMM,374,SP14 ;RESET THE COUNT
      MICPC=MICPC+1
      <MOVE!SPX!IMM!374!SP14>
      ALWAYS IDLE
      MICPC=MICPC+1
      <JUMP!ALCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
RD4: BRWRT BR,SELA!SPO
      MICPC=MICPC+1
      <MOVE!WRITEBR!BR!<SELA!SPO>>
      BRWRT BR,SUBTC!SP12
      MICPC=MICPC+1
      <MOVE!WRITEBR!BR!<SUBTC!SP12>
    
```

DDCGAM.MAC 14-DEC-77 15:07

RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES

```

979 017546 000512 BR7 IDLE
(1) 017546 103447 MICPC=MICPC+1
980 017550 000513 <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1) 017550 104502 ALWAYS RD3
981 017552 000514 MICPC=MICPC+1
(1) 017552 060600 <JUMP!ALCOND!<RD3-INIT&3000*4>!<RD3-INIT&777/2>>
982 017554 000515 RC7: BRWRT BR SELA!SPO
(1) 017554 060372 MICPC=MICPC+1
983 017556 000516 <MOVE!WRTEBR!BR!<SELA!SPO>>
(1) 017556 105506 CMP BR SP12
984 017560 000517 MICPC=MICPC+1
(1) 017560 060530 <SUBTC!BR!SP12>
985 017562 000520 Z RD6
(1) 017562 113671 MICPC=MICPC+1
986 017564 000521 <JUMP!ZCOND!<RD6-INIT&3000*4>!<RD6-INIT&777/2>>
(1) 017564 100447 BRWRT BR AA!SP10 ;IF HDX
987 017566 000522 BR7 TIME2 ;QUEUE UP REP
988 017566 060601 MICPC=MICPC+1
(1) 017566 060530 <JUMP!BR7CON!<TIME2-INIT&3000*4>!<TIME2-INIT&777/2>>
989 017570 000523 ALWAYS IDLE
(1) 017570 107721 MICPC=MICPC+1
990 017572 000524 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1) 017572 020600 .SBTTL RCVE--ROUTINE TO HANDLE N FIELD OF NUMBERED MESSAGE
991 017574 000525 RCVE: BRWRT BR SELA!SP1 ;READ THE STATUS BYTE
(1) 017574 060371 MICPC=MICPC+1
992 017576 000526 <MOVE!WRTEBR!BR!<SELA!SP1>>
(1) 017576 105531 BR7 RCVQ
993 017600 000527 MICPC=MICPC+1
(1) 017600 063173 <JUMP!BR7CON!<RCVQ-INIT&3000*4>!<RCVQ-INIT&777/2>>
994 017602 000530 BRWRT IBUS RCVDAT ;INPUT THE CHARACTER
(1) 017602 104532 MICPC=MICPC+1
995 017604 000531 <MOVE!WRTEBR!IBUS!<RCVDAT>>
(1) 017604 063071 CMP BR SP11
996 017606 000532 MICPC=MICPC+1
(1) 017606 000533 <SUBTC!BR!SP11>
997 017610 000533 Z SS
(1) 017610 104425 MICPC=MICPC+1
998 017610 104425 <JUMP!ZCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
999 017610 104425 SP BR DECA,SP13 ;FORCE MSG TYPE TO -1
(1) 017610 000534 MICPC=MICPC+1
(1) 017610 000534 <MOVE!SPX!BR!DECA!SP13>
(1) 017610 000534 ALWAYS RE2
(1) 017610 000534 MICPC=MICPC+1
(1) 017610 000534 <JUMP!ALCOND!<RE2-INIT&3000*4>!<RE2-INIT&777/2>>
(1) 017610 000534 SP BR INCA,SP11 ;UPDATE R FIELD
(1) 017610 000534 STATE RCVF ;NEXT RECEIVE STATE IS F
(1) 017610 000534 MICPC=MICPC+1
(1) 017610 000534 <MOVE!WRTEBR!IMM!<RCVF-INIT&777/2>>
(1) 017610 000534 ALWAYS REXIT
(1) 017610 000534 MICPC=MICPC+1
(1) 017610 104425 <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>

```

DDCGAM.MAC 14-DEC-77 15:07

RCVF--ROUTINE TO IGNORE ADDRESS

```

999
1000 017612 000534
      (1) 017612 063164
1001 017614 000535
      (1) 017614 105137
1002 017616 000536
      (1) 017616 063165
1003 017620 000537
      (1) 017620 000542
1004 017622 000540
      (1) 017622 020200
1005 017624 000541
      (1) 017624 104425
1006
1007
1008
1009 017626 000542
      (1) 017626 000544
1010 017630 000543
      (1) 017630 104540
    
```

```

.SBTTL RCVF--ROUTINE TO IGNORE ADDRESS
RCVF:  SP BR,DECA,SP4 ;DECREMENTLOW BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP4>
      C RCVFO ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<RCVFO-INIT&3000*4>!<RCVFO-INIT&777/2>>
      SP BR,DECA,SP5 ;OVERFLOW - DECREMENT HIGH BYTE
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP5>
RCVFO: STATE RCVG
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVG-INIT&777/2>>
RCVF1: NOP IBUS,RCVDAT,0 ;INPUT CHARACTER - AND DISCARD
      MICPC=MICPC+1
      <IBUS!RCVDAT!0>
      ALWAYS REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
.SBTTL RCVG--ROUTINE TO IGNORE CRC1
RCVG: STATE RCVH ;NEXT STATE IS RCVH
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVH-INIT&777/2>>
      ALWAYS RCVF1
      MICPC=MICPC+1
      <JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>
    
```

```

1012
1013
1014 017632
1015 017632
(1) 000544
(1) 017632 02320C
1016 017634
(1) 000545
(1) 017634 020640
1017 017636
(1) 000546
(1) 017636 116142
1018 017640
(1) 000547
(1) 017640 060601
1019 017642
(1) 000550
(1) 017642 107753
1020 017644
(1) 000551
(1) 017644 060610
1021 017646
(1) 000552
(1) 017646 001620
1022 017650
(1) 000553
(1) 017650 117263
1023 017652
(1) 000554
(1) 017652 010153
1024 017654
(1) 000555
(1) 017654 062617
1025 017656
(1) 000556
(1) 017656 000701
1026 017660
(1) 000557
(1) 017660 010013
1027 017662
(1) 000560
(1) 017662 043220
1028 017664
(1) 000561
(1) 017664 062460
1029 017666
(1) 000562
(1) 017666 010151
1030 017670
(1) 000563
(1) 017670 016402
1031 017672
(1) 000564
(1) 017672 062620
1032 017674
(1) 000565

```

```

.SBTTL RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYP
RCVH:
SP IBUS,RCVCON,SPO ;GET CHAR IN SPO
MICPC=MICPC+1
<MOVE!SPX!IBUS!RCVCON!SPO>
BRWRT IBUS,RCVCON ;READ RECVR CONTROL REGISTER
MICPC=MICPC+1
<MOVE!WRTBR!IBUS!<RCVCON>>
BR0 TDON1 ;IF BCC MATCH SET CRC IS GOOD
MICPC=MICPC+1
<JUMP!BR0CON!<TDON1-INIT&3000*4>!<TDON1-INIT&777/2>>
BRWRT BR,SELA!SP1 ;READ STATUS BYTE
MICPC=MICPC+1
<MOVE!WRTBR!BR!<SELA!SP1>>
BR7 RHX ;MAINT MODE
MICPC=MICPC+1
<JUMP!BR7CON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
BRWRT DP,<SELA!SP10> ;READ PORT STATUS WORD TO BR
MICPC=MICPC+1
<MOVE!WRTBR!DP!<SELA!SP10>>
BRSHFT
MICPC=MICPC+1
<MOVE!SHFT!BR!WRTBR!SELB>
BR4 SNAK1 ;IF START MODE--PROCEED TO RESEND START
MICPC=MICPC+1
<JUMP!BR4CON!<SNAK1-INIT&3000*4>!<SNAK1-INIT&777/2>>
LDMA IMM,ISP17 ;ELSE BCC ERROR--POINT TO IMAGE OF SP17 IN RAM
MICPC=MICPC+1
<MOVE!LDMA!IMM!<ISP17&377>>
MEM BR,SELA!SP17 ;RESTORE LAST ACKED IMAGE
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELA!SP17>>
BRWRT IMM,301 ;WRITE HEADER BCC ERROR SUBTYPE
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<301>>
LDMA IMM,NHDS ;ADDRESS CUM ERROR COUNTER
MICPC=MICPC+1
<MOVE!LDMA!IMM!<NHDS&377>>
SP MEMX,SELB,SPO ;WRITE IT TO SPO
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SPO>
MEM BR,INCA!SPO ;INCREMENT IT
MICPC=MICPC+1
<MOVE!WRMEM!BR!<INCA!SPO>>
LDMA IMM,T ;LOAD ADDRESS OF TYPE FIELD
MICPC=MICPC+1
<MOVE!LDMA!IMM!<T&377>>
MEMINC IMM,2 ;WRITE NAK TYPE
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<2>>
MEM BR,SELB ;WRITE NAK SUBTYPE FIELD SUBTYPE
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELB>>
LDMA IMM,NAKST ;ADDRESS NAKS TMTED DYNAMIC
MICPC=MICPC+1

```

DDCGAH.MAC 14-DEC-77 15:07

RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYPES

(1) 017674 013001
 1033 (1) 017676 000566
 (1) 017676 040620
 1034 (1) 017700 000567
 (1) 017700 061620
 1035 (1) 017702 000570
 (1) 017702 062620
 1036 (1) 017704 000571
 (1) 017704 116232
 1037 (1) 017706 000572
 (1) 017706 114660
 1038 (1) 017710 000573
 (1) 017710 060573
 1039 (1) 017712 000574
 (1) 017712 115441
 1040 (1) 017714 000575
 (1) 017714 000400
 (1) 017716 000576
 (1) 017716 063223
 1041 (1) 017720 000577
 (1) 017720 060610
 1047 (1) 017722 000600
 (1) 017722 002212
 1049 (1) 017724 000601
 (1) 017724 001620
 1050 (1) 017726 000602
 (1) 017726 107210
 1051 (1) 017730 000603
 (1) 017730 010162
 1052 (1) 017732 000604
 (1) 017732 054373
 1053 (1) 017734 000605
 (1) 017734 111770
 1054 (1) 017736 000606
 (1) 017736 054373
 1055 (1) 017740 000607
 (1) 017740 115416
 1056 (1) 017742 000610

```

<MOVE!LDMAR!IMM!<NAKST&377>>
BRWRT MF,1X,SELB ;WRITE IT TO BR
MICPC=MICPC+1
<MOVE!WRTEBR!MEMX!<SELB>>
BSHFT B ;SHIFT IT RIGHT
MICPC=MICPC+1
<MOVE!SHFTBR!SELB!BR>
MEM BR,SELB ;UPDATE IT
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELB>>
BRQ NTHRES ;BRANCH IF THRESHOLD EXCEEDED
MICPC=MICPC+1
<JUMP!BROCON!<NTHRES-INIT&3000*4>!<NTHRES-INIT&777/2>>
ALWAYS SNAK
MICPC=MICPC+1
<JUMP!ALCOND!<SNAK-INIT&3000*4>!<SNAK-INIT&777/2>>
RH3: BRWRT DP<DECA!SP13> ;LOAD TYPE RECEIVED--DECREMENTING
MICPC=MICPC+1
<MOVE!WRTEBR!DP!<DECA!SP13>>
Z RH1 ;IF ALUOUT IS ALL ONES IS NUMBERED MSG
MICPC=MICPC+1
<JUMP!ZCOND!<RH1-INIT&3000*4>!<RH1-INIT&777/2>>
RSTATE RCVA
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVA-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>
BRWRT DP<SELA!SP10> ;LOAD LINE STATUS WORD IN BR
MICPC=MICPC+1
<MOVE!WRTEBR!DP!<SELA!SP10>>
OUTPUT IMM,<200!ORCVCO>
MICPC=MICPC+1
<MOVE!WROUT!IMM!<200!ORCVCO>>
BRSHFT ;SHIFT RIGHT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR4 10$
MICPC=MICPC+1
<JUMP!BR4CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
LDMA IMM,TYPTAB ;ADDRESS TYPE TABLE
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<TYPTAB&377>>
CMP <MEMX!INCMAR>,SP13
MICPC=MICPC+1
<SUBTC!MEMX!INCMAR!SP13>
Z REP
MICPC=MICPC+1
<JUMP!ZCOND!<REP-INIT&3000*4>!<REP-INIT&777/2>>
CMP <MEMX!INCMAR>,SP13
MICPC=MICPC+1
<SUBTC!MEMX!INCMAR!SP13>
Z NAK
MICPC=MICPC+1
<JUMP!ZCOND!<NAK-INIT&3000*4>!<NAK-INIT&777/2>>
10$: LDMA IMM,1'PST ;SET POINTER TO START TYPE
MICPC=MICPC+1

```

K14

DDCGAH.MAC 14-DEC-77 15:07

RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYPES

PAGE: 0179DM
SEQ 0179

(1)	017742	010164	<MOVE!LDMAR!IMM!<TYPSTT&377>>
1057	017744		CMP <MEMX!INCMAR>,SP13
(1)		000611	MICPC=MICPC+1
(1)	017744	054373	<SUBTC!MEMX!INCMAR!SP13>
1058	017746		Z START
(1)		000612	MICPC=MICPC+1
(1)	017746	115501	<JUMP!ZCOND!<START-INIT&3000*4>!<START-INIT&777/2>>
1059			;STACK TYPE
1060	017750		CMP <MEMX!INCMAR>,SP13
(1)		000613	MICPC=MICPC+1
(1)	017750	054373	<SUBTC!MEMX!INCMAR!SP13>
1061	017752		Z STACK
(1)		000614	MICPC=MICPC+1
(1)	017752	111777	<JUMP!ZCOND!<STACK-INIT&3000*4>!<STACK-INIT&777/2>>
1062	017754		CMP <MEMX!INCMAR>,SP13 ;ACK TYPE
(1)		000615	MICPC=MICPC+1
(1)	017754	054373	<SUBTC!MEMX!INCMAR!SP13>
1063	017756		Z ACK
(1)		000616	MICPC=MICPC+1
(1)	017756	115761	<JUMP!ZCOND!<ACK-INIT&3000*4>!<ACK-INIT&777/2>>
1064	017760		ALWAYS IDLE ;OTHERWISE IGNORE--MUST BE OBS MSG
(1)		000617	MICPC=MICPC+1
(1)	017760	100447	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>


```

1072
1073
1074 017762 000620
(1) 017762 123600
1075 017764 000621
(1) 017764 102047
1076 017766 000622
(1) 017766 022203
1077 017770 000623
(1) 017770 000671
1078 017772 000624
(1) 017772 063223
1079 017774 000625
(1) 017774 000621
1080 017776 000626
(1) 017776 104635
1081
1082 020000 000627
(1) 020000 123600
1084 020002 000630
(1) 020002 106227
1089 020004 000631
(1) 020004 000645
1090 020006 000632
(1) 020006 063223
1091 020010 000633
(1) 020010 022203
1092 020012 000634
(1) 020012 000421
1093 020014 000635
(1) 020014 061310
1094 020016 000636
(1) 020016 100447
1095 020020 000637
(1) 020020 000637
(1) 020020 000640
(1) 020022 063223
1099 020024 000641
(1)

```

```

;*****TIME CRITICAL CODE-- CHANGE WITH GREAT CARE*****
.SBTTL RCVK01--ROUTINE TO HANDLE FIRST BYTE ODD RECEIVE
RCVK01: SPBR IBUS,NPR,SPD ;READ NPR REGISTER
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!NPR!SPD>
        BRO IDLE
        MICPC=MICPC+1
        <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        OUTPUT IBUS,RCVDAT!OUTDA2 ;OUTPUT A CHAR
        MICPC=MICPC+1
        <MOVE!WROUT!IBUS!<RCVDAT!OUTDA2>>
        STATE RKE1
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<RKE1-INIT&777/2>>
RCVK02: SP BR,SELB,SP3 ;SET STATE
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP3>
        BRWRT IMM,21
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<21>>
        ALWAYS RK7
        MICPC=MICPC+1
        <JUMP!ALCOND!<RK7-INIT&3000*4>!<RK7-INIT&777/2>>
.SBTTL RCVK0--PROCESS ODD CHARACTER
RCVK0: SPBR IBUS,NPR,SPD ;IS AN NPR GOING
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!NPR!SPD>
        BRO RCVK0 ;IF SO, REITERATE ODD AND EXIT
        MICPC=MICPC+1
        <JUMP!BROCON!<RCVK0-INIT&3000*4>!<RCVK0-INIT&777/2>>
        STATE RCVKE
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<RCVKE-INIT&777/2>>
        SP BR,SELB,SP3 ;SET STATE
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP3>
        OUTPUT IBUS,RCVDAT!OUTDA2 ;OUTPUT A CHAR
        MICPC=MICPC+1
        <MOVE!WROUT!IBUS!<RCVDAT!OUTDA2>>
RK8: BRWRT IMM,21 ;SET OUT NPR (C1) AND NPR REQ
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<21>>
RK7: OUT BR,<AORB!ONPR> ;WRITE NPR REGISTER
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<AORB!ONPR>>
        ALWAYS IDLE
        MICPC=MICPC+1
        <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
RL4: RSTATE RL4
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<RL4-INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP3>
        SPBR IBUS,NPR,SPD
        MICPC=MICPC+1

```

(1)	020024	123600	<MOVE!SPBRX!IBUS!NPR!SPO>	
1101	020026		BRO IDLE	
(1)		000642	MICPC=MICPC+1	
(1)	020026	102047	<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
1103	020030		STATE RCVL	
(1)		000643	MICPC=MICPC+1	
(1)	020030	000723	<MOVE!WRTEBR!IMM!<RCVL-INIT&777/2>>	
1104	020032		ALWAYS RCVK02	
(1)		000644	MICPC=MICPC+1	
(1)	020032	104624	<JUMP!ALCOND!<RCVK02-INIT&3000*4>!<RCVK02-INIT&777/2>>	
1105			.SBTTL RCVKE--HANDLE EVEN BYTES	
1106	020034		RCVKE: BRWRTE IBUS,NPR	;READ NPR CONTROL REGISTER
(1)		000645	MICPC=MICPC+1	
(1)	020034	120600	<MOVE!WRTEBR!IBUS!<NPR>>	
1108	020036		BR4 RK4	;IF RECV NPR--BRANCH
(1)		000646	MICPC=MICPC+1	
(1)	020036	103376	<JUMP!BR4CON!<RK4-INIT&3000*4>!<RK4-INIT&777/2>>	
1113	020040		RK5: SP IBUS,IOBA1,SPO	;READ LOW BYTE OF BA TO SP
(1)		000647	MICPC=MICPC+1	
(1)	020040	023140	<MOVE!SPX!IBUS!IOBA1!SPO>	
1114	020042		OUTPUT DP,<INCA!OBA1>	;WRITE INCREMENTED BA
(1)		000650	MICPC=MICPC+1	
(1)	020042	062066	<MOVE!WROUT!DP!<INCA!OBA1>>	
1115	020044		RK50: SP BR,DECA,SP4	;DECREMENT CHARACTER COUNT
(1)		000651	MICPC=MICPC+1	
(1)	020044	063164	<MOVE!SPX!BR!DECA!SP4>	
1116	020046		C 10\$;NO OVERFLOW
(1)		000652	MICPC=MICPC+1	
(1)	020046	105255	<JUMP!CCOND!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>	
1117	020050		SP BR,DECA,SP5	;OVERFLOW - DECREMENT HIGH BYTE
(1)		000653	MICPC=MICPC+1	
(1)	020050	063165	<MOVE!SPX!BR!DECA!SP5>	
1118	020052		Z RL3	;BYTE COUNT ZERO
(1)		000654	MICPC=MICPC+1	
(1)	020052	105727	<JUMP!ZCOND!<RL3-INIT&3000*4>!<RL3-INIT&777/2>>	
1119	020054		IO\$: OUTPUT IBUS,<RCVDAT!OUTDA1>	;READ CHARACTER AND WRITE IT
(1)		000655	MICPC=MICPC+1	
(1)	020054	022202	<MOVE!WROUT!IBUS!<RCVDAT!OUTDA1>>	
1120	020056		SP IBUS,IOBA1,SPO	;READ INCREMENTED BA
(1)		000656	MICPC=MICPC+1	
(1)	020056	023140	<MOVE!SPX!IBUS!IOBA1!SPO>	
1121	020060		OUTPUT DP,<INCA!OBA1>	;WRITE INCREMENTED BA
(1)		000657	MICPC=MICPC+1	
(1)	020060	062066	<MOVE!WROUT!DP!<INCA!OBA1>>	
1126	020062		C ICBA23	;OVER FLOW
(1)		000660	MICPC=MICPC+1	
(1)	020062	115004	<JUMP!CCOND!<ICBA23-INIT&3000*4>!<ICBA23-INIT&777/2>>	
1127	020064		Rk3: SP BR,DECA,SP4	;DECREMENT THE COUNT OF BYTES
(1)		000661	MICPC=MICPC+1	
(1)	020064	063164	<MOVE!SPX!BR!DECA!SP4>	
1128	020066		C RK6	;NO OVERFLOW
(1)		000662	MICPC=MICPC+1	
(1)	020066	105265	<JUMP!CCOND!<RK6-INIT&3000*4>!<RK6-INIT&777/2>>	
1129	020070		SP BR,DECA,SP5	;DECREMENT HIGH BYTE OF COUNT
(1)		000663	MICPC=MICPC+1	
(1)	020070	063165	<MOVE!SPX!BR!DECA!SP5>	

DDCGAH MAC 14-DEC-77 15:07

RCVKE--HANDLE EVEN BYTES

1130 020072 000664
(1) 020072 105637
1132 020074 000665
(1) 020074 020640
1133 20076 000666
(1) 020076 107227
1134 020100 000667
(1) 020100 000627
1135 020102 000670
(1) 020102 104425
1141
1142 020104 000671
(1) 020104 123200
1144 020106 000672
(1) 020106 102047
1146 020110 000673
(1) 020110 000577
1147 020112 000674
(1) 020112 061270
1148 020114 000675
(1) 020114 104651
1149
1150
1151 020116 000676
(1) 020116 023200
1152 020120 000677
(1) 020120 062202
1153 020122 000700
(1) 020122 060601
1154 020124 000701
(1) 020124 117554
1155 020126 000702
(1) 020126 104661

```

Z          RL4          ;BYTE COUNT ZERO
MICPC=MICPC+1
<JUMP!ZCOND!<RL4-INIT&3000*4>!<RL4-INIT&777/2>>
Rk6: BRWRT IBUS,RCVCON ;READ RECEIVER CONTROL REGISTER
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<RCVCON>>
      BR4 RCVK0          ;IF ANOTHER CHARACTER--PROCESS
      MICPC=MICPC+1
      <JUMP!BR4CON!<RCVK0-INIT&3000*4>!<RCVK0-INIT&777/2>>
Rk66: STATE RCVK0
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVK0-INIT&777/2>>
      ALWAYS REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
RkE1: SP          IBUS,NPR,SPO ;READ NPR REGISTER
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!NPR!SPO>
      BRO IDLE          ;NPR STILL IN PROGRESS
      MICPC=MICPC+1
      <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      BRWRT IMM,177      ;MASK FOR ALL BUT CO
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<177>>
      OUT BR,<AANDB!ONPR> ;TURN OFF ALL BUT CO
      MICPC=MICPC+1
      <MOVE!WROUTX!BR!<AANDB!ONPR>>
      ALWAYS RK50
      MICPC=MICPC+1
      <JUMP!ALCOND!<RK50-INIT&3000*4>!<RK50-INIT&777/2>>
;*****END OF TIME CRITICAL PATH*****
RCVKE0: SP          IBUS,RCVDAT,SPO ;READ CHARACTER AND SAVE IN SPO
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!RCVDAT!SPO>
      OUTPUT BR,<SELA!OUTDA1> ;SEND NONSENSE CHARACTER
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELA!OUTDA1>>
      BRWRT BR,SELA!SP1 ;READ STATUS BYTE
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
      BR7 PASWRD        ;MAINT MODE - SEE IF RLD MESSAGE
      MICPC=MICPC+1
      <JUMP!BR7CON!<PASWRD-INIT&3000*4>!<PASWRD-INIT&777/2>>
      ALWAYS RK3        ;OTHERWISE PROCESS NORMALLY
      MICPC=MICPC+1
      <JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>

```

DOCGAM.MAC 14-DEC-77 15:07

RCVI--STORE UNNUMBERED MESSAGE TYPE

```

1157
1158 020130 000703
(1) 020130 023213
1159 020132 000704
(1) 020132 000706
1160 020134 000705
(1) 020134 104425
1161
1162
1163 020136
1168 020136 000706
(1) 020136 023205
1169 020140 000707
(1) 020140 000600
1170 020142 000710
(1) 020142 060665
1171 020144 000711
(1) 020144 063310
1173 020146 000712
(1) 020146 000714
1174 020150 000713
(1) 020150 104425
1175
1176
1177
1178 020152 000714
(1) 020152 000403
1179 020154 000715
(1) 020154 060353
1180 020156 000716
(1) 020156 000721
1181
1182 020160 000717
(1) 020160 105140
1183 020162 000720
(1) 020162 104471
1184
1185
1186
1187 020164 000721
(1) 020164 000534

```

```

RCVI: .SBTTL RCVI--STORE UNNUMBERED MESSAGE TYPE
      SP      IBUS,RCVDAT,SP13      ;STORE UNNUMBERED TYPE
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!RCVDAT!SP13>
      STATE   RCVJ      ;NEXT STATE IS J
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<RCVJ-INIT&777/2>>
      ALWAYS  REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
      .
RCVJ: .SBTTL RCVJ--ROUTINE TO HANDLE SUBTYPE FIELD,SELECT AND FINAL
      SP      IBUS,RCVDAT,SP5      ;GET CHARACTER
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!RCVDAT!SP5>
      BRWRITE IMM,200      ;CONDITIONALLY SET BIT
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<200>>
      BRWRITE BR,AANDB!SP5
      MICPC=MICPC+1
      <MOVE!WRITEBR!BR!<AANDB!SP5>>
      SP      BR,AORB,SP10
      MICPC=MICPC+1
      <MOVE!SPX!BR!AORB!SP10>
      STATE   RCVR      ;NEXT STATE IS N
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<RCVR-INIT&777/2>>
      ALWAYS  REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
      .SBTTL RCVR--UNNUMBERED MESSAGE RESPONSE FIELD
      ;ENTERED FROM IDLE LOOP
RCVR: BRWRITE IMM,3      ;REF MESSAGE TYPE TO BR
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<3>>
      NOP      BR,SUB,SP13      ;IS TYPE ACK OR NAK
      MICPC=MICPC+1
      <BR!SUB!SP13>
      STATE   RCVQ      ;NEXT STATE IS RCVQ
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<RCVQ-INIT&777/2>>
      ;***NOTE THIS INSTR DOES NOT CLOCK "C"
      ;IF NOT IGNORE
      C      RCVF1
      MICPC=MICPC+1
      <JUMP!CCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>
      ALWAYS  RD2      ;DO RANGE CHECKS
      MICPC=MICPC+1
      <JUMP!ALCOND!<RD2-INIT&3000*4>!<RD2-INIT&777/2>>
      .SBTTL RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD
      ;ENTER FROM IDLE
RCVQ: STATE   RCVF      ;NEXT STATE IS ADDRESS
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<RCVF-INIT&777/2>>

```

C15

DDCGAM.MAC 14-DEC-77 15:07

RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD

PAGE: C184DM
SEQ 0184

1188 020166
01 000722
02 020166 104540

ALWAYS RCVF1
MICPC=MICPC+1
<JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777*2>

JDCGAH.MAC 14-DEC-77 15:07

RCVL--PROCESS CRC3

```

1190 .SBTTL RCVL--PROCESS CRC3
1191 :ENTERED FROM IDLE LOOP
1192 020170 SPBR IBUS,NPP,SPO ;READ NPR CONTROL
(1) (1) 000723 MICPC=MICPC+1
(1) 020170 123600 <MOVE!SPBRX!IBUS!NPR!SPO>
1194 020172 BR4 RL1 ;RCV NPR BRANCH
(1) (1) 000724 MICPC=MICPC+1
(1) 020172 107331 <JUMP!BR4CON!<RL1-INIT&3000*4>!<RL1-INIT&777/2>>
1199 020174 RL2: BRWRT IMM,176 ;MASK TO TURN OFF CO
(1) (1) 000725 MICPC=MICPC+1
(1) 020174 000576 <MOVE!WRTEBR!IMM!<176>>
1200 020176 OUT BR,AANDB!ONPR
(1) (1) 000726 MICPC=MICPC+1
(1) 020176 061270 <MOVE!WROUTX!BR!<AANDB!ONPR>>
1201 .
1202 020200 RL3: STATE RCVM
(1) (1) 000727 MICPC=MICPC+1
(1) 020200 000733 <MOVE!WRTEBR!IMM!<RCVM-INIT&777/2>>
1203 020202 ALWAYS RCVF1
(1) (1) 000730 MICPC=MICPC+1
(1) 020202 104540 <JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>
1204 .
1206 020204 RL1: BRO IDLE
(1) (1) 000731 MICPC=MICPC+1
(1) 020204 102047 <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1207 020206 ALWAYS RL2
(1) (1) 000732 MICPC=MICPC+1
(1) 020206 104725 <JUMP!ALCOND!<RL2-INIT&3000*4>!<RL2-INIT&777/2>>
1209 .SBTTL RCVM--PROCESS CRC4--END OF DATA MESSAGE
1210 :ENTERED FROM IDLE LOOP
1211 :IF CRC CORRECT -- QUEUE INTERRUPT AND UPDATE RESPONSE
1212 .
1213 :IF CRC WRONG SEND NAK
1214 020210 RCVM: BRWRT IBUS,UBBR ;READ UNIBUS BR REGISTER
(1) (1) 000733 MICPC=MICPC+1
(1) 020210 120620 <MOVE!WRTEBR!IBUS!<UBBR>>
1215 020212 BRO NXMERR ;NON-EXISTANT MEMORY
(1) (1) 000734 MICPC=MICPC+1
(1) 020212 106364 <JUMP!BROCON!<NXMERR-INIT&3000*4>!<NXMERR-INIT&777/2>>
1216 020214 SP IBUS,RCVDAT,SPO ;READ CRC CHARACTER
(1) (1) 000735 MICPC=MICPC+1
(1) 020214 023200 <MOVE!SPX!IBUS!RCVDAT!SPO>
1217 020216 BRWRT IBUS,RCVCON ;READ RECEIVER CONTROL REGISTER
(1) (1) 000736 MICPC=MICPC+1
(1) 020216 020640 <MOVE!WRTEBR!IBUS!<RCVCON>>
1218 020220 BRO RCVM1 ;IF CRC GOOD -- PROCESS
(1) (1) 000737 MICPC=MICPC+1
(1) 020220 116171 <JUMP!BROCON!<RCVM1-INIT&3000*4>!<RCVM1-INIT&777/2>>
1219 020222 BRWRT BR,SELA!SP1 ;READ STATUS BYTE
(1) (1) 000740 MICPC=MICPC+1
(1) 020222 060601 <MOVE!WRTEBR!BR!<SELA!SP1>>
1220 020224 BR7 RHX ;CRC ERROR IN BOOT MODE - FLUSH
(1) (1) 000741 MICPC=MICPC+1
(1) 020224 107753 <JUMP!BR7CON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
1221 020226 BRWRT IMM,302 ;DATA ERROR SUBTYPE
(1) (1) 000742 MICPC=MICPC+1

```

DDCGAH.MAC 14-DEC-77 15:07

RCVM--PROCESS CRC4--END OF DATA MESSAGE

```

(1) 020226 000702          <MOVE!WRTEBR!IMM!<302>>
1222 020230          LDMA IMM,NDATS          ;ADDRESS CUM ERROR COUNTER
(1) 020230 000743          MICPC=MICPC+1
(1) 020230 010014          <MOVE!LDMAR!IMM!<NDATS&377>>
1223 020232          ALWAYS RHS          ;SEND NAK
(1) 020232 000744          MICPC=MICPC+1
(1) 020232 104560          <JUMP!ALCOND!<RHS-INIT&3000*4>!<RHS-INIT&777/2>>
1224
1225 020234          RCVMO: BRWRT IMM,10          ;MAINT MESS ERROR
(1) 020234 000745          MICPC=MICPC+1
(1) 020234 000410          <MOVE!WRTEBR!IMM!<10>>
1226 020236          RCVM2: LDMA IMM,<<RTHRS+3>>          ;POINT TO ERROR WORD
(1) 020236 000746          MICPC=MICPC+1
(1) 020236 010177          <MOVE!LDMAR!IMM!<<RTHRS+3>&377>>
1227 020240          ALWAYS RCEXY          ;GIVE FATAL ERROR
(1) 020240 000747          MICPC=MICPC+1
(1) 020240 114472          <JUMP!ALCOND!<RCEXY-INIT&3000*4>!<RCEXY-INIT&777/2>>
1228          .SBTTL EM2--PROCESS RLD MESSAGE
1229          ;ENTERED FROM IDLE LOOP
1230          ;IF RLD PASSWORD CHECKS TRIGGER THE BOOT ROM
1231
1232 020242          EM2: BRWRT IBUS,RCVDAT          ;READ THE CHAR
(1) 020242 000750          MICPC=MICPC+1
(1) 020242 020600          <MOVE!WRTEBR!IBUS!<RCVDAT>>
1233 020244          CMP BR,SP13          ;IS IT A MATCH
(1) 020244 000751          MICPC=MICPC+1
(1) 020244 060373          <SUBTC!BR!SP13>
1234 020246          Z EM3
(1) 020246 000752          MICPC=MICPC+1
(1) 020246 105761          <JUMP!ZCOND!<EM3-INIT&3000*4>!<EM3-INIT&777/2>>
1235
1236 020250          RHX: BRWRT BR,AA!SP1          ;FALL INTO RHX
(1) 020250 000753          MICPC=MICPC+1          ;READ STATUS BYTE SHIFTED LEFT
(1) 020250 060521          <MOVE!WRTEBR!BR!<AA!SP1>>
1237 020252          BR4 10$          ;DLE RECEIVED IN NORMAL MODE
(1) 020252 000754          MICPC=MICPC+1
(1) 020252 107356          <JUMP!BR4CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
1238 020254          ALWAYS FLUSH          ;ALREADY IN MAINT MODE
(1) 020254 000755          MICPC=MICPC+1
(1) 020254 104415          <JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
1239 020256          10$: BRWRT IMM,16$          ;MASK TO CLEAR ALL MAINT RELATED BITS
(1) 020256 000756          MICPC=MICPC+1
(1) 020256 000565          <MOVE!WRTEBR!IMM!<16$>>
1240 020260          SP BR,AANDB,SP1          ;CLEAR THEM
(1) 020260 000757          MICPC=MICPC+1
(1) 020260 063261          <MOVE!SPX!BR!AANDB!SP1>
1241 020262          ALWAYS FLUSH
(1) 020262 000760          MICPC=MICPC+1
(1) 020262 104415          <JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
1242 020264          EM3: SP BR,DECA,SP4
(1) 020264 000761          MICPC=MICPC+1
(1) 020264 063164          <MOVE!SPX!BR!DECA!SP4>
1243 020266          Z EMTRIG
(1) 020266 000762          MICPC=MICPC+1
(1) 020266 115666          <JUMP!ZCOND!<EMTRIG-INIT&3000*4>!<EMTRIG-INIT&777/2>>
1244 020270          ALWAYS IDLE

```

F15

DDCGAH.MAC 14-DEC-77 15:07

EM2--PROCESS RLD MESSAGE

PAGE: 0187DM
SEQ 0187

01) 020270 000763
01) 100447

MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

NXMERR ---NON EXISTANT MEMORY HANDLER

```

1247
1248 020272 000764
(1) 020272 010177
1249 020274 000765
(1) 020274 016401
1250 020276 000766
(1) 020276 002400
1251 020300 000767
(1) 020300 043230
1252 020302 000770
(1) 020302 114474

```

```

.SBTTL NXMERR ---NON EXISTANT MEMORY HANDLER
NXMERR: LDMA IMM, <<RTHRS+3>> :ADDRESS ERROR LINK
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<<RTHRS+3>&377>>
MEMINC IMM, 1
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<1>>
MEM IMM, 0 :NXM ERROR BIT
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<0>>
SP MEMX, SELB, SP10 :CLEAR STATUS
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SP10>
ALWAYS RCEXX
MICPC=MICPC+1
<JUMP!ALCOND!<RCEXX-INIT&3000*4>!<RCEXX-INIT&777/2>>

```

H15

DDG64H.MAC 14-DEC-77 15:07

NXMERR ---NON EXISTANT MEMORY HANDLER

PAGE: 0189DM
SEQ 0189

1271
1273 020304 000771
(1) 020304 060601
1274 020306 000772
(1) 020306 103745
1275 020310 000773
(1) 020310 000610
1276 020312 000774
(1) 020312 063301
1277 020314 000775
(1) 020314 100745
1279 020316 000776
(1) 020316 000402
1280 020320 000777
(1) 020320 114637
1281
1288

```
BOOT:  BRWRT  BR SELA!SP1           ;SEE IF IN MAINT. MODE
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<SELA!SP1>>
        BR7   RA3                   ;BRANCH IF SO AND TREAT DLE LIKE NUM. MSG.
        MICPC=MICPC+1
        <JUMP!BR7CON!<RA3-INIT&3000*4>!<RA3-INIT&777/2>>
        BRWRT  IMM,210              ;MASK TO SET MAINT MODE AND DLE RECV'D
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<210>>
        SP     BR AORB,SP1          ;SET THE BITS
        MICPC=MICPC+1
        <MOVE!SPX!BR!AORB!SP1>
        ALWAYS RA3                  ;TREAT LIKE NUMBERED MESSAGE
        MICPC=MICPC+1
        <JUMP!ALCOND!<RA3-INIT&3000*4>!<RA3-INIT&777/2>>
RTHRES: BRWRT  IMM,2
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<2>>
        ALWAYS ERRXX
        MICPC=MICPC+1
        <JUMP!ALCOND!<ERRXX-INIT&3000*4>!<ERRXX-INIT&777/2>>
```

⋮

DDCGAH MAC 14-DEC-77 15:07

NXMERR ---NON EXISTANT MEMORY HANDLER

```

1290      020322      . =INIT+2000
1291      000777      MICPC=777
1292      .SBTTL TMTA--FIRST CHARACTER OF HEADER
1293      ;
1294      020322      TMTA: BRWRT  BP,AA!SP10          ;SHIFT LEFT
1295      020322      MICPC=MICPC+1
1296      020324      <MOVE!WRTEBR!BR!<AA!SP10>>
1297      020324      BR7   RCVCK
1298      020324      MICPC=MICPC+1
1299      020324      <JUMP!BR7CON!<RCVCK-INIT&3000*4>!<RCVCK-INIT&777/2>>
1300      020326      TA1: BRWRT  BR,SELA!SP10        ;REREAD STATUS
1301      020326      MICPC=MICPC+1
1302      020326      <MOVE!WRTEBR!BR!<SELA!SP10>>
1303      020330      BR7   TA2          ;OK TO SEND?
1304      020330      MICPC=MICPC+1
1305      020330      <JUMP!BR7CON!<TA2-INIT&3000*4>!<TA2-INIT&777/2>>
1306      020332      TMTA6: ALWAYS I1          ;NO
1307      020332      MICPC=MICPC+1
1308      020332      <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
1309      020334      TA2: BR0   NUMSYN          ;IF UNNUMBPENDING -- SEND IT
1310      020334      MICPC=MICPC+1
1311      020334      <JUMP!BR0CON!<NUMSYN-INIT&3000*4>!<NUMSYN-INIT&777/2>>
1312      020336      BR1   10$
1313      020336      MICPC=MICPC+1
1314      020336      <JUMP!BR1CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
1315      020340      ALWAYS I1
1316      020340      MICPC=MICPC+1
1317      020340      <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
1318      020342      10$: BRSHFT
1319      020342      MICPC=MICPC+1
1320      020342      <MOVE!SHFTBR!WRTEBR!SELB>
1321      020344      BR4   I1          ;IF START MODE EXIT
1322      020344      MICPC=MICPC+1
1323      020344      <JUMP!BR4CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
1324      020346      NUMSYN: BRWRT  IBUS,MODEM          ;ARE WE STILL SENDING?
1325      020346      MICPC=MICPC+1
1326      020346      <MOVE!WRTEBR!IBUS!<MODEM>>
1327      020350      BRSHFT
1328      020350      MICPC=MICPC+1
1329      020350      <MOVE!SHFTBR!WRTEBR!SELB>
1330      020352      BR4   I1          ;RTS SET? IF SO WE ARE--STALL
1331      020352      MICPC=MICPC+1
1332      020352      <JUMP!BR4CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
1333      020354      TSTATE TMTA1
1334      020354      MICPC=MICPC+1
1335      020354      <MOVE!WRTEBR!IMM!<TMTA1-INIT&777/2>>
1336      020356      MICPC=MICPC+1
1337      020356      <MOVE!SPX!BR!SELB!SP2>
1338      020360      BRWRT  IMM,12
1339      020360      MICPC=MICPC+1
1340      020360      <MOVE!WRTEBR!IMM!<12>>
1341      020362      SP   BR,SELB,SP6          ;STORE IN SP6
1342      001020      MICPC=MICPC+1

```

DDCGAH.MAC 14-DEC-77 15:07

TMTA--FIRST CHARACTER OF HEADER

(1)	020362	063226	<MOVE!SPX!BR!SELB!SP6>	
1318	020364		ALWAYS I1	;BACK TO IDLE LOOP
(1)		001021	MICPC=MICPC+1	
(1)	020364	100451	<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>	
1320	020366		TMTTEXT: TSTATE TMTB	;STATE TO SEND COUNT
(1)		001022	MICPC=MICPC+1	
(1)	020366	000450	<MOVE!WRTEBR!IMM!<TMTB-INIT&777/2>>	
(1)		001023	MICPC=MICPC+1	
(1)	020370	063222	<MOVE!SPX!BR!SELB!SP2>	
1321	020372		BRWRTE BR,AA!SP1	;IN PORT STATUS
(1)		001024	MICPC=MICPC+1	
(1)	020372	060521	<MOVE!WRTEBR!BR!<AA!SP1>>	
1322	020374		BR4 10\$;DLE RCV'D SET-IGNORE MAINT MODE BIT
(1)		001025	MICPC=MICPC+1	
(1)	020374	113030	<JUMP!BR4CON!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>	
1323	020376		BRWRTE IMM,220	;WRITE DLE TO BR
(1)		001026	MICPC=MICPC+1	
(1)	020376	000620	<MOVE!WRTEBR!IMM!<220>>	
1324	020400		C TMTAS	;MAINT MODE - SEND DLE
(1)		001027	MICPC=MICPC+1	
(1)	020400	111033	<JUMP!CCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>	
1325	020402		BRWRTE BR,SELA!SP10	;NUMBERED MESSAGE?
(1)		001030	MICPC=MICPC+1	
(1)	020402	060610	<MOVE!WRTEBR!BR!<SELA!SP10>>	
1326	020404		BRO TMTUN	;NO - BRANCH
(1)		001031	MICPC=MICPC+1	
(1)	020404	112035	<JUMP!BROCON!<TMTUN-INIT&3000*4>!<TMTUN-INIT&777/2>>	
1327	020406		BRWRTE IMM,201	;OTHERWISE SEND AN SOH
(1)		001032	MICPC=MICPC+1	
(1)	020406	000601	<MOVE!WRTEBR!IMM!<201>>	
1328	020410		TMTAS: OUTPUT BR<SELB!TMTDAT>	;IN TMT SILO
(1)		001033	MICPC=MICPC+1	
(1)	020410	062230	<MOVE!WRTEBR!BR!<SELB!TMTDAT>>	
1329	020412		ALWAYS I1	
(1)		001034	MICPC=MICPC+1	
(1)	020412	100451	<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>	
1330	020414		TMTUN: TSTATE TMTI	
(1)		001035	MICPC=MICPC+1	
(1)	020414	000602	<MOVE!WRTEBR!IMM!<TMTI-INIT&777/2>>	
(1)		001036	MICPC=MICPC+1	
(1)	020416	063222	<MOVE!SPX!BR!SELB!SP2>	
1331	020420		BRWRTE IMM,5	;END TO BR
(1)		001037	MICPC=MICPC+1	
(1)	020420	000405	<MOVE!WRTEBR!IMM!<5>>	
1332	020422		ALWAYS TMTAS	
(1)		001040	MICPC=MICPC+1	
(1)	020422	110433	<JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>	

K15

DDCGAH.MAC 14-DEC-77 15:07

PAGE: 0192DM
SEQ 0192

TMTA--FIRST CHARACTER OF HEADER

1350	020424	001041	TMTA1: OUTPUT IMM,<1!OTMTCO>		;WRITE SOM TO TMTR CONTROL
(1)		002011	MICPC=MICPC+1		
(1)	020424		<MOVE!WROUT!IMM!<1!OTMTCO>>		
1351	020426	001042	BRWRITE IMM,226		;SYNC CHAR
(1)		000626	MICPC=MICPC+1		
(1)	020426		<MOVE!WRITEBR!IMM!<226>>		
1352	020430	001043	OUTPUT BR,SELB!TMTDAT>		;IN TMTR SILO
(1)		062230	MICPC=MICPC+1		
(1)	020430		<MOVE!WROUT!BR!<SELB!TMTDAT>>		
1353	020432	001044	SPBR BR,DECA,SP6		;DECREMENT SYNC COUNT
(1)		063566	MICPC=MICPC+1		
(1)	020432		<MOVE!SPBRX!BR!DECA!SP6>		
1354	020434	001045	Z TMTTEXT		;DONE?
(1)		111422	MICPC=MICPC+1		
(1)	020434		<JUMP!ZCOND!<TMTTEXT-INIT&3000*4>!<TMTTEXT-INIT&777/2>>		
1355	020436	001046	BRO TMTA1		;NO,OUTPUT SECOND SYNC?
(1)		112041	MICPC=MICPC+1		
(1)	020436		<JUMP!BROCON!<TMTA1-INIT&3000*4>!<TMTA1-INIT&777/2>>		
1356	020440	001047	ALWAYS I1		;NO
(1)		100451	MICPC=MICPC+1		
(1)	020440		<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>		

L15

DDCGAM.MAC

14-DEC-77 15:07

TMTB--OUTPUT FIRST CHAR OF COUNT

PAGE: 0193DM
SEQ 0193

1359
13EC

.SBTTL TMTB--OUTPUT FIRST CHAR OF COUNT
:

1362	020442	001050	TMTB:	SPBR BR, INCA, SP12	; INCREMENT MSG NO
(1)				MICPC=MICPC+1	
(1)	020442	063472		<MOVE!SPBRX!BR!INCA!SP12>	
1363	020444			OUTPUT BR, SELB!OINDAT1	
(1)		001051		MICPC=MICPC+1	
(1)	020444	062220		<MOVE!WROUT!BR!<SELB!OINDAT1>>	
1364	020446			LDMA BR, SELA!SP16	; GET POINTER TO NEXT TMT LINK
(1)		001052		MICPC=MICPC+1	
(1)	020446	070216		<MOVE!LDMA!BR!<SELA!SP16>>	
1365	020450			MEMINC IMM, 3	; WRITE MSG TMTED TO FLAGS
(1)		001053		MICPC=MICPC+1	
(1)	020450	016403		<MOVE!WRMEM!INCMAR!IMM!<3>>	
1366	020452			MEMINC BR, SELA!SP12	; PICK UP MSGNO
(1)		001054		MICPC=MICPC+1	
(1)	020452	076612		<MOVE!WRMEM!INCMAR!BR!<SELA!SP12>>	
1367	020454			BRWRTE IMM, TML8	; GET WRAPAROUND ADDRESS
(1)		001055		MICPC=MICPC+1	
(1)	020454	000543		<MOVE!WRTEBR!IMM!<TML8>>	
1368	020456			CMP BR, SP16	; WRAPAROUND
(1)		001056		MICPC=MICPC+1	
(1)	020456	060376		<SUBTC!BR!SP16>	
1369	020460			Z TB2	
(1)		001057		MICPC=MICPC+1	
(1)	020460	111504		<JUMP!ZCOND!<TB2-INIT&3000*4>!<TB2-INIT&777/2>>	
1370	020462			BRWRTE IMM, 6	; OFFSET TO NEXT LINK
(1)		001060		MICPC=MICPC+1	
(1)	020462	000406		<MOVE!WRTEBR!IMM!<6>>	
1371	020464		TB1:	SP BR, ADD, SP16	; UPDATE THE POINTER
(1)		001061		MICPC=MICPC+1	
(1)	020464	063016		<MOVE!SPX!BR!ADD!SP16>	
1372	020466			STATE TMT	; ADDRESS TMT STATE
(1)		001062		MICPC=MICPC+1	
(1)	020466	000506		<MOVE!WRTEBR!IMM!<TMT-INIT&777/2>>	
1373	020470		TB0:	SP BR, SELB, SP2	; UPDATE IT
(1)		001063		MICPC=MICPC+1	
(1)	020470	063222		<MOVE!SPX!BR!SELB!SP2>	
1374	020472			OUTPUT <MEMX!INCMAR>, SELB!IBA1	; WRITE LOW BYTE OF ADDRESS
(1)		001064		MICPC=MICPC+1	
(1)	020472	056224		<MOVE!WROUT!MEMX!INCMAR!<SELB!IBA1>>	
1375	020474			OUTPUT <MEMX!INCMAR>, SELB!IBA2	; WRITE HIGH BYTE OF ADDRESS
(1)		001065		MICPC=MICPC+1	
(1)	020474	056225		<MOVE!WROUT!MEMX!INCMAR!<SELB!IBA2>>	
1376	020476			SP MEMX, SELB, SP7 ; HIGH BYTE OF COUNT TO SP7	
(1)		001066		MICPC=MICPC+1	
(1)	020476	043227		<MOVE!SPX!MEMX!SELB!SP7>	
1377					; WAIT TO MASK OFF MEM EXT. BITS
1378	020500			SP IBUS, NPR, SPO	
(1)		001067		MICPC=MICPC+1	
(1)	020500	123200		<MOVE!SPX!IBUS!NPR!SPO>	
1379	020502			BRWRTE IMM, 220	
(1)		001070		MICPC=MICPC+1	
(1)	020502	000620		<MOVE!WRTEBR!IMM!<220>>	
1380	020504			SP BR, AANDB, SPO	
(1)		001071		MICPC=MICPC+1	
(1)	020504	063260		<MOVE!SPX!BR!AANDB!SPO>	
1381	020506			SP IMM, 300, SP6	; MASK FOR MXT

```

(1) 020506 001072 MICPC=MICPC+1
(1) 020510 003306 <MOVE!SPX!IMM!300!SP6>
1382 020510 001073 BRWRT MEMX!INCMAR,AANDB!SP6 ;TURN OFF CC2
(1) 020510 054666 MICPC=MICPC+1
(1) 020512 001074 <MOVE!WRTEBR!MEMX!INCMAR!<AANDB!SP6>>
1383 020512 042230 OUTPUT MEMX SELB!TMTDAT ;ALSO WRITE COUNT TO TMTR SILO
(1) 020512 001074 MICPC=MICPC+1
(1) 020514 001620 <MOVE!WROUT!MEMX!<SELB!TMTDAT>>
1384 020514 001075 BRSHFT ;SHIFT BITS INTO CORRECT POSITION
(1) 020514 001620 MICPC=MICPC+1
(1) 020516 001076 <MOVE!SHFTBR!WRTEBR!SELB>
1385 020516 001076 BRSHFT
(1) 020516 001620 MICPC=MICPC+1
(1) 020520 001077 <MOVE!SHFTBR!WRTEBR!SELB>
1386 020520 001620 BRSHFT
(1) 020522 001100 MICPC=MICPC+1
(1) 020524 001620 <MOVE!SHFTBR!WRTEBR!SELB>
1388 020524 061310 OUT BR AORB!ONPR
(1) 020526 001101 MICPC=MICPC+1
1389 020526 043626 <MOVE!WROUTX!BR!<AORB!ONPR>>
(1) 020526 001102 SPBR MEMX SELB,SP6 ;LOWBYTE OF COUNT TO SP6
(1) 020530 001103 MICPC=MICPC+1
(1) 020530 100451 ALWAYS I1
1396 020532 001104 <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
(1) 020532 000726 BRWRT IMM,<<TML1-TML8>&377> ;GO BACK TOFIRST LINK
(1) 020534 001105 MICPC=MICPC+1
1397 020534 110461 <MOVE!WRTEBR!IMM!<<TML1-TML8>&377>>
(1) 020534 110461 ALWAYS TB1
1398 MICPC=MICPC+1
<JUMP!ALCOND!<TB1-INIT&3000*4>!<TB1-INIT&777/2>>
;

```


DDCGAH.MAC 14-DEC-77 15:07

TMTC--OUTPUT SECOND CHAR OF COUNT

```

1400
1401
1402 020536 001106
(1) 020536 000477
1403 020540 001107
(1) 020540 063667
1404 020542 001110
(1) 020542 062230
1409
1410
1411 020544 001111
(1) 020544 000520
1412 020546 001112
(1) 020546 063166
1413 020550 001113
(1) 020550 111115
1414 020552 001114
(1) 020552 063167
1415 020554 001115
(1) 020554 010171
1416 020556 001116
(1) 020556 042230
1417 020560 001117
(1) 020560 110556
1418
1419 020562
1420 020562 001120
(1) 020562 123600
1421 020564 001121
(1) 020564 102051
1422 020566 001122
(1) 020566 022010
1423 020570 001123
(1) 020570 000525
1424 020572 001124
(1) 020572 110573
1425
1426
1427 020574 001125
(1) 020574 000532
    
```

```

.SBTTL TMTC--OUTPUT SECOND CHAR OF COUNT
TMTC: BRWRT IMM,77 ;MASK TO CLEAR MXT BITS
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<77>>
      SPBR BP, AANDB, SP7 ;CLEAR THEM
      MICPC=MICPC+1
      <MOVE!SPBRX!BR!AANDB!SP7>
      OUTPUT DP, <SELB!TMTDAT> ;WRITE TO TMT SILO
      MICPC=MICPC+1
      <MOVE!WROUT!DP!<SELB!TMTDAT>>
      ;
.SBTTL TMTD--RESPONSE FIELD-NUMBERED MESSAGE
TMTD: STATE TMTD
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTD-INIT&777/2>>
      SP BR, DECA, SP6 ;ADJUST COUNT FOR TWO'S COMPLEMENT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP6>
      C TD2 ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<TD2-INIT&3000*4>!<TD2-INIT&777/2>>
      SP BR, DECA, SP7 ;DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP7>
      TD2: LDMA IMM, ISP11 ;RESP FIELD ADDR TO MAR
      MICPC=MICPC+1
      <MOVE!LDMA!IMM!<ISP11&377>>
      TD3: OUTPUT MEMX, SELB!TMTDAT ;WRITE IT TO SILO
      MICPC=MICPC+1
      <MOVE!WROUT!MEMX!<SELB!TMTDAT>>
      ALWAYS XEXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
      ;
.SBTTL TMTE--NUMBER FIELD--NUMBERED MESSAGE
TMTE: SPBR IBUS, NPR, SPO ;READ NPR CONTROL REGISTER
      MICPC=MICPC+1
      <MOVE!SPBRX!IBUS!NPR!SPO>
      BRO I1 ;BUSY - GET OUT
      MICPC=MICPC+1
      <JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
      OUTPUT IBUS, <INDAT1!TMTDAT> ;WRITE IT TO THE SILO
      MICPC=MICPC+1
      <MOVE!WROUT!IBUS!<INDAT1!TMTDAT>>
      STATE TMTF
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTF-INIT&777/2>>
      ALWAYS TH3
      MICPC=MICPC+1
      <JUMP!ALCOND!<TH3-INIT&3000*4>!<TH3-INIT&777/2>>
      ;
.SBTTL TMTF--NUMBERED MSG ADDRESS FIELD
TMTF: STATE TF1
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TF1-INIT&777/2>>
    
```

DDCGAM.MAC 14-DEC-77 15:07

TMTF--NUMBERED MSG ADDRESS FIELD

```

1428 020576 001126
(1) 020576 063222
1429 020600 001127
(1) 020600 000401
1431 020602 001130
(1) 020602 062230
1432 020604 001131
(1) 020604 100451
1437
1438 020606 001132
(1) 020606 000402
1439 020610 001133
(1) 020610 062231
1440 020612 001134
(1) 020612 062230
1441 020614 001135
(1) 020614 020500
1442 020616 001136
(1) 020616 112155
1443 020620 001137
(1) 020620 000541
1444 020622 001140
(1) 020622 110556
    
```

```

TF2.  SP      BR,SELB,SP2
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      BRWRT IMM,1 ;LOAD ADDRESS
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<1>>

TF3.  OUTPUT BR,SELB!TMTDAT>
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!TMTDAT>>
      ALWAYS I1
      MICPC=MICPC+1
      <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
      .SATT! TF!-NUMBERED MSG HEADER EOM

TF1:  BRWRT IMM,2 ;EOM MASK TO BR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<2>>
      OUTPUT BR,SELB!OTMTCO ;UPDATE TMTR CONTROL REGISTER
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!OTMTCO>>
      OUTPUT BR,SELB!TMTDAT ;OUTPUT A GARBAGE CHAR
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!TMTDAT>>
      BRWRT IBUS,IIBA1 ;READ LOW ORDER FROM INBA
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<IIBA1>>
      JRU TMTF1 ;IF ODD BYTE--BRANCH
      MICPC=MICPC+1
      <JUMP!BROCON!<TMTF1-INIT&3000*4>!<TMTF1-INIT&777/2>>
      STATE TMTH
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM! TMTH-INIT&777/2>>
      ALWAYS XEXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
    
```

COGAM.MAC 14-DEC-77 15:07

TF1-NUMBERED MSG HEADER EOM

1446
1447
1448 020624 001141
(1) 020624 123600
1450 020626 001142
(1) 020626 113144
1452 020630 001143
(1) 020630 102051
1453 020632 001144
(1) 020632 022010
1454 020634 001145
(1) 020634 023100
1455 020636 001146
(1) 020636 062064
1456 020640 001147
(1) 020640 063166
1457 020642 001150
(1) 020642 111153
1458 020644 001151
(1) 020644 063167
1459 020646 001152
(1) 020646 101771
1460 020650
1462 020650 001153
(1) 020650 020620
1463 020652 001154
(1) 020652 113160
1465 020654 001155
(1) 020654 000560
1466 020656 001156
(1) 020656 063222
1467 020660 001157
(1) 020660 100451
1468 020662
1473 020662 001160
(1) 020662 022030
1474 020664 001161
(1) 020664 023100
1475 020666

```

:*****TIME CRITICAL PATH--MODIFY WITH GREAT CARE
.SATTL TMTH--ROUTINE TO OUTPUT DATA CHARACTERS
TMTH: SPBR IBUS,NPP,SPO ;READ NPR CONTROL
      MICPC=MICPC+1
      <MOVE!SPBRX!IBUS!NPR!SPO>
      BR4 55 ;IF RECV NPR --PROCESS
      MICPC=MICPC+1
      <JUMP!BR4CON!<55-INIT&3000*4>!<55-INIT&777/2>>
      BRO I1 ;IF NPR IN PROGRESS --BRANCH
      MICPC=MICPC+1
      <JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
SS: OUTPUT IBUS <INDAT1!TMTDAT> ;WRITE THE EVEN CHAR TO TMT SILO
      MICPC=MICPC+1
      <MOVE!WROUT!IBUS!<INDAT1!TMTDAT>>
      SP IBUS,IIBA1,SPO ;READ LOW BYTE OF BA TO SP
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IIBA1!SPO>
      OUTPUT BR <INCA!IBA1> ;OUTPUT INCREMENTED BA
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<INCA!IBA1>>
      SP BR,DECA,SP6 ;DECREMENT CHARACTER COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP6>
      C TH6 ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<TH6-INIT&3000*4>!<TH6-INIT&777/2>>
      SP BR,DECA,SP7 ;DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP7>
      Z HEH1 ;BYTE COUNT ZERO
      MICPC=MICPC+1
      <JUMP!ZCOND!<HEH1-INIT&3000*4>!<HEH1-INIT&777/2>>
TH6: BRWRT IBUS,TMTCON ;READ TMTR CONTROL CSR
      MICPC=MICPC+1
      <MOVE!WRTBR!IBUS!<TMTCON>>
      BR4 TH9 ;IF MORE ROOM IN SILO--BRANCH
      MICPC=MICPC+1
      <JUMP!BR4CON!<TH9-INIT&3000*4>!<TH9-INIT&777/2>>
TMTF1: STATE TMTHO
      MICPC=MICPC+1
      <MOVE!WRTBR!IMM!<TMTHO-INIT&777/2>>
EXIT: SP BR,SELB,SP2 ;STORE NEW TRANSMIT STATE
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      ALWAYS I1
      MICPC=MICPC+1
      <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
TMTHO:
TH9: OUTPUT IBUS <INDAT2!TMTDAT> ;ODD CHAR TO SILO
      MICPC=MICPC+1
      <MOVE!WROUT!IBUS!<INDAT2!TMTDAT>>
      SP IBUS,IIBA1,SPO ;READ LOW BYTE TO BA
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IIBA1!SPO>
      OUTPUT BR <INCA!IBA1> ;OUTPUT THE INCREMENTED BA
    
```

DDCGAM.MAC 14-DEC-77 15:07

TMTH--ROUTINE TO OUTPUT DATA CHARACTERS

```

(1) 001162
(1) 020666 062064
1480 020670
(1) 001163
(1) 020670 111360
1481 020672
(1) 001164
(1) 020672 063166
1482 020674
(1) 001165
(1) 020674 111170
1483 020676
(1) 001166
(1) 020676 063167
1484 020700
(1) 001167
(1) 020700 101771
1485 020702
(1) 001170
(1) 020702 123600
1487 020704
(1) 001171
(1) 020704 112200
1489 020706
(1) 001172
(1) 020706 000541
1490 020710
(1) 001173
(1) 020710 063222
1491 020712
(1) 001174
(1) 020712 000556
1492 020714
(1) 001175
(1) 020714 063260
1493 020716
(1) 001176
(1) 020716 061070
1498 020720
(1) 001177
(1) 020720 100451
1499 020722
(1) 001200
(1) 020722 000570
1500 020724
(1) 001201
(1) 020724 110556
1502

MICPC=MICPC+1
<MOVE!WROUT!BR!<INCA!IBA1>>
C HOINCH
MICPC=MICPC+1
<JUMP!CCOND!<HOINCH-INIT&3000*4>!<HOINCH-INIT&777/2>>
TH8: SP BR,DECA,SP6 ;DECREMENT CHARACTERCOUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP6>
C TH7 ;NO OVERFLOW
MICPC=MICPC+1
<JUMP!CCOND!<TH7-INIT&3000*4>!<TH7-INIT&777/2>>
SP BR,DECA,SP7 ;DECREMENT HIGH BYTE OF COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP7>
Z HEH1 ;BYTE COUNT ZERO
MICPC=MICPC+1
<JUMP!ZCOND!<HEH1-INIT&3000*4>! HEH1-INIT&777/2>>
TH7: SPBR IBUS,NPR,SPO ;READ NPR REGISTER
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!NPR!SPO>
BR0 TH2 ;IF NPR BUSY WAIT TO GO
MICPC=MICPC+1
<JUMP!BROCON!<TH2-INIT&3000*4>!<TH2-INIT&777/2>>
STATE TMTH
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTH-INIT&777/2>>
TH3: SP BR,SELB,SP2 ;SAVE TSTATE
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
TH3X: BRWRTE IMM,156 ;CLEAR CO AND CI
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<156>>
SP BR,AANDB,SPO ;CLEAR THE BITS
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SPO>
OUT BR,<INCA!ONPR>
MICPC=MICPC+1
<MOVE!WROUTX!BR!<INCA!ONPR>>
ALWAYS I1
MICPC=MICPC+1
<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
TH2: STATE TH7
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TH7-INIT&777/2>>
ALWAYS XEXIT
MICPC=MICPC+1
<JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
:*****END TIME CRITICAL PATH*****

```

0000AM.MAC 14-DEC-77 15:07

TMTI--SEND UNNUMBERED TYPE FIELD

1504
 1505 020726 001202
 (1)
 (1) 020726 010151
 1506 020730 001203
 (1)
 (1) 020730 043226
 1507 020732 001204
 (1)
 (1) 020732 000606
 1508 020734 001205
 (1)
 (1) 020734 110516
 1509
 1510
 1511 020736 001206
 (1)
 (1) 020736 010152
 1512 020740 001207
 (1)
 (1) 020740 000611
 1513 020742 001210
 (1)
 (1) 020742 110516
 1514
 1515
 1516 020744 001211
 (1)
 (1) 020744 000403
 1517 020746 001212
 (1)
 (1) 020746 060346
 1518 020750 001213
 (1)
 (1) 020750 000617
 (1) 001214
 (1) 020752 063222
 1519 020754 001215
 (1)
 (1) 020754 111224
 1520 020756 001216
 (1)
 (1) 020756 110515
 1521
 1522
 1523 020760 001217
 (1)
 (1) 020760 000631
 (1) 001220
 (1) 020762 063222
 1524 020764 001221
 (1)
 (1) 020764 000403
 1525 020766 001222
 (1)
 (1) 020766 060366

```

.TSBTTL TMTI--SEND UNNUMBERED TYPE FIELD
TMTI: LDMA IMM,T ;ADDRESS OF TYPE FIELD TO MAR
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<T&377>>
      SP MEMX,SELB,SP6 ;COPY IT TO SP6
      MICPC=MICPC+1
      <MOVE!SPX!MEMX!SELB!SP6>
      STATE TMTJ
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTJ-INIT&777/2>>
      ALWAYS TD3
      MICPC=MICPC+1
      <JUMP!ALCOND!<TD3-INIT&3000*4>!<TD3-INIT&777/2>>
      .
.TSBTTL TMTJ--SEND SUB-TYPE FIELD
TMTJ: LDMA IMM,ST ;ADDRESS OF SUB-TYPE FIELD TO MAR
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<ST&377>>
      STATE TMTK
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTK-INIT&777/2>>
      ALWAYS TD3
      MICPC=MICPC+1
      <JUMP!ALCOND!<TD3-INIT&3000*4>!<TD3-INIT&777/2>>
      .SBTTL TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG)
TMTK: BRWRT IMM,3 ;WRITE A 3 TO BR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<3>>
      NOP BR,SUB,SP6 ;IF TYPE LESS THAN 3
      MICPC=MICPC+1
      <BR!SUB!SP6>
      TSTATE TMTL
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTL-INIT&777/2>>
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      C TMTLO
      MICPC=MICPC+1
      <JUMP!CCOND!<TMTLO-INIT&3000*4>!<TMTLO-INIT&777/2>>
      ALWAYS TD2
      MICPC=MICPC+1
      <JUMP!ALCOND!<TD2-INIT&3000*4>!<TD2-INIT&777/2>>
      .
.TSBTTL TMTL--UNNUMB MSG NUMBER FIELD
TMTL: TSTATE TMTM
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTM-INIT&777/2>>
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      BRWRT IMM,3
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<3>>
      CMP BR,SP6 ;IS MESSAGE REP
      MICPC=MICPC+1
      <SUBTC!BR!SP6>

```

DDCGAH.MAC 14-DEC-77 15:07

TMTL--UNNUMB MSG NUMBER FIELD

1526	020770	001223	Z	TMTL1	;YES
(1)			MICPC=MICPC+1		
(1)	020770	111627	<JUMP!ZCOND!<TMTL1-INIT&3000*4>!<TMTL1-INIT&777/2>>		
1527	020772	001224	TMTLO: BRWRT IMM,0		;ADDRESS CONTNAT OF ZERO
(1)			MICPC=MICPC+1		
(1)	020772	000400	<MOVE!WRTEBR!IMM!<0>>		
1532	020774	001225	OUTPUT BR <SELB!TMTDAT>		;SEND IT OUT
(1)			MICPC=MICPC+1		
(1)	020774	062230	<MOVE!WROUT!BR!<SELB!TMTDAT>>		
1533	020776	001226	ALWAYS I1		;BACK TO IDLE LOOP
(1)			MICPC=MICPC+1		
(1)	020776	100451	<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>		
1535			TMTL1:		
1536	021000	001227	BRWRT BR SELA!SP12		;WRITE A RESPONSE
(1)			MICPC=MICPC+1		
(1)	021000	060612	<MOVE!WRTEBR!BR!<SELA!SP12>>		
1537	021002	001230	ALWAYS TMTAS		
(1)			MICPC=MICPC+1		
(1)	021002	110433	<JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>		
1538			:		
1539			SBTTL TMTM--UNNUMB MSG--STATION ADDRESS		
1540	021004	001231	STATE TNEOM		
(1)			MICPC=MICPC+1		
(1)	021004	000633	<MOVE!WRTEBR!IMM!<TNEOM-INIT&777/2>>		
1541	021006	001232	ALWAYS TF2		
(1)			MICPC=MICPC+1		
(1)	021006	110526	<JUMP!ALCOND!<TF2-INIT&3000*4>!<TF2-INIT&777/2>>		
1542	021010	001233	TNEOM: BRWRT IMM,2		;END OF MESSAGE TO BR
(1)			MICPC=MICPC+1		
(1)	021010	000402	<MOVE!WRTEBR!IMM!<2>>		
1543	021012	001234	OUTPUT BR <SELB!OTMTCO>		
(1)			MICPC=MICPC+1		
(1)	021012	062231	<MOVE!WROUT!BR!<SELB!OTMTCO>>		
1544	021014	001235	OUTPUT BR <SELB!TMTDAT>		;OUTPUT A GARBAGE CHARACTER
(1)			MICPC=MICPC+1		
(1)	021014	062230	<MOVE!WROUT!BR!<SELB!TMTDAT>>		
1549	021016	001236	BRWRT BR AA!SP10		;SHIFT STATUS LEFT
(1)			MICPC=MICPC+1		
(1)	021016	060530	<MOVE!WRTEBR!BR!<AA!SP10>>		
1550	021020	001237	BR7 TNEOM2		;IF HDX SET---BRANCH TO CLEAR OK TO SEND
(1)			MICPC=MICPC+1		
(1)	021020	113644	<JUMP!BR7CON!<TNEOM2-INIT&3000*4>!<TNEOM2-INIT&777/2>>		
1551	021022	001240	BRWRT IMM,376		;MASK TO TURN OFF UNNUMB PENDDING
(1)			MICPC=MICPC+1		
(1)	021022	000776	<MOVE!WRTEBR!IMM!<376>>		
1552	021024	001241	TNEOM1: SP BR AANDB,SP10		;MASK TO LINE STATUS WORD
(1)			MICPC=MICPC+1		
(1)	021024	063270	<MOVE!SPX!BR!AANDB!SP10>		
1553	021026	001242	TEOM2: STATE TMTA		
(1)			MICPC=MICPC+1		
(1)	021026	000400	<MOVE!WRTEBR!IMM!<TMTA-INIT&777/2>>		
1554	021030	001243	ALWAYS XEXIT		
(1)			MICPC=MICPC+1		
(1)	021030	110556	<JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>		
1555	021032	001244	TNEOM2: BRWRT IMM,176		;CLEAR OK TO SEND AND UNNUMB PENDDING
(1)			MICPC=MICPC+1		

H16

0000AM.MAC 14-DEC-77 15:07

TMTM--UNNUMB MSG--STATION ADDRESS

PAGE: 02020M
SEQ 0202

01	021032	000576
1556	021034	
01		001245
01	021034	110641

<MOVE!WRTEBR!IMM!<176 /
ALWAYS TNEOM1
MICPC=MICPC+1
<JUMP!ALCOND!<TNEOM1-INIT\$3000+4 /!<TNEOM1-INIT\$777 2 /

DDCGAH.MAC 14-DEC-77 15:07

TIMSRV--TIMEOUT ROUTINE--SENDS REP

```

1558          .SBTTL TIMSRV--TIMEOUT ROUTINE--SENDS REP
1559          ;
1560          ;ENABLE LSB
TIMSRV: BRWRT IMM,177          ;MASK OFF BR REQ
          MICPC=MICPC+1
          <MOVE!WRTEBR!IMM!<177>>
1561 021036 001246
          (1) 021036 000577
1562          ;RESET TIMER---SLICK MOVE
1563          ;SINCE TIMER IS RESET BY WRITING
1564          ;A 1 AND THE EXPIRATION LOOKS
1565          ;LIKE 1--VOILA
1566 021040 001247
          (1) 021040 061271
          OUT BR,<AANDB!OBR>          ;AND THE BIT ON
          MICPC=MICPC+1
          <MOVE!WROUTX!BR!<AANDB!OBR>>
1567 021042 001250
          (1) 021042 063074
          SP BR,INCA,SP14          ;INCREMENT THE COUNT
          MICPC=MICPC+1
          <MOVE!SPX!BR!INCA!SP14>
1568          ;NOTE: STATE OF "C" BIT
1569          ;MUST BE PRESERVED FROM THIS
1570          ;POINT UNTIL IT IS TESTED
1571 021044
          (1) 021044
          TIME0: BRWRT BR,SELA!SP1          ;READ STATUS BYTE
          MICPC=MICPC+1
          <MOVE!WRTEBR!BR!<SELA!SP1>>
1573 021046 001252
          (1) 021046 102047
          BRO IDLE
          MICPC=MICPC+1
          <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1574 021050 001253
          (1) 021050 103447
          BR7 IDLE          ;IF IN MAINT. MODE DISABLE TIMER
          MICPC=MICPC+1
          <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1575 021052 001254
          (1) 021052 060610
          BRWRT BR,SELA!SP10          ;READ LINE STATUS REGISTER
          MICPC=MICPC+1
          <MOVE!WRTEBR!BR!<SELA!SP10>>
1576 021054 001255
          (1) 021054 061620
          BSHFTB          ;ROTATE IT RIGHT ONE BIT
          MICPC=MICPC+1
          <MOVE!SHFTBR!SELB!BR>
1577 021056 001256
          (1) 021056 103161
          BR4 BS1          ;START MODE - RESEND A START
          MICPC=MICPC+1
          <JUMP!BR4CON!<BS1-INIT&3000*4>!<BS1-INIT&777/2>>
1578          ;TEST STATE OF C BIT FROM INC SP14
1579 021060 001257
          (1) 021060 111263
          C 20$          ;IF CARRY SET COUNT HAS EXPIRED
          MICPC=MICPC+1
          <JUMP!CCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
1580 021062 001260
          (1) 021062 116315
          BRO TABUPD          ;NUMBERED MESSAGE IN PROGRESS
          MICPC=MICPC+1
          <JUMP!BROCON!<TABUPD-INIT&3000*4>!<TABUPD-INIT&777/2>>
1581 021064 001261
          (1) 021064 117715
          BR7 TABUPD          ;UNNUMBMSGIN PROGRESS
          MICPC=MICPC+1
          <JUMP!BR7CON!<TABUPD-INIT&3000*4>!<TABUPD-INIT&777/2>>
1582 021066 001262
          (1) 021066 110723
          ALWAYS SNDACK          ;ELSE TRY TO SEND AN ACK
          MICPC=MICPC+1
          <JUMP!ALCOND!<SNDACK-INIT&3000*4>!<SNDACK-INIT&777/2>>
1583 021070
          (1) 021070
          TIME1:
          20$: BRWRT IMM,205          ;SET OK TO SEND,UNNUMB PEND,AND
          MICPC=MICPC+1
          <MOVE!WRTEBR!IMM!<205>>
1585 021070 001263
          (1) 021070 000605

```


DDCGAH.MAC 14-DEC-77 15:07

TIMSRV--TIMEOUT ROUTINE--SENDS REP

```

1586                                     ;LINE GONE IDLE
1587 021072 SP BR AORB,SP10
(1) MICPC=MICPC+1
(1) 021072 063310 <MOVE!SPX!BR!AORB!SP10>
1588 021074 SP IMM 374,SP14 ;RESET THE TIMER TICK COUNT
(1) MICPC=MICPC+1
(1) 021074 001265 <MOVE!SPX!IMM!374!SP14>
1589 021076 BRWRT BR SELA!SP12 ;GET LAST NUMBER SENT
(1) MICPC=MICPC+1
(1) 021076 060612 <MOVE!WRTEBR!BR!<SELA!SP12>>
1590 021100 CMP BR SP17 ;COMPARE TO LAST ACKED
(1) MICPC=MICPC+1
(1) 021100 001267 <SUBTC!BR!SP17>
(1) 021100 060377 Z SNDACK ;IF EQ --SEND ACK
1591 021102 MICPC=MICPC+1
(1) 021102 001270 <JUMP!ZCOND!<SNDACK-INIT&3000*4>!<SNDACK-INIT&777/2>>
(1) 021102 111723 LDMA IMM REPCS ;CUMULATIVE REPS SENT
1592 021104 MICPC=MICPC+1
(1) 021104 001271 <MOVE!LDMAR!IMM!<REPCS&377>>
(1) 021104 010015 SP MEMX SELB,SPO ;COPY IT TO SPO
1593 021106 MICPC=MICPC+1
(1) 021106 001272 <MOVE!SPX!MEMX!SELB!SPO>
(1) 021106 043220 MEM BR INCA!SPO ;INCREMENT IT
1594 021110 MICPC=MICPC+1
(1) 021110 001273 <MOVE!WRMEM!BR!<INCA!SPO>>
(1) 021110 062460 LDMA IMM REPST ;ADDRESS DYNAMIC REP COUNTER
1595 021112 MICPC=MICPC+1
(1) 021112 001274 <MOVE!LDMAR!IMM!<REPST&377>>
(1) 021112 010003 BRWRT MEMX SELB ;COPY IT TO THE BR
1596 021114 MICPC=MICPC+1
(1) 021114 001275 <MOVE!WRTEBR!MEMX!<SELB
1597 021116 BSHFTB
(1) MICPC=MICPC+1
(1) 021116 001276 <MOVE!SHFTBR!SELB!BR>
(1) 021116 061620 MEM BR SELB
1598 021120 MICPC=MICPC+1
(1) 021120 001277 <MOVE!WRMEM!BR!<SELB>>
(1) 021120 062620 BRO RTHRES
1599 021122 MICPC=MICPC+1
(1) 021122 001300 <JUMP!BROCON!<RTHRES-INIT&3000*4>!<RTHRES-INIT&777/2>>
(1) 021122 106376 LDMA IMM T ;LOAD ADDRESS OF TYPE FIELD IN UNNUMB SK
1600 021124 MICPC=MICPC+1
(1) 021124 001301 <MOVE!LDMAR!IMM!<T&377>>
(1) 021124 010151 MEMINC IMM 3 ;LOAD REP TYPE
1601 021126 MICPC=MICPC+1
(1) 021126 001302 <MOVE!WRMEM!INCMAR!IMM!<3>>
(1) 021126 016403 ALWAYS SA1
1602 021130 MICPC=MICPC+1
(1) 021130 001303 <JUMP!ALCOND!<SA1-INIT&3000*4>!<SA1-INIT&777/2>>
(1) 021130 110725 .DSABLE LSB
1603
1604

```

DDCGAH.MAC 14-DEC-77 15:07

TIMSRV--TIMEOUT ROUTINE--SENDS REP

```

1606 021132 001304
(1) 021132 120620
1607 021134 001305
(1) 021134 106364
1608 021136 001306
(1) 021136 000402
1609 021140 001307
(1) 021140 062231
1610 021142 001310
(1) 021142 062230
1611 021144 001311
(1) 021144 060601
1612 021146 001312
(1) 021146 113750
1613 021150 001313
(1) 021150 070216
1614 021152 001314
(1) 021152 040620
1615 021154 001315
(1) 021154 112242
1616 021156 001316
(1) 021156 000775
1617 021160 001317
(1) 021160 063670
1618 021162 001320
(1) 021162 112242
1619 021164 001321
(1) 021164 000400
(1) 021166 001322
(1) 021166 063222
1620
1621 021170 001323
(1) 021170 010151
1622 021172 001324
(1) 021172 016401
1623 021174 001325
(1) 021174 000405
1624 021176 001326
(1)

```

```

TEOM:  BRWRT  IBUS,UBBR
        MICPC=MICPC+1
        <MOVE!WRTEBR!IBLS!<UBBR>>
        BR0      NXMERR ;NON-EXISTANT MEMORY
        MICPC=MICPC+1
        <JUMP!BROCON!<NXMERR-INIT&3000*4>!<NXMERR-INIT&777/2>>
        BRWRT  IMM,2 ;EOM TO BR
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<2>>
        OUTPUT BR <SELB!OTMTCO> ;WRITE TMTR CONTROL
        MICPC=MICPC+1
        <MOVE!WROUT!BR!<SELB!OTMTCO>>
        OUTPUT BR <SELB!TMTDAT> ;WRITE GARBAGE DATA
        MICPC=MICPC+1
        <MOVE!WROUT!BR!<SELB!TMTDAT>>
        BRWRT  BR SELA!SP1 ;CKECK FOR BOOT MODE
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<SELA!SP1>>
        BR7      BTEOM ;---IF SET IS MAINT MSG
        MICPC=MICPC+1
        <JUMP!BR7CON!<BTEOM-INIT&3000*4>!<BTEOM-INIT&777/2>>
TEOM1: LDMA  BR SELA!SP16 ;ADDRESS LAST TMT LINK
        MICPC=MICPC+1
        <MOVE!LDMAR!BR!<SELA!SP16>>
        BRWRT  MEMX SELB
        MICPC=MICPC+1
        <MOVE!WRTEBR!MEMX!<SELB>>
        BR0      TEOM2
        MICPC=MICPC+1
        <JUMP!BROCON!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>
TEOM3: BRWRT  IMM,375 ;TURN OFF MESSAGE PENDING
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<375>>
        SPBR  BR AANDB,SP10 ;
        MICPC=MICPC+1
        <MOVE!SPBRX!BR!AANDB!SP10>
        BR0      TEOM2 ;IF UNNUMB PENDING--GO AWAY
        MICPC=MICPC+1
        <JUMP!BROCON!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>
        TSTATE TMTA
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<TMTA-INIT&777.2>>
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP2>
        .SBTTL SNDACK--ROUTINE TO SEND AN ACK
SNDACK: LDMA  IMM,T
        MICPC=MICPC+1
        <MOVE!LDMAR!IMM!<T&377>>
        MEMINC IMM,1
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IMM!<1>>
SA1:  BRWRT  IMM,5
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<5>>
SA2:  MEMINC IMM,300
        MICPC=MICPC+1

```

DDCGA4.MAC 14-DEC-77 15:07

SNDACK--ROUTINE TO SEND AN ACK

```

(1) 021176 016700
1625 021200 001327
(1) 021200 063310
1626 021202 001330
(1) 021202 100447
1627
1628 021204 001331
(1) 021204 120600
1629 021206 001332
(1) 021206 102047
1630 021210 001333
(1) 021210 070604
1631 021212 001334
(1) 021212 103571
1632 021214 001335
(1) 021214 036400
1633 021216 001336
(1) 021216 036420
1634 021220 001337
(1) 021220 000402
1635 021222 001340
(1) 021222 063004
1636 021224 001341
(1) 021224 023100
1637 021226 001342
(1) 021226 062004
1638 021230 001343
(1) 021230 023120
1639 021232 001344
(1) 021232 062105
1640 021234 001345
(1) 021234 123200
1641 021236 001346
(1) 021236 115151
1642 021240 001347
(1) 021240 110574

```

```

SA3:  <MOVE!WRMEM!INCMAR!IMM!<300>>
      SP BR,AORB,SP10
      MICPC=MICPC+1
      <MOVE!SPX!BR!AORB!SP10>
      ALWAYS IDLE
      MICPC=MICPC+1
      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

FUDGE: BRWRT IBUS,NPR ;READ NPR CONTROL
        MICPC=MICPC+1
        <MOVE!WRTEBR!IBUS!<NPR>>
        BRO IDLE ;IF NPR GOING---LEAVE
        MICPC=MICPC+1
        <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        BRWRT BR!LDMAR,SELA!SP4 ;LOAD THE MAR
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>
        BR7 BS2 ;IF SET - READ BACK ALL 200
        MICPC=MICPC+1
        <JUMP!BR7CON!<BS2-INIT&3000*4>!<BS2-INIT&777/2>>
        MEMINC IBUS,INDAT1 ;OTHERWISE RESTORE TWO BYTES
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IBUS!<INDAT1>>
        MEMINC IBUS,INDAT2 ;...
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IBUS!<INDAT2>>
        BRWRT IMM,2 ;UPDATE---UNIBUS ADDRESS
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<2 >
        SP BR,ADD,SP4 ;UPDATE NPR COUNTER
        MICPC=MICPC+1
        <MOVE!SPX!BR!ADD!SP4>
        SP IBUS,IIBA1,SPC ;UPDATE ADDRESS LOW
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!IIBA1!SF 1>
        OUTPUT BR,ADD!IBA1
        MICPC=MICPC+1
        <MOVE!WROUT!BR!<ADD!IBA1>
        SP IBUS,IIBA2,SPO ;READ HIGH ADDRESS
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!IIBA2!SPO>
        OUTPUT BR,AC!IBA2 ;UPDATE HIGH
        MICPC=MICPC+1
        <MOVE!WROUT!BR!<AC!IBA2>>
        SP IBUS,NPR,SPO ;READ NPR REGISTER
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!NPR!SPO>
        C RESEXT ;IF CARRY---UPDATE MXT
        MICPC=MICPC+1
        <JUMP!CCOND!<RESEXT-INIT&3000*4>!<RESEXT-INIT&777/2>>
        ALWAYS TH3X ;GO DO ANOTHER NPR
        MICPC=MICPC+1
        <JUMP!ALCOND!<TH3X-INIT&3000*4>!<TH3X-INIT&777/2 >

```

DDCGAH.MAC 14-DEC-77 15:07

SNDACK--ROUTINE TO SEND AN ACK

```

1644 021242 001350
(1) 021242 000774
1645 021244 001351
(1) 021244 063270
1646 021246 001352
(1) 021246 063233
1647
1648 021250 001353
(1) 021250 010067
1649 021252 001354
(1) 021252 043220
1654 021254 001355
(1) 021254 000400
(1) 021256 001356
(1) 021256 063222
1655 021260 001357
(1) 021260 114514
1656
1657 021262
1659 021262 001360
(1) 021262 023120
1660 021264 001361
(1) 021264 062065
1661 021266 001362
(1) 021266 111364
1662 021270 001363
(1) 021270 110564
1663 021272
1665 021272 001364
(1) 021272 123200
1666 021274 001365
(1) 021274 000404
1667 021276 001366
(1) 021276 061010
1668 021300 001367
(1) 021300 110564
1669
1670 021302 001370
(1) 021302 010016
1671 021304

```

```

BTEOM: BRWRT IMM,374 ;MASK FOR CLEAR MSG PENDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<374>>
SP BR AANDB,SP10 ;TURN THEM OFF IN LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SP10>
SP BR SELB,SP13 ;STORE UNRECOGNIZABLE VALUE INTO SP13
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP13>
;SO "RH3" WILL EXIT BACK TO IDLE LOOP
LDMA IMM,STC ;ADDRESS START OF TMT CHAIN
MICPC=MICPC+1
<MOVE!LDMA!IMM!<STC&377>>
SP MEMX,SELB,SPO ;COPY LINK ADDRESS
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SPO>
TSTATE TMTA ;CHANGE XMIT STATE TO LINE IS IDLE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTA-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPY!BR!SELB!SP2>
ALWAYS TDON2 ;POST A DONE
MICPC=MICPC+1
<JUMP!ALCOND!<TDON2-INIT&3000*4>!<TDON2-INIT&777/2>>
;INCREMENT MXT BITS
HOINCH: SP IBUS,IIBA2,SPO ;GET HIGH BYTE
MICPC=MICPC+1
<MOVE!SPX!IBUS!IIBA2!SPO>
OUTPUT BR <INCA!IBA2> ;INCREMENT AND OUTPUT
MICPC=MICPC+1
<MOVE!WROUT!BR!<INCA!IBA2>>
C 10$ ;OVERFLOW-GO INCREMENT MXT BITS
MICPC=MICPC+1
<JUMP!CCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
ALWAYS TH8 ;OTHERWISE RETURN
MICPC=MICPC+1
<JUMP!ALCOND!<TH8-INIT&3000*4>!<TH8-INIT&777/2>>
10$: SP IBUS,NPR,SPO ;READ NPR REG IWTH CURRENT MXT BITS
MICPC=MICPC+1
<MOVE!SPX!IBUS!NPR!SPO>
BRWRT IMM,4 ;WRITE BIT TO ADD
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<4>>
OUT BR <ADD!ONPR> ;TURN ON PROPER MXT BITS
MICPC=MICPC+1
<MOVE!WROUTX!BR!<ADD!ONPR>>
ALWAYS TH8
MICPC=MICPC+1
<JUMP!ALCOND!<TH8-INIT&3000*4>!<TH8-INIT&777/2>>
SBTTL REP HANDLER
LDMA IMM,REPCR ;LOAD MAP ADDRESS WITH POINTER TO REPS RECC
MICPC=MICPC+1
<MOVE!LDMA!IMM!<REPCR&377>>
SP MEMX,SELB,SPO ;READ NUMBER OF REPS RECC

```

JDCGAH.MAC 14-DEC-77 15:07

REP HANDLER

```

(1) 001371
(1) 021304 043220
1672 021306
(1) 001372
(1) 021306 062460
1673 021310
(1) 001373
(1) 021310 010151
1674 021312
(1) 001374
(1) 021312 015402
1675 021314
(1) 001375
(1) 021314 016703
1676 021316
(1) 001376
(1) 021316 114660
1677
1678 021320
(1) 001377
(1) 021320 117001
1679 021322
(1) 001400
(1) 021322 100447
1680 021324
(1) 001401
(1) 021324 000727
1681 021326
(1) 001402
(1) 021326 063270
1682 021330
(1) 001403
(1) 021330 110723
    
```

```

MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SP0>
MEM DP <INCA!SPO> ; INCREMNT REPS RECD
MICPC=MICPC+1
<MOVE!WRMEM!DP!<INCA!SPO>>
LDMA IMM,T ; LOAD ADDRESS OF TYPE FIELD
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<T&377>> ; LOAD NAK TYPE
MEMINC IMM,2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<2>> ; LOAD REP RESPONSE SUB-TYPE
MEMINC IMM,303
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<303>> ; SEND AN UNNUMB MSG
ALWAYS SNAK
MICPC=MICPC+1
<JUMP!ALCOND!<SNAK-INIT&3000*4>!<SNAK-INIT&777/2>>
; IF START MODE ACCEPT STACK
STACK: BR4 S$
MICPC=MICPC+1
<JUMP!BR4CON!<S$-INIT&3000*4>!<S$-INIT&777/2>>
ALWAYS IDLE ; IGNORE STACK
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
S$: BRWTE IMM,327 ; MASK TO CLEAR START
MICPC=MICPC+1
<MOVE!WTEBR!IMM!<327>>
SP BR AANDB,SP10 ; CLEAR START MODE
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SP10>
ALWAYS SNDACK ; RESET TIMER AND IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<SNDACK-INIT&3000*4>!<SNDACK-INIT&777/2>>
    
```

DCCGAM.MAC 14-DEC-77 15:07

REP HANDLER

```

1684 021332
1686 021332
(1) 001404
(1) 021332 023160
1687 021334
(1) 001405
(1) 021334 062067
1688 021336
(1) 001406
(1) 021336 115010
1689 021340
(1) 001407
(1) 021340 104661
1690 021342
1692 021342
(1) 001410
(1) 021342 123220
1693 021344
(1) 001411
(1) 021344 000515
1694 021346
(1) 001412
(1) 021346 063260
1695 021350
(1) 001413
(1) 021350 000404
1696 021352
(1) 001414
(1) 021352 061011
1697 021354
(1) 001415
(1) 021354 104661
1702 021356
(1) 001416
(1) 021356 010011
1703 021360
(1) 001417
(1) 021360 043220
1704 021362
(1) 001420
(1) 021362 042460
1705 021364
(1) 001421
(1) 021364 060617
1706 021366
(1) 001422
(1) 021366 063232
1707 021370
(1) 001423
(1) 021370 000406
1708
1709 021372
(1) 001424
(1) 021372 010067
1710 021374
(1) 001425
    
```

```

ICBA23:
SP      IBUS, IOBA2, SPO          :GET HIGH BYTE
MICPC=MICPC+1
<MOVE!SPX!IBUS!IOBA2!SPO>
OUTPUT DP <INCA!OBA2>          ;INCREMENT IT
MICPC=MICPC+1
<MOVE!WROUT!DP!<INCA!OBA2>>
C      10$                        ;OVERFLOW - INCREMENT MXT BITS
MICPC=MICPC+1
<JUMP!CCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
ALWAYS RK3
MICPC=MICPC+1
<JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>

10$:
SP      IBUS, UBBR, SPO
MICPC=MICPC+1
<MOVE!SPX!IBUS!UBBR!SPO>
BRWRT  IMM, 115
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<115>>
SP      BR, AANDB, SPO
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SPO>
BRWRT  IMM, 4
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<4>>
OUT    BR <ADD!OBR>
MICPC=MICPC+1
<MOVE!WROUTX!BR!<ADD!OBR>>
ALWAYS RK3
MICPC=MICPC+1
<JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>

NAK:
LDMA   IMM, NDATA                ;CUMMULATIVE NAK COUNTER
MICPC=MICPC+1
<MOVE!LDMA!IMM!<NDATA&377>>
SP      MEMX, SELB, SPO          ;READ IT
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SPO>
MEM     MEMX, INCA!SPO          ;INCREMENT THE COUNTER
MICPC=MICPC+1
<MOVE!WRMEM!MEMX!<INCA!SPO>>
BRWRT  BR, SELA!SP17            ;GETLASTMESSAGE ACKED
MICPC=MICPC+1
<MOVE!WRTBR!BR!<SELA!SP17>>
SP      BR, SELB, SP12          ;COPY TO CURRENT NUMBER
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP12>
BRWRT  IMM, 6                    ;WRITE NUMBERED MSG PENDING
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<6>>
;AND LINE HAS GONE IDLE
;ADDRESS START OF XMIT CHAIN

NAK1:
LDMA   IMM, STC
MICPC=MICPC+1
<MOVE!LDMA!IMM!<STC&377>>
SP      MEMX!LDMA, SELB, SP16   ;COPY START OF CHAIN TO LAST XMIT
MICPC=MICPC+1
    
```

DDCGAH.MAC 14-DEC-77 15:07

REP HANDLER

```

(1) 021374 053236      <MOVE!SPX!MEMX!LOMAR!SELB!SP16>
1711 021376 001426      SP      BR,AORB,SP10      ;SET IT IN LINE STATUS WORD
(1) 021376 063310      MICPC=MICPC+1
1712 021400 001427      <MOVE!SPX!BR!AORB!SP10>
(1) 021400 003374      SP      IMM,374,SP14      ;RESET TIMER COUNT
1713 021402 001430      MICPC=MICPC+1
(1) 021402 113713      <MOVE!SPX!IMM!374!SP14>
1714 021404 001431      BR7     TEOM1
(1) 021404 040620      MICPC=MICPC+1
1715 021406 001432      <JUMP!BR7CON!<TEOM1-INIT&3000*4>!<TEOM1-INIT&777/2>>
(1) 021406 102051      BRWRTE  MEMX,SELB
1716 021410 001433      MICPC=MICPC+1
(1) 021410 110716      <MOVE!WRTEBR!MEMX!<SELB>>
1717 021412 001434      BRO     I1
(1) 021412 000600      MICPC=MICPC+1
1718 021414 001435      <JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
1719 021416 001436      ALWAYS  TEOM3
(1) 021416 061300      MICPC=MICPC+1
1720 021420 001440      <JUMP!ALCOND!<TEOM3-INIT&3000*4>!<TEOM3-INIT&777/2>>
(1) 021420 100447      INWAT1: BRWRTE  IMM,200      ;MASK FOR RDI
1721 021422 002502      <MOVE!WRTEBR!IMM!<200>>
1722 021422 100447      SP      IBUS,INCON,SP0      ;READ INPUT CONTROL CSR
(1) 021422 001440      MICPC=MICPC+1
(1) 021422 100447      <MOVE!SPX!IBUS!INCON!SP0>
(1) 021422 100447      OUT     BR,AORB!OINCON      ;SET RDI
(1) 021422 100447      MICPC=MICPC+1
(1) 021422 100447      <MOVE!WROUTX!BR!<AORB!OINCON>>
(1) 021422 100447      PSTATE IEIWAT      ;CHANGE STATE TO WAIT FOR IEI OR RDI
(1) 021422 100447      MEM     IMM,<<IEIWAT-INIT&777/2>>
(1) 021422 100447      MICPC=MICPC+1
(1) 021422 100447      <MOVE!WRMEM!IMM!<<IEIWAT-INIT&777/2>>>
(1) 021422 100447      ALWAYS  IDLE
(1) 021422 100447      MICPC=MICPC+1
(1) 021422 100447      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1722 021422 100447      :

```

E01

DDCGAH.MAC 14-DEC-77 15:07

REP HANDLER

1732
1733 021424 001441
(1) 021424 070075
1734 021426 001442
(1) 021426 060601
1735 021430 001443
(1) 021430 116455
1736 021432 001444
(1) 021432 014477
1737 021434 001445
(1) 021434 063265
1738 021436 001446
(1) 021436 077220
1739 021440 001447
(1) 021440 054660
1740 021442 001450
(1) 021442 060365
1741 021444 001451
(1) 021444 115461
1742 021446 001452
(1) 021446 115063
1743 021450 001453
(1) 021450 020540
1744 021452 001454
(1) 021452 116057
1745 021454 001455
(1) 021454 000676
1746 021456 001456
(1) 021456 104425
1747 021460 001457
(1) 021460 000620
1748 021462 001460
(1) 021462 104425
1749
1750 021464 001461
(1) 021464 040364
1751 021466 001462
(1) 021466 114452

```

;FUGITIVE RECEIVE ROUTINES---DON'T FIT IN PAGE
RH1:  LDMA  BR,<INCA!SP15>          ;LD MAR WITH ADDR OF BA
      MICPC=MICPC+1
      <MOVE!LDMAR!BR!<INCA!SP15>>
      BRWRT  BR,SELA!SP1          ;READ STATUS BYTE
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
      BR1    RH4                  ;NO BUFF ASSIGNED IN MAINT MODE
      MICPC=MICPC+1
      <JUMP!BR1CON!<RH4-INIT&3000*4>!<RH4-INIT&777/2>>
      BRWRT  IMM!INCMAR,77
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!INCMAR!<77>>
      SP     BR,AANDB,SPS
      MICPC=MICPC+1
      <MOVE!SPX!BR!AANDB!SPS>
      SP     BR!INCMAR,SELB,SPO    ;SAVE BYTE COUNT MASK
      MICPC=MICPC+1
      <MOVE!SPX!BR!INCMAR!SELB!SPO>
      BRWRT  MEMX!INCMAR,AANDB!SPO ;GET HIGH BYTE COUNT BITS
      MICPC=MICPC+1
      <MOVE!WRTEBR!MEMX!INCMAR!<AANDB!SPO>>
      CMP    BR,SPS                ;COMPARE HIGH ORDER BITS OF COUNT
      MICPC=MICPC+1
      <SUBTC!BR!SPS>
      Z      RCLW                  ;IF EQUAL COMPARE LOW ORDER BITS OF COUNT
      MICPC=MICPC+1
      <JUMP!ZCOND!<RCLW-INIT&3000*4>!<RCLW-INIT&777/2>>
RH6:  C      RCFATL                ;IF CARRY--TOO BIG ERROR
      MICPC=MICPC+1
      <JUMP!CCOND!<RCFATL-INIT&3000*4>!<RCFATL-INIT&777/2>>
RH2:  BRWRT  IBUS,I0BA1            ;READ LOW BYTE OF IN BA
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<I0BA1>>
      BRO    RCVODD                ;IF SET IS ODD TRANSFER
      MICPC=MICPC+1
      <JUMP!BROCON!<RCVODD-INIT&3000*4>!<RCVODD-INIT&777/2>>
RH4:  STATE  RCVKED
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVKED-INIT&777/2>>
      ALWAYS REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
RCVODD: STATE  RCVK01
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVK01-INIT&777/2>>
      ALWAYS REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
RCLW: CMP    MEMX,SP4              ;COMPARE LOW ORDER BITS OF COUNT
      MICPC=MICPC+1
      <SUBTC!MEMX!SP4>
      ALWAYS RH6
      MICPC=MICPC+1
      <JUMP!ALCOND!<RH6-INIT&3000*4>!<RH6-INIT&777/2>>

```


REP HANDLER

```

1752 021470 001463
(1) 021470 010151
1753 021472 001464
(1) 021472 016402
1754 021474 001465
(1) 021474 002711
1755 021476 001466
(1) 021476 010175
1756 021500 001467
(1) 021500 036540
1757 021502 001470
(1) 021502 036560
1758 021504 001471
(1) 021504 000420
1759 021506 001472
(1) 021506 016400
1760 021510 001473
(1) 021510 062620
1761 021512 001474
(1) 021512 002212
1762 021514 001475
(1) 021514 000404
(1) 021514 001476
(1) 021516 063222
1763 021520 001477
(1) 021520 003001
1764 021522 001500
(1) 021522 114642
1765
1766
1767 021524 001501
(1) 021524 060610
1768 021526 001502
(1) 021526 001620
1769 021530 001503
(1) 021530 117106
1770
1771 021532 001504
(1) 021532 000600
    
```

```

RCFATL: LDMA IMM,T
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<T&377>>
MEMINC IMM,2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<2>>
MEM IMM,311
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<311>>
LDMA IMM,<RTHRS+1> ;ADDRESS ERROR LINK
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<<RTHRS+1>&377>>
MEMINC IBUS,IOBA1
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<IOBA1>>
MEMINC IBUS,IOBA2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<IOBA2>>
BRWRT IMM,20
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<20>>
RCEXY: MEMINC IMM,0
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<0>>
MEM BR,SELB
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELB>>
RCEXX: OUTPUT IMM,<200!ORCVCO> ;FLUSH INPUT SILO
MICPC=MICPC+1
<MOVE!WROUT!IMM!<200!ORCVCO>>
TSTATE TMTA6
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTA6-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
SP IMM,1,SP1 ;SET INIT MODE IN PORT STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!IMM!1!SP1>
ALWAYS NTRS1
MICPC=MICPC+1
<JUMP!ALCOND!<NTRS1-INIT&3000*4>!<NTRS1-INIT&777/2>>
;
START: .SBTTL START HANDLER
BRWRT DP,<SELA!SP10> ;READ LINE STATUS WORD
MICPC=MICPC+1
<MOVE!WRTEBR!DP!<SELA!SP10>>
BRSHFT ;GET START MODE BIT IN POSITION
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR4 10$ ;IF IN START MODE SET STACK
MICPC=MICPC+1
<JUMP!BR4CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
;ELSE SET UP START ERROR
BRWRT IMM,200
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<200>
    
```

GO1

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

PAGE: 02130M
SEQ 0213

1772 021534 001505
(1)
(1) 021534 104746
1773 021536 001506
(1)
(1) 021536 010151
1774 021540 001507
(1)
(1) 021540 016407
1775 021542 001510
(1)
(1) 021542 000411
1776 021544 001511
(1)
(1) 021544 110726

```
10$: ALWAYS RCV2
MICPC=MICPC+1
<JUMP!ALCOND!<RCV2-INIT&3000*4>!<RCV2-INIT&777'2>>
LDMA IMM T ;SET UP ADDR OF TYPE FIELD
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<T&377>>
MEMINC IMM 7 ;WRITE STACK TYPE
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<7>>
BRWTE IMM 11 ;SEND THE UNNUMB MESS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<11>>
ALWAYS SA2
MICPC=MICPC+1
<JUMP!ALCOND!<SA2-INIT&3000*4>!<SA2-INIT&777'2>>
```

H01

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

1778	021546	001512	TDON3:	BRWRTE MEMX, SUB!SP17	;COMPARE RESPONSE TO MSG NO
(1)				MICPC=MICPC+1	
(1)	021546	040757		<MOVE!WRTEBR!MEMX!<SUB!SP17>>	
1779	021550			BR7 RH3	;IF NEGATIVE EXIT
(1)		001513		MICPC=MICPC+1	
(1)	021550	107573		<JUMP!BR7CON!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>	
1780	021552		TDON2:	LDMA BR, SELA!SPO	;ADDRESS THE TRANSMIT LINK
(1)		001514		MICPC=MICPC+1	
(1)	021552	070200		<MOVE!LDMAR!BR!<SELA!SPO>>	
1781	021554			MEM IMM, 0	;TURN OF ASSIGNED AND TMTED BITS IN FLAG
(1)		001515		MICPC=MICPC+1	
(1)	021554	002400		<MOVE!WRMEM!IMM!<0>>	
1782	021556			LDMA IMM, STC	
(1)		001516		MICPC=MICPC+1	
(1)	021556	010067		<MOVE!LDMAR!IMM!<STC&377>>	
1783	021560			MEM IMM, TML1	;ASSUME WRAPAROUND
(1)		001517		MICPC=MICPC+1	
(1)	021560	002471		<MOVE!WRMEM!IMM!<TML1>>	
1784	021562			BRWRTE IMM, TMLB	;WRAPAROUND?
(1)		001520		MICPC=MICPC+1	
(1)	021562	000543		<MOVE!WRTEBR!IMM!<TMLB>>	
1785	021564			CMP BR, SPO	
(1)		001521		MICPC=MICPC+1	
(1)	021564	060360		<SUBTC!BR!SPO>	
1786	021566			Z TDON4	;YES
(1)		001522		MICPC=MICPC+1	
(1)	021566	115525		<JUMP!ZCOND!<TDON4-INIT&3000*4>!<TDON4-INIT&777/2>>	
1787	021570			BRWRTE IMM, 6	;OFFSET FOR NEXT TMT LINK
(1)		001523		MICPC=MICPC+1	
(1)	021570	000406		<MOVE!WRTEBR!IMM!<6>>	
1788	021572			MEM BR, ADD!SPO	;UPDATE THE POINTER
(1)		001524		MICPC=MICPC+1	
(1)	021572	062400		<MOVE!WRMEM!BR!<ADD!SPO>>	
1789	021574		TDON4:	LDMA IMM, NXTSP	;ADDRESS DONE LINK
(1)		001525		MICPC=MICPC+1	
(1)	021574	010241		<MOVE!LDMAR!IMM!<NXTSP&377>>	
1790	021576			LDMA MEMX, SELB!SPX!SP3	;ADDRESS THE LINK, COPYING
(1)		001526		MICPC=MICPC+1	
(1)	021576	053223		<MOVE!LDMAR!MEMX!<SELB!SPX!SP3>>	
1791					;ITS ADDRESS TO SPO
1792	021600			MEMINC IMM, 200	;WRITE THE INTERRUPT TYPE
(1)		001527		MICPC=MICPC+1	
(1)	021600	016600		<MOVE!WRMEM!INCMAR!IMM!<200>>	
1793	021602			MEM BR, INCA!SPO	;COPY ACTUAL LINK ADDRESS
(1)		001530		MICPC=MICPC+1	
(1)	021602	062460		<MOVE!WRMEM!BR!<INCA!SPO>>	
1794	021604			LDMA IMM, NXTSP	;ADDRESS PTR INT STACK
(1)		001531		MICPC=MICPC+1	
(1)	021604	010241		<MOVE!LDMAR!IMM!<NXTSP&377>>	
1795	021606			MEM IMM, INTSTK	;ASSUME WRAP AROUND
(1)		001532		MICPC=MICPC+1	
(1)	021606	002642		<MOVE!WRMEM!IMM!<INTSTK>>	
1796	021610			BRWRTE IMM, <<MMEND-2>>	;ADDRESS ENDOFINT STACK
(1)		001533		MICPC=MICPC+1	
(1)	021610	000776		<MOVE!WRTEBR!IMM!<<MMEND-2>>>	
1797	021612			CMP BR, SP3	;WRAPAROUND?

DOCGAM.MAC 14-DEC-77 15:07

START HANDLER

```

(1) 001534
(1) 021612 060363
1798 021614
(1) 001535
(1) 021614 115540
1799 021616
(1) 001536
(1) 021616 000402
1800 021620
(1) 001537
(1) 021620 062403
1801 021622
(1) 001540
(1) 021622 000420
1802 021624
(1) 001541
(1) 021624 063301
1803
1804 021626
(1) 001542
(1) 021626 010153
1805 021630
(1) 001543
(1) 021630 043237
1806 021632
(1) 001544
(1) 021632 010067
1807 021634
(1) 001545
(1) 021634 053620
1808 021636
(1) 001546
(1) 021636 054620
1809 021640
(1) 001547
(1) 021640 116512
1810 021642
(1) 001550
(1) 021642 104573
1811
1812
1813 021644
(1) 001551
(1) 021644 000404
1814 021646
(1) 001552
(1) 021646 063000
1815 021650
(1) 001553
(1) 021650 110574

MICPC=MICPC+1
<SUBTC!BR!SP3>
Z TDON40 ;YES---BRANCH
MICPC=MICPC+1
<JUMP!ZCOND!<TDON40-INIT&3000*4>!<TDON40-INIT&777/2>>
BRWRT IMM,2 ;OFFSET TO NEXT PAIR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>>
MEM BR,ADD!SP3 ;UPDATE POINTER
MICPC=MICPC+1
<MOVE!WRMEM!BR!<ADD!SP3>>
TDON40: BRWRT IMM,20 ;WRITE INTERUPT PENDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<20>>
SP BR,AORB,SP1 ;IN PORT STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP1>

TDON1: LDMA IMM,ISP17 ;GET LAST ACKED
MICPC=MICPC+1
<MOVE!LDMA!IMM!<ISP17&377>>
SP MEMX,SELB,SP17 ;STORE IT IN SP17
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SP17>
LDMA IMM,STC ;GET START OF TMT CHAIN
MICPC=MICPC+1
<MOVE!LDMA!IMM!<STC&377>>
LDMA MEMX,SELB!SPBRX!SPO ;ADDRESS THE LINK
MICPC=MICPC+1
<MOVE!LDMA!MEMX!<SELB!SPBRX!SPO>>
BRWRT MEMX!INCMAR,SELB ;GET THE FLAGS
MICPC=MICPC+1
<MOVE!WRTEBR!MEMX!INCMAR!<SELB>>
BR1 TDON3 ;IF BUFFER ASSIGNED PROCEED
MICPC=MICPC+1
<JUMP!BR1CON!<TDON3-INIT&3000*4>!<TDON3-INIT&777/2>>
ALWAYS RH3 ;ELSE---EXIT
MICPC=MICPC+1
<JUMP!ALCOND!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>

RESEXT: BRWRT IMM,4 ;ADD TO MXT BITS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<4>>
SP BR,ADD,SPO
MICPC=MICPC+1
<MOVE!SPX!BR!ADD!SPO>
ALWAYS TH3X
MICPC=MICPC+1
<JUMP!ALCOND!<TH3X-INIT&3000*4>!<TH3X-INIT&777/2>>

```

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

1817
 1818
 1819
 1820
 1821 021652 001554
 (1) 021652 023333
 1822 021654 001555
 (1) 021654 115561
 1823 021656 001556
 (1) 021656 000406
 1824 021660 001557
 (1) 021660 060360
 1825 021662 001560
 (1) 021662 115566
 1826 021664 001561
 (1) 021664 060601
 1827 021666 001562
 (1) 021666 106753
 1828 021670 001563
 (1) 021670 060521
 1829 021672 001564
 (1) 021672 107345
 1830 021674 001565
 (1) 021674 104661
 1831 021676 001566
 (1) 021676 063164
 1832 021700 001567
 (1) 021700 000750
 1833 021702 001570
 (1) 021702 104425
 1834
 1835
 1836
 1837 021704 001571
 (1) 021704 010001
 1838 021706 001572
 (1) 021706 002401
 1839 021710 001573
 (1) 021710 010167
 1840 021712

```

: INPUTS:
: SPO = RECEIVE CHARACTER
PASWRD: SP      IBUS, LNOSW, SP13      :READ PASSWD SWITCH
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!LNOSW!SP13>
        Z      10$                      :IF ALL ONES NO RLD ENABLED
        MICPC=MICPC+1
        <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
        BRWRT  IMM,6                      :CHECK FOR ENTER MOP MODE
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<6>>
        CMP    BR, SPO
        MICPC=MICPC+1
        <SUBTC!BR!SPO>
        Z      20$                      :IF EQUAL===ENTER MOP
        MICPC=MICPC+1
        <JUMP!ZCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
10$:    BRWRT  BR, SELA!SP1                :READ STATUS BYTE
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<SELA!SP1>>
        BR1    RHX                          :MESSAGE WITH NO BUFFER ASSIGNED
        MICPC=MICPC+1
        <JUMP!BR1CON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
        BRWRT  BR, AA!SP1
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<AA!SP1>>
        BR4    RCVMO                          :DLE RECEIVED IN NORMAL MODE
        MICPC=MICPC+1
        <JUMP!BR4CON!<RCVMO-INIT&3000*4>!<RCVMO-INIT&777/2>>
        ALWAYS RK3                          :HANDLE MAINT MODE MESSAGE
        MICPC=MICPC+1
        <JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>
20$:    SP      BR, DECA, SP4                :COUNT FOR NUMB OF COMPARES
        MICPC=MICPC+1
        <MOVE!SPX!BR!DECA!SP4>
        STATE  EM2
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<EM2-INIT&777/2>>
        ALWAYS REXIT
        MICPC=MICPC+1
        <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
:
: .ENABL  LSB
:
RCVM1: LDMA    IMM, NAKST                    :RESET NAKS SENT
        MICPC=MICPC+1
        <MOVE!LDMAR!IMM!<NAKST&377>>
        MEM    IMM, 1                        ;...
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<1>>
        LDMA    IMM, BC                      :ADDRESS ORIGINAL RECV BYTE COUNT
        MICPC=MICPC+1
        <MOVE!LDMAR!IMM!<BC&377>>
        SP      MEMX!INCMAR, SELB, SP4        :MOVE BYTE COUNT TO SP4

```

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

```

(1)          001574          MICPC=MICPC+1
(1) 021712 057224          <MOVE!SPX!MEMX!INCMAR!SELB!SP4>
1841 021714          SP      MEMX!INCMAR,SELB,SP5      ;AND SP5
(1)          001575          MICPC=MICPC+1
(1) 021714 057225          <MOVE!SPX!MEMX!INCMAR!SELB!SP5>
1842 021716          MEM      BR,DECA!SP11          ;COPY SP11 TO MEMORY
(1)          001576          MICPC=MICPC+1
(1) 021716 062571          <MOVE!WRMEM!BR!<DECA!SP11
1843 021720          LDMA     IMM,NXTSP
(1)          001577          MICPC=MICPC+1
(1) 021720 010241          <MOVE!LDMAR!IMM!<NXTSP&377>>
1844 021722          SP      MEMX!LDMAR,SELB,SP3          ;COPY TO SP3
(1)          001600          MICPC=MICPC+1
(1) 021722 053223          <MOVE!SPX!MEMX!LDMAR!SELB!SP3>
1845 021724          MEMINC  IMM,204          ;RECEIVE DONE IMAGE
(1)          001601          MICPC=MICPC+1
(1) 021724 016604          <MOVE!WRMEM!INCMAR!IMM!<204>>
1846 021726          MEM      BR!LDMAR,SELA!SP15          ;COPY LINK ADDRESS TO NEXT INTER
(1)          001602          MICPC=MICPC+1
(1) 021726 072615          <MOVE!WRMEM!BR!LDMAR!<SELA!SP15>>
1847 021730          MEMINC  IMM,0          ;ZERO THE FLAGS
(1)          001603          MICPC=MICPC+1
(1) 021730 016400          <MOVE!WRMEM!INCMAR!IMM!<0>>
1848 021732          SP      IMM!INCMAR,SPO,300          ;WRITE A 300 TO SPO
(1)          001604          MICPC=MICPC+1
(1) 021732 017300          <MOVE!SPX!IMM!INCMAR!SPO!300>
1849 021734          BRWRITE IMM!INCMAR,2          ;PREPARE TO ADDRESS NEXT
(1)          001605          MICPC=MICPC+1
(1) 021734 014402          <MOVE!WRITEBR!IMM!INCMAR!<2>>
1850          ;INTERRUPT STACK AND INCREMENT
1851          ;THE MAR
1852 021736          MEM      MEMX,AANDB!SPO          ;MASK OFF ORIGINAL HIGH BYTE
(1)          001606          MICPC=MICPC+1
(1) 021736 042660          <MOVE!WRMEM!MEMX!<AANDB!SPO>>
1853          ;OF COUNT SAVING EXTENDED MEM BITS
1854 021740          MEMINC  MEMX,AORB!SP5          ;COPY TO MEMORY LINK
(1)          001607          MICPC=MICPC+1
(1) 021740 056705          <MOVE!WRMEM!INCMAR!MEMX!<AORB!SP5>>
1855 021742          MEMINC  BR,SELA!SP4
(1)          001610          MICPC=MICPC+1
(1) 021742 076604          <MOVE!WRMEM!INCMAR!BR!<SELA!SP4>>
1856 021744          LDMA     IMM,NXTSP          ;ADDRESS NEXT INT STACK
(1)          001611          MICPC=MICPC+1
(1) 021744 010241          <MOVE!LDMAR!IMM!<NXTSP&377>>
1857 021746          MEM      BR,ADD!SP3
(1)          001612          MICPC=MICPC+1
(1) 021746 062403          <MOVE!WRMEM!BR!<ADD!SP3>>
1858 021750          BRWRITE IMM!<<MMEND-2>>          ;ADDRESSEND OF INT STACK
(1)          001613          MICPC=MICPC+1
(1) 021750 000776          <MOVE!WRITEBR!IMM!<<MMEND-2>>>
1859 021752          CMP      BR,SP3          ;WRAP AROUND
(1)          001614          MICPC=MICPC+1
(1) 021752 060363          <SUBTC!BR!SP3>
1860 021754          Z      40$          ;IF YES-- BRANCH
(1)          001615          MICPC=MICPC+1
(1) 021754 115626          <JUMP!ZCOND!<40$-INIT&3000*4>!<40$-INIT&777'2>>

```

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

```

1861 021756
1862 021756
(1) 001616
(1) 021756 000462
1863 021760
(1) 001617
(1) 021760 060375
1864 021762
(1) 001620
(1) 021762 115630
1865 021764
(1) 001621
(1) 021764 000405
1866 021766
(1) 001622
(1) 021766 063015
1867 021770
(1) 001623
(1) 021770 000420
1868 021772
(1) 001624
(1) 021772 063301
1877 021774
(1) 001625
(1) 021774 104415
1879 021776
(1) 001626
(1) 021776 002642
1880 022000
(1) 001627
(1) 022000 114616
1881 022002
(1) 001630
(1) 022002 000742
1882 022004
(1) 001631
(1) 022004 114622
1883
1884 022006
(1) 001632
(1) 022006 010152
1885 022010
(1) 001633
(1) 022010 043620
1886 022012
(1) 001634
(1) 022012 060400
1887 022014
(1) 001635
(1) 022014 103367
1888 022016
(1) 001636
(1) 022016 000401
1889 022020
1890 022020
(1) 001637
    
```

```

20$: BRWRT IMM,RCL7
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCL7>>
      CMP BR,SP15
      MICPC=MICPC+1
      <SUBTC!BR!SF15>
      Z 50$
      MICPC=MICPC+1
      <JUMP!ZCOND!<50$-INIT&3000*4>!<50$-INIT&777/2>>
      BRWRT IMM,5
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<5>>
30$: SP BR,ADD,SP15
      MICPC=MICPC+1
      <MOVE!SPX!BR!ADD!SP15
      BRWRT IMM,20 ;MASK FOR INTERUPT PENDING
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<20>>
      SP DP,AORB,SP1 ;UPDATE PORT STATUS WORD
      MICPC=MICPC+1
      <MOVE!SF!DP!AORB!SP1>
      ALWAYS FLUSH
      MICPC=MICPC+1
      <JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
40$: MEM IMM,INTSTK ;POINT TO START OF INTERRUPT STACK
      MICPC=MICPC+1
      <MOVE!WRMEM!IMM!<INTSTK>>
      ALWAYS 20$
      MICPC=MICPC+1
      <JUMP!ALCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
50$: BRWRT IMM,<<RCL1-RCL7>&377> ;POINT TO START OF QUEUE
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<<RCL1-RCL7>&377>>
      ALWAYS 30$
      MICPC=MICPC+1
      <JUMP!ALCOND!<30$-INIT&3000*4>!<30$-INIT&777/2>>
      .DSABL LSB
NTHRS: LDMA IMM,ST
      MICPC=MICPC+1
      <MOVE!LDMA!IMM!<ST&377>>
      SPBR MEMX,SELB,SPO
      MICPC=MICPC+1
      <MOVE!SPBRX!MEMX!SELB!SPO>
      BRWRT BR,ADD!SPO ;SHIFT LEFT
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<ADD!SPO>>
      BR4 OVRUN
      MICPC=MICPC+1
      <JUMP!BR4CON!<OVRUN-INIT&3000*4>!<OVRUN-INIT&777/2>>
      BRWRT IMM,1
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<1>>
ERRXX:
NTRSD: LDMA IMM,<<RTHRS+3>>
      MICPC=MICPC+1
    
```

MO1

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

(1)	022020	010177	<MOVE!LDMAR!IMM!<<RTHRS+3>&377>>	
1891	022022		MEMINC IMM,0	
(1)		001640	MICPC=MICPC+1	
(1)	022022	016400	<MOVE!WRMEM!INCMAR!IMM!<0>>	
1892	022024		MEM BR SELB	
(1)		001641	MICPC=MICPC+1	
(1)	022024	062620	<MOVE!WRMEM!BR!<SELB>>	
1893	022026		NTRS1: LDMA IMM NXTSP	
(1)		001642	MICPC=MICPC+1	
(1)	022026	010241	<MOVE!LDMAR!IMM!<NXTSP&377>>	
1894	022030		LDMA MEMX SELB!SPX!SPO	
(1)		001643	MICPC=MICPC+1	
(1)	022030	053270	<MOVE!LDMAR!MEMX!<SELB!SPX!SPO>>	
1895	022032		MEMINC IMM,201	
(1)		001644	MICPC=MICPC+1	
(1)	022032	016601	<MOVE!WRMEM!INCMAR!IMM!<201>>	
1896	022034		MEM IMM <<RTHRS>>	
(1)		001645	MICPC=MICPC+1	
(1)	022034	002574	<MOVE!WRMEM!IMM!<<RTHRS>>>	
1897	022036		LDMA IMM NXTSP	
(1)		001646	MICPC=MICPC+1	
(1)	022036	010241	<MOVE!LDMAR!IMM!<NXTSP&377>>	
1898	022040		MEM IMM INTSTK	;ASSUME QUEUE WRAP AROUND
(1)		001647	MICPC=MICPC+1	
(1)	022040	002642	<MOVE!WRMEM!IMM!<INTSTK>>	
1899	022042		BRWRTE IMM,<<MMEND-2>>	
(1)		001650	MICPC=MICPC+1	
(1)	022042	000776	<MOVE!WRTEBR!IMM!<<MMEND-2>>>	
1900	022044		CMP BR SPO	
(1)		001651	MICPC=MICPC+1	
(1)	022044	000360	<SUBTC!BR!SPO>	
1901	022046		Z NTRS2	;IT DID WRAP AROUND
(1)		001652	MICPC=MICPC+1	
(1)	022046	115655	<JUMP!ZCOND!<NTRS2-INIT&3000*4>!<NTRS2-INIT&777/2>>	
1902	022050		BRWRTE IMM,2	;OFFSET TO NEXT PAIR
(1)		001653	MICPC=MICPC+1	
(1)	022050	000402	<MOVE!WRTEBR!IMM!<2>>	
1903	022052		MEM BR ADD!SPO	;UPDATE QUEUE POINTER
(1)		001654	MICPC=MICPC+1	
(1)	022052	062400	<MOVE!WRMEM!BR!<ADD!SPO>>	
1904	022054		NTRS2: BRWRTE IMM,20	
(1)		001655	MICPC=MICPC+1	
(1)	022054	000420	<MOVE!WRTEBR!IMM!<20>>	
1905	022056		SPBR BR AORB,SP1	
(1)		001656	MICPC=MICPC+1	
(1)	022056	063701	<MOVE!SPBRX!BR!AORB!SP1>	
1906	022060		BRD TAB1	;FLAGGED BY ERROR TYPE
(1)		001657	MICPC=MICPC+1	
(1)	022060	116320	<JUMP!BROCON!<TAB1-INIT&3000*4>!<TAB1-INIT&777/2>>	
1907	022062		SNAK: LDMA IMM,ISP11	
(1)		001660	MICPC=MICPC+1	
(1)	022062	010171	<MOVE!LDMAR!IMM!<ISP11&377>>	
1908	022064		SP MEMX SELB,SP11	
(1)		001661	MICPC=MICPC+1	
(1)	022064	043231	<MOVE!SPX!MEMX!SELB!SP11>	
1909	022066		SP BR,INCA,SP11	;INCREMENT MSG EXPECTED

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

```

(1)
(1) 022066 001662 063071
1910 022070
(1)
(1) 022070 001663 000401
1911 022072
(1)
(1) 022072 001664 063310
1912 022074
(1)
(1) 022074 001665 104415

```

```

MICPC=MICPC+1
<MOVE!SPX!BR!INCA!SP11>
SNAK1: BRWRT IMM,1 ;UNNUMB PENING BIT TO BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<1>>
SNAK2: SP BP,AORB,SP10 ;UPDATE LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP10>
ALWAYS FLUSH
MICPC=MICPC+1
<JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>

```

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

```

1914 022076 001666
(1) 022076 000424
1915 022100 001667
(1) 022100 062226
1916 022102 001670
(1) 022102 000400
1917 022104 001671
(1) 022104 062227
1918 022106 001672
(1) 022106 023740
1919 022110 001673
(1) 022110 062222
1920 022112 001674
(1) 022112 000766
1921 022114 001675
(1) 022114 062223
1922 022116 001676
(1) 022116 000421
1923 022120 001677
(1) 022120 061230
1924 022122 001700
(1) 022122 061231
1925 022124 001701
(1) 022124 120620
1926 022126 001702
(1) 022126 103355
1927 022130 001703
(1) 022130 120600
1928 022132 001704
(1) 022132 116301
1929 022134 001705
(1) 022134 020540
1930 022136 001706
(1) 022136 116713
1931 022140 001707
(1) 022140 002026
1932 022142 001710
    
```

```

EMTRIG: BRWRT IMM,24
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<24>>
OUTPUT BR <SELB!OBA1>
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELB!OBA1>>
BRWRT IMM,0
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<0>>
OUTPUT BR <SELB!OBA2>
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELB!OBA2>>
SPBR IBUS, BM873, SPD ;READ BM873 ADDRESS---
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!BM873!SPD>
OUTPUT BR SELB!OUTDA1 ;SET UP LOW BYTE OF ADDRESS
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELB!OUTDA1>>
BRWRT IMM,366 ;HIGH BYTE BASE FOR ROM BOOT
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<366>>
EM4: OUTPUT BR SELB!OUTDA2 ;SET UP HIGH BYTE OF ADDRESS
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELB!OUTDA2>>
EM6: BRWRT IMM,21 ;MASK TO START NPR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<21>>
OUT BR <SELB!ONPR>
MICPC=MICPC+1
<MOVE!WROUTX!BR!<SELB!ONPR>>
OUT BR <SELB!OBR> ;START TIMER
MICPC=MICPC+1
<MOVE!WROUTX!BR!<SELB!OBR>>
CKTIME: BRWRT IBUS,UBBR ;CHECK TIMER
MICPC=MICPC+1
<MOVE!WRTEBR!IBUS!<UBBR>>
BR4 HALTED ;NPR TIMED OUT DO ACLOW
MICPC=MICPC+1
<JUMP!BR4CON!<HALTED-INIT&3000*4>!<HALTED-INIT&777/2>>
EM1: BRWRT IBUS,NPR ;READ NPR CONTROL
MICPC=MICPC+1
<MOVE!WRTEBR!IBUS!<NPR>>
BRO CKTIME
MICPC=MICPC+1
<JUMP!BROCON!<CKTIME-INIT&3000*4>!<CKTIME-INIT&777/2>>
BRWRT IBUS,IOBA1 ;READ LOW BYTE OF ADDRESS BACK
MICPC=MICPC+1
<MOVE!WRTEBR!IBUS!<IOBA1>>
BR1 EMS ;ADDRESS IS 26-DONE
MICPC=MICPC+1
<JUMP!BR1CON!<EMS-INIT&3000*4>!<EMS-INIT&777/2>>
OUTPUT IMM,<26!OBA1> ;SET UP LOW BYTE OF ADDRESS
MICPC=MICPC+1
<MOVE!WROUT!IMM!<26!OBA1>>
BRWRT IMM,0 ;CLEAR BR TO CLEAR LOW BYTE OF
MICPC=MICPC+1
    
```

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

```

(1) 022142 000400      <MOVE!WRTEBR!IMM!<0>>
1933 022142 000400      <MOVE!WRTEBR!IMM!<0>>
1934 022144 001711      OUTPUT BR SELB!OUTDA1      ;DATA
(1) 022144 062222      MICPC=MICPC+1              ;ZERO LOW BYTE OF DATA
(1) 022144 062222      <MOVE!WROUT!BR!<SELB!OUTDA1>>
1935 022146 001712      ALWAYS EM4                ;GO DO NPR
(1) 022146 114675      MICPC=MICPC+1
(1) 022146 114675      <JUMP!ALCOND!<EM4-INIT&3000*4>!<EM4-INIT&777/2>>
1936 022150 001713      EMS: BRADOR RM1           ;IF NPR DONE
(1) 022150 000757      MICPC=MICPC+1
(1) 022150 000757      <MOVE!WRTEBR!<RM1-INIT&777/2>>
1937 022152 001714      ALWAYS ACLOW
(1) 022152 100756      MICPC=MICPC+1
(1) 022152 100756      <JUMP!ALCOND!<ACLOW-INIT&3000*4>!<ACLOW-INIT&777/2>>
1938 022154 001715      TABUPD: SPBR IBUS RCVCON,SP0 ;READ RECEIVER CONTROL REG
(1) 022154 023640      MICPC=MICPC+1
(1) 022154 023640      <MOVE!SPBRX!IBUS!RCVCON!SP0>
1939 022156 001716      BRWRT BR ADD!SP0         ;SHIFT LEFT
(1) 022156 060400      MICPC=MICPC+1
(1) 022156 060400      <MOVE!WRTEBR!BR!<ADD!SP0>>
1940 022160 001717      BR7 IDLE                ;RECEIVE ACTIVE--IDLE
(1) 022160 103447      MICPC=MICPC+1
(1) 022160 103447      <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1941 022162 001720      TAB1: LDMA IMM,IMG10
(1) 022162 010154      MICPC=MICPC+1
(1) 022162 010154      <MOVE!LDMAR!IMM!<IMG10&377>>
1942 022164 001721      BRWRT IMM,42
(1) 022164 000442      MICPC=MICPC+1
(1) 022164 000442      <MOVE!WRTEBR!IMM!<42>>
1943 022166 001722      MEMINC BR AANDB!SP10    ;SAVE BITS 1 AND 5 OF SP10
(1) 022166 076670      MICPC=MICPC+1
(1) 022166 076670      <MOVE!WRMEM!INCMAR!BR!<AANDB!SP10>>
1944 022170 001723      MEMINC BR SELA!SP11    ;SAVE SP11
(1) 022170 076611      MICPC=MICPC+1
(1) 022170 076611      <MOVE!WRMEM!INCMAR!BR!<SELA!SP11>>
1945 022172 001724      MEMINC BR SELA!SP15    ;SAVE SP15
(1) 022172 076615      MICPC=MICPC+1
(1) 022172 076615      <MOVE!WRMEM!INCMAR!BR!<SELA!SP15>>
1946 022174 001725      MEMINC BR SELA!SP17    ;SAVE SP17
(1) 022174 076617      MICPC=MICPC+1
(1) 022174 076617      <MOVE!WRMEM!INCMAR!BR!<SELA!SP17>>
1947 022174 076617      ;MAR NOW POINTS TO BASE
1948 022176 001726      STATE RB2              ;DO NOT CHANGE BR UNTIL RBO
(1) 022176 000461      MICPC=MICPC+1
(1) 022176 000461      <MOVE!WRTEBR!IMM!<RB2-INIT&777/2>>
1949 022200 001727      LDMA IMM,TABST        ;POINT TO TABLE UPDATE STATE
(1) 022200 010210      MICPC=MICPC+1
(1) 022200 010210      <MOVE!LDMAR!IMM!<TABST&377>>
1950 022202 001730      PSTATE TBU1          ;NEW PORT STATE ADDRESS
(1) 022202 002764      MEM IMM, <<TBU1-INIT&777/2>>
(2) 022202 002764      MICPC=MICPC+1
(2) 022202 002764      <MOVE!WRMEM!IMM!<<TBU1-INIT&777/2>>>
1951 022204 001731      SP IMM,4,SP4          ;INITIALIZE COUNT
(1) 022204 003004      MICPC=MICPC+1
(1) 022204 003004      <MOVE!SPX!IMM!4!SP4>
1952 022204 003004      ;NOTE: FIRST 6 RAM LOCATIONS ARE NOT WRITTEN

```

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

1953		
1954	022206	
(1)		001732
(1)	022206	010017
1955	022210	
(1)		001733
(1)	022210	104443

```

;TO CORE TABLE.
LDMA IMM, BASE
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<BASE&377>>
ALWAYS RBO
MICPC=MICPC+1
<JUMP!ALCONC!<RBO-INIT&3000*4>!<RBO-INIT&777/2>>

```

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

1957 022212 001734
 (1) 022212 000402
 1958 022214 001735
 (1) 022214 063004
 1959 022216 001736
 (1) 022216 023140
 1960 022220 001737
 (1) 022220 062006
 1961 022222 001740
 (1) 022222 023160
 1962 022224 001741
 (1) 022224 062107
 1963 022226 001742
 (1) 022226 101270
 1964 022230 001743
 (1) 022230 060601
 1965 022232 001744
 (1) 022232 116347
 1966 022234 001745
 (1) 022234 070604
 1968 022236 001746
 (1) 022236 117355
 1969 022240 001747
 (1) 022240 070604
 1970 022242 001750
 (1) 022242 117755
 1971 022244 001751
 (1) 022244 056222
 1972 022246 001752
 (1) 022246 056223
 1973 022250 001753
 (1) 022250 123200
 1974 022252 001754
 (1) 022252 104634
 1975 022254 001755
 (1) 022254 010210
 1976 022256

```

EC2:  BRWRT  IMM,2           ;INCREMENT COUNT/TEST
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<2>>
      SP BR,ADD,SP4
      MICPC=MICPC+1
      <MOVE!SPX!BR!ADD!SP4>
      SP IBUS,I0BA1,SPO       ;POINT TO NEXT ADDRESS
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!I0BA1!SPO>
      OUTPUT BR,ADD!OBA1
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<ADD!OBA1>>
      SP IBUS,I0BA2,SPO
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!I0BA2!SPO>
      OUTPUT BR,AC!OBA2
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<AC!OBA2>>
      C TABMXT
      MICPC=MICPC+1
      <JUMP!CCOND!<TABMXT-INIT&3000*4>!<TABMXT-INIT&777/2>>
ECX:  BRWRT  BR,SELA!SP1       ;READ PORT STATUS
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
      BR0 30$                 ;INIT MODE, WRITE OUT 200 BYTES
      MICPC=MICPC+1
      <JUMP!BROCON!<30$-INIT&3000*4>!<30$-INIT&777/2>>
      ;OTHERWISE ONLY WRITE OUT ERROR COUNTERS
      ;READ COUNTER
      BRWRT BR!LDMAR,SELA!SP4
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>
      BR4 20$                 ;ALL DONE
      MICPC=MICPC+1
      <JUMP!BR4CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>
30$:  BRWRT  BR!LDMAR,SELA!SP4   ;READ CTR
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>
      BR7 20$                 ;ALL DONE
      MICPC=MICPC+1
      <JUMP!BR7CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>
      OUTPUT MEMX!INCMAR,SELB!OUTDA1 ;STORE COUNTS OF ERRORS
      MICPC=MICPC+1
      <MOVE!WROUT!MEMX!INCMAR!<SELB!OUTDA1>>
      OUTPUT MEMX!INCMAR,SELB!OUTDA2
      MICPC=MICPC+1
      <MOVE!WROUT!MEMX!INCMAR!<SELB!OUTDA2>>
      SP IBUS,NPR,SPO
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!NPR!SPO>
      ALWAYS RKB
      MICPC=MICPC+1
      <JUMP!ALCOND!<RKB-INIT&3000*4>!<RKB-INIT&777/2>>
20$:  LDMA IMM,TABST
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<TABST&377>>
      PSTATE I3
  
```

DDCGAH.MAC 14-DEC-77 15:07

START HANDLER

```

(1) 022256 001756 MEM IMM, <<I3-INIT&777/2>>
(2) 022256 002455 MICPC=MICPC+1
1980 022260 <MOVE!WRMEM!IMM!<<I3-INIT&777/2>>>
1981 022260 RM1: STATE RCVA
(1) 001757 MICPC=MICPC+1
(1) 022260 000400 <MOVE!WRTEBR!IMM!<RCVA-INIT&777/2>>
1982 022262 ALWAYS REXIT
(1) 001760 MICPC=MICPC+1
(1) 022262 104425 <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
1983
1984 022264 ACK: BRWRT BR,AA!SP10 ;READ LINE STATUS SHIFTING LEFT
(1) 001761 MICPC=MICPC+1
(1) 022264 060530 <MOVE!WRTEBR!BR!<AA!SP10>>
1985 022266 BR4 ACK1 ;IF START RCVD--CLEAR START MODE
(1) 001762 MICPC=MICPC+1
(1) 022266 103373 <JUMP!BR4CON!<ACK1-INIT&3000*4>!<ACK1-INIT&777/2>>
1986 022270 BR7 FLUSHA ;IF HALF DUPLEX-SEND AN ACK
(1) 001763 MICPC=MICPC+1
(1) 022270 107422 <JUMP!BR7CON!<FLUSHA-INIT&3000*4>!<FLUSHA-INIT&777/2>>
1987 ;IF NOTHING ELSE TO SEND
1988 022272 ALWAYS IDLE
(1) 001764 MICPC=MICPC+1
(1) 022272 100447 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1990 022274 CBOOT: BRWRT IMM,200 ;MASK FOR BOOT MODE
(1) 001765 MICPC=MICPC+1
(1) 022274 000600 <MOVE!WRTEBR!IMM!<200>>
1991 022276 SP BR,AORB,SP1 ;IN PORT STATUS WORD
(1) 001766 MICPC=MICPC+1
(1) 022276 063301 <MOVE!SPX!BR!AORB!SP1>
1992 022300 BRWRT IMM,204 ;MASK FOR OK TOSEND AND LINE IDLE
(1) 001767 MICPC=MICPC+1
(1) 022300 000604 <MOVE!WRTEBR!IMM!<204>>
1993 022302 SP BR,SELB,SP10 ;IN LINE STATUS
(1) 001770 MICPC=MICPC+1
(1) 022302 063230 <MOVE!SPX!BR!SELB!SP10>
1994 022304 TSTATE TMTA
(1) 001771 MICPC=MICPC+1
(1) 022304 000400 <MOVE!WRTEBR!IMM!<TMTA-INIT&777/2>>
(1) 001772 MICPC=MICPC+1
(1) 022306 063222 <MOVE!SPX!BR!SELB!SP2>
1995 022310 ALWAYS INS12
(1) 001773 MICPC=MICPC+1
(1) 022310 100662 <JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
2004 022312 RCVCK: SPBR IBUS,RCVCON,SPO
(1) 001774 MICPC=MICPC+1
(1) 022312 023640 <MOVE!SPBRX!IBUS!RCVCON!SPO>
2005 022314 BRWRT BR,ADD!SPO
(1) 001775 MICPC=MICPC+1
(1) 022314 060400 <MOVE!WRTEBR!BR!<ADD!SPO>>
2006 022316 BR7 I1
(1) 001776 MICPC=MICPC+1
(1) 022316 103451 <JUMP!BR7CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
2007 022320 ALWAYS TAI
(1) 001777 MICPC=MICPC+1
(1) 022320 110402 <JUMP!ALCOND!<TAI-INIT&3000*4>!<TAI-INIT&777/2>>

```

G02

DDCGAH.MAC 14-DEC-77 15:07
2009 000001

START HANDLER
.END

PAGE: 02260M
SEQ 0226

. ABS. 022322 000

ERRORS DETECTED: 0

.DDCMP/CRF/DS:CRF+DMCNEW,HI1,DDCGAH
RUN-TIME: 6 10 0 SECONDS
RUN-TIME RATIO: 77/16=4.6
CORE USED: 6K (11 PAGES)

1671

CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685 022324 012737 000001 001226 TST1:
1686 022332 012737 022762 001216
1687
1688 022340 005737 022362
1689 022344 001002
1690 022346 104402 022364
1691 022352 012737 177777 022362 1$:
1692 022360 104400
1693 022362 000000
1694 022364 005377 055103 046504 ROMNUM: 0
(1) 022447 377 042012 041515 ROM1:
(1) 022475 377 031462 033055
(1) 022532 005015 031462 033055
(1) 022570 005015 031462 033055
(1) 022614 005377 046504 030503
(1) 022642 031377 026463 031066
(1) 022677 015 031012 026463
(1) 022735 015 031012 026463
(1) 022747 015 031012 026463
(1)
(3)
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708 022762 012737 000002 001226 TST2:
1709 022770 012737 023054 001216
1710
1711 022776 004737 027014
1712 023002 032737 100000 001366
1713 023010 001420
1714 023012 005000
1715 023014 013702 012322
1716 023020 012711 002000 1$:

```

```

:***** TEST 1 *****
:THIS IS A SPECIAL TEST WHICH IS ONLY EXECUTED ONE TIME.
:THE FIRST PASS AFTER THE DIAGNOSTIC IS LOADED. IT TYPES ON
:THE CONSOLE THE PART NUMBERS OF THE CROMS WHICH THIS
:REVISION SUPPORTS. TO FORCE A TYPE OUT PATCH LOCATION
:ROMNUM: TO A ZERO.
:*****

```

TEST 1

```

-----
MOV #1,TSTNO
MOV #TST2,NEXT
TST ROMNUM ;R1 CONTAINS BASE DMC11 ADDRESS
BNE 1$ ;FIRST TIME HERE?
TYPE, ROM1 ;SKIP IF NOT
MOV #-1,ROMNUM ;TYPE PART NUMBERS
SCOPE ;SET FLAG TO ONLY TYPE ONCE
ROMNUM: 0
ROM1: .ASCII <37><12>/CZDMG-0 SUPPORTS THE FOLLOWING CROM PART NUMBERS:/
.ASCII <37><12>/DMC11-AR (M8200-YA)/
.ASCII <37>/23-630A9/<15><12>/23-631A9/<15><12>/23-632A9/
.ASCII <15><12>/23-632A9/<15><12>/23-634A9/<15><12>/23-635A9/
.ASCII <15><12>/23-636A9/<15><12>/23-637A9/
.ASCII <37><12>/DMC11-AL (M8200-YB)/
.ASCII <37>/23-622A9/<15><12>/23-623A9/<15><12>/23-624A9/
.ASCII <15><12>/23-625A9/<15><12>/23-626A9/<15><12>/23-627A9/
.ASCII <15><12>/23-628A9/
.ASCIIZ <15><12>/23-629A9/

```

.EVEN

```

:***** TEST 2 *****
:THIS IS A SPECIAL TEST WHICH WILL RUN ON A KMC (DMC WITH
:WRITTABLE CONTROL STORE) TO LOAD THE CROM WITH THE JDCMP
:MICRO-CODE. FIRST BE SURE BIT1 OF STAT3 IS SET UP AS FOLLOWS
:1=LOCAL HIGH SPEED CODE, 0=REMOTE LOW SPEED CODE THE STAT3
:OF STAT3 BIT1 DETERMINES WHICH MICRO-CODE WILL
:BE LOADED IN THE KMC. LOOP ON THIS TEST FOR A FEW SECONDS
:TO LOAD THE KMC.
:*****

```

TEST 2

```

-----
MOV #2,TSTNO
MOV #TST3,NEXT
JSR PC_MAPCK ;R1 CONTAINS BASE DMC11 ADDRESS
BIT #BIT15,STAT1 ;CHECK FOR HI OR LO
BEQ 2$ ;BE SURE DMC HAS CROM
CLR R0 ;SKIP IF NO CROM
MOV ROMMAP,R2 ;R0=CRAM ADDRESS
MOV #BIT10,(R1) ;R2 POINTS TO ROMMAP
;SET ROM0

```


CZDMGD.P11 05-JAN-78 11:08

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1717 023024 010061 000004      MOV      R0,4(R1)      ;LOAD CRAM ADDRESS
1718 023030 012261 000006      MOV      (R2)+,6(R1)  ;LOAD WORD TO BE WRITTEN
1719 023034 052711 020000      BIS      #BIT13,(R1)  ;WRITE IT!
1720 023040 005200                INC      R0            ;NEXT ADDRESS
1721 023042 022700 002000      CMP      #2000,R0     ;DONE YET?
1722 023046 001364                BNE      1$           ;BR IF NO
1723 023050 005011                CLR      (R1)         ;CLEAR SELO
1724 023052 104400      2$: SCOPE                ;SCOPE THIS TEST
1725
1726
1727 ;***** TEST 3 *****
1728 ;*TEST OF BR RIGHT SHIFT
1729 ;*VERIFY THAT A DEST OF BR RSH (011) OF A MICRO-INSTRUCTION
1730 ;*SHIFTS THE RESULTING BR DATA RIGHT ONCE.
1731 ;*****
1732
1733 ; TEST 3
1734 ;-----
1735 023054 012737 000003 001226 TST3: MOV      #3,TSTNO
1736 023062 012737 023170 001216      MOV      #TST4,NEXT
1737
1738 023070 104412                MSTCLR
1739 023072 104412                MSTCLR
1740 023074 013701 001404      MOV      DMCSR,R1
1741 023100 005011                CLR      (R1)
1742 023102 012705 052525      MOV      #52525,R5
1743 023106 010561 000004      MOV      R5,4(R1)
1744 023112 104414                ROMCLK
1745 023114 120500                120500
1746 023116 104414                ROMCLK
1747 023120 061620                061620
1748 023122 104414                ROMCLK
1749 023124 061225                061225
1750 023126 006005                ROR      R5
1751 023130 116104 000005      MOV      5(R1),R4
1752 023134 120504                CMP      R5,R4
1753 023136 001401                BEQ      1$
1754 023140 104012                HLT      12
1755 023142                1$:
1756 023142 104414                ROMCLK
1757 023144 061620                061620
1758 023146 104414                ROMCLK
1759 023150 061225                061225
1760 023152 006005                ROR      R5
1761 023154 116104 000005      MOV      5(R1),R4
1762 023160 120504                CMP      R5,R4
1763 023162 001401                BEQ      2$
1764 023164 104012                HLT      12
1765 023166 104400      2$: SCOPE                ;SCOPE THIS TEST
1766
1767
1768 ;***** TEST 4 *****
1769 ;*CROM READ TEST
1770 ;*THIS TEST READS EACH ROM LOCATION AND COMPARES
1771 ;*IT TO A SOFTWARE DUPLICATE OF THE CROM. THIS TEST
1772 ;*ALSO TESTS THE JUMP(I) MICRO-PROCESSOR INSTRUCTION.

```

CZDMGD.P11 05-JAN-78 11:08

CROM READ TESTS

```

1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799 023170 012737 000004 001226
1800 023176 012737 023364 001216
1801 023204 012737 023242 001220
1802
1803 023212 104412
1804 023214 032737 100000 001366
1805 023222 001057
1806 023224 004737 027014
1807 023230 005011
1808 023232 013700 012322
1809 023236 005002
1810 023240 005003
1811 023242 042737 014377 023262 1$:
1812 023250 050237 023262
1813 023254 050337 023262
1814 023260 104414
1815 023262 100400 2$:
1816 023264 012711 002000
1817 023270 011005
1818 023272 016104 000006
1819 023276 020504
1820 023300 001414
1821 023302 010337 001252
1822 023306 000241
1823 023310 006037 001252
1824 023314 006037 001252
1825 023320 006037 001252
1826 023324 050237 001252
1827 023330 104004
1828 023332 104401 3$:

```

```

: *IF THIS TEST FAILS CHECK YOUR CROM PART NUMBERS.
: *CZDMG-D SUPPORTS THE FOLLOWING PART NUMBERS:
: *
: *DMC11-AR (M8200-YA)
: * 23-630A9
: * 23-631A9
: * 23-632A9
: * 23-633A9
: * 23-634A9
: * 23-635A9
: * 23-636A9
: * 23-637A9
: *
: *DMC11-AL (M8200-YB)
: * 23-622A9
: * 23-623A9
: * 23-624A9
: * 23-625A9
: * 23-626A9
: * 23-627A9
: * 23-628A9
: * 23-629A9
: *****

```

TEST 4

```

:-----
TST4: MOV #4,TSTNO
MOV #TST5,NEXT
MOV #1$,LOCK

MSTCLR
BIT #BIT15,STAT1
BNE 4$
JSR PC,MAPCK
CLR (R1)
MOV ROMMAP,R0
CLR R2
CLR R3
BIC #14377,2$
BIS R2,2$
BIS R3,2$
ROMCLK
100400
MOV #BIT10,(R1)
MOV (R0),R5
MOV 6(R1),R4
CMP R5,R4
BEQ 3$
MOV R3,TEMP3
CLC
ROR TEMP3
ROR TEMP3
ROR TEMP3
BIS R2,TEMP3
HLT 4
SCOP1

```

```

: R1 CONTAINS BASE DMC11 ADDRESS
: MASTER CLEAR DMC11
: IS IT RAM OR ROM
: SKIP TEST IF CROM
: CHECK FOR HI OR LO
: CLEAR RUN
: R0 POINTS TO SOFTWARE ROM MAP
: R2 CONTAINS ROM ADDRESS BITS 0-7
: R3 CONTAINS ROM ADDRESS BITS 8&9 IN BITS 11&12
: CLEAR ADDRESS FIELDS OF INSTRUCTION
: ADD BITS 0-7 TO INSTRUCTION
: ADD BITS 11&12 TO INSTRUCTION
: NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
: JUMP(I) TO ROM ADDRESS IN R2 & R3
: SET ROMO
: PUT "EXPECTED" IN R5
: PUT "FOUND" IN R4
: COMPARE ROM CONTENTS TO SOFT DUP
: BR IF OK
: PUT ROM ADDRESS IN TEMP3
: FOR ERROR TYPEOUT

: TEMP3 NOW CONTAINS CORRECT ADDRESS
: ROM READ ERROR
: LOOP TO 1$ IF SW09=1

```

CZDMGD.P11 05-JAN-78 11:08

CROM READ TESTS

1829	023334	005720	
1830	023336	005202	
1831	023340	022702	000400
1832	023344	001336	
1833	023346	005002	
1834	023350	062703	004000
1835	023354	022703	020000
1836	023360	001330	
1837	023362	104400	

4\$:

```

TST (R0)+ ;BUMP SOFT POINTER
INC R2 ;BUMP ROM ADDRESS
CMP #400,R2 ;IS R2 TO MAX YET?
BNE 1$ ;BR IF NO
CLR R2 ;YES, RESET R2 TO 0
ADD #4000,R3 ;INC TO NEXT PAGE OF ROM
CMP #20000,R3 ;DONE YET?
BNE 1$ ;BR IF NO
SCOPE ;SCOPE THIS TEST

```

1838
1839
1840
1841
1842
1843
1844
1845
1846
1847

```

***** TEST 5 *****
*CROM TEST OF JUMP(I) NEVER MICRO-PROCESSOR INSTRUCTION.
*PERFORM THE JUMP INSTRUCTION
*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
*THE CONTENTS OF THE NEW ROM PC (IT SHOULD INCREMENT BY ONE).
*****

```

TEST 5

1848					
1849	023364	012737	000005	001226	TST5:
1850	023372	012737	023560	001216	
1851	023400	012737	023424	001220	
1852					
1853	023406	104412			
1854	023410	032737	100000	001366	
1855	023416	001057			
1856	023420	004737	027014		
1857	023424				1\$:
1858	023424	004737	026660		
1859	023430	104414			
1860	023432	100400			
1861	023434	104414			
1862	023436	114377			
1863	023440	004737	026752		
1864	023444	000002			
1865	023446	020504			
1866	023450	001401			
1867	023452	104006			
1868	023454	104401			2\$:
1869	023456	012737	023464	001220	
1870	023464				3\$:
1871	023464	004737	026660		
1872	023470	104414			
1873	023472	100403			
1874	023474	104414			
1875	023476	100000			
1876	023500	004737	026752		
1877	023504	000010			
1878	023506	020504			
1879	023510	001401			
1880	023512	104006			
1881	023514	104401			4\$:
1882	023516	012737	023524	001220	
1883	023524				5\$:
1884	023524	004737	026660		

TST5:

1\$:

2\$:

3\$:

4\$:

5\$:

```

MOV #5,TSTNO
MOV #TST6,NEXT
MOV #1$,LOCK ;R1 CONTAINS BASE DMC11 ADDRESS
MSTCLR ;MASTER CLEAR DMC11
BIT #BIT15,STAT1 ;IS IT CRAM?
BNE 6$+2 ;SKIP TEST IF YES
JSR PC,MAPCK ;CHECK FOR HI OR LO
JSR PC,CLRALL ;CLEAR ALL CONDITIONS
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 ;START AT ROM PC=0
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
114377! <400*0> ;JUMP TO ROM PC OF 1777
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 2$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
SCOPI ;LOOP TO 1$ IF SW09=1
MOV #3$,LOCK ;NEW SCOPI
JSR PC,CLRALL ;CLEAR ALL CONDITIONS
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100403 ;START AT ROM PC=3
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100000! <400*0> ;JUMP TO ROM PC OF 0
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
10 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 4$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
SCOPI ;LOOP TO 3$ IF SW09=1
MOV #5$,LOCK ;NEW SCOPI
JSR PC,CLRALL ;CLEAR ALL CONDITIONS

```

CZDMGD.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

1885 023530 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1886 023532 100406 100406 ;START AT ROM PC=6
1887 023534 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1888 023536 104125 104125! <400*0> ;JUMP TO ROM PC OF 525
1889 023540 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1890 023544 000016 16 ;INDEX
1891 023546 020504 CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
1892 023550 001401 BEQ 6$ ;BR IF YES
1893 023552 104006 HLT 6 ;ERROR, CROM PC IS WRONG
1894 023554 104401 6$: SCOPI ;LOOP TO 5$ IF SW59=1
1895 023556 104400 SCOPE ;SCOPE THIS TEST
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906 02 560 012737 000006 001226 TST6: MOV #6,TSTNO ;***** TEST 6 *****
1907 02 566 012737 023740 001216 MOV #TST7,NEXT ;*CROM TEST OF JUMP(I) ALWAYS MICRO-PROCESSOR INSTRUCTION.
1908 023574 012737 023620 001220 MOV #1$,LOCK ;*PERFORM THE JUMP INSTRUCTION
1909
1910 023602 104412 MSTCLR ;*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
1911 023604 032737 100000 001366 BIT #BIT15,STAT1 ;*****
1912 023612 001051 BNE 6$+2 ;RI CONTAINS BASE DMC11 ADDRESS
1913 023614 004737 027014 JSR PC,MAPCK ;MASTER CLEAR DMC11
1914 023620 15: ;IS IT CROM?
1915 023620 104414 ROMCLK ;CHECK FOR HI OR LO
1916 023622 100400 100400 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1917 023624 104414 ROMCLK ;START AT ROM PC=0
1918 023626 114777 114777! <400*1> ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1919 023630 004737 026752 JSR PC,ROMDAT ;JUMP TO ROM PC OF 1777
1920 023634 003776 3776 ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1921 023636 020504 CMP R5,R4 ;INDEX
1922 023640 001401 BEQ 2$ ;ARE NEW PC CONTENTS CORRECT?
1923 023642 104006 HLT 6 ;BR IF YES
1924 023644 104401 2$: SCOPI ;ERROR, CROM PC IS WRONG
1925 023646 012737 023654 001220 MOV #3$,LOCK ;LOOP TO 1$ IF SW09=1
1926 023654 3$: ;NEW SCOPI
1927 023654 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1928 023656 100403 100403 ;START AT ROM PC=3
1929 023660 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1930 023662 100400 100000! <400*1> ;JUMP TO ROM PC OF 0
1931 023664 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1932 023670 000000 0 ;INDEX
1933 023672 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
1934 023674 001401 BEQ 4$ ;BR IF YES
1935 023676 104006 HLT 6 ;ERROR, CROM PC IS WRONG
1936 023700 104401 4$: SCOPI ;LOOP TO 3$ IF SW09=1
1937 023702 012737 023710 001220 MOV #5$,LOCK ;NEW SCOPI
1938 023710 5$:
1939 023710 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1940 023712 100406 100406 ;START AT ROM PC=6

```

CZDMGD.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

1941 023714 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1942 023716 104525 104125! <400*1> ;JUMP TO ROM PC OF 525
1943 023720 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1944 023724 001252 ;INDEX
1945 023726 020504 CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
1946 023730 001401 BEQ 6$ ;BR IF YES
1947 023732 104006 HLT 6 ;ERROR, CROM PC IS WRONG
1948 023734 104401 6$: SCOPI ;LOOP TO 5$ IF SW59=1
1949 023736 104400 SCOPE ;SCOPE THIS TEST
1950
1951
1952 ;***** TEST 7 *****
1953 ;*CROM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
1954 ;*SET THE C BIT, PERFORM THE JUMP INSTRUCTION.
1955 ;*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
1956 ;*****
1957
1958 ; TEST 7
1959 ;-----
1960 023740 012737 000007 001226 TST7: MOV #7,TSTNO
1961 023746 012737 024134 001216 MOV #TST10,NEXT
1962 023754 012737 024000 001220 MOV #1$,LOCK
1963
1964 023762 104412 MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
1965 023764 032737 100000 001366 BIT #BIT15,STAT1 ;MASTER CLEAR DMC11
1966 023772 001057 BNE 6$+2 ;IS IT CROM?
1967 023774 004737 027014 JSR PC,MAPCK ;SKIP TEST IF YES
1968 024000 1$: ;CHECK FOR HI OR LO
1969 024000 004737 026726 JSR PC,SETC ;SET THE C BIT'
1970 024004 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1971 024006 100400 100400 ;START AT ROM PC=0
1972 024010 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1973 024012 115377 114377! <400*2> ;JUMP TO ROM PC OF 1777
1974 024014 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1975 024020 003776 3776 ;INDEX
1976 024022 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
1977 024024 001401 BEQ 2$ ;BR IF YES
1978 024026 104006 HLT 6 ;ERROR, CROM PC IS WRONG
1979 024030 104401 2$: SCOPI ;LOOP TO 1$ IF SW09=1
1980 024032 012737 024040 001220 MOV #3$,LOCK ;NEW SCOPI
1981 024040 3$:
1982 024040 004737 026726 JSR PC,SETC ;SET THE C BIT'
1983 024044 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1984 024046 100403 100403 ;START AT ROM PC=3
1985 024050 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1986 024052 101000 100000! <400*2> ;JUMP TO ROM PC OF 0
1987 024054 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1988 024060 000000 0 ;INDEX
1989 024062 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
1990 024064 001401 BEQ 4$ ;BR IF YES
1991 024066 104006 HLT 6 ;ERROR, CROM PC IS WRONG
1992 024070 104401 4$: SCOPI ;LOOP TO 3$ IF SW09=1
1993 024072 012737 024100 001220 MOV #5$,LOCK ;NEW SCOPI
1994 024100 5$:
1995 024100 004737 026726 JSR PC,SETC ;SET THE C BIT'
1996 024104 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

CZDMGD.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

1997 024106 100406 100406 ; START AT ROM PC=6
1998 024110 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1999 024112 105125 104125! <400*2> ; JUMP TO ROM PC OF 525
2000 024114 004737 026752 JSR PC,ROMDAT ; R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2001 024120 001252 1252 ; INDEX
2002 024122 020504 CMP R5,R4 ; ARE NEW ROM PC CONTENTS CORRECT?
2003 024124 001401 BEQ 6$ ; BR IF YES
2004 024126 104006 HLT 6 ; ERROR, CROM PC IS WRONG
2005 024130 104401 6$: SCOP1 ; LOOP TO 5$ IF SW59=1
2006 024132 104400 SCOPE ; SCOPE THIS TEST
2007
2008
2009 ; ***** TEST 10 *****
2010 ; *CROM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
2011 ; *SET THE Z BIT, PERFORM THE JUMP INSTRUCTION.
2012 ; *VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
2013 ; *****
2014
2015 ; TEST 10
2016 ; -----
2017 024134 012737 000010 001226 TST10: MOV #10,TSTNO
2018 024142 012737 024330 001216 MOV #TST11,NEXT
2019 024150 012737 024174 001220 MOV #1$,LOCK
2020 ; R1 CONTAINS BASE DMC11 ADDRESS
2021 024156 104412 MSTCLR ; MASTER CLEAR DMC11
2022 024160 032737 100000 001366 BIT #BIT15,STAT1 ; IS IT CRAM?
2023 024166 001057 BNE 6$+2 ; SKIP TEST IF YES
2024 024170 004737 027014 JSR PC,MAPCK ; CHECK FOR HI OR LO
2025 024174 1$:
2026 024174 004737 026744 JSR PC,SETZ ; SET THE Z BIT'
2027 024200 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2028 024202 100400 100400 ; START AT ROM PC=0
2029 024204 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2030 024206 115777 114377! <400*3> ; JUMP TO ROM PC OF 1777
2031 024210 004737 026752 JSR PC,ROMDAT ; R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2032 024214 003776 3776 ; INDEX
2033 024216 020504 CMP R5,R4 ; ARE NEW PC CONTENTS CORRECT?
2034 024220 001401 BEQ 2$ ; BR IF YES
2035 024222 104006 HLT 6 ; ERROR, CROM PC IS WRONG
2036 024224 104401 2$: SCOP1 ; LOOP TO 1$ IF SW09=1
2037 024226 012737 024234 001220 MOV #3$,LOCK ; NEW SCOPE
2038 024234 3$:
2039 024234 004737 026744 JSR PC,SETZ ; SET THE Z BIT'
2040 024240 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2041 024242 100403 100403 ; START AT ROM PC=3
2042 024244 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2043 024246 101400 100000! <400*3> ; JUMP TO ROM PC OF 0
2044 024250 004737 026752 JSR PC,ROMDAT ; R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2045 024254 000000 0 ; INDEX
2046 024256 020504 CMP R5,R4 ; ARE NEW PC CONTENTS CORRECT?
2047 024260 001401 BEQ 4$ ; BR IF YES
2048 024262 104006 HLT 6 ; ERROR, CROM PC IS WRONG
2049 024264 104401 4$: SCOP1 ; LOOP TO 3$ IF SW09=1
2050 024266 012737 024274 001220 MOV #5$,LOCK ; NEW SCOPE
2051 024274 5$:
2052 024274 004737 026744 JSR PC,SETZ ; SET THE Z BIT'

```

DMG0.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

2053 024300 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2054 024302 100406 100406 ;START AT ROM PC=6
2055 024304 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2056 024306 105525 104125! <400*3> ;JUMP TO ROM PC OF 525
2057 024310 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2058 024314 001252 1252 ;INDEX
2059 024316 020504 CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
2060 024320 001401 BEQ 6$ ;BR IF YES
2061 024322 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2062 024324 104401 6$: SCOPI ;LOOP TO 5$ IF SW59=1
2063 024326 104400 SCOPE ;SCOPE THIS TEST
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074 024330 012737 000011 001226 TST11: MOV #11,TSTNO
2075 024336 012737 024524 001216 MOV #TST12,NEXT
2076 024344 012737 024370 001220 MOV #1$,LOCK
2077
2078 024352 104412 MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
2079 024354 032737 100000 001366 BIT #BIT15,STAT1 ;MASTER CLEAR DMC11
2080 024362 001057 BNE 6$+2 ;IS IT CRAM?
2081 024364 004737 027014 JSR PC,MAPCK ;SKIP TEST IF YES
2082 024370 004737 026676 1$: JSR PC,SETBRO ;CHECK FOR HI OR LO
2083 024370 004737 026676 ;SET THE BRO BIT'
2084 024374 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2085 024376 100400 100400 ;START AT ROM PC=0
2086 024400 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2087 024402 116377 114377! <400*4> ;JUMP TO ROM PC OF 1777
2088 024404 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2089 024410 003776 3776 ;INDEX
2090 024412 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2091 024414 001401 BEQ 2$ ;BR IF YES
2092 024416 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2093 024420 104401 2$: SCOPI ;LOOP TO 1$ IF SW09=1
2094 024422 012737 024430 001220 MOV #3$,LOCK ;NEW SCOPI
2095 024430 004737 026676 3$: JSR PC,SETBRO ;SET THE BRO BIT'
2096 024430 004737 026676 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2097 024434 104414 ROMCLK ;START AT ROM PC=3
2098 024436 100403 100403 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2099 024440 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2100 024442 102000 100000! <400*4> ;JUMP TO ROM PC OF 0
2101 024444 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2102 024450 000000 0 ;INDEX
2103 024452 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2104 024454 001401 BEQ 4$ ;BR IF YES
2105 024456 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2106 024460 104401 4$: SCOPI ;LOOP TO 3$ IF SW09=1
2107 024462 012737 024470 001220 MOV #5$,LOCK ;NEW SCOPI
2108 024470 024470 001220 5$:

```

```

:***** TEST 11 *****
:*CROM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
:*SET THE BRO BIT, PERFORM THE JUMP INSTRUCTION.
:*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
:*****

```

: TEST 11

```

-----
MOV #11,TSTNO
MOV #TST12,NEXT
MOV #1$,LOCK

;R1 CONTAINS BASE DMC11 ADDRESS
MASTER CLEAR DMC11
IS IT CRAM?
SKIP TEST IF YES
CHECK FOR HI OR LO

;SET THE BRO BIT'
NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
START AT ROM PC=0
NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
JUMP TO ROM PC OF 1777
R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
INDEX
ARE NEW PC CONTENTS CORRECT?
BR IF YES
ERROR, CROM PC IS WRONG
LOOP TO 1$ IF SW09=1
NEW SCOPI

;SET THE BRO BIT'
NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
START AT ROM PC=3
NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
JUMP TO ROM PC OF 0
R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
INDEX
ARE NEW PC CONTENTS CORRECT?
BR IF YES
ERROR, CROM PC IS WRONG
LOOP TO 3$ IF SW09=1
NEW SCOPI

```

CZOMGD.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

2109 024470 004737 026676 JSR PC,SETBRO ;SET THE BRO BIT'
2110 024474 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2111 024476 100406 100406 ;START AT ROM PC=6
2112 024500 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2113 024502 106125 104125! <400*4> ;JUMP TO ROM PC OF 525
2114 024504 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2115 024510 001252 1252 ;INDEX
2116 024512 020504 CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
2117 024514 001401 BEQ 6$ ;BR IF YES
2118 024516 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2119 024520 104401 6$: SCOP1 ;LOOP TO 5$ IF SW59=1
2120 024522 104400 SCOPE ;SCOPE THIS TEST

:***** TEST 12 *****
:*CROM TEST OF JUMP(I) ON BRI SET MICRO-PROCESSOR INSTRUCTION.
:*SET THE BRI BIT, PERFORM THE JUMP INSTRUCTION,
:*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
:*****

; TEST 12
-----
2131 024524 012737 000012 001226 TST12: MOV #12,TSTNO
2132 024532 012737 024720 001216 MOV #TST13,NEXT
2133 024540 012737 024564 001220 MOV #1$,LOCK ;R1 CONTAINS BASE DMC11 ADDRESS
2134 ;MASTER CLEAR DMC11
2135 024546 104412 MSTCLR ;IS IT CROM?
2136 024550 032737 100000 001366 BIT #BIT15,STAT1 ;SKIP TEST IF YES
2137 024556 001057 0NE 6$+2 ;CHECK FOR HI OR LO
2138 024560 004737 027014 JSR PC,MAPCK
2139 024564 1$: JSR PC,SETBRI ;SET THE BRI BIT'
2140 024564 004737 026704 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2141 024570 104414 100400 ;START AT ROM PC=0
2142 024572 100400 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2143 024574 104414 ROMCLK ;JUMP TO ROM PC OF 1777
2144 024576 116777 114377! <400*5> ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2145 024600 004737 026752 JSR PC,ROMDAT ;INDEX
2146 024604 003776 3776 ;ARE NEW PC CONTENTS CORRECT?
2147 024606 020504 CMP R5,R4 ;BR IF YES
2148 024610 001401 BEQ 2$ ;ERROR, CROM PC IS WRONG
2149 024612 104006 HLT 6 ;LOOP TO 1$ IF SW09=1
2150 024614 104401 2$: SCOP1 ;NEW SCOPE
2151 024616 012737 024624 001220 3$: MOV #3$,LOCK
2152 024624 026704 JSR PC,SETBRI ;SET THE BRI BIT'
2153 024624 004737 026704 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2154 024630 104414 100403 ;START AT ROM PC=3
2155 024632 100403 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2156 024634 104414 ROMCLK ;JUMP TO ROM PC OF 0
2157 024636 102400 100000! <400*5> ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2158 024640 004737 026752 JSR PC,ROMDAT ;INDEX
2159 024644 000000 0 ;ARE NEW PC CONTENTS CORRECT?
2160 024646 020504 CMP R5,R4 ;BR IF YES
2161 024650 001401 BEQ 4$ ;ERROR, CROM PC IS WRONG
2162 024652 104006 HLT 6 ;LOOP TO 3$ IF SW09=1
2163 024654 104401 4$: SCOP1 ;NEW SCOPE
2164 024656 012737 024664 001220 MOV #5$,LOCK

```


CZDMGO.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

2165 024664 004737 026704 5$: JSR PC,SETBR1 ;SET THE BR1 BIT'
2166 024664 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2167 024670 104414 100406 ;START AT ROM PC=6
2168 024672 100406 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2169 024674 104414 104125! <400*5> ;JUMP TO ROM PC OF 525
2170 024676 106525 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2171 024700 004737 026752 1252 ;INDEX
2172 024704 001252 CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
2173 024706 020504 BEQ 6$ ;BR IF YES
2174 024710 001401 HLT 6 ;ERROR, CROM PC IS WRONG
2175 024712 104006 6$: SCOPE1 ;LOOP TO 5$ IF SW59=1
2176 024714 104401 SCOPE ;SCOPE THIS TEST
2177 024716 104400
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188 024720 012737 000013 001226 TST13: MOV #13,TSTNO
2189 024726 012737 025114 001216 MOV #TST14,NEXT
2190 024734 012737 024760 001220 MOV #1$,LOCK ;R1 CONTAINS BASE DMC11 ADDRESS
2191 ;MASTER CLEAR DMC11
2192 024742 104412 MSTCLR ;IS IT CROM?
2193 024744 032737 100000 001366 BIT #BIT15,STAT1 ;SKIP TEST IF YES
2194 024752 001057 BNE 6$+2 ;CHECK FOR HI OR LO
2195 024754 004737 027014 JSR PC,MAPCK
2196 024760 1$: JSR PC,SETBR4 ;SET THE BR4 BIT'
2197 024760 004737 026712 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2198 024764 104414 100400 ;START AT ROM PC=0
2199 024766 100400 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2200 024770 104414 114377! <400*6> ;JUMP TO ROM PC OF 1777
2201 024772 117377 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2202 024774 004737 026752 3776 ;INDEX
2203 025000 003776 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2204 025002 020504 BEQ 2$ ;BR IF YES
2205 025004 001401 HLT 6 ;ERROR, CROM PC IS WRONG
2206 025006 104006 2$: SCOPE1 ;LOOP TO 1$ IF SW09=1
2207 025010 104401 MOV #3$,LOCK ;NEW SCOPE1
2208 025012 012737 025020 001220 3$: JSR PC,SETBR4 ;SET THE BR4 BIT'
2209 025020 025020 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2210 025020 004737 026712 100403 ;START AT ROM PC=3
2211 025024 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2212 025026 100403 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2213 025030 104414 100000! <400*6> ;JUMP TO ROM PC OF 0
2214 025032 103000 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2215 025034 004737 026752 0 ;INDEX
2216 025040 000000 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2217 025042 020504 BEQ 4$ ;BR IF YES
2218 025044 001401 HLT 6 ;ERROR, CROM PC IS WRONG
2219 025046 104006 4$: SCOPE1 ;LOOP TO 3$ IF SW09=1
2220 025050 104401

```

```

:***** TEST 13 *****
:*CROM TEST OF JUMP(I) ON EXT. GET MICRO-PROCESSOR INSTRUCTION.
:*SET THE BR4 BIT, PERFORM THE JUMP INSTRUCTION.
:*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
:*****

```

: TEST 13

02DMGD.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

2221 025052 012737 025060 001220      5$:  MOV      #5$,LOCK      ;NEW SCOPI
2222 025060
2223 025060 004737 026712      JSR      PC,SETBP4      ;SET THE BR4 BIT'
2224 025064 104414      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2225 025066 100406      100406      ;START AT ROM PC=6
2226 025070 104414      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2227 025072 107125      104125! <400*6>      ;JUMP TO ROM PC OF 525
2228 025074 004737 026752      JSR      PC,ROMDAT      ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2229 025100 001252      1252      ;INDEX
2230 025102 020504      CMP      R5,R4      ;ARE NEW ROM PC CONTENTS CORRECT?
2231 025104 001401      BEQ      6$      ;BR IF YES
2232 025106 104006      HLT      6      ;ERROR, CROM PC IS WRONG
2233 025110 104401      6$:  SCOPI      ;LOOP TO 5$ IF SW59=1
2234 025112 104400      SCOPE      ;SCOPE THIS TEST

```

```

:***** TEST 14 *****
:*CROM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
:*SET THE BR7 BIT, PERFORM THE JUMP INSTRUCTION.
:*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
:*****

```

TEST 14

```

2245 025114 012737 000014 001226  TST14:  MOV      #14,TSTNO
2246 025122 012737 025310 001216      MOV      #TST15,NEXT
2247 025130 012737 025154 001220      MOV      #1$,LOCK
2248
2249 025136 104412      MSTCLR      ;R1 CONTAINS BASE DMC11 ADDRESS
2250 025140 032737 100000 001366      BIT      #BIT15,STAT1  ;MASTER CLEAR DMC11
2251 025146 001057      BNE      6$+2      ;IS IT CRAM?
2252 025150 004737 027014      JSR      PC,MAPCK      ;SKIP TEST IF YES
2253 025154
2254 025154 004737 026720      1$:  JSR      PC,SETBR7      ;SET THE BR7 BIT'
2255 025160 104414      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2256 025162 100400      100400      ;START AT ROM PC=0
2257 025164 104414      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2258 025166 117777      114377! <400*7>      ;JUMP TO ROM PC OF 1777
2259 025170 004737 026752      JSR      PC,ROMDAT      ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2260 025174 003776      3776      ;INDEX
2261 025176 020504      CMP      R5,R4      ;ARE NEW PC CONTENTS CORRECT?
2262 025200 001401      BEQ      2$      ;BR IF YES
2263 025202 104006      HLT      6      ;ERROR, CROM PC IS WRONG
2264 025204 104401      2$:  SCOPI      ;LOOP TO 1$ IF SW09=1
2265 025206 012737 025214 001220      MOV      #3$,LOCK      ;NEW SCOPI
2266 025214
2267 025214 004737 026720      3$:  JSR      PC,SETBR7      ;SET THE BR7 BIT'
2268 025220 104414      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2269 025222 100403      100403      ;START AT ROM PC=3
2270 025224 104414      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2271 025226 103400      100000! <400*7>      ;JUMP TO ROM PC OF 0
2272 025230 004737 026752      JSR      PC,ROMDAT      ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2273 025234 000000      0      ;INDEX
2274 025236 020504      CMP      R5,R4      ;ARE NEW PC CONTENTS CORRECT?
2275 025240 001401      BEQ      4$      ;BR IF YES
2276 025242 104006      HLT      6      ;ERROR, CROM PC IS WRONG

```

CZDMGD.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

2277 025244 104401 4$: SCOP1 ;LOOP TO 3$ IF SW09=1
2278 025246 012737 025254 001220 MOV #5$,LOCK ;NEW SCOPI
2279 025254 5$: JSR PC,SETBR7 ;SET THE BR7 BIT'
2280 025254 004737 026720 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2281 025260 104414 100406 ;START AT ROM PC=6
2282 025262 100406 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2283 025264 104414 104125! <400*7> ;JUMP TO ROM PC OF 525
2284 025266 107525 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2285 025270 004737 026752 1252 ;INDEX
2286 025274 001252 CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
2287 025276 020504 BEQ 6$ ;BR IF YES
2288 025300 001401 HLT 6 ;ERROR, CROM PC IS WRONG
2289 025302 104006 6$: SCOP1 ;LOOP TO 5$ IF SW59=1
2290 025304 104401 SCOPE ;SCOPE THIS TEST
2291 025306 104400
2292
2293
2294
2295 ;***** TEST 15 *****
2296 ;*CROM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
2297 ;*CLEAR THE C BIT, PERFORM THE JUMP INSTRUCTION.
2298 ;*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
2299 ;*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).
2300 ;*****
2301 ; TEST 15
2302 ;-----
2303 025310 012737 000015 001226 TST15: MOV #15,TSTNO
2304 025316 012737 025504 001216 MOV #TST16,NEXT
2305 025324 012737 025350 001220 MOV #1$,LOCK
2306 ;R1 CONTAINS BASE DMC11 ADDRESS
2307 025332 104412 MSTCLR ;MASTER CLEAR DMC11
2308 025334 032737 100000 001366 BIT #BIT15,STAT1 ;IS IT CRAM?
2309 025342 001057 BNE 6$+2 ;SKIP TEST IF YES
2310 025344 004737 027014 JSR PC,MAPCK ;CHECK FOR HI OR LO
2311 025350 1$: JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2312 025350 004737 026660 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2313 025354 104414 100400 ;START AT ROM PC=0
2314 025356 100400 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2315 025360 104414 114377! <400*2> ;JUMP TO ROM PC OF 1777
2316 025362 115377 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2317 025364 004737 026752 2 ;INDEX
2318 025370 000002 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2319 025372 020504 BEQ 2$ ;BR IF YES
2320 025374 001401 HLT 6 ;ERROR, CROM PC IS WRONG
2321 025376 104006 2$: SCOP1 ;LOOP TO 1$ IF SW09=1
2322 025400 104401 MOV #3$,LOCK ;NEW SCOPI
2323 025402 012737 025410 001220 3$: JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2324 025410 004737 026660 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2325 025410 104414 100403 ;START AT ROM PC=3
2326 025414 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2327 025416 100403 100000! <400*2> ;JUMP TO ROM PC OF 0
2328 025420 104414 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2329 025422 101000 10 ;INDEX
2330 025424 004737 026752 10 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2331 025430 000010
2332 025432 020504

```

020MGC.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

2333 025434 001401      BEQ      4$      ;BR IF YES
2334 025436 104006      HLT      6      ;ERROR, CROM PC IS WRONG
2335 025440 104401      4$:      SCOP1    ;LOOP TO 3$ IF SW09=1
2336 025442 012737 025450 001220  MOV      #5$,LOCK ;NEW SCOPI
2337 025450      5$:
2338 025450 004737 026660      JSR      PC,CLRALL ;CLEAR ALL CONDITIONS
2339 025454 104414      ROMCLK   ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2340 025456 100406      ;START AT ROM PC=6
2341 025460 104414      ROMCLK   ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2342 025462 105125      104125! <400*2> ;JUMP TO ROM PC OF 525
2343 025464 004737 026752      JSR      PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2344 025470 000016      16      ;INDEX
2345 025472 020504      CMP      R5,R4   ;ARE NEW ROM PC CONTENTS CORRECT?
2346 025474 001401      BEQ      6$      ;BR IF YES
2347 025476 104006      HLT      6      ;ERROR, CROM PC IS WRONG
2348 025500 104401      6$:      SCOP1    ;LOOP TO 5$ IF SW59=1
2349 025502 104400      SCOPE    ;SCOPE THIS TEST
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361 025504 012737 000016 001226 TST16:  MOV      #16,TSTNO
2362 025512 012737 025700 001216  MOV      #TST17,NEXT
2363 025520 012737 025544 001220  MOV      #1$,LOCK
2364
2365 025526 104412      MSTCLR   ;R1 CONTAINS BASE DMC11 ADDRESS
2366 025530 032737 100000 001366  BIT      #BIT15,STAT1 ;MASTER CLEAR DMC11
2367 025536 001057      BNE      6$+2    ;IS IT CRAM?
2368 025540 004737 027014      JSR      PC,MAPCK ;SKIP TEST IF YES
2369 025544      1$:
2370 025544 004737 026660      JSR      PC,CLRALL ;CHECK FOR HI OR LO
2371 025550 104414      ROMCLK   ;CLEAR ALL CONDITIONS
2372 025552 100400      100400   ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2373 025554 104414      ROMCLK   ;START AT ROM PC=0
2374 025556 115777      114377! <400*3> ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2375 025560 004737 026752      JSR      PC,ROMDAT ;JUMP TO ROM PC OF 1777
2376 025564 000002      2      ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2377 025566 020504      CMP      R5,R4   ;INDEX
2378 025570 001401      BEQ      2$      ;ARE NEW PC CONTENTS CORRECT?
2379 025572 104006      HLT      6      ;BR IF YES
2380 025574 104401      2$:      SCOP1    ;ERROR, CROM PC IS WRONG
2381 025576 012737 025604 001220  MOV      #3$,LOCK ;LOOP TO 1$ IF SW09=1
2382 025604      3$:
2383 025604 004737 026660      JSR      PC,CLRALL ;NEW SCOPI
2384 025610 104414      ROMCLK   ;CLEAR ALL CONDITIONS
2385 025612 100403      100403   ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2386 025614 104414      ROMCLK   ;START AT ROM PC=3
2387 025616 101400      100000! <400*3> ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2388 025620 004737 026752      JSR      PC,ROMDAT ;JUMP TO ROM PC OF 0
;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA

```

```

:***** TEST 16 *****
:*CROM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
:*CLEAR THE Z BIT, PERFORM THE JUMP INSTRUCTION,
:*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
:*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).
:*****

```

: TEST 16

JDMGD.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

2389 025624 000010      10      :INDEX
2390 025626 020504      CMP      R5,R4      :ARE NEW PC CONTENTS CORRECT?
2391 025630 001401      BEQ      4$          :BR IF YES
2392 025632 104006      HLT      6          :ERROR, CROM PC IS WRONG
2393 025634 104401      SCOPI   :LOOP TO 3$ IF SW09=1
2394 025636 012737 025644 001220 4$:      MOV      #5$,LOCK  :NEW SCOPI
2395 025644
2396 025644 004737 026660 5$:      JSR      PC,CLRALL  :CLEAR ALL CONDITIONS
2397 025650 104414      ROMCLK   :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2398 025652 100406      ROMCLK   :START AT ROM PC=6
2399 025654 104414      ROMCLK   :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2400 025656 105525      104125! <400*3> :JUMP TO ROM PC OF 525
2401 025660 004737 026752      JSR      PC,ROMDAT :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2402 025664 000016      16      :INDEX
2403 025666 020504      CMP      R5,R4      :ARE NEW ROM PC CONTENTS CORRECT?
2404 025670 001401      BEQ      6$          :BR IF YES
2405 025672 104006      HLT      6          :ERROR, CROM PC IS WRONG
2406 025674 104401      SCOPI   :LOOP TO 5$ IF SW59=1
2407 025676 104400      SCOPE   :SCOPE THIS TEST
2408
2409
2410 ***** TEST 17 *****
2411 *CROM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
2412 *CLEAR THE BRO BIT, PERFORM THE JUMP INSTRUCTION.
2413 *VERIFY THAT THE JUMP DID NOT OCCUR BY READING
2414 *THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).
2415 *****
2416
2417 ; TEST 17
2418 -----
2419 025700 012737 000017 001226 TST17: MOV      #17,TSTNO
2420 025706 012737 026074 001216      MOV      #TST20,NEXT
2421 025714 012737 025740 001220      MOV      #1$,LOCK
2422
2423 ;R1 CONTAINS BASE DMC11 ADDRESS
2424 025722 104412      MSTCLR  :MASTER CLEAR DMC11
2425 025724 032737 100000 001366      BIT      #BIT15,STAT1 :IS IT CRAM?
2426 025732 001057      BNE     6$+2       :SKIP TEST IF YES
2427 025734 004737 027014      JSR     PC,MAPCK   :CHECK FOR HI OR LO
2428 025740
2429 025740 004737 026660 1$:      JSR     PC,CLRALL  :CLEAR ALL CONDITIONS
2430 025744 104414      ROMCLK   :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2431 025746 100400      ROMCLK   :START AT ROM PC=0
2432 025750 104414      ROMCLK   :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2433 025752 116377      114377! <400*4> :JUMP TO ROM PC OF 1777
2434 025754 004737 026752      JSR     PC,ROMDAT :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2435 025760 000002      2       :INDEX
2436 025762 020504      CMP     R5,R4      :ARE NEW PC CONTENTS CORRECT?
2437 025764 001401      BEQ     2$          :BR IF YES
2438 025766 104006      HLT     6          :ERROR, CROM PC IS WRONG
2439 025770 104401      SCOPI   :LOOP TO 1$ IF SW09=1
2440 025772 012737 026000 001220 2$:      MOV     #3$,LOCK  :NEW SCOPI
2441 026000
2442 026000 004737 026660 3$:      JSR     PC,CLRALL  :CLEAR ALL CONDITIONS
2443 026004 104414      ROMCLK   :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2444 026006 100403      ROMCLK   :START AT ROM PC=3
2445 026010 104414      ROMCLK   :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

CZDMGC.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

2445 026012 102000 100000! <400*4> JUMP TO ROM PC OF 0
2446 026014 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2447 026020 000010 10 INDEX
2448 026022 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2449 026024 001401 BEQ 4$ ;BR IF YES
2450 026026 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2451 026030 104401 SCOPI ;LOOP TO 3$ IF SW09=1
2452 026032 012737 026040 001220 4$: MOV #5$,LOCK ;NEW SCOPI
2453 026040 5$:
2454 026040 004737 026660 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2455 026044 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2456 026046 100406 100406 ;START AT ROM PC=6
2457 026050 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2458 026052 106125 104125! <400*4> JUMP TO ROM PC OF 525
2459 026054 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2460 026060 000016 16 INDEX
2461 026062 020504 CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
2462 026064 001401 BEQ 6$ ;BR IF YES
2463 026066 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2464 026070 104401 SCOPI ;LOOP TO 5$ IF SW59=1
2465 026072 104400 SCOPE ;SCOPE THIS TEST

:***** TEST 20 *****
:*CROM TEST OF JUMP(I) ON BRI SET MICRO-PROCESSOR INSTRUCTION.
:*CLEAR THE BRI BIT, PERFORM THE JUMP INSTRUCTION,
:*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
:*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).
:*****

: TEST 20
:-----
2477 026074 012737 000020 001226 TST20: MOV #20,TSTNO
2478 026102 012737 026270 001216 MOV #TST21,NEXT
2479 026110 012737 026134 001220 MOV #1$,LOCK
2480 ;R1 CONTAINS BASE DMC11 ADDRESS
2481 026116 104412 MSTCLR ;MASTER CLEAR DMC11
2482 026120 032737 100000 001366 BIT #BIT15,STAT1 ;IS IT CRAM?
2483 026126 001057 BNE 6$+2 ;SKIP TEST IF YES
2484 026130 004737 027014 JSR PC,MAPCK ;CHECK FOR HI OR LO
2485 026134 1$:
2486 026134 004737 026660 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2487 026140 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2488 026142 100400 100400 ;START AT ROM PC=0
2489 026144 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2490 026146 116777 114377! <400*5> JUMP TO ROM PC OF 1777
2491 026150 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2492 026154 000002 2 INDEX
2493 026156 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2494 026160 001401 BEQ 2$ ;BR IF YES
2495 026162 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2496 026164 104401 SCOPI ;LOOP TO 1$ IF SW09=1
2497 026166 012737 026174 001220 2$: MOV #3$,LOCK ;NEW SCOPI
2498 026174 3$:
2499 026174 004737 026660 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2500 026200 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

02DMGD.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

2501 026202 100403          100403          :START AT ROM PC=3
2502 026204 104414          ROMCLK          :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2503 026206 102400          100000! <400*5> : JUMP TO
2504 026210 004737 026752    JSR PC,ROMDAT  :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2505 026214 000010          10          :INDEX
2506 026216 020504          CMP R5,R4      :ARE NEW PC CONTENTS CORRECT?
2507 026220 001401          BEQ 4$         :BR IF YES
2508 026222 104006          HLT 6          :ERROR, CROM PC IS WRONG
2509 026224 104401          SCOPI         :LOOP TO 3$ IF SW09=1
2510 026226 012737 026234 001220 4$: MOV #5$,LOCK  :NEW SCOPI
2511 026234          5$:
2512 026234 004737 026660    JSR PC,CLRALL  :CLEAR ALL CONDITIONS
2513 026240 104414          ROMCLK          :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2514 026242 100406          100406          :START AT ROM PC=6
2515 026244 104414          ROMCLK          :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2516 026246 106525          104125! <400*5> : JUMP TO ROM PC OF 525
2517 026250 004737 026752    JSR PC,ROMDAT  :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2518 026254 000016          16          :INDEX
2519 026256 020504          CMP R5,R4      :ARE NEW ROM PC CONTENTS CORRECT?
2520 026260 001401          BEQ 6$         :BR IF YES
2521 026262 104006          HLT 6          :ERROR, CROM PC IS WRONG
2522 026264 104401          SCOPI         :LOOP TO 5$ IF SW59=1
2523 026266 104400          SCOPE         :SCOPE THIS TEST
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535 026270 012737 000021 001226 TST21: MOV #21,TSTNO
2536 026276 012737 026464 001216 MOV #TST22,NEXT
2537 026304 012737 026330 001220 MOV #1$,LOCK
2538
2539 026312 104412          MSTCLR         :R1 CONTAINS BASE DMC11 ADDRESS
2540 026314 032737 100000 001366 BIT #BIT15,STAT1 :MASTER CLEAR DMC11
2541 026322 001057          BNE 6$+2       :IS IT CRAM?
2542 026324 004737 027014          JSR PC,MAPCK   :SKIP TEST IF YES
2543 026330          1$:          :CHECK FOR HI OR LO
2544 026330 004737 026660    JSR PC,CLRALL  :CLEAR ALL CONDITIONS
2545 026334 104414          ROMCLK          :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2546 026336 100400          100400          :START AT ROM PC=0
2547 026340 104414          ROMCLK          :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2548 026342 117377          114377! <400*6> : JUMP TO ROM PC OF 177?
2549 026344 004737 026752    JSR PC,ROMDAT  :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2550 026350 000002          2          :INDEX
2551 026352 020504          CMP R5,R4      :ARE NEW PC CONTENTS CORRECT?
2552 026354 001401          BEQ 2$         :BR IF YES
2553 026356 104006          HLT 6          :ERROR, CROM PC IS WRONG
2554 026360 104401          SCOPI         :LOOP TO 1$ IF SW09=1
2555 026362 012737 026370 001220 2$: MOV #3$,LOCK  :NEW SCOPI
2556 026370          3$:

```

```

:***** TEST 21 *****
:*CROM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
:*CLEAR THE BR4 BIT, PERFORM THE JUMP INSTRUCTION.
:*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
:*THE CONTENTS OF THE NEW ROM PC (IT SHOULD INCREMENT BY ONE).
:*****

```

: TEST 21

```

:-----
:MOV #21,TSTNO
:MOV #TST22,NEXT
:MOV #1$,LOCK
:R1 CONTAINS BASE DMC11 ADDRESS
:MSTCLR
:MASTER CLEAR DMC11
:BIT #BIT15,STAT1
:IS IT CRAM?
:BNE 6$+2
:SKIP TEST IF YES
:JSR PC,MAPCK
:CHECK FOR HI OR LO
:1$:
:CLEAR ALL CONDITIONS
:ROMCLK
:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
:100400
:START AT ROM PC=0
:ROMCLK
:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
:114377! <400*6>
:JUMP TO ROM PC OF 177?
:JSR PC,ROMDAT
:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
:2
:INDEX
:ARE NEW PC CONTENTS CORRECT?
:BEQ 2$
:BR IF YES
:HLT 6
:ERROR, CROM PC IS WRONG
:2$:
:LOOP TO 1$ IF SW09=1
:MOV #3$,LOCK
:NEW SCOPI
:3$:

```

CZDMGD.P11 05-JAN-78 11:08

CROM JUMP TESTS

```

2557 026370 004737 026660 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2558 026374 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2559 026376 100403 100403 ;START AT ROM PC=3
2560 026400 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2561 026402 103000 100000! <400*6> JUMP TO ROM PC OF 0
2562 026404 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2563 026410 000010 10 ;INDEX
2564 026412 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2565 026414 001401 BEQ 4$ ;BR IF YES
2566 026416 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2567 026420 104401 4$: SCOPI ;LOOP TO 3$ IF SW09=1
2568 026422 012737 026430 001220 MOV #5$,LOCK ;NEW SCOPI
2569 026430 5$:
2570 026430 004737 026660 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2571 026434 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2572 026436 100406 100406 ;START AT ROM PC=6
2573 026440 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2574 026442 107125 104125! <400*6> JUMP TO ROM PC OF 525
2575 026444 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2576 026450 000016 16 ;INDEX
2577 026452 020504 CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
2578 026454 001401 BEQ 6$ ;BR IF YES
2579 026456 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2580 026460 104401 6$: SCOPI ;LOOP TO 5$ IF SW59=1
2581 026462 104400 SCOPE ;SCOPE THIS TEST
2582
2583
2584 ;***** TEST 22 *****
2585 ;*CROM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
2586 ;*CLEAR THE BR7 BIT, PERFORM THE JUMP INSTRUCTION,
2587 ;*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
2588 ;*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).
2589 ;*****
2590
2591 ; TEST 22
2592 ;
2593 026464 012737 000022 001226 TST22: MOV #22,TSTNO
2594 026472 012737 003364 001216 MOV #.EOP,NEXT
2595 026500 012737 026524 001220 MOV #1$,LOCK
2596 ;R1 CONTAINS BASE DMC11 ADDRESS
2597 026506 104416 MSTCLR ;MASTER CLEAR DMC11
2598 026510 032737 100000 001366 BIT #BIT15,STAT1 ;IS IT CRAM?
2599 026516 001057 BNE 6$+2 ;SKIP TEST IF YES
2600 026520 004737 027014 JSR PC,MAPCK ;CHECK FOR HI OR LO
2601 026524 15$:
2602 026524 004737 026660 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2603 026530 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2604 026532 100400 100400 ;START AT ROM PC=0
2605 026534 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2606 026536 117777 114377! <400*7> JUMP TO ROM PC OF 1777
2607 026540 004737 026752 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2608 026544 000002 2 ;INDEX
2609 026546 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2610 026550 001401 BEQ 2$ ;BR IF YES
2611 026552 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2612 026554 104401 2$: SCOPI ;LOOP TO 1$ IF SW09=1

```


CZDMGC.P11

05-JAN-78 11:08

CROM JUMP TESTS

```

2613 026556 012737 026564 001220      MOV      #3$,LOCK      ;NEW SCOPI
2614 026564
2615 026564 004737 026660      3$:     JSR      PC,CLRALL ;CLEAR ALL CONDITIONS
2616 026570 104414      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2617 026572 100403      ROMCLK ;START AT ROM PC=3
2618 026574 104414      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2619 026576 103400      100000! <400*7> ;JUMP TO ROM PC OF 0
2620 026600 004737 026752      JSR      PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2621 026604 000010      10 ;INDEX
2622 026606 020504      CMP      R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2623 026610 001401      BEQ      4$ ;BR IF YES
2624 026612 104006      HLT      6 ;ERROR, CROM PC IS WRONG
2625 026614 104401      4$:     SCOPI ;LOOP TO 3$ IF SW09=1
2626 026616 012737 026624 001220      MOV      #5$,LOCK ;NEW SCOPI
2627 026624
2628 026624 004737 026660      5$:     JSR      PC,CLRALL ;CLEAR ALL CONDITIONS
2629 026630 104414      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2630 026632 100406      ROMCLK ;START AT ROM PC=6
2631 026634 104414      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2632 026636 107525      104125! <400*7> ;JUMP TO ROM PC OF 525
2633 026640 004737 026752      JSR      PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2634 026644 000016      16 ;INDEX
2635 026646 020504      CMP      R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
2636 026650 001401      BEQ      6$ ;BR IF YES
2637 026652 104006      HLT      6 ;ERROR, CROM PC IS WRONG
2638 026654 104401      6$:     SCOPI ;LOOP TO 5$ IF SW59=1
2639 026656 104400      SCOPE ;SCOPE THIS TEST
2640
2641
2642 ;SUBROUTINES
2643 ;-----
2644
2645 026660      CLRALL: ;THIS SUBROUTINE CLEARS THE C&Z BITS AND THE BR
2646
2647
2648 026660 104414      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2649 026662 000400      000400 ;BR+0
2650 026664 104414      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2651 026666 063220      063220 ;SP(0)+BR
2652 026670 104414      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2653 026672 060400      060400 ;BR+SP(0)+BR
2654 026674 000207      RTS      PC
2655
2656
2657 026676      SETBRO: ;THIS SUBROUTINE SETS BRO BIT
2658
2659
2660 026676 104414      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2661 026700 000401      000401 ;BR+001
2662 026702 000207      RTS      PC
2663
2664
2665 026704      SETBRI: ;THIS SUBROUTINE SETS BRI BIT
2666
2667
2668 026704 104414      ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

M03

CZDMGD.P11 05-JAN-78 11:08

SUBROUTINES

2669	026706	000402			000402 ;BR+002 RTS PC
2670	026710	000207			
2671					
2672					
2673	026712				SETBR4: ;THIS SUBROUTINE SETS BR4 BIT
2674					
2675					
2676	026712	104414			ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 000420 ;BR+020 RTS PC
2677	026714	000420			
2678	026716	000207			
2679					
2680					
2681	026720				SETBR7: ;THIS SUBROUTINE SETS BR7 BIT
2682					
2683					
2684	026720	104414			ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 000600 ;BR+200 RTS PC
2685	026722	000600			
2686	026724	000207			
2687					
2688					
2689	026726				SETC: ;THIS SUBROUTINE SETS THE C BIT
2690					
2691					
2692	026726	104414			ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 000777 ;BR+377
2693	026730	000777			ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 063220 ;SP(0)+BR ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 060400 ;BR+SP(0)+BR RTS PC
2694	026732	104414			
2695	026734	063220			
2696	026736	104414			
2697	026740	060400			
2698	026742	000207			
2699					
2700					
2701	026744				SETZ: ;THIS SUBROUTINE SETS THE Z BIT
2702					
2703					
2704	026744	104414			ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304 000777 ;BR+377 RTS PC
2705	026746	000777			
2706	026750	000207			
2707					
2708					
2709	026752				ROMDAT: ;THIS SUBROUTINE LOADS R5 WITH EXPECTED ROM CONTENTS ;AND LOADS R4 WITH ACTUAL ROM CONTENTS
2710					
2711					
2712					
2713	026752	017600	000000		MOV 2(SP),R0 ;INDEX FOR COMPARE
2714	026756	062716	000002		ADD #2,(SP) ;ADJUST STACK
2715	026762	012711	002000		MOV #BIT10,(R1) ;SET ROMO
2716	026766	016005	012324		MOV LOMAP(R0),R5 ;PUT EXPECTED IN R5 (LOSPEED)
2717	026772	032737	000002	001372	BIT #BIT1,STAT3 ;LOW OR HIGH SPEED?
2718	027000	001402			BEG 1\$;BR IF LOW SPEED
2719	027002	016005	016324		MOV HIMAP(R0),R5 ;PUT EXPECTED IN R5 (HISPEED)
2720	027006	016104	000006		MOV 6(R1),R4 ;PUT "FOUND" IN R4
2721	027012	000207			RTS PC ;RETURN
2722					
2723	027014				MAPCK: ;THIS SUBROUTINE CHECKS THE STATUS TABLE AND LOADS
2724					

CZDMGD.P11 05-JAN-78 11:08

SUBROUTINES

```

2725 ;THE ROMMAP POINTER TO POINT TO EITHER THE HIGH OR
2726 ;LOW SPEED MICRO-CODE.
2727
2728 027014 012737 012324 012322 MOV #LOMAP,ROMMAP ;LOAD POINTER TO LOW SPEED
2729 027022 032737 000002 001372 BIT #BIT1,STAT3 ;CHECK STATUS TABLE
2730 027030 001403 BEQ 1% ;BR IF LOW SPEED
2731 027032 012737 016324 012322 MOV #HIMAP,ROMMAP ;LOAD POINTER TO HIGH SPEED
2732 027040 000207 1%: RTS PC ;RETURN
2733
2734
027042 041777 040522 020115 EM1: .ASCIZ <377>/CRAM DATA ERROR/
027063 377 051103 046501 EM2: .ASCIZ <377>/CRAM DUAL ADDRESSING ERROR/
027117 377 051103 046517 EM3: .ASCIZ <377>/CRAM DATA ERROR/
027140 045377 046525 020120 EM4: .ASCIZ <377>/JUMF ERROR/
027154 047777 052104 042440 EM5: .ASCIZ <377>/ODT ERROR IN IBUS* REG10/
027206 044777 050117 046440 EM7: .ASCIZ <377>/ICP MAR TEST/
027224 041377 020122 044522 EM10: .ASCIZ <377>/GR RIGHT SHIFT TEST/
027251 377 042522 042503 EM11: .ASCIZ <377>/RECEIVE DATA ERROR/
027275 377 051106 042505 EM12: .ASCIZ <377>/FREE RUNNING ERROR/
027321 377 047503 052116 EM13: .ASCIZ <377>/CONTROL OUT ERROR/

027344 042777 050130 041505 DH1: .ASCIZ <377>/EXPECTED FOUND ADDRESS/
027376 042777 050130 041505 DH2: .ASCIZ <377>/EXPECTED FOUND/
027417 377 051440 046105 DH3: .ASCIZ <377>/SEL4 SEL6/
.EVEN

027440 000003 DT1: 3
027442 006 004 .BYTE 6,4
027444 001264 SAVR2
027446 006 004 .BYTE 6,4
027450 001277 SAVR4
027452 004 002 .BYTE 4,2
027454 001260 SAVR0
027456 000003 DT2: 3
027460 006 004 .BYTE 6,4
027462 001272 SAVR5
027464 006 004 .BYTE 6,4
027466 001270 SAVR4
027470 004 002 .BYTE 4,2
027472 001264 SAVR2
027474 000003 DT3: 3
027476 006 004 .BYTE 6,4
027500 001272 SAVR5
027502 006 004 .BYTE 6,4
027504 001270 SAVR4
027506 004 002 .BYTE 4,2
027510 001252 TEMP3
027512 000002 DT4: 2
027514 003 007 .BYTE 3,7
027516 001272 SAVR5
027520 003 002 .BYTE 3,2
027522 001270 SAVR4
027524 000002 DT5: 2
027526 006 004 .BYTE 6,4
027530 001272 SAVR5
027532 006 002 .BYTE 6,2

```

02DMGD.P11 05-JAN-78 11:08

SUBROUTINES

027534 001270
027536 000003
027540 003 010
027542 001272
027544 003 004
027546 001270
027550 004 002
027552 001264
027554 000003
027556 003 007
027560 001272
027562 003 004
027564 001270
027566 006 002
027570 001252
027572 000002
027574 006 004
027576 001252
027600 006 002
027602 001254

SAVR4
DT7: 3
.BYTE 3,10
SAVR5
.BYTE 3,4
SAVR4
.BYTE 4,2
SAVR2
DT10: 3
.BYTE 3,7
SAVR5
.BYTE 3,4
SAVR4
.BYTE 6,2
TEMP3
DT11: 2
.BYTE 6,4
TEMP3
.BYTE 6,2
TEMP4

027604
027604 000000
027606 000000
027610 000000
027612 027042
027614 027344
027616 027440
027620 027063
027622 027344
027624 027440
027626 027042
027630 027344
027632 027456
027634 027117
027636 027344
027640 027474
027642 027140
027644 027376
027646 027512
027650 027140
027652 027376
027654 027524
027656 027154
027660 027376
027662 027512
027664 000000
027666 000000
027670 000000
027672 027206
027674 027344
027676 027536
027700 027224
027702 027376
027704 027512
027706 027251

.ERRTAB:
0
0
0
EM1
DH1 :HLT 1
DT1
EM2
DH1 :HLT 2
DT1
EM1
DH1 :HLT 3
DT2
EM3
DH1 :HLT 4
DT3
EM4
DH2 :HLT 5
DT4
EM4
DH2 :HLT 6
DT5
EM5
DH2 :HLT 7
DT4
0
0
0
EM7
DH1 :HLT 11
DT7
EM10
DH2 :HLT 12
DT4
EM11

CZDMGD.P11 05-JAN-78 11:08

027710 027344
027712 027554
027714 027275
027716 000000
027720 000000
027722 027275
027724 027376
027726 027524
027730 027321
027732 027417
027734 027572

027736 000001

SUBROUTINES

DH1 ;HLT 13
DT10
EM12
O ;HLT 14
O
EM12
DH2 ;HLT 15
DT5
EM13
DH3 ;HLT 16
DT11

CORMAX:
.END

02DMGD.P11 05-JAN-78 11:08

CROSS REFERENCE TABLE -- USER SYMBOLS

MODU	006705	1187#	1453												
MPASSX	006107	741	1187#												
MPFAIL	005675	1121	1187#												
MQM	005666	861	1187#	1352	1427	1437	1447	1462	1476						
MR	005756	723	1187#	1346											
MRESET=	004000	96#													
MSYCLR=	104412	235#	1126	1738	1739	1803	1853	1910	1964	2021	2078	2135	2192	2249	
		2307	2365	2423	2481	2539	2597								
MTITLE	001000	136#	511												
MTSTN	006131	1059	1187#	1330											
MTSTPC	006032	1187#													
MVECX	006101	739	1187#												
NEXT	001216	157#	799	1092	1686*	1709*	1736*	1800*	1850*	1907*	1961*	2018*	2075*	2132*	
		2189#	2246#	2304*	2362*	2420*	2478*	2536*	2594*						
NOACT	007155	523	1187#	1282											
NODEV	002674	577	616#												
NUM	006451	1187#	1382												
OK	002646	603	610#	639											
ONE	001302	187#													
PACT00	001702	365#													
PACT01	001706	368#													
PACT02	001712	371#													
PACT03	001716	374#													
PACT04	001722	377#													
PACT05	001726	380#													
PACT06	001732	383#													
PACT07	001736	386#													
PACT10	001742	389#													
PACT11	001746	392#													
PACT12	001752	395#													
PACT13	001756	398#													
PACT14	001762	401#													
PACT15	001766	404#													
PACT16	001772	407#													
PACT17	001776	410#													
PARAM =	104405	225#	1331	1383	1396	1405	1484	1493							
PARAM1	004234	881#	898												
PARBIT=	040000	96#													
PARERR	004310	884	886	888	897#	904	906	908							
PASCNT	001230	162#	734*	735	746	771	1304*								
PERFOR=	004537	96#													
PFTAB	005430	1122	1128#												
POPRO =	012600	72#	1086												
POP1SP=	005726	70#													
POP2SP=	022626	74#	801												
PRI0	006550	1187#	1413												
PS =	177776	63#	476#	711*	1617*	1627*									
PUSHRO=	010046	71#	1083												
PUSH1S=	005746	69#													
PUSH2S=	024646	73#													
QV.FLG	001327	204#	482*	750*	790										
RESREG	005220	1075	1078#												
RESTAR	005350	1108	1114#												
RESTRY	003534	749	753	761#											
RESOS =	104407	229#	1078												
RETURN	001214	156#	492*	720*	724	761*	799*	803	1092*	1095	1127	1345*	1355*	1357	

CZDMGD.P11 05-JAN-78 11:08

CROSS REFERENCE TABLE -- USER SYMBOLS

.INSTE	004154	224	861#		
.ISTR	004050	222	840#		
.INSTI	004070	844#	864		
.MSG	004072	842*	845#		
.MSTCL	005466	236	1143#		
.PARAM	004174	226	872#		
.PFAIL	005336	113	478	1107#	1115
.RES05	004434	230	943#		
.ROMCL	005504	240	1148#		
.SAV05	004374	228	929#		
.SCOPE	003576	216	779#		
.SCOPI	003736	218	811#		
.START	002002	132	476#	492	1247
.TIMER	005616	244	1170#		
.TRPSR	004716	117	1014#		
.TRPTA	001330	214#	1019		
.TYPE	003766	220	822#		

CZDMGD.P11 05-JAN-78 11:08

CROSS REFERENCE TABLE -- MACRO NAMES

DMEND	1#	725													
DMFRNT	1#														
HLT	75#	1754	1764	1827	1867	1880	1893	1923	1935	1947	1978	1991	2004	2035	2048
	2061	2092	2105	2118	2149	2162	2175	2206	2219	2232	2263	2276	2289	2321	2334
	2347	2379	2392	2405	2437	2450	2463	2495	2508	2521	2553	2566	2579	2611	2624
	2637														
\$AUTO	1#	547													
\$BRRSH	1#	1725													
\$BUFFE	1#	1199													
\$COMP	1#														
\$SCRAM	1#	1694													
\$CYCLE	1#	1271													
\$EOP	1#	725													
\$FINI	1#	2640													
\$GETPA	1#														
\$HEADE	1#														
\$JUMP	1#	1838	1896	1950	2007	2064	2121	2178	2235	2292	2350	2408	2466	2524	2582
\$MARHI	1#														
\$MOCK	1#														
\$MSG	1#	1187													
\$PFAIL	1#	1103													
\$QUEST	1#	1381	1394	1403	1482	1491									
\$RAMCL	1#	1131													
\$RCLK	1#	1134	1137	1174	1179	1744	1746	1748	1755	1758	1814	1859	1861	1872	1874
	1885	1887	1915	1917	1927	1929	1939	1941	1970	1972	1983	1985	1996	1998	2027
	2029	2040	2042	2053	2055	2084	2086	2097	2099	2110	2112	2141	2143	2154	2156
	2167	2169	2198	2200	2211	2213	2224	2226	2255	2257	2268	2270	2281	2283	2313
	2315	2326	2328	2339	2341	2371	2373	2384	2386	2397	2399	2429	2431	2442	2444
	2455	2457	2487	2489	2500	2502	2513	2515	2545	2547	2558	2560	2571	2573	2603
	2605	2616	2618	2629	2631	2648	2650	2652	2660	2668	2676	2684	2692	2694	2695
	2704														
\$ROMNU	1#	1673													
\$ROMRD	1#	1766													
\$SCOPE	1#	775													
\$SIMBC	1#														
\$SOFTC	1#	1207													
\$TRPDE	1#	215	217	219	221	223	225	227	229	231	233	235	237	239	241
	243														
\$TSTN	1#	1683	1706	1733	1797	1847	1904	1958	2015	2072	2129	2186	2243	2301	2359
	2417	2475	2533	2591											
\$VARIA	1#	134													
\$XZ	1#	1673	1681	1694	1704	1725	1731	1766	1795	1838	1845	1896	1902	1950	1956
	2007	2013	2064	2070	2121	2127	2178	2184	2235	2241	2292	2299	2350	2357	2408
	2415	2466	2473	2524	2531	2582	2589								

. ABS. 027736 000

ERRORS DETECTED: 0

CZDMGD,CZDMGD/SOL/CRF+IPLUT1,CZDMGD
 RUN-TIME: 11 16 1 SECONDS
 RUN-TIME RATIO: 496/28=17.3
 CORE USED: 21K (41 PAGES)

N04

CZDMGD.P11

05-JAN-78 11:08

CROSS REFERENCE TABLE -- MACRO NAMES

PAGE: 0259CZ
SEQ 0259

