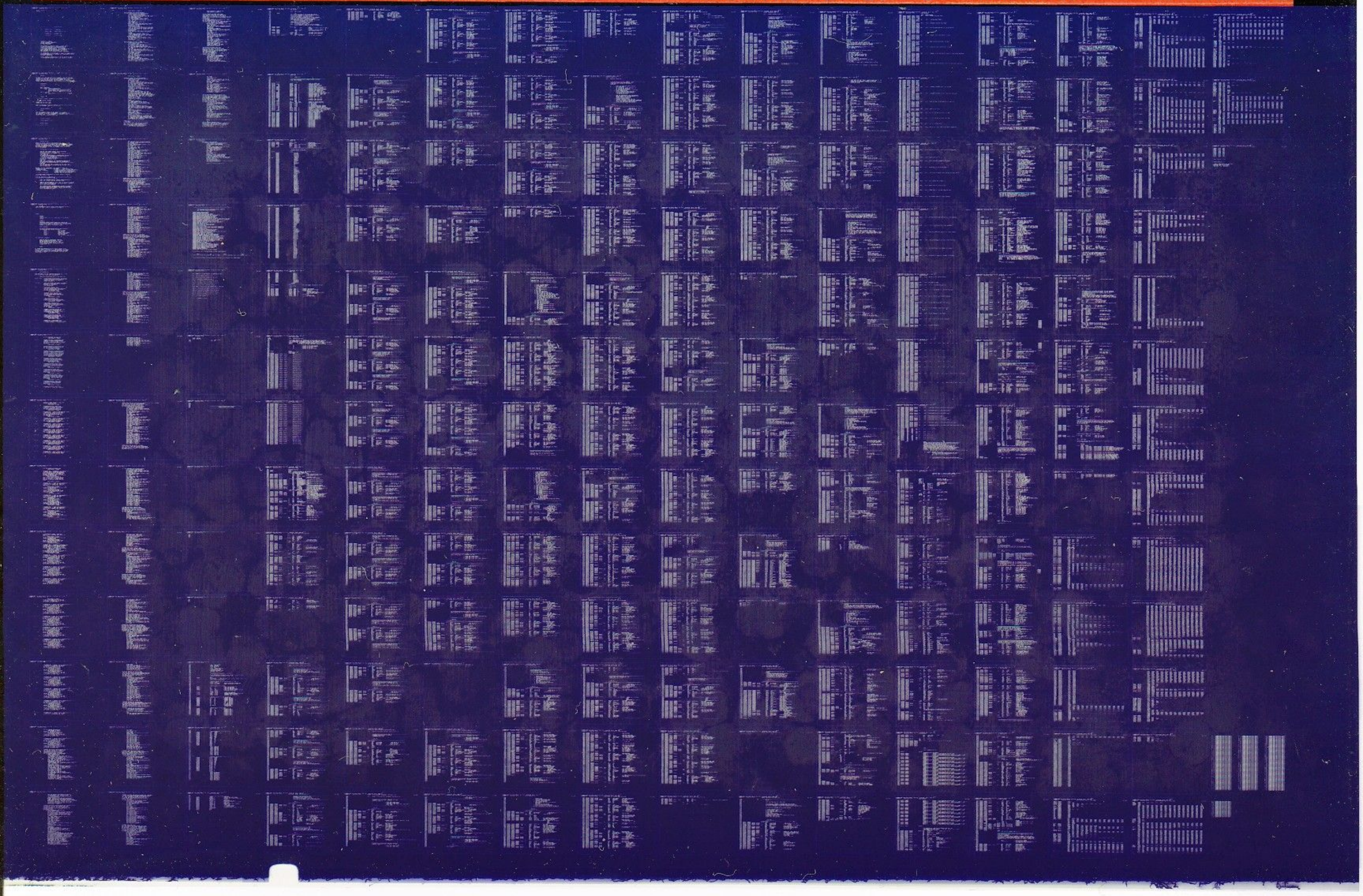
DMS11-DA

DMS11-DA STATIC
CZKMDAO

AH-S883A-MC
FICHE 1 OF 1



```
        .NLIST  SEQ,BIN,LOC
        .REM %
```

IDENTIFICATION

PRODUCT CODE: AC-S881A-MC

PRODUCT NAME: CZKMDA0 DMS11-DA STATIC

PROGRAM DATE: SEPTEMBER 1981

MAINTAINER: CSS/NPG DIAGNOSTICS

1.0 ABSTRACT

THE DMS11,-D,-A,-DA LINE UNIT STATIC DIAGNOSTIC TEST THE LINE UNIT
FOR BOTH CCP & BOP MODES IN THE USYRT MAINTENANCE MODE. THIS DIAGNOSTIC
EXERCISES THE LINE UNIT WITHOUT RESIDENT FIRMWARE IN THE KMC11
GENERAL PURPOSE MICROPROCESSOR. THIS DIAGNOSTIC IS A STAND ALONE
PACKAGE AND SHOULD BE EXERCISED PRIOR TO EXERCISING THE DYNAMIC DIAGNOSTIC

2.0 REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
KMC11 GENERAL PURPOSE MICROPROCESSOR
NOTE: THIS DIAGNOSTIC WILL TEST UP TO 16 KMC11
GENERAL PURPOSE MICROPROCESSOR'S, BUT THEY
MUST BE INSTALLED SEQUENTIALLY ON THE BUSS
DMS11-D LINE UNIT
NOTE: THIS DIAGNOSTIC WILL TEST UP TO 8 LINES PER
KMC11

2.2 STORAGE

THIS DIAGNOSTIC NEEDS A MINIMUM OF 16K OF MEMORY LOCATIONS

2.3 PRELIMINARY PROGRAMS

CZKMBA0 KMC11-B STATIC PART1
CZKMCA0 KMC11-B STATIC PART2

PRIOR TO EXERCISING THE STATIC DIAGNOSTIC THE KMC11 MUST BE ERROR
FREE. THIS DIAGNOSTIC DOES NOT TEST THE KMC11

3.0 LOADING PROCEDURE

3.1 METHOD

LOAD THE DIAGNOSTIC USING THE ABS LOADER. THIS DIAGNOSTIC IS
SELF STARTING. TO RESTART THE DIAGNOSTIC LOAD ADDRESS 200 (8)
AND DEPRESS START. IF THE DIAGNOSTIC IS ON MAGNETIC MEDIA, FOLLOW THE
INSTRUCTIONS FOR THE MONITOR BEING USED.

4.0 STARTING PROCEDURE

4.1 STARTING ADDRESS (200)

4.2 RESTART ADDRESS

RESTART ADDRESS 200 (8)

4.3 OPERATOR ACTION

DURING THE INITIAL START UP THE DIAGNOSTIC WILL OUTPUT TO THE TELETYPE THE TITLE OF THE DIAGNOSTIC AND ENTER THE MONITOR. DURING A RESTART THE DIAGNOSTIC WILL NOT OUTPUT THE TITLE OF THE DIAGNOSTIC BUT ENTER THE MONITOR DIRECTLY. AFTER ENTRY INTO THE DIAGNOSTIC MONITOR THE OPERATOR CAN ENTER THE COMMANDS LISTED IN SECTION 4.4 OF THIS WRITEUP.

5.0 STARTING AND TEST INITIALIZATION:

LOAD AND START MEMORY ADDRESS 200 BY PRESCRIBED METHOD AS PER PDP-11 PROCESSOR TYPE.
THE CONSOLE DEVICE WILL IDENTIFY THE DIAGNOSTIC ON THE FIRST START UP.
THE CONSOLE MONITOR ROUTINE IS ENTERED EACH TIME ON START UP AND OFFERS THE OPERATOR THE FOLLOWING CHOICES AS
(1) DIAG. TEST (2) CONFIGURE
TO WHICH THE OPERATOR MUST RESPOND WITH A 1 OR 2 FOLLOWED BY A CARRIAGE RETURN.

(1) DIAG. TEST
DIAG. TEST IS ENTERED IF THE CONFIGURATION ROUTINE HAS BEEN PREVIOUSLY ENTERED AND IF NOT WILL AUTOMATICALLY GO INTO THE CONFIGURATION ROUTINE TO SPECIFY KMC11(S) STATUS.

(2) CONFIGURE
WHEN THE CONFIGURE MODE IS ENTERED FOR THE FIRST TIME THE CONFIGURE TABLE IS ENTERED FROM THE TOP AND THE FIRST KMC11 IS SPECIFIED

```
DEV.ADR.= (6 OCTAL CHARACTERS)  ;BUS ADDRESS OF KMC11 CSR
LINES(0-7)(I.E..#,#,#) OR (A)=ALL(N)=NONE       ;LINES TO BE TESTED
(1)ENABLE(0)DISABLE             ;TO ENABLE OR DISABLE THIS UNIT FROM TESTING
(0)EXTERNAL (1)INTERNAL         ;TO ENABLE RUNNING TESTS REQUIRING THE
                                ; EXTERNAL DATA LOOP (EXT.LOOP CONNECTOR)
(1) CRC32  (0) NO CRC32         ; DMS11-DA, DMS11-A HAVE NO CRC32
```

* IF STATUS IS NOT TO BE CHANGED ON CURRENT INQUIRY A CARRIAGE RETURN (ONLY) MAY BE TYPED

AFTER THE INPUTTING IS COMPLETED AN "*" IS PRINTED AT WHICH POINT THE OPERATOR HAS 3 CHOICES BY TYPING AN

"E" (EXIT) THE CONFIGURE ROUTINE (ENTERS CONSOLE ROUTINE OR DIAG. TEST IF DIAG. TEST WAS THE ENTRY POINT

"M" (MODIFY) AN M FOLLOWED BY A #1-16 (DEC) TO SPECIFY THE DESIRED KMC11'S STATUS TO BE MODIFIED. THE STATUS BLOCK IS PRINTED AND THE NEW STATUS INVOLKED.

"A" (ALL) FROM LAST KMC11 PARAMETER BLOCK POINTED TO EITHER BY THE INITIAL ENTRY OR THE "MODIFY" ROUTINE THE REMAINING BLOCKS ARE LOADED WITH SEQUENTIAL DEVICE STATUS TO THE END OF THE CONFIGURATION TABLE.

* ANY COMBINATION OF DEVICE STATUS CAN BE CONFIGURED AND RUN AS LONG AS AT LEAST 1 KMC11 IS ENABLED IN THE TABLE

CONTROL:

CONTROL C

TYPING A CNTR C (^C) WILL FORCE ENTRY TO THE MONITOR ROUTINE.

CONTROL G

PROCESSORS WITHOUT CONSOLE PANELS WILL HAVE A SOFT SWITCH REGISTER WHICH IS INVOLKED BY THE OPERATOR TYPING A CNTR G (^G). AFTER THE NEW SWITCH VALUE IS ENTERED THE PROGRAM IS RESUMED AT THE POINT OF INVOLK.

5.1 SWITCH OPTIONS:

THE SWITCH REGISTER SETTING MUST BE DONE ACCORDING TO PROCESSOR TYPE AND HARDWARE ARCHITECT.

```
BIT15 = 1          ;HALT ON ERROR
BIT14 = 1          ;LOOP ON TEST
BIT13 = 1          ;INHIBIT ERROR TYPEOUTS
BIT10 = 1          ;RING BELL ON ERROR
BIT09 = 1          ;LOOP ON ERROR
BIT08 = 1          ;LOOP ON TEST IN SWR BITS 07-00
```

TEST EXECUTION + SEQUENCES:

EACH TEST MUST BE SUCCESSIVELY COMPLETED BEFORE THE NEXT CAN BE ENTERED. ALL TESTS ARE RUN SEQUENTIALLY ON EACH UNIT (KMC + DMS11D) AND THE CURRENT DEVICE IS SPECIFIED BY THE SEQUENCE OF THE CONFIGURATION TABLE (ENABLED UNITS ONLY).

6.0 ERRORS

ALL ERRORS ARE REPORTED ON THE TTY UNLESS CONSOLE SWITCH 13 IS SET (REFER TO SECTION 5.1 OF THIS WRITEUP). ALL ERRORS REPORTED HAVE A ENGLISH TYPE STATEMENT EXPLAINING THE ERROR WITH OUTPUT OF CERTAIN REGISTERS OR MEMORY LOCATIONS WHICH CAN BE USED AS A AID.

7.0 TEST DESCRIPTION

EACH INDIVIDUAL TEST HAS A WRITEUP EXPLAINING WHAT IT DOES
REFER TO THE TEST ITSELF FOR A EXPLANATION.

SUBROUTINE DESCRIPTIONS USED IN TEST WRITE UPS

MASTER CLEAR: SETTING MASTER CLEAR ON KMC11 (BIT OF SEL)
CLEARING CONTROL AND STATUS REGISTERS AND
LOADING THE CURRENT LINE NUMBER IN
PRIMARY REGISTER 11.

LINE RESET: LOADS THE CURRENT LINE NUMBER (IN PRIMARY
REGISTER 11) THEN SETS BIT 0 IN PRIMARY
REGISTER 12 (LINE RESET).
TEST DESCRIPTIONS

TEST 1 : (LINE SELECT REGISTER TEST)
THIS TEST ISSUES A RESET PULSE AND CLEARS
THE CSRS THEN READS AND TEST PRIMARY
REGISTER 11 FOR BEING CLEAR
ERROR REPORT IS ERROR 1

TEST 2 : (SECONDARY SELECT REGISTER TEST)
THIS TEST READS PRIMARY REGISTER 12
FOR BEING CLEAR AFTER A RESET PULSE
IS EXECUTED.
ERROR REPORT IS ERROR 1

TEST 3 : (LINE STATUS REGISTER TEST)
A RESET PULSE IS ISSUED AND PRIMARY
REGISTER 13 IS TESTED FOR BEING CLEARED.
ERROR REPORT IS ERROR 1

TEST 4 : (LINE CONTROL REGISTER TEST)
A RESET PULSE IS ISSUED AND PRIMARY
REGISTER 14 IS TESTED FOR BEING CORRECTLY
INITIALIZED. BITS 1 AND 2
ARE NOT TESTED.
ERROR REPORT IS ERROR 1

TEST 5 : (TRANSMIT STATUS MULTIPLEXER REG. TEST)
A RESET PULSE IS ISSUED AND THE
CORRESPONDING LINE (CURRENTLY UNDER TEST)
IS CHECKED FOR BEING CLEAR.
ERROR REPORT IS ERROR 1

TEST 6 : (RECEIVE STATUS MULTIPLEXER REG. TEST)
A RESET PULSE IS ISSUED AND THE
CORRESPONDING LINE (CURRENTLY UNDER TEST)
IS CHECKED FOR BEING CLEAR.
ERROR REPORT IS ERROR 1

TEST 7 : (PRIMARY REG. 11 WRITE/READ TEST)
A ONE BIT IS WALKED THROUGH BITS 6,5 AND 4
OF PRIMARY REGISTER 11. THE BITS ARE

WRITTEN THEN READ FOR BEING SET.
ERROR REPORT IS ERROR 2

TEST 10: (PRIMARY REGISTER 12 WRITE/READ TEST)
A ONE BIT IS WALKED THROUGH PRIMARY
REGISTER 12, BITS 4,5,6 AND 7. THE BITS
ARE WRITTEN THEN READ AND CHECKED FOR
BEING SET.
ERROR REPORT IS ERROR 2

TEST 11: (PRIMARY REGISTER 13 READ/WRITE TEST)
BIT 3 OF PRIMARY REGISTER IS WRITTEN
AND READ IF CRC32 IS ENABLED.
ERROR REPORT IS ERROR 2

TEST12: (PRIMARY REGISTER 14 READ/WRITE TEST)
A ONE BIT IS WALKED THROUGH PRIMARY
REGISTER 14, BITS 0,4,5,6, AND 7. BIT
3 IS ALSO TESTED IF CRC32 IS ENABLED.
THE BITS ARE WRITTEN THEN READ AND
CHECKED FOR BEING SET AND CLEARED.
ERROR REPORT IS ERROR 2

TEST 13: (SECONDARY REGISTER 0 READ TEST)
A LINE RESET IS DONE AND SECONDARY
REGISTER 0 (RECEIVE BUFFER) IS READ
AND CHECKED FOR BEING CLEAR.
ERROR REPORT IS ERROR 3

TEST 14: (SECONDARY REGISTER 1 READ TEST)
A LINE RESET IS ISSUED AND SECONDARY
REGISTER 1 IS READ AND CHECKED FOR
BEING CLEAR.
ERROR REPORT IS ERROR 3

TEST 15: (SECOND REGISTER 2 READ TEST)
A LINE RESET IS ISSUED AND SECONDARY
REGISTER 2 IS READ AND CHECKED FOR
BEING CLEAR.
ERROR REPORT IS ERROR 3

TEST 16: (SECONDARY REGISTER 3 READ TEST)
A LINE RESET IS ISSUED AND SECONDARY
REGISTER 3 IS READ AND CHECKED FOR
BEING CLEAR. (BITS 6,5 AND 4 ARE
NOT CHECKED).
FRROR REPORT IS ERROR 3

TEST 17: (SECONDARY REGISTER 4 READ TEST)
A LINE RESET IS ISSUED SECONDARY
REGISTER 4 IS READ CHECKED FOR
BEING CLEAR.
ERROR REPORT IS ERROR 3

TEST 20: (SECONDARY REGISTER 5 READ TEST)
A LINE RESET IS ISSUED AND SECONDARY

```
                    REGISTER 5 (MODE CONTROL) IS READ AND
                    CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 3

        TEST 21:    (SECONDARY REGISTER 7 READ TEST)
                    A LINE RESET IS ISSUED AND SECONDARY
                    REGISTER 7 (CHARACTER LENGTH) IS
                    READ AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 3

        TEST 22:    (SECONDARY REG. 1 WRITE TO READ ONLY BIT 'RSOM')
                    A LINE RESET IS ISSUED AND THE 'RSOM'
                    (BIT 0) IS WRITTEN WITH A ONE BIT
                    THEN READ AND CHECKED FOR NOT BEING
                    SET (READ ONLY).
                            ERROR REPORT IS ERROR 4

        TEST 23:    (SECONDARY REG. 1 WRITE TO READ ONLY BIT 'REOM')
                    A LINE RESET IS ISSUED AND THE 'REOM'
                    (BIT 1) IS WRITTEN WITH A ONE BIT
                    THEN READ AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 4

        TEST 24:    (SECONDARY REG. 1 WRITE TO READ ONLY BIT 'RXGA')
                    A LINE RESET IS ISSUED AND THE 'RXGA'
                    (BIT 2) IS WRITTEN WITH A ONE BIT
                    THEN READ AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 4

        TEST 25:    (SECONDARY REG 1 WRITE TO READ ONLY BIT 'ROR')
                    A LINE RESET IS ISSUED AND THE 'ROR'
                    (BIT 3) IS WRITTEN WITH A ONE BIT
                    THEN READ AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 4

        TEST 26:    (SECONDARY REG. 1 WRITE TO READ ONLY BIT 'ABCA')
                    A LINE RESET IS ISSUED AND THE 'ABCA'
                    (BIT 4) IS WRITTEN WITH A ONE BIT THEN
                    READ AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 4

        TEST 27:    (SECONDARY REG. 1 WRITE TO READ ONLY BIT 'ABCB')
                    A LINE RESET IS ISSUED AND THE 'ABCB'
                    (BIT 5) IS WRITTEN WITH A ONE BIT THEN
                    READ AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 4

        TEST 30:    (SECONDARY REG. 1 WRITE TO READ ONLY BIT 'ABCC')
                    A LINE RESET IS ISSUED AND THE 'ABCC'
                    (BIT 6) IS WRITTEN WITH A ONE BIT THEN
                    READ AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 4

        TEST 31:    (SECONDARY REG. 1 WRITE TO READ ONLY BIT 'ERRCK')
                    A LINE RESET IS ISSUED AND THE 'ERRCK'
                    BIT 7) IS WRITTEN WITH A ONE BIT THEN
```

```
                    READ AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 4

TEST 32:            (SECONDARY REG. 2 INCREMENTAL DATA PATTERN TEST)
                    A MASTER CLEAR IS ISSUED AND SECONDARY
                    REG. 2 IS WRITTEN WITH THE CURRENT DATA
                    PATTERN (INCREMENTAL 1-377) THEN THE
                    REGISTER IS READ AND CHECK FOR CONTAINING
                    THE CURRENT DATA PATTERN.
                            ERROR REPORT IS ERROR 10
                    WHEN THE FULL DATA PATTERN IS REACHED
                    (377) A LINE RESET IS ISSUED AND THE
                    REGISTER IS READ AND CHECKED FOR BEING
                    CLEAR.
                            ERROR REPORT IS ERROR 11

TEST 33:            (SECONDARY REG. 3 "TSOM" BIT WRITE/READ/RESET TEST)
                    A LINE RESET IS ISSUED, THE "TSOM" BIT
                    (BIT 0) IS SET THEN READ AND CHECKED FOR
                    BEING SET.
                            ERROR REPORT IS ERROR 5
                    THE BIT IS THEN WRITTEN TO A 0, READ AND
                    CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 5
                    THE BIT IS THEN REWRITTEN AND A LINE
                    RESET ISSUED AND REREAD AND CHECKED
                    FOR BEING CLEARED.
                            ERROR REPORT IS ERROR 8

TEST 34:            (SECONDARY REG. 3 "TEOM" BIT WRITE/READ/RESET TEST)
                    A LINE RESET IS ISSUED, THE "TEOM" BIT
                    (BIT 1) IS SET AND THEN READ AND
                    CHECKED FOR BEING SET.
                            ERROR REPORT IS ERROR 5
                    THE BIT IS THEN WRITTEN TO A 0, READ
                    AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 5
                    THE BIT IS THEN REWRITTEN AND A
                    LINE RESET ISSUED AND THE BIT REREAD
                    AND CHECKED FOR BEING CLEARED.
                            ERROR REPORT IS ERROR 8

TEST 35:            (SECONDARY REG. 3 "TXAB" WRITE/READ/RESET TEST)
                    A LINE RESET IS ISSUED, THE "TXAB" BIT
                    (BIT 2) IS SET THEN READ AND CHECKED FOR
                    BEING SET.
                            ERROR REPORT IS ERROR 5
                    THE BIT IS THEN WRITTEN TO A 0, READ AND
                    CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 5
                    THE BIT IS THEN REWRITTEN AND A LINE
                    RESET ISSUED AND THE BIT REREAD AND CHECKED
                    FOR BEING CLEARED.
                            ERROR REPORT IS ERROR 8

TEST 36:            (SECONDARY REG. 3 "TXGA" WRITE/READ/RESET TEST)
```

```
                    A LINE RESET IS ISSUED, "TXGA" BIT
                    (BIT 3) IS SET THEN READ AND CHECKED FOR
                    BEING SET.
                            ERROR REPORT IS ERROR 5
                    THE BIT IS THEN WRITTEN TO A 0, READ AND
                    CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 5
                    THE BIT IS THEN REWRITTEN AND A LINE
                    RESET ISSUED AND THE BIT REREAD AND
                    CHECKED FOR BEING CLEARED.
                            ERROR REPORT IS ERROR 8

TEST 37:            (SECONDARY REG. 4 INCREMENTAL WRITE/READ TEST)
                    A MASTER CLEAR IS ISSUED AND AN INCREMENTAL
                    DATA PATTERN (1-377) IS WRITTEN TO THE
                    SYNC REG. (REG. 4) THEN READ AND CHECKED
                    TO BE THE SAME AS THE CURRENT DATA PATTERN.
                            ERROR REPORT IS ERROR 10
                    ON THE FINAL DATA PATTERN (377) A LINE RESET
                    IS ISSUED AND THE REGISTER IS REREAD
                    AND CHECKED FOR BEING CLEARED.
                            ERROR REPORT IS ERROR 11

TEST 40:            (SECONDARY REG. 5 "PROTEX" (BIT 0) WRITE/READ/RESET TEST)
                    A LINE RESET IS ISSUED AND THE "PROTEX"
                    BIT (BIT 0) IS WRITTEN TO A ONE THEN
                    READ AND CHECKED FOR BEING SET.
                            ERROR REPORT IS ERROR 6
                    THE BIT IS THEN WRITTEN TO A 0
                    REREAD AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 6
                    THE BIT IS THEN REWRITTEN TO A ONE AND
                    A LINE RESET IS ISSUED AND THE BIT REREAD
                    AND CHECKED FOR BEING CLEAR.

TEST 41:            (SECONDARY REG. 5 "PROTEY" (BIT 1) WRITE/READ/RESET TEST)
                    A LINE RESET IS ISSUED AND THE "PROTEY"
                    BIT (BIT 1) IS WRITTEN TO A ONE THEN
                    READ AND CHECKED FOR BEING SET.
                            ERROR REPORT IS ERROR 6
                    THE BIT IS THEN WRITTEN TO A 0
                    REREAD AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 6
                    THE BIT IS THEN REWRITTEN TO A ONE AND
                    A LINE RESET IS ISSUED AND THE BIT REREAD
                    AND CHECKED FOR BEING CLEAR.

TEST 42:            (SECONDARY REG. 5 "PROTEZ" (BIT 2) WRITE/READ/RESET TEST)
                    A LINE RESET IS ISSUED AND THE "PROTEZ"
                    BIT (BIT 2) IS WRITTEN TO A ONE THEN
                    READ AND CHECKED FOR BEING SET.
                            ERROR REPORT IS ERROR 6
                    THE BIT IS THEN WRITTEN TO A 0
                    REREAD AND CHECKED FOR BEING CLEAR.
```

```
                ERROR REPORT IS ERROR 6
        THE BIT IS THEN REWRITTEN TO A ONE AND
        A LINE RESET IS ISSUED AND THE BIT REREAD
        AND CHECKED FOR BEING CLEAR.

TEST 43:        (SECONDARY REG. 5 'NIDLE' (BIT 3) WRITE/READ/RESET TEST)
        A LINE RESET IS ISSUED AND THE 'NIDLE'
        BIT (BIT 3) IS WRITTEN TO A ONE THEN
        READ AND CHECKED FOR BEING SET.
                ERROR REPORT IS ERROR 6
        THE BIT IS THEN WRITTEN TO A 0
        REREAD AND CHECKED FOR BEING CLEAR.
                ERROR REPORT IS ERROR 6
        THE BIT IS THEN REWRITTEN TO A ONE AND
        A LINE RESET IS ISSUED AND THE BIT REREAD
        AND CHECKED FOR BEING CLEAR.

TEST 44:        (SECONDARY REG. 5 'SAM' (BIT 4) WRITE/READ/RESET TEST)
        A LINE RESET IS ISSUED AND THE 'SAM'
        BIT (BIT 4) IS WRITTEN TO A ONE THEN
        READ AND CHECKED FOR BEING SET.
                ERROR REPORT IS ERROR 6
        THE BIT IS THEN WRITTEN TO A 0
        REREAD AND CHECKED FOR BEING CLEAR.
                ERROR REPORT IS ERROR 6
        THE BIT IS THEN REWRITTEN TO A ONE AND
        A LINE RESET IS ISSUED AND THE BIT REREAD
        AND CHECKED FOR BEING CLEAR.

TEST 45:        (SECONDARY REG. 5 'SSL' (BIT 5) WRITE/READ/RESET TEST)
        A LINE RESET IS ISSUED AND THE 'SSL'
        BIT (BIT 5) IS WRITTEN TO A ONE THEN
        READ AND CHECKED FOR BEING SET.
                ERROR REPORT IS ERROR 6
        THE BIT IS THEN WRITTEN TO A 0
        REREAD AND CHECKED FOR BEING CLEAR.
                ERROR REPORT IS ERROR 6
        THE BIT IS THEN REWRITTEN TO A ONE AND
        A LINE RESET IS ISSUED AND THE BIT REREAD
        AND CHECKED FOR BEING CLEAR.

TEST 46:        (SECONDARY REG. 5 'PROTO' (BIT 6) WRITE/READ/RESET TEST)
        A LINE RESET IS ISSUED AND THE 'PROTO'
        BIT (BIT 6) IS WRITTEN TO A ONE THEN
        READ AND CHECKED FOR BEING SET.
                ERROR REPORT IS ERROR 6
        THE BIT IS THEN WRITTEN TO A 0
        REREAD AND CHECKED FOR BEING CLEAR.
                ERROR REPORT IS ERROR 6
        THE BIT IS THEN REWRITTEN TO A ONE AND
        A LINE RESET IS ISSUED AND THE BIT REREAD
        AND CHECKED FOR BEING CLEAR.
```

```
TEST 47:        (SECONDARY REG. 7 "RDATA1" (BIT 0) WRITE/READ/RESET TEST)
                A LINE RESET IS ISSUED AND THE "RDATA1"
                BIT (BIT 0) IS WRITTEN TO A ONE THEN
                READ AND CHECKED FOR BEING SET.
                        ERROR REPORT IS ERROR 6
                THE BIT IS THEN WRITTEN TO A 0
                REREAD AND CHECKED FOR BEING CLEAR.
                        ERROR REPORT IS ERROR 6
                THE BIT IS THEN REWRITTEN TO A ONE AND
                A LINE RESET IS ISSUED AND THE BIT REREAD
                AND CHECKED FOR BEING CLEAR.

TEST 50:        (SECONDARY REG. 7 "RDATA2" (BIT 1) WRITE/READ/RESET TEST)
                A LINE RESET IS ISSUED AND THE "RDATA2"
                BIT (BIT 1) IS WRITTEN TO A ONE THEN
                READ AND CHECKED FOR BEING SET.
                        ERROR REPORT IS ERROR 6
                THE BIT IS THEN WRITTEN TO A 0
                REREAD AND CHECKED FOR BEING CLEAR.
                        ERROR REPORT IS ERROR 6
                THE BIT IS THEN REWRITTEN TO A ONE AND
                A LINE RESET IS ISSUED AND THE BIT REREAD
                AND CHECKED FOR BEING CLEAR.

TEST 51:        (SECONDARY REG. 7 "RDATA3" (BIT 2) WRITE/READ/RESET TEST)
                A LINE RESET IS ISSUED AND THE "RDATA3"
                BIT (BIT 2) IS WRITTEN TO A ONE THEN
                READ AND CHECKED FOR BEING SET.
                        ERROR REPORT IS ERROR 6
                THE BIT IS THEN WRITTEN TO A 0
                REREAD AND CHECKED FOR BEING CLEAR.
                        ERROR REPORT IS ERROR 6
                THE BIT IS THEN REWRITTEN TO A ONE AND
                A LINE RESET IS ISSUED AND THE BIT REREAD
                AND CHECKED FOR BEING CLEAR.

TEST 52:        (SECONDARY REG. 7 "TDATA1" (BIT 5) WRITE/READ/RESET TEST)
                A LINE RESET IS ISSUED AND THE "TDATA1"
                BIT (BIT 5) IS WRITTEN TO A ONE THEN
                READ AND CHECKED FOR BEING SET.
                        ERROR REPORT IS ERROR 6
                THE BIT IS THEN WRITTEN TO A 0
                REREAD AND CHECKED FOR BEING CLEAR.
                        ERROR REPORT IS ERROR 6
                THE BIT IS THEN REWRITTEN TO A ONE AND
                A LINE RESET IS ISSUED AND THE BIT REREAD
                AND CHECKED FOR BEING CLEAR.

TEST 53:        (SECONDARY REG. 7 "TDATA2" (BIT 6) WRITE/READ/RESET TEST)
                A LINE RESET IS ISSUED AND THE "TDATA2"
                BIT (BIT 6) IS WRITTEN TO A ONE THEN
```

```
                    READ AND CHECKED FOR BEING SET.
                            ERROR REPORT IS ERROR 6
                    THE BIT IS THEN WRITTEN TO A 0
                    REREAD AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 6
                    THE BIT IS THEN REWRITTEN TO A ONE AND
                    A LINE RESET IS ISSUED AND THE BIT REREAD
                    AND CHECKED FOR BEING CLEAR.


TEST 54:            (SECONDARY REG. 7 "TDATA3" (BIT 7) WRITE/READ/RESET TEST)
                    A LINE RESET IS ISSUED AND THE "TDATA3"
                    BIT (BIT 7) IS WRITTEN TO A ONE THEN
                    READ AND CHECKED FOR BEING SET.
                            ERROR REPORT IS ERROR 6
                    THE BIT IS THEN WRITTEN TO A 0
                    REREAD AND CHECKED FOR BEING CLEAR.
                            ERROR REPORT IS ERROR 6
                    THE BIT IS THEN REWRITTEN TO A ONE AND
                    A LINE RESET IS ISSUED AND THE BIT REREAD
                    AND CHECKED FOR BEING CLEAR.

TEST 55:            (TRANSMIT AND RECEIVE INITIALIZATION TEST FOR CCP PROTOCOL)
                    THIS TEST WILL UTILIZE THE INTERNAL DATA
                    LOOP (LOOPING THE OUTPUT OF THE USYRT
                    TRANSMITTER TO THE USYRT RECEIVER)
                    AND THE ON BOARD SINGLE STEP MAINTENANCE CLOCK.
                    THE FOLLOWING PROGRAMING STEPS ARE DONE
                            ISSUE A MASTER CLEAR
                            SET PROTOCOL TO CCP (BIT 6-SEC. REG. 5)
                            LOAD SECONDARY SYNC REGISTER WITH A 26 (OCT.)
                            ENABLE TENA, RENA AND MAINT. CLK. BITS MB & MC
                            SET "TSOM" BIT 0-SECONDARY REG. 3
                            READ "TBMT" (BIT-PRI 13) AND TEST BIT IS CLEAR
                            AFTER SETTING "TSOM" (AND NO CLK PULSES)
                    *       ERROR REPORT IS ERROR 12
                            THE MAINTENANCE CLOCK IS STEPPED 2 (FULL) CLOCK TICKS.
                            "TBMT" IS REREAD AND CHECKED FOR BEING SET AFTER
                            THE 2 CLOCK TICKS.
                    *       ERROR REPORT IS ERROR 13
                            TRANSMIT DATA BUFFER (SECONDARY REGISTER 2) IS LOADED
                            WITH THE "STX" (START OF TEXT) CHARACTER (2).
                            THE MAINTENANCE CLOCK IS STEPPED 7 TICKS
                            THE "SYNC FLAG" RECEIVED IS READ (BITS 3-SEC 13)
                            AND CHECKED FOR BEING SET.
                    *       ERROR REPORT IS ERROR 15
                            THE "TSOM" BIT IS WRITTEN TO A 0
                            THE MAINTENANCE CLOCK IS STEPPED 8 TICKS
                            THE "SYNC FLAG" RECEIVED IS REREAD AND CHECKED
                            FOR BEING SET.
                    *       ERROR REPORT IS ERROR 15
                            THE MAINTENANCE CLOCK IS STEPPED 9 TICKS
                            THE "SYNC FLAG" RECEIVED IS READ AND CHECKED
                            FOR BEING CLEAR (CLOCKING IN A NON FLAG CHARACTER)
                    *       ERROR REPORT IS ERROR 16
                            A SERIES OF 3 DATA CHARACTERS (101) ARE LOADED
```

```
                        INTO THE TRANSMIT DATA BUFFER FOLLOWED BY
                        THE MAINTENANCE CLOCK STEPPED 8 TICKS
                        THE 4 DATA CHARACTER IS LOADED AND THE MAINT.
                        CLOCK STEPPED 2 TICKS.
                        PRIMARY REGISTER 13 IS READ AND STATUS BITS
                        0 AND 2 (RDA AND RAC) ARE CHECKED FOR BEING SET
                *       ERROR REPORT FOR "RAC" (RECEIVER ACTIVE) IS ERROR 17
                *       ERROR REPORT FOR "RDA" (RECEIVE DATA AVAIL.) IS ERROR 18
                        THE RECEIVE DATA BUFFER IS READ (SEC 0) AND
                        CHECKED FOR CONTAINING THE SYNC FLAG VALUE (26)
                *       ERROR REPORT IS ERROR 20

TEST 56:        (TRANSMIT AND RECEIVE INITIALIZATION TEST FOR BOP PROTOCOL)
                THIS TEST WILL UTILIZE THE INTERNAL DATA LOOP
                (LOOPING THE OUTPUT OF THE USYRT TRANSMITTER TO
                THE USYRT RECEIVER) AND THE ON BOARD SINGLE STEP
                MAINTENANCE CLOCK.
                THE FOLLOWING PROGRAMING STEPS ARE DONE
                        ISSUE A MASTER CLEAR
                        SET PROTOCOL TO BOP AND SECONDARY ADDRESS MODE
                        LOAD SYNC SECONDARY ADDRESS REG. WITH A 376
                        SET TRANSMIT ENABLE, RECEIVER ENABLE AND
                        INTERNAL DATA LOOP MODE (MB & MC) (PRIMARY
                        14-BITS 6 & 7)
                        SET "TSOM" TRANSMIT START OF MESSAGE (SEC 3-BIT 0)
                        TEST "TBMT" IS NOT SET
                *       ERROR REPORT IS ERROR 12
                        STEP CLOCK 2 TICKS
                        TEST "TBMT" IS NOW SET
                *       ERROR REPORT IS ERROR 13
                        CLEAR "TSOM" BIT
                        LOAD THE TRANSMIT DATA BUFFER SECONDARY SYNC
                        ADDRESS (376)
                        STEP MAINT CLOCK 7 TICKS
                        TEST THAT SYNC FLAG RECEIVED (PRI 13-BIT 3)
                        IS NOW SET.
                *       ERROR REPORT IS ERROR 14
                        THE TRANSMIT DATA BUFFER IS LOADED WITH A 0
                        THE MAINTENANCE CLOCK IS STEP 9 TICKS
                        THE SYNC FLAG RECEIVED BIT IS THEN TESTED
                        FOR BEING CLEAR
                *       ERROR REPORT IS ERROR 16
                        A SERIES OF 3 DATA CHARACTERS (101) ARE LOADED
                        INTO THE TRANSMIT DATA BUFFER AND EACH FOLLOWED
                        BY STEPPING THE MAINTENANCE CLOCK 8 TICKS
                        THE FOURTH DATA CHARACTER IS LOADED FOLLOWED
                        BY STEPPING THE MAINTENANCE CLOCK 2 TICKS
                        THE RECEIVER STATUS BITS "RAC" & "RDA" (PRI 13-BITS 2 & 0)
                        ARE READ AND CHECKED FOR BEING SET.
                *       ERROR REPORT FOR "RAC" IS ERROR 17
                *       ERROR REPORT FOR "RDA" IS ERROR 18
                        THE "RSOM" RECEIVE START OF MESSAGE BIT
                        (SEC 1-BIT 0) IS TESTED FOR BEING SET.
                *       ERROR REPORT IS ERROR 19
                        THE RECEIVE DATA BUFFER IS READ AND TESTED
                        FOR CONTAINING THE 376 (SYNC ADDRESS)
```

```
                    *       ERROR REPORT IS ERROR 20

TEST 57:            (END OF MESSAGE TEST FOR BOP PROTOCOL)
                    THIS TEST EXECUTES IN THE FOLLOWING SEQUENCE
                            ISSUES A MASTER CLEAR SEQUENCE
                            SET THE CHARACTER LENGTH REGISTER (SEC 7) A 0 (8 BIT))
                            SETS TENA, RENA AND INTERNAL DATA LOOP SINGLE
                            STEP MAINTENANCE CLOCK
                            SET "TSOM" TRANSMIT START OF MESSAGE
                            STEP THE MAINTENANCE CLOCK 1 TICK
                            TEST "TBMT" FOR SET AND CONTINUE STEPPING
                            CLOCK TILL SET
                            LOAD TRANSMIT DATA BUFFER WITH A 0
                            CLEAR "TSOM" BIT
                            RUN A STATUS CHECK LOOP (400 OCTAL) STEPPING
                            THE CLOCK 1 TICK AND MONITORING SYNC FLAG
                            RECEIVED FOR SETTING
                    *       ERROR REPORT IS ERROR 14
                            A LOOP TO MONITOR THAT RECEIVE DATA AVAILABLE
                            (RDA) WILL SET WITHIN 400 (OCTAL) CLOCK TICKS
                            WHILE ALSO MONITORING THE TRANSMITTER (TBMT)
                            TO SUPPLY DATA
                    *       ERROR REPORT IF "RDA" FAILS TO SET IS ERROR 34
                            AFTER "RDA" SETS THE RECEIVE DATA BUFFER
                            IS READ.
                            TRANSMIT END OF MESSAGE (TEOM) IS SET
                            A LOOP TO MONITOR THAT THE SYNC FLAG RECEIVED
                            (S/F) SETS WITHIN 400 (OCTAL) CLOCK TICKS.
                    *       ERROR REPORT IS ERROR 22
                            THE MAINTENANCE CLOCK IS STEPPED 2 TICKS
                            AND "REOM" RECEIVE END OF MESSAGE IS TESTED
                            FOR BEING SET.
                    *       ERROR REPORT IS ERROR 21.

TEST 60:            (END OF MESSAGE TEST FOR CCP MODE)
                    THIS TEST EXECUTES IN THE FOLLOWING SEQUENCE
                            ISSUE A MASTER CLEAR
                            SET CHARACTER LENGTH REGISTER TO 0 (8 BITS)
                            SET PROTOCOL TO CCP (SEC 5-BIT 6=1)
                            LOAD THE SYNC ADDRESS REGISTER WITH A 26
                            SET TENA, RENA AND INTERNAL SINGLE STEP MAINT CLOCK
                            SET "TSOM"
                            STEP CLOCK AND LOOP UNTIL "TBMT" SETS
                            LOAD THE TRANSMIT DATA BUFFER WITH AN STX (2)
                            TEST THAT SYNC FLAG RECEIVED (S/F) SETS WITHIN
                            400 (OCTAL) CLOCK TICKS
                    *       ERROR REPORT IS ERROR 15
                            CLEAR THE "TSOM" BIT
                            TEST THAT "RDA" SETS WITHIN 400 (OCTAL) CLOCK
                            TICKS (THE TRANSMITTER IS ALSO SERVICED WHILE
                            MONITORING THE "RDA")
                    *       ERROR REPORT IS ERROR 34
                            SET "TEOM" (TRANSMIT END OF MESSAGE)
                            TEST THAT SYNC FLAG RECEIVED (S/F) SETS WITHIN
                            400 (OCTAL) CLOCK TICKS. (THE RECEIVE IS ALSO
                            MONITOR TO PREVENT AN OVERRUN)
```

```
                        *       ERROR REPORT IS ERROR 22

        TEST 61:        (CHARACTER LENGTH TEST FOR BOP PROTOCOL)
                        THIS TEST EXECUTES IN THE FOLLOWING SEQUENCE
                                ISSUE A MASTER CLEAR
                                SET PROTOCOL TO BOP
                                SET CHARACTER LENGTH REG.(SEC 7)
                                RECEIVE BITS TO ACCEPT 7 BIT CHARACTER ALTERS
                                SET TENA, RENA AND INTERNAL SINGLE STEP MAINT CLOCK
                                STEP CLOCK UNTIL "TBMT" SETS
                                TEST THAT SYNC FLAG RECEIVED (S/F) SETS
                                WITHIN 400 (OCTAL) CLOCK TICKS
                        *       ERROR REPORT IS ERROR 14
                                A SEQUENCE IS RUN TO MONITOR THAT SYNC
                                FLAG RECEIVED (S/F) GOES AWAY (WITHIN 400(OCTAL)
                                CLOCK TICKS
                        *       ERROR REPORT IS ERROR 33
                                THE SEQUENCE MONITORS "TBMT" AND "RDA"
                                TO OUTPUT 2 DATA CHARACTERS THEN LOAD THE
                                DATA LENGTH REGISTER TO TRANSMIT AND RECEIVE
                                CHARACTERS AS SPECIFIED BY THE CURRENT CHARACTER
                                LENGTH UNDER TEST.  WHEN "RDA" SETS THE RECEIVE
                                DATA BUFFER IS READ AND THE DATA IS COMPARED
                                AGAINST THE DATA SENT.
                        *       ERROR REPORT IS ERROR 23
                                IF "RDA" FAILS TO SET WITHIN 400 (OCTAL) CLOCK
                                TICKS.
                        *       ERROR REPORT IS ERROR 34
                                SET "TEOM" TRANSMIT END OF MESSAGE
                                STEP MAINTENANCE CLOCK AND CHECK AFTER
                                EACH TICK THAT SYNC FLAG RECEIVER IS NOT SET,
                                UNTIL "RDA" SETS
                        *       ERROR REPORT IS ERROR 36 (S/F SET)
                        *       ERROR REPORT IS ERROR 34 (RDA FAILED TO SET)
                                WHEN "RDA" SETS THE DATA REGISTER IS REREAD
                                AND THE DATA COMPARED AGAINST THE TRANSMITTED
                                VALUE.
                        *       ERROR REPORT IS ERROR 23
                                "REOM" IS MONITORED FOR SETTING WITHIN 400
                                (OCTAL) CLOCK TICKS
                        *       ERROR REPORT IS ERROR 21
                                THE TEST IS RERUN FOR REMAINING CHARACTER
                                LENGTHS 6,5,4,3,2 & 1
                                THE DATA VALUE TRANSMITTED IS DECREASED ONE
                                BIT POSITION FOR EACH DATA CHARACTER LENGTH.
                        NOTE:   THE CHARACTER LENGTH FRAMING (TRANSMIT AND RECEIVE)
                                IS CHECKED BY THE 2 CONSECUTIVE DATA CHARACTERS
                                BEING TRANSMITTED AND RECEIVED.

        TEST 62:        (ERROR CHECK TEST FOR CCITT-CRC WITH INITIALIZE SET TO
                        -1 (BOP MODE))
                        THIS TEST WILL CHECK THAT THE CCITT POLYNOMIAL
                        (X16 + X12 + X15 + 1) WITH A PRESET OF -1
                        THERE IS NOT FORCED ERROR CONDITION IN THIS TEST.
                        THIS TEST PERFORMS THE FOLLOWING SEQUENCE
                                ISSUE A MASTER CLEAR SEQUENCE
```

```
                        SET CHARACTER LENGTH TO 8 BITS/CHARACTER
                        SET PROTOCOL TO BOP AND ENABLE CCITT-CRC16
                        WITH A PRESET OF -1
                        SET TENA, RENA AND ENABLE INTERNAL SINGLE
                        STEP MAINTENANCE CLOCK
                        SET ''TSOM''
                        STEP CLOCK UNTIL ''TBMT'' SETS
                        LOAD ADDRESS WORD (TDB)
                        CLEAR ''TSOM''
                        STEP CLOCK AND TEST SYNC FLAG RECEIVED
                        SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 14
                        STEP CLOCK, OUTPUT DATA (ON ''TBMT'') AND TEST
                        ''RDA'' SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORS IS ERROR 34
                        TEST THAT DATA READ IS SAME AS SENT
                *       ERROR REPORT IS ERROR 23
                        TEST THAT ''ERRCK'' BIT IS NOT SET
                *       ERROR REPORT IS ERROR 25
                        SET ''TEOM''
                        STEP CLOCK AND READ RECEIVED DATA WORDS
                        UNTIL SYNC FLAG RECEIVED SETS
                        TEST SYNC FLAG RECEIVED SETS WITHIN 400
                        (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 36
                        TEST ALL DATA RECEIVED WITHIN 400 (OCTAL)
                        CLOCK TICKS
                *       ERROR REPORT IS ERROR 34
                        TEST ALL DATA RECEIVED IS SAME AS SENT
                *       ERROR REPORT IS ERROR 23
                        TEST THAT ''ERRCK'' IS NOT SET
                *       ERROR REPORT IS ERROR 25
                        STEP CLOCK, READ DATA AND TEST SYNC FLAG
                        RECEIVED SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 22
                        STEP CLOCK 2 TICKS AND TEST ''REOM'' IS SET
                *       ERROR REPORT IS ERROR 21
                        TEST ''ERRCK'' IS NOT SET
                *       ERROR REPORT IS ERROR 25

TEST 63:        (ERROR CHECK TEST FOR CCITT-CRC WITH INITIALIZE SET TO 0 (BOP MODE))
                THIS TEST WILL CHECK THE CCITT POLYNOMIAL
                (X16 + X12 + X5 +1) WITH A PRESET OF 0.  THERE IS
                NO FORCED ERROR CONDITION IN THIS TEST.
                THIS TEST PERFORMS THE FOLLOWING SEQUENCE
                        ISSUE A MASTER CLEAR SEQUENCE
                        SET CHARACTER LENGTH TO 8 BITS/CHARACTER
                        SET PROTOCOL TO BOP AND ENABLE CCITT-CRC16
                        WITH A PRESET OF -1
                        SET TENA, RENA AND ENABLE INTERNAL SINGLE
                        STEP MAINTENANCE CLOCK
                        SET ''TSOM''
                        STEP CLOCK UNTIL ''TBMT'' SETS
                        LOAD ADDRESS WORD (TDB)
                        CLEAR ''TSOM''
                        STEP CLOCK AND TEST SYNC FLAG RECEIVED
```

```
                        SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 14
                        STEP CLOCK, OUTPUT DATA (ON "TBMT") AND TEST
                        "RDA" SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 34
                        TEST THAT DATA READ IS SAME AS SENT
                *       ERROR REPORT IS ERROR 23
                        TEST THAT "ERRCK" BIT IS NOT SET
                *       ERROR REPORT IS ERROR 25
                        SET "TEOM"
                        STEP CLOCK AND READ RECEIVED DATA WORDS
                        UNTIL SYNC FLAG RECEIVED SETS
                        TEST SYNC FLAG RECEIVED SETS WITHIN 400
                        (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 36
                        TEST ALL DATA RECEIVED WITHIN 400 (OCTAL)
                        CLOCK TICKS
                *       ERROR REPORT IS ERROR 34
                        TEST ALL DATA RECEIVED IS SAME AS SENT
                *       ERROR REPORT IS ERROR 23
                        TEST THAT "ERRCK" IS NOT SET
                *       ERROR REPORT IS ERROR 25
                        STEP CLOCK, READ DATA AND TEST SYNC FLAG
                        RECEIVED SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 22
                        STEP CLOCK 2 TICKS AND TEST "REOM" IS SET
                *       ERROR REPORT IS ERROR 21
                        TEST "ERRCK" IS NOT SET
                *       ERROR REPORT IS ERROR 25

TEST 64:        (ERROR CHECK TEST FOR CRC-16 (BOP MODE))
                THIS TEST WILL CHECK THAT THE CRC-16 POLYNOMIAL
                (X16 + X15 + X2 +1) WILL FUNCTION WITHOUT SETTING
                THE ERROR CHECK BIT.  THERE IS NO FORCED ERROR
                CONDITION IN THIS TEST.
                THIS TEST PERFORMS THE FOLLOWING SEQUENCE
                        ISSUE A MASTER CLEAR SEQUENCE
                        SET CHARACTER LENGTH TO 8 BITS/CHARACTER
                        SET PROTOCOL TO BOP AND ENABLE CCITT-CRC16
                        WITH A PRESET OF -1
                        SET TENA, RENA AND ENABLE INTERNAL SINGLE
                        STEP MAINTENANCE CLOCK
                        SET "TSOM"
                        STEP CLOCK UNTIL "TBMT" SETS
                        LOAD ADDRESS WORD (TDB)
                        CLEAR "TSOM"
                        STEP CLOCK AND TEST SYNC FLAG RECEIVED
                        SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 14
                        STEP CLOCK, OUTPUT DATA (ON "TBMT") AND TEST
                        "RDA" SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 34
                        TEST THAT DATA READ IS SAME AS SENT
                *       ERROR REPORT IS ERROR 23
                        TEST THAT "ERRCK" BIT IS NOT SET
                *       ERROR REPORT IS ERROR 25
```

```
                        SET "TEOM"
                        STEP CLOCK AND READ RECEIVED DATA WORDS
                        UNTIL SYNC FLAG RECEIVED SETS
                        TEST SYNC FLAG RECEIVED SETS WITHIN 400
                        (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 36
                        TEST ALL DATA RECEIVED WITHIN 400 (OCTAL)
                        CLOCK TICKS
                *       ERROR REPORT IS ERROR 34
                        TEST ALL DATA RECEIVED IS SAME AS SENT
                *       ERROR REPORT IS ERROR 23
                        TEST THAT "ERRCK" IS NOT SET
                *       ERROR REPORT IS ERROR 25
                        STEP CLOCK, READ DATA AND TEST SYNC FLAG
                        RECEIVED SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 22
                        STEP CLOCK 2 TICKS AND TEST "REOM" IS SET
                *       ERROR REPORT IS ERROR 21
                        TEST "ERRCK" IS NOT SET
                *       ERROR REPORT IS ERROR 25

TEST 65:        (ERROR TEST FOR CCITT-CRC WITH INITIALIZE SET TO -1 (CCP MODE))
                THIS TEST WILL CHECK THAT THE CCITT POLYNOMIAL
                (X16 + X12 + X15 + 1) WITH A PRESET OF -1
                THERE IS NOT FORCED ERROR CONDITION IN THIS TEST.
                THIS TEST PERFORMS THE FOLLOWING SEQUENCE
                        ISSUE A MASTER CLEAR SEQUENCE
                        SET CHARACTER LENGTH TO 8 BITS/CHARACTER
                        SET PROTOCOL TO BOP AND ENABLE CCITT-CRC16
                        WITH A PRESET OF -1
                        SET TENA, RENA AND ENABLE INTERNAL SINGLE
                        STEP MAINTENANCE CLOCK
                        SET "TSOM"
                        STEP CLOCK UNTIL "TBMT" SETS
                        LOAD ADDRESS WORD (TDB)
                        CLEAR "TSOM"
                        STEP CLOCK AND TEST SYNC FLAG RECEIVED
                        SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 14
                        STEP CLOCK, OUTPUT DATA (ON "TBMT") AND TEST
                        "RDA" SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORS IS ERROR 34
                        TEST THAT DATA READ IS SAME AS SENT
                *       ERROR REPORT IS ERROR 23
                        TEST THAT "ERRCK" BIT IS NOT SET
                *       ERROR REPORT IS ERROR 25
                        SET "TEOM"
                        STEP CLOCK AND READ RECEIVED DATA WORDS
                        UNTIL SYNC FLAG RECEIVED SETS
                        TEST SYNC FLAG RECEIVED SETS WITHIN 400
                        (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 36
                        TEST ALL DATA RECEIVED WITHIN 400 (OCTAL)
                        CLOCK TICKS
                *       ERROR REPORT IS ERROR 34
                        TEST ALL DATA RECEIVED IS SAME AS SENT
```

```
                        *       ERROR REPORT IS ERROR 23
                                TEST THAT "ERRCK" IS NOT SET
                        *       ERROR REPORT IS ERROR 25
                                STEP CLOCK, READ DATA AND TEST SYNC FLAG
                                RECEIVED SETS WITHIN 400 (OCTAL) CLOCK TICKS
                        *       ERROR REPORT IS ERROR 22
                                STEP CLOCK 2 TICKS AND TEST "REOM" IS SET
                        *       ERROR REPORT IS ERROR 21
                                TEST "ERRCK" IS NOT SET
                        *       ERROR REPORT IS ERROR 24
```

```
        TEST 66:        (ERROR CHECK TEST FOR CCITT-CRC WITH INITIALIZE SET TO 0 (CCP MODE))
                        THIS TEST WILL CHECK THE CCITT POLYNOMIAL
                        (X16 + X12 + X5 +1) WITH A PRESET OF 0.  THERE IS
                        NO FORCED ERROR CONDITION IN THIS TEST.
                        THIS TEST PERFORMS THE FOLLOWING SEQUENCE
                                ISSUE A MASTER CLEAR SEQUENCE
                                SET CHARACTER LENGTH TO 8 BITS/CHARACTER
                                SET PROTOCOL TO BOP AND ENABLE CCITT-CRC16
                                WITH A PRESET OF -1
                                SET TENA, RENA AND ENABLE INTERNAL SINGLE
                                STEP MAINTENANCE CLOCK
                                SET "TSOM"
                                STEP CLOCK UNTIL "TBMT" SETS
                                LOAD ADDRESS WORD (TDB)
                                CLEAR "TSOM"
                                STEP CLOCK AND TEST SYNC FLAG RECEIVED
                                SETS WITHIN 400 (OCTAL) CLOCK TICKS
                        *       ERROR REPORT IS ERROR 14
                                STEP CLOCK, OUTPUT DATA (ON "TBMT") AND TEST
                                "RDA" SETS WITHIN 400 (OCTAL) CLOCK TICKS
                        *       ERROR REPORS IS ERROR 34
                                TEST THAT DATA READ IS SAME AS SENT
                        *       ERROR REPORT IS ERROR 23
                                TEST THAT "ERRCK" BIT IS NOT SET
                        *       ERROR REPORT IS ERROR 25
                                SET "TEOM"
                                STEP CLOCK AND READ RECEIVED DATA WORDS
                                UNTIL SYNC FLAG RECEIVED SETS
                                TEST SYNC FLAG RECEIVED SETS WITHIN 400
                                (OCTAL) CLOCK TICKS
                        *       ERROR REPORT IS ERROR 36
                                TEST ALL DATA RECEIVED WITHIN 400 (OCTAL)
                                CLOCK TICKS
                        *       ERROR REPORT IS ERROR 34
                                TEST ALL DATA RECEIVED IS SAME AS SENT
                        *       ERROR REPORT IS ERROR 23
                                TEST THAT "ERRCK" IS NOT SET
                        *       ERROR REPORT IS ERROR 25
                                STEP CLOCK, READ DATA AND TEST SYNC FLAG
                                RECEIVED SETS WITHIN 400 (OCTAL) CLOCK TICKS
                        *       ERROR REPORT IS ERROR 22
                                STEP CLOCK 2 TICKS AND TEST "REOM" IS SET
                        *       ERROR REPORT IS ERROR 21
                                TEST "ERRCK" IS NOT SET
                        *       ERROR REPORT IS ERROR =24

        TEST 67:        (ERROR CHECK TEST FOR CRC-16 (CLP MODE).
                        THIS TEST WILL CHECK THAT THE CRC-16 POLYNOMIAL
                        (X16 + X15 + X2 +1) WILL FUNCTION WITHOUT SETTING
                        THE ERROR CHECK BIT.  THERE IS NO FORCED ERROR
                        CONDITION IN THIS TEST.
```

```
                    THIS TEST PERFORMS THE FOLLOWING SEQUENCE
                            ISSUE A MASTER CLEAR SEQUENCE
                            SET CHARACTER LENGTH TO 8 BITS/CHARACTER
                            SET PROTOCOL TO BOP AND ENABLE CCITT-CRC16
                            WITH A PRESET OF -1
                            SET TENA, RENA AND ENABLE INTERNAL SINGLE
                            STEP MAINTENANCE CLOCK
                            SET "TSOM"
                            STEP CLOCK UNTIL "TBMT" SETS
                            LOAD ADDRESS WORD (TDB)
                            CLEAR "TSOM"
                            STEP CLOCK AND TEST SYNC FLAG RECEIVED
                            SETS WITHIN 400 (OCTAL) CLOCK TICKS
                    *       ERROR REPORT IS ERROR 14
                            STEP CLOCK, OUTPUT DATA (ON "TBMT") AND TEST
                            "RDA" SETS WITHIN 400 (OCTAL) CLOCK TICKS
                    *       ERROR REPORS IS ERROR 34
                            TEST THAT DATA READ IS SAME AS SENT
                    *       ERROR REPORT IS ERROR 23
                            TEST THAT "ERRCK" BIT IS NOT SET
                    *       ERROR REPORT IS ERROR 25
                            SET "TEOM"
                            STEP CLOCK AND READ RECEIVED DATA WORDS
                            UNTIL SYNC FLAG RECEIVED SETS
                            TEST SYNC FLAG RECEIVED SETS WITHIN 400
                            (OCTAL) CLOCK TICKS
                    *       ERROR REPORT IS ERROR 36
                            TEST ALL DATA RECEIVED WITHIN 400 (OCTAL)
                            CLOCK TICKS
                    *       ERROR REPORT IS ERROR 34
                            TEST ALL DATA RECEIVED IS SAME AS SENT
                    *       ERROR REPORT IS ERROR 23
                            TEST THAT "ERRCK" IS NOT SET
                    *       ERROR REPORT IS ERROR 25
                            STEP CLOCK, READ DATA AND TEST SYNC FLAG
                            RECEIVED SETS WITHIN 400 (OCTAL) CLOCK TICKS
                    *       ERROR REPORT IS ERROR 22
                            STEP CLOCK 2 TICKS AND TEST "REOM" IS SET
                    *       ERROR REPORT IS ERROR 21
                            TEST "ERRCK" IS NOT SET
                    *       ERROR REPORT IS ERROR =24

TEST 70:            (CCP MODE ODD PARITY "ERRCHK" MODE)
                    THIS TEST WILL CHECK THAT THE ODD PARITY
                    GENERATION LOGIC (IN USYRT) WILL NOT RAISE
                    THE "ERRCHK" BIT WHEN THE RECEIVER RECEIVES
                    THE DATA CHARACTER.
                    THIS TEST EXECUTES IN THE FOLLOWING SEQUENCE.
                            ISSUE A MASTER CLEAR
                            SET TRANSMIT AND RECEIVE TO 7 BIT CHARACTER
                            LENGTH
                            SET PROTOCOL (CCP) STRIP SYNC AND ODD PARITY
                            GENERATION AND DETECTION
                            LOAD SYNC REGISTER (SEC 4) WITH SYNC CHARACTERS (26)
                            SET TENA, TENA AND INTERNAL SINGLE STEP LOOP
                            SET "TSOM" (TRANSMIT START OF MESSAGE)
```

```
                        STEP CLOCK UNTIL "TBMT" SETS
                        TEST THAT SYNC FLAG RECEIVED (S/F) SETS
                        WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 15
                        CLEAR "TSOM" BIT
                        STEP CLOCK AND OUTPUT DATA AND MONITOR THAT
                        "RDA" SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 34
                        READ DATA RECEIVED AND VERIFY DATA IS
                        SAME AS SENT
                *       ERROR REPORT IS ERROR 23
                        THE "ERRCHK" BIT IS READ AND TESTED FOR
                        NOT BEING SET
                *       ERROR REPORT IS ERROR 25

TEST 71:        (CCP MODE EVEN PARITY "ERRCHK" MODE)
                THIS TEST WILL CHECK THAT THE EVEN PARITY
                GENERATION LOGIC (IN USYRT) WILL NOT RAISE THE
                "ERRCHK" BIT WHEN THE RECEIVER RECEIVES THE DATA
                CHARACTER
                THIS TEST EXECUTES IN THE FOLLOWING SEQUENCE.
                        ISSUE A MASTER CLEAR
                        SET TRANSMIT AND RECEIVE TO 7 BIT CHARACTER
                        LENGTH
                        SET PROTOCOL (CCP) STRIP SYNC AND ODD PARITY
                        GENERATION AND DETECTION
                        LOAD SYNC REGISTER (SEC 4) WITH SYNC CHARACTERS (26)
                        SET TENA, TENA AND INTERAL SINGLE STEP LOOP
                        SET "TSOM" (TRANSMIT START OF MESSAGE)
                        STEP CLOCK UNTIL "TBMT" SETS
                        TEST THAT SYNC FLAG RECEIVED (S/F) SETS
                        WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 15
                        CLEAR "TSOM" BIT
                        STEP CLOCK AND OUTPUT DATA AND MONITOR THAT
                        "RDA" SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 34
                        READ DATA RECEIVED AND VERIFY DATA IS
                        SAME AS SENT
                *       ERROR REPORT IS ERROR 23
                        THE "ERRCHK" BIT IS READ AND TESTED FOR
                        NOT BEING SET
                *       ERROR REPORT IS ERROR 25

TEST 72:        (TRANSMITTER UNDERRUN TEST-BOP MODE)
                THIS TEST WILL CHECK THAT A FORCED UNDERRUN
                CONDITION WILL SET "TSA" (TRANSMITTER STATUS
                AVAILABLE) "TERR"" (TRANSMITTER ERROR) AND
                THAT THE RECEIVER WILL SET THE "RXAB"
                (RECEIVER ABORT)
                THIS TEST EXECUTES IN THE FOLLOWING SEQUENCE
                        ISSUE A MASTER CLEAR SEQUENCE
                        SET CHARACTER LENGTH TO 8 BITS
                        SET PROTOCOL TO BOP
                        SET TENA, RENA AND INTERNAL SINGLE STEP MAINT CLOCK
                        SET "TSOM" (TRANSMIT START OF MESSAGE)
```

```
                        STEP CLOCK UNTIL "TBMT" SETS
                        LOAD AN ADDRESS CHARACTER
                        CLEAR "TSOM" BIT
                        STEP CLOCK AND OUTPUT DATA (ON "TBMT" SET)
                        UNTIL SYNC FLAG RECEIVED (S/F) SETS
                        TEST SYNC FLAG RECEIVED SETS WITHIN 400
                        (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 14
                        STEP CLOCK AND OUTPUT DATA UNTIL "RDA"
                        (RECEIVE DATA AVAILABLE) SETS
                        TEST "RDA" SETS WITHIN 400 (OCTAL)
                        CLOCK TICKS
                *       ERROR REPORT IS ERROR 34
                        STEP CLOCK AND READ DATA BUFFER WHEN
                        "RDA" SETS UNTIL "TSA" (TRANSMIT STATUS AVAILABLE)
                        SETS
                        TEST "TSA" SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 36
                        TEST "TERR" (TRANSMIT ERROR (SEC 3-7) IS SET
                *       ERROR REPORT IS ERROR 26
                        STEP CLOCK AND READ RECEIVE DATA BUFFER
                        WHEN "RDA" SETS AND TEST "RXAB" (RECEIVER
                        ABORT) SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 27
                NOTE:   THE "RXAB" (177) IS GENERATED BY THE USYRT
                        TRANSMITTER LOGIC ON AN UNDERRUN CONDITION
                        IN CCP MODE

TEST 73:        (TRANSMITTER UNDERRUN TEST FOR CCP MODE)
                THIS TEST WILL CHECK THAT A FORCED UNDERRUN
                CONDITION WILL SET "TSA" (TRANSMITTER STATUS
                AVAILABLE) "TERR" (TRANSMITTER ERROR) THEN CONTINUE
                STEPPING THE CLOCK UNTIL A SYNC FLAG IS RECEIVED
                INDICATING THE TRANSMITTER SENT SYNC CHARACTERS
                (SECONDARY REG 4).  WHEN THE UNDERRUN CONDITION OCCURRED
                THIS TEST EXECUTES IN THE FOLLOWING SEQUENCE
                        ISSUE A MASTER CLEAR SEQUENCE
                        SET CHARACTER LENGTH TO 8 BITS
                        SET PROTOCOL TO CCP
                        LOAD SYNC REGISTER WITH SYNC CHARACTER (26)
                        SET TENA, RENA AND ENABLE INTERNAL SINGLE
                        STEP MAINTENANCE CLOCK
                        SET "TSOM"
                        STEP MAINT. CLOCK UNTIL "TBMT" SETS
                        LOAD TRANSMIT DATA BUFFER WITH AN STX (2)
                        STEP CLOCK AND OUTPUT DATA AND TEST SYNC
                        FLAG RECEIVED (S/F) SETS WITHIN 400 (OCTAL
                        CLOCK TICKS
                *       ERROR REPORT IS ERROR 15
                        CLEAR "TSOM"
                        STEP CLOCK AND OUTPUT DATA AND TEST
                        "RDA" SETS WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 34
                        STEP CLOCK, READ RECEIVE DATA BUFFER
                        WHEN "RDA" SETS UNTIL "TSA" SETS
                        TEST THAT "TSA" SETS WITHIN 400 (OCTAL)
```

```
                        CLOCK TICKS
                *       ERROR REPORT IS ERROR 36
                        TEST THAT "TERR" (TRANSMITTER ERROR) IS SET
                *       ERROR REPORT IS ERROR 26
                        STEP CLOCK AND TEST SYNC FLAG RECEIVED SETS
                        WITHIN 400 (OCTAL) CLOCK TICKS
                *       ERROR REPORT IS ERROR 28

TEST 74:        (RECEIVER OVERRUN TEST FOR CCP MODE)
                THIS TEST WILL FORCE A RECEIVER OVERRUN CONDITION
                BY NOT READING THE SECOND DATA CHARACTER RECEIVED.
                THE "ROR" BIT IS READ (SECONDARY REG. 1-BIT 3)
                AND CHECKED FOR BEING CLEAR (RESULT OF
                ORIGINAL READING).
                THIS TEST EXECUTES THE FOLLOWING SEQUENCE
                        ISSUE A MASTER CLEAR SEQUENCE
                        SET CHARACTER LENGTH TO 8 BITS
                        SET PROTOCOL TO CCP
                        LOAD SYNC ADDRESS REGISTER WITH SYNC (26)
                        SET TENA, RENA AND ENABLE INTERAL SINGLE
                        STEP MAINTENANCE CLOCK
                        SET "TSOM"
                        STEP CLOCK 9 TICKS
                        CLEAR "TSOM"
                        LOAD DATA REGISTER WITH CHARACTER (SOH=1)
                        STEP CLOCK 16 TICKS
                        LOAD FIRST DATA CHARACTER
                        STEP CLOCK 8 TICKS
                        LOAD 2ND DATA CHARACTER
                        STEP CLOCK 8 TICKS
                        LOAD 3RD DATA CHARACTER
                        STEP CLOCK 8 TICKS
                        READ RECEIVE DATA BUFFER
                        LOAD 4TH DATA CHARACTER
                        STEP CLOCK 16 TICKS (FORCE RECEIVER OVERRUN)
                        TEST THAT "ROR" IS NOW SET
                *       ERROR REPORT IS ERROR 29
                        READ "ROR" 2ND TIME AND TEST BIT
                        IS CLEAR
                *       ERROR REPORT IS ERROR 30

TEST 75:        (RECEIVER OVERRUN TEST FOR BOP MODE)
                THIS TEST WILL FORCE A RECEIVER OVERRUN CONDITION BY
                NOT READING THE SECOND DATA CHARACTER RECEIVED.  THE
                "ROR" BIT IS READ (SECONDARY REG. 1-BIT 3)
                AND CHECKED BOR BEING SET.  THE BIT IS THEN REREAD
                AND CHECKED FOR BEING CLEAR (RESULT OF ORIGINAL
                READING).
                THIS TEST EXECUTES THE FOLLOWING SEQUENCE
                        ISSUE A MASTER CLEAR SEQUENCE
                        SET CHARACTER LENGTH TO 8 BITS
                        SET PROTOCOL TO BOP AND ENABLE SECONDARY ADDRESS
                        MODE.
                        LOAD SYNC REGISTER (SECONDARY 4) WITH A 376.
                        SET TENA, RENA AND ENABLE INTERNAL SINGLE
                        STEP MAINTENANCE CLOCK.
```

```
                        SET "TSOM"
                        STEP CLOCK 2 TICKS
                        CLEAR "TSOM"
                        LOAD TRANSMIT DATA BUFFER WITH SECONDARY
                        ADDRESS (376)
                        STEP CLOCK 14 TICKS
                        LOAD DATA BUFFER WITH CONTROL CHARACTER (0)
                        STEP CLOCK 8 TICKS
                        LOAD 1ST DATA CHARACTER
                        STEP CLOCK 8 TICKS
                        LOAD 2ND DATA CHARACTER
                        STEP CLOCK 8 TICKS
                        LOAD 3RD DATA CHARACTER
                        STEP CLOCK 8 TICKS
                        READ RECEIVE DATA BUFFER
                        LOAD 4TH DATA CHARACTER
                        STEP CLOCK 16 TICKS (FORCE RECEIVER OVERRUN)
                        TEST "ROR" BIT IS SET.
                *       ERROR REPORT IS ERROR 29
                        REREAD THE "ROR" BIT AND TEST THAT BIT
                        WAS CLEARED AND INITIAL READ.
                *       ERROR REPORT IS ERROR 30

TEST 76:        (TRANSMIT ABORT AND RECEIVE ABORT TEST)
                THIS TEST WILL CHECK THAT A "TXAB" (TRANSMITTER
                ABORT) WILL TRANSMIT AN ABORT CHARACTER
                AND THE USYRT RECEIVE LOGIC WILL SET THE "RXAB"
                (RECEIVE ABORT) BIT WHEN THE CHARACTER IS
                CLOCKED INTO THE RECEIVER.
                THIS TEST EXECUTES THE FOLLOWING SEQUENCE
                        ISSUE A MASTER CLEAR SEQUENCE
                        SET PROTOCOL TO BOP AND ENABLE SECONDARY
                        ADDRESS MODE
                        LOAD SECONDARY ADDRESS REGISTER WITH A 376
                        SET TENA, TENA AND ENABLE INTERNAL SINGLE
                        STEP CLOCK
                        SET "TSOM"
                        STEP CLOCK 2 TICKS
                        CLEAR "TSOM"
                        WRITE TRANSMIT DATA BUFFER WITH SECONDARY
                        ADDRESS CHARACTER (376)
                        STEP CLOCK 14 TICKS TO SYNC THE RECEIVE LOGIC
                        WRITE TRANSMIT DATA BUFFER WITH CONTROL
                        CHARACTER (0)
                        STEP CLOCK 8 TICKS TO TRANSMIT OUT THE
                        CONTROL CHARACTER
                        TRANSMIT OUT 3 DATA CHARACTERS (10)
                        SET "TEOM" AND "TXAB"
                        STEP CLOCK 8 TICKS
                        READ RECEIVE DATA BUFFER (SECONDARY ADDRESS)
                        STEP CLOCK 8 TICKS
                        READ RECEIVE DATA BUFFER (1ST DATA CHARACTER)
                        STEP CLOCK 9 TICKS (TRANSMIT IN ABORT CHARACTER)
                        READ "RXAB" AND TEST FOR BEING SET
                *       ERROR REPORT IS ERROR 31
```

```
TEST 77:        (TRANSMIT GO AHEAD AND RECEIVE GO AHEAD TEST (BOP MODE))
                THIS TEST WILL FORCE A GO AHEAD CONDITION BY SETTING
                THE "TXGA" (TRANSMIT GO AHEAD) BIT AFTER THE LOGIC
                IS IN SYNC AND TESTING THAT THE RECEIVE LOGIC
                DETECTS THE GO AHEAD CONDITION BY SETTING "RXGA"
                (RECEIVE GO AHEAD).
                THIS TEST WILL EXECUTE THE FOLLOWING SEQUENCE
                        ISSUE A MASTER CLEAR SEQUENCE
                        SET PROTOCOL TO BOP AND ENABLE LOOP MODE
                        SET TENA, RENA AND ENABLE INTERNAL SINGLE
                        STEP MAINTENANCE CLOCK
                        SET "TSOM"
                        STEP CLOCK 2 TICKS
                        CLEAR "TSOM"
                        WRITE TRANSMIT DATA BUFFER WITH FIRST
                        DATA CHARACTER (101)
                        STEP CLOCK 14 TICKS (SYNC LOGIC AND XMIT
                        FIRST CHARACTER)
                        LOAD 2ND DATA CHARACTER (101)
                        STEP CLOCK 8 TICKS
                        LOAD 3RD DATA CHARACTER (101)
                        STEP CLOCK 8 TICKS
                        SET "TEOM" AND "TXGA"
                        STEP  CLOCK 8 TICKS (SEND CRC #1)
                        READ DATA REGISTER (ADDRESS:LM=1(BOP))
                        STEP CLOCK 8 TICKS (SEND CRC #2)
                        READ DATA REGISTER (1ST DATA CHARACTER)
                        STEP CLOCK 8 TICKS
                        READ DATA REGISTER (2ND DATA CHARACTER)
                        STEP CLOCK 9 TICKS
                        TEST THAT "RXGA" IS NOW SET
                *       ERROR REPORT IS ERROR 32

TEST 100:       (EXTERNAL DATA LOOP TEST)
                THIS TEST CHECKS THAT THE EXTERNAL DATA LOOP
                AND CLOCK PATH IS FUNCTIONAL (EXTERNAL DRIVERS
                AND RECEIVERS)  THE TEST WILL TRANSMIT AND
                RECEIVE A WALKING ONES DATA PATTERN (1 TO 200)
                THIS TEST WILL EXECUTE ONLY IF THE OPERATOR
                HAS ENABLED THE EXTERNAL TEST (CONFIGURATION)
                THIS TEST WILL EXECUTE THE FOLLOWING SEQUENCE
                        ISSUE MASTER CLEAR SEQUENCE
                        SET PROTOCOL TO BOP
                        SET TENA, RENA AND ENABLE EXTERNAL SINGLE
                        STEP MAINTENANCE CLOCK
                        SET "TSOM"
                        STEP CLOCK 2 TICKS
                        CLEAR "TSOM"
                        LOAD TRANSMIT DATA BUFFER (CURRENT DATA PATTERN)
                        STEP CLOCK AND TEST "TBMT" SETS WITHIN
                        9 CLOCK TICKS
                *       ERROR REPORT IS ERROR 13
                        STEP CLOCK AND TEST "RDA" SETS WITHIN
                        9 CLOCK TICKS
                *       ERROR REPORT IS ERROR 42
                        READ RECEIVE DATA BUFFER AND TEST
```

```
                        DATA CHARACTER IS SAME AS SENT
                    *   ERROR REPORT IS ERROR 23

TEST 101:           (CRC-32 TRANSMITTER ERROR CHECK TEST)
                    THIS TEST WILL CHECK THAT THE TRANSMITTER
                    CHECK ERROR FLAG (PRI14-BIT 2) WILL SET BY
                    TRANSMITTING A DATA MESSAGE WITH TRANSMIT CHECK MODE
                    ENABLED AND AN ILLEGAL CRC-32 CODE APPENDED TO THE
                    DATA MESSAGE.
                    THE DATA WORD TRANSMITTED IS A BYTE OF ALL ZEROS
                    AND THE APPENDED 32 BIT CRC IS A 1
                    THIS TEST EXECUTES ONLY IF EXT. TESTS ARE CONFIGURED
                    THIS TEST WILL EXECUTE THE FOLLOWING SEQUENCE
                            ISSUE A MASTER CLEAR SEQUENCE
                            SET PROTOCOL TO BOP AND DISABLE USYRT ERROR
                            CHECK FUNCTIONS
                            ENABLE CRC-32 TRANSMIT CHECK MODE (PRI13-BIT 5)
                            TEST THAT BITS 1 & 2 OF PRIMARY REGISTER 14
                            ARE NOT SET.
                    *       ERROR REPORT IS ERROR 38
                            SET TENA, RENA, ENABLE CRC-32 (CHECK MODE)
                            AND ENABLE EXTERNAL SINGLE STEP MAINT. CLOCK
                            SET "TSOM"
                            STEP CLOCK 2 TICKS
                            CLEAR "TSOM"
                            SEQUENCE TO TRANSMIT THE DATA BYTE
                            AND 2 CRC BYTES, STEPPING CLOCK 8 TICKS
                            FOR EACH AND TESTING "TBMT" SETS
                    *       ERROR REPORT IS ERROR 13
                            SET "TEOM"
                            STEP CLOCK 17 17 TICKS
                            TEST "TCRCE" IS SET AND "RCRCE" IS CLEAR
                    *       ERROR REPORT IS ERROR 38
                            FIRST TIME THROUGH TEST CHECK "TCRCE"
                            CLEARS BY WRITING THE BIT WITH A ONE.
                    *       ERROR REPORT IS ERROR 38
                            SECOND TIME THROUGH TEST CHECK "TCRCE"
                            CLEARS BY ISSUING A LINE RESET.
                    *       ERROR REPORT IS ERROR 38

TEST 102:           (CRC-32 RECEIVER ERROR CHECK TEST)
                    THIS TEST WILL CHECK THAT RECEIVE CHECK
                    ERROR FLAG (PRI14-BIT 1) WILL SET BY TRANSMITTING
                    A MESSAGE (1 BYTE=1) WITH AN APPENDED CRC-32
                    CODE (32 BITS=0) AND THE CRC LOGIC PLACED IN
                    CHECK MODE (NON-GENERATE).
                    THIS TEST WILL EXECUTE ONLY IF EXTERNAL TEST ARE
                    ENABLED (CONFIGURATION)
                    THIS TEST WILL EXECUTE THE FOLLOWING SEQUENCE
                            ISSUE A MASTER CLEAR SEQUENCE
                            SET PROTOCOL TO BOP AND DISABLE USYRT
                            ERROR CHECK FUNCTIONS
                            TEST THAT "TCRCE" AND "RCRCE" ARE NOT SET
                    *       ERROR REPORT IS ERROR 38
                            SET TENA, RENA AND ENABLE CRC-32
                            AND EXTERNAL SINGLE STEP MAINTENANCE CLOCK.
```

```
                        SET ''TSOM''
                        STEP CLOCK 2 TICKS
                        CLEAR ''TSOM''
                        TRANSMIT THE DATA CHARACTER AND 4 CRC-32
                        BYTES, TESTING AFTER EACH CHARACTER CLOCKED
                        OUT THAT ''TBMT'' IS SET
                *       ERROR REPORT IS ERROR 13
                        SET ''TEOM''
                        CLOCK IN 4 CHARACTERS (READ DATA REG. AFTER
                        EACH CHARACTER RECEIVED)
                        TEST ''RCRCE'' (PRI14-BIT 1) IS SET
                *       ERROR REPORT IS ERROR 38
                        FIRST TIME THROUGH TEST CHECK ''RCRCE''
                        IS CLEANED BY WRITING THE BIT WITH A ONE
                *       ERROR REPORT IS ERROR 38
                        SECOND TIME THROUGH TEST CHECK ''TCRCE''
                        IS CLEANED BY ISSUING A LINE RESET.
                *       ERROR REPORT IS ERROR 38


TEST 103:       (CRC-32 NON-ERROR TRANSMIT & RECEIVE TEST)
                THIS TEST WILL CHECK A NON-ERROR CRC RECEIVE
                CHECK AND NON-ERROR TRANSMIT CHECK BY TRANSMITTING
                A DATA STREAM (NON-GENERATE) WITH A PRECALCULATED
                CRC-32 APPENDED.  THE DATA STREAM CONSISTS OF
                WALKING A ONE ACCROSS 16 BITS (1 TO 100000).
                THIS TEST EXECUTES ONLY OF THE EXTERNAL DATA TESTS
                ARE ENABLED (CONFIGURATION)
                THIS TEST EXECUTES THE FOLLOWING SEQUENCE
                        ISSUE A MASTER CLEAR SEQUENCE
                        SET PROTOCOL TO BOP AND DISABLE USYRT
                        ERROR CHECKING
                        SET CRC-32 TO TRANSMIT CHECK MODE
                        SET TENA, RENA AND ENABLE CRC-32 AND
                        EXTERNAL SINGLE STEP MAINTENANCE CLOCK
                        TEST THAT ''TCRCE'' AND ''RCRCE'' ARE NOT SET
                *       ERROR REPORT IS ERROR 38
                        SET ''TSOM''
                        STEP CLOCK 2 TICKS
                        CLEAR ''TSOM''
                        SEQUENCE TO OUTPUT AND RECEIVE THE DATA AND
                        CRC-32 CHARACTERS MONITORING THAT THE CHARACTERS
                        RECEIVE WITHIN 11 CLOCK TICKS
                *       ERROR REPORT IS ERROR 41
                        TEST THAT ''TCRCE'' AND ''RCRCE'' ARE NOT SET
                *       ERROR REPORT IS ERROR 39

                        REPEAT TEST FOR EACH DATA VALUE AND CALCULATED
                        CRC-32

TEST 104:       (CRC-32 TRANSMIT GENERATE TEST)
                THIS TEST WILL CHECK CRC-32 TRANSMIT GENERATE BY
                TRANSMITTING A ONE WORD BUFFER OF A WALKING ONE
                (1 TO 100000) DATA PATTERN AND CHECKING THE
                RECEIVED CRC32 DATA.
                THIS TEST WILL EXECUTE ONLY IF EXTERNAL TESTS
```

```
                    ARE ENABLED (CONFIGURATION)
                    THIS TEST WILL EXECUTE THE FOLLOWING SEQUENCE
                            ISSUE A MASTER CLEAR SEQUENCE
                            SET PROTOCOL TO BOP AND DISABLE USYRT ERROR
                            TESTING
                            SET TENA, RENA, ENABLE CRC-32 AND EXTERNAL
                            SINGLE STEP MAINTENANCE CLOCK
                            SET TRANSMIT GENERATE MODE
                            TEST "TRCRE" AND "RCRCE" ARE NOT SET
                    *       ERROR REPORT IS ERROR 39
                            SET "TSOM"
                            STEP CLOCK 2 TICKS
                            CLEAR "TSOM"
                            SEQUENCE TO OUTPUT 2 DATA CHARACTERS
                            AND RECEIVE 2 DATA CHARACTERS AND
                            4 CRC BYTES (SETTING AND CLEARING
                            "TEOM" AT APPROPRIATE TIMES) AND MONITORING
                            NOT MORE THAN 11 CLOCKS TICKS ARE REQUIRED
                            FOR EACH CHARACTER
                    *       ERROR REPORT IS ERROR 41
                            TEST THAT GENERATED CRC-32 IS SAME AS CALCULATED
                    *       ERROR REPORT IS ERROR 40

                            REPEAT TEST FOR EACH DATA VALUE.
```

```
;*******************************************************************************

        THE FOLLOWING ARE THE ASSOCIATED ERRORS (NUMBERED)
        AND THEIR DATA HEADER STATEMENTS.

ERROR1:PRIMARY REGISTER ERROR
ERROR2:FLOATING ONE'S ERROR
ERROR3:SECONDARY REGISTER READ ERROR
ERROR4:THE FOLLOWING READ ONLY BIT CAN BE WRITTEN
ERROR5:TRANSMIT STATUS REGISTER BIT FAILURE
ERROR6:MODE CONTROL REGISTER BIT FAILURE
ERROR7:CHARACTER LENGTH REGISTER BIT FAILURE
ERROR8:TRANSMIT STATUS REGISTER BIT FAILED TO CLEAR AFTER LINE RESET
ERROR9:MODE CONTROL REGISTER BIT FAILED TO CLEAR AFTER LINE RESET
ERROR10:SECONDARY INCREMENT ONE'S TEST FAILURE
ERROR11:SECONDARY REGISTER DID NOT CLEAR AFTER LINE RESET
ERROR12:TBMT NOT RESET AFTER TSOM SET WITH TENA SET
ERROR13:TBMT FAILED TO SET
ERROR14:S/F DID NOT SET AFTER FLAG SENT
ERROR15:S/F DID NOT SET AFTER SYNC SENT
ERROR16:S/F DID NOT RESET
ERROR17:RAC FAILED TO SET AFTER 3 DATA BYTES TRANSMITTED
ERROR18:RDA FAILED TO SET AFTER 3 DATA BYTES TRANSFERED
ERROR19:RSOM FAILED TO SET
ERROR20:FIRST DATA BYTE TRANSMITTED IN ERROR
ERROR21:REOM FAILED TO SET AFTER TEOM SET
ERROR22:LINE FAILED TO RESYNC AFTER TEOM SET WITH IDLE RESET
ERROR23:DATA TRANSMITTED INCORRECTLY
ERROR24:ERROR CHECK BIT FAILED TO SET
ERROR25:ERROR CHECK BIT SET AND SHOULD NOT HAVE
ERROR26:NO TRANSMIT ERROR ON A FORCE UNDERRUN
ERROR27:NO ABORT INDICATIONS RECEIVED ON A FORCE UNDERRUN
ERROR28:LINE DID NOT RESYNCE ON A FORCE UNDERRUN
ERROR29:RECEIVER OVERRUN DID NOT SET ON A FORCE OVERRUN
ERROR30:RECEIVER OVERRUN DID NOT RESET AFTER STAUS READ
ERROR31:NO RECEIVE ABORT ON A TRANSMIT OF A ABORT
ERROR32:NO RECEIVE GO AHEAD ON A TRANSMIT OF GO AHEAD
ERROR33:DEVICE IS STILL TRYING TO SYNC
ERROR34:THE KMC11 IS HUNG
ERROR35:NO S/F AFTER TEOM WAS SET
ERROR36:S/F SET BEFORE LAST DATA BYTE RECEIVED
ERROR37:RSA DID NOT SET ON A TRANSMIT UNDERRUN
ERROR38:CRC-32 ERROR STATUS BITS (PR14-1&2)
ERROR39:CRC-32 TRANSFER CAUSED STATUS ERROR
ERROR40:CRC-32 GENERATED IS NOT CORRECT
ERROR41:TRANSFER INCOMPLETE
ERROR42:'RDA" FAILED TO SET ON DATA TRANSFER

THE FOLLOWING FOUR FIELDS ARE PESENT IN EVERY ERROR MESSAGE.


        TEST                            PC                                  KMC11                                   LINE
(OCTAL # OF FAILING TEST) (PROGRAM ADDR. OF ERROR) (1-16) # OF CONFIGURED UNIT) (# OF FAILING LINE (0-7)


THE FOLLOWING DESCRIBES THE OTHER POTENTIAL DATA AND STATUS FIELDS OF THE ERROR REPORTS.
```

REG = NUMBER OF FAILING REGISTER AS ADDRESSED BY KMC11.

EXP = VAUE OF EXPECTED DATA OR STATUS

REC = VALUE OF RECEIVED OR READ DATA OR STATUS

PRI12 = CONTENTS OF PRIMARY REGISTER 12

CRC-HI = HI ORDER 16 BITS OF CALCULATED CRC-32

CRC-LO = LOW ORDER 16 BITS OF CALCULATED CRC-32

PRI14 = VALUE OF PRI14 (LINE UNIT STATUS)

PRI14(SHBE) = STATUS VALUE THAT PRIMARY REG 14 SHOULD BE

SHBE-HI = CALCULATED HIGH ORDER 16 BITS OF CRC-32

SHBE-LO = CALCULATED LOW ORDER 16 BITS OF CRC-32

WAS-HI = RECEIVER HIGH ORDER 16 BITS OF CRC-32

WAS-LO = RECEIVED LOW ORDER 16 BITS OF CRC-32

DATA = DATA WORD TRANSMITTED

8.0 DIAGNOSTIC LISTING

%

```
        .ENABL  ABS,AMA
        .NLIST  MD,MC,CND
1623                                    .LIST   SEQ,LOC,BIN
1624                                    .LIST   ME
1625
```

```
1626
1627
1628
1629                            ;*****************************************************************
1630                            ;                START OF STATIC TEST LISTING
1631                            ;*****************************************************************
```

```
1632
```

```
1633
1634
1635
1636
```

```
1637
1638
1639
```

```
                                .LIST   SEQ,LOC,BIN
                                .ENABL  ABS,AMA
                                .NLIST  MC,MD,CND
                                .LIST   ME
                                .TITLE  CZKMDA0 DMS11-DA STATIC
                                ;*COPYRIGHT (C) 1981
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY RAY BALDWIN/R. J. COLLINS
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
                                ;*
000001                          $TN=1
160000                          $SWR=160000     ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPOUT
                                .SBTTL  BASIC DEFINITIONS

                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                          STACK=  1100
                                .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                                .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

                                ;*MISCELLANEOUS DEFINITIONS
000011                          HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                          LF=     12              ;;CODE FOR LINE FEED
000015                          CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                          PS=     177776          ;;PROCESSOR STATUS WORD
                                .EQUIV  PS,PSW
177774                          STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER

                                ;*GENERAL PURPOSE 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                          R6=     %6              ;;GENERAL REGISTER
000007                          R7=     %7              ;;GENERAL REGISTER
000006                          SP=     %6              ;;STACK POINTER
000007                          PC=     %7              ;;PROGRAM COUNTER

                                ;*PRIORITY LEVEL DEFINITIONS
000000                          PR0=    0               ;;PRIORITY LEVEL 0
000040                          PR1=    40              ;;PRIORITY LEVEL 1
000100                          PR2=    100             ;;PRIORITY LEVEL 2
000140                          PR3=    140             ;;PRIORITY LEVEL 3
000200                          PR4=    200             ;;PRIORITY LEVEL 4
000240                          PR5=    240             ;;PRIORITY LEVEL 5
000300                          PR6=    300             ;;PRIORITY LEVEL 6
000340                          PR7=    340             ;;PRIORITY LEVEL 7
```

```
1696
1697                                    ;*"SWITCH REGISTER" SWITCH DEFINITIONS
1698            100000                  SW15=   100000
1699            040000                  SW14=   40000
1700            020000                  SW13=   20000
1701            010000                  SW12=   10000
1702            004000                  SW11=   4000
1703            002000                  SW10=   2000
1704            001000                  SW09=   1000
1705            000400                  SW08=   400
1706            000200                  SW07=   200
1707            000100                  SW06=   100
1708            000040                  SW05=   40
1709            000020                  SW04=   20
1710            000010                  SW03=   10
1711            000004                  SW02=   4
1712            000002                  SW01=   2
1713            000001                  SW00=   1
1714                                    .EQUIV  SW09,SW9
1715                                    .EQUIV  SW08,SW8
1716                                    .EQUIV  SW07,SW7
1717                                    .EQUIV  SW06,SW6
1718                                    .EQUIV  SW05,SW5
1719                                    .EQUIV  SW04,SW4
1720                                    .EQUIV  SW03,SW3
1721                                    .EQUIV  SW02,SW2
1722                                    .EQUIV  SW01,SW1
1723                                    .EQUIV  SW00,SW0
1724
1725                                    ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
1726            100000                  BIT15=  100000
1727            040000                  BIT14=  40000
1728            020000                  BIT13=  20000
1729            010000                  BIT12=  10000
1730            004000                  BIT11=  4000
1731            002000                  BIT10=  2000
1732            001000                  BIT09=  1000
1733            000400                  BIT08=  400
1734            000200                  BIT07=  200
1735            000100                  BIT06=  100
1736            000040                  BIT05=  40
1737            000020                  BIT04=  20
1738            000010                  BIT03=  10
1739            000004                  BIT02=  4
1740            000002                  BIT01=  2
1741            000001                  BIT00=  1
1742                                    .EQUIV  BIT09,BIT9
1743                                    .EQUIV  BIT08,BIT8
1744                                    .EQUIV  BIT07,BIT7
1745                                    .EQUIV  BIT06,BIT6
1746                                    .EQUIV  BIT05,BIT5
1747                                    .EQUIV  BIT04,BIT4
1748                                    .EQUIV  BIT03,BIT3
1749                                    .EQUIV  BIT02,BIT2
1750                                    .EQUIV  BIT01,BIT1
1751                                    .EQUIV  BIT00,BIT0
```

```
                                    N 3
1752
1753                            ;*BASIC "CPU" TRAP VECTOR ADDRESSES
1754            000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
1755            000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
1756            000014          TBITVEC=14              ;;"T" BIT
1757            000014          TRTVEC= 14              ;;TRACE TRAP
1758            000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
1759            000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
1760            000024          PWRVEC= 24              ;;POWER FAIL
1761            000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
1762            000034          TRAPVEC=34              ;;"TRAP" TRAP
1763            000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
1764            000064          TPVEC=  64              ;;TTY PRINTER VECTOR
1765            000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
1766            163400          $SWR=163400
1767            000001          $TN=1
1768            000001          $N=1
```

```
                                .SBTTL  TRAP CATCHER

                        000000          .=0
                                ;*ALL UNUSED LOCATIONS OF THE VECTOR AREA CONTAIN
                                ;*A ".+2, IOT" SEQUENCE TO CATCH AND PROCESS ILLEGAL
                                ;*TRAPS AND INTERRUPTS THAT MIGHT OCCUR.
                                ;*THE IOT TRAP WHICH IS TAKEN ON THE ILLEGAL TRAP/INT
                                ;*TRAPS TO THE $SCOPE ROUTINE WHICH (IF THE RETURN PC IS
                                ;*LESS THAN 1002) JUMPS TO THE $ERROR ROUTINE.
                                ;*THE $ERROR ROUTINE WILL REPORT THE ERROR AS FOLLOWS:
                                ;*      PC=YYYYYY UNEXPECTED TRAP TO XXX
                                ;*AND RETURN TO THE PROGRAM AT PC=YYYYYY+2
                                ;*WHERE XXX=LOCATION OF ILLEGAL TRAP
                                ;*      YYYYYY=PC AT TIME OF TRAP
                                ;*NOTE: IF THE PROCESSOR IS NOT AN 11/05 THE PROGRAM
                                ;*      CAN BE STARTED AT ADDRESS 0 AS WELL AS ADDRESS 200.

000000  000000                  $40CAT: HALT                    ;;HALT
000002  000737                          BR      .-100           ;;BRANCH TO 177700 & TIME OUT (NOT ON
                                                                ;;11/05)
000004  002004                          .WORD   .START          ;;VECTOR TO STARTING ADDRESS
000006  000340                          .WORD   340             ;;WITH PRIORITY LEVEL 7
                        000174          .=174
000174  000000                  DISPREG:.WORD   0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:  .WORD   0               ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRES(ES)
000200  000137  002004                  JMP     @#.START        ;;GO TO START OF PROGRAM
```

```
                                .SBTTL  COMMON TAGS

                                ;;**********************************************************************
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102     000          $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103     000          $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114     000          $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115     001          $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134     000          $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135     000          $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154     000          $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155     002          $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156     012          $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157     000          $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000          $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001162  000000          $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164  000000          $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001166  000000          $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001170  000000          $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001172  000000          $REG4:  .WORD   0               ;;CONTAINS (($REGAD)+10)
001174  000000          $REG5:  .WORD   0               ;;CONTAINS (($REGAD)+12)
001176  000000          $REG6:  .WORD   0               ;;CONTAINS (($REGAD)+14)
001200  000000          $REG7:  .WORD   0               ;;CONTAINS (($REGAD)+16)
001202  000000          $TMP0:  .WORD   0               ;;USER DEFINED
001204  000000          $TMP1:  .WORD   0               ;;USER DEFINED
001206  000000          $TMP2:  .WORD   0               ;;USER DEFINED
001210  000000          $TMP3:  .WORD   0               ;;USER DEFINED
001212  000000          $TMP4:  .WORD   0               ;;USER DEFINED
001214  000000          $TMP5:  .WORD   0               ;;USER DEFINED
001216  000000          $TMP6:  .WORD   0               ;;USER DEFINED
001220  000000          $TMP7:  .WORD   0               ;;USER DEFINED
001222  000000          $TMP10: .WORD   0               ;;USER DEFINED
```

```
 1852  001224  000000                  $TMP11: .WORD   0               ;;USER DEFINED
 1853  001226  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
 1854  001230  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
 1855  001234     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
 1856  001235     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
 1857  001236  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
 1858                                  ;;*****************************************************************
 1859                                  ;
 1860                                  ;  PROGRAM CONTROL PARAMETERS
 1861                                  ;--------------------------------
 1862  001240                          .ROVAR:
 1863          000207                          EXIT=207
 1864          000002                          EXITT=RTI
 1865          000002                          EXITI=RTI
 1866          000207                          EXITS=207
 1867          000020                          TENA=BIT4
 1868          000001                          TSOM=BIT0
 1869          000100                          TAC=BIT6
 1870          000200                          TSA=BIT7
 1871          000020                          TBMT=BIT4
 1872          000100                          CCP=BIT6
 1873          000000                          BOP=0
 1874          000001                          RENA=BIT0
 1875          000001                          RDA=BIT0
 1876          000001                          X=1
 1877          000002                          Y=2
 1878          000004                          Z=4
 1879          000001                          CCITT=X
 1880          000003                          CRC16=X!Y
 1881          000004                          ODDP=Z
 1882          000005                          EVENP=X!Z
 1883          000010                          CRCEN=BIT3
 1884          000007                          CRC32=X!Y!Z
 1885          000200                          ERRCK=BIT7
 1886          000001                          SOH=001
 1887          000010                          SF=BIT3
 1888          000100                          MB=BIT6
 1889          000040                          MA=BIT5
 1890          000200                          MC=BIT7
 1891          000040                          LM=BIT5
 1892          000004                          RAC=BIT2
 1893          000002                          RSA=BIT1
 1894          000001                          RSOM=BIT0
 1895          000002                          REOM=BIT1
 1896          000004                          RXGA=BIT2
 1897          000004                          RXAB=BIT2
 1898          000010                          ROR=BIT3
 1899          000020                          ABCA=BIT4
 1900          000040                          ABCB=BIT5
 1901          000100                          ABCC=BIT6
 1902          000200                          ERRCK=BIT7
 1903          000002                          TEOM=BIT1
 1904          000004                          TXAB=BIT2
 1905          000010                          TXGA=BIT3
 1906          000200                          TERR=BIT7
 1907          000001                          PROTEX=BIT0
```

```
1908    000002          PROTEY=BIT1
1909    000004          PROTEZ=BIT2
1910    000010          NIDLE=BIT3
1911    000020          SAM=BIT4
1912    000040          SSL=BIT5
1913    000100          PROTO=BIT6
1914    000200          APA=BIT7
1915    000001          RDATA1=BIT0
1916    000002          RDATA2=BIT1
1917    000004          RDATA3=BIT2
1918    000010          EXCON=BIT3
1919    000020          EXADD=BIT4
1920    000040          TDATA1=BIT5
1921    000100          TDATA2=BIT6
1922    000200          TDATA3=BIT7
1923    000001          LNRST=BIT0
1924    005227          INCL=05227
1925    000007          DISCRC=7
1926    000026          SYN=26
1927    000002          STX=2
1928    000001          SOH=1
1929    000004          EOT=4
1930    000003          ETX=3
1931    000001          INIT=BIT0
1932    000002          LOCK=BIT1
1933    000004          LNLOCK=BIT2
1934    000010          TLOCK=BIT3
1935    000010          PRI10=10
1936    000011          PRI11=11
1937    000012          PRI12=12
1938    000013          PRI13=13
1939    000014          PRI14=14
1940    000015          PRI15=15
1941    000016          PRI16=16
1942    000017          PRI17=17
1943    000000          SEC0=0
1944    000001          SEC1=1
1945    000002          SEC2=2
1946    000003          SEC3=3
1947    000004          SEC4=4
1948    000005          SEC5=5
1949    000006          SEC6=6
1950    000007          SEC7=7
1951    000000          LN0=0
1952    000040          LN1=40
1953    000100          LN2=100
1954    000140          LN3=140
1955    000200          LN4=200
1956    000240          LN5=240
1957    000300          LN6=300
1958    000340          LN7=340
1959    000000          SEL0=0
1960    000002          SEL2=2
1961    000004          SEL4=4
1962    000006          SEL6=6
1963    000000          BSEL0=0
```

```
 1964            000001                          BSEL1=1
 1965            000002                          BSEL2=2
 1966            000003                          BSEL3=3
 1967            000004                          BSEL4=4
 1968            000005                          BSEL5=5
 1969            000006                          BSEL6=6
 1970            000007                          BSEL7=7
 1971            000010                          ECRC=BIT3
 1972            000040                          CRCHK=BIT5
 1973            000004                          TCRCE=BIT2
 1974            000002                          RCRCE=BIT1
 1975                                    ;   PROGRAM
 1976                                    ;----------------------
 1977  001240    000000                  REGIST: 0                       ;REGISTER NUMBER STORAGE FOR TYPE OUT
 1978                                            .EVEN
 1979  001242                            .ROEND:
 1980
 1981  001242    000000                  CLINE:  .WORD   0               ;CURRENT LINE POINTER
 1982  001244    000000                  CLINED: .WORD   0               ;CURRENT LINE POINTER DECIMAL
 1983  001246    000000                  $DVICE: .WORD   0               ;CURRENT DEVICE #
 1984  001250    000001                  CMFLAG: .WORD   BIT0                    ;PROGRAM COMMAND FLAG
 1985  001252    000000                  NXTLIN: .WORD   0                       ;FLAG FOR NEXT LINE TO BE TESTED
 1986  001254    000000                  LINBYT: .WORD   0                       ;HOLDS LINES ENABLED
 1987  001256    000000                  CRC32F: .WORD   0               ;CRC-32 FLAG
 1988  001260    000000                  CLKDAT: .WORD   0                       ;INT/EXT FLAG
 1989  001262    000000                  TSTNUM: .WORD   0               ;TEST # FOR ERROR REPORT
```

```
                                        .SBTTL  ERROR POINTER TABLE

                                        ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                        ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                        ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                        ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                        ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

                                        ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                        ;*      DH              ;;POINTS TO THE DATA HEADER
                                        ;*      DT              ;;POINTS TO THE DATA
                                        ;*      DF              ;;POINTS TO THE DATA FORMAT


001264                                  $ERRTB:
                                        ;BCRAD ERROR TABLE
001264                                  BCRAD.ERTBL:
001264  035124  040140  040624                  EM1,DH1,DT1,DF1
001272  041012
001274  035153  040140  040624                  EM2,DH1,DT1,DF1
001302  041012
001304  035200  040140  040624                  EM3,DH1,DT1,DF1
001312  041012
001314  035236  040140  040624                  EM4,DH1,DT1,DF1
001322  041012
001324  035311  040216  040644                  EM5,DH2,DT2,DF2
001332  041022
001334  035356  040216  040644                  EM6,DH2,DT2,DF2
001342  041022
001344  035420  040216  040644                  EM7,DH2,DT2,DF2
001352  041022
001354  035466  040216  040644                  EM8,DH2,DT2,DF2
001362  041022
001364  035564  040216  040644                  EM9,DH2,DT2,DF2
001372  041022
001374  035657  040140  040624                  EM10,DH1,DT1,DF1
001402  041012
001404  035726  040140  040624                  EM11,DH1,DT1,DF1
001412  041012
001414  036010  040216  040662                  EM12,DH2,DT3,DF2
001422  041022
001424  036064  040216  040662                  EM13,DH2,DT3,DF2
001432  041022
001434  036107  040216  040662                  EM14,DH2,DT3,DF2
001442  041022
001444  036147  040216  040662                  EM15,DH2,DT3,DF2
001452  041022
001454  036207  040216  040662                  EM16,DH2,DT3,DF2
001462  041022
001464  036231  040216  040662                  EM17,DH2,DT3,DF2
001472  041022
001474  036312  040216  040662                  EM18,DH2,DT3,DF2
001502  041022
001504  036372  040216  040662                  EM19,DH2,DT3,DF2
001512  041022
001514  036415  040216  040644                  EM20,DH2,DT2,DF2
```

H 4

ERROR POINTER TABLE

SEQ 0046

2046	001522	041022			
2047	001524	036462	040216	040662	EM21,DH2,DT3,DF2
2048	001532	041022			
2049	001534	036524	040322	040712	EM22,DH5,DT5,DF5
2050	001542	041034			
2051	001544	036611	040216	040644	EM23,DH2,DT2,DF2
2052	001552	041022			
2053	001554	036646	040216	040662	EM24,DH2,DT3,DF2
2054	001562	041022			
2055	001564	036704	040216	040662	EM25,DH2,DT3,DF2
2056	001572	041022			
2057	001574	036754	040266	040700	EM26,DH4,DT4,DF4
2058	001602	041030			
2059	001604	037022	040266	040700	EM27,DH4,DT4,DF4
2060	001612	041030			
2061	001614	037104	040266	040700	EM28,DH4,DT4,DF4
2062	001622	041030			
2063	001624	037155	040266	040700	EM29,DH4,DT4,DF4
2064	001632	041030			
2065	001634	037235	040266	040700	EM30,DH4,DT4,DF4
2066	001642	041030			
2067	001644	037315	040266	040700	EM31,DH4,DT4,DF4
2068	001652	041030			
2069	001654	037367	040266	040700	EM32,DH4,DT4,DF4
2070	001662	041030			
2071	001664	037445	040322	040712	EM33,DH5,DT5,DF5
2072	001672	041034			
2073	001674	037504	040266	040700	EM34,DH4,DT4,DF4
2074	001702	041030			
2075	001704	037526	040322	040712	EM35,DH5,DT5,DF5
2076	001712	041034			
2077	001714	037560	040322	040712	EM36,DH5,DT5,DF5
2078	001722	041034			
2079	001724	037627	040322	040712	EM37,DH5,DT5,DF5
2080	001732	041034			
2081	001734	037676	040366	040726	EM38,DH6,DT6,DF6
2082	001742	041041			
2083	001744	037742	040436	040744	EM39,DH7,DT7,DF7
2084	001752	041047			
2085	001754	040006	040526	040766	EM40,DH8,DT8,DF1
2086	001762	041012			
2087	001764	040113	040266	040700	EM41,DH4,DT4,DF4
2088	001772	041030			
2089	001774	040046	040216	040662	EM42,DH2,DT3,DF2
2090	002002	041022			

```
                                        ;BEGINNING ROUTINE
002004                                  .START:
002004                                  .BEGIN: ;CLEAN UP THE REST OF THE VARIABLE TABLE
                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
002004  012706  001100                          MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
002010  005026                          10$:    CLR     (R6)+           ;;CLEAR MEMORY LOCATION
002012  022706  001140                          CMP     #SWR,R6 ;;DONE?
002016  001374                                  BNE     .-6             ;;LOOP BACK IF NO
002020  012706  001100                          MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                        ;;INITIALIZE A FEW VECTORS
002024  012737  043564  000020                  MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
002032  012737  000340  000022                  MOV     #340,@#IOTVEC+2 ;;LEVEL 7
002040  012737  043114  000030                  MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
002046  012737  000340  000032                  MOV     #340,@#EMTVEC+2 ;;LEVEL 7
002054  012737  050052  000034                  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
002062  012737  000340  000036                  MOV     #340,@#TRAPVEC+2;LEVEL 7
002070  013737  035036  035030                  MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
002076  005037  001226                          CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
002102  112737  000001  001115                  MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
002110  012737  002110  001106                  MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
002116  012737  002116  001110                  MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                        ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                        ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
002124  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
002130  012737  002164  000004                  MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
002136  012737  177570  001140                  MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
002144  012737  177570  001142                  MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
002152  022777  177777  176760                  CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
002160  001012                                  BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
002162  000403                                  BR      65$             ;;BRANCH IF NO TIMEOUT
002164  012716  002172          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
002170  000002                          RTI
002172  012737  000176  001140  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
002200  012737  000174  001142                  MOV     #DISPREG,DISPLAY
002206  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
                                        .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
002212  005227  177777                          INC     #-1             ;;FIRST TIME?
002216  001041                                  BNE     67$             ;;BRANCH IF NO
002220  022737  035070  000042                  CMP     #$ENDAD,@#42    ;;ACT-11?
002226  001435                                  BEQ     67$             ;;BRANCH IF YES
002230  104401  002266                          TYPE    ,68$            ;;TYPE ASCIZ STRING
                                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
002234  005737  000042                          TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
002240  001006                                  BNE     69$             ;;BRANCH IF YES
002242  023727  001140  000176                  CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
002250  001005                                  BNE     70$             ;;BRANCH IF NO
002252  104406                                  GTSWR                   ;;GET SOFT-SWR SETTINGS
002254  000403                                  BR      70$
002256  112737  000001  001134  69$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
002264                          70$:
002264  000416                                  BR      67$             ;;GET OVER THE ASCIZ
                                        ;;68$:  .ASCIZ  <CRLF>#CZKMDA0 DMS11-DA STATIC #<CRLF>
```

```
                                67$:
002322                          
002322  004737  046542          JSR     PC,$SIZE
002326  004737  044736          JSR     PC,@#$TKINT             ;<<-------<<<<<<-
002332  012706  001100  .MONIT: MOV     #STACK,SP
002336  012737  000340  177776  MOV     #340,PSW
002344  104401  002352          TYPE    ,65$                    ;;TYPE ASCIZ STRING
002350  000420                  BR      64$                     ;;GET OVER THE ASCIZ
                        ;;65$:  .ASCIZ  <15><12>/(1)DIAG.TEST(2)CONFIGURATION /
002412                          64$:
002412  104412                  RDOCT
002414  012600                  MOV     (SP)+,R0
002416  022700  000001          CMP     #1,R0                   ;LOGIC TEST?
002422  001413                  BEQ     LOOP
002424  022700  000002          CMP     #2,R0                   ;GO TO LOGIC TEST LOOP
002430  001003                  BNE     3$                      ;CONFIGURATION MODE
002432  004737  041134          JSR     PC,MODTAB               ;MODIFY THE CONFIGURATION ROUTINE
002436  000735                  BR      .MONIT
002440                          3$:
002440  104401  002446          TYPE    ,67$                    ;;TYPE ASCIZ STRING
002444  000401                  BR      66$                     ;;GET OVER THE ASCIZ
                        ;;67$:  .ASCIZ  ??
002450                          66$:
002450  000730                  BR      .MONIT
                                ;STATIC TEST LOOPING ROUTINE
002452  012706  001100  LOOP:   MOV     #STACK,SP
002456  142737  000340  177776  BICB    #340,PSW                ;<--<<< ALLOW INTS.
002464  005037  001250          CLR     CMFLAG
002470  005037  001242          CLR     CLINE
002474  005037  001244          CLR     CLINED
002500  005037  001246          CLR     $DVICE
002504  012737  042460  041070  MOV     #KMCTAB-6,CURTAB
002512                          NXTUN:
002512  004737  042630          JSR     PC,GETKMC

002516  000401                  BR      1$                      ;B= NO KMCS ENABLED
002520  000426                  BR      CYCLE
002522  005737  041064  1$:     TST     KMCNT                   ;IS THERE KMC'S ENABLED
002526  001402                  BEQ     2$                      ;B=NO
002530  000137  035006          JMP     $EOP
002534                          2$:
002534  104401  002542          TYPE    ,65$                    ;;TYPE ASCIZ STRING
002540  000412                  BR      64$                     ;;GET OVER THE ASCIZ
                        ;;65$:  .ASCIZ  <15><12>/NO UNITS ENABLED/
002566                          64$:
002566  004737  041134          JSR     PC,MODTAB               ;SET UP CONFIGURATION TABLE
002572  000137  002452          JMP     LOOP                    ;NOW ENTER LOGIC TEST
002576                          CYCLE:
002576  004737  043002          JSR     PC,GETLIN
002602  000743                  BR      NXTUN                   ;FINISHED ALL LINES GET NEXT UNIT
002604  005037  001102          CLR     $TSTNM
002610  104401  002616          TYPE    ,65$                    ;;TYPE ASCIZ STRING
002614  000412                  BR      64$                     ;;GET OVER THE ASCIZ
                        ;;65$:  .ASCIZ  <15><12>/NOW TESTING LINE/
002642                          64$:
002642  013746  001244          MOV     CLINED,-(SP)            ;;SAVE CLINED FOR TYPEOUT
002646  104403                  TYPOS                           ;;GO TYPE--OCTAL ASCII
```

GET VALUE FOR SOFTWARE SWITCH REGISTER SEQ 0049

```
2203  002650     006                        .BYTE   6               ;;TYPE 6 DIGITS
2204  002651     000                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
2205  002652  104401  002660                TYPE    ,67$            ;;TYPE ASCIZ STRING
2206  002656  000404                        BR      66$             ;;GET OVER THE ASCIZ
2207                                ;;67$:  .ASCIZ  /  KMC #/
2208  002670                        66$:
2209  002670  013746  001246                MOV     $DVICE,-(SP)    ;;SAVE $DVICE FOR TYPEOUT
2210  002674  104405                        TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
2211  002676  104401  002704                TYPE    ,69$            ;;TYPE ASCIZ STRING
2212  002702  000402                        BR      68$             ;;GET OVER THE ASCIZ
2213                                ;;69$:  .ASCIZ  /  /
2214  002710                        68$:
2215  002710  000137  002714                JMP     LTEST           ;RUN LOGIC TESTS
```

```
                                .SBTTL  PRELIMINARY PRIMARY REGISTER READ TESTS
002714                          LTEST:
                                        ;****************************** TEST 1 ******************************
                                        ;*LINE SELECT REGISTER TEST
                                        ;*DO A MASTER CLEAR, VERIFY THAT ALL
                                        ;*BITS ARE IN THE CORRECT STATE OF REGISTER PRI11
                                        ;:***************************************************************************

                                        ;  TEST 1
                                        ;-------------
                                ;;*****************************************************************
002714  000004                  TST1:   SCOPE
002716  012737  000001  001102          MOV     #1,$TSTNM               ; LOAD THE NO. OF THIS TEST
002724  012737  000001  001262          MOV     #1,TSTNUM               ;# FOR TYPE OUT
002732  000005                  101$:   RESET                   ;CLEAR THE WORLD
002734  004737  044736                  JSR     PC,@#$TKINT                     ;INITIAL TTY INPUT
002740  005000                          CLR     R0
002742  105200                  10$:    INCB    R0
002744  001376                          BNE     10$
002746  005014                          CLR     (R4)            ;INITIALIZE THE CSR REGISTERS
002750  005064  000002                  CLR     2(R4)           ;       ...
002754  005064  000004                  CLR     4(R4)           ;       ...
002760  005064  000006                  CLR     6(R4)           ;       ...
002764  012737  000011  001240  102$:   MOV     #PRI11,REGIST   ;SAVE THE REGISTER FOR TYPE OUT
002772  104420  000011                  READ    ,PRI11                  ;READ PRI11
002776  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
003002  005037  001204                  CLR     $TMP1           ;PUT "EXPECTED" IN $TMP1
003006  023737  001202  001204          CMP     $TMP0,$TMP1             ;COMPARE RESULTS
003014  001403                          BEQ     TST2            ;;
003016  104001                          ERROR   1               ;ERROR IN  PRI11
003020  000137  002732                  JMP     101$            ;LOOP ON ERROR


                                        ;****************************** TEST 2 ******************************
                                        ;*SECONDARY SELECT REGISTER TEST
                                        ;*DO A MASTER CLEAR, VERIFY THAT ALL
                                        ;*BITS ARE IN THE CORRECT STATE OF REGISTER PRI12
                                        ;:***************************************************************************

                                        ;  TEST 2
                                        ;-------------
                                ;;*****************************************************************
003024  000004                  TST2:   SCOPE
003026  012737  000002  001102          MOV     #2,$TSTNM               ; LOAD THE NO. OF THIS TEST
003034  012737  000002  001262          MOV     #2,TSTNUM               ;# FOR TYPE OUT
003042  000005                  101$:   RESET                   ;CLEAR THE WORLD
003044  004737  044736                  JSR     PC,@#$TKINT                     ;INITIAL TTY INPUT
003050  005000                          CLR     R0
003052  105200                  10$:    INCB    R0
003054  001376                          BNE     10$
003056  005014                          CLR     (R4)            ;INITIALIZE THE CSR REGISTERS
003060  005064  000002                  CLR     2(R4)           ;       ...
003064  005064  000004                  CLR     4(R4)           ;       ...
003070  005064  000006                  CLR     6(R4)           ;       ...
```

```
2272  003074  012737  000012  001240  102$:   MOV     #PRI12,REGIST   ;SAVE THE REGISTER FOR TYPE OUT
2273  003102  104420  000012                  READ    ,PRI12                  ;READ PRI12
2274  003106  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
2275  003112  005037  001204                  CLR     $TMP1           ;PUT "EXPECTED" IN $TMP1
2276  003116  023737  001202  001204          CMP     $TMP0,$TMP1             ;COMPARE RESULTS
2277  003124  001403                          BEQ     TST3            ;;
2278  003126  104001                          ERROR   1               ;ERROR IN  PRI12
2279  003130  000137  003042                  JMP     101$            ;LOOP ON ERROR
2280
2281
2282                                          ;****************************** TEST 3 ******************************
2283                                          ;*LINE STATUS REGISTER TEST
2284                                          ;*DO A LINE CLEAR, VERIFY THAT ALL
2285                                          ;*BITS ARE IN THE CORRECT STATE OF REGISTER PRI13
2286                                          ;;**************************************************************
2287
2288                                          ;  TEST 3
2289                                          ;--------------
2290                                  ;;**************************************************************
2291  003134  000004                  TST3:   SCOPE
2292  003136  012737  000003  001102          MOV     #3,$TSTNM               ; LOAD THE NO. OF THIS TEST
2293  003144  012737  000003  001262          MOV     #3,TSTNUM               ;# FOR TYPE OUT
2294  003152  104417                  101$:   LRESET
2295  003154  012737  000013  001240          MOV     #PRI13,REGIST   ;SAVE THE REGISTER FOR TYPE OUT
2296  003162  104420  000013                  READ    ,PRI13                  ;READ PRI13
2297  003166  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
2298  003172  012737  000020  001204          MOV     #20,$TMP1               ;PUT "EXPECTED" IN $TMP1
2299  003200  023737  001202  001204          CMP     $TMP0,$TMP1             ;COMPARE RESULTS
2300  003206  001403                          BEQ     TST4            ;;
2301  003210  104001                          ERROR   1               ;ERROR IN  PRI13
2302  003212  000137  003152                  JMP     101$            ;LOOP ON ERROR
2303
2304
2305                                          ;****************************** TEST 4 ******************************
2306                                          ;*LINE CONTROL REGISTER TEST
2307                                          ;*DO A LINE CLEAR, VERIFY THAT ALL
2308                                          ;*BITS ARE IN THE CORRECT STATE OF REGISTER PRI14
2309                                          ;;**************************************************************
2310
2311                                          ;  TEST 4
2312                                          ;--------------
2313                                  ;;**************************************************************
2314  003216  000004                  TST4:   SCOPE
2315  003220  012737  000004  001102          MOV     #4,$TSTNM               ; LOAD THE NO. OF THIS TEST
2316  003226  012737  000004  001262          MOV     #4,TSTNUM               ;# FOR TYPE OUT
2317  003234  104417                  101$:   LRESET
2318  003236  012737  000014  001240          MOV     #PRI14,REGIST   ;SAVE THE REGISTER FOR TYPE OUT
2319  003244  104420  000014                  READ    ,PRI14                  ;READ PRI14
2320  003250  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
2321  003254  005037  001204                  CLR     $TMP1           ;PUT "EXPECTED" IN $TMP1
2322  003260  023737  001202  001204          CMP     $TMP0,$TMP1             ;COMPARE RESULTS
2323  003266  001403                          BEQ     TST5            ;;
2324  003270  104001                          ERROR   1               ;ERROR IN  PRI14
2325  003272  000137  003234                  JMP     101$            ;LOOP ON ERROR
2326
2327
```

```
                                        ;****************************** TEST 5 ******************************
                                        ;*TRANSMIT STATUS MX REGISTER TEST
                                        ;*DO A MASTER CLEAR, VERIFY THAT ALL
                                        ;*BITS ARE IN THE CORRECT STATE OF REGISTER PRI16
                                        ;:************************************************************************

                                        ;  TEST 5
                                        ;--------------
                                    ;;**************************************************************************
003276  000004                  TST5:   SCOPE
003300  012737  000005  001102          MOV     #5,$TSTNM               ; LOAD THE NO. OF THIS TEST
003306  012737  000005  001262          MOV     #5,TSTNUM               ;# FOR TYPE OUT
003314  000005                  101$:   RESET                   ;CLEAR THE WORLD
003316  004737  044736                  JSR     PC,@#$TKINT                     ;INITIAL TTY INPUT
003322  005000                          CLR     R0
003324  105200                  10$:    INCB    R0
003326  001376                          BNE     10$
003330  005014                          CLR     (R4)            ;INITIALIZE THE CSR REGISTERS
003332  005064  000002                  CLR     2(R4)           ;       ...
003336  005064  000004                  CLR     4(R4)           ;       ...
003342  005064  000006                  CLR     6(R4)           ;       ...
003346  012737  000016  001240  102$:   MOV     #PRI16,REGIST   ;SAVE THE REGISTER FOR TYPE OUT
003354  104420  000016                  READ    ,PRI16                  ;READ PRI16
003360  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
003364  012737  000377  001204          MOV     #377,$TMP1              ;PUT "EXPECTED" IN $TMP1
003372  004737  047274                  JSR     PC,.MASK        ;CALCULATE THE MASK
003376  043737  047352  001202          BIC     MASK,$TMP0      ;MASK OUT LINES NOT INUSE
003404  043737  047352  001204          BIC     MASK,$TMP1      ;MASK OUT LINES NOT IN USE
003412  023737  001202  001204          CMP     $TMP0,$TMP1             ;COMPARE RESULTS
003420  001403                          BEQ     TST6            ;;
003422  104001                          ERROR   1               ;ERROR IN  PRI16
003424  000137  003314                  JMP     101$            ;LOOP ON ERROR


                                        ;****************************** TEST 6 ******************************
                                        ;*RECEIVE STATUS MX REGISTER TEST
                                        ;*DO A MASTER CLEAR, VERIFY THAT ALL
                                        ;*BITS ARE IN THE CORRECT STATE OF REGISTER PRI17
                                        ;:************************************************************************

                                        ;  TEST 6
                                        ;--------------
                                    ;;**************************************************************************
003430  000004                  TST6:   SCOPE
003432  012737  000006  001102          MOV     #6,$TSTNM               ; LOAD THE NO. OF THIS TEST
003440  012737  000006  001262          MOV     #6,TSTNUM               ;# FOR TYPE OUT
003446  000005                  101$:   RESET                   ;CLEAR THE WORLD
003450  004737  044736                  JSR     PC,@#$TKINT                     ;INITIAL TTY INPUT
003454  005000                          CLR     R0
003456  105200                  10$:    INCB    R0
003460  001376                          BNE     10$
003462  005014                          CLR     (R4)            ;INITIALIZE THE CSR REGISTERS
003464  005064  000002                  CLR     2(R4)           ;       ...
003470  005064  000004                  CLR     4(R4)           ;       ...
003474  005064  000006                  CLR     6(R4)           ;       ...
003500  012737  000017  001240  102$:   MOV     #PRI17,REGIST   ;SAVE THE REGISTER FOR TYPE OUT
```

```
 2384  003506  104420  000017                READ    ,PRI17                  ;READ PRI17
 2385  003512  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
 2386  003516  005037  001204                CLR     $TMP1           ;PUT "EXPECTED" IN $TMP1
 2387  003522  004737  047274                JSR     PC,.MASK        ;CALCULATE THE MASK
 2388  003526  043737  047352  001202        BIC     MASK,$TMP0      ;MASK OUT LINES NOT INUSE
 2389  003534  043737  047352  001204        BIC     MASK,$TMP1      ;MASK OUT LINES NOT IN USE
 2390  003542  023737  001202  001204        CMP     $TMP0,$TMP1             ;COMPARE RESULTS
 2391  003550  001403                        BEQ     TST7            ;;
 2392  003552  104001                        ERROR   1               ;ERROR IN  PRI17
 2393  003554  000137  003446                JMP     101$            ;LOOP ON ERROR
```

```
                                .SBTTL  FLOATING ONE TEST OF PRIMARY REGISTER 11,12,13


                                        ;***************************** TEST 7 *****************************
                                        ;*PRIMARY REGISTER WRITE/READ TEST
                                        ;*FLOAT A 1 THROUGH PRI11
                                        ;*THE FOLLOWING BITS ARE NOT TESTED
                                        ;* **<BIT0!BIT1!BIT2!BIT3!BIT7>**
                                        ;:*************************************************************************

                                        ;  TEST 7
                                        ;-------------
                                ;;*************************************************************************
003560  000004                  TST7:   SCOPE
003562  012737  000007  001102          MOV     #7,$TSTNM               ; LOAD THE NO. OF THIS TEST
003570  012737  000007  001262          MOV     #7,TSTNUM               ;# FOR TYPE OUT
003576  012737  000011  001240          MOV     #PRI11,REGIST   ;SAVE REGISTER FOR TYPE OUT
003604  012737  000200  003622          MOV     #BIT7,66$       ;START WITH BIT 7
003612  000004                          SCOPE
003614  104414                  101$:   MSTCLR
003616                          64$:
003616  104416  000011                  WRITE   ,PRI11          ;WRITE DATA INTO PRI11
003622  000000                  66$:    0                       ;DATA
003624  104420  000011                  READ    ,PRI11          ;READ PRI11
003630  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
003634  013737  003622  001204          MOV     66$,$TMP1       ;PUT EXPECTED INTO $TMP1
003642  042737  000217  001202          BIC     #BIT0!BIT1!BIT2!BIT3!BIT7,$TMP0         ;CLEAR DON'T CARE
003650  042737  000217  001204          BIC     #BIT0!BIT1!BIT2!BIT3!BIT7,$TMP1         ;CLEAR UNWANTED BITS
003656  123737  001202  001204          CMPB    $TMP0,$TMP1                     ;DATA CORRECT?
003664  001403                          BEQ     65$             ;BR IF YES
003666  104002                          ERROR   2               ;FLOATING ONE'S ERROR OF PRI11
003670  000137  003614                  JMP     101$            ;LOOP ON ERROR
003674                          65$:
003674  006237  003622                  ASR     66$             ;SHIFT BIT IN 66$
003700  001346                          BNE     64$             ;IF 66$=0 THEN DONE
003702  000400                          BR      TST10           ;;


                                        ;***************************** TEST 10 *****************************
                                        ;*PRIMARY REGISTER WRITE/READ TEST
                                        ;*FLOAT A 1 THROUGH PRI12
                                        ;*THE FOLLOWING BITS ARE NOT TESTED
                                        ;* **<BIT0!BIT1!BIT2!BIT3>**
                                        ;:*************************************************************************

                                        ;  TEST 10
                                        ;-------------
                                ;;*************************************************************************
003704  000004                  TST10:  SCOPE
003706  012737  000010  001102          MOV     #10,$TSTNM              ; LOAD THE NO. OF THIS TEST
003714  012737  000010  001262          MOV     #10,TSTNUM              ;# FOR TYPE OUT
003722  012737  000012  001240          MOV     #PRI12,REGIST   ;SAVE REGISTER FOR TYPE OUT
003730  012737  000200  003746          MOV     #BIT7,66$       ;START WITH BIT 7
003736  000004                          SCOPE
003740  104414                  101$:   MSTCLR
003742                          64$:
```

```
2450  003742  104416  000012            WRITE   ,PRI12          ;WRITE DATA INTO PRI12
2451  003746  000000              66$:  0                       ;DATA
2452  003750  104420  000012            READ    ,PRI12          ;READ PRI12
2453  003754  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
2454  003760  013737  003746  001204      MOV     66$,$TMP1       ;PUT EXPECTED INTO $TMP1
2455  003766  042737  000017  001202      BIC     #BIT0!BIT1!BIT2!BIT3,$TMP0              ;CLEAR DON'T CARE
2456  003774  042737  000017  001204      BIC     #BIT0!BIT1!BIT2!BIT3,$TMP1              ;CLEAR UNWANTED BITS
2457  004002  123737  001202  001204      CMPB    $TMP0,$TMP1             ;DATA CORRECT?
2458  004010  001403                      BEQ     65$             ;BR IF YES
2459  004012  104002                      ERROR   2               ;FLOATING ONE'S ERROR OF PRI12
2460  004014  000137  003740              JMP     101$            ;LOOP ON ERROR
2461  004020                        65$:
2462  004020  006237  003746            ASR     66$             ;SHIFT BIT IN 66$
2463  004024  001346                    BNE     64$             ;IF 66$=0 THEN DONE
2464  004026  000400                    BR      TST11           ;;
2465
2466
2467                                    ;***************************** TEST 11 ******************************
2468                                    ;*PRIMARY REGISTER WRITE/READ TEST
2469                                    ;*FLOAT A 1 THROUGH PRI13
2470                                    ;*THE FOLLOWING ARE READ ONLY BITS
2471                                    ;* **<BIT0!BIT1!BI12!BIT3!BIT4!BIT6!BIT7>**
2472                                    ;*AND SHOULD BE READ AS ZERO
2473                                    ;;*************************************************************************
2474
2475                                    ;  TEST 11
2476                                    ;--------------
2477                                  ;;*************************************************************************
2478  004030  000004              TST11:  SCOPE
2479  004032  012737  000011  001102      MOV     #11,$TSTNM              ; LOAD THE NO. OF THIS TEST
2480  004040  012737  000011  001262      MOV     #11,TSTNUM              ;# FOR TYPE OUT
2481  004046  105737  001256              TSTB    CRC32F          ; CRC32 ENABLED?
2482  004052  100043                      BPL     TST12           ;;
2483  004054  012737  000013  001240      MOV     #PRI13,REGIST   ;SAVE REGISTER FOR TYPE OUT
2484  004062  012737  000200  004100      MOV     #BIT7,66$       ;START WITH BIT 7
2485  004070  000004                      SCOPE
2486  004072  104414              101$:   MSTCLR
2487  004074                        64$:
2488  004074  104416  000013            WRITE   ,PRI13          ;WRITE DATA INTO PRI13
2489  004100  000000              66$:  0                       ;DATA
2490  004102  104420  000013            READ    ,PRI13          ;READ PRI13
2491  004106  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
2492  004112  013737  004100  001204      MOV     66$,$TMP1       ;PUT EXPECTED INTO $TMP1
2493  004120  042737  000337  001204      BIC     #BIT0!BIT1!BIT2!BIT3!BIT4!BIT6!BIT7,$TMP1       ;CLEAR ALL READ ONLY BIT
2494  004126  052737  000020  001204      BIS     #TBMT,$TMP1     ;TBMT SHOULD BE SET
2495  004134  123737  001202  001204      CMPB    $TMP0,$TMP1             ;DATA CORRECT?
2496  004142  001403                      BEQ     65$             ;BR IF YES
2497  004144  104002                      ERROR   2               ;FLOATING ONE'S ERROR OF PRI13
2498  004146  000137  004072              JMP     101$            ;LOOP ON ERROR
2499  004152                        65$:
2500  004152  006237  004100            ASR     66$             ;SHIFT BIT IN 66$
2501  004156  001346                    BNE     64$             ;IF 66$=0 THEN DONE
2502  004160  000400                    BR      TST12           ;;
2503
2504
2505                                    ;***************************** TEST 12 ******************************
```

```
                                        ;*PRIMARY REGISTER WRITE/READ TEST
                                        ;*FLOAT A 1 THROUGH PRI14
                                        ;*THE FOLLOWING ARE READ ONLY BITS
                                        ;* **<BIT1!BIT2>**
                                        ;*AND SHOULD BE READ AS ZERO
                                        ;;*************************************************************************

                                        ;  TEST 12
                                        ;--------------
                                    ;;*****************************************************************
004162  000004                  TST12:  SCOPE
004164  012737  000012  001102          MOV     #12,$TSTNM              ; LOAD THE NO. OF THIS TEST
004172  012737  000012  001262          MOV     #12,TSTNUM              ;# FOR TYPE OUT
004200  012737  000014  001240          MOV     #PRI14,REGIST   ;SAVE REGISTER FOR TYPE OUT
004206  012737  000200  004224          MOV     #BIT7,66$       ;START WITH BIT 7
004214  000004                          SCOPE
004216  104414                  101$:   MSTCLR
004220                          64$:
004220  104416  000014                  WRITE   ,PRI14          ;WRITE DATA INTO PRI14
004224  000000                  66$:    0                       ;DATA
004226  104420  000014                  READ    ,PRI14          ;READ PRI14
004232  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
004236  013737  004224  001204          MOV     66$,$TMP1       ;PUT EXPECTED INTO $TMP1
004244  042737  000006  001204          BIC     #BIT1!BIT2,$TMP1        ;CLEAR ALL READ ONLY BITS
004252  105737  001256                  TSTB    CRC32F          ; CRC32 ENABLED?
004256  100406                          BMI     110$            ; SKIP IF YES
004260  042737  000010  001202          BIC     #BIT3,$TMP0     ; CLEAR CRC32 EN.
004266  042737  000010  001204          BIC     #BIT3,$TMP1
004274                          110$:
004274  123737  001202  001204          CMPB    $TMP0,$TMP1             ;DATA CORRECT?
004302  001403                          BEQ     65$             ;BR IF YES
004304  104002                          ERROR   2               ;FLOATING ONE'S ERROR OF PRI14
004306  000137  004216                  JMP     101$            ;LOOP ON ERROR
004312                          65$:
004312  006237  004224                  ASR     66$             ;SHIFT BIT IN 66$
004316  001340                          BNE     64$             ;IF 66$=0 THEN DONE
004320  000400                          BR      TST13           ;;
```

```
                                        .SBTTL  PRELIMINARY SECONDARY REGISTER READ ONLY TEST


                                                ;****************************** TEST 13 ******************************
                                                ;*RECEIVE DATA BUFFER SECONDARY READ TEST
                                                ;*DOES A LINE RESET
                                                ;*READS SECONDARY REGISTER SEC0
                                                ;;************************************************************************

                                                ;  TEST 13
                                                ;--------------
                                        ;;************************************************************************
004322  000004                          TST13:  SCOPE
004324  012737  000013  001102                  MOV     #13,$TSTNM              ; LOAD THE NO. OF THIS TEST
004332  012737  000013  001262                  MOV     #13,TSTNUM              ;# FOR TYPE OUT
004340  104417                          1$:     LRESET
004342  012737  000000  001240                  MOV     #SEC0,REGIST    ;SAVE REGISTER FOR TYP OUT
004350  104420  000000                          READ    ,SEC0           ;READ SECONDARY DATA
004354  012637  001202                          MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
004360  012737  000000  001204                  MOV     #0,$TMP1        ;LOAD EXPECTED INTO $TMP1
004366  023737  001202  001204                  CMP     $TMP0,$TMP1             ;COMPARE RESULTS
004374  001403                                  BEQ     TST14           ;;
004376  104003                                  ERROR   3               ;READ ERROR OF SEC0
004400  000137  004340                          JMP     1$              ;LOOP ON ERROR


                                                ;****************************** TEST 14 ******************************
                                                ;*RECIEVE STATUS SECONDARY READ TEST
                                                ;*READS SECONDARY REGISTER SEC1
                                                ;;************************************************************************

                                                ;  TEST 14
                                                ;--------------
                                        ;;************************************************************************
004404  000004                          TST14:  SCOPE
004406  012737  000014  001102                  MOV     #14,$TSTNM              ; LOAD THE NO. OF THIS TEST
004414  012737  000014  001262                  MOV     #14,TSTNUM              ;# FOR TYPE OUT
004422  104417                          1$:     LRESET
004424  012737  000001  001240                  MOV     #SEC1,REGIST    ;SAVE REGISTER FOR TYP OUT
004432  104420  000001                          READ    ,SEC1           ;READ SECONDARY DATA
004436  012637  001202                          MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
004442  012737  000000  001204                  MOV     #0,$TMP1        ;LOAD EXPECTED INTO $TMP1
004450  023737  001202  001204                  CMP     $TMP0,$TMP1             ;COMPARE RESULTS
004456  001403                                  BEQ     TST15           ;;
004460  104003                                  ERROR   3               ;READ ERROR OF SEC1
004462  000137  004422                          JMP     1$              ;LOOP ON ERROR


                                                ;****************************** TEST 15 ******************************
                                                ;*TRANSMIT DATA BUFFER SECONDARY READ TEST
                                                ;*READS SECONDARY REGISTER SEC2
                                                ;;************************************************************************

                                                ;  TEST 15
                                                ;--------------
                                        ;;************************************************************************
```

```
2599  004466  000004                    TST15:  SCOPE
2600  004470  012737  000015  001102            MOV     #15,$TSTNM              ; LOAD THE NO. OF THIS TEST
2601  004476  012737  000015  001262            MOV     #15,TSTNUM              ;# FOR TYPE OUT
2602  004504  104417                    1$:     LRESET
2603  004506  012737  000002  001240            MOV     #SEC2,REGIST    ;SAVE REGISTER FOR TYP OUT
2604  004514  104420  000002                    READ    ,SEC2           ;READ SECONDARY DATA
2605  004520  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
2606  004524  012737  000000  001204            MOV     #0,$TMP1        ;LOAD EXPECTED INTO $TMP1
2607  004532  023737  001202  001204            CMP     $TMP0,$TMP1             ;COMPARE RESULTS
2608  004540  001403                            BEQ     TST16           ;;
2609  004542  104003                            ERROR   3               ;READ ERROR OF SEC2
2610  004544  000137  004504                    JMP     1$              ;LOOP ON ERROR
2611
2612
2613                                    ;****************************** TEST 16 ******************************
2614                                    ;*TRANSMIT STATUS SECONDARY READ TEST
2615                                    ;*READS SECONDARY REGISTER SEC3
2616                                    ;*THE FOLLOWING BIT(S) ARE NOT CHECKED
2617                                    ;* ** <BIT6!BIT5!BIT4> **
2618                                    ;;*************************************************************************
2619
2620                                    ;  TEST 16
2621                                    ;-------------
2622                                ;;*****************************************************************
2623  004550  000004                    TST16:  SCOPE
2624  004552  012737  000016  001102            MOV     #16,$TSTNM              ; LOAD THE NO. OF THIS TEST
2625  004560  012737  000016  001262            MOV     #16,TSTNUM              ;# FOR TYPE OUT
2626  004566  104417                    1$:     LRESET
2627  004570  012737  000003  001240            MOV     #SEC3,REGIST    ;SAVE REGISTER FOR TYP OUT
2628  004576  104420  000003                    READ    ,SEC3           ;READ SECONDARY DATA
2629  004602  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
2630  004606  012737  000000  001204            MOV     #0,$TMP1        ;LOAD EXPECTED INTO $TMP1
2631  004614  042737  000160  001202            BIC     #BIT6!BIT5!BIT4,$TMP0   ;MASK DON'T CARE BITS
2632  004622  023737  001202  001204            CMP     $TMP0,$TMP1             ;COMPARE RESULTS
2633  004630  001403                            BEQ     TST17           ;;
2634  004632  104003                            ERROR   3               ;READ ERROR OF SEC3
2635  004634  000137  004566                    JMP     1$              ;LOOP ON ERROR
2636
2637
2638                                    ;****************************** TEST 17 ******************************
2639                                    ;*SYNC SECONDARY ADDRESS SECONDARY READ TEST
2640                                    ;*READS SECONDARY REGISTER SEC4
2641                                    ;;*************************************************************************
2642
2643                                    ;  TEST 17
2644                                    ;-------------
2645                                ;;*****************************************************************
2646  004640  000004                    TST17:  SCOPE
2647  004642  012737  000017  001102            MOV     #17,$TSTNM              ; LOAD THE NO. OF THIS TEST
2648  004650  012737  000017  001262            MOV     #17,TSTNUM              ;# FOR TYPE OUT
2649  004656  104417                    1$:     LRESET
2650  004660  012737  000004  001240            MOV     #SEC4,REGIST    ;SAVE REGISTER FOR TYP OUT
2651  004666  104420  000004                    READ    ,SEC4           ;READ SECONDARY DATA
2652  004672  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
2653  004676  012737  000000  001204            MOV     #0,$TMP1        ;LOAD EXPECTED INTO $TMP1
2654  004704  023737  001202  001204            CMP     $TMP0,$TMP1             ;COMPARE RESULTS
```

```
                                        BEQ     TST20           ;;
                                        ERROR   3               ;READ ERROR OF SEC4
004716  000137  004656                JMP     1$              ;LOOP ON ERROR


                                        ;***************************** TEST 20 ******************************
                                        ;*MODE CONTROL SECONDARY READ TEST
                                        ;*READS SECONDARY REGISTER SEC5
                                        ;;************************************************************************

                                        ;  TEST 20
                                        ;--------------
                                ;;**************************************************************
                                TST20:  SCOPE
004722  000004
004724  012737  000020  001102          MOV     #20,$TSTNM              ; LOAD THE NO. OF THIS TEST
004732  012737  000020  001262          MOV     #20,TSTNUM              ;# FOR TYPE OUT
004740  104417                        1$:     LRESET
004742  012737  000005  001240          MOV     #SEC5,REGIST    ;SAVE REGISTER FOR TYP OUT
004750  104420  000005                  READ    ,SEC5           ;READ SECONDARY DATA
004754  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
004760  012737  000000  001204          MOV     #0,$TMP1        ;LOAD EXPECTED INTO $TMP1
004766  023737  001202  001204          CMP     $TMP0,$TMP1             ;COMPARE RESULTS
004774  001403                          BEQ     TST21           ;;
004776  104003                          ERROR   3               ;READ ERROR OF SEC5
005000  000137  004740                  JMP     1$              ;LOOP ON ERROR


                                        ;***************************** TEST 21 ******************************
                                        ;*CHARACTER LENGTH SECONDARY READ TEST
                                        ;*READS SECONDARY REGISTER SEC7
                                        ;;************************************************************************

                                        ;  TEST 21
                                        ;--------------
                                ;;**************************************************************
005004  000004                  TST21:  SCOPE
005006  012737  000021  001102          MOV     #21,$TSTNM              ; LOAD THE NO. OF THIS TEST
005014  012737  000021  001262          MOV     #21,TSTNUM              ;# FOR TYPE OUT
005022  104417                        1$:     LRESET
005024  012737  000007  001240          MOV     #SEC7,REGIST    ;SAVE REGISTER FOR TYP OUT
005032  104420  000007                  READ    ,SEC7           ;READ SECONDARY DATA
005036  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
005042  012737  000000  001204          MOV     #0,$TMP1        ;LOAD EXPECTED INTO $TMP1
005050  023737  001202  001204          CMP     $TMP0,$TMP1             ;COMPARE RESULTS
005056  001403                          BEQ     TST22           ;;
005060  104003                          ERROR   3               ;READ ERROR OF SEC7
005062  000137  005022                  JMP     1$              ;LOOP ON ERROR
```

```
                                .SBTTL  RECEIVE STATUS REGISTER WRITE TEST


                                ;************************** TEST 22 ****************************
                                ;*RECEIVE STATUS REGISTER  SECONDARY) BIT TEST
                                ;*THIS TEST DOES A WRITE TO THE READ ONLY BITS
                                ;* ** RSOM **
                                ;*AND THEN CHECKS TO MAKE SURE IT WAS NOT WRITTEN
                                ;:***************************************************************

                                ;  TEST 22
                                ;-------------
                                ;;***************************************************************
005066  000004          TST22:  SCOPE
005070  012737  000022  001102          MOV     #22,$TSTNM              ; LOAD THE NO. OF THIS TEST
005076  012737  000022  001262          MOV     #22,TSTNUM              ;# FOR TYPE OUT
005104  104417          1$:     LRESET
005106  104416  000001  000001          WRITE   ,SEC1,RSOM      ;WRITE THE READ ONLY BIT RSOM
005114  012737  000001  001240          MOV     #SEC1,REGIST    ;SAVE REGISTER FOR TYP OUT
005122  104420  000001                  READ    ,SEC1
005126  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
005132  042737  177776  001202          BIC     #^C<RSOM>,$TMP0 ;CLEAR UNWANTED BITS
005140  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED INTO $TMP1
005144  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST IF SET
005152  001403                          BEQ     TST23           ;;
005154  104004                          ERROR   4               ;ERROR RSOM CAN BE WRITTEN
005156  000137  005104                  JMP     1$              ;LOOP ON ERROR


                                ;************************** TEST 23 ****************************
                                ;*RECEIVE STATUS REGISTER (SECONDARY) BIT TEST
                                ;*THIS TEST DOES A WRITE TO THE READ ONLY BITS
                                ;* ** REOM **
                                ;*AND THEN CHECKS TO MAKE SURE IT WAS NOT WRITTEN
                                ;:***************************************************************

                                ;  TEST 23
                                ;-------------
                                ;;***************************************************************
005162  000004          TST23:  SCOPE
005164  012737  000023  001102          MOV     #23,$TSTNM              ; LOAD THE NO. OF THIS TEST
005172  012737  000023  001262          MOV     #23,TSTNUM              ;# FOR TYPE OUT
005200  104417          1$:     LRESET
005202  104416  000001  000002          WRITE   ,SEC1,REOM      ;WRITE THE READ ONLY BIT REOM
005210  012737  000001  001240          MOV     #SEC1,REGIST    ;SAVE REGISTER FOR TYP OUT
005216  104420  000001                  READ    ,SEC1
005222  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
005226  042737  177775  001202          BIC     #^C<REOM>,$TMP0 ;CLEAR UNWANTED BITS
005234  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED INTO $TMP1
005240  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST IF SET
005246  001403                          BEQ     TST24           ;;
005250  104004                          ERROR   4               ;ERROR REOM CAN BE WRITTEN
005252  000137  005200                  JMP     1$              ;LOOP ON ERROR


                                ;************************** TEST 24 ****************************
```

```
                                        ;*RECEIVE STATUS REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A WRITE TO THE READ ONLY BITS
                                        ;* ** RXGA **
                                        ;*AND THEN CHECKS TO MAKE SURE IT WAS NOT WRITTEN
                                        ;;*****************************************************************************

                                        ;  TEST 24
                                        ;--------------
                                    ;;*****************************************************************
005256  000004                      TST24:  SCOPE
005260  012737  000024  001102              MOV     #24,$TSTNM              ; LOAD THE NO. OF THIS TEST
005266  012737  000024  001262              MOV     #24,TSTNUM              ;# FOR TYPE OUT
005274  104417                      1$:     LRESET
005276  104416  000001  000004              WRITE   ,SEC1,RXGA      ;WRITE THE READ ONLY BIT RXGA
005304  012737  000001  001240              MOV     #SEC1,REGIST    ;SAVE REGISTER FOR TYP OUT
005312  104420  000001                      READ    ,SEC1
005316  012637  001202                      MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
005322  042737  177773  001202              BIC     #^C<RXGA>,$TMP0 ;CLEAR UNWANTED BITS
005330  005037  001204                      CLR     $TMP1           ;LOAD EXPECTED INTO $TMP1
005334  023737  001202  001204              CMP     $TMP0,$TMP1     ;TEST IF SET
005342  001403                              BEQ     TST25           ;;
005344  104004                              ERROR   4               ;ERROR RXGA CAN BE WRITTEN
005346  000137  005274                      JMP     1$              ;LOOP ON ERROR

                                        ;****************************** TEST 25 ******************************
                                        ;*RECEIVE STATUS REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A WRITE TO THE READ ONLY BITS
                                        ;* ** ROR **
                                        ;*AND THEN CHECKS TO MAKE SURE IT WAS NOT WRITTEN
                                        ;;*****************************************************************************

                                        ;  TEST 25
                                        ;--------------
                                    ;;*****************************************************************
005352  000004                      TST25:  SCOPE
005354  012737  000025  001102              MOV     #25,$TSTNM              ; LOAD THE NO. OF THIS TEST
005362  012737  000025  001262              MOV     #25,TSTNUM              ;# FOR TYPE OUT
005370  104417                      1$:     LRESET
005372  104416  000001  000010              WRITE   ,SEC1,ROR       ;WRITE THE READ ONLY BIT ROR
005400  012737  000001  001240              MOV     #SEC1,REGIST    ;SAVE REGISTER FOR TYP OUT
005406  104420  000001                      READ    ,SEC1
005412  012637  001202                      MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
005416  042737  177767  001202              BIC     #^C<ROR>,$TMP0  ;CLEAR UNWANTED BITS
005424  005037  001204                      CLR     $TMP1           ;LOAD EXPECTED INTO $TMP1
005430  023737  001202  001204              CMP     $TMP0,$TMP1     ;TEST IF SET
005436  001403                              BEQ     TST26           ;;
005440  104004                              ERROR   4               ;ERROR ROR CAN BE WRITTEN
005442  000137  005370                      JMP     1$              ;LOOP ON ERROR


                                        ;****************************** TEST 26 ******************************
                                        ;*RECEIVE STATUS REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A WRITE TO THE READ ONLY BITS
                                        ;* ** ABCA **
                                        ;*AND THEN CHECKS TO MAKE SURE IT WAS NOT WRITTEN
```

```
                                        ;;*************************************************************************
                                        ;  TEST 26
                                        ;--------------
                                    ;;*************************************************************************
005446  000004                  TST26:  SCOPE
005450  012737  000026  001102          MOV     #26,$TSTNM              ; LOAD THE NO. OF THIS TEST
005456  012737  000026  001262          MOV     #26,TSTNUM              ;# FOR TYPE OUT
005464  104417                  1$:     LRESET
005466  104416  000001  000020          WRITE   ,SEC1,ABCA      ;WRITE THE READ ONLY BIT ABCA
005474  012737  000001  001240          MOV     #SEC1,REGIST    ;SAVE REGISTER FOR TYP OUT
005502  104420  000001                  READ    ,SEC1
005506  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
005512  042737  177757  001202          BIC     #^C<ABCA>,$TMP0 ;CLEAR UNWANTED BITS
005520  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED INTO $TMP1
005524  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST IF SET
005532  001403                          BEQ     TST27           ;;
005534  104004                          ERROR   4               ;ERROR ABCA CAN BE WRITTEN
005536  000137  005464                  JMP     1$              ;LOOP ON ERROR


                                        ;****************************** TEST 27 ******************************
                                        ;*RECEIVE STATUS REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A WRITE TO THE READ ONLY BITS
                                        ;* ** ABCB **
                                        ;*AND THEN CHECKS TO MAKE SURE IT WAS NOT WRITTEN
                                        ;;*************************************************************************

                                        ;  TEST 27
                                        ;--------------
                                    ;;*************************************************************************
005542  000004                  TST27:  SCOPE
005544  012737  000027  001102          MOV     #27,$TSTNM              ; LOAD THE NO. OF THIS TEST
005552  012737  000027  001262          MOV     #27,TSTNUM              ;# FOR TYPE OUT
005560  104417                  1$:     LRESET
005562  104416  000001  000040          WRITE   ,SEC1,ABCB      ;WRITE THE READ ONLY BIT ABCB
005570  012737  000001  001240          MOV     #SEC1,REGIST    ;SAVE REGISTER FOR TYP OUT
005576  104420  000001                  READ    ,SEC1
005602  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
005606  042737  177737  001202          BIC     #^C<ABCB>,$TMP0 ;CLEAR UNWANTED BITS
005614  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED INTO $TMP1
005620  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST IF SET
005626  001403                          BEQ     TST30           ;;
005630  104004                          ERROR   4               ;ERROR ABCB CAN BE WRITTEN
005632  000137  005560                  JMP     1$              ;LOOP ON ERROR


                                        ;****************************** TEST 30 ******************************
                                        ;*RECEIVE STATUS REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A WRITE TO THE READ ONLY BITS
                                        ;* ** ABCC **
                                        ;*AND THEN CHECKS TO MAKE SURE IT WAS NOT WRITTEN
                                        ;;*************************************************************************

                                        ;  TEST 30
                                        ;--------------
```

```
2870                                    ;;*****************************************************************
2871  005636  000004                    TST30:  SCOPE
2872  005640  012737  000030  001102            MOV     #30,$TSTNM              ; LOAD THE NO. OF THIS TEST
2873  005646  012737  000030  001262            MOV     #30,TSTNUM              ;# FOR TYPE OUT
2874  005654  104417                    1$:     LRESET
2875  005656  104416  000001  000100            WRITE   ,SEC1,ABCC      ;WRITE THE READ ONLY BIT ABCC
2876  005664  012737  000001  001240            MOV     #SEC1,REGIST    ;SAVE REGISTER FOR TYP OUT
2877  005672  104420  000001                    READ    ,SEC1
2878  005676  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
2879  005702  042737  177677  001202            BIC     #^C<ABCC>,$TMP0 ;CLEAR UNWANTED BITS
2880  005710  005037  001204                    CLR     $TMP1           ;LOAD EXPECTED INTO $TMP1
2881  005714  023737  001202  001204            CMP     $TMP0,$TMP1     ;TEST IF SET
2882  005722  001403                            BEQ     TST31           ;;
2883  005724  104004                            ERROR   4               ;ERROR ABCC CAN BE WRITTEN
2884  005726  000137  005654                    JMP     1$              ;LOOP ON ERROR
2885
2886
2887                                    ;****************************** TEST 31 ******************************
2888                                    ;*RECEIVE STATUS REGISTER (SECONDARY) BIT TEST
2889                                    ;*THIS TEST DOES A WRITE TO THE READ ONLY BITS
2890                                    ;* ** ERRCK **
2891                                    ;*AND THEN CHECKS TO MAKE SURE IT WAS NOT WRITTEN
2892                                    ;;*************************************************************************
2893
2894                                    ;  TEST 31
2895                                    ;-------------
2896                                    ;;*****************************************************************
2897  005732  000004                    TST31:  SCOPE
2898  005734  012737  000031  001102            MOV     #31,$TSTNM              ; LOAD THE NO. OF THIS TEST
2899  005742  012737  000031  001262            MOV     #31,TSTNUM              ;# FOR TYPE OUT
2900  005750  104417                    1$:     LRESET
2901  005752  104416  000001  000200            WRITE   ,SEC1,ERRCK     ;WRITE THE READ ONLY BIT ERRCK
2902  005760  012737  000001  001240            MOV     #SEC1,REGIST    ;SAVE REGISTER FOR TYP OUT
2903  005766  104420  000001                    READ    ,SEC1
2904  005772  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
2905  005776  042737  177577  001202            BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
2906  006004  005037  001204                    CLR     $TMP1           ;LOAD EXPECTED INTO $TMP1
2907  006010  023737  001202  001204            CMP     $TMP0,$TMP1     ;TEST IF SET
2908  006016  001403                            BEQ     TST32           ;;
2909  006020  104004                            ERROR   4               ;ERROR ERRCK CAN BE WRITTEN
2910  006022  000137  005750                    JMP     1$              ;LOOP ON ERROR
```

```
                                .SBTTL  TRANSMIT DATA BUFFER REGISTER FLOATING ONE'S TEST


                                ;****************************** TEST 32 ******************************
                                ;*TRANSMIT DATA REGISTER (SECONDARY)
                                ;*INCREMENT ONE'S TEST
                                ;:*************************************************************************

                                ;  TEST 32
                                ;-------------
                                ;;*************************************************************************
006026  000004                  TST32:  SCOPE
006030  012737  000032  001102          MOV     #32,$TSTNM              ; LOAD THE NO. OF THIS TEST
006036  012737  000032  001262          MOV     #32,TSTNUM              ;# FOR TYPE OUT
006044  012737  000002  006070          MOV     #BIT1,65$               ;START WITH BIT 0
006052  012737  000002  001240          MOV     #SEC2,REGIST    ;SAVE REGISTER FOR TYP OUT
006060  000004                          SCOPE
006062  104414                  1$:     MSTCLR
006064  104416  000002          64$:    WRITE   ,SEC2
006070  000000                  65$:    0                               ;DATA
006072  104420  000002                  READ    ,SEC2                   ;READ THE DATA JUST WRITTEN
006076  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
006102  013737  006070  001204          MOV     65$,$TMP1               ;LOAD EXPECTED
006110  123737  001202  001204          CMPB    $TMP0,$TMP1             ;TEST IF CORRECT
006116  001403                          BEQ     66$                     ;YES GO TO 66$
006120  104012                          ERROR   10.             ;SECONDARY REGISTER INCREMENT ONE'S FAILURE
006122  000137  006062                  JMP     1$              ;LOOP ON ERROR
006126  005237  006070          66$:    INC     65$             ;LOAD NEXT DATA
006132  032737  000400  006070          BIT     #BIT8,65$       ;TEST IF ALL DONE
006140  001751                          BEQ     64$             ;NO GO TO 64$
006142  104417                  2$:     LRESET
006144  104420  000002                  READ    ,SEC2           ;READ AND TEST IF CLEARED
006150  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
006154  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
006160  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST IF ALL CLEAR
006166  001403                          BEQ     TST33           ;;
006170  104013                          ERROR   11.             ;DID NOT CLEAR AFTER RESET
006172  000137  006142                  JMP     2$              ;LOOP ON ERROR
```

```
                                .SBTTL  TRANSMIT STATUS REGISTER (SECONDARY) READ WRITE TEST


                                        ;****************************** TEST 33 ******************************
                                        ;*TRANSMIT STATUS REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A:
                                        ;*                      1. WRITE AND A READ VERIFY
                                        ;*                      2. CLEAR AND A READ VERIFY
                                        ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
                                        ;*                        OF BIT TSOM
                                        ;:*************************************************************************

                                        ;  TEST 33
                                        ;-------------
                                ;;*************************************************************************
006176  000004                  TST33:  SCOPE
006200  012737  000033  001102          MOV     #33,$TSTNM              ; LOAD THE NO. OF THIS TEST
006206  012737  000033  001262          MOV     #33,TSTNUM              ;# FOR TYPE OUT
006214  104417                  4$:     LRESET
006216  104416  000003  000001          WRITE   ,SEC3,TSOM      ;SET BIT TSOM
006224  104420  000003                  READ    ,SEC3           ;READ VERIFY
006230  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
006234  042737  177776  001202          BIC     #^C<TSOM>,$TMP0 ;CLEAR UNWANTED
006242  012737  000001  001204          MOV     #TSOM,$TMP1     ;LOAD EXPECTED
006250  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
006256  001403                          BEQ     1$
006260  104005                          ERROR   5               ;TRANSMIT STATUS REGISTER TSOM WRITE FAILURE
006262  000137  006214                  JMP     4$              ;LOOP ON ERROR
006266                          1$:
006266  104416  000003  000000          WRITE   ,SEC3,0         ;RESET BIT TSOM
006274  104420  000003                  READ    ,SEC3           ;READ VERIFY
006300  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
006304  042737  177776  001202          BIC     #^C<TSOM>,$TMP0 ;CLEAR UNWANTED BITS
006312  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
006316  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
006324  001403                          BEQ     2$              ;YES SKIP ERROR
006326  104005                          ERROR   5               ;TRANSMIT STATUS REGISTER TSOM FAILED TO CLEAR
006330  000137  006214                  JMP     4$              ;LOOP ON ERROR
006334                          2$:
006334  104416  000003  000001          WRITE   ,SEC3,TSOM      ;WRITE TSOM
006342  104417                  5$:     LRESET
006344  104420  000003                  READ    ,SEC3           ;READ VERIFY
006350  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
006354  042737  177776  001202          BIC     #^C<TSOM>,$TMP0 ;CLEAR UNWANTED BITS
006362  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
006366  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
006374  001403                          BEQ     TST34           ;;
006376  104010                          ERROR   8.              ;TRANSMIT STATUS REGISTER TSOM FAILED TO CLEAR AFTER RES
006400  000137  006342                  JMP     5$              ;LOOP ONERROR


                                        ;****************************** TEST 34 ******************************
                                        ;*TRANSMIT STATUS REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A:
                                        ;*                      1. WRITE AND A READ VERIFY
                                        ;*                      2. CLEAR AND A READ VERIFY
```

```
                                        ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
                                        ;*                        OF BIT TEOM
                                        ;:*************************************************************************

                                        ;  TEST 34
                                        ;-------------
                                    ;;*************************************************************************
006404  000004                  TST34:  SCOPE
006406  012737  000034  001102          MOV     #34,$TSTNM              ; LOAD THE NO. OF THIS TEST
006414  012737  000034  001262          MOV     #34,TSTNUM              ;# FOR TYPE OUT
006422  104417                  4$:     LRESET
006424  104416  000003  000002          WRITE   ,SEC3,TEOM      ;SET BIT TEOM
006432  104420  000003                  READ    ,SEC3           ;READ VERIFY
006436  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
006442  042737  177775  001202          BIC     #^C<TEOM>,$TMP0 ;CLEAR UNWANTED
006450  012737  000002  001204          MOV     #TEOM,$TMP1     ;LOAD EXPECTED
006456  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
006464  001403                          BEQ     1$
006466  104005                          ERROR   5               ;TRANSMIT STATUS REGISTER TEOM WRITE FAILURE
006470  000137  006422                  JMP     4$              ;LOOP ON ERROR
006474                          1$:
006474  104416  000003  000000          WRITE   ,SEC3,0         ;RESET BIT TEOM
006502  104420  000003                  READ    ,SEC3           ;READ VERIFY
006506  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
006512  042737  177775  001202          BIC     #^C<TEOM>,$TMP0 ;CLEAR UNWANTED BITS
006520  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
006524  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
006532  001403                          BEQ     2$              ;YES SKIP ERROR
006534  104005                          ERROR   5               ;TRANSMIT STATUS REGISTER TEOM FAILED TO CLEAR
006536  000137  006422                  JMP     4$              ;LOOP ON ERROR
006542                          2$:
006542  104416  000003  000002          WRITE   ,SEC3,TEOM      ;WRITE TEOM
006550  104417                  5$:     LRESET
006552  104420  000003                  READ    ,SEC3           ;READ VERIFY
006556  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
006562  042737  177775  001202          BIC     #^C<TEOM>,$TMP0 ;CLEAR UNWANTED BITS
006570  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
006574  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
006602  001403                          BEQ     TST35           ;;
006604  104010                          ERROR   8.              ;TRANSMIT STATUS REGISTER TEOM FAILED TO CLEAR AFTER RES
006606  000137  006550                  JMP     5$              ;LOOP ONERROR


                                        ;***************************** TEST 35 *****************************
                                        ;*TRANSMIT STATUS REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A:
                                        ;*                      1. WRITE AND A READ VERIFY
                                        ;*                      2. CLEAR AND A READ VERIFY
                                        ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
                                        ;*                        OF BIT TXAB
                                        ;:*************************************************************************

                                        ;  TEST 35
                                        ;-------------
                                    ;;*************************************************************************
006612  000004                  TST35:  SCOPE
```

```
3061  006614  012737  000035  001102          MOV     #35,$TSTNM              ; LOAD THE NO. OF THIS TEST
3062  006622  012737  000035  001262          MOV     #35,TSTNUM              ;# FOR TYPE OUT
3063  006630  104417                  4$:     LRESET
3064  006632  104416  000003  000004          WRITE   ,SEC3,TXAB      ;SET BIT TXAB
3065  006640  104420  000003                  READ    ,SEC3           ;READ VERIFY
3066  006644  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3067  006650  042737  177773  001202          BIC     #^C<TXAB>,$TMP0 ;CLEAR UNWANTED
3068  006656  012737  000004  001204          MOV     #TXAB,$TMP1     ;LOAD EXPECTED
3069  006664  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
3070  006672  001403                          BEQ     1$
3071  006674  104005                          ERROR   5               ;TRANSMIT STATUS REGISTER TXAB WRITE FAILURE
3072  006676  000137  006630                  JMP     4$              ;LOOP ON ERROR
3073  006702                           1$:
3074  006702  104416  000003  000000          WRITE   ,SEC3,0         ;RESET BIT TXAB
3075  006710  104420  000003                  READ    ,SEC3           ;READ VERIFY
3076  006714  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3077  006720  042737  177773  001202          BIC     #^C<TXAB>,$TMP0 ;CLEAR UNWANTED BITS
3078  006726  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3079  006732  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
3080  006740  001403                          BEQ     2$              ;YES SKIP ERROR
3081  006742  104005                          ERROR   5               ;TRANSMIT STATUS REGISTER TXAB FAILED TO CLEAR
3082  006744  000137  006630                  JMP     4$              ;LOOP ON ERROR
3083  006750                           2$:
3084  006750  104416  000003  000004          WRITE   ,SEC3,TXAB      ;WRITE TXAB
3085  006756  104417                  5$:     LRESET
3086  006760  104420  000003                  READ    ,SEC3           ;READ VERIFY
3087  006764  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3088  006770  042737  177773  001202          BIC     #^C<TXAB>,$TMP0 ;CLEAR UNWANTED BITS
3089  006776  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3090  007002  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
3091  007010  001403                          BEQ     TST36           ;;
3092  007012  104010                          ERROR   8.              ;TRANSMIT STATUS REGISTER TXAB FAILED TO CLEAR AFTER RES
3093  007014  000137  006756                  JMP     5$              ;LOOP ONERROR
3094
3095
3096                                   ;***************************** TEST 36 *****************************
3097                                   ;*TRANSMIT STATUS REGISTER (SECONDARY) BIT TEST
3098                                   ;*THIS TEST DOES A:
3099                                   ;*                      1. WRITE AND A READ VERIFY
3100                                   ;*                      2. CLEAR AND A READ VERIFY
3101                                   ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
3102                                   ;*                        OF BIT TXGA
3103                                   ;;*************************************************************************
3104
3105                                   ;  TEST 36
3106                                   ;--------------
3107                                ;;*************************************************************************
3108  007020  000004                TST36:  SCOPE
3109  007022  012737  000036  001102          MOV     #36,$TSTNM              ; LOAD THE NO. OF THIS TEST
3110  007030  012737  000036  001262          MOV     #36,TSTNUM              ;# FOR TYPE OUT
3111  007036  104417                  4$:     LRESET
3112  007040  104416  000003  000010          WRITE   ,SEC3,TXGA      ;SET BIT TXGA
3113  007046  104420  000003                  READ    ,SEC3           ;READ VERIFY
3114  007052  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3115  007056  042737  177767  001202          BIC     #^C<TXGA>,$TMP0 ;CLEAR UNWANTED
3116  007064  012737  000010  001204          MOV     #TXGA,$TMP1     ;LOAD EXPECTED
```

```
3117  007072  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
3118  007100  001403                          BEQ     1$
3119  007102  104005                          ERROR   5               ;TRANSMIT STATUS REGISTER TXGA WRITE FAILURE
3120  007104  000137  007036                  JMP     4$              ;LOOP ON ERROR
3121  007110                          1$:
3122  007110  104416  000003  000000          WRITE   ,SEC3,0         ;RESET BIT TXGA
3123  007116  104420  000003                  READ    ,SEC3           ;READ VERIFY
3124  007122  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3125  007126  042737  177767  001202          BIC     #^C<TXGA>,$TMP0 ;CLEAR UNWANTED BITS
3126  007134  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3127  007140  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
3128  007146  001403                          BEQ     2$              ;YES SKIP ERROR
3129  007150  104005                          ERROR   5               ;TRANSMIT STATUS REGISTER TXGA FAILED TO CLEAR
3130  007152  000137  007036                  JMP     4$              ;LOOP ON ERROR
3131  007156                          2$:
3132  007156  104416  000003  000010          WRITE   ,SEC3,TXGA      ;WRITE TXGA
3133  007164  104417                  5$:     LRESET
3134  007166  104420  000003                  READ    ,SEC3           ;READ VERIFY
3135  007172  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3136  007176  042737  177767  001202          BIC     #^C<TXGA>,$TMP0 ;CLEAR UNWANTED BITS
3137  007204  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3138  007210  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
3139  007216  001403                          BEQ     TST37           ;;
3140  007220  104010                          ERROR   8.              ;TRANSMIT STATUS REGISTER TXGA FAILED TO CLEAR AFTER RES
3141  007222  000137  007164                  JMP     5$              ;LOOP ONERROR
```

```
                                .SBTTL  SYNC SECONDARY ADDRESS REGISTER (SECONDARY) FLOATING ONE'S TEST

                                ;****************************** TEST 37 ******************************
                                ;*SYNC SECONDARY REGISTER (SECONDARY)
                                ;*INCREMENT ONE'S TEST
                                ;*********************************************************************

                                ;  TEST 37
                                ;-------------
                                ;;*********************************************************************
007226  000004                  TST37:  SCOPE
007230  012737  000037  001102          MOV     #37,$TSTNM              ; LOAD THE NO. OF THIS TEST
007236  012737  000037  001262          MOV     #37,TSTNUM              ;# FOR TYPE OUT
007244  012737  000002  007270          MOV     #BIT1,65$               ;START WITH BIT 0
007252  012737  000004  001240          MOV     #SEC4,REGIST    ;SAVE REGISTER FOR TYP OUT
007260  000004                          SCOPE
007262  104414                  1$:     MSTCLR
007264  104416  000004          64$:    WRITE   ,SEC4
007270  000000                  65$:    0                               ;DATA
007272  104420  000004                  READ    ,SEC4                   ;READ THE DATA JUST WRITTEN
007276  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
007302  013737  007270  001204          MOV     65$,$TMP1               ;LOAD EXPECTED
007310  123737  001202  001204          CMPB    $TMP0,$TMP1             ;TEST IF CORRECT
007316  001403                          BEQ     66$                     ;YES GO TO 66$
007320  104012                          ERROR   10.             ;SECONDARY REGISTER INCREMENT ONE'S FAILURE
007322  000137  007262                  JMP     1$              ;LOOP ON ERROR
007326  005237  007270          66$:    INC     65$             ;LOAD NEXT DATA
007332  032737  000400  007270          BIT     #BIT8,65$       ;TEST IF ALL DONE
007340  001751                          BEQ     64$             ;NO GO TO 64$
007342  104417                  2$:     LRESET
007344  104420  000004                  READ    ,SEC4           ;READ AND TEST IF CLEARED
007350  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
007354  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
007360  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST IF ALL CLEAR
007366  001403                          BEQ     TST40           ;;
007370  104013                          ERROR   11.             ;DID NOT CLEAR AFTER RESET
007372  000137  007342                  JMP     2$              ;LOOP ON ERROR
```

```
                                .SBTTL  MODE CONTROL REGISTER (SECONDARY) READ WRITE TEST

                                ;****************************** TEST 40 ******************************
                                ;*MODE CONTROL REGISTER (SECONDARY) BIT TEST
                                ;*THIS TEST DOES A:
                                ;*                          1. WRITE AND A READ VERIFY
                                ;*                          2. CLEAR AND A READ VERIFY
                                ;*                          3. WRITE AND A LINE RESET THEN A READ VERIFY
                                ;*                            OF BIT PROTEX
                                ;;*************************************************************************

                                ;  TEST 40
                                ;--------------
                                ;;*************************************************************************
007376  000004                  TST40:  SCOPE
007400  012737  000040  001102          MOV     #40,$TSTNM              ; LOAD THE NO. OF THIS TEST
007406  012737  000040  001262          MOV     #40,TSTNUM              ;# FOR TYPE OUT
007414  104417                  4$:     LRESET
007416  104416  000005  000001          WRITE   ,SEC5,PROTEX    ;WRITE BIT PROTEX
007424  104420  000005                  READ    ,SEC5           ;READ VERIFY
007430  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
007434  042737  177776  001202          BIC     #^C<PROTEX>,$TMP0       ;CLEAR UNWANTED
007442  012737  000001  001204          MOV     #PROTEX,$TMP1   ;LOAD EXPECTED
007450  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
007456  001403                          BEQ     1$
007460  104006                          ERROR   6               ;MODE CONTROL REGISTER PROTEX WRITE FAILURE
007462  000137  007414                  JMP     4$              ;LOOP ON ERROR
007466                          1$:
007466  104416  000005  000000          WRITE   ,SEC5,0         ;RESET BIT PROTEX
007474  104420  000005                  READ    ,SEC5           ;READ VERIFY
007500  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
007504  042737  177776  001202          BIC     #^C<PROTEX>,$TMP0       ;CLEAR UNWANTED BITS
007512  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
007516  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
007524  001403                          BEQ     2$              ;YES SKIP ERROR
007526  104006                          ERROR   6               ;MODE CONTROL REGISTER PROTEX FAILED TO CLEAR
007530  000137  007414                  JMP     4$              ;LOOP ON ERROR
007534                          2$:
007534  104416  000005  000001          WRITE   ,SEC5,PROTEX    ;WRITE PROTEX
007542  104417                  5$:     LRESET
007544  104420  000005                  READ    ,SEC5           ;READ VERIFY
007550  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
007554  042737  177776  001202          BIC     #^C<PROTEX>,$TMP0       ;CLEAR UNWANTED BITS
007562  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
007566  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
007574  001403                          BEQ     TST41           ;;
007576  104011                          ERROR   9.              ;MODE CONTROL REGISTER PROTEX FAILED TO CLEAR AFTER RESE
007600  000137  007542                  JMP     5$              ;LOOP ONERROR


                                ;****************************** TEST 41 ******************************
                                ;*MODE CONTROL REGISTER (SECONDARY) BIT TEST
                                ;*THIS TEST DOES A:
                                ;*                          1. WRITE AND A READ VERIFY
                                ;*                          2. CLEAR AND A READ VERIFY
```

```
                                        ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
                                        ;*                        OF BIT PROTEY
                                        ;;**********************************************************************

                                        ;  TEST 41
                                        ;--------------
                                    ;;**********************************************************************
007604  000004                    TST41:  SCOPE
007606  012737  000041  001102            MOV     #41,$TSTNM              ; LOAD THE NO. OF THIS TEST
007614  012737  000041  001262            MOV     #41,TSTNUM              ;# FOR TYPE OUT
007622  104417                    4$:     LRESET
007624  104416  000005  000002            WRITE   ,SEC5,PROTEY    ;WRITE BIT PROTEY
007632  104420  000005                    READ    ,SEC5           ;READ VERIFY
007636  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
007642  042737  177775  001202            BIC     #^C<PROTEY>,$TMP0       ;CLEAR UNWANTED
007650  012737  000002  001204            MOV     #PROTEY,$TMP1   ;LOAD EXPECTED
007656  023737  001202  001204            CMP     $TMP0,$TMP1     ;TEST FOR A SET
007664  001403                            BEQ     1$
007666  104006                            ERROR   6               ;MODE CONTROL REGISTER PROTEY WRITE FAILURE
007670  000137  007622                    JMP     4$              ;LOOP ON ERROR
007674                            1$:
007674  104416  000005  000000            WRITE   ,SEC5,0         ;RESET BIT PROTEY
007702  104420  000005                    READ    ,SEC5           ;READ VERIFY
007706  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
007712  042737  177775  001202            BIC     #^C<PROTEY>,$TMP0       ;CLEAR UNWANTED BITS
007720  005037  001204                    CLR     $TMP1           ;LOAD EXPECTED
007724  023737  001202  001204            CMP     $TMP0,$TMP1     ;TEST FOR RESET
007732  001403                            BEQ     2$              ;YES SKIP ERROR
007734  104006                            ERROR   6               ;MODE CONTROL REGISTER PROTEY FAILED TO CLEAR
007736  000137  007622                    JMP     4$              ;LOOP ON ERROR
007742                            2$:
007742  104416  000005  000002            WRITE   ,SEC5,PROTEY    ;WRITE PROTEY
007750  104417                    5$:     LRESET
007752  104420  000005                    READ    ,SEC5           ;READ VERIFY
007756  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
007762  042737  177775  001202            BIC     #^C<PROTEY>,$TMP0       ;CLEAR UNWANTED BITS
007770  005037  001204                    CLR     $TMP1           ;LOAD EXPECTED
007774  023737  001202  001204            CMP     $TMP0,$TMP1     ;TEST FOR A RESET
010002  001403                            BEQ     TST42           ;;
010004  104011                            ERROR   9.              ;MODE CONTROL REGISTER PROTEY FAILED TO CLEAR AFTER RESE
010006  000137  007750                    JMP     5$              ;LOOP ONERROR


                                        ;***************************** TEST 42 ******************************
                                        ;*MODE CONTROL REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A:
                                        ;*                      1. WRITE AND A READ VERIFY
                                        ;*                      2. CLEAR AND A READ VERIFY
                                        ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
                                        ;*                        OF BIT PROTEZ
                                        ;;**********************************************************************

                                        ;  TEST 42
                                        ;--------------
                                    ;;**********************************************************************
010012  000004                    TST42:  SCOPE
```

```
 3292  010014  012737  000042  001102          MOV     #42,$TSTNM              ; LOAD THE NO. OF THIS TEST
 3293  010022  012737  000042  001262          MOV     #42,TSTNUM              ;# FOR TYPE OUT
 3294  010030  104417                  4$:     LRESET
 3295  010032  104416  000005  000004          WRITE   ,SEC5,PROTEZ    ;WRITE BIT PROTEZ
 3296  010040  104420  000005                  READ    ,SEC5           ;READ VERIFY
 3297  010044  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
 3298  010050  042737  177773  001202          BIC     #^C<PROTEZ>,$TMP0       ;CLEAR UNWANTED
 3299  010056  012737  000004  001204          MOV     #PROTEZ,$TMP1   ;LOAD EXPECTED
 3300  010064  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
 3301  010072  001403                          BEQ     1$
 3302  010074  104006                          ERROR   6               ;MODE CONTROL REGISTER PROTEZ WRITE FAILURE
 3303  010076  000137  010030                  JMP     4$              ;LOOP ON ERROR
 3304  010102                           1$:
 3305  010102  104416  000005  000000          WRITE   ,SEC5,0         ;RESET BIT PROTEZ
 3306  010110  104420  000005                  READ    ,SEC5           ;READ VERIFY
 3307  010114  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
 3308  010120  042737  177773  001202          BIC     #^C<PROTEZ>,$TMP0       ;CLEAR UNWANTED BITS
 3309  010126  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
 3310  010132  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
 3311  010140  001403                          BEQ     2$              ;YES SKIP ERROR
 3312  010142  104006                          ERROR   6               ;MODE CONTROL REGISTER PROTEZ FAILED TO CLEAR
 3313  010144  000137  010030                  JMP     4$              ;LOOP ON ERROR
 3314  010150                           2$:
 3315  010150  104416  000005  000004          WRITE   ,SEC5,PROTEZ    ;WRITE PROTEZ
 3316  010156  104417                  5$:     LRESET
 3317  010160  104420  000005                  READ    ,SEC5           ;READ VERIFY
 3318  010164  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
 3319  010170  042737  177773  001202          BIC     #^C<PROTEZ>,$TMP0       ;CLEAR UNWANTED BITS
 3320  010176  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
 3321  010202  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
 3322  010210  001403                          BEQ     TST43           ;;
 3323  010212  104011                          ERROR   9.              ;MODE CONTROL REGISTER PROTEZ FAILED TO CLEAR AFTER RESE
 3324  010214  000137  010156                  JMP     5$              ;LOOP ONERROR
 3325
 3326
 3327                                   ;***************************** TEST 43 *****************************
 3328                                   ;*MODE CONTROL REGISTER (SECONDARY) BIT TEST
 3329                                   ;*THIS TEST DOES A:
 3330                                   ;*                      1. WRITE AND A READ VERIFY
 3331                                   ;*                      2. CLEAR AND A READ VERIFY
 3332                                   ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
 3333                                   ;*                        OF BIT NIDLE
 3334                                   ;;*************************************************************************
 3335
 3336                                   ;  TEST 43
 3337                                   ;--------------
 3338                                ;;*************************************************************************
 3339  010220  000004                TST43:  SCOPE
 3340  010222  012737  000043  001102          MOV     #43,$TSTNM              ; LOAD THE NO. OF THIS TEST
 3341  010230  012737  000043  001262          MOV     #43,TSTNUM              ;# FOR TYPE OUT
 3342  010236  104417                  4$:     LRESET
 3343  010240  104416  000005  000010          WRITE   ,SEC5,NIDLE     ;WRITE BIT NIDLE
 3344  010246  104420  000005                  READ    ,SEC5           ;READ VERIFY
 3345  010252  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
 3346  010256  042737  177767  001202          BIC     #^C<NIDLE>,$TMP0        ;CLEAR UNWANTED
 3347  010264  012737  000010  001204          MOV     #NIDLE,$TMP1    ;LOAD EXPECTED
```

```
3348  010272  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
3349  010300  001403                          BEQ     1$
3350  010302  104006                          ERROR   6               ;MODE CONTROL REGISTER NIDLE WRITE FAILURE
3351  010304  000137  010236                  JMP     4$              ;LOOP ON ERROR
3352  010310                          1$:
3353  010310  104416  000005  000000          WRITE   ,SEC5,0         ;RESET BIT NIDLE
3354  010316  104420  000005                  READ    ,SEC5           ;READ VERIFY
3355  010322  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3356  010326  042737  177767  001202          BIC     #^C<NIDLE>,$TMP0        ;CLEAR UNWANTED BITS
3357  010334  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3358  010340  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
3359  010346  001403                          BEQ     2$              ;YES SKIP ERROR
3360  010350  104006                          ERROR   6               ;MODE CONTROL REGISTER NIDLE FAILED TO CLEAR
3361  010352  000137  010236                  JMP     4$              ;LOOP ON ERROR
3362  010356                          2$:
3363  010356  104416  000005  000010          WRITE   ,SEC5,NIDLE     ;WRITE NIDLE
3364  010364  104417                  5$:     LRESET
3365  010366  104420  000005                  READ    ,SEC5           ;READ VERIFY
3366  010372  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3367  010376  042737  177767  001202          BIC     #^C<NIDLE>,$TMP0        ;CLEAR UNWANTED BITS
3368  010404  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3369  010410  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
3370  010416  001403                          BEQ     TST44           ;;
3371  010420  104011                          ERROR   9.              ;MODE CONTROL REGISTER NIDLE FAILED TO CLEAR AFTER RESET
3372  010422  000137  010364                  JMP     5$              ;LOOP ONERROR
3373
3374
3375                                          ;****************************** TEST 44 ******************************
3376                                          ;*MODE CONTROL REGISTER (SECONDARY) BIT TEST
3377                                          ;*THIS TEST DOES A:
3378                                          ;*                      1. WRITE AND A READ VERIFY
3379                                          ;*                      2. CLEAR AND A READ VERIFY
3380                                          ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
3381                                          ;*                        OF BIT SAM
3382                                          ;;*************************************************************************
3383
3384                                          ;  TEST 44
3385                                          ;--------------
3386                                  ;;*************************************************************************
3387  010426  000004                  TST44:  SCOPE
3388  010430  012737  000044  001102          MOV     #44,$TSTNM              ; LOAD THE NO. OF THIS TEST
3389  010436  012737  000044  001262          MOV     #44,TSTNUM              ;# FOR TYPE OUT
3390  010444  104417                  4$:     LRESET
3391  010446  104416  000005  000020          WRITE   ,SEC5,SAM       ;WRITE BIT SAM
3392  010454  104420  000005                  READ    ,SEC5           ;READ VERIFY
3393  010460  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3394  010464  042737  177757  001202          BIC     #^C<SAM>,$TMP0  ;CLEAR UNWANTED
3395  010472  012737  000020  001204          MOV     #SAM,$TMP1      ;LOAD EXPECTED
3396  010500  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
3397  010506  001403                          BEQ     1$
3398  010510  104006                          ERROR   6               ;MODE CONTROL REGISTER SAM WRITE FAILURE
3399  010512  000137  010444                  JMP     4$              ;LOOP ON ERROR
3400  010516                          1$:
3401  010516  104416  000005  000000          WRITE   ,SEC5,0         ;RESET BIT SAM
3402  010524  104420  000005                  READ    ,SEC5           ;READ VERIFY
3403  010530  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
```

```
3404  010534  042737  177757  001202          BIC     #^C<SAM>,$TMP0  ;CLEAR UNWANTED BITS
3405  010542  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3406  010546  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
3407  010554  001403                          BEQ     2$              ;YES SKIP ERROR
3408  010556  104006                          ERROR   6               ;MODE CONTROL REGISTER SAM FAILED TO CLEAR
3409  010560  000137  010444                  JMP     4$              ;LOOP ON ERROR
3410  010564                          2$:
3411  010564  104416  000005  000020          WRITE   ,SEC5,SAM       ;WRITE SAM
3412  010572  104417                  5$:     LRESET
3413  010574  104420  000005                  READ    ,SEC5           ;READ VERIFY
3414  010600  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3415  010604  042737  177757  001202          BIC     #^C<SAM>,$TMP0  ;CLEAR UNWANTED BITS
3416  010612  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3417  010616  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
3418  010624  001403                          BEQ     TST45           ;;
3419  010626  104011                          ERROR   9.              ;MODE CONTROL REGISTER SAM FAILED TO CLEAR AFTER RESET
3420  010630  000137  010572                  JMP     5$              ;LOOP ONERROR
3421
3422
3423                                  ;****************************** TEST 45 ******************************
3424                                  ;*MODE CONTROL REGISTER (SECONDARY) BIT TEST
3425                                  ;*THIS TEST DOES A:
3426                                  ;*                      1. WRITE AND A READ VERIFY
3427                                  ;*                      2. CLEAR AND A READ VERIFY
3428                                  ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
3429                                  ;*                         OF BIT SSL
3430                                  ;;*************************************************************************
3431
3432                                  ;  TEST 45
3433                                  ;--------------
3434                          ;;*************************************************************************
3435  010634  000004          TST45:  SCOPE
3436  010636  012737  000045  001102          MOV     #45,$TSTNM              ; LOAD THE NO. OF THIS TEST
3437  010644  012737  000045  001262          MOV     #45,TSTNUM              ;# FOR TYPE OUT
3438  010652  104417          4$:     LRESET
3439  010654  104416  000005  000040          WRITE   ,SEC5,SSL       ;WRITE BIT SSL
3440  010662  104420  000005                  READ    ,SEC5           ;READ VERIFY
3441  010666  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3442  010672  042737  177737  001202          BIC     #^C<SSL>,$TMP0  ;CLEAR UNWANTED
3443  010700  012737  000040  001204          MOV     #SSL,$TMP1      ;LOAD EXPECTED
3444  010706  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
3445  010714  001403                          BEQ     1$
3446  010716  104006                          ERROR   6               ;MODE CONTROL REGISTER SSL WRITE FAILURE
3447  010720  000137  010652                  JMP     4$              ;LOOP ON ERROR
3448  010724                          1$:
3449  010724  104416  000005  000000          WRITE   ,SEC5,0         ;RESET BIT SSL
3450  010732  104420  000005                  READ    ,SEC5           ;READ VERIFY
3451  010736  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3452  010742  042737  177737  001202          BIC     #^C<SSL>,$TMP0  ;CLEAR UNWANTED BITS
3453  010750  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3454  010754  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
3455  010762  001403                          BEQ     2$              ;YES SKIP ERROR
3456  010764  104006                          ERROR   6               ;MODE CONTROL REGISTER SSL FAILED TO CLEAR
3457  010766  000137  010652                  JMP     4$              ;LOOP ON ERROR
3458  010772                          2$:
3459  010772  104416  000005  000040          WRITE   ,SEC5,SSL       ;WRITE SSL
```

```
3460  011000  104417                  5$:     LRESET
3461  011002  104420  000005                  READ    ,SEC5                   ;READ VERIFY
3462  011006  012637  001202                  MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
3463  011012  042737  177737  001202          BIC     #^C<SSL>,$TMP0          ;CLEAR UNWANTED BITS
3464  011020  005037  001204                  CLR     $TMP1                   ;LOAD EXPECTED
3465  011024  023737  001202  001204          CMP     $TMP0,$TMP1             ;TEST FOR A RESET
3466  011032  001403                          BEQ     TST46                   ;;
3467  011034  104011                          ERROR   9.                      ;MODE CONTROL REGISTER SSL FAILED TO CLEAR AFTER RESET
3468  011036  000137  011000                  JMP     5$                      ;LOOP ONERROR
3469
3470
3471                                          ;***************************** TEST 46 *****************************
3472                                          ;*MODE CONTROL REGISTER (SECONDARY) BIT TEST
3473                                          ;*THIS TEST DOES A:
3474                                          ;*                      1. WRITE AND A READ VERIFY
3475                                          ;*                      2. CLEAR AND A READ VERIFY
3476                                          ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
3477                                          ;*                        OF BIT PROTO
3478                                          ;;*************************************************************************
3479
3480                                          ;  TEST 46
3481                                          ;-------------
3482                                  ;;**************************************************************************
3483  011042  000004                  TST46:  SCOPE
3484  011044  012737  000046  001102          MOV     #46,$TSTNM                      ; LOAD THE NO. OF THIS TEST
3485  011052  012737  000046  001262          MOV     #46,TSTNUM                      ;# FOR TYPE OUT
3486  011060  104417                  4$:     LRESET
3487  011062  104416  000005  000100          WRITE   ,SEC5,PROTO             ;WRITE BIT PROTO
3488  011070  104420  000005                  READ    ,SEC5                   ;READ VERIFY
3489  011074  012637  001202                  MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
3490  011100  042737  177677  001202          BIC     #^C<PROTO>,$TMP0                ;CLEAR UNWANTED
3491  011106  012737  000100  001204          MOV     #PROTO,$TMP1            ;LOAD EXPECTED
3492  011114  023737  001202  001204          CMP     $TMP0,$TMP1             ;TEST FOR A SET
3493  011122  001403                          BEQ     1$
3494  011124  104006                          ERROR   6                       ;MODE CONTROL REGISTER PROTO WRITE FAILURE
3495  011126  000137  011060                  JMP     4$                      ;LOOP ON ERROR
3496  011132                          1$:
3497  011132  104416  000005  000000          WRITE   ,SEC5,0                 ;RESET BIT PROTO
3498  011140  104420  000005                  READ    ,SEC5                   ;READ VERIFY
3499  011144  012637  001202                  MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
3500  011150  042737  177677  001202          BIC     #^C<PROTO>,$TMP0                ;CLEAR UNWANTED BITS
3501  011156  005037  001204                  CLR     $TMP1                   ;LOAD EXPECTED
3502  011162  023737  001202  001204          CMP     $TMP0,$TMP1             ;TEST FOR RESET
3503  011170  001403                          BEQ     2$                      ;YES SKIP ERROR
3504  011172  104006                          ERROR   6                       ;MODE CONTROL REGISTER PROTO FAILED TO CLEAR
3505  011174  000137  011060                  JMP     4$                      ;LOOP ON ERROR
3506  011200                          2$:
3507  011200  104416  000005  000100          WRITE   ,SEC5,PROTO             ;WRITE PROTO
3508  011206  104417                  5$:     LRESET
3509  011210  104420  000005                  READ    ,SEC5                   ;READ VERIFY
3510  011214  012637  001202                  MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
3511  011220  042737  177677  001202          BIC     #^C<PROTO>,$TMP0                ;CLEAR UNWANTED BITS
3512  011226  005037  001204                  CLR     $TMP1                   ;LOAD EXPECTED
3513  011232  023737  001202  001204          CMP     $TMP0,$TMP1             ;TEST FOR A RESET
3514  011240  001403                          BEQ     TST47                   ;;
3515  011242  104011                          ERROR   9.                      ;MODE CONTROL REGISTER PROTO FAILED TO CLEAR AFTER RESET
```

```
 3516  011244  000137  011206                    JMP     5$              ;LOOP ONERROR
```

```
                                .SBTTL  CHARACTER LENGTH REGISTER (SECONDARY) READ WRITE TEST


                                        ;****************************** TEST 47 ******************************
                                        ;*CHARACTER LENGTH REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A:
                                        ;*                          1. WRITE AND A READ VERIFY
                                        ;*                          2. CLEAR AND A READ VERIFY
                                        ;*                          3. WRITE AND A LINE RESET THEN A READ VERIFY
                                        ;*                            OF BIT RDATA1
                                        ;;*************************************************************************

                                        ;  TEST 47
                                        ;-------------
                                ;;*************************************************************************
011250  000004                  TST47:  SCOPE
011252  012737  000047  001102          MOV     #47,$TSTNM              ; LOAD THE NO. OF THIS TEST
011260  012737  000047  001262          MOV     #47,TSTNUM              ;# FOR TYPE OUT
011266  104417                  4$:     LRESET
011270  104416  000007  000001          WRITE   ,SEC7,RDATA1    ;WRITE BIT RDATA1
011276  104420  000007                  READ    ,SEC7           ;READ VERIFY
011302  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
011306  042737  177776  001202          BIC     #^C<RDATA1>,$TMP0       ;CLEAR UNWANTED
011314  012737  000001  001204          MOV     #RDATA1,$TMP1   ;LOAD EXPECTED
011322  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
011330  001403                          BEQ     1$
011332  104007                          ERROR   7               ;CHARACTER LENGTH REGISTER RDATA1 WRITE FAILURE
011334  000137  011266                  JMP     4$              ;LOOP ON ERROR
011340                  1$:
011340  104416  000007  000000          WRITE   ,SEC7,0         ;RESET BIT RDATA1
011346  104420  000007                  READ    ,SEC7           ;READ VERIFY
011352  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
011356  042737  177776  001202          BIC     #^C<RDATA1>,$TMP0       ;CLEAR UNWANTED BITS
011364  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
011370  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
011376  001403                          BEQ     2$              ;YES SKIP ERROR
011400  104007                          ERROR   7               ;CHARACTER LENGTH REGISTER RDATA1 FAILED TO CLEAR
011402  000137  011266                  JMP     4$              ;LOOP ON ERROR
011406                  2$:
011406  104416  000007  000001          WRITE   ,SEC7,RDATA1    ;WRITE RDATA1
011414  104417                  5$:     LRESET
011416  104420  000007                  READ    ,SEC7           ;READ VERIFY
011422  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
011426  042737  177776  001202          BIC     #^C<RDATA1>,$TMP0       ;CLEAR UNWANTED BITS
011434  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
011440  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
011446  001403                          BEQ     TST50           ;;
011450  104025                          ERROR   21.             ;CHARACTER LENGTH REGISTER RDATA1 FAILED TO CLEAR AFTER
011452  000137  011414                  JMP     5$              ;LOOP ONERROR


                                        ;****************************** TEST 50 ******************************
                                        ;*CHARACTER LENGTH REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A:
                                        ;*                          1. WRITE AND A READ VERIFY
                                        ;*                          2. CLEAR AND A READ VERIFY
```

```
                                        ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
                                        ;*                        OF BIT RDATA2
                                        ;;*************************************************************************

                                        ;  TEST 50
                                        ;--------------
                                    ;;*************************************************************************
011456  000004                  TST50:  SCOPE
011460  012737  000050  001102          MOV     #50,$TSTNM              ; LOAD THE NO. OF THIS TEST
011466  012737  000050  001262          MOV     #50,TSTNUM              ;# FOR TYPE OUT
011474  104417                  4$:     LRESET
011476  104416  000007  000002          WRITE   ,SEC7,RDATA2    ;WRITE BIT RDATA2
011504  104420  000007                  READ    ,SEC7           ;READ VERIFY
011510  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
011514  042737  177775  001202          BIC     #^C<RDATA2>,$TMP0       ;CLEAR UNWANTED
011522  012737  000002  001204          MOV     #RDATA2,$TMP1   ;LOAD EXPECTED
011530  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
011536  001403                          BEQ     1$
011540  104007                          ERROR   7               ;CHARACTER LENGTH REGISTER RDATA2 WRITE FAILURE
011542  000137  011474                  JMP     4$              ;LOOP ON ERROR
011546                          1$:
011546  104416  000007  000000          WRITE   ,SEC7,0         ;RESET BIT RDATA2
011554  104420  000007                  READ    ,SEC7           ;READ VERIFY
011560  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
011564  042737  177775  001202          BIC     #^C<RDATA2>,$TMP0       ;CLEAR UNWANTED BITS
011572  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
011576  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
011604  001403                          BEQ     2$              ;YES SKIP ERROR
011606  104007                          ERROR   7               ;CHARACTER LENGTH REGISTER RDATA2 FAILED TO CLEAR
011610  000137  011474                  JMP     4$              ;LOOP ON ERROR
011614                          2$:
011614  104416  000007  000002          WRITE   ,SEC7,RDATA2    ;WRITE RDATA2
011622  104417                  5$:     LRESET
011624  104420  000007                  READ    ,SEC7           ;READ VERIFY
011630  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
011634  042737  177775  001202          BIC     #^C<RDATA2>,$TMP0       ;CLEAR UNWANTED BITS
011642  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
011646  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
011654  001403                          BEQ     TST51.          ;;
011656  104025                          ERROR   21.             ;CHARACTER LENGTH REGISTER RDATA2 FAILED TO CLEAR AFTER
011660  000137  011622                  JMP     5$              ;LOOP ONERROR


                                        ;***************************** TEST 51 ****************************
                                        ;*CHARACTER LENGTH REGISTER (SECONDARY) BIT TEST
                                        ;*THIS TEST DOES A:
                                        ;*                      1. WRITE AND A READ VERIFY
                                        ;*                      2. CLEAR AND A READ VERIFY
                                        ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
                                        ;*                        OF BIT RDATA3
                                        ;;*************************************************************************

                                        ;  TEST 51
                                        ;--------------
                                    ;;*************************************************************************
011664  000004                  TST51:  SCOPE
```

```
3629  011666  012737  000051  001102          MOV     #51,$TSTNM              ; LOAD THE NO. OF THIS TEST
3630  011674  012737  000051  001262          MOV     #51,TSTNUM              ;# FOR TYPE OUT
3631  011702  104417                  4$:     LRESET
3632  011704  104416  000007  000004          WRITE   ,SEC7,RDATA3    ;WRITE BIT RDATA3
3633  011712  104420  000007                  READ    ,SEC7           ;READ VERIFY
3634  011716  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3635  011722  042737  177773  001202          BIC     #^C<RDATA3>,$TMP0       ;CLEAR UNWANTED
3636  011730  012737  000004  001204          MOV     #RDATA3,$TMP1   ;LOAD EXPECTED
3637  011736  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
3638  011744  001403                          BEQ     1$
3639  011746  104007                          ERROR   7               ;CHARACTER LENGTH REGISTER RDATA3 WRITE FAILURE
3640  011750  000137  011702                  JMP     4$              ;LOOP ON ERROR
3641  011754                           1$:
3642  011754  104416  000007  000000          WRITE   ,SEC7,0         ;RESET BIT RDATA3
3643  011762  104420  000007                  READ    ,SEC7           ;READ VERIFY
3644  011766  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3645  011772  042737  177773  001202          BIC     #^C<RDATA3>,$TMP0       ;CLEAR UNWANTED BITS
3646  012000  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3647  012004  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
3648  012012  001403                          BEQ     2$              ;YES SKIP ERROR
3649  012014  104007                          ERROR   7               ;CHARACTER LENGTH REGISTER RDATA3 FAILED TO CLEAR
3650  012016  000137  011702                  JMP     4$              ;LOOP ON ERROR
3651  012022                           2$:
3652  012022  104416  000007  000004          WRITE   ,SEC7,RDATA3    ;WRITE RDATA3
3653  012030  104417                  5$:     LRESET
3654  012032  104420  000007                  READ    ,SEC7           ;READ VERIFY
3655  012036  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3656  012042  042737  177773  001202          BIC     #^C<RDATA3>,$TMP0       ;CLEAR UNWANTED BITS
3657  012050  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3658  012054  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
3659  012062  001403                          BEQ     TST52           ;;
3660  012064  104025                          ERROR   21.             ;CHARACTER LENGTH REGISTER RDATA3 FAILED TO CLEAR AFTER
3661  012066  000137  012030                  JMP     5$              ;LOOP ONERROR
3662
3663
3664                                          ;****************************** TEST 52 ******************************
3665                                          ;*CHARACTER LENGTH REGISTER (SECONDARY) BIT TEST
3666                                          ;*THIS TEST DOES A:
3667                                          ;*                      1. WRITE AND A READ VERIFY
3668                                          ;*                      2. CLEAR AND A READ VERIFY
3669                                          ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
3670                                          ;*                        OF BIT TDATA1
3671                                          ;;*************************************************************************
3672
3673                                          ;  TEST 52
3674                                          ;-------------
3675                                   ;;*************************************************************************
3676  012072  000004                  TST52:  SCOPE
3677  012074  012737  000052  001102          MOV     #52,$TSTNM              ; LOAD THE NO. OF THIS TEST
3678  012102  012737  000052  001262          MOV     #52,TSTNUM              ;# FOR TYPE OUT
3679  012110  104417                  4$:     LRESET
3680  012112  104416  000007  000040          WRITE   ,SEC7,TDATA1    ;WRITE BIT TDATA1
3681  012120  104420  000007                  READ    ,SEC7           ;READ VERIFY
3682  012124  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3683  012130  042737  177737  001202          BIC     #^C<TDATA1>,$TMP0       ;CLEAR UNWANTED
3684  012136  012737  000040  001204          MOV     #TDATA1,$TMP1   ;LOAD EXPECTED
```

```
3685  012144  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
3686  012152  001403                          BEQ     1$
3687  012154  104007                          ERROR   7               ;CHARACTER LENGTH REGISTER TDATA1 WRITE FAILURE
3688  012156  000137  012110                  JMP     4$              ;LOOP ON ERROR
3689  012162                          1$:
3690  012162  104416  000007  000000          WRITE   ,SEC7,0         ;RESET BIT TDATA1
3691  012170  104420  000007                  READ    ,SEC7           ;READ VERIFY
3692  012174  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3693  012200  042737  177737  001202          BIC     #^C<TDATA1>,$TMP0       ;CLEAR UNWANTED BITS
3694  012206  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3695  012212  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR RESET
3696  012220  001403                          BEQ     2$              ;YES SKIP ERROR
3697  012222  104007                          ERROR   7               ;CHARACTER LENGTH REGISTER TDATA1 FAILED TO CLEAR
3698  012224  000137  012110                  JMP     4$              ;LOOP ON ERROR
3699  012230                          2$:
3700  012230  104416  000007  000040          WRITE   ,SEC7,TDATA1    ;WRITE TDATA1
3701  012236  104417                  5$:     LRESET
3702  012240  104420  000007                  READ    ,SEC7           ;READ VERIFY
3703  012244  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3704  012250  042737  177737  001202          BIC     #^C<TDATA1>,$TMP0       ;CLEAR UNWANTED BITS
3705  012256  005037  001204                  CLR     $TMP1           ;LOAD EXPECTED
3706  012262  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A RESET
3707  012270  001403                          BEQ     TST53           ;;
3708  012272  104025                          ERROR   21.             ;CHARACTER LENGTH REGISTER TDATA1 FAILED TO CLEAR AFTER
3709  012274  000137  012236                  JMP     5$              ;LOOP ONERROR
3710
3711
3712                                  ;****************************** TEST 53 ******************************
3713                                  ;*CHARACTER LENGTH REGISTER (SECONDARY) BIT TEST
3714                                  ;*THIS TEST DOES A:
3715                                  ;*                      1. WRITE AND A READ VERIFY
3716                                  ;*                      2. CLEAR AND A READ VERIFY
3717                                  ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
3718                                  ;*                        OF BIT TDATA2
3719                                  ;;*************************************************************************
3720
3721                                  ;  TEST 53
3722                                  ;-------------
3723                          ;;*************************************************************************
3724  012300  000004                  TST53:  SCOPE
3725  012302  012737  000053  001102          MOV     #53,$TSTNM              ; LOAD THE NO. OF THIS TEST
3726  012310  012737  000053  001262          MOV     #53,TSTNUM              ;# FOR TYPE OUT
3727  012316  104417                  4$:     LRESET
3728  012320  104416  000007  000100          WRITE   ,SEC7,TDATA2    ;WRITE BIT TDATA2
3729  012326  104420  000007                  READ    ,SEC7           ;READ VERIFY
3730  012332  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3731  012336  042737  177677  001202          BIC     #^C<TDATA2>,$TMP0       ;CLEAR UNWANTED
3732  012344  012737  000100  001204          MOV     #TDATA2,$TMP1   ;LOAD EXPECTED
3733  012352  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR A SET
3734  012360  001403                          BEQ     1$
3735  012362  104007                          ERROR   7               ;CHARACTER LENGTH REGISTER TDATA2 WRITE FAILURE
3736  012364  000137  012316                  JMP     4$              ;LOOP ON ERROR
3737  012370                          1$:
3738  012370  104416  000007  000000          WRITE   ,SEC7,0         ;RESET BIT TDATA2
3739  012376  104420  000007                  READ    ,SEC7           ;READ VERIFY
3740  012402  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
```

```
3741  012406  042737  177677  001202          BIC     #^C<TDATA2>,$TMP0       ;CLEAR UNWANTED BITS
3742  012414  005037  001204                  CLR     $TMP1                   ;LOAD EXPECTED
3743  012420  023737  001202  001204          CMP     $TMP0,$TMP1             ;TEST FOR RESET
3744  012426  001403                          BEQ     2$                      ;YES SKIP ERROR
3745  012430  104007                          ERROR   7                       ;CHARACTER LENGTH REGISTER TDATA2 FAILED TO CLEAR
3746  012432  000137  012316                  JMP     4$                      ;LOOP ON ERROR
3747  012436                          2$:
3748  012436  104416  000007  000100          WRITE   ,SEC7,TDATA2            ;WRITE TDATA2
3749  012444  104417                  5$:     LRESET
3750  012446  104420  000007                  READ    ,SEC7                   ;READ VERIFY
3751  012452  012637  001202                  MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
3752  012456  042737  177677  001202          BIC     #^C<TDATA2>,$TMP0       ;CLEAR UNWANTED BITS
3753  012464  005037  001204                  CLR     $TMP1                   ;LOAD EXPECTED
3754  012470  023737  001202  001204          CMP     $TMP0,$TMP1             ;TEST FOR A RESET
3755  012476  001403                          BEQ     TST54                   ;;
3756  012500  104025                          ERROR   21.                     ;CHARACTER LENGTH REGISTER TDATA2 FAILED TO CLEAR AFTER
3757  012502  000137  012444                  JMP     5$                      ;LOOP ONERROR
3758
3759
3760                                  ;****************************** TEST 54 ******************************
3761                                  ;*CHARACTER LENGTH REGISTER (SECONDARY) BIT TEST
3762                                  ;*THIS TEST DOES A:
3763                                  ;*                      1. WRITE AND A READ VERIFY
3764                                  ;*                      2. CLEAR AND A READ VERIFY
3765                                  ;*                      3. WRITE AND A LINE RESET THEN A READ VERIFY
3766                                  ;*                        OF BIT TDATA3
3767                                  ;;*************************************************************************
3768
3769                                  ;  TEST 54
3770                                  ;--------------
3771                          ;;*************************************************************************
3772  012506  000004          TST54:  SCOPE
3773  012510  012737  000054  001102          MOV     #54,$TSTNM              ; LOAD THE NO. OF THIS TEST
3774  012516  012737  000054  001262          MOV     #54,TSTNUM              ;# FOR TYPE OUT
3775  012524  104417                  4$:     LRESET
3776  012526  104416  000007  000200          WRITE   ,SEC7,TDATA3            ;WRITE BIT TDATA3
3777  012534  104420  000007                  READ    ,SEC7                   ;READ VERIFY
3778  012540  012637  001202                  MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
3779  012544  042737  177577  001202          BIC     #^C<TDATA3>,$TMP0       ;CLEAR UNWANTED
3780  012552  012737  000200  001204          MOV     #TDATA3,$TMP1           ;LOAD EXPECTED
3781  012560  023737  001202  001204          CMP     $TMP0,$TMP1             ;TEST FOR A SET
3782  012566  001403                          BEQ     1$
3783  012570  104007                          ERROR   7                       ;CHARACTER LENGTH REGISTER TDATA3 WRITE FAILURE
3784  012572  000137  012524                  JMP     4$                      ;LOOP ON ERROR
3785  012576                          1$:
3786  012576  104416  000007  000000          WRITE   ,SEC7,0                 ;RESET BIT TDATA3
3787  012604  104420  000007                  READ    ,SEC7                   ;READ VERIFY
3788  012610  012637  001202                  MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
3789  012614  042737  177577  001202          BIC     #^C<TDATA3>,$TMP0       ;CLEAR UNWANTED BITS
3790  012622  005037  001204                  CLR     $TMP1                   ;LOAD EXPECTED
3791  012626  023737  001202  001204          CMP     $TMP0,$TMP1             ;TEST FOR RESET
3792  012634  001403                          BEQ     2$                      ;YES SKIP ERROR
3793  012636  104007                          ERROR   7                       ;CHARACTER LENGTH REGISTER TDATA3 FAILED TO CLEAR
3794  012640  000137  012524                  JMP     4$                      ;LOOP ON ERROR
3795  012644                          2$:
3796  012644  104416  000007  000200          WRITE   ,SEC7,TDATA3            ;WRITE TDATA3
```

```
 3797  012652  104417                  5$:     LRESET
 3798  012654  104420  000007                  READ    ,SEC7                   ;READ VERIFY
 3799  012660  012637  001202                  MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
 3800  012664  042737  177577  001202          BIC     #^C<TDATA3>,$TMP0       ;CLEAR UNWANTED BITS
 3801  012672  005037  001204                  CLR     $TMP1                   ;LOAD EXPECTED
 3802  012676  023737  001202  001204          CMP     $TMP0,$TMP1             ;TEST FOR A RESET
 3803  012704  001403                          BEQ     TST55                   ;;
 3804  012706  104025                          ERROR   21.                     ;CHARACTER LENGTH REGISTER TDATA3 FAILED TO CLEAR AFTER
 3805  012710  000137  012652                  JMP     5$                      ;LOOP ONERROR
```

```
                                .SBTTL  TRANSMIT AND RECEIVE INITIALIZATION TEST

                        ;****************************** TEST 55 ******************************
                        ;*TRANSMIT AND RECEIVE INITIALIZATION TEST.
                        ;*
                        ;*THIS TEST WILL SYNC THE SEND RECEIVE LOGIC.
                        ;*PROTOCOL = CCP
                        ;*THIS TEST WILL PERFORM THE FOLLOWING SEQUENCE:
                        ;*
                        ;*
                        ;*      1. CLEAR THE SYNC REGISTER
                        ;*      2. ISSUE A MASTER CLEAR
                        ;*      3. SET PROTOCOL = CCP
                        ;*      4. LOAD SYNC CHARACTER INTO SYNC SECONDARY
                        ;*         REGISTER
                        ;*      5. SET TENA,RENA,MB,MC
                        ;*      6. SET TSOM
                        ;*      7. TEST FOR TBMT RESET
                        ;*      8. STEP THE CLOCK 2 (10)
                        ;*      9. TEST FOR TBMT SET AND TAC SET
                        ;*      10. RESET TSOM, TEST THAT TBMT RESET, STEP
                        ;*          THE CLOCK 6 (10) AND LOCK FOR THE FIRST
                        ;*          S/F ON RECEPTION OF THE FIRST SYNC
                        ;*      11. STEP THE CLOCK 2 (10) AND LOOK FOR
                        ;*          S/F RESET, THEN STEP THE CLOCK 6 (10) FOR
                        ;*          FOR A LOOK AT S/F TO BE SET ON RECEPTION
                        ;*          OF THE SECOND SYNC.
                        ;*      12. LOAD STX INTO TDB
                        ;*      13. STEP THE CLOCK 2 (10)
                        ;*      14. LOOK FOR S/F TO BE RESET
                        ;*      15. STEP THE CLOCK 6 (10)
                        ;*      16. LOAD A DATA CHARACTER INTO TDB
                        ;*          AND TRANSMIT
                        ;*      17. LOAD A DATA CHARACTER INTO TDB
                        ;*          AND TRANSMIT
                        ;*      18. LOAD A DATA CHARACTER INTO TDB
                        ;*          AND TRANSMIT
                        ;*      19. TEST FOR THE FOLLOWING CONDITIONS:
                        ;*              A. RAC TO BE SET
                        ;*              B. RDA TO BE SET
                        ;*      20. READ THE DATA REGISTER TO HOLD THE STX CHARACTER
                        ;:*****************************************************************

                        ;  TEST 55
                        ;--------------
                        ;;*****************************************************************
012714  000004          TST55:  SCOPE
012716  012737  000055  001102          MOV     #55,$TSTNM              ; LOAD THE NO. OF THIS TEST
012724  012737  000055  001262          MOV     #55,TSTNUM              ;# FOR TYPE OUT
012732  012737  012740  001106          MOV     #1$,$LPADR      ;SET RETURN ADDRESS TO 1$
012740  104414          1$:     MSTCLR                          ;ISSUE A MASTER CLEAR
012742  104416  000005  000100          WRITE   ,SEC5,CCP               ;SET MODE CONTROL REGISTER
                                                                ;       PROTOCOL = CCP
012750  104416  000004  000026          WRITE   ,SEC4,SYN               ;LOAD SYNC SECONDARY ADDRESS REGISTER
012756  104416  000014  000321          WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
```

```
                                                        ;       TENA
                                                        ;       RENA
                                                        ;       MB AND MC
012764  104416  000003  000001        WRITE   ,SEC3,TSOM      ;SET TSOM
012772  104420  000013                READ    ,PRI13          ;READ REGISTER
012776  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
013002  042737  177757  001202        BIC     #^C<TBMT>,$TMP0 ;CLEAR UNWANTED BITS
013010  032737  000020  001202        BIT     #TBMT,$TMP0             ;TEST TBMT
013016  001414                        BEQ     3$              ;RESET SKIP ERROR
013020  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
013026  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
013034  042737  000020  001206        BIC     #TBMT,$TMP2             ;LOAD EXPECTED
013042  104014                        ERROR   12.             ;TBMT NOT RESET AFTER TSOM AND TENA
013044  000137  012740                JMP     1$              ;LOOP ON ERROR
                                                        ;ENABLED
013050  104421  000004        3$:     STEP    ,2*2            ;STEP THE CLOCK 2 TICKS
013054  104420  000013                READ    ,PRI13          ;READ REGISTER
013060  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
013064  042737  177757  001202        BIC     #^C<TBMT>,$TMP0 ;CLEAR UNWANTED BITS
013072  032737  000020  001202        BIT     #TBMT,$TMP0             ;TEST TBMT
013100  001014                        BNE     4$              ;SET SKIP ERROR
013102  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
013110  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
013116  052737  000020  001206        BIS     #TBMT,$TMP2             ;LOAD EXPECTED
013124  104015                        ERROR   13.             ;TBMT FAILED TO SET WITH TSOM SET
                                                        ;AND CLOCK STEPPED 2 TIMES
013126  000137  012740                JMP     1$              ;LOOP ON ERROR
013132                          4$:
013132  104416  000002  000002        WRITE   ,SEC2,STX       ;LOAD START OF TEXT CHARACTER
013140  104421  000016                STEP    ,7*2            ;SEND FIRST SYNC CHARACTER
013144  104420  000013                READ    ,PRI13          ;READ REGISTER
013150  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
013154  042737  177767  001202        BIC     #^C<SF>,$TMP0   ;CLEAR UNWANTED BITS
013162  032737  000010  001202        BIT     #SF,$TMP0               ;TEST SF
013170  001014                        BNE     6$              ;SET SKIP ERROR
013172  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
013200  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
013206  052737  000010  001206        BIS     #SF,$TMP2               ;LOAD EXPECTED
013214  104017                        ERROR   15.             ;S/F DID NOT SET AFTER SYNC SENT
013216  000137  012740                JMP     1$
013222  104416  000003  000000  6$:   WRITE   ,SEC3,0         ;RESET TSOM
013230  104421  000020                STEP    ,8.*2           ;SEND SECOND SYNC
013234  104420  000013                READ    ,PRI13          ;READ REGISTER
013240  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
013244  042737  177767  001202        BIC     #^C<SF>,$TMP0   ;CLEAR UNWANTED BITS
013252  032737  000010  001202        BIT     #SF,$TMP0               ;TEST SF
013260  001014                        BNE     5$              ;SET SKIP ERROR
013262  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
013270  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
013276  052737  000010  001206        BIS     #SF,$TMP2               ;LOAD EXPECTED
013304  104017                        ERROR   15.             ;S/F DID NOT SET AFTER SECOND
                                                        ;SYNC
013306  000137  012740                JMP     1$              ;LOOP ON ERROR
013312                          5$:
013312  104421  000022                STEP    ,22             ;STEP CLOCK 11 TICKS
013316  104420  000013                READ    ,PRI13          ;READ REGISTER
```

```
3918  013322  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3919  013326  042737  177767  001202          BIC     #^C<SF>,$TMP0   ;CLEAR UNWANTED BITS
3920  013334  032737  000010  001202          BIT     #SF,$TMP0               ;TEST SF
3921  013342  001414                          BEQ     7$                      ;RESET SKIP ERROR
3922  013344  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
3923  013352  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
3924  013360  042737  000010  001206          BIC     #SF,$TMP2               ;LOAD EXPECTED
3925  013366  104020                          ERROR   16.             ;S/F DID NOT RESET AFTER 1 CLOCK
3926                                                                  ;TICK
3927  013370  000137  012740                  JMP     1$              ;LOOP ON ERROR
3928  013374                          7$:
3929  013374  104416  000002  000101          WRITE   ,SEC2,'A        ;LOAD DATA #1
3930  013402  104421  000020                  STEP    ,8.*2           ;SEND CHARACTER
3931  013406  104416  000002  000101          WRITE   ,SEC2,'A        ;LOAD DATA #2
3932  013414  104421  000020                  STEP    ,8.*2           ;SEND CHARACTER
3933  013420  104416  000002  000101          WRITE   ,SEC2,'A        ;LOAD DATA #3
3934  013426  104421  000020                  STEP    ,8.*2           ;SEND CHARACTER
3935  013432  104416  000002  000101          WRITE   ,SEC2,'A
3936  013440  104421  000004                  STEP    ,2*2
3937  013444  104420  000013                  READ    ,PRI13          ;READ REGISTER
3938  013450  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3939  013454  042737  177772  001202          BIC     #^C<RAC!RDA>,$TMP0      ;CLEAR UNWANTED BITS
3940  013462  032737  000004  001202          BIT     #RAC,$TMP0              ;TEST RAC
3941  013470  001014                          BNE     8$                      ;SET SKIP ERROR
3942  013472  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
3943  013500  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
3944  013506  052737  000004  001206          BIS     #RAC,$TMP2              ;LOAD EXPECTED
3945  013514  104021                          ERROR   17.             ;RECEIVER FAILED TO BECOME ACTIVE
3946                                                                  ;AFTER DATA SENT
3947  013516  000137  012740                  JMP     1$              ;LOOP ON ERROR
3948  013522                          8$:
3949  013522  032737  000001  001202          BIT     #RDA,$TMP0              ;TEST RDA
3950  013530  001014                          BNE     9$                      ;SET SKIP ERROR
3951  013532  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
3952  013540  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
3953  013546  052737  000001  001206          BIS     #RDA,$TMP2              ;LOAD EXPECTED
3954  013554  104022                          ERROR   18.             ;RECEIVE DATA AVAILIBLE DID
3955                                                                  ;NOT SET AFTER DATA SENT
3956  013556  000137  012740                  JMP     1$              ;LOOP ON ERROR
3957  013562                          9$:
3958  013562  104420  000000                  READ    ,SEC0           ;READ THE DATA RECEIVED
3959  013566  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
3960  013572  012737  000026  001204          MOV     #SYN,$TMP1      ;LOAD EXPECTED DATA
3961  013600  023737  001202  001204          CMP     $TMP0,$TMP1     ;COMPARE
3962  013606  001403                          BEQ     TST56           ;;
3963  013610  104024                          ERROR   20.             ;DATA SEND DOES NOT COMPARE
3964  013612  000137  012740                  JMP     1$              ;LOOP ON TEST
3965
3966
3967                                  ;****************************** TEST 56 ******************************
3968                                  ;*TRANSMIT AND RECEIVE INITIALIZATION TEST.
3969                                  ;*
3970                                  ;*THIS TEST WILL SYNC THE SEND RECEIVE LOGIC.
3971                                  ;*PROTOCOL = BOP
3972                                  ;*THIS TEST WILL PERFORM THE FOLLOWING SEQUENCE:
3973                                  ;*
```

```
                                        ;*
                                        ;*      1. CLEAR THE ADDRESS REGISTER
                                        ;*      2. ISSUE A MASTER CLEAR
                                        ;*      3. SET PROTOCOL = BOP, AND SET
                                        ;*         SECONDARY ADDRESS MODE
                                        ;*      4. LOAD THE ADDRESS INTO THE SYNC SECONDARY
                                        ;*         REGISTER
                                        ;*      5. SET TENA,RENA,MB,MC
                                        ;*      6. SET TSOM
                                        ;*      7. TEST FOR TBMT RESET
                                        ;*      8. STEP THE CLOCK 2 (10)
                                        ;*      9. TEST FOR TBMT SET AND TAC SET
                                        ;*      10. RESET TSOM
                                        ;*      11. STEP THE CLOCK 14 (10) FOR TRANSMIT
                                        ;*          OF FLAG AND ADDRESS AND LOOK FOR S/F
                                        ;*          TO BE SET.
                                        ;*      12. LOAD A CONTROL BYTE OF ZERO INTO THE TDB
                                        ;*      13. STEP THE CLOCK 2 (10)
                                        ;*      14. LOOK FOR S/F TO BE RESET
                                        ;*      15. STEP THE CLOCK 6 (10)
                                        ;*      16. LOAD A DATA CHARACTER INTO TDB
                                        ;*          AND TRANSMIT
                                        ;*      17. LOAD A DATA CHARACTER INTO TDB
                                        ;*          AND TRANSMIT
                                        ;*      18. LOAD A DATA CHARACTER INTO TDB
                                        ;*          AND TRANSMIT
                                        ;*      19. TEST FOR THE FOLLOWING CONDITIONS:
                                        ;*              A. RAC TO BE SET
                                        ;*              B. RDA TO BE SET
                                        ;*              C. RSOM TO BE SET
                                        ;*      20. READ THE DATA REGISTER TO HOLD THE CONTROL
                                        ;*          CHARACTER OF ZERO
                                        ;;*****************************************************************

                                        ;  TEST 56
                                        ;--------------
                                        ;;*****************************************************************
013616  000004                  TST56:  SCOPE
013620  012737  000056  001102          MOV     #56,$TSTNM              ; LOAD THE NO. OF THIS TEST
013626  012737  000056  001262          MOV     #56,TSTNUM              ;# FOR TYPE OUT
013634  012737  013642  001106          MOV     #1$,$LPADR      ;SET RETURN ADDRESS TO 1$
013642  104414                  1$:     MSTCLR                  ;ISSUE A MASTER CLEAR
013644  104416  000005  000020          WRITE   ,SEC5,BOP!SAM   ;SET MODE CONTROL REGISTER
                                                                ;       PROTOCOL = BOP
                                                                ;       SET SAM BIT4
013652  104416  000004  000376          WRITE   ,SEC4,376               ;LOAD SYNC SECONDARY ADDRESS REGISTER
013660  104416  000014  000321          WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
                                                                ;       TENA
                                                                ;       RENA
                                                                ;       MB AND MC
013666  104416  000003  000001          WRITE   ,SEC3,TSOM      ;SET TSOM
013674  104420  000013                  READ    ,PRI13          ;READ REGISTER
013700  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
013704  042737  177757  001202          BIC     #^C<TBMT>,$TMP0 ;CLEAR UNWANTED BITS
013712  032737  000020  001202          BIT     #TBMT,$TMP0             ;TEST TBMT
013720  001414                          BEQ     3$                      ;RESET SKIP ERROR
```

```
4030  013722  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4031  013730  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4032  013736  042737  000020  001206          BIC     #TBMT,$TMP2             ;LOAD EXPECTED
4033  013744  104014                          ERROR   12.             ;TBMT NOT RESET AFTER TSOM AND TENA
4034  013746  000137  013642                  JMP     1$              ;LOOP ON ERROR
4035                                                                  ;ENABLED
4036  013752  104421  000004          3$:     STEP    ,2*2            ;STEP THE CLOCK 2 TICKS
4037  013756  104420  000013                  READ    ,PRI13          ;READ REGISTER
4038  013762  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4039  013766  042737  177757  001202          BIC     #^C<TBMT>,$TMP0 ;CLEAR UNWANTED BITS
4040  013774  032737  000020  001202          BIT     #TBMT,$TMP0             ;TEST TBMT
4041  014002  001014                          BNE     4$              ;SET SKIP ERROR
4042  014004  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4043  014012  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4044  014020  052737  000020  001206          BIS     #TBMT,$TMP2             ;LOAD EXPECTED
4045  014026  104015                          ERROR   13.             ;TBMT FAILED TO SET WITH TSOM SET
4046                                                                  ;AND CLOCK STEPPED 2 TIMES
4047  014030  000137  013642                  JMP     1$              ;LOOP ON ERROR
4048  014034                          4$:
4049  014034  104416  000003  000000          WRITE   ,SEC3,0         ;RESET TSOM
4050  014042  104416  000002  000376          WRITE   ,SEC2,376               ;LOAD THE ADDRESS
4051  014050  104421  000016                  STEP    ,7.*2           ;SEND FLAG
4052  014054  104420  000013                  READ    ,PRI13          ;READ REGISTER
4053  014060  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4054  014064  042737  177767  001202          BIC     #^C<SF>,$TMP0   ;CLEAR UNWANTED BITS
4055  014072  032737  000010  001202          BIT     #SF,$TMP0               ;TEST SF
4056  014100  001014                          BNE     5$              ;SET SKIP ERROR
4057  014102  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4058  014110  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4059  014116  052737  000010  001206          BIS     #SF,$TMP2               ;LOAD EXPECTED
4060  014124  104016                          ERROR   14.             ;S/F DID NOT SET AFTER FLAG + ADDRESS
4061                                                                  ;SENT
4062  014126  000137  013642                  JMP     1$
4063  014132                          5$:
4064  014132  104416  000002  000000          WRITE   ,SEC2,0         ;LOAD CONTROL CHARACTER
4065  014140  104421  000022                  STEP    ,22             ;STEP CLOCK 11 TICKS
4066  014144  104420  000013                  READ    ,PRI13          ;READ REGISTER
4067  014150  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4068  014154  042737  177767  001202          BIC     #^C<SF>,$TMP0   ;CLEAR UNWANTED BITS
4069  014162  032737  000010  001202          BIT     #SF,$TMP0               ;TEST SF
4070  014170  001414                          BEQ     7$              ;RESET SKIP ERROR
4071  014172  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4072  014200  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4073  014206  042737  000010  001206          BIC     #SF,$TMP2               ;LOAD EXPECTED
4074  014214  104020                          ERROR   16.             ;S/F DID NOT RESET AFTER 1 CLOCK
4075                                                                  ;TICK
4076  014216  000137  013642                  JMP     1$              ;LOOP ON ERROR
4077  014222                          7$:
4078  014222  104416  000002  000101          WRITE   ,SEC2,'A        ;LOAD DATA #1
4079  014230  104421  000020                  STEP    ,8.*2           ;SEND CHARACTER
4080  014234  104416  000002  000101          WRITE   ,SEC2,'A        ;LOAD DATA #2
4081  014242  104421  000020                  STEP    ,8.*2           ;SEND CHARACTER
4082  014246  104416  000002  000101          WRITE   ,SEC2,'A        ;LOAD DATA #3
4083  014254  104421  000020                  STEP    ,8.*2           ;SEND CHARACTER
4084  014260  104416  000002  000101          WRITE   ,SEC2,'A
4085  014266  104421  000004                  STEP    ,2*2
```

```
4086  014272  104420  000013                  READ    ,PRI13          ;READ REGISTER
4087  014276  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4088  014302  042737  177772  001202          BIC     #^C<RAC!RDA>,$TMP0      ;CLEAR UNWANTED BITS
4089  014310  032737  000004  001202          BIT     #RAC,$TMP0              ;TEST RAC
4090  014316  001014                          BNE     8$                      ;SET SKIP ERROR
4091  014320  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4092  014326  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4093  014334  052737  000004  001206          BIS     #RAC,$TMP2              ;LOAD EXPECTED
4094  014342  104021                          ERROR   17.             ;RECEIVER FAILED TO BECOME ACTIVE
4095                                                                  ;AFTER DATA SENT
4096  014344  000137  013642                  JMP     1$              ;LOOP ON ERROR
4097  014350                          8$:
4098  014350  032737  000001  001202          BIT     #RDA,$TMP0              ;TEST RDA
4099  014356  001014                          BNE     9$                      ;SET SKIP ERROR
4100  014360  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4101  014366  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4102  014374  052737  000001  001206          BIS     #RDA,$TMP2              ;LOAD EXPECTED
4103  014402  104022                          ERROR   18.             ;RECEIVE DATA AVAILIBLE DID
4104                                                                  ;NOT SET AFTER DATA SENT
4105  014404  000137  013642                  JMP     1$              ;LOOP ON ERROR
4106  014410                          9$:
4107  014410  104420  000001                  READ    ,SEC1           ;READ REGISTER
4108  014414  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4109  014420  042737  177776  001202          BIC     #^C<RSOM>,$TMP0 ;CLEAR UNWANTED BITS
4110  014426  032737  000001  001202          BIT     #RSOM,$TMP0             ;TEST RSOM
4111  014434  001014                          BNE     10$                     ;SET SKIP ERROR
4112  014436  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4113  014444  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4114  014452  052737  000001  001206          BIS     #RSOM,$TMP2             ;LOAD EXPECTED
4115  014460  104023                          ERROR   19.             ;RSOM FAILED TO SET
4116  014462  000137  013642                  JMP     1$              ;LOOP ON ERROR
4117  014466                          10$:
4118  014466  104420  000000                  READ    ,SEC0           ;READ THE DATA RECEIVED
4119  014472  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4120  014476  012737  000376  001204          MOV     #376,$TMP1      ;FIRST DATA CHARACTER IS ADDRESS
4121  014504  023737  001202  001204          CMP     $TMP0,$TMP1     ;COMPARE
4122  014512  001403                          BEQ     TST57           ;;
4123  014514  104024                          ERROR   20.             ;DATA SEND DOES NOT COMPARE
4124  014516  000137  013642                  JMP     1$              ;LOOP ON TEST
```

```
                                        .SBTTL  END OF MESSAGE TEST

                                                ;****************************** TEST 57 ******************************
                                                ;*END OF MESSAGE TEST.
                                                ;*THIS TEST WILL PERFORM THE END
                                                ;*OF THE MESSAGE FOR PROTOCOL = BOP
                                                ;*
                                                ;*THIS TEST WILL PERFORM THE FOLLOWING SEQUENCE:
                                                ;*
                                                ;*      1. MASTER CLEAR
                                                ;*      2. SYNC THE SEND RECEIVE LOGIC
                                                ;*      3. TRANSMIT 3 DATA CHARACTERS
                                                ;*      4. TEST FOR RAC AND RDA
                                                ;*      5. WHEN LOADING THE THIRD DATA CHARACTER
                                                ;*         SET TEOM
                                                ;*      6. READ THE DATA AND DISCARD
                                                ;*      7. ACCEPT THE CRC CHECK CHARCATERS
                                                ;*      8. TEST FOR REOM SET
                                                ;:*****************************************************************

                                                ;  TEST 57
                                                ;-------------
                                        ;:*****************************************************************
014522  000004                          TST57:  SCOPE
014524  012737  000057  001102                  MOV     #57,$TSTNM              ; LOAD THE NO. OF THIS TEST
014532  012737  000057  001262                  MOV     #57,TSTNUM              ;# FOR TYPE OUT
014540  104414                          101$:   MSTCLR                  ;CLEAR THE KMC11
014542  005037  014674                  1$:     CLR     50$+2
014546  005037  015006                          CLR     11$+2
014552  005037  015054                          CLR     52$+2
014556  104416  000007  000000                  WRITE   ,SEC7,0         ;SET CHARACTER LENGTH TO 8 BITS
014564  104416  000005  000000                  WRITE   ,SEC5,BOP       ;LOAD MODE CONTROL REGISTER
014572  104416  000014  000321                  WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
                                                                        ;       TENA
                                                                        ;       RENA
                                                                        ;       MAINTENANCE BITS: MB, MC
014600  104416  000003  000001                  WRITE   ,SEC3,TSOM      ;SET TSOM
014606  104421  000002                  2$:     STEP    ,2              ;STEP THE CLOCK ONCE
014612  104420  000013                          READ    ,PRI13          ;READ THE STATUS REGISTER
014616  012637  001202                          MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
014622  032737  000020  001202                  BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
014630  001766                                  BEQ     2$              ;NO CONTINUE STEPPING THE CLOCK
014632  104416  000002  000000                  WRITE   ,SEC2,0         ;LOAD ADDRESS
014640  104416  000003  000000                  WRITE   ,SEC3,0         ;RESET TSOM
014646  104421  000002                  3$:     STEP    ,2              ;STEP THE CLOCK ONCE
014652  104420  000013                          READ    ,PRI13          ;READ THE STATUS REGISTER
014656  012637  001202                          MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
014662  032737  000010  001202                  BIT     #SF,$TMP0       ;TEST FOR S/F
014670  001017                                  BNE     4$
014672  105227  000000                  50$:    INCB    #0
014676  001363                                  BNE     3$
014700  013737  001202  001204                  MOV     $TMP0,$TMP1
014706  013737  001202  001206                  MOV     $TMP0,$TMP2
014714  052737  000010  001206                  BIS     #SF,$TMP2
014722  104016                                  ERROR   14.             ;S/F DID NOT SET AFTER FLAG SENT
```

```
4181  014724  000137  014542            JMP     1$              ;LOOP ON ERROR
4182  014730                      4$:
4183  014730  104421  000002      7$:   STEP    ,2              ;STEP THE CLOCK ONCE
4184  014734  104420  000013            READ    ,PRI13          ;READ THE STATUS REGISTER
4185  014740  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4186  014744  032737  000020  001202    BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
4187  014752  001403                    BEQ     10$
4188  014754  104416  000002  000101    WRITE   ,SEC2,'A        ;LOAD A DATA CHARACTER
4189  014762  032737  000001  001202 10$:BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
4190  014770  001405                    BEQ     11$
4191  014772  104420  000000            READ    ,SEC0           ;READ THE DATA REGISTER
4192  014776  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4193  015002  000406                    BR      12$
4194  015004  105227  000000      11$:  INCB    #0
4195  015010  001347                    BNE     7$
4196  015012  104042                    ERROR   34.             ;DEVICE IS HUNG
4197  015014  000137  014542            JMP     1$              ;LOOP ON ERROR
4198  015020  104416  000003  000002 12$:WRITE   ,SEC3,TEOM              ;END THE MESSAGE
4199  015026  104421  000002      13$:  STEP    ,2              ;STEP THE CLOCKONCE
4200  015032  104420  000013            READ    ,PRI13          ;READ THE STATUS REGISTER
4201  015036  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4202  015042  032737  000010  001202    BIT     #SF,$TMP0       ;TEST FOR SF AFTER TEOM SENT
4203  015050  001023                    BNE     15$             ;YES BRANCH
4204  015052  105227  000000      52$:  INCB    #0
4205  015056  001007                    BNE     14$
4206  015060  104026                    ERROR   22.             ;NO S/F AFTER TEOM SET
4207  015062  000137  014542            JMP     1$              ;LOOP ON ERROR
4208  015066  104420  000013            READ    ,PRI13
4209  015072  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4210  015076  032737  000001  001202 14$:BIT     #RDA,$TMP0      ;TEST FOR DATA
4211  015104  001750                    BEQ     13$
4212  015106  104420  000000            READ    ,SEC0           ;READ THE DATA
4213  015112  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4214  015116  000743                    BR      13$
4215  015120                      15$:
4216  015120  104421  000004            STEP    ,4
4217  015124  104420  000001            READ    ,SEC1           ;READ REGISTER
4218  015130  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4219  015134  042737  177775  001202    BIC     #^C<REOM>,$TMP0 ;CLEAR UNWANTED BITS
4220  015142  032737  000002  001202    BIT     #REOM,$TMP0             ;TEST REOM
4221  015150  001014                    BNE     16$                     ;SET SKIP ERROR
4222  015152  013737  001202  001204    MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4223  015160  013737  001204  001206    MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4224  015166  052737  000002  001206    BIS     #REOM,$TMP2             ;LOAD EXPECTED
4225  015174  104025                    ERROR   21.             ;REOM FAILED TO SET AFTER END OF MESSAGE
4226  015176  000137  014542            JMP     1$              ;LOOP ON ERROR
4227  015202                      16$:
4228  015202  000400                    BR      TST60           ;;
4229
4230
4231                                    ;****************************** TEST 60 ******************************
4232                                    ;*END OF MESSAGE TEST.
4233                                    ;*THIS TEST WILL PERFORM THE END
4234                                    ;*OF THE MESSAGE FOR PROTOCOL = CCP
4235                                    ;*
4236                                    ;*THIS TEST WILL PERFORM THE FOLLOWING SEQUENCE:
```

```
                                          ;*
                                          ;*          1. MASTER CLEAR
                                          ;*          2. SYNC THE SEND RECEIVE LOGIC
                                          ;*          3. TRANSMIT 3 DATA CHARACTERS
                                          ;*          4. TEST FOR RAC AND RDA
                                          ;*          5. WHEN LOADING THE THIRD DATA CHARACTER
                                          ;*             SET TEOM
                                          ;*          6. READ THE DATA AND DISCARD
                                          ;*          7. ACCEPT THE CRC CHECK CHARCATERS
                                          ;*          8. TEST FOR RESYNC OF THE LINE BY TESTING S/F
                                          ;:*******************************************************************

                                          ;  TEST 60
                                          ;--------------

                                  ;;*******************************************************************
015204  000004                    TST60:  SCOPE
015206  012737  000060  001102            MOV     #60,$TSTNM              ; LOAD THE NO. OF THIS TEST
015214  012737  000060  001262            MOV     #60,TSTNUM              ;# FOR TYPE OUT
015222  104414                    101$:   MSTCLR                  ;CLEAR THE KMC11
015224  005037  015356            1$:     CLR     50$+2
015230  005037  015472                    CLR     11$+2
015234  005037  015540                    CLR     52$+2
015240  104416  000007  000000            WRITE   ,SEC7,0                 ;SET CHARACTER LENGTH TO 8 BITS
015246  104416  000005  000100            WRITE   ,SEC5,CCP               ;LOAD MODE CONTROL REGISTER
015254  104416  000004  000026            WRITE   ,SEC4,SYN               ;LOAD A SYNC CHARACTER
015262  104416  000014  000321            WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
                                                                  ;         TENA
                                                                  ;         RENA
                                                                  ;         MAINTENANCE BITS: MB, MC
015270  104416  000003  000001            WRITE   ,SEC3,TSOM              ;SET TSOM
015276  104421  000002            2$:     STEP    ,2                      ;STEP THE CLOCK ONCE
015302  104420  000013                    READ    ,PRI13                  ;READ THE STATUS REGISTER
015306  012637  001202                    MOV     (SP)+,$TMPO             ;;POP STACK`INTO $TMPO
015312  032737  000020  001202            BIT     #TBMT,$TMPO             ;TEST FOR BUFFER EMPTY
015320  001766                            BEQ     2$                      ;NO CONTINUE STEPPING THE CLOCK
015322  104416  000002  000002            WRITE   ,SEC2,STX               ;LOAD A CHARACTER
015330  104421  000002            3$:     STEP    ,2                      ;STEP THE CLOCK ONCE
015334  104420  000013                    READ    ,PRI13                  ;READ THE STATUS REGISTER
015340  012637  001202                    MOV     (SP)+,$TMPO             ;;POP STACK INTO $TMPO
015344  032737  000010  001202            BIT     #SF,$TMPO               ;TEST FOR S/F
015352  001015                            BNE     4$
015354  105227  000000            50$:    INCB    #0
015360  001363                            BNE     3$
015362  013737  001202  001204            MOV     $TMPO,$TMP1
015370  013737  001202  001206            MOV     $TMPO,$TMP2
015376  052737  000010  001206            BIS     #SF,$TMP2
015404  104017                            ERROR   15.                     ;S/F DID NOT SET AFTER SYNC SENT
015406                            4$:
015406  104416  000003  000000            WRITE   ,SEC3,0                 ;RESET TSOM
015414  104421  000002            7$:     STEP    ,2                      ;STEP THE CLOCK ONCE
015420  104420  000013                    READ    ,PRI13                  ;READ THE STATUS REGISTER
015424  012637  001202                    MOV     (SP)+,$TMPO             ;;POP STACK INTO $TMPO
015430  032737  000020  001202            BIT     #TBMT,$TMPO             ;TEST FOR BUFFER EMPTY
015436  001403                            BEQ     10$
015440  104416  000002  000101            WRITE   ,SEC2,'A                ;LOAD A DATA CHARACTER
015446  032737  000001  001202    10$:    BIT     #RDA,$TMPO              ;TEST FOR DATA AVAILIBLE
```

```
 4293  015454  001405                          BEQ     11$
 4294  015456  104420  000000                  READ    ,SEC0           ;READ THE DATA REGISTER
 4295  015462  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
 4296  015466  000406                          BR      12$
 4297  015470  105227  000000          11$:    INCB    #0
 4298  015474  001347                          BNE     7$
 4299  015476  104042                          ERROR   34.             ;DEVICE IS HUNG
 4300  015500  000137  015224                  JMP     1$              ;LOOP ON ERROR
 4301  015504  104416  000003  000002  12$:    WRITE   ,SEC3,TEOM              ;END THE MESSAGE
 4302  015512  104421  000002          13$:    STEP    ,2              ;STEP THE CLOCKONCE
 4303  015516  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
 4304  015522  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
 4305  015526  032737  000010  001202          BIT     #SF,$TMP0       ;TEST FOR SF AFTER TEOM SENT
 4306  015534  001023                          BNE     15$             ;YES BRANCH
 4307  015536  105227  000000          52$:    INCB    #0
 4308  015542  001007                          BNE     14$
 4309  015544  104026                          ERROR   22.             ;NO S/F AFTER TEOM SET
 4310  015546  000137  015224                  JMP     1$              ;LOOP ON ERROR
 4311  015552  104420  000013                  READ    ,PRI13
 4312  015556  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
 4313  015562  032737  000001  001202  14$:    BIT     #RDA,$TMP0      ;TEST FOR DATA
 4314  015570  001750                          BEQ     13$
 4315  015572  104420  000000                  READ    ,SEC0           ;READ THE DATA
 4316  015576  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
 4317  015602  000743                          BR      13$
 4318  015604                          15$:
 4319  015604  000400                          BR      TST61           ;;
```

```
                                .SBTTL  CHARACTER LENGTH TEST

                                ;****************************** TEST 61 ******************************
                                ;*CHARACTER LENGTH TEST FOR PROTOCOL = BOP
                                ;*THIS TEST WILL PERFORM THE FOLLOWING
                                ;*SEQUENCE:
                                ;*
                                ;*      1. RESET THE CHARACTER LENGTH REGISTER
                                ;*         TO 7 BITS/CHAR
                                ;*      2. SET THE ONE BIT IN BIT POSITION 6 OF
                                ;*         OF THE DATA BIT REGISTER
                                ;*      3. SET THE DATA STEP CLOCK FOR 7 BITS/CHAR
                                ;*      4. ISSUE A MASTER CLEAR
                                ;*      5. SET THE CHARACTER LENGTH REGISTER TO
                                ;*         8 BITS/CHAR FOR INITIALIZATION OF
                                ;*         THE SEND RECEIVE LOGIC
                                ;*
                                ;*      6. SYNC THE SEND RECEIVE LOGIC FOR PROTOCOL
                                ;*         BOP BY SENDING THE FLAG AND ADDRESS PLUS
                                ;*         CONTROL BYTE AT 8 BITS/CHAR
                                ;*      7. SET THE CHARACTER LENGTH
                                ;*      8. TRANSMIT 3 DATA BYTES AND TEST THE
                                ;*         DATA BYTE RECEIVED TO THE DATA BYTE TRANSMITED
                                ;*      9. TEST IF ALL CHARACTER LENGTHS HAVE BEEN TESTED
                                ;*         IF NOT LOAD NEW VALUES AND CONTINUE
                                ;;*************************************************************

                                ;  TEST 61
                                ;--------------
                                ;;*************************************************************
015606  000004          TST61:  SCOPE
015610  012737  000061  001102        MOV     #61,$TSTNM              ; LOAD THE NO. OF THIS TEST
015616  012737  000061  001262        MOV     #61,TSTNUM              ;# FOR TYPE OUT
015624  012737  000347  016134        MOV     #347,4$         ;LOAD THE CHARACTER LENGTH REGISTER
015632  012737  000007  015722        MOV     #7,75$
                                                                ;TO 7 BITS/CHAR
015640  012737  000100  016120        MOV     #BIT6,6$        ;SET THE ONE BIT IN BIT POSITION
                                                                ;6 FOR 7 BITS/CHAR
015646  012737  015654  001106        MOV     #23$,$LPADR     ;SET THE RETURN AT 3$
015654  104414              23$:    MSTCLR                  ;ISSUE A MASTER CLEAR
015656  005037  016034              CLR     50$+2
015662  005037  016230              CLR     11$+2
015666  005037  016304              CLR     52$+2
015672  005037  016474              CLR     81$+2
015676  005037  016150              CLR     61$+2
015702  104416  000005  000000      WRITE   ,SEC5,BOP       ;LOAD THE MODE CONTROL REGISTER
015710  012737  177776  016124      MOV     #-2,70$+2
015716  104416  000007              WRITE   ,SEC7           ;LOAD THE CHARACTER LENGHT
015722  000000              75$:    .WORD   0
015724  104416  000014  000321      WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
                                                                ;       TENA
                                                                ;       RENA
                                                                ;       MAINTENANCE BITS: MB, MC
                                                                ;       MAINTENANCE BITS: MB, MC
015732  104416  000003  000001      WRITE   ,SEC3,TSOM      ;SET TSOM
```

```
4376  015740  104421  000002          2$:     STEP    ,2              ;STEP THE CLOCK ONCE
4377  015744  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
4378  015750  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4379  015754  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
4380  015762  001766                          BEQ     2$              ;NO CONTINUE STEPPING THE CLOCK
4381  015764  013737  016120  015776          MOV     6$,..+12
4382  015772  104416  000002  000000          WRITE   ,SEC2,0         ;LOAD ADDRESS
4383  016000  104416  000003  000000          WRITE   ,SEC3,0         ;RESET TSOM
4384  016006  104421  000002          3$:     STEP    ,2              ;STEP THE CLOCK ONCE
4385  016012  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
4386  016016  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4387  016022  032737  000010  001202          BIT     #SF,$TMP0       ;TEST FOR S/F
4388  016030  001017                          BNE     24$
4389  016032  105227  000000          50$:    INCB    #0
4390  016036  001363                          BNE     3$
4391  016040  013737  001202  001204          MOV     $TMP0,$TMP1
4392  016046  013737  001202  001206          MOV     $TMP0,$TMP2
4393  016054  052737  000010  001206          BIS     #SF,$TMP2
4394  016062  104016                          ERROR   14.             ;S/F DID NOT SET AFTER FLAG SENT
4395  016064  000137  015654                  JMP     23$             ;LOOP ON ERROR
4396  016070                          24$:
4397  016070  104421  000002          7$:     STEP    ,2              ;STEP THE CLOCK ONCE
4398  016074  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
4399  016100  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4400  016104  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
4401  016112  001411                          BEQ     8$
4402  016114  104416  000002                  WRITE   ,SEC2   ;LOAD A DATA CHARACTER
4403  016120  000000          6$:     .WORD   0
4404  016122  005227  177776          70$:    INC     #-2
4405  016126  001003                          BNE     71$
4406  016130  104416  000007                  WRITE   ,SEC7
4407  016134  000000          4$:     .WORD   0
4408  016136                          71$:
4409
4410  016136  032737  000010  001202  8$:     BIT     #SF,$TMP0       ;TEST IF SF STILL SET
4411  016144  001406                          BEQ     10$             ;NO BRANCH
4412  016146  105227  000000          61$:    INCB    #0              ;TIMEOUT FOR S/F
4413                                                                  ;TO GO AWAY
4414  016152  001003                          BNE     10$             ;BRANCH IF NOT TIMEOUT
4415  016154  104041                          ERROR   33.             ;DEVICE IS STILL SENDING SYNC CHARACTERS
4416  016156  000137  015654                  JMP     23$             ;LOOP ON ERROR
4417  016162  032737  000001  001202  10$:    BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
4418  016170  001416                          BEQ     11$
4419  016172  104420  000000                  READ    ,SEC0           ;READ THE DATAREGISTER
4420  016176  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4421  016202  013737  016120  001204          MOV     6$,$TMP1        ;GET EXPECTED
4422  016210  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR CORRECT DATA
4423  016216  001411                          BEQ     12$
4424  016220  104027                          ERROR   23.             ;DATA WAS NOT TRANSMITTED CORRECTLY
4425  016222  000137  015654                  JMP     23$             ;LOOP ON ERROR
4426  016226  105227  000000          11$:    INCB    #0
4427  016232  001316                          BNE     7$
4428  016234  104042                          ERROR   34.             ;DEVICE IS HUNG
4429  016236  000137  015654                  JMP     23$             ;LOOP ON ERROR
4430  016242  104416  000003  000002  12$:    WRITE   ,SEC2,TEOM              ;END THE MESSAGE
4431  016250  104421  000002          13$:    STEP    ,2              ;STEP THE CLOCKONCE
```

```
4432  016254  104420  000013                READ    ,PRI13          ;READ THE STATUS REGISTER
4433  016260  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4434  016264  032737  000010  001202        BIT     #SF,$TMP0       ;TEST FOR SF AFTER TEOM SENT
4435  016272  001403                        BEQ     52$             ;NO BRANCH
4436  016274  104044                        ERROR   36.             ;S/F SET BEFORE SECOND DATA BYTE RECEIVED
4437  016276  000137  015654                JMP     23$             ;LOOP ON ERROR
4438  016302  105227  000000        52$:    INCB    #0
4439  016306  001005                        BNE     14$
4440  016310  104042                        ERROR   34.             ;DEVICE IS HUNG
4441  016312  104420  000013                READ    ,PRI13
4442  016316  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4443  016322  032737  000001  001202  14$:   BIT     #RDA,$TMP0      ;TEST FOR DATA
4444  016330  001747                        BEQ     13$
4445  016332  104420  000000                READ    ,SEC0           ;READ THE DATA
4446  016336  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4447  016342  013737  016120  001204        MOV     6$,$TMP1
4448  016350  023737  001202  001204        CMP     $TMP0,$TMP1     ;TEST THE DATA
4449  016356  001403                        BEQ     15$
4450  016360  104027                        ERROR   23.             ;DATA WAS NOT TRANSMITTED CORRECTLY
4451  016362  000137  015654                JMP     23$             ;LOOP ON ERROR
4452  016366                        15$:
4453  016366  104421  000002                STEP    ,2              ;STEP THE CLOCK ONCE
4454  016372  104420  000013                READ    ,PRI13          ;READ THE STATUS REGISTER
4455  016376  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4456  016402  032737  000001  001202        BIT     #RDA,$TMP0      ;TEST IF DATA AVAILIBLE
4457  016410  001404                        BEQ     80$             ;NO BRANCH TO 80$
4458  016412  104420  000000                READ    ,SEC0           ;READ THE DATA
4459  016416  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4460  016422                        80$:
4461  016422  104420  000001                READ    ,SEC1           ;READ REGISTER
4462  016426  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4463  016432  042737  177775  001202        BIC     #^C<REOM>,$TMP0 ;CLEAR UNWANTED BITS
4464  016440  032737  000002  001202        BIT     #REOM,$TMP0             ;TEST REOM
4465  016446  001017                        BNE     16$                     ;SET SKIP ERROR
4466  016450  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4467  016456  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4468  016464  052737  000002  001206        BIS     #REOM,$TMP2             ;LOAD EXPECTED
4469  016472  105227  000000        81$:    INCB    #0
4470  016476  001333                        BNE     15$
4471  016500  104025                        ERROR   21.             ;REOM FAILED TO SET AFTER END OF MESSAGE
4472  016502  000137  015654                JMP     23$             ;LOOP ON ERROR
4473  016506                        16$:
4474  016506  000241                        CLC
4475  016510  006237  016120                ASR     6$
4476  016514  162737  000041  016134        SUB     #41,4$
4477  016522  005337  015722                DEC     75$
4478  016526  105737  016120                TSTB    6$
4479  016532  001403                        BEQ     TST62           ;;
4480  016534  000137  015654                JMP     23$
4481  016540                        17$:
4482  016540  000400                        BR      TST62           ;;
```

```
                                .SBTTL  ERROR CHECK TEST FOR ALL ERROR DETECTION

                                ;*************************** TEST 62 ****************************
                                ;*ERROR CHECK TEST FOR PROTOCOL = BOP
                                ;*ERROR CHECK X Y Z = CCITT-1
                                ;:************************************************************************
                                ;  TEST 62
                                ;--------------
                                ;:************************************************************************
016542  000004                  TST62:  SCOPE
016544  012737  000062  001102          MOV     #62,$TSTNM              ; LOAD THE NO. OF THIS TEST
016552  012737  000062  001262          MOV     #62,TSTNUM              ;# FOR TYPE OUT
016560  012737  000001  017042          MOV     #1,65$          ;LOAD CHARACTER BUFFER
016566  012737  016574  001106          MOV     #64$,$LPADR     ;SET RETURN ADDRESS TO 64$
016574  104414                  64$:    MSTCLR          ;ISSUE A MASTER CLEAR
016576  005037  017676                  CLR     19$
016602  104416  000007  000000          WRITE   ,SEC7,0         ;SET CHARACTER LENGTH TO 8 BITS/CHAR
016610  104416  000005  000000          WRITE   ,SEC5,BOP!CCITT-1       ;LOAD MODE CONTROL REGISTER
016616  005037  017250                  CLR     100$+2
016622  005037  017250                  CLR     100$+2
016626  005037  017274                  CLR     110$+2
016632  005037  017472                  CLR     52$+2
016636  005037  016756                  CLR     50$+2
016642  005037  017202                  CLR     11$+2
016646  104416  000014  000321          WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
                                                                ;       TENA
                                                                ;       RENA
                                                                ;       MAINTENANCE BITS: MB ,MC
016654  104416  000003  000001          WRITE   ,SEC3,TSOM      ;SET TSOM
016662  104421  000002          2$:     STEP    ,2              ;STEP THE CLOCK ONCE
016666  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
016672  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
016676  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
016704  001766                          BEQ     2$              ;NO CONTINUE STEPPING THE CLOCK
016706  013737  017042  016720          MOV     65$,,+12
016714  104416  000002  000000          WRITE   ,SEC2,0         ;LOAD ADDRESS
016722  104416  000003  000000          WRITE   ,SEC3,0         ;RESET TSOM
016730  104421  000002          3$:     STEP    ,2              ;STEP THE CLOCK ONCE
016734  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
016740  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
016744  032737  000010  001202          BIT     #SF,$TMP0       ;TEST FOR S/F
016752  001017                          BNE     24$
016754  105227  000000          50$:    INCB    #0
016760  001363                          BNE     3$
016762  013737  001202  001204          MOV     $TMP0,$TMP1
016770  013737  001202  001206          MOV     $TMP0,$TMP2
016776  052737  000010  001206          BIS     #SF,$TMP2
017004  104016                          ERROR   14.             ;S/F DID NOT SET AFTER FLAG SENT
017006  000137  016574                  JMP     64$             ;LOOP ON ERROR
017012                          24$:
017012  104421  000002          7$:     STEP    ,2              ;STEP THE CLOCK ONCE
017016  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
017022  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
017026  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
```

```
4539  017034  001405                        BEQ     8$
4540  017036  104416  000002                WRITE   ,SEC2           ;LOAD A DATA CHARACTER
4541  017042  000000                65$:    .WORD   0
4542  017044  005237  017676                INC     19$             ;+1 TO DATA OUT
4543  017050                        8$:
4544  017050  032737  000001  001202  10$:   BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
4545  017056  001450                        BEQ     11$
4546  017060  104420  000000                READ    ,SEC0           ;READ THE DATA REGISTER
4547  017064  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4548  017070  005337  017676                DEC     19$             ;-1 TO DATA (CHR. REC.)
4549  017074  013737  017042  001204        MOV     65$,$TMP1       ;GET EXPECTED
4550  017102  023737  001202  001204        CMP     $TMP0,$TMP1     ;TEST FOR CORRECT DATA
4551  017110  001403                        BEQ     30$             ;DATA OK BRANCH
4552  017112  104027                        ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
4553  017114  000137  016574                JMP     64$             ;LOOP ON ERROR
4554  017120                        30$:
4555  017120  104420  000001                READ    ,SEC1           ;READ REGISTER
4556  017124  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4557  017130  042737  177577  001202        BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
4558  017136  032737  000200  001202        BIT     #ERRCK,$TMP0            ;TEST ERRCK
4559  017144  001414                        BEQ     31$                     ;RESET SKIP ERROR
4560  017146  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4561  017154  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4562  017162  042737  000200  001206        BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
4563  017170  104031                        ERROR   25.             ;ERROR CHECK SET
4564  017172  000137  016574                JMP     64$             ;LOOP ON ERROR
4565  017176                        31$:
4566  017176  000406                        BR      12$
4567  017200  105227  000000        11$:    INCB    #0
4568  017204  001302                        BNE     7$
4569  017206  104042                        ERROR   34.             ;DEVICE IS HUNG
4570  017210  000137  016574                JMP     64$             ;LOOP ON ERROR
4571  017214  104416  000003  000002  12$:   WRITE   ,SEC3,TEOM             ;END THE MESSAGE
4572  017222  104421  000002        13$:    STEP    ,2              ;STEP THE CLOCKONCE
4573  017226  104420  000013                READ    ,PRI13          ;READ THE STATUS REGISTER
4574  017232  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4575  017236  032737  000010  001202        BIT     #SF,$TMP0       ;TEST FOR SF AFTER TEOM SENT
4576  017244  001406                        BEQ     14$             ;NO BRANCH
4577  017246  105227  000000        100$:   INCB    #0
4578  017252  001003                        BNE     14$
4579  017254  104044                        ERROR   36.             ;S/F CAME UP BEFORE LAST DATA BYTE RECEIVED
4580  017256  000137  016574                JMP     64$             ;LOOP ON TEST
4581  017262  032737  000001  001202  14$:   BIT     #RDA,$TMP0      ;TEST FOR DATA
4582  017270  001006                        BNE     120$
4583  017272  105227  000000        110$:   INCB    #0
4584  017276  001351                        BNE     13$
4585  017300  104042                        ERROR   34.             ;DEVICE IS HUNG
4586  017302  000137  016574                JMP     64$             ;LOOP ON ERROR
4587  017306                        120$:
4588  017306  104420  000000                READ    ,SEC0           ;READ THE DATA
4589  017312  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4590  017316  013737  017042  001204        MOV     65$,$TMP1       ;GET EXPECTED DATA
4591  017324  023737  001202  001204        CMP     $TMP0,$TMP1     ;TEST DATA
4592  017332  001403                        BEQ     40$             ;OK BRANCH
4593  017334  104027                        ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
4594  017336  000137  016574                JMP     64$             ;LOOP ON ERROR
```

```
4595  017342                                40$:
4596  017342  104420  000001                      READ    ,SEC1           ;READ REGISTER
4597  017346  012637  001202                      MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4598  017352  042737  177577  001202              BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
4599  017360  032737  000200  001202              BIT     #ERRCK,$TMP0            ;TEST ERRCK
4600  017366  001414                              BEQ     41$                     ;RESET SKIP ERROR
4601  017370  013737  001202  001204              MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4602  017376  013737  001204  001206              MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4603  017404  042737  000200  001206              BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
4604  017412  104031                              ERROR   25.
4605  017414  000137  016574                      JMP     64$             ;LOOP ON ERROR
4606  017420                                41$:
4607  017420  104421  000002                53$:  STEP    ,2              ;STEP THE CLOCK ONE TIME
4608  017424  104420  000013                      READ    ,PRI13          ;READ THE STATUS REGISTER
4609  017430  012637  001202                      MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4610  017434  032737  000001  001202              BIT     #RDA,$TMP0      ;TEST FOR DATA
4611  017442  001406                              BEQ     70$
4612  017444  104420  000000                      READ    ,SEC0           ;READ THE DATA
4613  017450  012637  001204                      MOV     (SP)+,$TMP1     ;;POP STACK INTO $TMP1
4614  017454  005337  017676                      DEC     19$             ;-1 TO DATA CHRS.
4615  017460                                70$:
4616  017460  032737  000010  001202              BIT     #SF,$TMP0       ;TEST FOR S/F AFTER TEOM SET
4617  017466  001006                              BNE     15$             ;YES BRANCH
4618  017470  105227  000000                52$:  INCB    #0
4619  017474  001351                              BNE     53$
4620  017476  104026                              ERROR   22.             ;S/F DID NOT SET AFTER TEOM SET
4621  017500  000137  016574                      JMP     64$             ;LOOP ON ERROR
4622  017504                                54$:
4623  017504                                15$:
4624  017504  104421  000004                      STEP    ,4
4625  017510  104420  000001                      READ    ,SEC1           ;READ REGISTER
4626  017514  012637  001202                      MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4627  017520  042737  177775  001202              BIC     #^C<REOM>,$TMP0 ;CLEAR UNWANTED BITS
4628  017526  032737  000002  001202              BIT     #REOM,$TMP0             ;TEST REOM
4629  017534  001014                              BNE     16$                     ;SET SKIP ERROR
4630  017536  013737  001202  001204              MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4631  017544  013737  001204  001206              MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4632  017552  052737  000002  001206              BIS     #REOM,$TMP2             ;LOAD EXPECTED
4633  017560  104025                              ERROR   21.             ;REOM FAILED TO SET AFTER END OF MESSAGE
4634  017562  000137  016574                      JMP     64$             ;LOOP ON ERROR
4635  017566                                16$:
4636  017566  104420  000001                      READ    ,SEC1           ;READ REGISTER
4637  017572  012637  001202                      MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4638  017576  042737  177577  001202              BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
4639  017604  032737  000200  001202              BIT     #ERRCK,$TMP0            ;TEST ERRCK
4640  017612  001414                              BEQ     17$                     ;RESET SKIP ERROR
4641  017614  013737  001202  001204              MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4642  017622  013737  001204  001206              MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4643  017630  042737  000200  001206              BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
4644  017636  104031                              ERROR   25.
4645  017640  000137  016574                      JMP     64$             ;LOOP ON ERROR
4646  017644  005237  017042                17$:  INC     65$
4647  017650  022737  000176  017042              CMP     #176,65$                ;TEST IF GO AHEAD CHARACTER
4648  017656  001772                              BEQ     17$
4649  017660  022737  000377  017042              CMP     #377,65$                ;TEST IF ALL CHARACTER'S TESTED
4650  017666  001402                              BEQ     18$
```

```
                                        JMP     64$
                                18$:
                                        BR      TST63           ;;
                                19$:    0                       ;CHR. COUNTER


                                        ;***************************** TEST 63 *****************************
                                        ;*ERROR CHECK TEST FOR PROTOCOL = BOP
                                        ;*ERROR CHECK X Y Z = CCITT-0
                                        ;;*************************************************************************


                                        ;  TEST 63
                                        ;-------------
                                ;;*************************************************************************
                                TST63:  SCOPE
                                        MOV     #63,$TSTNM              ; LOAD THE NO. OF THIS TEST
                                        MOV     #63,TSTNUM              ;# FOR TYPE OUT
                                        MOV     #1,65$          ;LOAD CHARACTER BUFFER
                                        MOV     #64$,$LPADR     ;SET RETURN ADDRESS TO 64$
                                64$:    MSTCLR          ;ISSUE A MASTER CLEAR
                                        CLR     19$
                                        WRITE   ,SEC7,0         ;SET CHARACTER LENGTH TO 8 BITS/CHAR
                                        WRITE   ,SEC5,BOP!CCITT-0       ;LOAD MODE CONTROL REGISTER
                                        CLR     100$+2
                                        CLR     100$+2
                                        CLR     110$+2
                                        CLR     52$+2
                                        CLR     50$+2
                                        CLR     11$+2
                                        WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
                                                                        ;       TENA
                                                                        ;       RENA
                                                                        ;       MAINTENANCE BITS: MB ,MC
                                        WRITE   ,SEC3,TSOM      ;SET TSOM
                                2$:     STEP    ,2              ;STEP THE CLOCK ONCE
                                        READ    ,PRI13          ;READ THE STATUS REGISTER
                                        MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
                                        BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
                                        BEQ     2$              ;NO CONTINUE STEPPING THE CLOCK
                                        MOV     65$,,+12
                                        WRITE   ,SEC2,0         ;LOAD ADDRESS
                                        WRITE   ,SEC3,0         ;RESET TSOM
                                3$:     STEP    ,2              ;STEP THE CLOCK ONCE
                                        READ    ,PRI13          ;READ THE STATUS REGISTER
                                        MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
                                        BIT     #SF,$TMP0       ;TEST FOR S/F
                                        BNE     24$
                                50$:    INCB    #0
                                        BNE     3$
                                        MOV     $TMP0,$TMP1
                                        MOV     $TMP0,$TMP2
                                        BIS     #SF,$TMP2
                                        ERROR   14.             ;S/F DID NOT SET AFTER FLAG SENT
                                        JMP     64$             ;LOOP ON ERROR
                                24$:
                                7$:     STEP    ,2              ;STEP THE CLOCK ONCE
```

```
4707  020154  104420  000013            READ    ,PRI13          ;READ THE STATUS REGISTER
4708  020160  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4709  020164  032737  000020  001202    BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
4710  020172  001405                    BEQ     8$
4711  020174  104416  000002            WRITE   ,SEC2           ;LOAD A DATA CHARACTER
4712  020200  000000              65$:  .WORD   0
4713  020202  005237  021034            INC     19$             ;+1 TO DATA OUT
4714  020206                        8$:
4715  020206  032737  000001  001202  10$:   BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
4716  020214  001450                    BEQ     11$
4717  020216  104420  000000            READ    ,SEC0           ;READ THE DATA REGISTER
4718  020222  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4719  020226  005337  021034            DEC     19$             ;-1 TO DATA (CHR. REC.)
4720  020232  013737  020200  001204    MOV     65$,$TMP1       ;GET EXPECTED
4721  020240  023737  001202  001204    CMP     $TMP0,$TMP1     ;TEST FOR CORRECT DATA
4722  020246  001403                    BEQ     30$             ;DATA OK BRANCH
4723  020250  104027                    ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
4724  020252  000137  017732            JMP     64$             ;LOOP ON ERROR
4725  020256                        30$:
4726  020256  104420  000001            READ    ,SEC1           ;READ REGISTER
4727  020262  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4728  020266  042737  177577  001202    BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
4729  020274  032737  000200  001202    BIT     #ERRCK,$TMP0            ;TEST ERRCK
4730  020302  001414                    BEQ     31$                     ;RESET SKIP ERROR
4731  020304  013737  001202  001204    MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4732  020312  013737  001204  001206    MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4733  020320  042737  000200  001206    BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
4734  020326  104031                    ERROR   25.             ;ERROR CHECK SET
4735  020330  000137  017732            JMP     64$             ;LOOP ON ERROR
4736  020334                        31$:
4737  020334  000406                    BR      12$
4738  020336  105227  000000      11$:  INCB    #0
4739  020342  001302                    BNE     7$
4740  020344  104042                    ERROR   34.             ;DEVICE IS HUNG
4741  020346  000137  017732            JMP     64$             ;LOOP ON ERROR
4742  020352  104416  000003  000002  12$:   WRITE   ,SEC3,TEOM          ;END THE MESSAGE
4743  020360  104421  000002      13$:  STEP    ,2              ;STEP THE CLOCKONCE
4744  020364  104420  000013            READ    ,PRI13          ;READ THE STATUS REGISTER
4745  020370  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4746  020374  032737  000010  001202    BIT     #SF,$TMP0       ;TEST FOR SF AFTER TEOM SENT
4747  020402  001406                    BEQ     14$             ;NO BRANCH
4748  020404  105227  000000     100$:  INCB    #0
4749  020410  001003                    BNE     14$
4750  020412  104044                    ERROR   36.             ;S/F CAME UP BEFORE LAST DATA BYTE RECEIVED
4751  020414  000137  017732            JMP     64$             ;LOOP ON TEST
4752  020420  032737  000001  001202  14$:   BIT     #RDA,$TMP0      ;TEST FOR DATA
4753  020426  001006                    BNE     120$
4754  020430  105227  000000     110$:  INCB    #0
4755  020434  001351                    BNE     13$
4756  020436  104042                    ERROR   34.             ;DEVICE IS HUNG
4757  020440  000137  017732            JMP     64$             ;LOOP ON ERROR
4758  020444                       120$:
4759  020444  104420  000000            READ    ,SEC0           ;READ THE DATA
4760  020450  012637  001202            MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4761  020454  013737  020200  001204    MOV     65$,$TMP1       ;GET EXPECTED DATA
4762  020462  023737  001202  001204    CMP     $TMP0,$TMP1     ;TEST DATA
```

```
4763  020470  001403                        BEQ     40$             ;OK BRANCH
4764  020472  104027                        ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
4765  020474  000137  017732                JMP     64$             ;LOOP ON ERROR
4766  020500                          40$:
4767  020500  104420  000001                READ    ,SEC1           ;READ REGISTER
4768  020504  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4769  020510  042737  177577  001202        BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
4770  020516  032737  000200  001202        BIT     #ERRCK,$TMP0            ;TEST ERRCK
4771  020524  001414                        BEQ     41$                     ;RESET SKIP ERROR
4772  020526  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4773  020534  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4774  020542  042737  000200  001206        BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
4775  020550  104031                        ERROR   25.
4776  020552  000137  017732                JMP     64$             ;LOOP ON ERROR
4777  020556                          41$:
4778  020556  104421  000002          53$:  STEP    ,2              ;STEP THE CLOCK ONE TIME
4779  020562  104420  000013                READ    ,PRI13          ;READ THE STATUS REGISTER
4780  020566  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4781  020572  032737  000001  001202        BIT     #RDA,$TMP0      ;TEST FOR DATA
4782  020600  001406                        BEQ     70$
4783  020602  104420  000000                READ    ,SEC0           ;READ THE DATA
4784  020606  012637  001204                MOV     (SP)+,$TMP1     ;;POP STACK INTO $TMP1
4785  020612  005337  021034                DEC     19$             ;-1 TO DATA CHRS.
4786  020616                          70$:
4787  020616  032737  000010  001202        BIT     #SF,$TMP0       ;TEST FOR S/F AFTER TEOM SET
4788  020624  001006                        BNE     15$             ;YES BRANCH
4789  020626  105227  000000          52$:  INCB    #0
4790  020632  001351                        BNE     53$
4791  020634  104026                        ERROR   22.             ;S/F DID NOT SET AFTER TEOM SET
4792  020636  000137  017732                JMP     64$             ;LOOP ON ERROR
4793  020642                          54$:
4794  020642                          15$:
4795  020642  104421  000004                STEP    ,4
4796  020646  104420  000001                READ    ,SEC1           ;READ REGISTER
4797  020652  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4798  020656  042737  177775  001202        BIC     #^C<REOM>,$TMP0 ;CLEAR UNWANTED BITS
4799  020664  032737  000002  001202        BIT     #REOM,$TMP0             ;TEST REOM
4800  020672  001014                        BNE     16$                     ;SET SKIP ERROR
4801  020674  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4802  020702  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4803  020710  052737  000002  001206        BIS     #REOM,$TMP2             ;LOAD EXPECTED
4804  020716  104025                        ERROR   21.             ;REOM FAILED TO SET AFTER END OF MESSAGE
4805  020720  000137  017732                JMP     64$             ;LOOP ON ERROR
4806  020724                          16$:
4807  020724  104420  000001                READ    ,SEC1           ;READ REGISTER
4808  020730  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4809  020734  042737  177577  001202        BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
4810  020742  032737  000200  001202        BIT     #ERRCK,$TMP0            ;TEST ERRCK
4811  020750  001414                        BEQ     17$                     ;RESET SKIP ERROR
4812  020752  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4813  020760  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4814  020766  042737  000200  001206        BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
4815  020774  104031                        ERROR   25.
4816  020776  000137  017732                JMP     64$             ;LOOP ON ERROR
4817  021002  005237  020200          17$:  INC     65$
4818  021006  022737  000176  020200        CMP     #176,65$                ;TEST IF GO AHEAD CHARACTER
```

```
4819  021014  001772'                        BEQ     17$
4820  021016  022737  000377  020200        CMP     #377,65$                ;TEST IF ALL CHARACTER'S TESTED
4821  021024  001402                        BEQ     18$
4822  021026  000137  017732                JMP     64$
4823  021032                          18$:
4824  021032  000401                        BR      TST64           ;;
4825  021034  000000                  19$:  0                       ;CHR. COUNTER
4826
4827
4828                                        ;****************************** TEST 64 ******************************
4829                                        ;*ERROR CHECK TEST FOR PROTOCOL = BOP
4830                                        ;*ERROR CHECK X Y Z = CRC16
4831                                        ;;*************************************************************************
4832
4833                                        ;  TEST 64
4834                                        ;-------------
4835                                  ;;****************************************************************
4836  021036  000004                  TST64:  SCOPE
4837  021040  012737  000064  001102        MOV     #64,$TSTNM              ; LOAD THE NO. OF THIS TEST
4838  021046  012737  000064  001262        MOV     #64,TSTNUM              ;# FOR TYPE OUT
4839  021054  012737  000001  021336        MOV     #1,65$          ;LOAD CHARACTER BUFFER
4840  021062  012737  021070  001106        MOV     #64$,$LPADR     ;SET RETURN ADDRESS TO 64$
4841  021070  104414                  64$:  MSTCLR          ;ISSUE A MASTER CLEAR
4842  021072  005037  022172                CLR     19$
4843  021076  104416  000007  000000        WRITE   ,SEC7,0         ;SET CHARACTER LENGTH TO 8 BITS/CHAR
4844  021104  104416  000005  000003        WRITE   ,SEC5,BOP!CRC16 ;LOAD MODE CONTROL REGISTER
4845  021112  005037  021544                CLR     100$+2
4846  021116  005037  021544                CLR     100$+2
4847  021122  005037  021570                CLR     110$+2
4848  021126  005037  021766                CLR     52$+2
4849  021132  005037  021252                CLR     50$+2
4850  021136  005037  021476                CLR     11$+2
4851  021142  104416  000014  000321        WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
4852                                                                ;       TENA
4853                                                                ;       RENA
4854                                                                ;       MAINTENANCE BITS: MB ,MC
4855  021150  104416  000003  000001        WRITE   ,SEC3,TSOM      ;SET TSOM
4856  021156  104421  000002          2$:   STEP    ,2              ;STEP THE CLOCK ONCE
4857  021162  104420  000013                READ    ,PRI13          ;READ THE STATUS REGISTER
4858  021166  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4859  021172  032737  000020  001202        BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
4860  021200  001766                        BEQ     2$              ;NO CONTINUE STEPPING THE CLOCK
4861  021202  013737  021336  021214        MOV     65$,,+12
4862  021210  104416  000002  000000        WRITE   ,SEC2,0         ;LOAD ADDRESS
4863  021216  104416  000003  000000        WRITE   ,SEC3,0         ;RESET TSOM
4864  021224  104421  000002          3$:   STEP    ,2              ;STEP THE CLOCK ONCE
4865  021230  104420  000013                READ    ,PRI13          ;READ THE STATUS REGISTER
4866  021234  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4867  021240  032737  000010  001202        BIT     #SF,$TMP0       ;TEST FOR S/F
4868  021246  001017                        BNE     24$
4869  021250  105227  000000          50$:  INCB    #0
4870  021254  001363                        BNE     3$
4871  021256  013737  001202  001204        MOV     $TMP0,$TMP1
4872  021264  013737  001202  001206        MOV     $TMP0,$TMP2
4873  021272  052737  000010  001206        BIS     #SF,$TMP2
4874  021300  104016                        ERROR   14.             ;S/F DID NOT SET AFTER FLAG SENT
```

```
4875  021302  000137  021070                    JMP     64$             ;LOOP ON ERROR
4876  021306                            24$:
4877  021306  104421  000002            7$:     STEP    ,2              ;STEP THE CLOCK ONCE
4878  021312  104420  000013                    READ    ,PRI13          ;READ THE STATUS REGISTER
4879  021316  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4880  021322  032737  000020  001202            BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
4881  021330  001405                            BEQ     8$
4882  021332  104416  000002                    WRITE   ,SEC2           ;LOAD A DATA CHARACTER
4883  021336  000000                    65$:    .WORD   0
4884  021340  005237  022172                    INC     19$             ;+1 TO DATA OUT
4885  021344                            8$:
4886  021344  032737  000001  001202    10$:    BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
4887  021352  001450                            BEQ     11$
4888  021354  104420  000000                    READ    ,SEC0           ;READ THE DATA REGISTER
4889  021360  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4890  021364  005337  022172                    DEC     19$             ;-1 TO DATA (CHR. REC.)
4891  021370  013737  021336  001204            MOV     65$,$TMP1       ;GET EXPECTED
4892  021376  023737  001202  001204            CMP     $TMP0,$TMP1     ;TEST FOR CORRECT DATA
4893  021404  001403                            BEQ     30$             ;DATA OK BRANCH
4894  021406  104027                            ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
4895  021410  000137  021070                    JMP     64$             ;LOOP ON ERROR
4896  021414                            30$:
4897  021414  104420  000001                    READ    ,SEC1           ;READ REGISTER
4898  021420  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4899  021424  042737  177577  001202            BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
4900  021432  032737  000200  001202            BIT     #ERRCK,$TMP0            ;TEST ERRCK
4901  021440  001414                            BEQ     31$                     ;RESET SKIP ERROR
4902  021442  013737  001202  001204            MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4903  021450  013737  001204  001206            MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4904  021456  042737  000200  001206            BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
4905  021464  104031                            ERROR   25.             ;ERROR CHECK SET
4906  021466  000137  021070                    JMP     64$             ;LOOP ON ERROR
4907  021472                            31$:
4908  021472  000406                            BR      12$
4909  021474  105227  000000            11$:    INCB    #0
4910  021500  001302                            BNE     7$
4911  021502  104042                            ERROR   34.             ;DEVICE IS HUNG
4912  021504  000137  021070                    JMP     64$             ;LOOP ON ERROR
4913  021510  104416  000003  000002    12$:    WRITE   ,SEC3,TEOM              ;END THE MESSAGE
4914  021516  104421  000002            13$:    STEP    ,2              ;STEP THE CLOCKONCE
4915  021522  104420  000013                    READ    ,PRI13          ;READ THE STATUS REGISTER
4916  021526  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4917  021532  032737  000010  001202            BIT     #SF,$TMP0       ;TEST FOR SF AFTER TEOM SENT
4918  021540  001406                            BEQ     14$             ;NO BRANCH
4919  021542  105227  000000            100$:   INCB    #0
4920  021546  001003                            BNE     14$
4921  021550  104044                            ERROR   36.             ;S/F CAME UP BEFORE LAST DATA BYTE RECEIVED
4922  021552  000137  021070                    JMP     64$             ;LOOP ON TEST
4923  021556  032737  000001  001202    14$:    BIT     #RDA,$TMP0      ;TEST FOR DATA
4924  021564  001006                            BNE     120$
4925  021566  105227  000000            110$:   INCB    #0
4926  021572  001351                            BNE     13$
4927  021574  104042                            ERROR   34.             ;DEVICE IS HUNG
4928  021576  000137  021070                    JMP     64$             ;LOOP ON ERROR
4929  021602                            120$:
4930  021602  104420  000000                    READ    ,SEC0           ;READ THE DATA
```

```
4931  021606  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4932  021612  013737  021336  001204            MOV     65$,$TMP1       ;GET EXPECTED DATA
4933  021620  023737  001202  001204            CMP     $TMP0,$TMP1     ;TEST DATA
4934  021626  001403                            BEQ     40$             ;OK BRANCH
4935  021630  104027                            ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
4936  021632  000137  021070                    JMP     64$             ;LOOP ON ERROR
4937  021636                          40$:
4938  021636  104420  000001                    READ    ,SEC1           ;READ REGISTER
4939  021642  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4940  021646  042737  177577  001202            BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
4941  021654  032737  000200  001202            BIT     #ERRCK,$TMP0            ;TEST ERRCK
4942  021662  001414                            BEQ     41$                     ;RESET SKIP ERROR
4943  021664  013737  001202  001204            MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4944  021672  013737  001204  001206            MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4945  021700  042737  000200  001206            BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
4946  021706  104031                            ERROR   25.
4947  021710  000137  021070                    JMP     64$             ;LOOP ON ERROR
4948  021714                          41$:
4949  021714  104421  000002          53$:      STEP    ,2              ;STEP THE CLOCK ONE TIME
4950  021720  104420  000013                    READ    ,PRI13          ;READ THE STATUS REGISTER
4951  021724  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4952  021730  032737  000001  001202            BIT     #RDA,$TMP0      ;TEST FOR DATA
4953  021736  001406                            BEQ     70$
4954  021740  104420  000000                    READ    ,SEC0           ;READ THE DATA
4955  021744  012637  001204                    MOV     (SP)+,$TMP1     ;;POP STACK INTO $TMP1
4956  021750  005337  022172                    DEC     19$             ;-1 TO DATA CHRS.
4957  021754                          70$:
4958  021754  032737  000010  001202            BIT     #SF,$TMP0       ;TEST FOR S/F AFTER TEOM SET
4959  021762  001006                            BNE     15$             ;YES BRANCH
4960  021764  105227  000000          52$:      INCB    #0
4961  021770  001351                            BNE     53$
4962  021772  104026                            ERROR   22.             ;S/F DID NOT SET AFTER TEOM SET
4963  021774  000137  021070                    JMP     64$             ;LOOP ON ERROR
4964  022000                          54$:
4965  022000                          15$:
4966  022000  104421  000004                    STEP    ,4
4967  022004  104420  000001                    READ    ,SEC1           ;READ REGISTER
4968  022010  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4969  022014  042737  177775  001202            BIC     #^C<REOM>,$TMP0 ;CLEAR UNWANTED BITS
4970  022022  032737  000002  001202            BIT     #REOM,$TMP0             ;TEST REOM
4971  022030  001014                            BNE     16$                     ;SET SKIP ERROR
4972  022032  013737  001202  001204            MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4973  022040  013737  001204  001206            MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4974  022046  052737  000002  001206            BIS     #REOM,$TMP2             ;LOAD EXPECTED
4975  022054  104025                            ERROR   21.             ;REOM FAILED TO SET AFTER END OF MESSAGE
4976  022056  000137  021070                    JMP     64$             ;LOOP ON ERROR
4977  022062                          16$:
4978  022062  104420  000001                    READ    ,SEC1           ;READ REGISTER
4979  022066  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
4980  022072  042737  177577  001202            BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
4981  022100  032737  000200  001202            BIT     #ERRCK,$TMP0            ;TEST ERRCK
4982  022106  001414                            BEQ     17$                     ;RESET SKIP ERROR
4983  022110  013737  001202  001204            MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
4984  022116  013737  001204  001206            MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
4985  022124  042737  000200  001206            BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
4986  022132  104031                            ERROR   25.
```

```
                                    JMP     64$             ;LOOP ON ERROR
                            17$:    INC     65$
                                    CMP     #176,65$                ;TEST IF GO AHEAD CHARACTER
                                    BEQ     17$
                                    CMP     #377,65$                ;TEST IF ALL CHARACTER'S TESTED
                                    BEQ     18$
                                    JMP     64$
                            18$:
                                    BR      TST65           ;;
                            19$:    0                       ;CHR. COUNTER


                                    ;***************************** TEST 65 *****************************
                                    ;*ERROR CHECK TEST FOR PROTOCOL = CCP
                                    ;*ERROR CHECK X Y Z = CCITT-1
                                    ;;***************************************************************************

                                    ;  TEST 65
                                    ;-------------
                            ;;*******************************************************************
                            TST65:  SCOPE
                                    MOV     #65,$TSTNM              ; LOAD THE NO. OF THIS TEST
                                    MOV     #65,TSTNUM              ;# FOR TYPE OUT
                                    MOV     #1,65$          ;LOAD CHARACTER BUFFER
                                    MOV     #64$,$LPADR     ;SET RETURN ADDRESS TO 64$
                            64$:    MSTCLR          ;ISSUE A MASTER CLEAR
                                    CLR     19$
                                    WRITE   ,SEC7,0         ;SET CHARACTER LENGTH TO 8 BITS/CHAR
                                    WRITE   ,SEC5,CCP!SSL!CCITT-1   ;LOAD MODE CONTROL
                                    WRITE   ,SEC4,SYN       ;LOAD SYN CHARACTER
                                    CLR     100$+2
                                    CLR     110$+2
                                    CLR     52$+2
                                    CLR     50$+2
                                    CLR     11$+2
                                    WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
                                                            ;       TENA
                                                            ;       RENA
                                                            ;       MAINTENANCE BITS: MB ,MC
                                    WRITE   ,SEC3,TSOM      ;SET TSOM
                            2$:     STEP    ,2              ;STEP THE CLOCK ONCE
                                    READ    ,PRI13          ;READ THE STATUS REGISTER
                                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
                                    BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
                                    BEQ     2$              ;NO CONTINUE STEPPING THE CLOCK
                                    MOV     65$,.+12
                                    WRITE   ,SEC2,0 ;LOAD A CHARACTER
                                    INC     19$
                            3$:     STEP    ,2              ;STEP THE CLOCK ONCE
                                    READ    ,PRI13          ;READ THE STATUS REGISTER
                                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
                                    BIT     #SF,$TMP0       ;TEST FOR S/F
                                    BNE     24$
                            50$:    INCB    #0
                                    BNE     3$
                                    MOV     $TMP0,$TMP1
```

```
5043  022422  013737  001202  001206          MOV     $TMP0,$TMP2
5044  022430  052737  000010  001206          BIS     #SF,$TMP2
5045  022436  104017                          ERROR   15.             ;S/F DID NOT SET AFTER SYNC SENT
5046  022440  000137  022226                  JMP     64$             ;LOOP ON ERROR
5047  022444                          24$:
5048  022444  104416  000003  000000          WRITE   ,SEC3,0         ;RESET TSOM
5049  022452  104421  000002          7$:     STEP    ,2              ;STEP THE CLOCK ONCE
5050  022456  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
5051  022462  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5052  022466  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
5053  022474  001405                          BEQ     8$
5054  022476  104416  000002                  WRITE   ,SEC2           ;LOAD A DATA CHARACTER
5055  022502  000000                  65$:    .WORD   0
5056  022504  005237  023174                  INC     19$             ;+1 TO DATA OUT
5057  022510                          8$:
5058  022510  032737  000001  001202  10$:    BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
5059  022516  001450                          BEQ     11$
5060  022520  104420  000000                  READ    ,SEC0           ;READ THE DATA REGISTER
5061  022524  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5062  022530  005337  023174                  DEC     19$             ;-1 TO DATA (CHR. REC.)
5063  022534  013737  022502  001204          MOV     65$,$TMP1       ;GET EXPECTED
5064  022542  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR CORRECT DATA
5065  022550  001403                          BEQ     30$             ;DATA OK BRANCH
5066  022552  104027                          ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
5067  022554  000137  022226                  JMP     64$             ;LOOP ON ERROR
5068  022560                          30$:
5069  022560  104420  000001                  READ    ,SEC1           ;READ REGISTER
5070  022564  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5071  022570  042737  177577  001202          BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
5072  022576  032737  000200  001202          BIT     #ERRCK,$TMP0            ;TEST ERRCK
5073  022604  001414                          BEQ     31$                     ;RESET SKIP ERROR
5074  022606  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5075  022614  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5076  022622  042737  000200  001206          BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
5077  022630  104031                          ERROR   25.             ;ERROR CHECK SET
5078  022632  000137  022226                  JMP     64$             ;LOOP ON ERROR
5079  022636                          31$:
5080  022636  000406                          BR      12$
5081  022640  105227  000000          11$:    INCB    #0
5082  022644  001302                          BNE     7$
5083  022646  104042                          ERROR   34.             ;DEVICE IS HUNG
5084  022650  000137  022226                  JMP     64$             ;LOOP ON ERROR
5085  022654  104416  000003  000002  12$:    WRITE   ,SEC3,TEOM              ;END THE MESSAGE
5086  022662  104421  000002          13$:    STEP    ,2              ;STEP THE CLOCKONCE
5087  022666  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
5088  022672  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5089  022676  032737  000010  001202          BIT     #SF,$TMP0       ;TEST FOR SF AFTER TEOM SENT
5090  022704  001406                          BEQ     14$             ;NO BRANCH
5091  022706  105227  000000          100$:   INCB    #0
5092  022712  001003                          BNE     14$
5093  022714  104044                          ERROR   36.             ;S/F CAME UP BEFORE LAST DATA BYTE RECEIVED
5094  022716  000137  022226                  JMP     64$             ;LOOP ON TEST
5095  022722  032737  000001  001202  14$:    BIT     #RDA,$TMP0      ;TEST FOR DATA
5096  022730  001006                          BNE     120$
5097  022732  105227  000000          110$:   INCB    #0
5098  022736  001351                          BNE     13$
```

```
5099  022740  104042                        ERROR   34.             ;DEVICE IS HUNG
5100  022742  000137  022226                JMP     64$             ;LOOP ON ERROR
5101  022746                        120$:
5102  022746  104420  000000                READ    ,SEC0           ;READ THE DATA
5103  022752  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5104  022756  013737  022502  001204        MOV     65$,$TMP1       ;GET EXPECTED DATA
5105  022764  023737  001202  001204        CMP     $TMP0,$TMP1     ;TEST DATA
5106  022772  001403                        BEQ     40$             ;OK BRANCH
5107  022774  104027                        ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
5108  022776  000137  022226                JMP     64$             ;LOOP ON ERROR
5109  023002                        40$:
5110  023002  005337  023174                DEC     19$             ;-1 TO CHR COUNT
5111  023006                        41$:
5112  023006  104421  000002        53$:    STEP    ,2              ;STEP THE CLOCK ONE TIME
5113  023012  104420  000013                READ    ,PRI13          ;READ THE STATUS REGISTER
5114  023016  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5115  023022  032737  000001  001202        BIT     #RDA,$TMP0      ;TEST FOR DATA
5116  023030  001407                        BEQ     70$
5117  023032  104420  000000                READ    ,SEC0           ;READ THE DATA
5118  023036  012637  001204                MOV     (SP)+,$TMP1     ;;POP STACK INTO $TMP1
5119  023042  005337  023174                DEC     19$             ;-1 TO DATA CHRS.
5120  023046  001406                        BEQ     15$             ;B=ALL CHRS REC.
5121  023050                        70$:
5122  023050  105227  000000        52$:    INCB    #0
5123  023054  001354                        BNE     53$
5124  023056  104026                        ERROR   22.             ;S/F DID NOT SET AFTER TEOM SET
5125  023060  000137  022226                JMP     64$             ;LOOP ON ERROR
5126  023064                        54$:
5127  023064                        15$:
5128  023064  104420  000001                READ    ,SEC1           ;READ REGISTER
5129  023070  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5130  023074  042737  177577  001202        BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
5131  023102  032737  000200  001202        BIT     #ERRCK,$TMP0            ;TEST ERRCK
5132  023110  001014                        BNE     17$                     ;SET SKIP ERROR
5133  023112  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5134  023120  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5135  023126  052737  000200  001206        BIS     #ERRCK,$TMP2            ;LOAD EXPECTED
5136  023134  104030                        ERROR   24.
5137  023136  000137  022226                JMP     64$
5138  023142  005237  022502        17$:    INC     65$
5139  023146  022737  000026  022502        CMP     #26,65$
5140  023154  001772                        BEQ     17$
5141  023156  022737  000177  022502        CMP     #177,65$                ;TEST IF ALL CHARACTER'S BEEN TRANSMITTED
5142  023164  001402                        BEQ     18$
5143  023166  000137  022226                JMP     64$
5144  023172                        18$:
5145  023172  000001                        BR      TST66           ;;
5146  023174  000000                19$:    0                       ;CHR. COUNTER
5147
5148
5149                                        ;****************************** TEST 66 ******************************
5150                                        ;*ERROR CHECK TEST FOR PROTOCOL = CCP
5151                                        ;*ERROR CHECK X Y Z = CCITT-0
5152                                        ;;*************************************************************************
5153
5154                                        ;  TEST 66
```

```
                                        ;-------------
                                        ;;*****************************************************************
023176 000004                   TST66:  SCOPE
023200 012737 000066 001102             MOV     #66,$TSTNM              ; LOAD THE NO. OF THIS TEST
023206 012737 000066 001262             MOV     #66,TSTNUM              ;# FOR TYPE OUT
023214 012737 000001 023504             MOV     #1,65$          ;LOAD CHARACTER BUFFER
023222 012737 023230 001106             MOV     #64$,$LPADR     ;SET RETURN ADDRESS TO 64$
023230 104414                   64$:    MSTCLR          ;ISSUE A MASTER CLEAR
023232 005037 024176                    CLR     19$
023236 104416 000007 000000             WRITE   ,SEC7,0         ;SET CHARACTER LENGTH TO 8 BITS/CHAR
023244 104416 000005 000141             WRITE   ,SEC5,CCP!SSL!CCITT-0   ;LOAD MODE CONTROL
023252 104416 000004 000026             WRITE   ,SEC4,SYN       ;LOAD SYN CHARACTER
023260 005037 023712                    CLR     100$+2
023264 005037 023736                    CLR     110$+2
023270 005037 024054                    CLR     52$+2
023274 005037 023412                    CLR     50$+2
023300 005037 023644                    CLR     11$+2
023304 104416 000014 000321             WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
                                                                ;       TENA
                                                                ;       RENA
                                                                ;       MAINTENANCE BITS: MB ,MC
023312 104416 000003 000001             WRITE   ,SEC3,TSOM      ;SET TSOM
023320 104421 000002            2$:     STEP    ,2              ;STEP THE CLOCK ONCE
023324 104420 000013                    READ    ,PRI13          ;READ THE STATUS REGISTER
023330 012637 001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
023334 032737 000020 001202             BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
023342 001766                           BEQ     2$              ;NO CONTINUE STEPPING THE CLOCK
023344 013737 023504 023356             MOV     65$,.+12
023352 104416 000002 000000             WRITE   ,SEC2,0 ;LOAD A CHARACTER
023360 005237 024176                    INC     19$
023364 104421 000002            3$:     STEP    ,2              ;STEP THE CLOCK ONCE
023370 104420 000013                    READ    ,PRI13          ;READ THE STATUS REGISTER
023374 012637 001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
023400 032737 000010 001202             BIT     #SF,$TMP0       ;TEST FOR S/F
023406 001017                           BNE     24$
023410 105227 000000            50$:    INCB    #0
023414 001363                           BNE     3$
023416 013737 001202 001204             MOV     $TMP0,$TMP1
023424 013737 001202 001206             MOV     $TMP0,$TMP2
023432 052737 000010 001206             BIS     #SF,$TMP2
023440 104017                           ERROR   15.             ;S/F DID NOT SET AFTER SYNC SENT
023442 000137 023230                    JMP     64$             ;LOOP ON ERROR
023446                          24$:
023446 104416 000003 000000             WRITE   ,SEC3,0         ;RESET TSOM
023454 104421 000002            7$:     STEP    ,2              ;STEP THE CLOCK ONCE
023460 104420 000013                    READ    ,PRI13          ;READ THE STATUS REGISTER
023464 012637 001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
023470 032737 000020 001202             BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
023476 001405                           BEQ     8$
023500 104416 000002                    WRITE   ,SEC2           ;LOAD A DATA CHARACTER
023504 000000                   65$:    .WORD   0
023506 005237 024176                    INC     19$             ;+1 TO DATA OUT
023512                          8$:
023512 032737 000001 001202     10$:    BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
023520 001450                           BEQ     11$
023522 104420 000000                    READ    ,SEC0           ;READ THE DATA REGISTER
```

```
5211  023526  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5212  023532  005337  024176                    DEC     19$             ;-1 TO DATA (CHR. REC.)
5213  023536  013737  023504  001204           MOV     65$,$TMP1       ;GET EXPECTED
5214  023544  023737  001202  001204           CMP     $TMP0,$TMP1     ;TEST FOR CORRECT DATA
5215  023552  001403                            BEQ     30$             ;DATA OK BRANCH
5216  023554  104027                            ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
5217  023556  000137  023230                    JMP     64$             ;LOOP ON ERROR
5218  023562                          30$:
5219  023562  104420  000001                    READ    ,SEC1           ;READ REGISTER
5220  023566  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5221  023572  042737  177577  001202            BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
5222  023600  032737  000200  001202            BIT     #ERRCK,$TMP0            ;TEST ERRCK
5223  023606  001414                            BEQ     31$                     ;RESET SKIP ERROR
5224  023610  013737  001202  001204           MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5225  023616  013737  001204  001206           MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5226  023624  042737  000200  001206           BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
5227  023632  104031                            ERROR   25.             ;ERROR CHECK SET
5228  023634  000137  023230                    JMP     64$             ;LOOP ON ERROR
5229  023640                          31$:
5230  023640  000406                            BR      12$
5231  023642  105227  000000            11$:    INCB    #0
5232  023646  001302                            BNE     7$
5233  023650  104042                            ERROR   34.             ;DEVICE IS HUNG
5234  023652  000137  023230                    JMP     64$             ;LOOP ON ERROR
5235  023656  104416  000003  000002    12$:    WRITE   ,SEC3,TEOM              ;END THE MESSAGE
5236  023664  104421  000002            13$:    STEP    ,2              ;STEP THE CLOCKONCE
5237  023670  104420  000013                    READ    ,PRI13          ;READ THE STATUS REGISTER
5238  023674  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5239  023700  032737  000010  001202            BIT     #SF,$TMP0       ;TEST FOR SF AFTER TEOM SENT
5240  023706  001406                            BEQ     14$             ;NO BRANCH
5241  023710  105227  000000            100$:   INCB    #0
5242  023714  001003                            BNE     14$
5243  023716  104044                            ERROR   36.             ;S/F CAME UP BEFORE LAST DATA BYTE RECEIVED
5244  023720  000137  023230                    JMP     64$             ;LOOP ON TEST
5245  023724  032737  000001  001202    14$:    BIT     #RDA,$TMP0      ;TEST FOR DATA
5246  023732  001006                            BNE     120$
5247  023734  105227  000000            110$:   INCB    #0
5248  023740  001351                            BNE     13$
5249  023742  104042                            ERROR   34.             ;DEVICE IS HUNG
5250  023744  000137  023230                    JMP     64$             ;LOOP ON ERROR
5251  023750                          120$:
5252  023750  104420  000000                    READ    ,SEC0           ;READ THE DATA
5253  023754  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5254  023760  013737  023504  001204           MOV     65$,$TMP1       ;GET EXPECTED DATA
5255  023766  023737  001202  001204           CMP     $TMP0,$TMP1     ;TEST DATA
5256  023774  001403                            BEQ     40$             ;OK BRANCH
5257  023776  104027                            ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
5258  024000  000137  023230                    JMP     64$             ;LOOP ON ERROR
5259  024004                          40$:
5260  024004  005337  024176                    DEC     19$             ;-1 TO CHR COUNT
5261  024010                          41$:
5262  024010  104421  000002            53$:   STEP    ,2              ;STEP THE CLOCK ONE TIME
5263  024014  104420  000013                    READ    ,PRI13          ;READ THE STATUS REGISTER
5264  024020  012637  001202                    MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5265  024024  032737  000001  001202            BIT     #RDA,$TMP0      ;TEST FOR DATA
5266  024032  001407                            BEQ     70$
```

```
                                                READ    ,SEC0           ;READ THE DATA
                                                MOV     (SP)+,$TMP1     ;;POP STACK INTO $TMP1
                                                DEC     19$             ;-1 TO DATA CHRS.
                                                BEQ     15$             ;B=ALL CHRS REC.
                                        70$:
                                        52$:    INCB    #0
                                                BNE     53$
                                                ERROR   22.             ;S/F DID NOT SET AFTER TEOM SET
                                                JMP     64$             ;LOOP ON ERROR
                                        54$:
                                        15$:
                                                READ    ,SEC1           ;READ REGISTER
                                                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
                                                BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
                                                BIT     #ERRCK,$TMP0            ;TEST ERRCK
                                                BNE     17$                     ;SET SKIP ERROR
                                                MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
                                                MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
                                                BIS     #ERRCK,$TMP2            ;LOAD EXPECTED
                                                ERROR   24.
                                                JMP     64$
                                        17$:    INC     65$
                                                CMP     #26,65$
                                                BEQ     17$
                                                CMP     #177,65$                ;TEST IF ALL CHARACTER'S BEEN TRANSMITTED
                                                BEQ     18$
                                                JMP     64$
                                        18$:
                                                BR      TST67           ;;
                                        19$:    0                       ;CHR. COUNTER


                                                ;***************************** TEST 67 *****************************
                                                ;*ERROR CHECK TEST FOR PROTOCOL = CCP
                                                ;*ERROR CHECK X Y Z = CRC16
                                                ;;*************************************************************************

                                                ;  TEST 67
                                                ;-------------
                                        ;;*************************************************************************
                                        TST67:  SCOPE
                                                MOV     #67,$TSTNM              ; LOAD THE NO. OF THIS TEST
                                                MOV     #67,TSTNUM              ;# FOR TYPE OUT
                                                MOV     #1,65$          ;LOAD CHARACTER BUFFER
                                                MOV     #64$,$LPADR     ;SET RETURN ADDRESS TO 64$
                                        64$:    MSTCLR          ;ISSUE A MASTER CLEAR
                                                CLR     19$
                                                WRITE   ,SEC7,0         ;SET CHARACTER LENGTH TO 8 BITS/CHAR
                                                WRITE   ,SEC5,CCP!SSL!CRC16     ;LOAD MODE CONTROL
                                                WRITE   ,SEC4,SYN       ;LOAD SYN CHARACTER
                                                CLR     100$+2
                                                CLR     110$+2
                                                CLR     52$+2
                                                CLR     50$+2
                                                CLR     11$+2
                                                WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
```

```
5323                                                    ;       TENA
5324                                                    ;       RENA
5325                                                    ;       MAINTENANCE BITS: MB ,MC
5326  024314  104416  000003  000001          WRITE   ,SEC3,TSOM      ;SET TSOM
5327  024322  104421  000002          2$:     STEP    ,2              ;STEP THE CLOCK ONCE
5328  024326  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
5329  024332  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5330  024336  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
5331  024344  001766                          BEQ     2$              ;NO CONTINUE STEPPING THE CLOCK
5332  024346  013737  024506  024360          MOV     65$,,+12
5333  024354  104416  000002  000000          WRITE   ,SEC2,0 ;LOAD A CHARACTER
5334  024362  005237  025200                  INC     19$
5335  024366  104421  000002          3$:     STEP    ,2              ;STEP THE CLOCK ONCE
5336  024372  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
5337  024376  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5338  024402  032737  000010  001202          BIT     #SF,$TMP0       ;TEST FOR S/F
5339  024410  001017                          BNE     24$
5340  024412  105227  000000          50$:    INCB    #0
5341  024416  001363                          BNE     3$
5342  024420  013737  001202  001204          MOV     $TMP0,$TMP1
5343  024426  013737  001202  001206          MOV     $TMP0,$TMP2
5344  024434  052737  000010  001206          BIS     #SF,$TMP2
5345  024442  104017                          ERROR   15.             ;S/F DID NOT SET AFTER SYNC SENT
5346  024444  000137  024232                  JMP     64$             ;LOOP ON ERROR
5347  024450                          24$:
5348  024450  104416  000003  000000          WRITE   ,SEC3,0         ;RESET TSOM
5349  024456  104421  000002          7$:     STEP    ,2              ;STEP THE CLOCK ONCE
5350  024462  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
5351  024466  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5352  024472  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
5353  024500  001405                          BEQ     8$
5354  024502  104416  000002                  WRITE   ,SEC2           ;LOAD A DATA CHARACTER
5355  024506  000000                  65$:    .WORD   0
5356  024510  005237  025200                  INC     19$             ;+1 TO DATA OUT
5357  024514                          8$:
5358  024514  032737  000001  001202  10$:    BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
5359  024522  001450                          BEQ     11$
5360  024524  104420  000000                  READ    ,SEC0           ;READ THE DATA REGISTER
5361  024530  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5362  024534  005337  025200                  DEC     19$             ;-1 TO DATA (CHR. REC.)
5363  024540  013737  024506  001204          MOV     65$,$TMP1       ;GET EXPECTED
5364  024546  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR CORRECT DATA
5365  024554  001403                          BEQ     30$             ;DATA OK BRANCH
5366  024556  104027                          ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
5367  024560  000137  024232                  JMP     64$             ;LOOP ON ERROR
5368  024564                          30$:
5369  024564  104420  000001                  READ    ,SEC1           ;READ REGISTER
5370  024570  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5371  024574  042737  177577  001202          BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
5372  024602  032737  000200  001202          BIT     #ERRCK,$TMP0            ;TEST ERRCK
5373  024610  001414                          BEQ     31$                     ;RESET SKIP ERROR
5374  024612  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5375  024620  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5376  024626  042737  000200  001206          BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
5377  024634  104031                          ERROR   25.             ;ERROR CHECK SET
5378  024636  000137  024232                  JMP     64$             ;LOOP ON ERROR
```

```
5379  024642                          31$:
5380  024642  000406                        BR      12$
5381  024644  105227  000000          11$:  INCB    #0
5382  024650  001302                        BNE     7$
5383  024652  104042                        ERROR   34.             ;DEVICE IS HUNG
5384  024654  000137  024232                JMP     64$             ;LOOP ON ERROR
5385  024660  104416  000003  000002  12$:  WRITE   ,SEC3,TEOM              ;END THE MESSAGE
5386  024666  104421  000002          13$:  STEP    ,2              ;STEP THE CLOCKONCE
5387  024672  104420  000013                READ    ,PRI13          ;READ THE STATUS REGISTER
5388  024676  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5389  024702  032737  000010  001202        BIT     #SF,$TMP0       ;TEST FOR SF AFTER TEOM SENT
5390  024710  001406                        BEQ     14$             ;NO BRANCH
5391  024712  105227  000000          100$: INCB    #0
5392  024716  001003                        BNE     14$
5393  024720  104044                        ERROR   36.             ;S/F CAME UP BEFORE LAST DATA BYTE RECEIVED
5394  024722  000137  024232                JMP     64$             ;LOOP ON TEST
5395  024726  032737  000001  001202  14$:  BIT     #RDA,$TMP0      ;TEST FOR DATA
5396  024734  001006                        BNE     120$
5397  024736  105227  000000          110$: INCB    #0
5398  024742  001351                        BNE     13$
5399  024744  104042                        ERROR   34.             ;DEVICE IS HUNG
5400  024746  000137  024232                JMP     64$             ;LOOP ON ERROR
5401  024752                          120$:
5402  024752  104420  000000                READ    ,SEC0           ;READ THE DATA
5403  024756  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5404  024762  013737  024506  001204        MOV     65$,$TMP1       ;GET EXPECTED DATA
5405  024770  023737  001202  001204        CMP     $TMP0,$TMP1     ;TEST DATA
5406  024776  001403                        BEQ     40$             ;OK BRANCH
5407  025000  104027                        ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
5408  025002  000137  024232                JMP     64$             ;LOOP ON ERROR
5409  025006                          40$:
5410  025006  005337  025200                DEC     19$             ;-1 TO CHR COUNT
5411  025012                          41$:
5412  025012  104421  000002          53$:  STEP    ,2              ;STEP THE CLOCK ONE TIME
5413  025016  104420  000013                READ    ,PRI13          ;READ THE STATUS REGISTER
5414  025022  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5415  025026  032737  000001  001202        BIT     #RDA,$TMP0      ;TEST FOR DATA
5416  025034  001407                        BEQ     70$
5417  025036  104420  000000                READ    ,SEC0           ;READ THE DATA
5418  025042  012637  001204                MOV     (SP)+,$TMP1     ;;POP STACK INTO $TMP1
5419  025046  005337  025200                DEC     19$             ;-1 TO DATA CHRS.
5420  025052  001406                        BEQ     15$             ;B=ALL CHRS REC.
5421  025054                          70$:
5422  025054  105227  000000          52$:  INCB    #0
5423  025060  001354                        BNE     53$
5424  025062  104026                        ERROR   22.             ;S/F DID NOT SET AFTER TEOM SET
5425  025064  000137  024232                JMP     64$             ;LOOP ON ERROR
5426  025070                          54$:
5427  025070                          15$:
5428  025070  104420  000001                READ    ,SEC1           ;READ REGISTER
5429  025074  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5430  025100  042737  177577  001202        BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
5431  025106  032737  000200  001202        BIT     #ERRCK,$TMP0            ;TEST ERRCK
5432  025114  001014                        BNE     17$                     ;SET SKIP ERROR
5433  025116  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5434  025124  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
```

```
5435  025132  052737  000200  001206          BIS     #ERRCK,$TMP2            ;LOAD EXPECTED
5436  025140  104030                          ERROR   24.
5437  025142  000137  024232                  JMP     64$
5438  025146  005237  024506          17$:    INC     65$
5439  025152  022737  000026  024506          CMP     #26,65$
5440  025160  001772                          BEQ     17$
5441  025162  022737  000177  024506          CMP     #177,65$                ;TEST IF ALL CHARACTER'S BEEN TRANSMITTED
5442  025170  001402                          BEQ     18$
5443  025172  000137  024232                  JMP     64$
5444  025176                          18$:
5445  025176  000401                          BR      TST70                   ;;
5446  025200  000000                  19$:    0                               ;CHR. COUNTER
5447
5448
5449                                          ;****************************** TEST 70 ******************************
5450                                          ;*ERROR CHECK TEST FOR PROTOCOL = CCP
5451                                          ;*ERROR CKECK X Y Z = ODD PARITY
5452                                          ;;*************************************************************************
5453
5454                                          ;  TEST 70
5455                                          ;-------------
5456                                  ;;**************************************************************
5457  025202  000004                  TST70:  SCOPE
5458  025204  012737  000070  001102          MOV     #70,$TSTNM              ; LOAD THE NO. OF THIS TEST
5459  025212  012737  000070  001262          MOV     #70,TSTNUM              ;# FOR TYPE OUT
5460  025220  012737  000001  025500          MOV     #1,65$          ;LOAD CHARACTER BUFFER
5461  025226  012737  025234  001106          MOV     #64$,$LPADR     ;SET RETURN ADDRESS TO 64$
5462  025234  104414                  64$:    MSTCLR          ;ISSUE A MASTER CLEAR
5463  025236  005037  025754                  CLR     19$
5464  025242  104416  000007  000347          WRITE   ,SEC7,347       ;SET CHARACTER LENGTH TO 7 BITS/CHAR
5465  025250  104416  000005  000144          WRITE   ,SEC5,CCP!SSL!ODDP      ;LOAD MODE CONTROL
5466  025256  104416  000004  000026          WRITE   ,SEC4,SYN       ;LOAD SYN CHARACTER
5467  025264  005037  025640                  CLR     52$+2
5468  025270  005037  025406                  CLR     50$+2
5469  025274  005037  025562                  CLR     11$+2
5470  025300  104416  000014  000321          WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
5471                                                                  ;       TENA
5472                                                                  ;       RENA
5473                                                                  ;       MAINTENANCE BITS: MB ,MC
5474  025306  104416  000003  000001          WRITE   ,SEC3,TSOM      ;SET TSOM
5475  025314  104421  000002          2$:     STEP    ,2              ;STEP THE CLOCK ONCE
5476  025320  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
5477  025324  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5478  025330  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
5479  025336  001766                          BEQ     2$              ;NO CONTINUE STEPPING THE CLOCK
5480  025340  013737  025500  025352          MOV     65$,,+12
5481  025346  104416  000002  000000          WRITE   ,SEC2,0 ;LOAD A CHARACTER
5482  025354  005237  025754                  INC     19$
5483  025360  104421  000002          3$:     STEP    ,2              ;STEP THE CLOCK ONCE
5484  025364  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
5485  025370  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5486  025374  032737  000010  001202          BIT     #SF,$TMP0       ;TEST FOR S/F
5487  025402  001017                          BNE     24$
5488  025404  105227  000000          50$:    INCB    #0
5489  025410  001363                          BNE     3$
5490  025412  013737  001202  001204          MOV     $TMP0,$TMP1
```

```
5491  025420  013737  001202  001206          MOV     $TMP0,$TMP2
5492  025426  052737  000010  001206          BIS     #SF,$TMP2
5493  025434  104017                          ERROR   15.             ;S/F DID NOT SET AFTER SYNC SENT
5494  025436  000137  025234                  JMP     64$             ;LOOP ON ERROR
5495  025442                          24$:
5496  025442  104416  000003  000000          WRITE   ,SEC3,0         ;RESET TSOM
5497  025450  104421  000002          7$:     STEP    ,2              ;STEP THE CLOCK ONCE
5498  025454  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
5499  025460  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5500  025464  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
5501  025472  001405                          BEQ     8$
5502  025474  104416  000002                  WRITE   ,SEC2           ;LOAD A DATA CHARACTER
5503  025500  000000                  65$:    .WORD   0
5504  025502  005237  025754                  INC     19$             ;+1 TO DATA OUT
5505  025506                          8$:
5506  025506  032737  000001  001202  10$:    BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
5507  025514  001421                          BEQ     11$
5508  025516  104420  000000                  READ    ,SEC0           ;READ THE DATA REGISTER
5509  025522  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5510  025526  005337  025754                  DEC     19$             ;-1 TO DATA (CHR. REC.)
5511  025532  013737  025500  001204          MOV     65$,$TMP1       ;GET EXPECTED
5512  025540  023737  001202  001204          CMP     $TMP0,$TMP1     ;TEST FOR CORRECT DATA
5513  025546  001403                          BEQ     30$             ;DATA OK BRANCH
5514  025550  104027                          ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
5515  025552  000137  025234                  JMP     64$             ;LOOP ON ERROR
5516  025556                          30$:
5517  025556  000406                          BR      41$
5518  025560  105227  000000          11$:    INCB    #0
5519  025564  001331                          BNE     7$
5520  025566  104042                          ERROR   34.             ;DEVICE IS HUNG
5521  025570  000137  025234                  JMP     64$             ;LOOP ON ERROR
5522  025574                          41$:
5523  025574  104421  000002          53$:    STEP    ,2              ;STEP THE CLOCK ONE TIME
5524  025600  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
5525  025604  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5526  025610  032737  000001  001202          BIT     #RDA,$TMP0      ;TEST FOR DATA
5527  025616  001407                          BEQ     70$
5528  025620  104420  000000                  READ    ,SEC0           ;READ THE DATA
5529  025624  012637  001204                  MOV     (SP)+,$TMP1     ;;POP STACK INTO $TMP1
5530  025630  005337  025754                  DEC     19$             ;-1 TO DATA CHRS.
5531  025634  001406                          BEQ     15$             ;B=ALL CHRS REC.
5532  025636                          70$:
5533  025636  105227  000000          52$:    INCB    #0
5534  025642  001354                          BNE     53$
5535  025644  104026                          ERROR   22.             ;S/F DID NOT SET AFTER TEOM SET
5536  025646  000137  025234                  JMP     64$             ;LOOP ON ERROR
5537  025652                          54$:
5538  025652                          15$:
5539  025652  104420  000001                  READ    ,SEC1           ;READ REGISTER
5540  025656  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5541  025662  042737  177577  001202          BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
5542  025670  032737  000200  001202          BIT     #ERRCK,$TMP0            ;TEST ERRCK
5543  025676  001411                          BEQ     17$                     ;RESET SKIP ERROR
5544  025700  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5545  025706  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5546  025714  042737  000200  001206          BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
```

```
5547  025722  005237  025500          17$:    INC     65$
5548  025726  022737  000026  025500          CMP     #26,65$
5549  025734  001772                          BEQ     17$
5550  025736  022737  000177  025500          CMP     #177,65$                ;TEST IF ALL CHARACTER'S BEEN TRANSMITTED
5551  025744  001402                          BEQ     18$
5552  025746  000137  025234                  JMP     64$
5553  025752                          18$:
5554  025752  000401                          BR      TST71                   ;;
5555  025754  000000                  19$:    0                               ;CHR. COUNTER
5556
5557
5558                                          ;***************************** TEST 71 *****************************
5559                                          ;*ERROR CHECK TEST FOR PROTOCOL = CCP
5560                                          ;*ERROR CHECK X Y Z = EVEN PARITY
5561                                          ;;*************************************************************************
5562
5563                                          ;  TEST 71
5564                                          ;-------------
5565                                  ;;**************************************************************************
5566  025756  000004                  TST71:  SCOPE
5567  025760  012737  000071  001102          MOV     #71,$TSTNM              ; LOAD THE NO. OF THIS TEST
5568  025766  012737  000071  001262          MOV     #71,TSTNUM              ;# FOR TYPE OUT
5569  025774  012737  000001  026254          MOV     #1,65$                  ;LOAD CHARACTER BUFFER
5570  026002  012737  026010  001106          MOV     #64$,$LPADR             ;SET RETURN ADDRESS TO 64$
5571  026010  104414                  64$:    MSTCLR                  ;ISSUE A MASTER CLEAR
5572  026012  005037  026536                  CLR     19$
5573  026016  104416  000007  000347          WRITE   ,SEC7,347               ;SET CHARACTER LENGTH TO 7 BITS/CHAR
5574  026024  104416  000005  000145          WRITE   ,SEC5,CCP!SSL!EVENP     ;LOAD MODE CONTROL
5575  026032  104416  000004  000026          WRITE   ,SEC4,SYN               ;LOAD SYN CHARACTER
5576  026040  005037  026422                  CLR     52$+2
5577  026044  005037  026162                  CLR     50$+2
5578  026050  005037  026344                  CLR     11$+2
5579  026054  104416  000014  000321          WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
5580                                                                          ;       TENA
5581                                                                          ;       RENA
5582                                                                          ;       MAINTENANCE BITS: MB ,MC
5583  026062  104416  000003  000001          WRITE   ,SEC3,TSOM              ;SET TSOM
5584  026070  104421  000002          2$:     STEP    ,2                      ;STEP THE CLOCK ONCE
5585  026074  104420  000013                  READ    ,PRI13                  ;READ THE STATUS REGISTER
5586  026100  012637  001202                  MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
5587  026104  032737  000020  001202          BIT     #TBMT,$TMP0             ;TEST FOR BUFFER EMPTY
5588  026112  001766                          BEQ     2$                      ;NO CONTINUE STEPPING THE CLOCK
5589  026114  013737  026254  026126          MOV     65$,,+12
5590  026122  104416  000002  000000          WRITE   ,SEC2,0 ;LOAD A CHARACTER
5591  026130  005237  026536                  INC     19$
5592  026134  104421  000002          3$:     STEP    ,2                      ;STEP THE CLOCK ONCE
5593  026140  104420  000013                  READ    ,PRI13                  ;READ THE STATUS REGISTER
5594  026144  012637  001202                  MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
5595  026150  032737  000010  001202          BIT     #SF,$TMP0               ;TEST FOR S/F
5596  026156  001017                          BNE     24$
5597  026160  105227  000000          50$:    INCB    #0
5598  026164  001363                          BNE     3$
5599  026166  013737  001202  001204          MOV     $TMP0,$TMP1
5600  026174  013737  001202  001206          MOV     $TMP0,$TMP2
5601  026202  052737  000010  001206          BIS     #SF,$TMP2
5602  026210  104017                          ERROR   15.                     ;S/F DID NOT SET AFTER SYNC SENT
```

```
5603  026212  000137  026010              JMP     64$             ;LOOP ON ERROR
5604  026216                      24$:
5605  026216  104416  000003  000000      WRITE   ,SEC3,0         ;RESET TSOM
5606  026224  104421  000002      7$:     STEP    ,2              ;STEP THE CLOCK ONCE
5607  026230  104420  000013              READ    ,PRI13          ;READ THE STATUS REGISTER
5608  026234  012637  001202              MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5609  026240  032737  000020  001202      BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
5610  026246  001405                      BEQ     8$
5611  026250  104416  000002              WRITE   ,SEC2           ;LOAD A DATA CHARACTER
5612  026254  000000              65$:    .WORD   0
5613  026256  005237  026536              INC     19$             ;+1 TO DATA OUT
5614  026262                      8$:
5615  026262  032737  000001  001202  10$:  BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
5616  026270  001424                      BEQ     11$
5617  026272  104420  000000              READ    ,SEC0           ;READ THE DATA REGISTER
5618  026276  012637  001202              MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5619  026302  005337  026536              DEC     19$             ;-1 TO DATA (CHR. REC.)
5620  026306  042737  000200  001202      BIC     #BIT7,$TMP0     ;MASK THE PARITY BIT
5621  026314  013737  026254  001204      MOV     65$,$TMP1       ;GET EXPECTED
5622  026322  023737  001202  001204      CMP     $TMP0,$TMP1     ;TEST FOR CORRECT DATA
5623  026330  001403                      BEQ     30$             ;DATA OK BRANCH
5624  026332  104027                      ERROR   23.             ;DATA TRANSMITTED INCORRECTLY
5625  026334  000137  026010              JMP     64$             ;LOOP ON ERROR
5626  026340                      30$:
5627  026340  000406                      BR      41$
5628  026342  105227  000000      11$:    INCB    #0
5629  026346  001326                      BNE     7$
5630  026350  104042                      ERROR   34.             ;DEVICE IS HUNG
5631  026352  000137  026010              JMP     64$             ;LOOP ON ERROR
5632  026356                      41$:
5633  026356  104421  000002      53$:    STEP    ,2              ;STEP THE CLOCK ONE TIME
5634  026362  104420  000013              READ    ,PRI13          ;READ THE STATUS REGISTER
5635  026366  012637  001202              MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5636  026372  032737  000001  001202      BIT     #RDA,$TMP0      ;TEST FOR DATA
5637  026400  001407                      BEQ     70$
5638  026402  104420  000000              READ    ,SEC0           ;READ THE DATA
5639  026406  012637  001204              MOV     (SP)+,$TMP1     ;;POP STACK INTO $TMP1
5640  026412  005337  026536              DEC     19$             ;-1 TO DATA CHRS.
5641  026416  001406                      BEQ     15$             ;B=ALL CHRS REC.
5642  026420                      70$:
5643  026420  105227  000000      52$:    INCB    #0
5644  026424  001354                      BNE     53$
5645  026426  104026                      ERROR   22.             ;S/F DID NOT SET AFTER TEOM SET
5646  026430  000137  026010              JMP     64$             ;LOOP ON ERROR
5647  026434                      54$:
5648  026434                      15$:
5649  026434  104420  000001              READ    ,SEC1           ;READ REGISTER
5650  026440  012637  001202              MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5651  026444  042737  177577  001202      BIC     #^C<ERRCK>,$TMP0        ;CLEAR UNWANTED BITS
5652  026452  032737  000200  001202      BIT     #ERRCK,$TMP0            ;TEST ERRCK
5653  026460  001411                      BEQ     17$                     ;RESET SKIP ERROR
5654  026462  013737  001202  001204      MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5655  026470  013737  001204  001206      MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5656  026476  042737  000200  001206      BIC     #ERRCK,$TMP2            ;LOAD EXPECTED
5657  026504  005237  026254      17$:    INC     65$
5658  026510  022737  000026  026254      CMP     #26,65$
```

```
 5659  026516  001772                    BEQ     17$
 5660  026520  022737  000177  026254    CMP     #177,65$            ;TEST IF ALL CHARACTER'S BEEN TRANSMITTED
 5661  026526  001402                    BEQ     18$
 5662  026530  000137  026010            JMP     64$
 5663  026534                    18$:
 5664  026534  000401                    BR      TST72               ;;
 5665  026536  000000                    19$:    0                   ;CHR. COUNTER
```

```
                                .SBTTL  TRANSMIT UNDERRUN ERROR TEST


                                        ;****************************** TEST 72 ******************************
                                        ;*TRANSMITTER UNDERRUN TEST
                                        ;*FOR PROTOCOL = BOP
                                        ;*
                                        ;*THIS TEST WILL SYNC THE SEND RECEIVE LOGIC
                                        ;*TRANSMIT A ZERO DATA BYTE
                                        ;*READ AND DISCARD THE DATA SENT
                                        ;*STEP THE CLOCK 2 MORE CLOCK TICKS
                                        ;*TEST IF TERR SET TO INDICATE A UNDERRUN CONDITION
                                        ;*WITH THE IDLE MODE BIT RESET THE HANDLING OF A UNDERRUN
                                        ;*CODITION FOR PROTOCOL = BOP
                                        ;*AN ABORT WILL BE SENT AND THE INDICATION
                                        ;*WILL BE RABORT SET
                                        ;;*************************************************************************

                                        ;  TEST 72
                                        ;--------------
                                ;;*************************************************************************
026540  000004                  TST72:  SCOPE
026542  012737  000072  001102          MOV     #72,$TSTNM              ; LOAD THE NO. OF THIS TEST
026550  012737  000072  001262          MOV     #72,TSTNUM              ;# FOR TYPE OUT
026556  104414                  101$:   MSTCLR                  ;CLEAR THE KMC11
026560  005037  026734                  CLR     50$+2
026564  005037  027114                  CLR     52$+2
026570  005037  027034                  CLR     70$+2
026574  005037  027312                  CLR     61$+2
026600  104416  000007  000000          WRITE   ,SEC7,0         ;SET CHARACTER LENGTH TO 8 BITS
026606  104416  000005  000000          WRITE   ,SEC5,BOP       ;LOAD THE MODE CONTROL
026614  104416  000014  000321          WRITE   ,PRI14,TENA!RENA!MC!MB  ;ENABLE THE TRANSMITTER
                                                                ;AND RECEIVER PLUS MAINTENACE MC AND MB
026622  104416  000003  000001          WRITE   ,SEC3,TSOM      ;SET TSOM
026630  104421  000002          2$:     STEP    ,2              ;STEP THE CLOCK ONCE
026634  104420  000012                  READ    ,PRI12          ;READ THE STATUS REGISTER
026640  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
026644  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
026652  001766                  BEQ     2$                      ;NO CONTINUE STEPPING THE CLOCK
026654  104416  000002  000000          WRITE   ,SEC2,0         ;LOAD ADDRESS
026662  104416  000003  000000          WRITE   ,SEC3,0         ;RESET TSOM
026670  104421  000002          3$:     STEP    ,2              ;STEP THE CLOCK ONCE
026674  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
026700  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
026704  032737  000010  001202          BIT     #SF,$TMP0       ;TEST FOR S/F
026712  001026                  BNE     4$
026714  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
026722  001403                  BEQ     50$
026724  104416  000002  000101          WRITE   ,SEC2,'A        ;LOAD A DATA CHARACTER
026732  105227  000000          50$:    INCB    #0
026736  001354                  BNE     3$
026740  013737  001202  001204          MOV     $TMP0,$TMP1
026746  013737  001202  001206          MOV     $TMP0,$TMP2
026754  052737  000010  001206          BIS     #SF,$TMP2
026762  104016                  ERROR   14.                     ;S/F DID NOT SET AFTER FLAG SENT
026764  000137  026556                  JMP     101$            ;LOOP ON ERROR
```

```
5722  026770                          4$:
5723  026770  104421  000002          7$:     STEP    ,2              ;STEP THE CLOCK ONCE
5724  026774  104420  000013                  READ    ,PRI13          ;READ THE STATUS REGISTER
5725  027000  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5726  027004  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
5727  027012  001403                          BEQ     8$
5728  027014  104416  000002  000101          WRITE   ,SEC2,'A        ;LOAD A DATA CHARACTER
5729  027022  032737  000001  001202  8$:     BIT     #RDA,$TMP0      ;TEST FOR DATA
5730  027030  001006                          BNE     18$
5731  027032  1C5227  000000          70$:    INCB    #0
5732  027036  001354                          BNE     7$
5733  027040  104042                          ERROR   34.             ;DEVICE HUNG ERROR
5734  027042  000137  026556                  JMP     101$            ;LOOP ON ERROR
5735  027046  104421  000002          18$:    STEP    ,2
5736  027052  104420  000013                  READ    ,PRI13          ;TEST FOR TSA TO SET
5737  027056  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5738  027062  032737  000001  001202          BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
5739  027070  001404                          BEQ     60$
5740  027072  104420  000000                  READ    ,SEC0
5741  027076  012637  001212                  MOV     (SP)+,$TMP4     ;;POP STACK INTO $TMP4
5742  027102                          60$:
5743  027102  032737  000200  001202          BIT     #TSA,$TMP0
5744  027110  001006                          BNE     19$
5745  027112  105227  000000          52$:    INCB    #0
5746  027116  001353                          BNE     18$
5747  027120  104044                          ERROR   36.             ;TSA DID NOT SET ON A FORCE UNDERRUN ERROR
5748  027122  000137  026556                  JMP     101$            ;LOOP ON ERROR
5749  027126                          19$:
5750  027126  104420  000003                  READ    ,SEC3           ;READ REGISTER
5751  027132  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5752  027136  042737  177577  001202          BIC     #^C<TERR>,$TMP0 ;CLEAR UNWANTED BITS
5753  027144  032737  000200  001202          BIT     #TERR,$TMP0             ;TEST TERR
5754  027152  001014                          BNE     20$                     ;SET SKIP ERROR
5755  027154  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5756  027162  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5757  027170  052737  000200  001206          BIS     #TERR,$TMP2             ;LOAD EXPECTED
5758  027176  104032                          ERROR   26.             ;TERR DID NOT SET ON A FORCE UNDER RUN
5759  027200  000137  026556                  JMP     101$            ;LOOP ON ERROR
5760  027204  104421  000002          20$:    STEP    ,2              ;BEGIN LOOKING FO AN ABORT
5761  027210  104420  000013                  READ    ,PRI13
5762  027214  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5763  027220  032737  000001  001202          BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
5764  027226  001404                          BEQ     62$
5765  027230  104420  000000                  READ    ,SEC0
5766  027234  012637  001212                  MOV     (SP)+,$TMP4     ;;POP STACK INTO $TMP4
5767  027240                          62$:
5768  027240                          21$:
5769  027240  104420  000001                  READ    ,SEC1           ;READ REGISTER
5770  027244  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5771  027250  042737  177773  001202          BIC     #^C<RXAB>,$TMP0 ;CLEAR UNWANTED BITS
5772  027256  032737  000004  001202          BIT     #RXAB,$TMP0             ;TEST RXAB
5773  027264  001017                          BNE     22$                     ;SET SKIP ERROR
5774  027266  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5775  027274  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5776  027302  052737  000004  001206          BIS     #RXAB,$TMP2             ;LOAD EXPECTED
5777  027310  105227  000000          61$:    INCB    #0
```

```
5778  027314  001333                          BNE     20$
5779  027316  104033                          ERROR   27.             ;NO RECEIVE ABORT OCCURED ON
5780                                                                  ;A TRANSMIT UNDER RUN ERROR
5781  027320  000137  026556                  JMP     101$            ;LOOP ON ERROR
5782  027324                        22$:
5783  027324                        23$:
5784  027324  000400                          BR      TST73           ;;
5785
5786
5787                                          ;****************************** TEST 73 ******************************
5788                                          ;*TRANSMITTER UNDERRUN TEST
5789                                          ;*FOR PROTOCOL = CCP
5790                                          ;*
5791                                          ;*THIS TEST WILL SYNC THE SEND RECEIVE LOGIC
5792                                          ;*TRANSMIT A ZERO DATA BYTE
5793                                          ;*READ AND DISCARD THE DATA SENT
5794                                          ;*STEP THE CLOCK 2 MORE CLOCK TICKS
5795                                          ;*TEST IF TERR SET TO INDICATE A UNDERRUN CONDITION
5796                                          ;*WITH THE IDLE MODE BIT RESET THE HANDLING OF A UNDERRUN
5797                                          ;*CODITION FOR PROTOCOL = CCP
5798                                          ;*THE LINE WILL RESYNC AND THE INDICATION OF THIS
5799                                          ;*WILL BE S/F SET
5800                                          ;;*************************************************************************
5801
5802                                          ;  TEST 73
5803                                          ;-------------
5804                                  ;;**************************************************************************
5805  027326  000004                  TST73:  SCOPE
5806  027330  012737  000073  001102          MOV     #73,$TSTNM              ; LOAD THE NO. OF THIS TEST
5807  027336  012737  000073  001262          MOV     #73,TSTNUM              ;# FOR TYPE OUT
5808  027344  104414                  101$:   MSTCLR                  ;CLEAR THE KMC11
5809  027346  005037  027522                  CLR     50$+2
5810  027352  005037  027710                  CLR     52$+2
5811  027356  005037  027630                  CLR     70$+2
5812  027362  005037  030056                  CLR     53$+2
5813  027366  104416  000007  000000          WRITE   ,SEC7,0           ;SET CHARACTER LENGTH TO 8 BITS
5814  027374  104416  000005  000100          WRITE   ,SEC5,CCP         ;LOAD MODE CONTROL
5815  027402  104416  000004  000026          WRITE   ,SEC4,SYN
5816  027410  104416  000014  000321          WRITE   ,PRI14,TENA!RENA!MC!MB  ;ENABLE THE TRANSMITTER
5817                                                                  ;AND RECEIVER PLUS MAINTENACE MC AND MB
5818  027416  104416  000003  000001          WRITE   ,SEC3,TSOM        ;SET TSOM
5819  027424  104421  000002                  STEP    ,2              ;STEP THE CLOCK ONCE
5820  027430  104420  000012          2$:     READ    ,PRI12          ;READ THE STATUS REGISTER
5821  027434  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5822  027440  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
5823  027446  001766                          BEQ     2$              ;NO CONTINUE STEPPING THE CLOCK
5824  027450  104416  000002  000002          WRITE   ,SEC2,STX       ;LOAD A CHARACTER
5825  027456  104421  000002                  STEP    ,2              ;STEP THE CLOCK ONCE
5826  027462  104420  000013          3$:     READ    ,PRI13          ;READ THE STATUS REGISTER
5827  027466  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5828  027472  032737  000010  001202          BIT     #SF,$TMP0       ;TEST FOR S/F
5829  027500  001026                          BNE     4$
5830  027502  032737  000020  001202          BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
5831  027510  001403                          BEQ     50$
5832  027512  104416  000002  000101          WRITE   ,SEC2,'A        ;LOAD A DATA CHARACTER
5833  027520  105227  000000          50$:    INCB    #0
```

```
5834  027524  001354                        BNE     3$
5835  027526  013737  001202  001204        MOV     $TMP0,$TMP1
5836  027534  013737  001202  001206        MOV     $TMP0,$TMP2
5837  027542  052737  000010  001206        BIS     #SF,$TMP2
5838  027550  104017                        ERROR   15.              ;S/F DID NOT SET AFTER SYNC SENT
5839  027552  000137  027344                JMP     101$            ;LOOP ON ERROR
5840  027556                          4$:
5841  027556  104416  000003  000000        WRITE   ,SEC3,0          ;RESET TSOM
5842  027564  104421  000002          7$:   STEP    ,2              ;STEP THE CLOCK ONCE
5843  027570  104420  000013                READ    ,PRI13          ;READ THE STATUS REGISTER
5844  027574  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5845  027600  032737  000020  001202        BIT     #TBMT,$TMP0     ;TEST FOR BUFFER EMPTY
5846  027606  001403                        BEQ     8$
5847  027610  104416  000002  000101        WRITE   ,SEC2,'A        ;LOAD A DATA CHARACTER
5848  027616  032737  000001  001202  8$:   BIT     #RDA,$TMP0      ;TEST FOR DATA
5849  027624  001006                        BNE     18$
5850  027626  105227  000000          70$:  INCB    #0
5851  027632  001354                        BNE     7$
5852  027634  104042                        ERROR   34.             ;DEVICE HUNG ERROR
5853  027636  000137  027344                JMP     101$            ;LOOP ON ERROR
5854  027642  104421  000002          18$:  STEP    ,2
5855  027646  104420  000013                READ    ,PRI13          ;TEST FOR TSA TO SET
5856  027652  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5857  027656  032737  000001  001202        BIT     #RDA,$TMP0      ;TEST FOR DATA AVAILIBLE
5858  027664  001404                        BEQ     60$
5859  027666  104420  000000                READ    ,SEC0
5860  027672  012637  001212                MOV     (SP)+,$TMP4     ;;POP STACK INTO $TMP4
5861  027676                          60$:
5862  027676  032737  000200  001202        BIT     #TSA,$TMP0
5863  027704  001006                        BNE     19$
5864  027706  105227  000000          52$:  INCB    #0
5865  027712  001353                        BNE     18$
5866  027714  104044                        ERROR   36.             ;TSA DID NOT SET ON A FORCE UNDERRUN ERROR
5867  027716  000137  027344                JMP     101$            ;LOOP ON ERROR
5868  027722                          19$:
5869  027722  104420  000003                READ    ,SEC3           ;READ REGISTER
5870  027726  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5871  027732  042737  177577  001202        BIC     #^C<TERR>,$TMP0 ;CLEAR UNWANTED BITS
5872  027740  032737  000200  001202        BIT     #TERR,$TMP0             ;TEST TERR
5873  027746  001014                        BNE     20$                     ;SET SKIP ERROR
5874  027750  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5875  027756  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5876  027764  052737  000200  001206        BIS     #TERR,$TMP2             ;LOAD EXPECTED
5877  027772  104032                        ERROR   26.             ;TERR DID NOT SET ON A FORCE UNDER RUN
5878  027774  000137  027344                JMP     101$            ;LOOP ON ERROR
5879  030000  104421  000002          20$:  STEP    ,2              ;BEGIN LOOKING FO AN ABORT
5880  030004  104420  000013                READ    ,PRI13          ;READ REGISTER
5881  030010  012637  001202                MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
5882  030014  042737  177767  001202        BIC     #^C<SF>,$TMP0   ;CLEAR UNWANTED BITS
5883  030022  032737  000010  001202        BIT     #SF,$TMP0               ;TEST SF
5884  030030  001017                        BNE     23$                     ;SET SKIP ERROR
5885  030032  013737  001202  001204        MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5886  030040  013737  001204  001206        MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5887  030046  052737  000010  001206        BIS     #SF,$TMP2               ;LOAD EXPECTED
5888  030054  105227  000000          53$:  INCB    #0
5889  030060  001347                        BNE     20$
```

```
 5890  030062  104034                        ERROR   28.             ;LINE DID NOT RESYNC ON A FORCE
 5891                                                                  ;UNDERRUN ERROR WITH IDLE RESET
 5892  030064  000137  027344                JMP     101$            ;LOOP ON ERROR
 5893  030070                        23$:
 5894  030070  000400                        BR      TST74           ;;
```

```
                                .SBTTL  RECEIVER OVERRUN TEST

                                        ;****************************** TEST 74 ******************************
                                        ;*RECEIVER OVER RUN ERROR TEST FOR
                                        ;*PROTOCOL = CCP
                                        ;*THIS TEST WILL TRANSMIT DATA TO
                                        ;*THE RECEIVER IN MAINTENANCE MODE AND FAIL TO READ
                                        ;*THE SECOND DATA BIT TO FORCE THE OVERRUN CONDITION
                                        ;;************************************************************************

                                        ;  TEST 74
                                        ;--------------
                                ;;**************************************************************************
030072  000004                  TST74:  SCOPE
030074  012737  000074  001102          MOV     #74,$TSTNM              ; LOAD THE NO. OF THIS TEST
030102  012737  000074  001262          MOV     #74,TSTNUM              ;# FOR TYPE OUT
030110  104414                  101$:   MSTCLR                  ;CLEAR THE KMC11
030112  104416  000007  000000          WRITE   ,SEC7,0         ;SET CHARACTER LENGTH TO 8 BITS/CHAR
030120  104416  000005  000100          WRITE   ,SEC5,CCP       ;LOAD MODE CONTROL
030126  104416  000004  000026          WRITE   ,SEC4,SYN       ;LOAD SYN CHARACTER
030134  104416  000014  000321          WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
                                                                ;       TENA
                                                                ;       RENA
                                                                ;       MAINTENANCE BITS: MB, MC
030142  104416  000003  000001          WRITE   ,SEC3,TSOM      ;SET TSOM
030150  104421  000022                  STEP    ,9.*2
030154  104416  000003  000000          WRITE   ,SEC3,0         ;RESET TSOM
030162  104416  000002  000001          WRITE   ,SEC2,SOH       ;LOAD SOH INTO TDB
030170  104421  000040                  STEP    ,16.*2
030174  104416  000002  000101          WRITE   ,SEC2,101       ;LOAD DATA #1
030202  104421  000020                  STEP    ,8.*2           ;TRANSMIT DATA #1
030206  104416  000002  000101          WRITE   ,SEC2,101       ;LOAD DATA #2
030214  104421  000020                  STEP    ,8.*2           ;SEND DATA #2
030220  104416  000002  000101          WRITE   ,SEC2,101       ;LOAD DATA #3
030226  104421  000020                  STEP    ,8.*2           ;SEND DATA #3
                                ;AT THIS POINT THE FIRST DATA BIT SHOULD BE IN THE RECEIVER
                                ;BUFFER
030232  104420  000000                  READ    ,SEC0           ;READ THE DATA
030236  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
030242  104416  000002  000101          WRITE   ,SEC2,101       ;LOAD DATA #4
030250  104421  000040                  STEP    ,16.*2          ;SEND DATA #4 AND FORCE THE OVER RUN
030254  104420  000001                  READ    ,SEC1           ;READ REGISTER
030260  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
030264  042737  177767  001202          BIC     #^C<ROR>,$TMP0  ;CLEAR UNWANTED BITS
030272  032737  000010  001202          BIT     #ROR,$TMP0              ;TEST ROR
030300  001014                          BNE     1$                      ;SET SKIP ERROR
030302  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
030310  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
030316  052737  000010  001206          BIS     #ROR,$TMP2              ;LOAD EXPECTED
030324  104035                          ERROR   29.             ;RECEIVER OVERRUN DID NOT SET ON
                                                                ;A FORCE OVERRUN
030326  000137  030110                  JMP     101$            ;LOOP ON ERROR
030332                          1$:
030332  104420  000001                  READ    ,SEC1           ;READ REGISTER
030336  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
```

```
5951  030342  042737  177767  001202          BIC     #^C<ROR>,$TMP0  ;CLEAR UNWANTED BITS
5952  030350  032737  000010  001202          BIT     #ROR,$TMP0              ;TEST ROR
5953  030356  001414                          BEQ     2$                      ;RESET SKIP ERROR
5954  030360  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
5955  030366  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
5956  030374  042737  000010  001206          BIC     #ROR,$TMP2              ;LOAD EXPECTED
5957  030402  104036                          ERROR   30.             ;ROR DID NOT RESET AFTER READING
5958                                                                  ;RECEIVER STATUS REGISTER
5959  030404  000137  030110                  JMP     101$            ;LOOP ON ERROR
5960  030410                          2$:
5961  030410  000400                          BR      TST75           ;;
5962
5963
5964                                          ;****************************** TEST 75 ******************************
5965                                          ;*RECEIVER OVER RUN ERROR TEST FOR
5966                                          ;*PROTOCOL = BOP
5967                                          ;*THIS TEST WILL TRANSMIT DATA TO
5968                                          ;*THE RECEIVER IN MAINTENANCE MODE AND FAIL TO READ
5969                                          ;*THE SECOND DATA BIT TO FORCE THE OVERRUN CONDITION
5970                                          ;;*************************************************************************
5971
5972                                          ;  TEST 75
5973                                          ;-------------
5974                                  ;;**************************************************************
5975  030412  000004                  TST75:  SCOPE
5976  030414  012737  000075  001102          MOV     #75,$TSTNM              ; LOAD THE NO. OF THIS TEST
5977  030422  012737  000075  001262          MOV     #75,TSTNUM              ;# FOR TYPE OUT
5978  030430  104414                  101$:   MSTCLR                  ;CLEAR THE KMC11
5979  030432  104416  000007  000000          WRITE   ,SEC7,0         ;SET CHARACTER LENGTH TO 8 BITS/CHAR
5980  030440  104416  000005  000020          WRITE   ,SEC5,BOP!SAM   ;LOAD MODE CONTROL
5981  030446  104416  000004  000376          WRITE   ,SEC4,376       ;LOAD ADDRESS
5982  030454  104416  000014  000321          WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
5983                                                                  ;       TENA
5984                                                                  ;       RENA
5985                                                                  ;       MAINTENANCE BITS: MB, MC
5986  030462  104416  000003  000001          WRITE   ,SEC3,TSOM      ;SET TSOM
5987  030470  104421  000004                  STEP    ,2*2            ;STEP THE CLOCK 2 TICKS
5988  030474  104416  000003  000000          WRITE   ,SEC3,0         ;RESET TSOM
5989  030502  104416  000002  000376          WRITE   ,SEC2,376       ;LOAD ADDRESS INTO TDB
5990  030510  104421  000034                  STEP    ,14.*2          ;SYNC THE LOGIC
5991  030514  104416  000002  000000          WRITE   ,SEC2,0         ;LOAD CONTROL CHARACTER
5992  030522  104421  000020                  STEP    ,8.*2           ;SEND CHARACTER
5993  030526  104416  000002  000101          WRITE   ,SEC2,101       ;LOAD DATA #1
5994  030534  104421  000020                  STEP    ,8.*2           ;TRANSMIT DATA #1
5995  030540  104416  000002  000101          WRITE   ,SEC2,101       ;LOAD DATA #2
5996  030546  104421  000020                  STEP    ,8.*2           ;SEND DATA #2
5997  030552  104416  000002  000101          WRITE   ,SEC2,101       ;LOAD DATA #3
5998  030560  104421  000020                  STEP    ,8.*2           ;SEND DATA #3
5999                                  ;AT THIS POINT THE FIRST DATA BIT SHOULD BE IN THE RECEIVER
6000                                  ;BUFFER
6001  030564  104420  000000                  READ    ,SEC0           ;READ THE DATA
6002  030570  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
6003  030574  104416  000002  000101          WRITE   ,SEC2,101       ;LOAD DATA #4
6004  030602  104421  000040                  STEP    ,16.*2          ;SEND DATA #4 AND FORCE THE OVER RUN
6005  030606  104420  000001                  READ    ,SEC1           ;READ REGISTER
6006  030612  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
```

```
 6007  030616  042737  177767  001202          BIC     #^C<ROR>,$TMP0  ;CLEAR UNWANTED BITS
 6008  030624  032737  000010  001202          BIT     #ROR,$TMP0              ;TEST ROR
 6009  030632  001014                          BNE     1$                      ;SET SKIP ERROR
 6010  030634  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
 6011  030642  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
 6012  030650  052737  000010  001206          BIS     #ROR,$TMP2              ;LOAD EXPECTED
 6013  030656  104035                          ERROR   29.             ;RECEIVER OVERRUN DID NOT SET ON
 6014                                                                  ;A FORCE OVERRUN
 6015  030660  000137  030430                  JMP     101$            ;LOOP ON ERROR
 6016  030664                          1$:
 6017  030664  104420  000001                  READ    ,SEC1           ;READ REGISTER
 6018  030670  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
 6019  030674  042737  177767  001202          BIC     #^C<ROR>,$TMP0  ;CLEAR UNWANTED BITS
 6020  030702  032737  000010  001202          BIT     #ROR,$TMP0              ;TEST ROR
 6021  030710  001414                          BEQ     2$                      ;RESET SKIP ERROR
 6022  030712  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
 6023  030720  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
 6024  030726  042737  000010  001206          BIC     #ROR,$TMP2              ;LOAD EXPECTED
 6025  030734  104036                          ERROR   30.             ;ROR DID NOT RESET AFTER READING
 6026                                                                  ;RECEIVER STATUS REGISTER
 6027  030736  000137  030430                  JMP     101$            ;LOOP ON ERROR
 6028  030742                          2$:
 6029  030742  000400                          BR      TST76           ;;
```

```
 6030                                   .SBTTL  TRANSMIT AND RECEIVE ABORT TEST
 6031
 6032
 6033                                           ;****************************** TEST 76 ******************************
 6034                                           ;$LB07
 6035                                           ;*TRANSMIT ABORT AND RECEIVE ABORT TEST
 6036                                           ;*THIS TEST WILL FORCE AN ABORT
 6037                                           ;*SITUATION WITH THE IDLE BIT SET TO 0
 6038                                           ;*THE RESPONCE AT THE RECEIVER WILL BE
 6039                                           ;*A RECEPTION OF AN ABORT
 6040                                           ;;*************************************************************************
 6041
 6042                                           ;  TEST 76
 6043                                           ;-------------
 6044                                   ;;*************************************************************************
 6045  030744  000004                 TST76:  SCOPE
 6046  030746  012737  000076  001102         MOV     #76,$TSTNM                      ; LOAD THE NO. OF THIS TEST
 6047  030754  012737  000076  001262         MOV     #76,TSTNUM                      ;# FOR TYPE OUT
 6048  030762  104414                 101$:   MSTCLR                          ;CLEAR THE KMC11
 6049  030764  104416  000005  000020         WRITE   ,SEC5,BOP!SAM           ;LOAD MODE CONTROL
 6050  030772  104416  000004  000376         WRITE   ,SEC4,376               ;LOAD THE ADDRESS
 6051  031000  104416  000014  000321         WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
 6052                                                                          ;       TENA
 6053                                                                          ;       RENA
 6054                                                                          ;       MAINTENANCE BITS: MB, MC
 6055  031006  104416  000003  000001         WRITE   ,SEC3,TSOM              ;SET TSOM
 6056  031014  104421  000004                 STEP    ,2*2                    ;STEP 2 CLOCK TICKS
 6057  031020  104416  000003  000000         WRITE   ,SEC3,0                 ;RESET TSOM
 6058  031026  104416  000002  000376         WRITE   ,SEC2,376               ;LOAD THE ADDRESS
 6059  031034  104421  000034                 STEP    ,14.*2                  ;SYNC THE LOGIC
 6060  031040  104416  000002  000000         WRITE   ,SEC2,0                 ;LOAD CONTROL CHARACTER
 6061  031046  104421  000020                 STEP    ,8.*2                   ;SEND CONTROL CHARACTER
 6062  031052  104416  000002  000101         WRITE   ,SEC2,101               ;LOAD DATA #1
 6063  031060  104421  000020                 STEP    ,8.*2                   ;SEND DATA #1
 6064  031064  104416  000002  000101         WRITE   ,SEC2,101               ;LOAD DATA #2
 6065  031072  104421  000020                 STEP    ,8.*2                   ;SEND DATA #2
 6066  031076  104416  000002  000101         WRITE   ,SEC2,101               ;LOAD DATA #3
 6067  031104  104416  000003  000006         WRITE   ,SEC3,TEOM!TXAB         ;SET TEOM + TXABORT
 6068  031112  104421  000020                 STEP    ,8.*2                   ;SEND CRC #1
 6069  031116  104420  000000                 READ    ,SEC0                   ;READ ADDRESS
 6070  031122  012637  001202                 MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
 6071  031126  104421  000020                 STEP    ,8.*2                   ;SEND CRC #2
 6072  031132  104420  000000                 READ    ,SEC0                   ;READ CONTROL CHARACTER
 6073  031136  012637  001202                 MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
 6074  031142  104421  000020                 STEP    ,8.*2                   ;SEND
 6075  031146  104420  000000                 READ    ,SEC0                   ;READ DATA #1
 6076  031152  012637  001202                 MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
 6077  031156  104421  000022                 STEP    ,9.*2                   ;TRANSMIT ABORT CHARACTER
 6078  031162  104420  000001                 READ    ,SEC1                   ;READ REGISTER
 6079  031166  012637  001202                 MOV     (SP)+,$TMP0             ;;POP STACK INTO $TMP0
 6080  031172  042737  177773  001202         BIC     #^C<RXAB>,$TMP0 ;CLEAR UNWANTED BITS
 6081  031200  032737  000004  001202         BIT     #RXAB,$TMP0                     ;TEST RXAB
 6082  031206  001014                         BNE     1$                              ;SET SKIP ERROR
 6083  031210  013737  001202  001204         MOV     $TMP0,$TMP1                     ;PREPARE TO REPORT ERROR
 6084  031216  013737  001204  001206         MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
 6085  031224  052737  000004  001206         BIS     #RXAB,$TMP2                     ;LOAD EXPECTED
```

```
 6086  031232  104037                    ERROR   31.             ;RXAB DID NOT SET ON A TRANSMIT
 6087                                                              ;OF AN ABORT
 6088  031234  000137  030762            JMP     101$            ;LOOP ON ERROR
 6089  031240                    1$:
 6090  031240  000400                    BR      TST77           ;;
```

```
                                .SBTTL  TRANSMIT AND RECEIVE GO AHEAD TEST


                                        ;*****************************  TEST 77  *****************************
                                        ;$LB08
                                        ;*TRANSMIT GO AHEAD AND RECEIVE GO AHEAD TEST
                                        ;*THIS TEST WILL FORCE A GO AHEAD
                                        ;*SITUATION WITH THE IDLE BIT SET TO 0
                                        ;*THE RESPONCE AT THE RECEIVER WILL BE
                                        ;*A RECEPTION OF A GO AHEAD CHRACTER
                                        ;;***************************************************************************


                                        ;  TEST 77
                                        ;-------------
                                ;;***************************************************************************
031242  000004                  TST77:  SCOPE
031244  012737  000077  001102          MOV     #77,$TSTNM              ; LOAD THE NO. OF THIS TEST
031252  012737  000077  001262          MOV     #77,TSTNUM              ;# FOR TYPE OUT
031260  104414                  101$:   MSTCLR                  ;CLEAR THE KMC11
031262  104416  000005  000040          WRITE   ,SEC5,BOP!LM    ;LOAD MODE CONTROL (USYRT LOOP MODE)
031270  104416  000014  000321          WRITE   ,PRI14,TENA!RENA!MB!MC  ;ENABLE THE FOLLOWING:
                                                                ;       TENA
                                                                ;       RENA
                                                                ;       MAINTENANCE BITS: MB, MC
031276  104416  000003  000001          WRITE   ,SEC3,TSOM      ;SET TSOM
031304  104421  000004                  STEP    ,2*2            ;STEP 2 CLOCK TICKS
031310  104416  000003  000000          WRITE   ,SEC3,0         ;RESET TSOM
031316  104416  000002  000101          WRITE   ,SEC2,101               ;LOAD THE 1ST. DATA
031324  104421  000034                  STEP    ,14.*2          ;SYNC THE LOGIC & SEND 1ST.DATA
031330  104416  000002  000101          WRITE   ,SEC2,101               ;LOAD 2ND.DATA
031336  104421  000020                  STEP    ,8.*2           ;SEND 2ND.DATA
031342  104416  000002  000101          WRITE   ,SEC2,101       ;LOAD DATA #3
031350  104421  000020                  STEP    ,8.*2           ;SEND DATA #3
031354  104416  000003  000012          WRITE   ,SEC3,TEOM!TXGA ;SET TEOM + TXGA
031362  104421  000020                  STEP    ,8.*2           ;SEND CRC#1
031366  104420  000000                  READ    ,SEC0           ;READ ADDRESS
031372  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
031376  104421  000020                  STEP    ,8.*2           ;SEND CRC #2
031402  104420  000000                  READ    ,SEC0           ;READ DATA #1
031406  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
031412  104421  000020                  STEP    ,8.*2           ;SEND CRC #2
031416  104420  000000                  READ    ,SEC0           ;READ DATA #3
031422  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
031426  104421  000022                  STEP    ,9.*2           ;TRANSMIT GO AHEAD CHARACTER
031432  104420  000001                  READ    ,SEC1           ;READ REGISTER
031436  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
031442  042737  177773  001202          BIC     #^C<RXGA>,$TMP0 ;CLEAR UNWANTED BITS
031450  032737  000004  001202          BIT     #RXGA,$TMP0             ;TEST RXGA
031456  001014                          BNE     1$                      ;SET SKIP ERROR
031460  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
031466  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
031474  052737  000004  001206          BIS     #RXGA,$TMP2             ;LOAD EXPECTED
031502  104040                          ERROR   32.             ;RXGA DID NOT SET ON A TRANSMIT
                                                                ;OF A GO AHEAD
031504  000137  031260                  JMP     101$            ;LOOP ON TEST
031510                          1$:
```

```
 6147  031510  000400                        BR      TST100          ;;
```

```
 6148                                   ;;*****************************************************************
 6149                                           ;$EXTRF
 6150
 6151                                           ;THIS TEST RUNS ONLY EXT. DATA LOOP
 6152
 6153                                           ;THIS TEST WILL TRANSMIT A WALKING ONES DATA PATTERN
 6154                                           ;IN THE BOP MODE USING THE EXTERNAL DATA LOOP CLOCK PATH.
 6155                                           ;THE TEST IS INTENDED TO CHECK THE EXTERNAL DRIVERS AND RECEIVERS.
 6156
 6157                                           ;THIS TEST WILL PERFORM THE FOLLOWING SEQUENCE
 6158
 6159                                           ;1.=ISSUE A MASTER CLEAR PULSE
 6160                                           ;2.=SET PROTOCOL = BOP
 6161                                           ;3.=SET TEMA,RENA,MB(EXTERNAL LOOP)
 6162                                           ;4.=SET TSOM
 6163                                           ;5.=STEP CLOCK 2 TICKS
 6164                                           ;6=.STEP CLOCK UNTIL TBMT SETS
 6165                                           ;7=TRANSMIT 7 DATA BYTES OF THE CURRENT DATA PATTERN
 6166                                           ;8=TEST THAT "RDA" IS SET
 6167                                           ;9=READ DATA BYTE AND VERIFY
 6168
 6169
 6170                                           ;  TEST 100
 6171                                           ;-------------
 6172                                   ;;*****************************************************************
 6173  031512  000004                  TST100: SCOPE
 6174  031514  012737  000100  001102           MOV     #100,$TSTNM             ; LOAD THE NO. OF THIS TEST
 6175  031522  012737  000100  001262           MOV     #100,TSTNUM             ;# FOR TYPE OUT
 6176  031530  132737  000200  001260  10$:     BITB    #200,CLKDAT                     ;IS EXT. TESTS ENABLED?
 6177  031536  001146                           BNE     8$                      ;B=NO SKIP TEST
 6178  031540  012737  031530  001106           MOV     #10$,$LPADR             ;SET RETURN ADDRESS (ERROR)
 6179  031546  012737  000001  001210           MOV     #1,$TMP3                ;INITIALIZE DATA PATTERN
 6180  031554  012737  000007  001212  1$:      MOV     #7,$TMP4                ;COUNT DATA SENT
 6181  031562  104414                           MSTCLR
 6182  031564  104416  000005  000000           WRITE   ,SEC5,BOP
 6183  031572  104416  000014  000121           WRITE   ,PRI14,TENA!RENA!MB     ;ENABLE EXT. LOOP, XMIT & REC.
 6184  031600  013737  001210  031632           MOV     $TMP3,2$                ;LOAD DATA FOR TRANSMITTING
 6185  031606  104416  000003  000001           WRITE   ,SEC3,TSOM              ;SET TRANSMIT START OF MESSAGE
 6186  031614  104421  000004                   STEP    ,2*2                    ;STEP CLOCK 2 TICKS
 6187  031620  104416  000003  000000           WRITE   ,SEC3,0                 ;CLEAR TSOM
 6188  031626  104416  000002          11$:     WRITE   ,SEC2                   ;LOAD THE DATA WORD
 6189  031632  000000                  2$:      0
 6190  031634  012737  000011  032056           MOV     #9.,9$
 6191  031642  104421  000002          21$:     STEP    ,2
 6192  031646  104420  000013                   READ    ,PRI13
 6193  031652  012637  001204                   MOV     (SP)+,$TMP1
 6194  031656  032737  000020  001204           BIT     #20,$TMP1               ;IS TBMT SET?
 6195  031664  001014                           BNE     22$                     ;B=YES TRY FOR 2ND BYTE
 6196  031666  005337  032056                   DEC     9$                      ;WITHIN WATCH LOOP
 6197  031672  001363                           BNE     21$                     ;B=YES
 6198  031674  013737  001204  001206           MOV     $TMP1,$TMP2             ;LOAD DATAA FOR ERROR
 6199  031702  052737  000020  001206           BIS     #20,$TMP2               ;SHOW TBMT SHOULD BE SET
 6200  031710  104015                           ERROR   13.
 6201  031712  000137  031554                   JMP     1$
 6202  031716  005337  001212          22$:     DEC     $TMP4                   ;-1 TO DATA SENT
 6203  031722  001341                           BNE     11$                     ;B=SEND NEXT DATA
```

```
6204  031724                          4$:     ;WATCHDOG COMPLETION OF DATA BYTE
6205  031724  012737  000011  032056          MOV     #9.,9$
6206  031732  104421  000002          5$:     STEP    ,2
6207  031736  104420  000013                  READ    ,PRI13                  ;CHECK IF RDA IS SET
6208  031742  012637  001204                  MOV     (SP)+,$TMP1
6209  031746  032737  000001  001204          BIT     #1,$TMP1                ;IS DATA AVAILABLE?
6210  031754  001014                          BNE     6$                      ;B=YES
6211  031756  005337  032056                  DEC     9$                      ;UPDATE WATCHDOG
6212  031762  001363                          BNE     5$                      ;B=KEEP TRYING
6213  031764  013737  001204  001206          MOV     $TMP1,$TMP2             ;LOAD ERROR DATA
6214  031772  052737  000001  001206          BIS     #1,$TMP2                ;SHOW RDA SHOULD BE SET
6215  032000  104052                          ERROR   42.
6216  032002  000137  031554                  JMP     1$
6217  032006                          6$:     ;READ DATA CHR. AND VERIFY
6218  032006  104420  000000                  READ    ,SEC0                   ;READ DATA
6219  032012  012637  001202                  MOV     (SP)+,$TMP0
6220  032016  013737  001210  001204          MOV     $TMP3,$TMP1
6221  032024  023737  001204  001202          CMP     $TMP1,$TMP0             ;IS XFER CORRECT?
6222  032032  001403                          BEQ     7$
6223  032034  104027                          ERROR   23.
6224  032036  000137  031554                  JMP     1$
6225  032042  106337  001210          7$:     ASLB    $TMP3                   ;UPDATE DATE DATA & TEST FOR DONE
6226  032046  103402                          BCS     8$
6227  032050  000137  031554                  JMP     1$                      ;TEST NEXT CHR.
6228  032054                          8$:
6229  032054  000401                          BR      TST101          ;;
6230  032056  000000                  9$:     0
6231                                  .SBTTL  CRC-32 TESTS
6232                                  ;;**********************************************************************
6233                                          ;$CRCTER
6234                                          ;THIS TEST WILL CHECK THAT THE TRANSMITTER
6235                                          ;CHECK (CRC-32)ERROR FLAG (PR14[2]:(1)) WILL SET BY TRANSMITTING
6236                                          ;A DATA MESSAGE WITH THE XMIT CHECK MODE ENABLED
6237                                          ;AND AN ILLEGAL CRC 32 CODE APPENDED TO THE DATA
6238                                          ;WORD.
6239                                          ;THE DATA WORD TRANSMITTED IS A BYTE OF ALL ZERO'S
6240                                          ;AND THE APPENDED 32 BIT CRC IS = 1.
6241                                          ;**BOP - MODE ONLY
6242
6243                                          ;THIS TEST WILL PERFORM THE FOLLOWING SEQUENCE
6244
6245                                          ;1. ISSUE A MASTER CLEAR PULSE
6246                                          ;2. SET PROTOCOL = BOP
6247                                          ;3. SET CRCHK (TRANSMITTER CHECK MODE)
6248                                          ;4. TEST TCRCE, AND RCRCE ARE NOT SET
6249                                          ;5. SET ECRC, TENA, MB (EXTERNAL LOOP)
6250                                          ;6. SET TSOM
6251                                          ;7. STEP CLOCK 2 TICKS
6252                                          ;8. TEST FOR "TBMT" SET (RDY.FOR DATA)
6253                                          ;9. LOOP TO TRANSMIT THE DATA BYTE (0) AND OUTPUT
6254                                          ;THE 4-8 BYTES OF CRC-32
6255                                          ;10. SET TEOM
6256                                          ;11. TICK CLOCK 15. MORE TIMES TO PRESENT TEOM TO THE TDB
6257
6258
6259                                          ;12. TEST THAT "TCRCE" IS SET (TRANSMIT CRC ERROR)(REPORT IF NOT)
```

```
                                        ;13. IF FIRST TIME TEST THAT "RCRCE" CLEARS BY WRITTING TO A ONE
                                        ;       (PREVIOUSLY READ REGISTER CONTENTS IS WRITTEN)
                                        ;13. IF SECOND TIME A LINE RESET IS ISSUED AND "TCRCE" IS TESTED
                                        ;       FOR BEING CLEARED.


                                        ;  TEST 101
                                        ;--------------
                                    ;;*****************************************************************
032060  000004                    TST101: SCOPE
032062  012737  000101  001102            MOV     #101,$TSTNM                 ; LOAD THE NO. OF THIS TEST
032070  012737  000101  001262            MOV     #101,TSTNUM                 ;# FOR TYPE OUT
032076  005037  032604                    CLR     12$                         ;INIT.PASS FLAG
032102  105737  001256                    TSTB    CRC32F              ;CRC32 ENABLED?
032106  100402                            BMI     1$                          ;B=YES
032110  000137  032574                    JMP     11$                         ;SKIP TEST EXT. LOOP NOT ENABLED
032114  012737  032114  001106    1$:     MOV     #1$,$LPADR                  ;SET RETURN ADDRESS (ERROR)
032122  104414                            MSTCLR                              ;ISSUE A MASTER CLEAR
032124  104416  000005  000007            WRITE   ,SEC5,BOP!DISCRC                    ;SET BOP MODE
032132  104416  000013  000040            WRITE   ,PRI13,CRCHK                ;SET TRANSMIT CHECK MODE
                                        ;TEST THAT BITS 1&2 OF PRIMARY 14 ARE CLEAR
032140  104420  000014                    READ    ,PRI14                      ;GET CRC-32 ERROR STATUS BITS
032144  012637  001202                    MOV     (SP)+,$TMP0
032150  032737  000006  001202            BIT     #TCRCE!RCRCE,$TMP0          ;ARE STATUS BITS CLRAR?
032156  001411                            BEQ     2$                          ;B=YES
032160  013737  001202  001204            MOV     $TMP0,$TMP1                 ;SET UP ERROR STATUS
032166  042737  000006  001204            BIC     #TCRCE!RCRCE,$TMP1          ;SHOW BITS SHOULD BE CLRAR
032174  104046                            ERROR   38.                         ;PRI14 BITS 1&2 SHOULD NOT BE SET
032176  000137  032114                    JMP     1$
032202  104416  000014  000131    2$:     WRITE   ,PRI14,ECRC!TENA!MB!RENA            ;SET EN. CRC-32,TENA,EXTERNAL S/S
032210  104416  000003  000001            WRITE   ,SEC3,TSOM
032216  104421  000004                    STEP    ,2*2                        ;STEP THE CLOCK 7 TICKS
032222  104416  000003  000000            WRITE   ,SEC3,0                     ;CLR TSOM
032230  012737  032612  032606            MOV     #BADCRC,13$                 ;POINT TO DATA & ILLEGAL CRC
032236  012737  000005  032610            MOV     #5,14$                      ;1 DATA 4 CRC
032244  000427                            BR      35$
032246                          3$:     ;TEST THAT "TBMT" IS SET (RDY. FOR DATA)
032246  104420  000013                    READ    ,PRI13              ;READ REGISTER
032252  012637  001202                    MOV     (SP)+,$TMP0         ;;POP STACK INTO $TMP0
032256  042737  177757  001202            BIC     #^C<TBMT>,$TMP0 ;CLEAR UNWANTED BITS
032264  032737  000020  001202            BIT     #TBMT,$TMP0                 ;TEST TBMT
032272  001014                            BNE     35$                         ;SET SKIP ERROR
032274  013737  001202  001204            MOV     $TMP0,$TMP1                 ;PREPARE TO REPORT ERROR
032302  013737  001204  001206            MOV     $TMP1,$TMP2         ;LOAD EXPECTED ALSO
032310  052737  000020  001206            BIS     #TBMT,$TMP2                 ;LOAD EXPECTED
032316  104015                            ERROR   13.                         ;TBMT FAILED TO SET W/TSOM SET & CLK STEPPED.
032320  000137  032114                    JMP     1$
032324  117737  000256  032342  35$:     MOVB    @13$,4$             ;LOAD A DATA BYTE AND CRC 32
032332  005237  0326C6                    INC     13$
032336  104416  000002                    WRITE   ,SEC2
032342  000000                    4$:     0
032344  104421  000020                    STEP    ,8.*2                       ;STEP CLOCK 8 TICKS
032350  005337  032610                    DEC     14$                         ;DONE?
032354  001334                            BNE     3$                          ;B=NO
032356  104416  000003  000002            WRITE   ,SEC3,TEOM                  ;SET TEOM
032364  104421  000042                    STEP    ,17.*2                      ;TICKS SHOULD LOAD THE TEOM (176)*TEOM NOT SHIFT
```

```
                                                ;TEST "TCRCE" IS SET AND "RCRCE" IS NOT
032370  104420  000014                          READ    ,PRI14
032374  012637  001202                          MOV     (SP)+,$TMP0             ;ACTUAL VALUE
032400  013737  001202  001204                  MOV     $TMP0,$TMP1             ;SET UP EXPECTED
032406  042737  000002  001202                  BIC     #BIT1,$TMP0             ;EXPECTED
032414  052737  000004  001202                  BIS     #BIT2,$TMP0
032422  023737  001204  001202                  CMP     $TMP1,$TMP0             ;IS "TCRCE" SET?
032430  001403                                  BEQ     6$                      ;B=YES
032432  104046                                  ERROR   38.
032434  000137  032114                          JMP     1$
032440                                  6$:     ;TEST FOR TYPE OF CLEAR TO BE TESTED
032440  005737  032604                          TST     12$                     ;1ST. TIME OR 2ND.
032444  001033                                  BNE     10$                     ;B=2ND
                                                ;TEST FOR CLEARING "TCRCE" BY WRITTING THE BIT WITH A ONE
032446  013737  001202  032460                  MOV     $TMP0,7$                ;REWRITE ORIGINAL CONTENTS OF REG.
032454  104416  000014                          WRITE   ,PRI14
032460  000000                          7$:     0
032462  104420  000014                  8$:     READ    ,PRI14                  ;READ ERROR STATUS BIT FOR BEING CLEAR
032466  012637  001202                          MOV     (SP)+,$TMP0             ;
032472  013737  001202  001204                  MOV     $TMP0,$TMP1             ;SET UP EXPECTED
032500  042737  000004  001202                  BIC     #BIT2,$TMP0             ;CLR THE "TCRCE" BIT
032506  032737  000004  001204                  BIT     #TCRCE,$TMP1            ;IS BIT CLEAR?
032514  001403                                  BEQ     9$                      ;B=YES
032516  104046                                  ERROR   38.                     ;"TCRCE" FAIL TO CLEAR BY REWRITE OF A 1
032520  000137  032114                          JMP     1$
032524  005137  032604                  9$:     COM     12$                     ;NOW RERUN AND DO 2ND CLEAR.
032530  000137  032114                          JMP     1$
032534                                  10$:    ;ISSUE A LINE RESET AND TEST THAT "TCRCE" CLEARS
032534  104416  000012  000001                  WRITE   ,PRI12,LNRST            ;LINE RESET
032542  104420  000014                          READ    ,PRI14                  ;READ ERROR STATUS BIT FOR BEING CLEAR
032546  012637  001202                          MOV     (SP)+,$TMP0
032552  013737  001202  001204                  MOV     $TMP0,$TMP1             ;SET UP EXPECTED
032560  042737  000004  001202                  BIC     #BIT2,$TMP0             ;CLEAR THE "TCRCE" BIT
032566  032737  000004  001204                  BIT     #TCRCE,$TMP1            ;IS BIT CLEAR?
032574                                  11$:
032574  001411                                  BEQ     TST102          ;;
032576  104046                                  ERROR   38.
032600  000137  032114                          JMP     1$
032604  000000                          12$:    0                               ;PASS FLAG FOR CLEARING ROUTINES
032606  000000                          13$:    0
032610  000000                          14$:    0

032612      000                 BADCRC: .BYTE   0                               ;DATA WORD
032613      001                         .BYTE   1                               ;LSB OF 32 BIT CRC
032614      000                         .BYTE   0                               ;CRC (ILLEGAL)
032615      000                         .BYTE   0                               ;CRC (ILLEGAL)
032616      000                         .BYTE   0                               ;CRC (ILLEGAL)
032617      000                         .BYTE   0                               ;CRC (ILLEGAL)
```

```
                                ;;*****************************************************************
                                        ;$CRCRER

                                        ;THIS TEST WILL CHECK THAT THE RECEIVER CHECK (CRC-32)
                                        ;ERROR FLAG (PR14[1]:(1) WILL SET BY TRANSMITTING A DATA
                                        ;MESSAGE WITH THE CRC-32 ENABLED AND THE GENERATE
                                        ;MODE NOT ENABLED (=CHECK MODE) (PR13[5](1), THE APPENDED
                                        ;CRC-32 IS 32 BITS =0000000001  AND THE MESSAGE IS 8 BITS
                                        ; =1.

                                        ;THIS TEST WILL PERFORM THE FOLLOWING SEQUENCE

                                        ;1. ISSUE A MASTER CLEAR PULSE
                                        ;2. SET PROTOCOL=BOP
                                        ;3. TEST TCRCE AND RCRCE ARE NOT SET
                                        ;4. SET ECRC, TENA, RENA, MB (EXTERNAL DATA LOOP)
                                        ;5. SET TSOM
                                        ;6. STEP CLOCK 2 TICKS
                                        ;7. TEST FOR "TBMT" SET (RDY FOR DATA
                                        ;8. LOOP TO TRANSMIT THE DATA BYTE (0) AND OUTPUT THE 4-8 BIT CRC
                                        ;9. SET TEOM
                                        ;10. TICK CLOCK 24. MORE TIMES TO PRESENT TEOM TO RDB
                                        ;11. TEST THAT "RCRCE" IS SET (RECEIVE CRC ERROR) (REPORT IF NOT)
                                        ;12. IF FIRST TIME TEST THAT "RCRCE" CLEARS BY WRITTING TO A ONE
                                        ;12. IF SECOND TIME A "LINE RESET" IS ISSUED AND "TCRCE" IS TESTED
                                        ;FOR BEING CLEARED.

                                        ;  TEST 102
                                        ;-------------
                                ;;*****************************************************************
032620  000004                TST102: SCOPE
032622  012737  000102  001102        MOV     #102,$TSTNM             ; LOAD THE NO. OF THIS TEST
032630  012737  000102  001262        MOV     #102,TSTNUM             ;# FOR TYPE OUT
032636  005037  033366                CLR     12$                     ;PASS FLAG
032642  105737  001256                TSTB    CRC32F          ;CRC32 ENABLED?
032646  100402                        BMI     1$                      ;B=YES
032650  000137  033356                JMP     11$                     ;EXT. LOOP NOT ENABLED
032654  012737  032654  001106  1$:   MOV     #1$,$LPADR              ;SET RETURN ADDRESS (ERROR)
032662  104414                        MSTCLR
032664  104416  000005  000007        WRITE   ,SEC5,BOP!DISCRC                ;SET THE BOP MODE
032672  104416  000013  000040        WRITE   ,PRI13,CRCHK            ;SET XMIT MODE -NO GEN.
                                        ;TEST THAT BITS 1&2 OF PRIMARY 14 ARE CLEAR.
032700  104420  000014                READ    ,PRI14
032704  012637  001202                MOV     (SP)+,$TMP0
032710  032737  000006  001202        BIT     #TCRCE!RCRCE,$TMP0              ;ARE STATUS BITS CLEAR?
032716  001411                        BEQ     2$                      ;B=YES
032720  013737  001202  001204        MOV     $TMP0,$TMP1             ;SET UP ERROR STATUS
032726  042737  000006  001202        BIC     #TCRCE!RCRCE,$TMP0              ;SHOW BITS SHOULD BE CLEAR
032734  104046                        ERROR   38.                     ;PRI 14 BITS 1&2 SHOULD NOT BE SET
032736  000137  032654                JMP     1$
032742  104416  000014  000131  2$:   WRITE   ,PRI14,ECRC!TENA!RENA!MB        ;EM.CRC,TENA,RENA!EXT.DATA LOOP
032750  104416  000003  000001        WRITE   ,SEC3,TSOM
032756  104421  000004                STEP    ,2*2                    ;STEP THE CLOCK
032762  104416  000003  000000        WRITE   ,SEC3,0                 ;CLR    TSOM
032770  012737  032612  033370        MOV     #BADCRC,13$             ;POINT TO DATA & ILLEGAL CRC
032776  012737  000005  033372        MOV     #5,14$                  ;1 DATA 4CRC
```

```
6421  033004  000427                          BR      3$
6422  033006                          25$:    ;TEST TBMT IS SET
6423  033006  104420  000013                  READ    ,PRI13          ;READ REGISTER
6424  033012  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
6425  033016  042737  177757  001202          BIC     #^C<TBMT>,$TMP0 ;CLEAR UNWANTED BITS
6426  033024  032737  000020  001202          BIT     #TBMT,$TMP0             ;TEST TBMT
6427  033032  001014                          BNE     3$                      ;SET SKIP ERROR
6428  033034  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
6429  033042  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
6430  033050  052737  000020  001206          BIS     #TBMT,$TMP2             ;LOAD EXPECTED
6431  033056  104015                          ERROR   13.                     ;TBMT FAILED TO SET W/TSOM & CLK STEPPED
6432  033060  000137  032654                  JMP     1$
6433                                  ;LOAD DATA CHR. AND STEP CLOCK
6434  033064  117737  000300  033102  3$:     MOVB    @13$,4$         ;LOAD DATA BYTE
6435  033072  005237  033370                  INC     13$
6436  033076  104416  000002                  WRITE   ,SEC2
6437  033102  000000                  4$:     0
6438  033104  104421  000020                  STEP    ,8.*2                   ;8 TICKS
6439  033110  104420  000000                  READ    ,SEC0           ;READ DATA TO PREVENT ROR
6440  033114  005726                          TST     (SP)+                   ;BUMP STACK
6441  033116  005337  033372                  DEC     14$                     ;DONE?
6442  033122  001331                          BNE     25$                     ;B= NO
6443  033124  104416  000003  000002          WRITE   ,SEC3,TEOM
6444  033132  012737  000004  033372          MOV     #4,14$
6445  033140  104421  000020          45$:    STEP    ,8.*2
6446  033144  104420  000000                  READ    ,SEC0           ;READ DATA TO PREVENT ROR
6447  033150  005726                          TST     (SP)+
6448  033152  005337  033372                  DEC     14$
6449  033156  001370                          BNE     45$
6450                                  ;TEST "RCRCE" IS SET
6451  033160  104420  000014                  READ    ,PRI14
6452  033164  012637  001202                  MOV     (SP)+,$TMP0
6453  033170  013737  001202  001204          MOV     $TMP0,$TMP1             ;SET UP EXPECTED
6454  033176  052737  000002  001202          BIS     #BIT1,$TMP0             ;SET EXPECTED
6455  033204  023737  001202  001204          CMP     $TMP0,$TMP1             ;IS RECEIVED  ERROR SET
6456  033212  001403                          BEQ     5$                      ;B=YES
6457  033214  104046                          ERROR   38.
6458  033216  000137  032654                  JMP     1$
6459
6460  033222                          5$:     ;TEST FOR TYPE OF CLEAR TO BE TESTED
6461  033222  005737  033366                  TST     12$                     ;1ST. OR 2ND. TIME
6462  033226  001033                          BNE     10$                     ;B=2ND.
6463                                  ;TEST FOR CLEARING "TCRCE" BY WRITTING THE BIT WITH A ONE
6464  033230  013737  001202  033242          MOV     $TMP0,7$                ;REWRITE THE ORIGINAL CONTENTS
6465  033236  104416  000014                  WRITE   ,PRI14
6466  033242  000000                  7$:     0
6467  033244  104420  000014          8$:     READ    ,PRI14
6468  033250  012637  001202                  MOV     (SP)+,$TMP0                     ;READ ERROR STATUS BIT FOR BEING CLEAR
6469  033254  013737  001202  001204          MOV     $TMP0,$TMP1             ;SET UP EXPECTED
6470  033262  042737  000002  001202          BIC     #BIT1,$TMP0             ;CLR THE "RCRCE" BIT
6471  033270  032737  000002  001202          BIT     #RCRCE,$TMP0            ;IS BIT CLEAR?
6472  033276  001403                          BEQ     9$                      ;B=YES
6473  033300  104046                          ERROR   38.                     ;"RCRCE" FAILED TO CLEAR BY REWRITE OF A
6474  033302  000137  032654                  JMP     1$
6475  033306  005137  033366          9$:     COM     12$                     ;RERUN TEST AND DO 2ND.
6476  033312  000137  032654                  JMP     1$
```

```
 6477
 6478
 6479  033316                          10$:    ;ISSUE A "LINE RESET" AND TEST THAT "RCRCE" CLEARS
 6480  033316  104416  000012  000001          WRITE   ,PRI12,LNRST                    ;LINE RESET
 6481  033324  104420  000014                  READ    ,PRI14                          ;
 6482  033330  012637  001202                  MOV     (SP)+,$TMP0
 6483  033334  013737  001202  001204          MOV     $TMP0,$TMP1                     ;SET UP EXPECTED
 6484  033342  042737  000002  001202          BIC     #BIT1,$TMP0                     ;CLEAR THE "RCRCE" BIT
 6485  033350  032737  000002  001202          BIT     #RCRCE,$TMP0                    ;IS BIT CLEAR?
 6486  033356                          11$:
 6487  033356  001406                          BEQ     TST103          ;;
 6488  033360  104046                          ERROR   38.
 6489  033362  000137  032654                  JMP     1$
 6490  033366  000000                  12$:    0                                       ;PASS FLAG
 6491  033370  000000                  13$:    0
 6492  033372  000000                  14$:    0
```

```
                                ;;*****************************************************************
                                        ;$CRCNER
                                        ;THIS TEST WILL CHECK A NON-ERROR CRC RECEIVE CHECK AND NON-ERROR TRANSMIT CHECK
                                        ;BY TRANSMITTING A DATA STREAM (NON-GENERATE) WITH A PRE-
                                        ;CALCULATED CRC APPENDED (3 WORDS (1 DATA & 2 CRC)
                                        ;THE DATA PATTERN CONSISTS OF WALKING A ONE ACCROSS 16 BITS
                                        ;(I.E. 1 TO 100000) AND THE 32 BIT CRC IS CALCULATED FROM
                                        ;ROUTINE CALLED BY "JSR         PC,CRCCAL".


                                        ;THIS TEST WILL PERFORM THE FOLLOWING SEQUENCE

                                        ;1. ISSUE A MASTER CLEAR PULSE
                                        ;2. SET PROTOCOL=BOP
                                        ;3. SET TENA,RENA,ECRC,MB
                                        ;4. SET "CRCHK" (TRANSMITTER CHECK MODE)
                                        ;5. TEST "TCRCE" AND "RCRCE" ARE NOT SET
                                        ;6. SET "TSOM"
                                        ;7. STEP CLOCK 2 TICKS
                                        ;8. TEST FOR "TBMT" SET
                                        ;9. CALCULATE CRC
                                        ;10. LOOP TO TRANSMIT THE DATA AND CRC
                                        ;11. TICK CLOCK TO COMPLETE DATA STREAM
                                        ;12. TEST THAT "TCRCE" AND "RCRCE" BITS ARE NOT SET (REPORT IF EITHER SET)
                                        ;13. UPDATE DATA BIT AND LOOP   UNTIL DONE


                                        ;  TEST 103
                                        ;--------------
                                ;;*****************************************************************
033374  000004                  TST103: SCOPE
033376  012737  000103  001102          MOV     #103,$TSTNM             ; LOAD THE NO. OF THIS TEST
033404  012737  000103  001262          MOV     #103,TSTNUM             ;# FOR TYPE OUT
033412  105737  001256                  TSTB    CRC32F  ;CRC32 ENABLED?
033416  100402                          BMI     101$
033420  000137  034110                  JMP     14$
033424  012737  033446  001106  101$:   MOV     #1$,$LPADR                      ;SET RETURN ADDRESS (ERROR)
033432  012737  000001  034074          MOV     #1,8$                           ;INITIALIZE DATA LOW WORD
033440  012737  100000  034106          MOV     #100000,13$                     ;BIT 15 OF 2ND. WORD IS LAST OUT
033446  104414                  1$:     MSTCLR
033450  104416  000005  000007          WRITE   ,SEC5,BOP!DISCRC                        ;BOP MODE AND DISABLE CRC-16
033456  104416  000013  000040          WRITE   ,PRI13,CRCHK                    ;SET XMIT MODE-NON GENERATE
033464  104416  000014  000131          WRITE   ,PRI14,RENA!ECRC!TENA!MB

                                        ;TEST THAT "TCRCE" AND "RCRCE" ARE NOT SET.
033472  104420  000014                  READ    ,PRI14
033476  012637  001202                  MOV     (SP)+,$TMP0
033502  032737  000006  001202          BIT     #TCRCE!RCRCE,$TMP0              ;ARE CRC STATUS BITS CLEAR?
033510  001411                          BEQ     2$                              ;B=Y
033512  013737  001202  001204          MOV     $TMP0,$TMP1
033520  042737  000006  001202          BIC     #TCRCE!RCRCE,$TMP0              ;SHOW EXPECTED
033526  104046                          ERROR   38.                             ;PRI14 1&2 SHOULD NOT BE SET
033530  000137  033446                  JMP     1$
033534  104416  000003  000001  2$:     WRITE   ,SEC3,TSOM                      ;SET TSOM
033542  104421  000004                  STEP    ,2*2                            ;2 TICKS
033546  104416  000003  000000          WRITE   ,SEC3,0                         ;CLR TSOM
```

```
6549  033554  013702  034074                  MOV     8$,R2
6550  033560  005003                          CLR     R3
6551  033562  004737  047462                  JSR     PC,CRCCAL                       ;CALCULATEHE 32-BIT CRC
6552                                          ;HIGH ORDER CRC IS IN $TMP4 AND LOW ORDER IN $TMP5
6553  033566  012702  034064                  MOV     #7$,R2
6554  033572  005022                          CLR     (R2)+                           ;1ST. 2 BYTES ARE 0
6555  033574  013722  034106                  MOV     13$,(R2)+                       ;STORE DATA
6556  033600  013722  001214                  MOV     $TMP5,(R2)+                     ;CRC
6557  033604  013712  001212                  MOV     $TMP4,(R2)
6558  033610  012737  034064  034076          MOV     #7$,9$                          ;REPOINT TO DATA
6559  033616  012737  000010  034100          MOV     #8.,10$                         ;COUNT 8 BYTES OF DATA OUT
6560  033624  012737  000010  034102          MOV     #8.,11$                         ;COUNT 8 CHARACTERS IN
6561  033632  117737  000240  033650  3$:     MOVB    @9$,35$
6562  033640  005237  034076                  INC     9$
6563  033644  104416  000002                  WRITE   ,SEC2
6564  033650  000000                35$:      0
6565                                          ;ADDITIONAL CLOCK TICKS MAY BE REQUIRED BECAUSE OF
6566                                          ;USYRT LOGIC INSERTING OF ZEROS IN DATA STREAM.
6567                                          ;THE FOLLOWING SEQUENCE WILL MONITOR BOTH XMIT AND REC
6568                                          ;SIGNALS TO COMPLETE DATA AND CRC MESSAGES.
6569  033652  012737  000013  034104  34$:     MOV     #11.,12$                        ;WATCH DOG XFER (CAN INSERT UP TTO 2 0'S/BYTE)
6570  033660  104421  000002                36$:      STEP    ,2
6571  033664  005337  034104                  DEC     12$                             ;TOO MANY TICKS AND NO RESPONSE?
6572  033670  001472                          BEQ     66$                             ;B= YES
6573  033672  104420  000013                  READ    ,PRI13                                  ;IS RDA SET?
6574  033676  032726  000001                  BIT     #RDA,(SP)+
6575  033702  001406                          BEQ     37$                                     ;B = NO
6576  033704  104420  000000                  READ    ,SEC0                                   ;YES READ DATA
6577  033710  005726                          TST     (SP)+                                   ;DROP DATA ON FLOOR
6578  033712  005337  034102                  DEC     11$                                     ;ARE ALL DATA CHRS. IN?
6579  033716  001421                          BEQ     5$                                      ;B=YES-GO
6580  033720  104420  000013                37$:      READ    ,PRI13                          ;IS TBMT SET?
6581  033724  032726  000020                  BIT     #TBMT,(SP)+                             ;
6582  033730  001753                          BEQ     36$                                     ;B=NO KEEP STEPPING
6583  033732  005337  034100                  DEC     10$                                     ;ARE ALL CHRS. SENT?
6584  033736  003335                          BGT     3$                                      ;NO LOAD NEXT
6585  033740  002404                          BLT     38$                                     ;B=FLAG SENT
6586  033742  104416  000003  000002          WRITE   ,SEC3,TEOM                              ;SET END OF MESSAGE
6587  033750  000740                          BR      34$
6588  033752  104416  000003  000000  38$:     WRITE   ,SEC3,0                                 ;CLR END OF MESSAGE
6589  033760  000734                          BR      34$
6590  033762                        5$:       ;NOW TEST THAT "TCRCE" AND "RCRCE" ARE NOT SET
6591  033762  104420  000014                  READ    ,PRI14                  ;READ REGISTER
6592  033766  012637  001202                  MOV     (SP)+,$TMP0             ;:POP STACK INTO $TMP0
6593  033772  042737  177771  001202          BIC     #^C<TCRCE!RCRCE>,$TMP0  ;CLEAR UNWANTED BITS
6594  034000  032737  000006  001202          BIT     #TCRCE!RCRCE,$TMP0                      ;TEST TCRCE!RCRCE
6595  034006  001414                          BEQ     6$                              ;RESET SKIP ERROR
6596  034010  013737  001202  001204          MOV     $TMP0,$TMP1                     ;PREPARE TO REPORT ERROR
6597  034016  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
6598  034024  042737  000006  001206          BIC     #TCRCE!RCRCE,$TMP2                      ;LOAD EXPECTED
6599  034032  104047                          ERROR   39.                             ;
6600  034034  000137  033446                  JMP     1$
6601  034040  006337  034074          6$:     ASL     8$                                      ;UPDATE THE DATA
6602  034044  103421                          BCS     14$
6603  034046  006037  034106                  ROR     13$                                     ;ALLIGN BIT XFERED
6604  034052  000137  033446                  JMP     1$
```

```
 6605  034056  104051                  66$:    ERROR   41.
 6606  034060  000137  033446                  JMP     1$
 6607
 6608  034064  000000                  7$:     0       ;STORE DATA AND CRC
 6609  034066  000000                          0       ;DATA CHR.
 6610  034070  000000                          0       ;1ST. 2 CRCS
 6611  034072  000000                          0       ;2ND. 2 CRCS
 6612  034074  000000                  8$:     0       ;CURRENT DATA PATTERN
 6613  034076  000000                  9$:     0       ;POINTER TO DATA FOR XMIT
 6614  034100  000000                  10$:    0       ;BYTES OF DATA SENT(STARTS @ 10 OCT)
 6615  034102  000000                  11$:    0       ;BYTES OF DATA REC. (STARTS @ 10 OCT.)
 6616  034104  000000                  12$:    0       ;WATCH DOG COUNTER FOR DATA BYTES TRANSFERED
 6617  034106  000000                  13$:    0       ;ACTUAL DATA XMITTED (MIRRORED)
 6618  034110                          14$:
 6619  034110  000400                          BR      TST104          ;;
 6620
```

```
                                ;;*****************************************************************
                                        ;$CRCTGN
                                        ;THIS TEST WILL CHECK CRC-32 TRANSMIT GENERATE BY TRANSMITTING
                                        ; A ONE WORD BUFFER OF A WALKING ONE (1 TO 100000) DATA PATTERN
                                        ;AND CHECKING THE RECEIVED CRC32 DATA.  THE EXPECTED CRC FOR THE DATA
                                        ;MESSAGE (WORD) IS CALCULATED IN THE ROUTINE CALLED BY "JSR PCCRCCAL"
                                        ;

                                        ;THIS TEST WILL PERFORM THE FOLLOWING SEQUENCE

                                        ;1. ISSUE A MASTER CLEAR PULSE
                                        ;2. SET PROTOCOL=BOP
                                        ;3. TEST "TCRCE" AND "RCRCE" ARE NOT SET
                                        ;4. SET ECRC,RENA,TENA,MB
                                        ;5. SET TSOM
                                        ;6. STEP CLOCK 2 TICKS
                                        ;7. TEST "TBMT" FOR SET (RDY. FOR DATA)
                                        ;8. LOOP TO TRANSMIT AND RECEIVE THE DATA WORD
                                        ;9. SET TEOM
                                        ;10. LOOP TO STEP CLOCK AND RECEIVEHE CRC-32
                                        ;11. TEST RECEIVED CRC-32
                                        ;12. UPDATE DATA PATTERN AND REPEAT UNTIL DONE

                                        ;  TEST 104
                                        ;-------------
                                ;;*****************************************************************
034112  000004                  TST104: SCOPE
034114  012737  000104  001102          MOV     #104,$TSTNM             ; LOAD THE NO. OF THIS TEST
034122  012737  000104  001262          MOV     #104,TSTNUM             ;# FOR TYPE OUT
034130  105737  001256                  TSTB    CRC32F          ;CRC32 ENABLED?
034134  100402                          BMI     101$
034136  000137  034762                  JMP     16$
034142  012737  034164  001106  101$:   MOV     #1$,$LPADR                      ;SET RETURN ADDRESS (ERROR)
034150  012737  000001  034742          MOV     #1,8$                           ;INITIALIZE DATA (LOW WORD)
034156  012737  100000  034754          MOV     #100000,13$             ;INITIALIZE DATA XMITTED (15=LAST BIT OUT)
034164  104414                  1$:     MSTCLR
034166  012737  034752  034750          MOV     #12$,11$                ;POINT TO DATA XMITTED
034174  005037  034756                  CLR     14$                     ;KEEP A RECORD OF TIMESDATA READ
034200  104416  000005  000007          WRITE   ,SEC5,BOP!DISCRC                        ;SET BOP MODE & DISABLE CRC-16
034206  104416  000014  000131          WRITE   ,PRI14,RENA!TENA!ECRC!MB
034214  104416  000013  000000          WRITE   ,PRI13,0                ;GEN.MODE (CRC-32)
034222  104420  000014                  READ    ,PRI14          ;READ REGISTER
034226  012637  001202                  MOV     (SP)+,$TMP0     ;;POP STACK INTO $TMP0
034232  042737  177771  001202          BIC     #^C<TCRCE!RCRCE>,$TMP0  ;CLEAR UNWANTED BITS
034240  032737  000006  001202          BIT     #TCRCE!RCRCE,$TMP0              ;TEST TCRCE!RCRCE
034246  001422                          BEQ     2$                      ;RESET SKIP ERROR
034250  013737  001202  001204          MOV     $TMP0,$TMP1             ;PREPARE TO REPORT ERROR
034256  013737  001204  001206          MOV     $TMP1,$TMP2     ;LOAD EXPECTED ALSO
034264  042737  000006  001206          BIC     #TCRCE!RCRCE,$TMP2              ;LOAD EXPECTED
034272  013737  001202  001204          MOV     $TMP0,$TMP1
034300  042737  000006  001204          BIC     #RCRCE!TCRCE,$TMP1              ;SHOW BITS SHOULD BE CLEAR
034306  104047                          ERROR   39.
034310  000137  034164                  JMP     1$
034314  104416  000003  000001  2$:     WRITE   ,SEC3,TSOM                      ;SET THE TSOM
034322  104421  000004                  STEP    ,2*2                    ;STEP CLOCK 2 TICKS
034326  104416  000003  000000          WRITE   ,SEC3,0                 ;CLR TSOM
```

```
 6677  034334  012737  000004  034744          MOV     #4,9$                   ;BYTE OUT COUNTER FOR DATA
 6678  034342  117737  000402  034360  35$:    MOVB    @11$,4$         ;LOAD BYTE TO BE XMITTED
 6679  034350  005237  034750                  INC     11$
 6680  034354  104416  000002                  WRITE   ,SEC2
 6681  034360  000000                  4$:     0
 6682  034362  012737  000013  034760  43$:    MOV     #11.,15$        ;WATCH DOG THE TICKS
 6683  034370  104421  000002          44$:    STEP    ,2                      ;MONITOR TBMT FOR SETTING
 6684  034374  104420  000013                  READ    ,PRI13                  ;IS RDA SET?
 6685  034400  032726  000001                  BIT     #RDA,(SP)+
 6686  034404  001405                          BEQ     46$                     ;B=NO
 6687  034406  104420  000000                  READ    ,SEC0                   ;READ D.R.
 6688  034412  005726                          TST     (SP)+                   ;POP STACK
 6689  034414  005237  034756                  INC     14$
 6690  034420  005337  034760          46$:    DEC     15$                     ;MORE THAN 11. CLOCKS SINCE A TBMT
 6691  034424  001535                          BEQ     61$                     ;B=YES
 6692  034426  104420  000013                  READ    ,PRI13                  ;IS TBMT SET?
 6693  034432  032726  000020                  BIT     #TBMT,(SP)+
 6694  034436  001754                          BEQ     44$                     ;B=NO STEP AGAIN
 6695  034440  000337  034360                  SWAB    4$                      ;EXCHANGE DATA BYTES
 6696  034444  005337  034744                  DEC     9$                      ;ARE WE XMITTING DATA OR FLAG?
 6697  034450  003334                          BGT     35$                     ;B=DATA
 6698  034452  002404                          BLT     47$                     ;B=FLAG SENT
 6699  034454  104416  000003  000002          WRITE   ,SEC3,TEOM              ;SET TEOM
 6700  034462  000737                          BR      43$                     ;GO CLOCK TEOM OUT
 6701  034464  104416  000003  000000  47$:    WRITE   ,SEC3,0                 ;CLR TEOM
 6702
 6703                                  ;CLOCK FLAG AND DATA UP TO CRC CHRS.
 6704  034472  012737  000004  034744          MOV     #4,9$                   ;BYTE COUNTER
 6705  034500  012737  000013  034760  42$:    MOV     #11.,15$                ;WATCH DOG DATA XFER
 6706  034506  104421  000002          45$:    STEP    ,2                      ;ZEROS MAY BE INSERTED AND STRPPED BY
 6707  034512  005337  034760                  DEC     15$                     ;TOO MANY TICKS?
 6708  034516  001500                          BEQ     61$                     ;B=YES
 6709  034520  104420  000013                  READ    ,PRI13                  ;USYRT SO ADDITIONAL CLOCK TICKS MUST
 6710  034524  032726  000001                  BIT     #RDA,(SP)+              ;BE COMPENSATED FOR - IS RDA SET?
 6711  034530  001766                          BEQ     45$                     ;B=NO STEP AGAIN
 6712  034532  104420  000000                  READ    ,SEC0                   ;READ DATA REGISTER
 6713  034536  005726                          TST     (SP)+                   ;POP STACK
 6714  034540  005237  034756                  INC     14$
 6715  034544  005337  034744                  DEC     9$
 6716  034550  001353                          BNE     42$                     ;STEP FOR NEXT CHR.
 6717                                  ;STORE NEXT 4 (8-BIT)BYTES
 6718  034552  012737  000004  034744          MOV     #4,9$                   ;BYTE COUNTER
 6719  034560  012737  001216  034746          MOV     #$TMP6,10$              ;DATA STOREAGE = $TMP6 AND $TMP7
 6720  034566  012737  000012  034760  51$:    MOV     #10.,15$
 6721  034574  104421  000002          5$:     STEP    ,2                      ;MONITOR RDA FOR DATA BYTE COMPLETION
 6722  034600  005337  034760                  DEC     15$                     ;TOO MANY TICKS?
 6723  034604  001445                          BEQ     61$                     ;B=YES
 6724  034606  104420  000013                  READ    ,PRI13                  ;IS RDA SET?
 6725  034612  032726  000001                  BIT     #RDA,(SP)+
 6726  034616  001766                          BEQ     5$                      ;B=RDA IS NOT SET
 6727  034620  104420  000000                  READ    ,SEC0                   ;READ CRC CHARACTER
 6728  034624  111677  000116                  MOVB    (SP),@10$               ;STORE CRC BYTE
 6729  034630  005237  034746                  INC     10$
 6730  034634  005726                          TST     (SP)+                   ;POP STACK
 6731  034636  005237  034756                  INC     14$
 6732  034642  005337  034744                  DEC     9$                      ;DONE?
```

```
6733  034646  001347                          BNE     51$                     ;B=NO
6734                                          ;CALCULATE THE CRC-32 FORHE CURRENT DATA MESSAGE
6735                                          ;AND RETURN WITH CRC IN $TMP4 & $TMP5
6736  034650  013702  034742                  MOV     8$,R2                   ;LOAD DATA FOR CRC CALU.
6737  034654  005003                          CLR     R3
6738  034656  004737  047462                  JSR     PC,CRCCAL               ;CALCULATE THE CRC
6739                                          ;TEST THAT THE GENERATED CRC IS SAME AS CALCULATED
6740  034662  023737  001214  001216          CMP     $TMP5,$TMP6             ;IS HIGH ORDER WORD CORRECT?
6741  034670  001016                          BNE     6$                      ;B=NO
6742  034672  023737  001212  001220          CMP     $TMP4,$TMP7             ;IS LOW ORDER WORD CORRECT?
6743  034700  001012                          BNE     6$                      ;B=NO
6744  034702  006337  034742                  ASL     8$                      ;DONE?
6745  034706  103425                          BCS     16$                     ;B=YES
6746  034710  006037  034754                  ROR     13$                     ;ALLIGN DATA XMITTED
6747  034714  000137  034164                  JMP     1$                      ;DO NEXT DATA VALUE
6748
6749  034720  104051                  61$:    ERROR   41.
6750  034722  000137  034164                  JMP     1$
6751
6752  034726  013737  034754  001222  6$:     MOV     13$,$TMP10
6753  034734  104050                          ERROR   40.                     ;CRC GENERATED IS NOT CORRECT
6754  034736  000137  034164                  JMP     1$
6755  034742  000000                  8$:     0
6756  034744  000000                  9$:     0
6757  034746  000000                  10$:    0
6758  034750  000000                  11$:    0                               ;POINTER FOR DATA XMITTED
6759  034752  000000                  12$:    0                               ;1ST. 2 DATA CHRS.
6760  034754  000000                  13$:    0                               ;2ND. 2 DATA CHRS.
6761  034756  000000                  14$:    0                               ;# TIMES D.R. READ
6762  034760  000000                  15$:    0
6763  034762                          16$:
6764  034762  000400                          BR      TST105          ;;
6765
6766                                          ;ENTER ROUTINE TO LOOP FOR NEXT LINE
6767
6768                                          ;  TEST 105
6769                                          ;--------------
6770                                  ;;**************************************************************************
6771  034764  000004                  TST105: SCOPE
6772  034766  012737  000105  001102          MOV     #105,$TSTNM             ; LOAD THE NO. OF THIS TEST
6773  034774  012737  000105  001262          MOV     #105,TSTNUM             ;# FOR TYPE OUT
6774  035002  000137  002576                  JMP     CYCLE
6775
6776                                  .SBTTL  END OF PASS ROUTINE
6777
6778                                  ;;**************************************************************************
6779                                  ;*INCREMENT THE PASS NUMBER ($PASS)
6780                                  ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
6781                                  ;*IF THERES A MONITOR GO TO IT
6782                                  ;*IF THERE ISN'T JUMP TO NXTUN
6783
6784  035006                          $EOP:
6785  035006  000004                          SCOPE
6786  035010  005037  001102                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
6787  035014  005237  001100                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
6788  035020  042737  100000  001100          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
```

```
 6789  035026  005327                          DEC     (PC)+           ;;LOOP?
 6790  035030  000001                  $EOPCT: .WORD   1
 6791  035032  003022                          BGT     $DOAGN          ;;YES
 6792  035034  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
 6793  035036  000001                  $ENDCT: .WORD   1
 6794  035040  035030                          $EOPCT
 6795  035042  104401  035107                  TYPE    ,$ENDMG         ;;TYPE 'END PASS #''
 6796  035046  013746  001100                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
 6797  035052  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
 6798  035054  104401  035104                  TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
 6799  035060  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
 6800  035064  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
 6801  035066  000005                          RESET                   ;;CLEAR THE WORLD
 6802  035070  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
 6803  035072  000240                          NOP                     ;;SAVE ROOM
 6804  035074  000240                          NOP                     ;;FOR
 6805  035076  000240                          NOP                     ;;ACT11
 6806  035100                          $DOAGN:
 6807  035100  000137                          JMP     @(PC)+          ;;RETURN
 6808  035102  002512                  $RTNAD: .WORD   NXTUN
 6809  035104     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
 6810  035107     015  042412  042116  $ENDMG: .ASCIZ  <15><12>/END PASS #/
 6811  035114  050040  051501  020123
 6812  035122  000043
```

```
 6813                                           .SBTTL  ERROR MESSAGES
 6814  035124  051120  046511  051101  EM1:    .ASCIZ  /PRIMARY REGISTER ERROR/
 6815  035132  020131  042522  044507
 6816  035140  052123  051105  042440
 6817  035146  051122  051117     000
 6818  035153     106  047514  052101  EM2:    .ASCIZ  /FLOATING ONE'S ERROR/
 6819  035160  047111  020107  047117
 6820  035166  023505  020123  051105
 6821  035174  047522  000122
 6822  035200  042523  047503  042116  EM3:    .ASCIZ  /SECONDARY REGISTER READ ERROR/
 6823  035206  051101  020131  042522
 6824  035214  044507  052123  051105
 6825  035222  051040  040505  020104
 6826  035230  051105  047522  000122
 6827  035236  044124  020105  047506  EM4:    .ASCIZ  /THE FOLLOWING READ ONLY BIT CAN BE WRITTEN/
 6828  035244  046114  053517  047111
 6829  035252  020107  042522  042101
 6830  035260  047440  046116  020131
 6831  035266  044502  020124  040503
 6832  035274  020116  042502  053440
 6833  035302  044522  052124  047105
 6834  035310     000
 6835  035311     124  040522  051516  EM5:    .ASCIZ  /TRANSMIT STATUS REGISTER BIT FAILURE/
 6836  035316  044515  020124  052123
 6837  035324  052101  051525  051040
 6838  035332  043505  051511  042524
 6839  035340  020122  044502  020124
 6840  035346  040506  046111  051125
 6841  035354  000105
 6842  035356  047515  042504  041440  EM6:    .ASCIZ  /MODE CONTROL REGISTER BIT FAILURE/
 6843  035364  047117  051124  046117
 6844  035372  051040  043505  051511
 6845  035400  042524  020122  044502
 6846  035406  020124  040506  046111
 6847  035414  051125  000105
 6848  035420  044103  051101  041501  EM7:    .ASCIZ  /CHARACTER LENGTH REGISTER BIT FAILURE/
 6849  035426  042524  020122  042514
 6850  035434  043516  044124  051040
 6851  035442  043505  051511  042524
 6852  035450  020122  044502  020124
 6853  035456  040506  046111  051125
 6854  035464  000105
 6855  035466  051124  047101  046523  EM8:    .ASCIZ  /TRANSMIT STATUS REGISTER BIT FAILED TO CLEAR AFTER LINE RESET/
 6856  035474  052111  051440  040524
 6857  035502  052524  020123  042522
 6858  035510  044507  052123  051105
 6859  035516  041040  052111  043040
 6860  035524  044501  042514  020104
 6861  035532  047524  041440  042514
 6862  035540  051101  040440  052106
 6863  035546  051105  046040  047111
 6864  035554  020105  042522  042523
 6865  035562  000124
 6866  035564  047515  042504  041440  EM9:    .ASCIZ  /MODE CONTROL REGISTER BIT FAILED TO CLEAR AFTER LINE RESET/
 6867  035572  047117  051124  046117
 6868  035600  051040  043505  051511
```

```
 6869  035606  042524  020122  044502
 6870  035614  020124  040506  046111
 6871  035622  042105  052040  020117
 6872  035630  046103  040505  020122
 6873  035636  043101  042524  020122
 6874  035644  044514  042516  051040
 6875  035652  051505  052105     000
 6876  035657     123  041505  047117  EM10:   .ASCIZ  /SECONDARY INCREMENT ONE'S TEST FAILURE/
 6877  035664  040504  054522  044440
 6878  035672  041516  042522  042515
 6879  035700  052116  047440  042516
 6880  035706  051447  052040  051505
 6881  035714  020124  040506  046111
 6882  035722  051125  000105
 6883  035726  042523  047503  042116  EM11:   .ASCIZ  /SECONDARY REGISTER DID NOT CLEAR AFTER LINE RESET/
 6884  035734  051101  020131  042522
 6885  035742  044507  052123  051105
 6886  035750  042040  042111  047040
 6887  035756  052117  041440  042514
 6888  035764  051101  040440  052106
 6889  035772  051105  046040  047111
 6890  036000  020105  042522  042523
 6891  036006  000124
 6892  036010  041124  052115  047040  EM12:   .ASCIZ  /TBMT NOT RESET AFTER TSOM SET WITH TENA SET/
 6893  036016  052117  051040  051505
 6894  036024  052105  040440  052106
 6895  036032  051105  052040  047523
 6896  036040  020115  042523  020124
 6897  036046  044527  044124  052040
 6898  036054  047105  020101  042523
 6899  036062  000124
 6900  036064  041124  052115  043040  EM13:   .ASCIZ  /TBMT FAILED TO SET/
 6901  036072  044501  042514  020104
 6902  036100  047524  051440  052105
 6903  036106     000
 6904  036107     123  043057  042040  EM14:   .ASCIZ  /S/<'/>/F DID NOT SET AFTER FLAG SENT/
 6905  036114  042111  047040  052117
 6906  036122  051440  052105  040440
 6907  036130  052106  051105  043040
 6908  036136  040514  020107  042523
 6909  036144  052116     000
 6910  036147     123  043057  042040  EM15:   .ASCIZ  /S/<'/>/F DID NOT SET AFTER SYNC SENT/
 6911  036154  042111  047040  052117
 6912  036162  051440  052105  040440
 6913  036170  052106  051105  051440
 6914  036176  047131  020103  042523
 6915  036204  052116     000
 6916  036207     123  043057  042040  EM16:   .ASCIZ  /S/<'/>/F DID NOT RESET/
 6917  036214  042111  047040  052117
 6918  036222  051040  051505  052105
 6919  036230     000
 6920  036231     122  041501  043040  EM17:   .ASCIZ  /RAC FAILED TO SET AFTER 3 DATA BYTES TRANSMITTED/
 6921  036236  044501  042514  020104
 6922  036244  047524  051440  052105
 6923  036252  040440  052106  051105
 6924  036260  031440  042040  052101
```

```
 6925  036266  020101  054502  042524
 6926  036274  020123  051124  047101
 6927  036302  046523  052111  042524
 6928  036310  000104
 6929  036312  042122  020101  040506  EM18:   .ASCIZ  /RDA FAILED TO SET AFTER 3 DATA BYTES TRANSFERED/
 6930  036320  046111  042105  052040
 6931  036326  020117  042523  020124
 6932  036334  043101  042524  020122
 6933  036342  020063  040504  040524
 6934  036350  041040  052131  051505
 6935  036356  052040  040522  051516
 6936  036364  042506  042522  000104
 6937  036372  051522  046517  043040  EM19:   .ASCIZ  /RSOM FAILED TO SET/
 6938  036400  044501  042514  020104
 6939  036406  047524  051440  052105
 6940  036414     000
 6941  036415     106  051111  052123  EM20:   .ASCIZ  /FIRST DATA BYTE TRANSMITTED IN ERROR/
 6942  036422  042040  052101  020101
 6943  036430  054502  042524  052040
 6944  036436  040522  051516  044515
 6945  036444  052124  042105  044440
 6946  036452  020116  051105  047522
 6947  036460  000122
 6948  036462  042522  046517  043040  EM21:   .ASCIZ  /REOM FAILED TO SET AFTER TEOM SET/
 6949  036470  044501  042514  020104
 6950  036476  047524  051440  052105
 6951  036504  040440  052106  051105
 6952  036512  052040  047505  020115
 6953  036520  042523  000124
 6954  036524  044514  042516  043040  EM22:   .ASCIZ  /LINE FAILED TO RESYNC AFTER TEOM SET WITH IDLE RESET/
 6955  036532  044501  042514  020104
 6956  036540  047524  051040  051505
 6957  036546  047131  020103  043101
 6958  036554  042524  020122  042524
 6959  036562  046517  051440  052105
 6960  036570  053440  052111  020110
 6961  036576  042111  042514  051040
 6962  036604  051505  052105     000
 6963  036611     104  052101  020101  EM23:   .ASCIZ  /DATA TRANSMITTED INCORRECTLY/
 6964  036616  051124  047101  046523
 6965  036624  052111  042524  020104
 6966  036632  047111  047503  051122
 6967  036640  041505  046124  000131
 6968  036646  051105  047522  020122  EM24:   .ASCIZ  /ERROR CHECK BIT FAILED TO SET/
 6969  036654  044103  041505  020113
 6970  036662  044502  020124  040506
 6971  036670  046111  042105  052040
 6972  036676  020117  042523  000124
 6973  036704  051105  047522  020122  EM25:   .ASCIZ  /ERROR CHECK BIT SET AND SHOULD NOT HAVE/
 6974  036712  044103  041505  020113
 6975  036720  044502  020124  042523
 6976  036726  020124  047101  020104
 6977  036734  044123  052517  042114
 6978  036742  047040  052117  044040
 6979  036750  053101  000105
 6980  036754  047516  052040  040522  EM26:   .ASCIZ  /NO TRANSMIT ERROR ON A FORCE UNDERRUN/
```

```
 6981  036762  051516  044515  020124
 6982  036770  051105  047522  020122
 6983  036776  047117  040440  043040
 6984  037004  051117  042503  052440
 6985  037012  042116  051105  052522
 6986  037020  000116
 6987  037022  047516  040440  047502  EM27:   .ASCIZ  /NO ABORT INDICATIONS RECEIVED ON A FORCE UNDERRUN/
 6988  037030  052122  044440  042116
 6989  037036  041511  052101  047511
 6990  037044  051516  051040  041505
 6991  037052  044505  042526  020104
 6992  037060  047117  040440  043040
 6993  037066  051117  042503  052440
 6994  037074  042116  051105  052522
 6995  037102  000116
 6996  037104  044514  042516  042040  EM28:   .ASCIZ  /LINE DID NOT RESYNCE ON A FORCE UNDERRUN/
 6997  037112  042111  047040  052117
 6998  037120  051040  051505  047131
 6999  037126  042503  047440  020116
 7000  037134  020101  047506  041522
 7001  037142  020105  047125  042504
 7002  037150  051122  047125     000
 7003  037155     122  041505  044505  EM29:   .ASCIZ  /RECEIVER OVERRUN DID NOT SET ON A FORCE OVERRUN/
 7004  037162  042526  020122  053117
 7005  037170  051105  052522  020116
 7006  037176  044504  020104  047516
 7007  037204  020124  042523  020124
 7008  037212  047117  040440  043040
 7009  037220  051117  042503  047440
 7010  037226  042526  051122  047125
 7011  037234     000
 7012  037235     122  041505  044505  EM30:   .ASCIZ  /RECEIVER OVERRUN DID NOT RESET AFTER STAUS READ/
 7013  037242  042526  020122  053117
 7014  037250  051105  052522  020116
 7015  037256  044504  020104  047516
 7016  037264  020124  042522  042523
 7017  037272  020124  043101  042524
 7018  037300  020122  052123  052501
 7019  037306  020123  042522  042101
 7020  037314     000
 7021  037315     116  020117  042522  EM31:   .ASCIZ  /NO RECEIVE ABORT ON A TRANSMIT OF A ABORT/
 7022  037322  042503  053111  020105
 7023  037330  041101  051117  020124
 7024  037336  047117  040440  052040
 7025  037344  040522  051516  044515
 7026  037352  020124  043117  040440
 7027  037360  040440  047502  052122
 7028  037366     000
 7029  037367     116  020117  042522  EM32:   .ASCIZ  /NO RECEIVE GO AHEAD ON A TRANSMIT OF GO AHEAD/
 7030  037374  042503  053111  020105
 7031  037402  047507  040440  042510
 7032  037410  042101  047440  020116
 7033  037416  020101  051124  047101
 7034  037424  046523  052111  047440
 7035  037432  020106  047507  040440
 7036  037440  042510  042101     000
```

```
 7037  037445     104  053105  041511  EM33:   .ASCIZ  /DEVICE IS STILL TRYING TO SYNC/
 7038  037452  020105  051511  051440
 7039  037460  044524  046114  052040
 7040  037466  054522  047111  020107
 7041  037474  047524  051440  047131
 7042  037502  000103
 7043  037504  044124  020105  046513  EM34:   .ASCIZ  /THE KMC11 IS HUNG/
 7044  037512  030503  020061  051511
 7045  037520  044040  047125  000107
 7046  037526  047516  051440  043057  EM35:   .ASCIZ  /NO S/<'/>/F AFTER TEOM WAS SET/
 7047  037534  040440  052106  051105
 7048  037542  052040  047505  020115
 7049  037550  040527  020123  042523
 7050  037556  000124
 7051  037560  027523  020106  042523  EM36:   .ASCIZ  /S/<'/>/F SET BEFORE LAST DATA BYTE RECEIVED/
 7052  037566  020124  042502  047506
 7053  037574  042522  046040  051501
 7054  037602  020124  040504  040524
 7055  037610  041040  052131  020105
 7056  037616  042522  042503  053111
 7057  037624  042105     000
 7058  037627     122  040523  042040  EM37:   .ASCIZ  /RSA DID NOT SET ON A TRANSMIT UNDERRUN/
 7059  037634  042111  047040  052117
 7060  037642  051440  052105  047440
 7061  037650  020116  020101  051124
 7062  037656  047101  046523  052111
 7063  037664  052440  042116  051105
 7064  037672  052522  000116
 7065  037676  051103  026503  031063  EM38:   .ASCIZ  /CRC-32 ERROR STATUS BITS (PR14-1&2)/
 7066  037704  042440  051122  051117
 7067  037712  051440  040524  052524
 7068  037720  020123  044502  051524
 7069  037726  024040  051120  032061
 7070  037734  030455  031046  000051
 7071  037742  051103  026503  031063  EM39:   .ASCIZ  /CRC-32 TRANSFER CAUSED STATUS ERROR/
 7072  037750  052040  040522  051516
 7073  037756  042506  020122  040503
 7074  037764  051525  042105  051440
 7075  037772  040524  052524  020123
 7076  040000  051105  047522  000122
 7077  040006  051103  026503  031063  EM40:   .ASCIZ  /CRC-32 GENERATED IS NOT CORRECT/
 7078  040014  043440  047105  051105
 7079  040022  052101  042105  044440
 7080  040030  020123  047516  020124
 7081  040036  047503  051122  041505
 7082  040044  000124
 7083  040046  051042  040504  020042  EM42:   .ASCIZ  /"RDA" FAILED TO SET ON DATA TRANSFER/
 7084  040054  040506  046111  042105
 7085  040062  052040  020117  042523
 7086  040070  020124  047117  042040
 7087  040076  052101  020101  051124
 7088  040104  047101  043123  051105
 7089  040112     000
 7090  040113     124  040522  051516  EM41:   .ASCIZ  /TRANSFER INCOMPLETE /
 7091  040120  042506  020122  047111
 7092  040126  047503  050115  042514
```

```
 7093  040134  042524  000040
 7094  040140  020040  042524  052123  DH1:    .ASCIZ  /  TEST    PC     KMC11   LINE    REG     EXP     REC/
 7095  040146  020040  020040  041520
 7096  040154  020011  045440  041515
 7097  040162  030461  020011  046040
 7098  040170  047111  004505  020040
 7099  040176  042522  004507  020040
 7100  040204  054105  004520  020040
 7101  040212  042522  000103
 7102  040216  020040  042524  052123  DH2:    .ASCIZ  /  TEST    PC     KMC11   LINE    EXP     REC/
 7103  040224  020040  020040  041520
 7104  040232  020011  045440  041515
 7105  040240  030461  020011  046040
 7106  040246  047111  004505  020040
 7107  040254  054105  004520  020040
 7108  040262  042522  000103
 7109  040266  020040  042524  052123  DH4:    .ASCIZ  /  TEST    PC     KMC11   LINE/
 7110  040274  020040  020040  041520
 7111  040302  020011  045440  041515
 7112  040310  030461  020011  046040
 7113  040316  047111  000105
 7114  040322  020040  042524  052123  DH5:    .ASCIZ  /  TEST    PC     KMC11   LINE    PRI12/
 7115  040330  020040  020040  041520
 7116  040336  020011  045440  041515
 7117  040344  030461  020011  046040
 7118  040352  047111  004505  020040
 7119  040360  051120  030511  000062
 7120  040366  020040  042524  052123  DH6:    .ASCIZ  /  TEST    PC     KMC11   LINE    EXP     REC/
 7121  040374  020040  020040  041520
 7122  040402  020011  045440  041515
 7123  040410  030461  020011  046040
 7124  040416  047111  004505  020040
 7125  040424  054105  004520  020040
 7126  040432  042522  000103
 7127  040436  020040  042524  052123  DH7:    .ASCIZ  /  TEST    PC     KMC11   LINE    CRC-HI  CRC-LO  PRI14   PRI14(SHBE)/
 7128  040444  020040  020040  041520
 7129  040452  045411  041515  030461
 7130  040460  046011  047111  004505
 7131  040466  051103  026503  044510
 7132  040474  041411  041522  046055
 7133  040502  004517  051120  030511
 7134  040510  004464  051120  030511
 7135  040516  024064  044123  042502
 7136  040524  000051
 7137  040526  020040  042524  052123  DH8:    .ASCIZ  /  TEST    PC     KMC11   LINE    SHBE-HI SHBE-LO WAS-HI  WAS-LO  DATA/
 7138  040534  020040  020040  041520
 7139  040542  045411  041515  030461
 7140  040550  046011  047111  004505
 7141  040556  044123  042502  044055
 7142  040564  004511  044123  042502
 7143  040572  046055  004517  040527
 7144  040600  026523  044510  053411
 7145  040606  051501  046055  020117
 7146  040614  020040  040504  040524
 7147  040622     000
 7148          040624                          .EVEN
```

```
 7149  040624  001262  001116  001246  DT1:    .WORD   TSTNUM,$ERRPC,$DVICE,CLINED,REGIST,$TMP1,$TMP0,0
 7150  040632  001244  001240  001204
 7151  040640  001202  000000
 7152  040644  001262  001116  001246  DT2:    .WORD   TSTNUM,$ERRPC,$DVICE,CLINED,$TMP1,$TMP0,0
 7153  040652  001244  001204  001202
 7154  040660  000000
 7155  040662  001262  001116  001246  DT3:    .WORD   TSTNUM,$ERRPC,$DVICE,CLINED,$TMP2,$TMP1,0
 7156  040670  001244  001206  001204
 7157  040676  000000
 7158  040700  001262  001116  001246  DT4:    .WORD   TSTNUM,$ERRPC,$DVICE,CLINED,0
 7159  040706  001244  000000
 7160  040712  001262  001116  001246  DT5:    .WORD   TSTNUM,$ERRPC,$DVICE,CLINED,$TMP0,0
 7161  040720  001244  001202  000000
 7162  040726  001262  001116  001246  DT6:    .WORD   TSTNUM,$ERRPC,$DVICE,CLINED,$TMP0,$TMP1,0
 7163  040734  001244  001202  001204
 7164  040742  000000
 7165  040744  001262  001116  001246  DT7:    .WORD   TSTNUM,$ERRPC,$DVICE,CLINED,$TMP4,$TMP5,$TMP0,$TMP2,0
 7166  040752  001244  001212  001214
 7167  040760  001202  001206  000000
 7168  040766  001262  001116  001246  DT8:    .WORD   TSTNUM,$ERRPC,$DVICE,CLINED,$TMP5,$TMP4,$TMP6,$TMP7,$TMP10,0
 7169  040774  001244  001214  001212
 7170  041002  001216  001220  001222
 7171  041010  000000
 7172  041012     000     000     001  DF1:    .BYTE   0,0,1,1,0,0,0,0
 7173  041015     001     000     000
 7174  041020     000     000
 7175  041022     000     000     001  DF2:    .BYTE   0,0,1,1,0,0
 7176  041025     001     000     000
 7177  041030     000     000     001  DF4:    .BYTE   0,0,1,1
 7178  041033     001
 7179  041034     000     000     001  DF5:    .BYTE   0,0,1,1,0
 7180  041037     001     000
 7181  041041     000     000     001  DF6:    .BYTE   0,0,1,1,0,0
 7182  041044     001     000     000
 7183  041047     000     000     001  DF7:    .BYTE   0,0,1,1,0,0,0,0
 7184  041052     001     000     000
 7185  041055     000     000
 7186          041060                          .EVEN
 7187                                          ;ROUTINE USED TO LOAD OR MODIFY THE KMC11 PARAMETER STATUS BLOCKS.
 7188                                          ;TWO ENTRY POINTS ARE USED, ONE AS THE INITIAL START-UP AND THE
 7189                                          ;SECOND AS THE MODIFY TABLE ROUTINE POINT.
 7190                                          ;WHEN INITIALLY ENTERED FROM THE MONITOR ROUTINE THE FIRST
 7191                                          ;PARAMETER BLOCKS STATUS IS INVOLKED FROM THE CONSOLE TO
 7192                                          ;WHICH THE OPERATOR MUST INPUT THE FIRST DEVICE ADDRESS INTERRUPT
 7193                                          ;VECTOR, PRIORITY AND WHETHER THE DEVICE IS TO BE ENABLED OR
 7194                                          ;DISABLED.  AFTER THE FIRST KMC11 IS SPECIFIED THE ROUTINE
 7195                                          ;ENTERS THE CONFIGURE MODE AS SPECIFIED BY THE ASTERISK (* OPERATOR THEN HAS THE
 7196                                          ;
 7197                                          ; NB = INCLUDE INTERRUPT VEC. + PRI.
 7198                                          ;       *M#     ;#=1->16., OPENS THE PARAMETER BLOCK AND INVOLKS
 7199                                          ;               ;THE CHANGE OF STATUS (AFTER INPUTTING STATUS
 7200                                          ;               ;THIS THEN IS LAST BLOCK OPEN)
 7201                                          ;
 7202                                          ;
 7203                                          ;       *A (CR) ;FILLS REMAINDER OF TABLE WITH SUCCESIVE DEV ADR
 7204                                          ;               ;AND IDENTICAL STATUS AS/PER LAST (THE LAST STATUS BLOCK
```

```
7205                                                    ;               ;ENTERED) *NOTE IF ISSUED AS FIRST COMMAND IN 'MODIFY
7206                                                    ;               ;MODE THE TABLE IS FILLED AS/THE FIRST KMC11.
7207                                                    ;               ;
7208                                                    ;       *E (CR) ;EXITS TO THE CALLING ROUTINE I.E. (MONITOR)
7209                                                    ;               ;
7210                                                    ;               ;
7211  041060  000000                        CHRADR:  0                      ;HOLD REC. TTY CHR.
7212  041062  000000                        CUNIT:   0                      ;POINTS TO CURRENT UNIT OPEN
7213  041064  000000                        KMCNT:   0                      ;# OF KMC11'S ENABLED
7214  041066  000000                        CONFIG:  0
7215  041070  000000                        CURTAB:  0                      ;CURRENT BLOCK OPEN
7216  041072  012701  042466                SETTAB:  MOV     #KMCTAB,R1     ;POINT TO PARAMETER TABLE-BLOCK 1
7217  041076  005021                        1$:      CLR     (R1)+          ;CLEAN THE MESS
7218  041100  020127  042626                         CMP     R1,#KMCTBE     ;DONE?
7219  041104  001374                                 BNE     1$             ;BR=N
7220  041106  012701  042466                         MOV     #KMCTAB,R1
7221  041112  010137  041070                         MOV     R1,CURTAB
7222  041116  000137  041166                         JMP     PT
7223                                                 ;MODIFY TABLE ROUTINE ENTRY POINT
7224  041122                                NOTR:
7225  041122  104401  041130                         TYPE    ,64$           ;;TYPE ASCIZ STRING
7226  041126  000402                                 BR      MODTAB         ;;GET OVER THE ASCIZ
7227                                        ;;64$:   .ASCIZ  /??/
7228  041134                                MODTAB:
7229  041134  005737  041066                MODTAB:  TST     CONFIG         ;FIRST TIME?
7230  041140  001002                                 BNE     1$             ;BR=NO
7231  041142  000137  041072                         JMP     SETTAB
7232  041146                                1$:
7233  041146  104401  041154                         TYPE    ,65$           ;;TYPE ASCIZ STRING
7234  041152  000404                                 BR      64$            ;;GET OVER THE ASCIZ
7235                                        ;;65$:   .ASCIZ  <15><12>/*   /
7236  041164                                64$:
7237  041164  000401                                 BR      PT+2
7238  041166  000555                        PT:      BR      5$             ;GET FIRST SETUP
7239
7240
7241  041170  104410                                 RDCHR
7242  041172  011637  041060                         MOV     (SP),CHRADR    ;ECHO
7243  041176  104401  041060                         TYPE    ,CHRADR
7244  041202  022726  000115                         CMP     #115,(SP)+     ;IS IT A M?
7245  041206  001433                                 BEQ     25$            ;BR=YES
7246  041210  022746  000101                         CMP     #101,-(SP)     ;IS IT AN A?
7247  041214  001002                                 BNE     111$
7248  041216  000137  042342                         JMP     12$
7249  041222  122726  000105                111$:    CMPB    #105,(SP)+     ;IS IT AN E?
7250  041226  001401                                 BEQ     113$           ;B=Y GO!
7251  041230  000734                                 BR      NOTR
7252                                                 ;FIND # OF KMC11'S ACTIVE AND STORE IN 'KMCNT''
7253  041232  005037  041064                113$:    CLR     KMCNT
7254  041236  012701  042466                         MOV     #KMCTAB,R1
7255  041242  005761  000002                1$:      TST     2(R1)                  ;IS UNIT ENABLED?
7256  041246  100005                                 BPL     2$             ;BR=NO
7257  041250  105761  000002                         TSTB    2(R1)          ;ARE LINES ENABLED?
7258  041254  001402                                 BEQ     2$             ;B=NO
7259  041256  005237  041064                         INC     KMCNT          ;YES
7260  041262  062701  000006                2$:      ADD     #6,R1
```

```
7261  041266  020127  042626                  CMP     R1,#KMCTBE      ;DONE?
7262  041272  001363                          BNE     1$              ;B=N
7263  041274  000207                          RTS     PC              ;YES
7264
7265                                          ;A 'M' HAS BEEN TYPED READ THE DECIMAL # FOLLOWING
7266  041276                          25$:
7267  041276  104413                          RDDEC
7268  041300  012600                          MOV     (SP)+,R0        ;IS # +?
7269  041302  010037  041062                  MOV     R0,CUNIT        ;PRINT LINE UNIT #
7270  041306  010001                          MOV     R0,R1
7271  041310  003704                          BLE     NOTR
7272  041312  120027  000020                  CMPB    R0,#16.         ;IS #<16.?
7273  041316  003301                          BGT     NOTR            ;BR=NO
7274  041320  000241                          CLC
7275  041322  005000                          CLR     R0
7276  041324  062700  000006          26$:    ADD     #6.,R0          ;MULTI X 6.
7277  041330  005301                          DEC     R1
7278  041332  001374                          BNE     26$
7279  041334  062700  042460                  ADD     #KMCTAB-6,R0
7280  041340  010037  041070                  MOV     R0,CURTAB
7281  041344  104401  041352                  TYPE    ,65$            ;;TYPE ASCIZ STRING
7282  041350  000432                          BR      64$             ;;GET OVER THE ASCIZ
7283                                  ;;65$:  .ASCIZ  <15><12>!(DEV.ADR.) (ENABL.-LINES) (CRC/32-EXT/INT) UNIT#!
7284  041436                          64$:
7285
7286                                          ;ROUTINE TO DISPLAY CONTENTS OF CURRENT BLOCK
7287  041436                          3$:
7288  041436  104401  041444                  TYPE    ,67$            ;;TYPE ASCIZ STRING
7289  041442  000403                          BR      66$             ;;GET OVER THE ASCIZ
7290                                  ;;67$:  .ASCIZ  <15><12>/  /
7291  041452                          66$:
7292  041452  012703  000003                  MOV     #3,R3
7293  041456  012046                  4$:     MOV     (R0)+,-(SP)
7294  041460  104403                          TYPOS
7295  041462     006                          .BYTE   6
7296  041463     001                          .BYTE   1               ;NO SUPPRESS
7297  041464  104401  041472                  TYPE    ,69$            ;;TYPE ASCIZ STRING
7298  041470  000404                          BR      68$             ;;GET OVER THE ASCIZ
7299                                  ;;69$:  .ASCIZ  /      /
7300  041502                          68$:
7301  041502  005303                          DEC     R3
7302  041504  001401                          BEQ     45$
7303  041506  000763                          BR      4$
7304  041510  013746  041062          45$:    MOV     CUNIT,-(SP)
7305  041514  104405                          TYPDS
7306  041516  013701  041070                  MOV     CURTAB,R1
7307
7308  041522                          5$:     ;ROUTINE TO MODIFY PARAMETER BLOCK
7309
7310  041522  104401  041530                  TYPE    ,71$            ;;TYPE ASCIZ STRING
7311  041526  000406                          BR      70$             ;;GET OVER THE ASCIZ
7312                                  ;;71$:  .ASCIZ  <15><12>/DEV.ADR.=/
7313  041544                          70$:
7314  041544  004737  042236                  JSR     PC,8$
7315  041550  104401  042412                  TYPE    ,120$
7316  041554  004737  042030                  JSR     PC,7$                   ;GET LINE ENABLE BYTE
```

```
7317  041560  104401  041566            TYPE    ,73$            ;;TYPE ASCIZ STRING
7318  041564  000413                    BR      ,72$            ;;GET OVER THE ASCIZ
7319                              ;;73$:  .ASCIZ  /(1)ENABLE(0)DISABLE /
7320  041614                      ,72$:
7321  041614  004737  042252            JSR     PC,10$
7322  041620  104401  041626            TYPE    ,75$            ;;TYPE ASCIZ STRING
7323  041624  000416                    BR      ,74$            ;;GET OVER THE ASCIZ
7324                              ;;75$:  .ASCIZ  <15><12>/(0)EXTERNAL (1)INTERNAL /
7325  041662                      ,74$:
7326  041662  004737  042252            JSR     PC,10$
7327  041666  152741  000100            BISB    #100,-(R1)              ;INSERT OTHER CLOCK BIT
7328  041672  105721                    TSTB    (R1)+                   ;UPDATE POINTER ,TEST FOR ENABLING CRC32
7329  041674                      50$:
7330  041674  104401  041702            TYPE    ,77$            ;;TYPE ASCIZ STRING
7331  041700  000415                    BR      ,76$            ;;GET OVER THE ASCIZ
7332                              ;;77$:  .ASCIZ  <15><12>/(1) CRC32 (0) NO CRC32 /
7333  041734                      ,76$:
7334  041734  004737  042252            JSR     PC,10$
7335  041740  105741                    TSTB    -(R1)           ;WITH CRC-32 ?
7336  041742  100025                    BPL     51$             ;B=NO
7337  041744  105741                    TSTB    -(R1)           ;INT LOOP MODE
7338  041746  100023                    BPL     51$             ;B=NO
7339  041750  104401  041756            TYPE    ,79$            ;;TYPE ASCIZ STRING
7340  041754  000420                    BR      ,78$            ;;GET OVER THE ASCIZ
7341                              ;;79$:  .ASCIZ  /  (WARNING)NO CRC-32 TESTING/<15><12>
7342  042016                      ,78$:
7343  042016  012737  177777  041066  51$:    MOV     #-1,CONFIG
7344  042024  000137  041134            JMP     MODTAB
7345
7346
7347  042030  104411            7$:     RDLIN
7348  042032  005726                    TST     (SP)+
7349  042034  012700  046124            MOV     #$TTYIN,R0              ;GET TTY BUFFER
7350  042040  105710                    TSTB    (R0)                    ;ANY CHANGE IN STATUS?
7351  042042  001447                    BEQ     110$                    ;B=NO
7352  042044  112711  177777            MOVB    #-1,(R1)
7353  042050  122710  000101            CMPB    #'A,(R0)        ;TEST FOR ALL LINES
7354  042054  001442                    BEQ     110$            ;B=ENABLE ALL LINES
7355  042056  105011                    CLRB    (R1)
7356  042060  122710  000116            CMPB    #'N,(R0)        ;IS IT N FOR NO LINES
7357  042064  001436                    BEQ     110$            ;B=YES NO LINES ENABLED
7358  042066  112037  001202    107$:   MOVB    (R0)+,$TMP0     ;DONE?
7359  042072  001433                    BEQ     110$            ;B=YES
7360  042074  122737  000054  0012C2    CMPB    #54,$TMP0       ;A COMMA
7361  042102  001771                    BEQ     107$            ;B=YES ONLY A SEPERATOR
7362  042104  123727  001202  000060    CMPB    $TMP0,#60       ;ONLY 0-7'S ARE VALID
7363  042112  002425                    BLT     112$            ;B=NOT VALID
7364  042114  123727  001202  000067    CMPB    $TMP0,#67       ;LESS THAN 7
7365  042122  003021                    BGT     112$            ;B=NOT VALID
7366  042124  142737  000360  001202    BICB    #360,$TMP0      ;SAVE ONLY OCTAL DIGIT
7367  042132  105237  001202            INCB    $TMP0
7368  042136  012702  000001            MOV     #1,R2           ;INIT MASKING BIT
7369  042142  105337  001202    108$:   DECB    $TMP0           ;FIND BIT POSITION OF ENABLED LINE
7370  042146  001403                    BEQ     109$            ;B=FOUND IT
7371  042150  106302                    ASLB    R2              ;NEXT LINE BIT
7372  042152  001373                    BNE     108$
```

```
7373  042154  000404                    BR      112$            ;INVALID CHR.
7374  042156  150211            109$:   BISB    R2,(R1)         ;SET BIT FOUND (THIS LINE ENABLED)
7375  042160  000742                    BR      107$            ;TRY AGAIN
7376
7377  042162  005201            110$:   INC     R1              ;MOVE TO NEXT BYTE
7378  042164  000207                    RTS     PC              ;DONE
7379  042166                    112$:
7380  042166  104401  042174            TYPE    ,81$            ;;TYPE ASCIZ STRING
7381  042172  000411                    BR      80$             ;;GET OVER THE ASCIZ
7382                            ;;81$:  .ASCIZ  <15><12>/SYNTAX ERROR/<15><12>
7383  042216                    80$:
7384  042216  000137  042030            JMP     7$
7385  042222  005726            89$:    TST     (SP)+           ;POP STACK
7386  042224  104401  001234            TYPE    ,$QUES          ;HIGH BYTE NOT =0
7387  042230  104401  001235            TYPE    ,$CRLF
7388  042234  000675                    BR      7$
7389  042236  104412            8$:     RDOCT
7390  042240  012600                    MOV     (SP)+,R0        ;CHANGE STATUS?
7391  042242  001401                    BEQ     9$              ;BR=NO
7392  042244  010011                    MOV     R0,(R1)         ;LOAD DATA
7393  042246  005721            9$:     TST     (R1)+
7394  042250  000207                    RTS     PC
7395  042252  104410            10$:    RDCHR
7396  042254  012600                    MOV     (SP)+,R0
7397  042256  122700  000015            CMPB    #15,R0          ;C.R.?
7398  042262  001425                    BEQ     114$
7399  042264  110037  041060            MOVB    R0,CHRADR
7400  042270  104401  041060            TYPE    ,CHRADR
7401  042274  122700  000060            CMPB    #60,R0          ;0?
7402  042300  001413                    BEQ     11$
7403  042302  122700  000061            CMPB    #61,R0          ;1?
7404  042306  001405                    BEQ     105$
7405  042310  104401  042316            TYPE    ,83$            ;;TYPE ASCIZ STRING
7406  042314  000401                    BR      82$             ;;GET OVER THE ASCIZ
7407                            ;;83$:  .ASCIZ  /?/
7408  042320                    82$:
7409  042320  000754                    BR      10$
7410  042322  152721  000200    105$:   BISB    #200,(R1)+
7411  042326  000404                    BR      115$
7412  042330  142721  000200    11$:    BICB    #200,(R1)+
7413  042334  000401                    BR      115$
7414  042336  005201            114$:   INC     R1
7415  042340  000207            115$:   RTS     PC
7416                                    ;AN "A" WAS DETECTED SET REMAINING BLOCKS WITH SUCCESSIVE
7417                                    ;PARAMETERS (BASED) AS LAST BLOCK OPEN "CURTAB".
7418
7419  042342  005726            12$:    TST     (SP)+
7420  042344  013700  041070            MOV     CURTAB,R0       ;GET CURRENT BLOCK
7421  042350                    125$:
7422  042350  010001                    MOV     R0,R1           ;
7423  042352  062701  000006            ADD     #6,R1
7424  042356  020127  042626    13$:    CMP     R1,#KMCTBE      ;AT END?
7425  042362  002010                    BGE     14$             ;B=Y
7426  042364  012011                    MOV     (R0)+,(R1)      ;GET NEW D.A.
7427  042366  062721  000010            ADD     #10,(R1)+
7428  042372  012021                    MOV     (R0)+,(R1)+     ;SAME LINES + UNIT ENABLE BIT
```

```
7429  042374  012021                          MOV     (R0)+,(R1)+     ;SAME E/I BIT
7430  042376  010037  041070                  MOV     R0,CURTAB
7431  042402  000765                          BR      13$
7432
7433  042404  000137  041134          14$:    JMP     MODTAB
7434  042410  000000                  15$:    0
7435  042412  044514  042516  024123  120$:   .ASCIZ  !LINES(0-7)(I.E..#,#,#) OR (A)=ALL(N)=NONE  !
7436  042420  026460  024467  044450
7437  042426  042456  027056  026043
7438  042434  026043  024443  047440
7439  042442  020122  040450  036451
7440  042450  046101  024114  024516
7441  042456  047075  047117  020105
7442  042464  000040
7443                                          .EVEN
7444
7445                                          ;KMC11 PARAMETER BLOCKS
7446
7447  042466                          KMCTAB:
7448                                          ;PARAMETER BLOCK FOR KMC#01
7449  042466  000000                          KMCR01: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #01
7450  042470  000000                          KMCP01: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7451  042472  000000                          KMCS01: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7452                                          ;PARAMETER BLOCK FOR KMC#02
7453  042474  000000                          KMCR02: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #02
7454  042476  000000                          KMCP02: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7455  042500  000000                          KMCS02: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7456                                          ;PARAMETER BLOCK FOR KMC#03
7457  042502  000000                          KMCR03: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #03
7458  042504  000000                          KMCP03: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7459  042506  000000                          KMCS03: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7460                                          ;PARAMETER BLOCK FOR KMC#04
7461  042510  000000                          KMCR04: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #04
7462  042512  000000                          KMCP04: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7463  042514  000000                          KMCS04: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7464                                          ;PARAMETER BLOCK FOR KMC#05
7465  042516  000000                          KMCR05: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #05
7466  042520  000000                          KMCP05: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7467  042522  000000                          KMCS05: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7468                                          ;PARAMETER BLOCK FOR KMC#06
7469  042524  000000                          KMCR06: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #06
7470  042526  000000                          KMCP06: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7471  042530  000000                          KMCS06: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7472                                          ;PARAMETER BLOCK FOR KMC#07
7473  042532  000000                          KMCR07: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #07
7474  042534  000000                          KMCP07: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7475  042536  000000                          KMCS07: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7476                                          ;PARAMETER BLOCK FOR KMC#08
7477  042540  000000                          KMCR08: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #08
7478  042542  000000                          KMCP08: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7479  042544  000000                          KMCS08: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7480                                          ;PARAMETER BLOCK FOR KMC#09
7481  042546  000000                          KMCR09: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #09
7482  042550  000000                          KMCP09: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7483  042552  000000                          KMCS09: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7484                                          ;PARAMETER BLOCK FOR KMC#10
```

```
7485  042554  000000                  KMCR10: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #10
7486  042556  000000                  KMCP10: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7487  042560  000000                  KMCS10: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7488                                  ;PARAMETER BLOCK FOR KMC#11
7489  042562  000000                  KMCR11: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #11
7490  042564  000000                  KMCP11: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7491  042566  000000                  KMCS11: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7492                                  ;PARAMETER BLOCK FOR KMC#12
7493  042570  000000                  KMCR12: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #12
7494  042572  000000                  KMCP12: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7495  042574  000000                  KMCS12: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7496                                  ;PARAMETER BLOCK FOR KMC#13
7497  042576  000000                  KMCR13: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #13
7498  042600  000000                  KMCP13: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7499  042602  000000                  KMCS13: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7500                                  ;PARAMETER BLOCK FOR KMC#14
7501  042604  000000                  KMCR14: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #14
7502  042606  000000                  KMCP14: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7503  042610  000000                  KMCS14: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7504                                  ;PARAMETER BLOCK FOR KMC#15
7505  042612  000000                  KMCR15: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #15
7506  042614  000000                  KMCP15: C       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7507  042616  000000                  KMCS15: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7508                                  ;PARAMETER BLOCK FOR KMC#16
7509  042620  000000                  KMCR16: 0       ;CONTROL + STATUS REGISTER FOR KMC11 #16
7510  042622  000000                  KMCP16: 0       ;LO BYTE = LINES ENABLED (BIT 0 = LINE 0 + 7 = LINE 7) - BIT 15
7511  042624  000000                  KMCS16: 0       ;LO BYTE INT.=300,EXT=100/ HIBYTE 200=CRC32ENABLED 0 = NOT
7512
7513  042626  000000          KMCTBE: 0
7514
7515                                  ;THIS ROUTINE WILL LOAD THE CSR ADDRESS INTO "R4" THE
7516                                  ;LINE ENABLE BYTE INTO "LINBYT" ,  EXT./INT. STATUS  INTO "CLKDAT"
7517                                  ;CRC32 ENABLE INTO "CRC32F"
7518  042630  013701  041070  GETKMC: MOV     CURTAB,R1       ;GET CURRENT TABLE POINTER
7519  042634  062701  000006  1$:     ADD     #6,R1
7520  042640  020127  042626          CMP     R1,#KMCS16+2    ;@ END OF TABLE?
7521  042644  001004                  BNE     2$              ;B=NO
7522  042646  012737  042460  041070  MOV     #KMCTAB-6,CURTAB
7523  042654  000207                  RTS     PC
7524  042656  005761  000002  2$:     TST     2(R1)   ;ENABLED?
7525  042662  100364                  BPL     1$              ;BR=NO
7526  042664  010137  041070          MOV     R1,CURTAB       ;NOW POINTING TO CURRENT KMC
7527  042670  012104                  MOV     (R1)+,R4        ;GETKMC CSR0
7528  042672  111137  001254          MOVB    (R1),LINBYT     ;GET LINES ENABLED
7529  042676  005721                  TST     (R1)+
7530  042700  112137  001260          MOVB    (R1)+,CLKDAT    ;LOAD CLOCK LOOP STATUS
7531  042704  005037  001256          CLR     CRC32F          ;CLEAR CRC32 FLAG
7532  042710  105711                  TSTB    (R1)            ;CRC32 ENABLED?
7533  042712  100007                  BPL     26$             ;B=NO
7534  042714  132737  000200  001260  BITB    #200,CLKDAT     ;EXT?
7535  042722  001003                  BNE     26$             ;B=NO
7536  042724  012737  000200  001256  MOV     #200,CRC32F     ;ENABLE CRC32 FOR TESTING
7537  042732                  26$:
7538  042732  012702  000001          MOV     #1,R2
7539  042736  013700  041070          MOV     CURTAB,R0       ;FIND UNIT#
7540  042742  162700  042466          SUB     #KMCTAB,R0
```

```
7541  042746  001405                        BEQ     4$
7542  042750  005202                3$:     INC     R2
7543  042752  162700  000006                SUB     #6,R0
7544  042756  001401                        BEQ     4$
7545  042760  000773                        BR      3$
7546  042762  010237  001246        4$:     MOV     R2,$DVICE       ;CURRENT UNIT #
7547  042766  012737  000001  001252        MOV     #1,NXTLIN       ;SET SWITCH FOR NEXT LINE TO BE TESTED
7548  042774  062716  000002                ADD     #2,(SP)
7549  043000  000207                        RTS     PC
7550                                        ;THIS ROUTINE WILL OBTAIN THE NEXT SEQUENTIAL LINE
7551                                        ;(IF ENABLED) FROM LOCTION 'NXTLIN:' (PRESENT BY GETKMC)
7552
7553  043002  005037  001244        GETLIN: CLR     CLINED  ;INIT. FORMATED LINE #
7554  043006  105737  001252        1$:     TSTB    NXTLIN          ;DONE THIS UNIT?
7555  043012  001437                        BEQ     3$              ;B=YES
7556  043014  033737  001252  001254        BIT     NXTLIN,LINBYT   ;IS THIS LINE ENABLED?
7557  043022  001003                        BNE     2$              ;B=YES
7558  043024  006337  001252                ASL     NXTLIN          ;TRY FOR NEXT LINE
7559  043030  000766                        BR      1$
7560  043032  012700  000001        2$:     MOV     #1,R0           ;FIND UNIT #
7561  043036  030037  001252        21$:    BIT     R0,NXTLIN       ;SCALE FOR LINE #
7562  043042  001004                        BNE     22$             ;ON THIS LINE?
7563  043044  005237  001244                INC     CLINED          ;UP BY ONE LINE
7564  043050  006300                        ASL     R0              ;ALLIGN BIT TO TRY AGAIN
7565  043052  000771                        BR      21$             ;TRY NEXT LINE
7566  043054  013737  001244  001242  22$:  MOV     CLINED,CLINE    ;OBTAIN REG. FORMAT
7567  043062  006337  001242                ASL     CLINE           ;4 LEFT
7568  043066  006337  001242                ASL     CLINE
7569  043072  006337  001242                ASL     CLINE
7570  043076  006337  001242                ASL     CLINE
7571  043102  006337  001252                ASL     NXTLIN          ;SET NXTLIN FOR NEXT TIME IN
7572  043106  062716  000002                ADD     #2,(SP)
7573  043112  000207                3$:     RTS     PC
7574
7575                                        .SBTTL  ERROR HANDLER ROUTINE
7576
7577                                        ;;**************************************************************************
7578                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
7579                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
7580                                        ;*AND GO TO $ERRTYP ON ERROR
7581                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
7582                                        ;*SW15=1         HALT ON ERROR
7583                                        ;*SW13=1         INHIBIT ERROR TYPEOUTS
7584                                        ;*SW10=1         BELL ON ERROR
7585                                        ;*SW09=1         LOOP ON ERROR
7586                                        ;*CALL
7587                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
7588
7589  043114                                $ERROR:
7590  043114  104407                        CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
7591  043116  105237  001103        7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
7592  043122  001775                        BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
7593  043124  013777  001102  136010        MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
7594  043132  032777  002000  136000        BIT     #BIT10,@SWR     ;;BELL ON ERROR?
7595  043140  001402                        BEQ     1$              ;;NO - SKIP
7596  043142  104401  001230                TYPE    ,$BELL          ;;RING BELL
```

```
7597  043146  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
7598  043152  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
7599  043156  162737  000002  001116          SUB     #2,$ERRPC
7600  043164  117737  135726  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
7601  043172  032777  020000  135740          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
7602  043200  001055                          BNE     20$             ;;SKIP TYPEOUTS
7603  043202  021627  001002                  CMP     (SP),#1002      ;;IF RETURN PC LESS THAN 1002
7604  043206  101046                          BHI     12$             ;;ERROR IS ILLEGAL TRAP
7605                                  ;;PROCESS UNEXPECTED TRAP OR INTERRUPT
7606  043210  016637  000004  001116          MOV     4(SP),$ERRPC    ;;GET PC AT TIME OF FALSE TRAP
7607  043216  162737  000002  001116          SUB     #2,$ERRPC       ;;ADJUST PC
7608  043224  104401  043270                  TYPE    ,10$            ;;TYPE HEADER
7609  043230  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
7610  043234  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7611  043236  104401  043276                  TYPE    ,11$
7612  043242  162716  000004                  SUB     #4,(SP)         ;;GET FALSE TRAP VECTOR ADDR
7613  043246  011637  001116                  MOV     (SP),$ERRPC
7614  043252  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
7615  043256  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7616  043260  104401  001235                  TYPE    ,$CRLF
7617  043264  022626                          CMP     (SP)+,(SP)+     ;;POP FALSE TRAP VECTOR PC&ADDR
7618  043266  000422                          BR      20$
7619  043270  050200  036503  000040  10$:     .ASCIZ  <200>'PC= '
7620  043276  020040  047125  054105  11$:     .ASCIZ  '  UNEXPECTED TRAP TO '
7621  043304  042520  052103  042105
7622  043312  052040  040522  020120
7623  043320  047524  000040
7624                                          .EVEN
7625  043324                          12$:
7626  043324  004737  043410                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
7627  043330  104401  001235                  TYPE    ,$CRLF
7628  043334                          20$:
7629  043334  005777  135600          2$:     TST     @SWR            ;;HALT ON ERROR
7630  043340  100002                          BPL     3$              ;;SKIP IF CONTINUE
7631  043342  000000                          HALT                    ;;HALT ON ERROR!
7632  043344  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
7633  043346  032777  001000  135564  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
7634  043354  001402                          BEQ     4$              ;;BR IF NO
7635  043356  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
7636  043362  005737  001226          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
7637  043366  001402                          BEQ     5$              ;;BR IF NONE
7638  043370  013716  001226                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
7639  043374                          5$:
7640  043374  022737  035070  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
7641  043402  001001                          BNE     6$              ;;BRANCH IF NO
7642  043404  000000                          HALT                    ;;YES
7643  043406                          6$:
7644  043406  000002                          RTI                     ;;RETURN
7645                                  .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
7646
7647                                  ;;**********************************************************************
7648                                  ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
7649                                  ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
7650                                  ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
7651
7652  043410                          $ERRTYP:
```

```
7653  043410  104401  001235          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
7654  043414  010046                  MOV     R0,-(SP)        ;;SAVE R0
7655  043416  005000                  CLR     R0              ;;PICKUP THE ITEM INDEX
7656  043420  153700  001114          BISB    @#$ITEMB,R0
7657  043424  001004                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
7658                                                          ;;TYPE THE PC OF THE ERROR
7659  043426  013746  001116          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
7660                                                          ;;ERROR ADDRESS
7661  043432  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7662  043434  000445                  BR      10$             ;;GET OUT
7663  043436  005300          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
7664  043440  006300                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
7665  043442  006300                  ASL     R0
7666  043444  006300                  ASL     R0
7667  043446  062700  001264          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
7668  043452  012037  043462          MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
7669  043456  001404                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
7670  043460  104401                  TYPE                    ;;TYPE THE "ERROR MESSAGE"
7671  043462  000000          2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
7672  043464  104401  001235          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
7673  043470  012037  043500          MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
7674  043474  001404                  BEQ     5$              ;;SKIP TYPEOUT IF 0
7675  043476  104401                  TYPE                    ;;TYPE THE "DATA HEADER"
7676  043500  000000          4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
7677  043502  104401  001235          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
7678  043506  010146          5$:     MOV     R1,-(SP)        ;;SAVE R1
7679  043510  012001                  MOV     (R0)+,R1        ;;PICKUP "DATA TABLE" POINTER
7680  043512  001415                  BEQ     9$              ;;BR IF NO DATA TO BE TYPED
7681  043514  012000                  MOV     (R0)+,R0        ;;PICKUP "DATA FORMAT" POINTER
7682  043516  105720          6$:     TSTB    (R0)+           ;;"OCTAL" OR "DECIMAL"
7683  043520  001003                  BNE     7$              ;;BR IF DECIMAL
7684  043522  013146                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
7685  043524  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7686  043526  000402                  BR      8$
7687  043530                  7$:
7688  043530  013146                  MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
7689  043532  104405                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
7690  043534  005711          8$:     TST     (R1)            ;;IS THERE ANOTHER NUMBER?
7691  043536  001403                  BEQ     9$              ;;BR IF NO
7692  043540  104401  043560          TYPE    ,11$            ;;TYPE TWO(2) SPACES
7693  043544  000764                  BR      6$              ;;LOOP
7694
7695  043546  012601          9$:     MOV     (SP)+,R1        ;;RESTORE R1
7696  043550  012600          10$:    MOV     (SP)+,R0        ;;RESTORE R0
7697  043552  104401  001235          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
7698  043556  000207                  RTS     PC              ;;RETURN
7699  043560  020040    000   11$:    .ASCIZ  /  /            ;;TWO(2) SPACES
7700          043564                  .EVEN
7701                          .SBTTL  SCOPE HANDLER ROUTINE
7702
7703                          ;;**************************************************************************
7704                          ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
7705                          ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
7706                          ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
7707                          ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
7708                          ;*SW14=1         LOOP ON TEST
```

```
                                ;*SW09=1        LOOP ON ERROR
                                ;*SW08=1        LOOP ON TEST IN SWR<7:0>
                                ;*CALL
                                ;*      SCOPE           ;;SCOPE=IOT
                                $SCOPE:
043564  104407                         CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
                                ;;GO TO ERROR ROUTINE IF RETURN PC LESS THAN 1002
                                ;;OTHERWISE CONTINUE
043566  021627  001002                 CMP     (SP),#1002      ;;UNEXPECTED TRAP OR INTERRRUPT
043572  101002                         BHI     1$              ;;ARE TRAPPED HERE VIA IOT
043574  000137  043114                 JMP     $ERROR          ;;GO PROCESS UNEXPECTED TRAP
043600  032777  040000  135332 1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
043606  001057                         BNE     $OVER           ;;YES IF SW14=1
                                ;#####START OF CODE FOR THE XOR TESTER#####
043610  000416                 $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
043612  013746  000004                 MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
043616  012737  043636  000004         MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
043624  005737  177060                 TST     @#177060        ;;TIME OUT ON XOR?
043630  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
043634  000431                         BR      $SVLAD          ;;GO TO THE NEXT TEST
043636  022626                 5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
043640  012637  000004                 MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
043644  000417                         BR      7$              ;;LOOP ON THE PRESENT TEST
043646                          6$:;#####END OF CODE FOR THE XOR TESTER#####
043646  032777  000400  135264         BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
043654  001404                         BEQ     2$              ;;BR IF NO
043656  127737  135256  001102         CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
043664  001430                         BEQ     $OVER           ;;BR IF YES
043666  105737  001103         2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
043672  001412                         BEQ     $SVLAD          ;;BR IF NO
043674  032777  001000  135236         BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
043702  001404                         BEQ     4$              ;;BR IF NO
043704  013737  001110  0011C6 7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
043712  000415                         BR      $OVER
043714  105037  001103         4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
043720  105237  001102         $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
043724  011637  001106                 MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
043730  011637  001110                 MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
043734  005037  001226                 CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
043740  112737  000001  001115         MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
043746  013777  001102  135166 $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
043754  013716  001106                 MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
043760  000002                         RTI                     ;;FIXES PS
                                .SBTTL  TYPE ROUTINE

                                ;;**************************************************************************
                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                ;*
                                ;*CALL:
                                ;*1) USING A TRAP INSTRUCTION
```

```
7765                                    ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
7766                                    ;*OR
7767                                    ;*      TYPE
7768                                    ;*      MESADR
7769                                    ;*
7770
7771  043762  105737  001157            $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
7772  043766  100002                            BPL     1$              ;;BR IF YES
7773  043770  000000                            HALT                    ;;HALT HERE IF NO TERMINAL
7774  043772  000407                            BR      3$              ;;LEAVE
7775  043774  010046                    1$:     MOV     R0,-(SP)        ;;SAVE R0
7776  043776  017600  000002                    MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
7777  044002  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
7778  044004  001005                            BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
7779  044006  005726                            TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
7780  044010  012600                    60$:    MOV     (SP)+,R0        ;;RESTORE R0
7781  044012  062716  000002            3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
7782  044016  000002                            RTI                     ;;RETURN
7783  044020  122716  000011            4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
7784  044024  001430                            BEQ     8$
7785  044026  122716  000200                    CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
7786  044032  001006                            BNE     5$
7787  044034  005726                            TST     (SP)+           ;;POP  <CR><LF> EQUIV
7788  044036  104401                            TYPE                    ;;TYPE A CR AND LF
7789  044040  001235                            $CRLF
7790  044042  105037  044250                    CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
7791  044046  000755                            BR      2$              ;;GET NEXT CHARACTER
7792  044050  004737  044132            5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
7793  044054  123726  001156            6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
7794  044060  001350                            BNE     2$              ;;IF NO GO GET NEXT CHAR.
7795  044062  013746  001154                    MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
7796                                            ;;AND THE NULL CHAR.
7797  044066  105366  000001            7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
7798  044072  002770                            BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
7799  044074  004737  044132                    JSR     PC,$TYPEC       ;;GO TYPE A NULL
7800  044100  105337  044250                    DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
7801  044104  000770                            BR      7$              ;;LOOP
7802
7803                                    ;HORIZONTAL TAB PROCESSOR
7804
7805  044106  112716  000040            8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
7806  044112  004737  044132            9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
7807  044116  132737  000007  044250            BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
7808  044124  001372                            BNE     9$              ;;TAB STOP
7809  044126  005726                            TST     (SP)+           ;;POP SPACE OFF STACK
7810  044130  000724                            BR      2$              ;;GET NEXT CHARACTER
7811  044132                            $TYPEC:
7812  044132  105777  135006                    TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
7813  044136  100022                            BPL     10$             ;;BR IF NOT                     ;MJD001
7814  044140  017746  135002                    MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
7815  044144  042716  177600                    BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
7816  044150  122716  000023                    CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
7817  044154  001012                            BNE     102$            ;;BR IF NOT                     ;MJD001
7818  044156                            101$:                           ;                               ;MJD001
7819  044156  105777  134762                    TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
7820  044162  100375                            BPL     101$            ;                               ;MJD001
```

```
7821  044164  117716  134756                  MOVB    @$TKB,(SP)      ;;GET CHAR                              ;MJD001
7822  044170  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT                              ;MJD001
7823  044174  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?                           ;MJD001
7824  044200  001366                          BNE     101$            ;;BR IF NOT                             ;MJD001
7825  044202                          102$:                                                                   ;MJD001
7826  044202  005726                          TST     (SP)+           ;;FIX STACK                             ;MJD001
7827  044204                          10$:                                                                    ;MJD001
7828  044204  105777  134740                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
7829  044210  100375                          BPL     10$                                                     ;MJD001
7830  044212  116677  000002  134732          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
7831  044220  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
7832  044226  001003                          BNE     1$              ;;BRANCH IF NO
7833  044230  105037  044250                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
7834  044234  000406                          BR      $TYPEX          ;;EXIT
7835  044236  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
7836  044244  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
7837  044246  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
7838  044250  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
7839  044252  000207                  $TYPEX: RTS     PC
7840
7841                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
7842
7843                                  ;;**************************************************************************
7844                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
7845                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
7846                                  ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
7847                                  ;*CALL:
7848                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
7849                                  ;*      TYPOS                   ;;CALL FOR TYPEOUT
7850                                  ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
7851                                  ;*      .BYTE   M               ;;M=1 OR 0
7852                                  ;*                                      ;;1=TYPE LEADING ZEROS
7853                                  ;*                                      ;;0=SUPPRESS LEADING ZEROS
7854                                  ;*
7855                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
7856                                  ;*$TYPOS OR $TYPOC
7857                                  ;*CALL:
7858                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
7859                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
7860                                  ;*
7861                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
7862                                  ;*CALL:
7863                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
7864                                  ;*      TYPOC                   ;;CALL FOR TYPEOUT
7865
7866  044254  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
7867  044260  116637  000001  044477          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
7868  044266  112637  044501                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
7869  044272  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
7870  044276  000406                          BR      $TYPON
7871  044300  112737  000001  044477  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
7872  044306  112737  000006  044501          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
7873  044314  112737  000005  044476  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
7874  044322  010346                          MOV     R3,-(SP)        ;;SAVE R3
7875  044324  010446                          MOV     R4,-(SP)        ;;SAVE R4
7876  044326  010546                          MOV     R5,-(SP)        ;;SAVE R5
```

```
7877  044330  113704  044501          MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
7878  044334  005404                  NEG     R4
7879  044336  062704  000006          ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
7880  044342  110437  044500          MOVB    R4,$OMODE       ;;SAVE IT FOR USE
7881  044346  113704  044477          MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
7882  044352  016605  000012          MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
7883  044356  005003                  CLR     R3              ;;CLEAR THE OUTPUT WORD
7884  044360  006105          1$:     ROL     R5              ;;ROTATE MSB INTO "C"
7885  044362  000404                  BR      3$              ;;GO DO MSB
7886  044364  006105          2$:     ROL     R5              ;;FORM THIS DIGIT
7887  044366  006105                  ROL     R5
7888  044370  006105                  ROL     R5
7889  044372  010503                  MOV     R5,R3
7890  044374  006103          3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
7891  044376  105337  044500          DECB    $OMODE          ;;TYPE THIS DIGIT?
7892  044402  100016                  BPL     7$              ;;BR IF NO
7893  044404  042703  177770          BIC     #177770,R3      ;;GET RID OF JUNK
7894  044410  001002                  BNE     4$              ;;TEST FOR 0
7895  044412  005704                  TST     R4              ;;SUPPRESS THIS 0?
7896  044414  001403                  BEQ     5$              ;;BR IF YES
7897  044416  005204          4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
7898  044420  052703  000060          BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
7899  044424  052703  000040          BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
7900  044430  110337  044474          MOVB    R3,8$           ;;SAVE FOR TYPING
7901  044434  104401  044474          TYPE    ,8$             ;;GO TYPE THIS DIGIT
7902  044440  105337  044476          DECB    $OCNT           ;;COUNT BY 1
7903  044444  003347                  BGT     2$              ;;BR IF MORE TO DO
7904  044446  002402                  BLT     6$              ;;BR IF DONE
7905  044450  005204                  INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
7906  044452  000744                  BR      2$              ;;GO DO THE LAST DIGIT
7907  044454  012605          6$:     MOV     (SP)+,R5        ;;RESTORE R5
7908  044456  012604                  MOV     (SP)+,R4        ;;RESTORE R4
7909  044460  012603                  MOV     (SP)+,R3        ;;RESTORE R3
7910  044462  016666  000002  000004  MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
7911  044470  012616                  MOV     (SP)+,(SP)
7912  044472  000002                  RTI                     ;;RETURN
7913  044474     000          8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
7914  044475     000                  .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
7915  044476     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
7916  044477     000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
7917  044500  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
7918                          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
7919
7920                          ;;**************************************************************************
7921                          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
7922                          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
7923                          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
7924                          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
7925                          ;*REPLACED WITH SPACES.
7926                          ;*CALL:
7927                          ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
7928                          ;*      TYPDS                   ;;GO TO THE ROUTINE
7929
7930  044502                  $TYPDS:
7931  044502  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
7932  044504  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
```

```
 7933  044506  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 7934  044510  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 7935  044512  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 7936  044514  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
 7937  044520  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
 7938  044524  100004                          BPL     1$              ;;BR IF INPUT IS POS.
 7939  044526  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
 7940  044530  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
 7941  044536  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
 7942  044540  012703  044716                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
 7943  044544  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
 7944  044550  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
 7945  044552  016001  044706                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
 7946  044556  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
 7947  044560  002402                          BLT     4$              ;;BR IF DONE
 7948  044562  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
 7949  044564  000774                          BR      3$
 7950  044566  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
 7951  044570  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
 7952  044572  001002                          BNE     5$              ;;FALL THROUGH IF 0
 7953  044574  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
 7954  044576  100407                          BMI     7$              ;;BR IF YES
 7955  044600  106316                  5$:     ASLB    (SP)            ;;MSD?
 7956  044602  103003                          BCC     6$              ;;BR IF NO
 7957  044604  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
 7958  044612  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
 7959  044616  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
 7960  044622  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
 7961  044624  005720                          TST     (R0)+           ;;JUST INCREMENTING
 7962  044626  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
 7963  044632  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
 7964  044634  003002                          BGT     8$              ;;GO TO EXIT
 7965  044636  010502                          MOV     R5,R2           ;;GET THE LSD
 7966  044640  000764                          BR      6$              ;;GO CHANGE TO ASCII
 7967  044642  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
 7968  044644  100003                          BPL     9$              ;;BR IF NO
 7969  044646  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
 7970  044654  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
 7971  044656  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
 7972  044660  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
 7973  044662  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
 7974  044664  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 7975  044666  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 7976  044670  104401  044716                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
 7977  044674  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
 7978  044702  012616                          MOV     (SP)+,(SP)
 7979  044704  000002                          RTI                     ;;RETURN TO USER
 7980  044706  023420                  $DTBL:  10000.
 7981  044710  001750                          1000.
 7982  044712  000144                          100.
 7983  044714  000012                          10.
 7984  044716  000004                  $DBLK:  .BLKW   4
 7985                                  .SBTTL  TTY INPUT ROUTINE
 7986
 7987                                  ;;**************************************************************************
 7988                                  .ENABL  LSB
```

```
 7989  044726  000000                 $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
 7990  044730  000000                 $TKQIN: .WORD   0               ;;INPUT POINTER
 7991  044732  000000                 $TKQOUT: .WORD  0               ;;OUTPUT POINTER
 7992  044734  000001                 $TKQSRT: .BLKB  X               ;;TTY KEYBOARD QUEUE
 7993          044735                 $TKQEND=.
 7994          044736                 .EVEN
 7995
 7996                                 ;*TK INITIALIZE ROUTINE
 7997                                 ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
 7998                                 ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
 7999                                 ;
 8000                                 ;*CALL:
 8001                                 ;*      JSR     PC,$TKINT
 8002                                 ;*      RETURN
 8003                                 ;
 8004  044736  005037  044726         $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
 8005  044742  012737  044734  044730         MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
 8006  044750  013737  044730  044732         MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
 8007  044756  012737  045006  000060         MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
 8008  044764  012737  000200  000062         MOV     #200,@#TKVEC+2  ;;'BR" LEVEL 4
 8009  044772  005777  134150                 TST     @$TKB           ;;CLEAR DONE FLAG
 8010  044776  012777  000100  134140         MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
 8011  045004  000207                         RTS     PC              ;;RETURN TO CALLER
 8012
 8013                                 ;*TK SERVICE ROUTINE
 8014                                 ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
 8015                                 ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
 8016                                 ;*IT IN THE QUEUE.
 8017                                 ;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
 8018                                 ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (.MONIT)
 8019                                 ;
 8020  045006  117746  134134         $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
 8021  045012  042716  177600                 BIC     #^C177,(SP)     ;;STRIP THE JUNK
 8022  045016  021627  000021                 CMP     (SP),#$XON      ;;IS IT A RANDOM XON?                   ;RAN001
 8023  045022  001002                         BNE     30$             ;;BRANCH IF NO                          ;RAN001
 8024  045024  005726                         TST     (SP)+           ;;CLEAN RANDOM XON OFF STACK            ;RAN001
 8025  045026  000002                         RTI                     ;;RETURN                                ;RAN001
 8026  045030                         30$:                                                                    ;RAN001
 8027  045030  021627  000003                 CMP     (SP),#3         ;;IS IT A CONTROL C?
 8028  045034  001007                         BNE     1$              ;;BRANCH IF NO
 8029  045036  104401  046162                 TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
 8030  045042  004737  044736                 JSR     PC,$TKINT       ;;INIT THE KEYBOARD
 8031  045046  005726                         TST     (SP)+           ;;CLEAN UP STACK
 8032  045050  000137  002332                 JMP     .MONIT          ;;CONTROL C RESTART
 8033  045054  021627  000007         1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
 8034  045060  001004                         BNE     2$              ;;BRANCH IF NO
 8035  045062  022737  000176  001140         CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
 8036  045070  001500                         BEQ     6$              ;;GO TO SWR CHANGE
 8037
 8038  045072                         2$:
 8039  045072  022737  000001  044726         CMP     #X,$TKCNT       ;;IS THE QUEUE FULL?
 8040  045100  001004                         BNE     3$              ;;BRANCH IF NO
 8041  045102  104401  001230                 TYPE    ,$BELL          ;;RING THE TTY BELL
 8042  045106  005726                         TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
 8043  045110  000451                         BR      5$              ;;EXIT
 8044  045112  021627  000023         3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
```

```
8045  045116  001021                          BNE     32$             ;;BRANCH IF NO
8046  045120  005077  134020                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
8047  045124  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
8048  045126  105777  134012          31$:    TSTB    @$TKS           ;;WAIT FOR A CHAR
8049  045132  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
8050  045134  117746  134006                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
8051  045140  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
8052  045144  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
8053  045150  001366                          BNE     31$             ;;BRANCH IF NO
8054  045152  012777  000100  133764          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
8055  045160  000002                          RTI                     ;;RETURN
8056  045162  005237  044726          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
8057  045166  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
8058  045172  002405                          BLT     4$              ;;BRANCH IF YES
8059  045174  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
8060  045200  003002                          BGT     4$              ;;BRANCH IF YES
8061  045202  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
8062  045206  112677  177516          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
8063  045212  005237  044730                  INC     $TKQIN          ;;UPDATE THE POINTER
8064  045216  023727  044730  044735          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
8065  045224  001003                          BNE     5$              ;;BRANCH IF NO
8066  045226  012737  044734  044730          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
8067  045234  000002                  5$:     RTI                     ;;RETURN
8068
8069                                  ;;**********************************************************************
8070                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
8071                                  ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
8072                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
8073                                  ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
8074  045236  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
8075  045244  001124                          BNE     15$             ;;EXIT IF NOT
8076  045246  105777  133672                  TSTB    @$TKS           ;;IS A CHAR WAITING?
8077  045252  100121                          BPL     15$             ;;IF NOT, EXIT
8078  045254  117746  133666                  MOVB    @$TKB,-(SP)     ;;YES
8079  045260  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
8080  045264  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
8081  045270  001300                          BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
8082                                                                  ;;AND EXIT
8083
8084                                  ;;**********************************************************************
8085                                  ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
8086                                  ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
8087                                  ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
8088  045272  123727  001134  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
8089  045300  001674                          BEQ     2$              ;;BRANCH IF YES
8090  045302  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
8091  045304  004737  044736                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
8092  045310  005077  133630                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
8093  045314  112737  000001  001135          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR
8094
8095  045322  104401  046174                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
8096  045326  104401  046201          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
8097  045332  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
8098  045336  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
8099  045340  104401  046212                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
8100  045344  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
```

```
8101  045346  005046                        CLR     -(SP)           ;;THE NEW SWR
8102  045350  105777  133570          7$:    TSTB    @$TKS           ;;CHAR THERE?
8103  045354  100375                        BPL     7$              ;;IF NOT TRY AGAIN
8104
8105  045356  117746  133564                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
8106  045362  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
8107
8108  045366  021627  000003                CMP     (SP),#3         ;;IS IT A CONTROL-C?
8109  045372  001015                        BNE     9$              ;;BRANCH IF NOT
8110  045374  104401  046162                TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
8111  045400  062706  000006                ADD     #6,SP           ;;CLEAN UP STACK
8112  045404  123727  001135  000001        CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
8113  045412  001003                        BNE     8$              ;;BRANCH IF NO
8114  045414  012777  000100  133522        MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
8115  045422  000137  002332          8$:    JMP     .MONIT          ;;CONTROL-C RESTART
8116
8117
8118  045426  021627  000025          9$:    CMP     (SP),#25        ;;IS IT A CONTROL-U?
8119  045432  001005                        BNE     10$             ;;BRANCH IF NOT
8120  045434  104401  046167                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
8121  045440  062706  000006          20$:   ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
8122  045444  000737                        BR      19$             ;;LET'S TRY IT AGAIN
8123
8124
8125  045446  021627  000015          10$:   CMP     (SP),#15        ;;IS IT A <CR>?
8126  045452  001022                        BNE     16$             ;;BRANCH IF NO
8127  045454  005766  000004                TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
8128  045460  001403                        BEQ     11$             ;;BRANCH IF YES
8129  045462  016677  000002  133450        MOV     2(SP),@SWR      ;;SAVE NEW SWR
8130  045470  062706  000006          11$:   ADD     #6,SP           ;;CLEAR UP STACK
8131  045474  104401  001235          14$:   TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
8132  045500  123727  001135  000001        CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
8133  045506  001003                        BNE     15$             ;;BRANCH IF NOT
8134  045510  012777  000100  133426        MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
8135  045516  000002                15$:    RTI                     ;;RETURN
8136  045520  004737  044132          16$:   JSR     PC,$TYPEC       ;;ECHO CHAR
8137  045524  021627  000060                CMP     (SP),#60        ;;CHAR < 0?
8138  045530  002420                        BLT     18$             ;;BRANCH IF YES
8139  045532  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
8140  045536  003015                        BGT     18$             ;;BRANCH IF YES
8141  045540  042726  000060                BIC     #60,(SP)+       ;;STRIP-OFF ASCII
8142  045544  005766  000002                TST     2(SP)           ;;IS THIS THE FIRST CHAR
8143  045550  001403                        BEQ     17$             ;;BRANCH IF YES
8144  045552  006316                        ASL     (SP)            ;;NO, SHIFT PRESENT
8145  045554  006316                        ASL     (SP)            ;;    CHAR OVER TO MAKE
8146  045556  006316                        ASL     (SP)            ;;    ROOM FOR NEW ONE.
8147  045560  005266  000002          17$:   INC     2(SP)           ;;KEEP COUNT OF CHAR
8148  045564  056616  177776                BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
8149  045570  000667                        BR      7$              ;;GET THE NEXT ONE
8150  045572  104401  001234          18$:   TYPE    ,$QUES          ;;TYPE ?<CR><LF>
8151  045576  000720                        BR      20$             ;;SIMULATE CONTROL-U
8152                                .DSABL  LSB
8153
8154
8155                                ;;**************************************************************************
8156                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
```

```
                                ;*CALL:
                                ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
045600  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
045602  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE PS
045610  005066  000004                  CLR     4(SP)           ;;GET READY FOR A CHARACTER
045614  005046                          CLR     -(SP)           ;;PUT NEW PS ON STACK
045616  012746  045624                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
045622  000002                          RTI                     ;;POP NEW PC AND PS
045624                          64$:
045624  005737  044726          1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
045630  001775                          BEQ     1$
045632  005337  044726                  DEC     $TKCNT          ;;DECREMENT THE COUNTER
045636  117766  177070  000004          MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
045644  005237  044732                  INC     $TKQOUT         ;;UPDATE THE POINTER
045650  023727  044732  044735          CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
045656  001003                          BNE     2$              ;;BRANCH IF NO
045660  012737  044734  044732          MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
045666  000002                  2$:     RTI                     ;;RETURN
                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
045670  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
045672  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
045674  012703  046124          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
045700  022703  046162          2$:     CMP     #$TTYIN+30.,R3  ;;BUFFER FULL?
045704  101456                          BLOS    4$              ;;BR IF YES
045706  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
045710  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
045712  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
045716  001022                          BNE     5$              ;;BR IF NO
045720  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
045722  001007                          BNE     6$              ;;BR IF NO
045724  112737  000134  046122          MOVB    #'\,9$          ;;TYPE A BACK SLASH
045732  104401  046122                  TYPE    ,9$
045736  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
045742  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
045744  020327  046124                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
045750  103434                          BLO     4$              ;;BR IF YES
045752  111337  046122                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
045756  104401  046122                  TYPE    ,9$             ;;GO TYPE
045762  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
045764  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
045766  001406                          BEQ     7$              ;;BR IF NO
045770  112737  000134  046122          MOVB    #'\,9$          ;;TYPE A BACK SLASH
045776  104401  046122                  TYPE    ,9$
046002  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
046004  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
046010  001003                          BNE     8$              ;;BR IF NO
```

```
8213  046012  104401  046167          TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
8214  046016  000726                  BR      1$              ;;GO START OVER
8215  046020  122713  000022  8$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
8216  046024  001011                  BNE     3$              ;;BRANCH IF NO
8217  046026  105013                  CLRB    (R3)            ;;CLEAR THE CHARACTER
8218  046030  104401  001235          TYPE    ,$CRLF          ;;TYPE A 'CR' & 'LF'
8219  046034  104401  046124          TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
8220  046040  000717                  BR      2$              ;;GO PICKUP ANOTHER CHACTER
8221  046042  104401  001234  4$:     TYPE    ,$QUES          ;;TYPE A '?'
8222  046046  000712                  BR      1$              ;;CLEAR THE BUFFER AND LOOP
8223  046050  111337  046122  3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
8224  046054  104401  046122          TYPE    ,9$
8225  046060  122723  000015          CMPB    #15,(R3)+       ;;CHECK FOR RETURN
8226  046064  001305                  BNE     2$              ;;LOOP IF NOT RETURN
8227  046066  105063  177777          CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
8228  046072  104401  001236          TYPE    ,$LF            ;;TYPE A LINE FEED
8229  046076  005726                  TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
8230  046100  012603                  MOV     (SP)+,R3        ;;RESTORE R3
8231  046102  011646                  MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
8232  046104  016666  000004  000002  MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
8233  046112  012766  046124  000004  MOV     #$TTYIN,4(SP)
8234  046120  000002                  RTI                     ;;RETURN
8235  046122     000  9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
8236  046123     000          .BYTE   0                       ;;TERMINATOR
8237  046124  000036  $TTYIN: .BLKB   30.                     ;;RESERVE 30. BYTES FOR TTY INPUT
8238  046162  041536  005015     000  $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL "C"
8239  046167     136  006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
8240  046174  043536  005015     000  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
8241  046201     015  051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
8242  046206  036440  000040
8243  046212  020040  042516  020127  $MNEW:  .ASCIZ  /  NEW = /
8244  046220  020075     000
8245          046224          .EVEN
8246                          .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
8247
8248                          ;;*****************************************************************
8249                          ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
8250                          ;*CHANGE IT TO BINARY.
8251                          ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
8252                          ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED
8253                          ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
8254                          ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
8255                          ;*CALL:
8256                          ;*      RDOCT                   ;;READ AN OCTAL NUMBER
8257                          ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
8258                          ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
8259
8260  046224  011646  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
8261  046226  016666  000004  000002  MOV     4(SP),2(SP)     ;;INPUT NUMBER
8262  046234  010046          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
8263  046236  010146          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
8264  046240  010246          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
8265  046242  104411  1$:     RDLIN                   ;;READ AN ASCIZ LINE
8266  046244  012600          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
8267  046246  010037  046352  MOV     R0,5$           ;;AND SAVE IT
8268  046252  005001          CLR     R1              ;;CLEAR DATA WORD
```

READ AN OCTAL NUMBER FROM THE TTY SEQ 0170

```
8269  046254  005002                  CLR     R2
8270  046256  112046                  MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
8271  046260  001420                  BEQ     3$              ;;IF ZERO GET OUT
8272  046262  122716  000060          CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
8273  046266  003026                  BGT     4$              ;;IS AN OCTAL DIGIT
8274  046270  122716  000067          CMPB    #'7,(SP)
8275  046274  002423                  BLT     4$
8276  046276  006301                  ASL     R1              ;;*2
8277  046300  006102                  ROL     R2
8278  046302  006301                  ASL     R1              ;;*4
8279  046304  006102                  ROL     R2
8280  046306  006301                  ASL     R1              ;;*8
8281  046310  006102                  ROL     R2
8282  046312  042716  177770          BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
8283  046316  062601                  ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
8284  046320  000756                  BR      2$              ;;LOOP
8285  046322  005726          3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
8286  046324  010166  000012          MOV     R1,12(SP)       ;;SAVE THE RESULT
8287  046330  010237  046362          MOV     R2,$HIOCT
8288  046334  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
8289  046336  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
8290  046340  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
8291  046342  000002                  RTI                     ;;RETURN
8292  046344  005726          4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
8293  046346  105010                  CLRB    (R0)            ;;SET A TERMINATOR
8294  046350  104401                  TYPE                    ;;TYPE UP THRU THE BAD CHAR.
8295  046352  000000          5$:     .WORD   0
8296  046354  104401  001234          TYPE    ,$QUES          ;;"?" "CR" & "LF"
8297  046360  000730                  BR      1$              ;;TRY AGAIN
8298  046362  000000          $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
8299                          .SBTTL  READ A DECIMAL NUMBER FROM THE TTY
8300
8301                          ;;**********************************************************************
8302                          ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
8303                          ;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
8304                          ;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
8305                          ;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
8306                          ;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
8307                          ;*POSITIVE 32767 TO NEGATIVE 32768.
8308                          ;*CALL:
8309                          ;*      RDDEC                   ;;READ A DECIMAL NUMBER
8310                          ;*      RETURN HERE             ;;NUMBER IS ON TOP OF THE STACK
8311                          ;
8312
8313  046364  011646          $RDDEC: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR
8314  046366  016666  000004  000002  MOV     4(SP),2(SP)     ;;THE INPUT NUMBER
8315  046374  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
8316  046376  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
8317  046400  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
8318  046402  104411          1$:     RDLIN                   ;;READ AN ASCIZ LINE
8319  046404  012600                  MOV     (SP)+,R0        ;;ADDRESS OF 1ST CHAR.
8320  046406  010037  046532          MOV     R0,6$           ;;SAVE INCASE OF BAD INPUT
8321  046412  005046                  CLR     -(SP)           ;;CLEAR DATA WORD
8322  046414  005002                  CLR     R2              ;;SIGN SET POSITIVE
8323  046416  122710  000055          CMPB    #'-,(R0)        ;;SEE IF A MINUS SIGN WAS TYPED
8324  046422  001001                  BNE     2$              ;;BR IF NO MINUS SIGN
```

```
 8325  046424  112002                  MOVB    (R0)+,R2        ;;SAVE FOR LATER USE
 8326  046426  112001            2$:   MOVB    (R0)+,R1        ;;PICKUP THIS CHARACTER
 8327  046430  001424                  BEQ     3$              ;;GET OUT IF ZERO
 8328  046432  122701  000060          CMPB    #'0,R1          ;;MAKE SURE THIS CHARACTER
 8329  046436  003032                  BGT     5$              ;;IS A DIGIT BETWEEN 0 & 9
 8330  046440  122701  000071          CMPB    #'9,R1
 8331  046444  002427                  BLT     5$
 8332  046446  032716  170000          BIT     #^C7777,(SP)    ;;DON'T LET NUMBER GET TO BIG
 8333  046452  001024                  BNE     5$              ;;BR IF NUMBER WOULD OVERFLOW
 8334  046454  006316                  ASL     (SP)            ;;*2
 8335  046456  011646                  MOV     (SP),-(SP)      ;;SAVE FOR LATER
 8336  046460  006316                  ASL     (SP)            ;;*4
 8337  046462  006316                  ASL     (SP)            ;;*8.
 8338  046464  062616                  ADD     (SP)+,(SP)      ;;*10.
 8339  046466  102416                  BVS     5$              ;;OVERFLOW ISN'T ALLOWED
 8340  046470  162701  000060          SUB     #'0,R1          ;;STRIP AWAY THE ASCII JUNK
 8341  046474  060116                  ADD     R1,(SP)         ;;ADD IN THIS DIGIT
 8342  046476  102412                  BVS     5$              ;;OVERFLOW ISN'T ALLOWED
 8343  046500  000752                  BR      2$              ;;LOOP
 8344  046502  005702            3$:   TST     R2              ;;CHECK IF NUMBER IS NEG
 8345  046504  001401                  BEQ     4$              ;;BR IF NO
 8346  046506  005416                  NEG     (SP)            ;;YES--NEGATE THE NUMBER
 8347  046510  012666  000012    4$:   MOV     (SP)+,12(SP)    ;;SAVE THE RESULT
 8348  046514  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
 8349  046516  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
 8350  046520  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
 8351  046522  000002                  RTI                     ;;RETURN
 8352
 8353  046524  005726            5$:   TST     (SP)+           ;;CLEAN PARTIAL NUMBER FROM STACK
 8354  046526  105010                  CLRB    (R0)            ;;SET A TERMINATOR
 8355  046530  104401                  TYPE                    ;;TYPE THE INPUT UP TO BAD CHAR.
 8356  046532  000000            6$:   .WORD   0               ;;POINTER GOES HERE
 8357  046534  104401  001234          TYPE    ,$QUES          ;;"?" "CR" &"LF"
 8358  046540  000720                  BR      1$              ;;TRY AGAIN
 8359                          .SBTTL  ROUTINE TO SIZE MEMORY
 8360
 8361                          ;;**************************************************************************
 8362                          ;*CALL:
 8363                          ;*      JSR     PC,$SIZE
 8364                          ;*      RETURN
 8365                          ;*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION
 8366
 8367  046542  010046          $SIZE:  MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
 8368  046544  010146                  MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
 8369  046546  013746  000114          MOV     @#114,-(SP)     ;;SAVE MEMORY ERROR VECTOR PS & PC
 8370  046552  013746  000116          MOV     @#116,-(SP)
 8371  046556  012737  000116  000114  MOV     #116,@#114      ;;IGNORE PARITY ERRORS WHILE SIZING
 8372  046564  012737  000002  000116  MOV     #RTI,@#116
 8373  046572  013746  000004          MOV     @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
 8374  046576  013746  000006          MOV     @#ERRVEC+2,-(SP)
 8375  046602  010600                  MOV     SP,R0           ;;SAVE THE STACK POINTER
 8376                          ;;SET THE ERRVEC PS TO THE PRESENT PS
 8377  046604  104400                  TRAP                    ;;PUSH OLD PSW AND PC ON STACK
 8378  046606  012637  000006          MOV     (SP)+,@#ERRVEC+2        ;;SAVE THE PSW IN @#ERRVEC+2
 8379  046612  012737  046632  000004  MOV     #2$,@#ERRVEC    ;;SET FOR TIMEOUT
 8380  046620  012701  020000          MOV     #20000,R1       ;;FIRST ADDRESS
```

```
 8381  046624  005711                  1$:     TST     (R1)            ;;TEST THIS ADDRESS
 8382  046626  005721                          TST     (R1)+           ;;STEP TO NEXT ADDRESS
 8383  046630  000775                          BR      1$              ;;TRY ANOTHER
 8384  046632  162701  000002          2$:     SUB     #2,R1           ;;DROP BACK
 8385  046636  010006                          MOV     R0,SP           ;;RESTORE THE STACK
 8386  046640  012637  000006                  MOV     (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
 8387  046644  012637  000004                  MOV     (SP)+,@#ERRVEC
 8388  046650  012637  000116                  MOV     (SP)+,@#116     ;;RESTORE MEMORY ERROR VECTOR
 8389  046654  012637  000114                  MOV     (SP)+,@#114
 8390  046660  010137  046672                  MOV     R1,$LSTAD       ;;LAST ADDRESS
 8391  046664  012601                          MOV     (SP)+,R1        ;;RESTORE R1
 8392  046666  012600                          MOV     (SP)+,R0        ;;RESTORE R0
 8393  046670  000207                          RTS     PC
 8394  046672  000000          $LSTAD: .WORD   0               ;;CONTAINS THE LAST ADDRESS
 8395                          ;WRITE KMC11 ROUTINE
 8396
 8397  046674  017600  000000  .WRITE: MOV     @(SP),R0        ;GET REGISTERS
 8398  046700  062716  000002          ADD     #2,(SP)         ;BUMP THE STACK
 8399  046704  017601  000000          MOV     @(SP),R1        ;GET THE DATA
 8400  046710  062716  000002          ADD     #2,(SP)         ;BUMP THE STACK
 8401  046714  020027  000010          CMP     R0,#10          ;TEST FOR SECONDARY REGISTER WRITE
 8402  046720  002414                  BLT     WSEC            ;YES GO TO WRITE SECONDARY
 8403  046722  042737  000017  046746 WPRI:   BIC     #17,WPINS       ;CLEAR OLD PRIMARY ADDRESS
 8404  046730  042700  177760          BIC     #^C17,R0
 8405  046734  050037  046746          BIS     R0,WPINS        ;LOAD NEW PRIMARY ADDRESS
 8406  046740  010164  000004          MOV     R1,4(R4)        ;LOAD DATA INTO SEL4
 8407  046744  104415                  ROMCLK
 8408  046746  122100          WPINS:  122100
 8409  046750  000423                  BR      WREX            ;GO TO EXIT
 8410  046752                  WSEC:
 8411  046752  012737  000020  047010          MOV     #BIT4,WINST1    ;SET THE WRITE BIT
 8412  046760  052700  004000          BIS     #BIT11,R0       ;SET A MARKER INTO R0
 8413  046764  006100                  1$:     ROL     R0              ;ROTATE TO ALIGNE ADDRESS
 8414  046766  103376                  BCC     1$              ;CONTINUE UNTIL MARK FOUND
 8415  046770  042700  177437          BIC     #^C340,R0       ;MASK EVERYTHING ELSE
 8416  046774  050037  047010          BIS     R0,WINST1       ;LOAD THE SECONDARY ADDRESS
 8417  047000  010137  047016          MOV     R1,WINST2       ;LOAD DATA
 8418  047004  104416  000012          WRITE   ,PRI12
 8419  047010  000000          WINST1: 0
 8420  047012  104416  000010          WRITE   ,PRI10
 8421  047016  000000          WINST2: 0
 8422  047020  000002          WREX:   EXITT
 8423                          ;READ THE KMC11 PRIMARY OR SECONDARY KMC11 REGISTER ROUTINE
 8424
 8425  047022  017600  000000  .READ:  MOV     @(SP),R0        ;GET THE REGISTER
 8426  047026  062716  000002          ADD     #2,(SP)         ;BUMP THE STACK
 8427  047032  020027  000010          CMP     R0,#10          ;TEST IF A SECONDARY
 8428  047036  002427                  BLT     RSEC            ;YES GO TO READ SECONDARY
 8429  047040  042737  000360  047072 RPRI:   BIC     #360,RINST      ;CLEAR THE OLD ADDRESS
 8430  047046  012701  000004          MOV     #4,R1           ;LOAD A SHIFT COUNT
 8431  047052  006300                  1$:     ASL     R0              ;SHIFT SECONDARY ADDRESS
 8432  047054  005301                  DEC     R1
 8433  047056  001375                  BNE     1$
 8434  047060  042700  177417          BIC     #^C360,R0
 8435  047064  050037  047072          BIS     R0,RINST        ;LOAD THE REGISTER ADDRESS
 8436  047070  104415                  ROMCLK
```

```
 8437  047072  021004                  RINST:  21004
 8438  047074  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 8439  047076  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 8440  047100  016446  000004                  MOV     4(R4),-(SP)     ;;PUSH 4(R4) ON STACK
 8441  047104  042716  177400                  BIC     #^C<377>,(SP)   ;CLEAR UPPER BYTE
 8442  047110  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 8443  047112  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 8444  047114  000425                          BR      REXIT           ;GO TO EXIT
 8445  047116  005037  047146          RSEC:   CLR     RINST1          ;CLEAR THE OLD INSTRUCTION
 8446  047122  052700  004000                  BIS     #BIT11,R0       ;SET ROTATION STOP MARK
 8447  047126  006100                  1$:     ROL     R0              ;ROTATE LEFT TO ALIGN ADDRESS
 8448  047130  103376                          BCC     1$              ;CONTINUE UNTIL MARK FOUND
 8449  047132  042700  177437                  BIC     #^C340,R0       ;MASK EVERYTHING ELSE
 8450  047136  050037  047146                  BIS     R0,RINST1       ;LOAD THE REGISTER ADDRESS
 8451  047142  104416  000012                  WRITE   ,PRI12
 8452  047146  000000                  RINST1: 0
 8453  047150  104420  000010                  READ    ,PRI10          ;READ PRIMARY 10
 8454  047154  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 8455  047156  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 8456  047160  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
 8457  047162  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 8458  047164  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 8459  047166  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 8460  047170  000002                  REXIT:  EXITT
 8461  047172                          .MSTCLR:
 8462  047172  005014                          CLR     (R4)             ;*;CLEAR RUN, IF UP
 8463  047174  152764  000100  000001          BISB    #BIT6,1(R4)      ;SET MASTER CLEAR
 8464  047202  005014                          CLR     (R4)            ;CLR THE CSR REGISTERS
 8465  047204  005064  000002                  CLR     2(R4)
 8466  047210  005064  000004                  CLR     4(R4)
 8467  047214  005064  000006                  CLR     6(R4)
 8468  047220  013737  001242  047232          MOV     CLINE,10$       ;RELOAD THE LINE NUMBER
 8469  047226  104416  000011                  WRITE   ,PRI11
 8470  047232  000000                  10$:    .WORD   0
 8471  047234  000002                          EXITT                   ;RETURN
 8472  047236                          .ROMCLK:
 8473  047236  152764  000002  000001          BISB    #BIT1,1(R4)     ;SET ROMI
 8474  047244  017664  000000  000006          MOV     @(SP),6(R4)     ;LOAD THE INSTRUCTION
 8475  047252  062716  000002                  ADD     #2,(SP)         ;BUMP THE STACK
 8476  047256  152764  000003  000001  1$:     BISB    #BIT1!BIT0,1(R4) ;CLOCK INSTRUCTION
 8477  047264  142764  000007  000001          BICB    #BIT2!BIT1!BIT0,1(R4)  ;CLEAR ROMO, ROMI, STEP
 8478  047272  000002                          EXITT
 8479
 8480                                          ;*THIS ROUTINE IS USED TO MASK DATA RETURNED
 8481                                          ;*FOR TESTS THAT READ PRIMARY REGISTER'S
 8482                                          ;*16,17. THIS ROUTINE WILL DETERMINE THE LINE
 8483                                          ;*NUMBER AND CALCULATE THE MASK CHARACTER
 8484  047274  012701  000010          .MASK:  MOV     #8.,R1          ;COUNT SHIFTS
 8485  047300  013737  001254  047350          MOV     LINBYT,9$
 8486  047306  012702  047350                  MOV     #9$,R2                      ;.@LINBYT
 8487  047312  000241                  1$:     CLC
 8488  047314  106012                          RORB    (R2)                        ;BIT 0 > C
 8489  047316  106162  000001                  ROLB    1(R2)                       ;C > BIT8
 8490  047322  005301                          DEC     R1                          ;COUNT SHIFTS
 8491  047324  001372                          BNE     1$                          ;NOT DONE=B
 8492  047326  000337  047350                  SWAB    9$
```

```
 8493  047332  012737  177777  047352          MOV     #-1,MASK
 8494  047340  143737  047350  047352          BICB    9$,MASK                 ;CALCULATE MASK CHARACTER
 8495
 8496  047346  000207                          EXITS                           ;EXIT
 8497  047350  000000                  9$:     0
 8498  047352  000000                  MASK:   0
 8499
 8500                                  ;
 8501                                  ;CRC-32 MAINTENACE CHECK
 8502                                  ;
 8503
 8504  047354                          .CRC:
 8505  047354  000137  047364                  JMP     CRCENT
 8506
 8507                                          ;THIS ROUTINE WILL CALCULATE THE 32 BIT CRC CODE GENERATED
 8508                                          ;AS A RESULT OF THE 32 BIT DATA STREAM.  THE CRC POLYMOMIAL
 8509                                          ;IS X32+X26+X23+X22+X16+X12+X11+X10+X8+X7+X5+X4+X2+X+1
 8510                                          ;OR EXPRESSED OCTALLY AS
 8511                                          ;THIS ROUTINE WILL READ UP TO A 32 BIT (BINARY) (10DIGIT OCTAL)
 8512                                          ;NUMBER FROM THE TELETYPE CONSOLE IF ENTERED FROM LOCTION
 8513                                          ;"CRCENT:" AND CALCULATE AND PRINT THE RESULTANT CRC REMAINDER.
 8514                                          ;THE RESULTANT CRC CODE IS STORED IN LOCATIONS "CRCHI & CRCLOW"
 8515  047360  000000                  CRCNT:  0
 8516  047362  000000                  CALFLG: 0       ;FLAG TO INDICATEENTRY POINT
 8517  047364  005037  047362          CRCENT: CLR     CALFLG  ;FLAG TO INDICATE ENTRY POINT
 8518  047370  104401  047376                  TYPE    ,65$            ;;TYPE ASCIZ STRING
 8519  047374  000425                          BR      64$             ;;GET OVER THE ASCIZ
 8520                                  ;;65$:  .ASCIZ  <15><12>/ENTER UP TO 10 DIGIT OCTAL MSG. CODE/<15><12>
 8521  047450                          64$:
 8522  047450  104412                          RDOCT
 8523  047452  012602                          MOV     (SP)+,R2        ;GET LOW ORDER DIGITS
 8524  047454  013703  046362                  MOV     $HIOCT,R3       ;HI ORDER DIGITS
 8525  047460  000403                          BR      .+10    ;
 8526                                          ;ENTER HERE WITH HIGH ORDDER DIGITS IN R3 AND
 8527                                          ;LOW ORDER IN R2
 8528  047462  012737  177777  047362  CRCCAL: MOV     #-1,CALFLG              ;INDICATE ENTRY POINT
 8529  047470  010237  001202                  MOV     R2,$TMP0                ;USE $TMP TO CALCULATE CRC
 8530  047474  010337  001204                  MOV     R3,$TMP1
 8531  047500  012737  000040  047360          MOV     #32.,CRCNT
 8532  047506  012702  016667                  MOV     #016667,R2              ;LOW ORDER OF POLY
 8533  047512  012703  002301                  MOV     #002301,R3              ;HIGH ORDER OF POLY
 8534  047516  000261                  1$:     SEC
 8535  047520  006137  001202                  ROL     $TMP0           ;LEFT AND ENTER A ONE IN BIT 0 (CARRY)
 8536  047524  006137  001204                  ROL     $TMP1
 8537  047530  103404                          BCS     2$                      ;B=CARRY SET (NO XOR)
 8538  047532  074237  001202                  XOR     R2,$TMP0                ;EXCLUSIVE OR LOW
 8539  047536  074337  001204                  XOR     R3,$TMP1                ;EXCLUSIVE OR HI
 8540  047542  005337  047360          2$:     DEC     CRCNT
 8541  047546  001363                          BNE     1$
 8542                                          ;$TMP0 + TMP1 CONTAIN THE CALCULATED CRC CODE
 8543                                          ;IF ENTRY WAS "CRCENT" TYPE 10 DIGIT CALCULATED CRC CODE-RET-CRCENT
 8544                                          ;IF ENTRY WAS "CRCCAL" RETURN TO CALLING ROUTINE
 8545  047550  005737  047362                  TST     CALFLG
 8546  047554  001425                          BEQ     5$                      ;B=TYPE RESULTS
 8547                                          ;MESSAGE CRC SO BIT31 IS BIT 0 OF $TMP2 AND
 8548                                          ;BIT 0 IS BIT 15 OF $TMP3, THE CRC IS ARRANGED TO
```

```
 8549                                           ;REFLECT THE SAME ORDER IN WHICH IT WOULD BE TRANSMITTED
 8550                                           ;IF IN GENERATE MODE
 8551
 8552  047556  013737  001202  001206           MOV     $TMP0,$TMP2             ;SAVE FORWARD ORDER CRC
 8553  047564  013737  001204  001210           MOV     $TMP1,$TMP3
 8554
 8555  047572  012737  000040  047360           MOV     #32.,CRCNT              ;COUNT
 8556  047600  006337  001206           3$:     ASL     $TMP2                   ;DOUBLE PRECISION LEFT
 8557  047604  006137  001210                   ROL     $TMP3
 8558  047610  006037  001212                   ROR     $TMP4                   ;SHIFT THE CARRY IN
 8559  047614  006037  001214                   ROR     $TMP5                   ;INTO LOWER WORD
 8560  047620  005337  047360                   DEC     CRCNT                   ;-1 TO TOTAL SHIFT COUNT
 8561  047624  001365                           BNE     3$
 8562  047626  000207                           RTS     PC              ;RETURN TO CALLER
 8563  047630                         5$:     ;TYPE THE 2 OCTAL WORDS
 8564  047630  104401  047636                   TYPE    ,65$                    ;;TYPE ASCIZ STRING
 8565  047634  000412                           BR      64$                     ;;GET OVER THE ASCIZ
 8566                                 ;;65$:    .ASCIZ  <15><12>/CALCULATED CRC = /
 8567  047662                         64$:
 8568  047662  013746  001204                   MOV     $TMP1,-(SP)             ;;SAVE $TMP1 FOR TYPEOUT
 8569                                                                          ;;TYPE HIGH ORDER
 8570  047666  104402                           TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 8571  047670  104401  047676                   TYPE    ,67$                    ;;TYPE ASCIZ STRING
 8572  047674  000401                           BR      66$                     ;;GET OVER THE ASCIZ
 8573                                 ;;67$:    .ASCIZ  / /
 8574  047700                         66$:
 8575  047700  013746  001202                   MOV     $TMP0,-(SP)             ;;SAVE $TMP0 FOR TYPEOUT
 8576                                                                          ;;TYPE LOW ORDER
 8577  047704  104402                           TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
 8578  047706  000137  047364                   JMP     CRCENT                  ;TRY AGAIN
 8579                                 ;*THIS ROUTINE ISSUES A LINE RESET TO THE CURRENT LINE
 8580                                 ;*BEING EXERCISED.
 8581
 8582  047712                         .LRESET:
 8583  047712  013737  001242  047724           MOV     CLINE,10$               ;LOAD THE CURRENT LINE NUMBER
 8584  047720  104416  000011                   WRITE   ,PRI11
 8585  047724  000000                 10$:      .WORD   0
 8586  047726  104416  000012  000001           WRITE   ,PRI12,BIT0                             ;ISSUE A LINE RESET PULSE
 8587  047734  000002                           EXITT
 8588                                 ;;*************************************************************************
 8589                                           ;THIS ROUTINE WILL TICK THE MAINTENACE CLOCK
 8590                                           ; 'N' NUMBER OF TIMES, 'N' WILL BE ON THE STACK
 8591                                           ;
 8592                                 ;;*************************************************************************
 8593
 8594  047736  017637  000000  050046   .TICK:  MOV     @(SP),67$               ;GET 'N'
 8595  047744  062716  000002                   ADD     #2,(SP)                 ;BUMP THE STACK
 8596  047750  005737  050046                   TST     67$
 8597  047754  003435                           BLE     .TICKEX
 8598  047756  104420  000014                   READ    ,PRI14                  ;GET CONTENTS OF PRIMARY 14
 8599  047762  012637  050042                   MOV     (SP)+,66$               ;;POP STACK INTO 66$
 8600  047766  042737  177400  050042           BIC     #177400,66$             ;CLEAR ERRONIOUSE DATA
 8601  047774  052737  000040  050042   64$:    BIS     #MA,66$                 ;SET MA BIT
 8602  050002  004737  050036                   JSR     PC,65$                  ;ISSUE HALF TICK
 8603  050006  005337  050046                   DEC     67$                     ;COUNT TICKS
 8604  050012  001416                           BEQ     .TICKEX                 ;DONE EXIT
```

```
 8605  050014  042737  000040  050042          BIC     #MA,66$         ;RESET MA
 8606  050022  004737  050036                  JSR     PC,65$          ;ISSUE OTHER HALF TICK
 8607  050026  005337  050046                  DEC     67$             ;COUNT TICK
 8608  050032  001360                          BNE     64$             ;NOT ZERO CONTINUE
 8609  050034  000405                          BR      .TICKEX         ;YES EXIT
 8610  050036  104416  000014          65$:    WRITE   ,PRI14
 8611  050042  000000                  66$:    0
 8612  050044  000207                          EXITS
 8613  050046  000000                  67$:    0                       ;TICK COUNT
 8614  050050  000002                  .TICKEX:        EXITT
 8615                                  .SBTTL  TRAP DECODER
 8616
 8617                                  ;;**********************************************************************
 8618                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
 8619                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
 8620                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
 8621                                  ;*GO TO THAT ROUTINE.
 8622
 8623  050052  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
 8624  050054  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
 8625  050060  005740                          TST     -(R0)           ;;BACKUP BY 2
 8626  050062  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
 8627  050064  006300                          ASL     R0              ;;POSITION FOR INDEXING
 8628  050066  016000  050106                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
 8629  050072  000200                          RTS     R0              ;;GO TO ROUTINE
 8630
 8631
 8632                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
 8633
 8634  050074  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
 8635  050076  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
 8636  050104  000002                          RTI                     ;;RESTORE THE PSW
 8637
 8638                                  .SBTTL  TRAP TABLE
 8639
 8640                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
 8641                                  ;*BY THE "TRAP" INSTRUCTION.
 8642
 8643                                  ;       ROUTINE
 8644                                  ;       -------
 8645  050106  050074                  $TRPAD: .WORD   $TRAP2
 8646  050110  043762                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
 8647  050112  044300                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
 8648  050114  044254                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
 8649  050116  044314                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
 8650  050120  044502                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
 8651
 8652  050122  045326                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
 8653
 8654  050124  045236                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
 8655  050126  045600                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
 8656  050130  045670                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
 8657  050132  046224                          $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
 8658  050134  046364                          $RDDEC  ;;CALL=RDDEC    TRAP+13(104413) READ A DECIMAL NUMBER FROM TTY
 8659  050136  047172                          .MSTCLR ;;CALL=MSTCLR   TRAP+14(104414) CALL TO ISSUE A MASTER CLEAR
 8660  050140  047236                          .ROMCLK ;;CALL=ROMCLK   TRAP+15(104415) CALL TO CLOCK ROM ONCE
```

```
 8661  050142  046674                          .WRITE  ;;CALL=WRITE    TRAP+16(104416) CALL TO WRITE ROUTINE
 8662  050144  047712                          .LRESET ;;CALL=LRESET   TRAP+17(104417) CALL TO LINE RESET ROUTINE
 8663  050146  047022                          .READ   ;;CALL=READ     TRAP+20(104420) CALL TO READ ROUTINE
 8664  050150  047736                          .TICK   ;;CALL=STEP     TRAP+21(104421) STEP CLOCK HANDLER
 8665  050152  000000                DIAGED:   .WORD
 8666           000001                          .END
```

```
ABCA   = 000020          1899#   2823    2827
ABCB   = 000040          1900#   2849    2853
ABCC   = 000100          1901#   2875    2879
APA    = 000200          1914#
BADCRC  032612           6293    6358#   6419
BCRAD.  001264           2006#
BIT0   = 000001          1751#   1868    1874    1875    1894    1907    1915    1923    1931    1984    2420    2421    2455
                         2456    2493    8476    8477    8586
BIT00  = 000001          1741#   1751
BIT01  = 000002          1740#   1750
BIT02  = 000004          1739#   1749
BIT03  = 000010          1738#   1748
BIT04  = 000020          1737#   1747
BIT05  = 000040          1736#   1746
BIT06  = 000100          1735#   1745
BIT07  = 000200          1734#   1744
BIT08  = 000400          1733#   1743    7735
BIT09  = 001000          1732#   1742    7633    7741
BIT1   = 000002          1750#   1893    1895    1903    1908    1916    1932    1974    2420    2421    2455    2456    2493
                         2529    2925    3156    6320    6454    6470    6484    8473    8476    8477
BIT10  = 002000          1731#   7594
BIT11  = 004000          1730#   8412    8446
BIT12  = 010000          1729#
BIT13  = 020000          1728#   7601
BIT14  = 040000          1727#   7721
BIT15  = 100000          1726#
BIT2   = 000004          1749#   1892    1896    1897    1904    1909    1917    1933    1973    2420    2421    2455    2456
                         2493    2529    6321    6336    6348    8477
BIT3   = 000010          1748#   1883    1887    1898    1905    1910    1918    1934    1971    2420    2421    2455    2456
                         2493    2532    2533
BIT4   = 000020          1747#   1867    1871    1899    1911    1919    2493    2631    8411
BIT5   = 000040          1746#   1889    1891    1900    1912    1920    1972    2631
BIT6   = 000100          1745#   1869    1872    1888    1901    1913    1921    2493    2631    4357    8463
BIT7   = 000200          1744#   1870    1885    1890    1902    1906    1914    1922    2411    2420    2421    2446    2484
                         2493    2520    5620
BIT8   = 000400          1743#   2939    3170
BIT9   = 001000          1742#
BOP    = 000000          1873#   4016    4157    4366    4502    4673    4844    5696    5980    6049    6110    6182    6278
                         6404    6533    6659
BPTVEC= 000014           1758#
BSEL0  = 000000          1963#
BSEL1  = 000001          1964#
BSEL2  = 000002          1965#
BSEL3  = 000003          1966#
BSEL4  = 000004          1967#
BSEL5  = 000005          1968#
BSEL6  = 000006          1969#
BSEL7  = 000007          1970#
CALFLG  047362           8516#   8517*   8528*   8545
CCITT  = 000001          1879#   4488    4501    4502    4504    4505    4549    4555    4571    4590    4659    4672    4673
                         4675    4676    4720    4726    4742    4761    5001    5014    5015    5017    5018    5063    5069
                         5085    5104    5131    5136    5137    5151    5164    5165    5167    5168    5213    5219    5235
                         5254    5281    5286    5287
CCP    = 000100          1872#   3858    4260    5015    5165    5315    5465    5574    5814    5914
CHRADR  041060           7211#   7242*   7243    7399*   7400
CKSWR  = 104407          7590    7632    7715    8654#
```

```
CLINE   001242          1981# 2174* 7566* 7567* 7568* 7569* 7570* 8468  8583
CLINED  001244          1982# 2175* 2201  7149  7152  7155  7158  7160  7162  7165  7168  7553* 7563*
                        7566
CLKDAT  001260          1988# 6176  7530* 7534
CMFLAG  001250          1984# 2173*
CONFIG  041066          7214# 7229  7343*
CR    = 000015          1666# 7831  7841
CRCCAL  047462          6551  6738  8528#
CRCEN = 000010          1883#
CRCENT  047364          8505  8517# 8578
CRCHK = 000040          1972# 6279  6405  6534
CRCNT   047360          8515# 8531* 8540* 8555* 8560*
CRC16 = 000003          1880# 4830  4843  4844  4846  4847  4891  4897  4913  4932  5301  5314  5315
                        5317  5318  5363  5369  5385  5404  5431  5436  5437
CRC32 = 000007          1884#
CRC32F  001256          1987# 2481  2530  6273  6399  6526  6650  7531* 7536*
CRLF  = 000200          1667# 2147  7785  7841
CUNIT   041062          7212# 7269* 7304
CURTAB  041070          2177* 7215# 7221* 7280* 7306  7420  7430* 7518  7522* 7526* 7539
CYCLE   002576          2182  2193# 6774
DDISP = 177570          1673# 1824  2118
DF1     041012          2007  2009  2011  2013  2025  2027  2085  7172#
DF2     041022          2015  2017  2019  2021  2023  2029  2031  2033  2035  2037  2039  2041  2043
                        2045  2047  2051  2053  2055  2089  7175#
DF4     041030          2057  2059  2061  2063  2065  2067  2069  2073  2087  7177#
DF5     041034          2049  2071  2075  2077  2079  7179#
DF6     041041          2081  7181#
DF7     041047          2083  7183#
DH1     040140          2007  2009  2011  2013  2025  2027  7094#
DH2     040216          2015  2017  2019  2021  2023  2029  2031  2033  2035  2037  2039  2041  2043
                        2045  2047  2051  2053  2055  2089  7102#
DH4     040266          2057  2059  2061  2063  2065  2067  2069  2073  2087  7109#
DH5     040322          2049  2071  2075  2077  2079  7114#
DH6     040366          2081  7120#
DH7     040436          2083  7127#
DH8     040526          2085  7137#
DIAGED  050152          8665#
DISCRC= 000007          1925# 6278  6404  6533  6659
DISPLA  001142          1824# 2118* 2126* 7593* 7751*
DISPRE  000174          1792# 2126
DSWR  = 177570          1672# 1823  2117
DT1     040624          2007  2009  2011  2013  2025  2027  7149#
DT2     040644          2015  2017  2019  2021  2023  2045  2051  7152#
DT3     040662          2029  2031  2033  2035  2037  2039  2041  2043  2047  2053  2055  2089  7155#
DT4     040700          2057  2059  2061  2063  2065  2067  2069  2073  2087  7158#
DT5     040712          2049  2071  2075  2077  2079  7160#
DT6     040726          2081  7162#
DT7     040744          2083  7165#
DT8     040766          2085  7168#
ECRC  = 000010          1971# 6289  6415  6535  6660
EMTVEC= 000030          1761# 2104* 2105*
EM1     035124          2007  6814#
EM10    035657          2025  6876#
EM11    035726          2027  6883#
EM12    036010          2029  6892#
EM13    036064          2031  6900#
```

```
EM14    036107              2033    6904#
EM15    036147              2035    6910#
EM16    036207              2037    6916#
EM17    036231              2039    6920#
EM18    036312              2041    6929#
EM19    036372              2043    6937#
EM2     035153              2009    6818#
EM20    036415              2045    6941#
EM21    036462              2047    6948#
EM22    036524              2049    6954#
EM23    036611              2051    6963#
EM24    036646              2053    6968#
EM25    036704              2055    6973#
EM26    036754              2057    6980#
EM27    037022              2059    6987#
EM28    037104              2061    6996#
EM29    037155              2063    7003#
EM3     035200              2011    6822#
EM30    037235              2065    7012#
EM31    037315              2067    7021#
EM32    037367              2069    7029#
EM33    037445              2071    7037#
EM34    037504              2073    7043#
EM35    037526              2075    7046#
EM36    037560              2077    7051#
EM37    037627              2079    7058#
EM38    037676              2081    7065#
EM39    037742              2083    7071#
EM4     035236              2013    6827#
EM40    040006              2085    7077#
EM41    040113              2087    7090#
EM42    040046              2089    7083#
EM5     035311              2015    6835#
EM6     035356              2017    6842#
EM7     035420              2019    6848#
EM8     035466              2021    6855#
EM9     035564              2023    6866#
EOT   = 000004              1929#
ERRCK = 000200              1885#   1902#   2901    2905    4557    4558    4562    4598    4599    4603    4638    4639    4643
                            4728    4729    4733    4769    4770    4774    4809    4810    4814    4899    4900    4904    4940
                            4941    4945    4980    4981    4985    5071    5072    5076    5130    5131    5135    5221    5222
                            5226    5280    5281    5285    5371    5372    5376    5430    5431    5435    5541    5542    5546
                            5651    5652    5656
ERRVEC= 000004              1754#   2115    2116*   2127*   7726    7727*   7729*   7732*   8373    8374    8378*   8379*   8386*
                            8387*
ETX   = 000003              1930#
EVENP = 000005              1882#   5560    5573    5574    5576    5620    5627    5628    5632    5652    5657
EXADD = 000020              1919#
EXCON = 000010              1918#
EXIT  = 000207              1863#
EXITI = 000002              1865#
EXITS = 000207              1866#   8496    8612
EXITT = 000002              1864#   8422    8460    8471    8478    8587    8614
GETKMC  042630              2179    7518#
GETLIN  043002              2194    7553#
GNS   = ****** U            1791    2146    2154    2167    2189    2199    2207    2213    7227    7235    7283    7290    7299
```

```
                        7312    7319    7324    7332    7341    7382    7407    8520    8566    8573    8646    8647    8648
                        8649    8650    8652    8654    8655    8656    8657    8658    8659    8660    8661    8662    8663
                        8664
GTSWR = 104406          2141    8652#
HT    = 000011          1664#   7783    7841
INCL  = 005227          1924#
INIT  = 000001          1931#
IOTVEC= 000020          1759#   2102*   2103*
KIPAR0= ****** U        8365
KMCNT   041064          2183    7213#   7253*   7259*
KMCP01  042470          7450#
KMCP02  042476          7454#
KMCP03  042504          7458#
KMCP04  042512          7462#
KMCP05  042520          7466#
KMCP06  042526          7470#
KMCP07  042534          7474#
KMCP08  042542          7478#
KMCP09  042550          7482#
KMCP10  042556          7486#
KMCP11  042564          7490#
KMCP12  042572          7494#
KMCP13  042600          7498#
KMCP14  042606          7502#
KMCP15  042614          7506#
KMCP16  042622          7510#
KMCR01  042466          7449#
KMCR02  042474          7453#
KMCR03  042502          7457#
KMCR04  042510          7461#
KMCR05  042516          7465#
KMCR06  042524          7469#
KMCR07  042532          7473#
KMCR08  042540          7477#
KMCR09  042546          7481#
KMCR10  042554          7485#
KMCR11  042562          7489#
KMCR12  042570          7493#
KMCR13  042576          7497#
KMCR14  042604          7501#
KMCR15  042612          7505#
KMCR16  042620          7509#
KMCS01  042472          7451#
KMCS02  042500          7455#
KMCS03  042506          7459#
KMCS04  042514          7463#
KMCS05  042522          7467#
KMCS06  042530          7471#
KMCS07  042536          7475#
KMCS08  042544          7479#
KMCS09  042552          7483#
KMCS10  042560          7487#
KMCS11  042566          7491#
KMCS12  042574          7495#
KMCS13  042602          7499#
KMCS14  042610          7503#
```

```
KMCS15  042616            7507#
KMCS16  042624            7511#   7520
KMCTAB  042466            2177    7216    7220    7254    7279    7447#   7522    7540
KMCTBE  042626            7218    7261    7424    7513#
LF    = 000012            1665#   7835    7841
LINBYT  001254            1986#   7528*   7556    8485
LM    = 000040            1891#   6110
LNLOCK= 000004            1933#
LNRST = 000001            1923#   6344    6480
LN0   = 000000            1951#
LN1   = 000040            1952#
LN2   = 000100            1953#
LN3   = 000140            1954#
LN4   = 000200            1955#
LN5   = 000240            1956#
LN6   = 000300            1957#
LN7   = 000340            1958#
LOCK  = 000002            1932#
LOOP    002452            2159    2171#   2192
LRESET= 104417            2294    2317    2558    2580    2602    2626    2649    2671    2693    2718    2744    2770    2796
                          2822    2848    2874    2900    2941    2967    2989    3015    3037    3063    3085    3111    3133
                          3172    3198    3220    3246    3268    3294    3316    3342    3364    3390    3412    3438    3460
                          3486    3508    3535    3557    3583    3605    3631    3653    3679    3701    3727    3749    3775
                          3797    8662#
LTEST   002714            2215    2217#
MA    = 000040            1889#   8601    8605
MASK    047352            2354    2355    2388    2389    8493*   8494*   8498#
MB    = 000100            1888#   3861    4020    4158    4262    4370    4509    4680    4851    5022    5172    5322    5470
                          5579    5697    5816    5916    5982    6051    6111    6183    6289    6415    6535    6660
MC    = 000200            1890#   3861    4020    4158    4262    4370    4509    4680    4851    5022    5172    5322    5470
                          5579    5697    5816    5916    5982    6051    6111
MODTAB  041134            2162    2191    7226    7228#   7229#   7344    7433
MSTCLR= 104414            2413    2448    2486    2522    2928    3159    3857    4015    4152    4255    4360    4499    4670
                          4841    5012    5162    5312    5462    5571    5690    5808    5912    5978    6048    6109    6181
                          6277    6403    6532    6656    8659#
NIDLE = 000010            1910#   3343    3346    3347    3356    3363    3367
NOTR    041122            7224#   7251    7271    7273
NXTLIN  001252            1985#   7547*   7554    7556    7558*   7561    7571*
NXTUN   002512            2178#   2195    6808
ODDP  = 000004            1881#   5451    5464    5465    5467    5511    5517    5518    5522    5542    5547
PIRQ  = 177772            1671#
PIRQVE= 000240            1765#
PRI10 = 000010            1935#   8420    8453
PRI11 = 000011            1936#   2241    2242    2245    2410    2415    2417    8469    8584
PRI12 = 000012            1937#   2272    2273    2276    2445    2450    2452    5701    5820    6344    6480    8418    8451
                          8586
PRI13 = 000013            1938#   2295    2296    2299    2483    2488    2490    3866    3878    3892    3904    3917    3937
                          4025    4037    4052    4066    4086    4164    4171    4184    4200    4208    4268    4274    4287
                          4303    4311    4377    4385    4398    4432    4441    4454    4515    4523    4536    4573    4608
                          4686    4694    4707    4744    4779    4857    4865    4878    4915    4950    5028    5036    5050
                          5087    5113    5178    5186    5200    5237    5263    5328    5336    5350    5387    5413    5476
                          5484    5498    5524    5585    5593    5607    5634    5708    5724    5736    5761    5826    5843
                          5855    5880    6192    6207    6279    6297    6405    6423    6534    6573    6580    6661    6684
                          6692    6709    6724
PRI14 = 000014            1939#   2318    2319    2322    2519    2524    2526    3861    4020    4158    4262    4370    4509
                          4680    4851    5022    5172    5322    5470    5579    5697    5816    5916    5982    6051    6111
```

```
                    6183    6281    6289    6317    6331    6333    6345    6407    6415    6451    6465    6467    6481
                    6535    6538    6591    6660    6662    8598    8610
PRI15 = 000015      1940#
PRI16 = 000016      1941#   2349    2350    2353
PRI17 = 000017      1942#   2383    2384    2387
PROTEX= 000001      1907#   3199    3202    3203    3212    3219    3223
PROTEY= 000002      1908#   3247    3250    3251    3260    3267    3271
PROTEZ= 000004      1909#   3295    3298    3299    3308    3315    3319
PROTO = 000100      1913#   3487    3490    3491    3500    3507    3511
PR0   = 000000      1688#
PR1   = 000040      1689#
PR2   = 000100      1690#
PR3   = 000140      1691#
PR4   = 000200      1692#
PR5   = 000240      1693#
PR6   = 000300      1694#
PR7   = 000340      1695#
PS    = 177776      1668#   1669
PSW   = 177776      1669#   2151*   2172*
PT      041166      7222    7237    7238#
PWRVEC= 000024      1760#
RAC   = 000004      1892#   3939    3940    3944    4088    4089    4093
RCRCE = 000002      1974#   6283    6286    6409    6412    6471    6485    6540    6543    6593    6594    6598    6664
                    6665    6669    6671
RDA   = 000001      1875#   3939    3949    3953    4088    4098    4102    4189    4210    4292    4313    4417    4443
                    4456    4544    4581    4610    4715    4752    4781    4886    4923    4952    5058    5095    5115
                    5208    5245    5265    5358    5395    5415    5506    5526    5615    5636    5729    5738    5763
                    5848    5857    6574    6685    6710    6725
RDATA1= 000001      1915#   3536    3539    3540    3549    3556    3560
RDATA2= 000002      1916#   3584    3587    3588    3597    3604    3608
RDATA3= 000004      1917#   3632    3635    3636    3645    3652    3656
RDCHR = 104410      7241    7395    8191    8655#
RDDEC = 104413      7267    8658#
RDLIN = 104411      7347    8265    8318    8656#
RDOCT = 104412      2156    7389    8522    8657#
READ  = 104420      2242    2273    2296    2319    2350    2384    2417    2452    2490    2526    2560    2582    2604
                    2628    2651    2673    2695    2721    2747    2773    2799    2825    2851    2877    2903    2931
                    2942    2969    2979    2990    3017    3027    3038    3065    3075    3086    3113    3123    3134
                    3162    3173    3200    3210    3221    3248    3258    3269    3296    3306    3317    3344    3354
                    3365    3392    3402    3413    3440    3450    3461    3488    3498    3509    3537    3547    3558
                    3585    3595    3606    3633    3643    3654    3681    3691    3702    3729    3739    3750    3777
                    3787    3798    3866    3878    3892    3904    3917    3937    3958    4025    4037    4052    4066
                    4086    4107    4118    4164    4171    4184    4191    4200    4208    4212    4217    4268    4274
                    4287    4294    4303    4311    4315    4377    4385    4398    4419    4432    4441    4445    4454
                    4458    4461    4515    4523    4536    4546    4555    4573    4588    4596    4608    4612    4625
                    4636    4686    4694    4707    4717    4726    4744    4759    4767    4779    4783    4796    4807
                    4857    4865    4878    4888    4897    4915    4930    4938    4950    4954    4967    4978    5028
                    5036    5050    5060    5069    5087    5102    5113    5117    5128    5178    5186    5200    5210
                    5219    5237    5252    5263    5267    5278    5328    5336    5350    5360    5369    5387    5402
                    5413    5417    5428    5476    5484    5498    5508    5524    5528    5539    5585    5593    5607
                    5617    5634    5638    5649    5701    5708    5724    5736    5740    5750    5761    5765    5769
                    5820    5826    5843    5855    5859    5869    5880    5933    5937    5949    6001    6005    6017
                    6069    6072    6075    6078    6126    6129    6132    6135    6192    6207    6218    6281    6297
                    6317    6333    6345    6407    6423    6439    6446    6451    6467    6481    6538    6573    6576
                    6580    6591    6662    6684    6687    6692    6709    6712    6724    6727    8453    8598    8663#
REGIST  001240      1977#   2241*   2272*   2295*   2318*   2349*   2383*   2410*   2445*   2483*   2519*   2559*   2581*
```

```
                    2603* 2627* 2650* 2672* 2694* 2720* 2746* 2772* 2798* 2824* 2850* 2876* 2902*
                    2926* 3157* 7149
RENA  = 000001      1874# 3861  4020  4158  4262  4370  4509  4680  4851  5022  5172  5322  5470
                    5579  5697  5816  5916  5982  6051  6111  6183  6289  6415  6535  6660
REOM  = 000002      1895# 2745  2749  4219  4220  4224  4463  4464  4468  4627  4628  4632  4798
                    4799  4803  4969  4970  4974
RESVEC= 000010      1755#
REXIT   047170      8444  8460#
RINST   047072      8429* 8435* 8437#
RINST1  047146      8445* 8450* 8452#
ROMCLK= 104415      8407  8436  8660#
ROR   = 000010      1898# 2797  2801  5939  5940  5944  5951  5952  5956  6007  6008  6012  6019
                    6020  6024
RPRI    047040      8429#
RSA   = 000002      1893#
RSEC    047116      8428  8445#
RSOM  = 000001      1894# 2719  2723  4109  4110  4114
RXAB  = 000004      1897# 5771  5772  5776  6080  6081  6085
RXGA  = 000004      1896# 2771  2775  6137  6138  6142
SAM   = 000020      1911# 3391  3394  3395  3404  3411  3415  4016  5980  6049
SEC0  = 000000      1943# 2559  2560  3958  4118  4191  4212  4294  4315  4419  4445  4458  4546
                    4588  4612  4717  4759  4783  4888  4930  4954  5060  5102  5117  5210  5252
                    5267  5360  5402  5417  5508  5528  5617  5638  5740  5765  5859  5933  6001
                    6069  6072  6075  6126  6129  6132  6218  6439  6446  6576  6687  6712  6727
SEC1  = 000001      1944# 2581  2582  2719  2720  2721  2745  2746  2747  2771  2772  2773  2797
                    2798  2799  2823  2824  2825  2849  2850  2851  2875  2876  2877  2901  2902
                    2903  4107  4217  4461  4555  4596  4625  4636  4726  4767  4796  4807  4897
                    4938  4967  4978  5069  5128  5219  5278  5369  5428  5539  5649  5769  5937
                    5949  6005  6017  6078  6135
SEC2  = 000002      1945# 2603  2604  2926  2929  2931  2942  3890  3929  3931  3933  3935  4050
                    4064  4078  4080  4082  4084  4168  4188  4272  4291  4382  4402  4520  4540
                    4691  4711  4862  4882  5033  5054  5183  5204  5333  5354  5481  5502  5590
                    5611  5705  5714  5728  5824  5832  5847  5923  5925  5927  5929  5935  5989
                    5991  5993  5995  5997  6003  6058  6060  6062  6064  6066  6118  6120  6122
                    6188  6309  6436  6563  6680
SEC3  = 000003      1946# 2627  2628  2968  2969  2978  2979  2988  2990  3016  3017  3026  3027
                    3036  3038  3064  3065  3074  3075  3084  3086  3112  3113  3122  3123  3132
                    3134  3865  3902  4024  4049  4162  4169  4198  4266  4285  4301  4375  4383
                    4430  4513  4521  4571  4684  4692  4742  4855  4863  4913  5026  5048  5085
                    5176  5198  5235  5326  5348  5385  5474  5496  5583  5605  5699  5706  5750
                    5818  5841  5869  5920  5922  5986  5988  6055  6057  6067  6115  6117  6124
                    6185  6187  6290  6292  6314  6416  6418  6443  6546  6548  6586  6588  6674
                    6676  6699  6701
SEC4  = 000004      1947# 2650  2651  3157  3160  3162  3173  3860  4019  4261  5016  5166  5316
                    5466  5575  5815  5915  5981  6050
SEC5  = 000005      1948# 2672  2673  3199  3200  3209  3210  3219  3221  3247  3248  3257  3258
                    3267  3269  3295  3296  3305  3306  3315  3317  3343  3344  3353  3354  3363
                    3365  3391  3392  3401  3402  3411  3413  3439  3440  3449  3450  3459  3461
                    3487  3488  3497  3498  3507  3509  3858  4016  4157  4260  4366  4502  4673
                    4844  5015  5165  5315  5465  5574  5696  5814  5914  5980  6049  6110  6182
                    6278  6404  6533  6659
SEC6  = 000006      1949#
SEC7  = 000007      1950# 2694  2695  3536  3537  3546  3547  3556  3558  3584  3585  3594  3595
                    3604  3606  3632  3633  3642  3643  3652  3654  3680  3681  3690  3691  3700
                    3702  3728  3729  3738  3739  3748  3750  3776  3777  3786  3787  3796  3798
                    4156  4259  4368  4406  4501  4672  4843  5014  5164  5314  5464  5573  5695
```

```
                    5813    5913    5979
SEL0   = 000000     1959#
SEL2   = 000002     1960#
SEL4   = 000004     1961#
SEL6   = 000006     1962#
SETTAB   041072     7216#   7231
SF     = 000010     1887#   3894    3895    3899    3906    3907    3911    3919    3920    3924    4054    4055    4059
                    4068    4069    4073    4173    4179    4202    4276    4282    4305    4387    4393    4410    4434
                    4525    4531    4575    4616    4696    4702    4746    4787    4867    4873    4917    4958    5038
                    5044    5089    5188    5194    5239    5338    5344    5389    5486    5492    5595    5601    5710
                    5719    5828    5837    5882    5883    5887
SOH    = 000001     1886#   1928#   5923
SSL    = 000040     1912#   3439    3442    3443    3452    3459    3463    5015    5165    5315    5465    5574
STACK  = 001100     1659#   2100    2150    2171
STEP   = 104421     3877    3891    3903    3916    3930    3932    3934    3936    4036    4051    4065    4079    4081
                    4083    4085    4163    4170    4183    4199    4216    4267    4273    4286    4302    4376    4384
                    4397    4431    4453    4514    4522    4535    4572    4607    4624    4685    4693    4706    4743
                    4778    4795    4856    4864    4877    4914    4949    4966    5027    5035    5049    5086    5112
                    5177    5185    5199    5236    5262    5327    5335    5349    5386    5412    5475    5483    5497
                    5523    5584    5592    5606    5633    5700    5707    5723    5735    5760    5819    5825    5842
                    5854    5879    5921    5924    5926    5928    5930    5936    5987    5990    5992    5994    5996
                    5998    6004    6056    6059    6061    6063    6065    6068    6071    6074    6077    6116    6119
                    6121    6123    6125    6128    6131    6134    6186    6191    6206    6291    6311    6315    6417
                    6438    6445    6547    6570    6675    6683    6706    6721    8664#
STKLMT= 177774      1670#
STX    = 000002     1927#   3890    4272    5824
SWR      001140     1823#   2098    2117*   2119    2125*   2139    7594    7601    7629    7633    7721    7735    7737
                    7741    8035    8074    8129*
SWREG    000176     1793#   2125    2139    8035    8074    8097
SW0    = 000001     1723#
SW00   = 000001     1713#   1723
SW01   = 000002     1712#   1722
SW02   = 000004     1711#   1721
SW03   = 000010     1710#   1720
SW04   = 000020     1709#   1719
SW05   = 000040     1708#   1718
SW06   = 000100     1707#   1717
SW07   = 000200     1706#   1716
SW08   = 000400     1705#   1715
SW09   = 001000     1704#   1714
SW1    = 000002     1722#
SW10   = 002000     1703#
SW11   = 004000     1702#
SW12   = 010000     1701#
SW13   = 020000     1700#
SW14   = 040000     1699#
SW15   = 100000     1698#
SW2    = 000004     1721#
SW3    = 000010     1720#
SW4    = 000020     1719#
SW5    = 000040     1718#
SW6    = 000100     1717#
SW7    = 000200     1716#
SW8    = 000400     1715#
SW9    = 001000     1714#
SYN    = 000026     1926#   3860    3960    4261    5016    5166    5316    5466    5575    5815    5915
```

```
TAC    = 000100          1869#
TBITVE= 000014           1756#
TBMT   = 000020          1871#  2494   3868   3869   3873   3880   3881   3885   4027   4028   4032   4039   4040
                         4044   4166   4186   4270   4289   4379   4400   4517   4538   4688   4709   4859   4880
                         5030   5052   5180   5202   5330   5352   5478   5500   5587   5609   5703   5712   5726
                         5822   5830   5845   6299   6300   6304   6425   6426   6430   6581   6693
TCRCE  = 000004          1973#  6283   6286   6337   6349   6409   6412   6540   6543   6593   6594   6598   6664
                         6665   6669   6671
TDATA1= 000040           1920#  3680   3683   3684   3693   3700   3704
TDATA2= 000100           1921#  3728   3731   3732   3741   3748   3752
TDATA3= 000200           1922#  3776   3779   3780   3789   3796   3800
TENA   = 000020          1867#  3861   4020   4158   4262   4370   4509   4680   4851   5022   5172   5322   5470
                         5579   5697   5816   5916   5982   6051   6111   6183   6289   6415   6535   6660
TEOM   = 000002          1903#  3016   3019   3020   3029   3036   3040   4198   4301   4430   4571   4742   4913
                         5085   5235   5385   6067   6124   6314   6443   6586   6699
TERR   = 000200          1906#  5752   5753   5757   5871   5872   5876
TKVEC  = 000060          1763#  8007*  8008*
TLOCK  = 000010          1934#
TPVEC  = 000064          1764#
TRAPVE= 000034           1762#  2106*  2107*
TRTVEC= 000014           1757#
TSA    = 000200          1870#  5743   5862
TSOM   = 000001          1868#  2968   2971   2972   2981   2988   2992   3865   4024   4162   4266   4375   4513
                         4684   4855   5026   5176   5326   5474   5583   5699   5818   5920   5986   6055   6115
                         6185   6290   6416   6546   6674
TSTNUM  001262           1989#  2231*  2262*  2293*  2316*  2339*  2373*  2409*  2444*  2480*  2518*  2557*  2579*
                         2601*  2625*  2648*  2670*  2692*  2717*  2743*  2769*  2795*  2821*  2847*  2873*  2899*
                         2924*  2966*  3014*  3062*  3110*  3155*  3197*  3245*  3293*  3341*  3389*  3437*  3485*
                         3534*  3582*  3630*  3678*  3726*  3774*  3855*  4013*  4151*  4254*  4353*  4496*  4667*
                         4838*  5009*  5159*  5309*  5459*  5568*  5689*  5807*  5911*  5977*  6047*  6108*  6175*
                         6271*  6397*  6525*  6649*  6773*  7149   7152   7155   7158   7160   7162   7165   7168
TST1     002714          2229#
TST10    003704          2429   2442#
TST100   031512          6147   6173#
TST101   032060          6229   6269#
TST102   032620          6351   6395#
TST103   033374          6487   6523#
TST104   034112          6619   6647#
TST105   034764          6764   6771#
TST11    004030          2464   2478#
TST12    004162          2482   2502   2516#
TST13    004322          2542   2555#
TST14    004404          2564   2577#
TST15    004466          2586   2599#
TST16    004550          2608   2623#
TST17    004640          2633   2646#
TST2     003024          2246   2260#
TST20    004722          2655   2668#
TST21    005004          2677   2690#
TST22    005066          2699   2715#
TST23    005162          2726   2741#
TST24    005256          2752   2767#
TST25    005352          2778   2793#
TST26    005446          2804   2819#
TST27    005542          2830   2845#
TST3     003134          2277   2291#
```

```
TST30   005636            2856    2871#
TST31   005732            2882    2897#
TST32   006026            2908    2922#
TST33   006176            2946    2964#
TST34   006404            2995    3012#
TST35   006612            3043    3060#
TST36   007020            3091    3108#
TST37   007226            3139    3153#
TST4    003216            2300    2314#
TST40   007376            3177    3195#
TST41   007604            3226    3243#
TST42   010012            3274    3291#
TST43   010220            3322    3339#
TST44   010426            3370    3387#
TST45   010634            3418    3435#
TST46   011042            3466    3483#
TST47   011250            3514    3532#
TST5    003276            2323    2337#
TST50   011456            3563    3580#
TST51   011664            3611    3628#
TST52   012072            3659    3676#
TST53   012300            3707    3724#
TST54   012506            3755    3772#
TST55   012714            3803    3853#
TST56   013616            3962    4011#
TST57   014522            4122    4149#
TST6    003430            2357    2371#
TST60   015204            4228    4252#
TST61   015606            4319    4351#
TST62   016542            4479    4482    4494#
TST63   017700            4653    4665#
TST64   021036            4824    4836#
TST65   022174            4995    5007#
TST66   023176            5145    5157#
TST67   024200            5295    5307#
TST7    003560            2391    2407#
TST70   025202            5445    5457#
TST71   025756            5554    5566#
TST72   026540            5664    5687#
TST73   027326            5784    5805#
TST74   030072            5894    5909#
TST75   030412            5961    5975#
TST76   030744            6029    6045#
TST77   031242            6090    6106#
TXAB  = 000004            1904#   3064    3067    3068    3077    3084    3088    6067
TXGA  = 000010            1905#   3112    3115    3116    3125    3132    3136    6124
TYPDS = 104405            2210    6797    7305    7689    8650#
TYPE  = 104401            2135    2152    2165    2187    2197    2205    2211    6795    6798    7225    7233    7243    7281
                          7288    7297    7310    7315    7317    7322    7330    7339    7380    7386    7387    7400    7405
                          7596    7608    7611    7616    7627    7653    7670    7672    7675    7677    7692    7697    7788
                          7901    7976    8029    8041    8095    8096    8099    8110    8120    8131    8150    8198    8204
                          8209    8213    8218    8219    8221    8224    8228    8294    8296    8355    8357    8518    8564
                          8571    8646#
TYPOC = 104402            7610    7615    7661    7685    8098    8570    8577    8647#
TYPON = 104404            8649#
TYPOS = 104403            2202    7294    8648#
```

```
WINST1  047010          8411*   8416*   8419#
WINST2  047016          8417*   8421#
WPINS   046746          8403*   8405*   8408#
WPRI    046722          8403#
WREX    047020          8409    8422#
WRITE = 104416          2415    2450    2488    2524    2719    2745    2771    2797    2823    2849    2875    2901    2929
                        2968    2978    2988    3016    3026    3036    3064    3074    3084    3112    3122    3132    3160
                        3199    3209    3219    3247    3257    3267    3295    3305    3315    3343    3353    3363    3391
                        3401    3411    3439    3449    3459    3487    3497    3507    3536    3546    3556    3584    3594
                        3604    3632    3642    3652    3680    3690    3700    3728    3738    3748    3776    3786    3796
                        3858    3860    3861    3865    3890    3902    3929    3931    3933    3935    4016    4019    4020
                        4024    4049    4050    4064    4078    4080    4082    4084    4156    4157    4158    4162    4168
                        4169    4188    4198    4259    4260    4261    4262    4266    4272    4285    4291    4301    4366
                        4368    4370    4375    4382    4383    4402    4406    4430    4501    4502    4509    4513    4520
                        4521    4540    4571    4672    4673    4680    4684    4691    4692    4711    4742    4843    4844
                        4851    4855    4862    4863    4882    4913    5014    5015    5016    5022    5026    5033    5048
                        5054    5085    5164    5165    5166    5172    5176    5183    5198    5204    5235    5314    5315
                        5316    5322    5326    5333    5348    5354    5385    5464    5465    5466    5470    5474    5481
                        5496    5502    5573    5574    5575    5579    5583    5590    5605    5611    5695    5696    5697
                        5699    5705    5706    5714    5728    5813    5814    5815    5816    5818    5824    5832    5841
                        5847    5913    5914    5915    5916    5920    5922    5923    5925    5927    5929    5935    5979
                        5980    5981    5982    5986    5988    5989    5991    5993    5995    5997    6003    6049    6050
                        6051    6055    6057    6058    6060    6062    6064    6066    6067    6110    6111    6115    6117
                        6118    6120    6122    6124    6182    6183    6185    6187    6188    6278    6279    6289    6290
                        6292    6309    6314    6331    6344    6404    6405    6415    6416    6418    6436    6443    6465
                        6480    6533    6534    6535    6546    6548    6563    6586    6588    6659    6660    6661    6674
                        6676    6680    6699    6701    8418    8420    8451    8469    8584    8586    8610    8661#
WSEC    046752          8402    8410#
X     = 000001          1876#   1879    1880    1882    1884    7992    8039
Y     = 000002          1877#   1880    1884
Z     = 000004          1878#   1881    1882    1884
$AUTOB  001134          1820#   2143*   8088    8245
$BDADR  001122          1815#
$BDDAT  001126          1817#
$BELL   001230          1854#   7596    7645    8041    8238
$CHARC  044250          7790*   7800*   7807    7833*   7838#
$CKSWR  045236          8074#   8654
$CMTAG  001100          1803#   2095    2096    2104    2108    2109
$CM1  = 000010          1835#   1836#   1837#   1838#   1839#   1840#   1841#   1842#   1843#
$CM2  = 000020          1835#   1836#   1837#   1838#   1839#   1840#   1841#   1842#   1843#
$CM3  = 000010          1833#   1835
$CM4  = 000012          1843#   1844#   1845#   1846#   1847#   1848#   1849#   1850#   1851#   1852#   1853#
$CNTLC  046162          8029    8110    8238#
$CNTLG  046174          8095    8240#
$CNTLU  046167          8120    8213    8239#
$CRAP = 177777          1628#   2218#   2221    2224#   2249#   2252    2255#   2280#   2283    2286#   2303#   2306    2309#
                        2326#   2329    2332#   2360#   2363    2366#   2395#   2398    2402#   2430#   2433    2437#   2465#
                        2468    2473#   2503#   2506    2511#   2544#   2547    2550#   2567#   2570    2572#   2589#   2592
                        2594#   2611#   2614    2618#   2636#   2639    2641#   2658#   2661    2663#   2680#   2683    2685#
                        2703#   2706    2710#   2729#   2732    2736#   2755#   2758    2762#   2781#   2784    2788#   2807#
                        2810    2814#   2833#   2836    2840#   2859#   2862    2866#   2885#   2888    2892#   2912#   2915
                        2917#   2950#   2953    2959#   2998#   3001    3007#   3046#   3049    3055#   3094#   3097    3103#
                        3143#   3146    3148#   3181#   3184    3190#   3229#   3232    3238#   3277#   3280    3286#   3325#
                        3328    3334#   3373#   3376    3382#   3421#   3424    3430#   3469#   3472    3478#   3518#   3521
                        3527#   3566#   3569    3575#   3614#   3617    3623#   3662#   3665    3671#   3710#   3713    3719#
                        3758#   3761    3767#   3807#   3810    3848#   3965#   3968    4006#   4126#   4129    4144#   4229#
```

```
                    4232    4247#   4321#   4324    4346#   4484#   4487    4489#   4655#   4658    4660#   4826#   4829
                    4831#   4997#   5000    5002#   5147#   5150    5152#   5297#   5300    5302#   5447#   5450    5452#
                    5556#   5559    5561#   5667#   5670    5682#   5785#   5788    5800#   5896#   5899    5904#   5962#
                    5965    5970#   6031#   6034    6040#   6092#   6095    6101#
$CRLF   001235      1856#   7387    7616    7627    7645    7653    7672    7677    7697    7789    7841    8131    8218
                    8238    8299    8359
$DBLK   044716      7942    7976    7984#
$DOAGN  035100      6791    6800    6806#
$DTBL   044706      7945    7980#
$DVICE  001246      1983#   2176*   2209    7149    7152    7155    7158    7160    7162    7165    7168    7546*
$ENDAD  035070      2133    6802#   7640
$ENDCT  035036      2108    6793#
$ENDMG  035107      6795    6810#
$ENULL  035104      6798    6809#
$EOP    035006      2185    6784#
$EOPCT  035030      2108*   6790#   6794
$ERFLG  001103      1806#   7591*   7645    7706    7739    7745*   7754
$ERMAX  001115      1812#   2110*   7750*   7754
$ERROR  043114      2104    7589#   7720
$ERRPC  001116      1813#   7149    7152    7155    7158    7160    7162    7165    7168    7598*   7599*   7600    7606*
                    7607*   7609    7613*   7614    7645    7659
$ERRTB  001264      2004#   7667
$ERRTY  043410      7626    7652#
$ERTTL  001112      1810#   7597*   7645
$ESCAP  001226      1853#   2109*   7636    7638    7645    7749*
$FILLC  001156      1831#   7793    7841
$FILLS  001155      1830#   7841
$GDADR  001120      1814#
$GDDAT  001124      1816#
$GET42  035060      6799#
$GTSWR  045326      8096#   8652
$HD   = 000003      1654    1655
$HIOCT  046362      8287*   8298#   8524
$ICNT   001104      1807#
$INTAG  001135      1821#   8093*   8112    8132    8245
$ITEMB  001114      1811#   7600*   7645    7656
$LF     001236      1857#   7645    7841    8228    8238    8299    8359
$LPADR  001106      1808#   2111*   3856*   4014*   4359*   4498*   4669*   4840*   5011*   5161*   5311*   5461*   5570*
                    6178*   6276*   6402*   6529*   6653*   7743*   7747*   7752    7754
$LPERR  001110      1809#   2112*   7635    7743    7748*   7754
$LSTAD  046672      8390*   8394#
$MAIL = ****** U    2129    2139    7629    7747    7777
$MNEW   046212      8099    8243#
$MSWR   046201      8096    8241#
$N    = 000106      1768#   2218    2224    2226    2232#   2249    2255    2257    2263#   2280    2286    2288    2294#
                    2303    2309    2311    2317#   2326    2332    2334    2340#   2360    2366    2368    2374#   2395
                    2402    2404    2410#   2430    2437    2439    2445#   2465    2473    2475    2481#   2503    2511
                    2513    2519#   2544    2550    2552    2558#   2567    2572    2574    2580#   2589    2594    2596
                    2602#   2611    2618    2620    2626#   2636    2641    2643    2649#   2658    2663    2665    2671#
                    2680    2685    2687    2693#   2703    2710    2712    2718#   2729    2736    2738    2744#   2755
                    2762    2764    2770#   2781    2788    2790    2796#   2807    2814    2816    2822#   2833    2840
                    2842    2848#   2859    2866    2868    2874#   2885    2892    2894    2900#   2912    2917    2919
                    2925#   2950    2959    2961    2967#   2998    3007    3009    3015#   3046    3055    3057    3063#
                    3094    3103    3105    3111#   3143    3148    3150    3156#   3181    3190    3192    3198#   3229
                    3238    3240    3246#   3277    3286    3288    3294#   3325    3334    3336    3342#   3373    3382
                    3384    3390#   3421    3430    3432    3438#   3469    3478    3480    3486#   3518    3527    3529
```

```
                    3535#  3566   3575   3577   3583#  3614   3623   3625   3631#  3662   3671   3673   3679#
                    3710   3719   3721   3727#  3758   3767   3769   3775#  3807   3848   3850   3856#  3965
                    4006   4008   4014#  4126   4144   4146   4153#  4229   4247   4249   4256#  4321   4346
                    4348   4354#  4484   4489   4491   4497#  4655   4660   4662   4668#  4826   4831   4833
                    4839#  4997   5002   5004   5010#  5147   5152   5154   5160#  5297   5302   5304   5310#
                    5447   5452   5454   5460#  5556   5561   5563   5569#  5667   5682   5684   5691#  5785
                    5800   5802   5809#  5896   5904   5906   5913#  5962   5970   5972   5979#  6031   6040
                    6042   6049#  6092   6101   6103   6110#  6170   6176#  6266   6272#  6392   6398#  6520
                    6526#  6644   6650#  6768   6774#
$NULL   001154      1829#  7795   7841
$NWTST= 000000      2228#  2259#  2290#  2313#  2336#  2370#  2406#  2441#  2477#  2515#  2554#  2576#  2598#
                    2622#  2645#  2667#  2689#  2714#  2740#  2766#  2792#  2818#  2844#  2870#  2896#  2921#
                    2963#  3011#  3059#  3107#  3152#  3194#  3242#  3290#  3338#  3386#  3434#  3482#  3531#
                    3579#  3627#  3675#  3723#  3771#  3852#  4010#  4148#  4251#  4350#  4493#  4664#  4835#
                    5006#  5156#  5306#  5456#  5565#  5686#  5804#  5908#  5974#  6044#  6105#  6172#  6268#
                    6394#  6522#  6646#  6770#
$OCNT   044476      7873*  7902*  7915#
$OMODE  044500      7868*  7872*  7877   7880*  7891*  7917#
$OVER   043746      7722   7738   7744   7751#
$PASS   001100      1804#  6787*  6788*  6796   6809
$QUES   001234      1855#  7386   7645   7841   8150   8221   8238   8296   8299   8357   8359
$RDCHR  045600      8163#  8655
$RDDEC  046364      8313#  8658
$RDLIN  045670      8186#  8656
$RDOCT  046224      8260#  8657
$RDSZ = 000036      8179#
$REGAD  001160      1833#
$REG0   001162      1835#
$REG1   001164      1836#
$REG2   001166      1837#
$REG3   001170      1838#
$REG4   001172      1839#
$REG5   001174      1840#
$REG6   001176      1841#
$REG7   001200      1842#
$RTNAD  035102      6808#
$R2A  = ****** U    8659
$SAVRE= ****** U    8659
$SCOPE  043564      2102   7714#
$SETUP= 000127      2091#  2101   2102   2104   2106   2108   2109   2111   2133   2136   6786   7590   7632
                    7640   7715   8033   8038   8039   8069   8245
$SIZE   046542      2148   8367#
$STUP = 177777      2091#
$SVLAD  043720      7730   7740   7746#
$SWR  = 163400      1654   1655#  1766#  1853   1854   2109   2111   2112   2230   2261   2292   2315   2338
                    2372   2408   2443   2479   2517   2556   2578   2600   2624   2647   2669   2691   2716
                    2742   2768   2794   2820   2846   2872   2898   2923   2965   3013   3061   3109   3154
                    3196   3244   3292   3340   3388   3436   3484   3533   3581   3629   3677   3725   3773
                    3854   4012   4150   4253   4352   4495   4666   4837   5008   5158   5308   5458   5567
                    5688   5806   5910   5976   6046   6107   6174   6270   6396   6524   6648   6772   6781
                    6787   6801   6807   6809   7581   7582   7583   7584   7585   7594   7601   7629   7633
                    7645   7707   7708   7709   7710   7716   7733   7735   7736   7739   7740   7741   7746
                    7748   7751   7754
$SWRMK= 000000      7710   7711   7737
$TKB    001146      1826#  7814   7821   7841   7988   8009   8020   8050   8078   8105
$TKCNT  044726      7989#  8004*  8039   8056*  8170   8172*
```

```
$TKINT  044736          2149    2233    2264    2341    2375    8004#   8030    8091
$TKQEN= 044735          7993#   8064    8175
$TKQIN  044730          7990#   8005*   8006    8062*   8063*   8064    8066*
$TKQOU  044732          7991#   8006*   8173    8174*   8175    8177*
$TKQSR  044734          7992#   8005    8066    8177
$TKS    001144          1825#   7812    7819    7841    7988    8010*   8046*   8048    8054*   8076    8092*   8102    8114*
                        8134*
$TKSRV  045006          8007    8020#
$TMP0   001202          1843#   2243*   2245    2274*   2276    2297*   2299    2320*   2322    2351*   2354*   2356    2385*
                        2388*   2390    2418*   2420*   2422    2453*   2455*   2457    2491*   2495    2527*   2532*   2535
                        2561*   2563    2583*   2585    2605*   2607    2629*   2631*   2632    2652*   2654    2674*   2676
                        2696*   2698    2722*   2723*   2725    2748*   2749*   2751    2774*   2775*   2777    2800*   2801*
                        2803    2826*   2827*   2829    2852*   2853*   2855    2878*   2879*   2881    2904*   2905*   2907
                        2932*   2934    2943*   2945    2970*   2971*   2973    2980*   2981*   2983    2991*   2992*   2994
                        3018*   3019*   3021    3028*   3029*   3031    3039*   3040*   3042    3066*   3067*   3069    3076*
                        3077*   3079    3087*   3088*   3090    3114*   3115*   3117    3124*   3125*   3127    3135*   3136*
                        3138    3163*   3165    3174*   3176    3201*   3202*   3204    3211*   3212*   3214    3222*   3223*
                        3225    3249*   3250*   3252    3259*   3260*   3262    3270*   3271*   3273    3297*   3298*   3300
                        3307*   3308*   3310    3318*   3319*   3321    3345*   3346*   3348    3355*   3356*   3358    3366*
                        3367*   3369    3393*   3394*   3396    3403*   3404*   3406    3414*   3415*   3417    3441*   3442*
                        3444    3451*   3452*   3454    3462*   3463*   3465    3489*   3490*   3492    3499*   3500*   3502
                        3510*   3511*   3513    3538*   3539*   3541    3548*   3549*   3551    3559*   3560*   3562    3586*
                        3587*   3589    3596*   3597*   3599    3607*   3608*   3610    3634*   3635*   3637    3644*   3645*
                        3647    3655*   3656*   3658    3682*   3683*   3685    3692*   3693*   3695    3703*   3704*   3706
                        3730*   3731*   3733    3740*   3741*   3743    3751*   3752*   3754    3778*   3779*   3781    3788*
                        3789*   3791    3799*   3800*   3802    3867*   3868*   3869    3871    3879*   3880*   3881    3883
                        3893*   3894*   3895    3897    3905*   3906*   3907    3909    3918*   3919*   3920    3922    3938*
                        3939*   3940    3942    3949    3951    3959*   3961    4026*   4027*   4028    4030    4038*   4039*
                        4040    4042    4053*   4054*   4055    4057    4067*   4068*   4069    4071    4087*   4088*   4089
                        4091    4098    4100    4108*   4109*   4110    4112    4119*   4121    4165*   4166    4172*   4173
                        4177    4178    4185*   4186    4189    4192*   4201*   4202    4209*   4210    4213*   4218*   4219*
                        4220    4222    4269*   4270    4275*   4276    4280    4281    4288*   4289    4292    4295*   4304*
                        4305    4312*   4313    4316*   4378*   4379    4386*   4387    4391    4392    4399*   4400    4410
                        4417    4420*   4422    4433*   4434    4442*   4443    4446*   4448    4455*   4456    4459*   4462*
                        4463*   4464    4466    4516*   4517    4524*   4525    4529    4530    4537*   4538    4544    4547*
                        4550    4556*   4557*   4558    4560    4574*   4575    4581    4589*   4591    4597*   4598*   4599
                        4601    4609*   4610    4616    4626*   4627*   4628    4630    4637*   4638*   4639    4641    4687*
                        4688    4695*   4696    4700    4701    4708*   4709    4715    4718*   4721    4727*   4728*   4729
                        4731    4745*   4746    4752    4760*   4762    4768*   4769*   4770    4772    4780*   4781    4787
                        4797*   4798*   4799    4801    4808*   4809*   4810    4812    4858*   4859    4866*   4867    4871
                        4872    4879*   4880    4886    4889*   4892    4898*   4899*   4900    4902    4916*   4917    4923
                        4931*   4933    4939*   4940*   4941    4943    4951*   4952    4958    4968*   4969*   4970    4972
                        4979*   4980*   4981    4983    5029*   5030    5037*   5038    5042    5043    5051*   5052    5058
                        5061*   5064    5070*   5071*   5072    5074    5088*   5089    5095    5103*   5105    5114*   5115
                        5129*   5130*   5131    5133    5179*   5180    5187*   5188    5192    5193    5201*   5202    5208
                        5211*   5214    5220*   5221*   5222    5224    5238*   5239    5245    5253*   5255    5264*   5265
                        5279*   5280*   5281    5283    5329*   5330    5337*   5338    5342    5343    5351*   5352    5358
                        5361*   5364    5370*   5371*   5372    5374    5388*   5389    5395    5403*   5405    5414*   5415
                        5429*   5430*   5431    5433    5477*   5478    5485*   5486    5490    5491    5499*   5500    5506
                        5509*   5512    5525*   5526    5540*   5541*   5542    5544    5586*   5587    5594*   5595    5599
                        5600    5608*   5609    5615    5618*   5620*   5622    5635*   5636    5650*   5651*   5652    5654
                        5702*   5703    5709*   5710    5712    5717    5718    5725*   5726    5729    5737*   5738    5743
                        5751*   5752*   5753    5755    5762*   5763    5770*   5771*   5772    5774    5821*   5822    5827*
                        5828    5830    5835    5836    5844*   5845    5848    5856*   5857    5862    5870*   5871*   5872
                        5874    5881*   5882*   5883    5885    5934*   5938*   5939*   5940    5942    5950*   5951*   5952
                        5954    6002*   6006*   6007*   6008    6010    6018*   6019*   6020    6022    6070*   6073*   6076*
```

```
                    6079* 6080* 6081  6083  6127* 6130* 6133* 6136* 6137* 6138  6140  6219* 6221
                    6282* 6283  6285  6298* 6299* 6300  6302  6318* 6319  6320* 6321* 6322  6330
                    6334* 6335  6336* 6346* 6347  6348* 6408* 6409  6411  6412* 6424* 6425* 6426
                    6428  6452* 6453  6454* 6455  6464  6468* 6469  6470* 6471  6482* 6483  6484*
                    6485  6539* 6540  6542  6543* 6592* 6593* 6594  6596  6663* 6664* 6665  6667
                    6670  7149  7152  7160  7162  7165  7358* 7360  7362  7364  7366* 7367* 7369*
                    8529* 8535* 8538* 8552  8575
$TMP1   001204      1844# 2244* 2245  2275* 2276  2298* 2299  2321* 2322  2352* 2355* 2356  2386*
                    2389* 2390  2419* 2421* 2422  2454* 2456* 2457  2492* 2493* 2494* 2495  2528*
                    2529* 2533* 2535  2562* 2563  2584* 2585  2606* 2607  2630* 2632  2653* 2654
                    2675* 2676  2697* 2698  2724* 2725  2750* 2751  2776* 2777  2802* 2803  2828*
                    2829  2854* 2855  2880* 2881  2906* 2907  2933* 2934  2944* 2945  2972* 2973
                    2982* 2983  2993* 2994  3020* 3021  3030* 3031  3041* 3042  3068* 3069  3078*
                    3079  3089* 3090  3116* 3117  3126* 3127  3137* 3138  3164* 3165  3175* 3176
                    3203* 3204  3213* 3214  3224* 3225  3251* 3252  3261* 3262  3272* 3273  3299*
                    3300  3309* 3310  3320* 3321  3347* 3348  3357* 3358  3368* 3369  3395* 3396
                    3405* 3406  3416* 3417  3443* 3444  3453* 3454  3464* 3465  3491* 3492  3501*
                    3502  3512* 3513  3540* 3541  3550* 3551  3561* 3562  3588* 3589  3598* 3599
                    3609* 3610  3636* 3637  3646* 3647  3657* 3658  3684* 3685  3694* 3695  3705*
                    3706  3732* 3733  3742* 3743  3753* 3754  3780* 3781  3790* 3791  3801* 3802
                    3871* 3872  3883* 3884  3897* 3898  3909* 3910  3922* 3923  3942* 3943  3951*
                    3952  3960* 3961  4030* 4031  4042* 4043  4057* 4058  4071* 4072  4091* 4092
                    4100* 4101  4112* 4113  4120* 4121  4177* 4222* 4223  4280* 4391* 4421* 4422
                    4447* 4448  4466* 4467  4529* 4549* 4550  4560* 4561  4590* 4591  4601* 4602
                    4613* 4630* 4631  4641* 4642  4700* 4720* 4721  4731* 4732  4761* 4762  4772*
                    4773  4784* 4801* 4802  4812* 4813  4871* 4891* 4892  4902* 4903  4932* 4933
                    4943* 4944  4955* 4972* 4973  4983* 4984  5042* 5063* 5064  5074* 5075  5104*
                    5105  5118* 5133* 5134  5192* 5213* 5214  5224* 5225  5254* 5255  5268* 5283*
                    5284  5342* 5363* 5364  5374* 5375  5404* 5405  5418* 5433* 5434  5490* 5511*
                    5512  5529* 5544* 5545  5599* 5621* 5622  5639* 5654* 5655  5717* 5755* 5756
                    5774* 5775  5835* 5874* 5875  5885* 5886  5942* 5943  5954* 5955  6010* 6011
                    6022* 6023  6083* 6084  6140* 6141  6193* 6194  6198  6208* 6209  6213  6220*
                    6221  6285* 6286* 6302* 6303  6319* 6322  6335* 6337  6347* 6349  6411* 6428*
                    6429  6453* 6455  6469* 6483* 6542* 6596* 6597  6667* 6668  6670* 6671* 7149
                    7152  7155  7162  8530* 8536* 8539* 8553  8568
$TMP10  001222      1851# 6752* 7168
$TMP11  001224      1852#
$TMP2   001206      1845# 3872* 3873* 3884* 3885* 3898* 3899* 3910* 3911* 3923* 3924* 3943* 3944*
                    3952* 3953* 4031* 4032* 4043* 4044* 4058* 4059* 4072* 4073* 4092* 4093* 4101*
                    4102* 4113* 4114* 4178* 4179* 4223* 4224* 4281* 4282* 4392* 4393* 4467* 4468*
                    4530* 4531* 4561* 4562* 4602* 4603* 4631* 4632* 4642* 4643* 4701* 4702* 4732*
                    4733* 4773* 4774* 4802* 4803* 4813* 4814* 4872* 4873* 4903* 4904* 4944* 4945*
                    4973* 4974* 4984* 4985* 5043* 5044* 5075* 5076* 5134* 5135* 5193* 5194* 5225*
                    5226* 5284* 5285* 5343* 5344* 5375* 5376* 5434* 5435* 5491* 5492* 5545* 5546*
                    5600* 5601* 5655* 5656* 5718* 5719* 5756* 5757* 5775* 5776* 5836* 5837* 5875*
                    5876* 5886* 5887* 5943* 5944* 5955* 5956* 6011* 6012* 6023* 6024* 6084* 6085*
                    6141* 6142* 6198* 6199* 6213* 6214* 6303* 6304* 6429* 6430* 6597* 6598* 6668*
                    6669* 7155  7165  8552* 8556*
$TMP3   001210      1846# 6179* 6184  6220  6225* 8553* 8557*
$TMP4   001212      1847# 5741* 5766* 5860* 6180* 6202* 6557  6742  7165  7168  8558*
$TMP5   001214      1848# 6556  6740  7165  7168  8559*
$TMP6   001216      1849# 6719  6740  7168
$TMP7   001220      1850# 6742  7168
$TN  =  000106      1654# 1767# 2228  2230# 2246  2259  2261# 2277  2290  2292# 2300  2313  2315#
                    2323  2336  2338# 2357  2370  2372# 2391  2406  2408# 2429  2441  2443# 2464
                    2477  2479# 2482  2502  2515  2517# 2542  2554  2556# 2564  2576  2578# 2586
```

```
                    2598    2600#   2608    2622    2624#   2633    2645    2647#   2655    2667    2669#   2677    2689
                    2691#   2699    2714    2716#   2726    2740    2742#   2752    2766    2768#   2778    2792    2794#
                    2804    2818    2820#   2830    2844    2846#   2856    2870    2872#   2882    2896    2898#   2908
                    2921    2923#   2946    2963    2965#   2995    3011    3013#   3043    3059    3061#   3091    3107
                    3109#   3139    3152    3154#   3177    3194    3196#   3226    3242    3244#   3274    3290    3292#
                    3322    3338    3340#   3370    3386    3388#   3418    3434    3436#   3466    3482    3484#   3514
                    3531    3533#   3563    3579    3581#   3611    3627    3629#   3659    3675    3677#   3707    3723
                    3725#   3755    3771    3773#   3803    3852    3854#   3962    4010    4012#   4122    4148    4150#
                    4228    4251    4253#   4319    4350    4352#   4479    4482    4493    4495#   4653    4664    4666#
                    4824    4835    4837#   4995    5006    5008#   5145    5156    5158#   5295    5306    5308#   5445
                    5456    5458#   5554    5565    5567#   5664    5686    5688#   5784    5804    5806#   5894    5908
                    5910#   5961    5974    5976#   6029    6044    6046#   6090    6105    6107#   6147    6172    6174#
                    6229    6268    6270#   6351    6394    6396#   6487    6522    6524#   6619    6646    6648#   6764
                    6770    6772#
$TPB      001152    1828#   7830*   7841
$TPFLG    001157    1832#   7771    7841
$TPS      001150    1827#   7828    7841
$TRAP     050052    2106    8623#
$TRAP2    050074    8634#   8645
$TRP    = 000022    8638#   8647#   8648#   8649#   8650#   8651#   8652    8653#   8654    8655#   8656#   8657#   8658#
                    8659#   8660#   8661#   8662#   8663#   8664#   8665#
$TRPAD    050106    8628    8645#
$TSTNM    001102    1805#   2196*   2230*   2261*   2292*   2315*   2338*   2372*   2408*   2443*   2479*   2517*   2556*
                    2578*   2600*   2624*   2647*   2669*   2691*   2716*   2742*   2768*   2794*   2820*   2846*   2872*
                    2898*   2923*   2965*   3013*   3061*   3109*   3154*   3196*   3244*   3292*   3340*   3388*   3436*
                    3484*   3533*   3581*   3629*   3677*   3725*   3773*   3854*   4012*   4150*   4253*   4352*   4495*
                    4666*   4837*   5008*   5158*   5308*   5458*   5567*   5688*   5806*   5910*   5976*   6046*   6107*
                    6174*   6270*   6396*   6524*   6648*   6772*   6786*   7593    7645    7706    7737    7746*   7751
                    7754
$TTYIN    046124    7349    8188    8189    8201    8219    8233    8237#
$TYPBN=   ****** U  8651
$TYPDS    044502    7930#   8650
$TYPE     043762    7771#   8638    8646
$TYPEC    044132    7792    7799    7806    7811#   8136
$TYPEX    044252    7834    7836    7839#
$TYPOC    044300    7871#   8647
$TYPON    044314    7870    7873#   8649
$TYPOS    044254    7866#   8648
$XOFF   = 000023    7816    7841
$XON    = 000021    7823    7841    8022
$XTSTR    043610    7724#
$$GET4=   000000    6801#
$OFILL    044477    7867*   7871*   7881    7916#
$40CAT    000000    1786#   7603    7716
.       = 050154    1771#   1787    1791#   1802#   1858    2099    2111    2112    2147#   2168#   2190#   2200#   2214#
                    4381*   4519*   4690*   4861*   5032*   5182*   5332*   5480*   5589*   6809    6813    7148#   7186#
                    7228#   7236#   7284#   7291#   7300#   7320#   7325#   7342#   7383#   7645    7700#   7754    7841
                    7984#   7988    7992#   7993    7994#   8237#   8238    8245#   8299    8359    8521#   8525
.BEGIN    002004    2093#
.CRC      047354    8504#
.LRESE    047712    8582#   8662
.MASK     047274    2353    2387    8484#
.MONIT    002332    2150#   2163    2169    8032    8115
.MSTCL    047172    8461#   8659
.READ     047022    8425#   8663
.ROEND    001242    1979#
```

```
.ROMCL  047236            8472#   8660
.ROVAR  001240            1862#
.START  002004            1789    1795    2092#
.TICK   047736            8594#   8664
.TICKE  050050            8597    8604    8609    8614#
.WRITE  046674            8397#   8661
```

```
BUMP    1628#  8398  8400  8426  8475  8595
COMMEN  1766#
ENDCOM  1766#
ERROR   1660#  2247  2278  2301  2324  2358  2392  2424  2459  2497  2537  2565  2587  2609  2634
        2656   2678  2700  2727  2753  2779  2805  2831  2857  2883  2909  2936  2947  2975  2985
        2996   3023  3033  3044  3071  3081  3092  3119  3129  3140  3167  3178  3206  3216  3227
        3254   3264  3275  3302  3312  3323  3350  3360  3371  3398  3408  3419  3446  3456  3467
        3494   3504  3515  3543  3553  3564  3591  3601  3612  3639  3649  3660  3687  3697  3708
        3735   3745  3756  3783  3793  3804  3874  3886  3900  3912  3925  3945  3954  3963  4033
        4045   4060  4074  4094  4103  4115  4123  4180  4196  4206  4225  4283  4299  4309  4394
        4415   4424  4428  4436  4440  4450  4471  4532  4552  4563  4569  4579  4585  4593  4604
        4620   4633  4644  4703  4723  4734  4740  4750  4756  4764  4775  4791  4804  4815  4874
        4894   4905  4911  4921  4927  4935  4946  4962  4975  4986  5045  5066  5077  5083  5093
        5099   5107  5124  5136  5195  5216  5227  5233  5243  5249  5257  5274  5286  5345  5366
        5377   5383  5393  5399  5407  5424  5436  5493  5514  5520  5535  5602  5624  5630  5645
        5720   5733  5747  5758  5779  5838  5852  5866  5877  5890  5945  5957  6013  6025  6086
        6143   6200  6215  6223  6287  6305  6324  6339  6352  6413  6431  6457  6473  6488  6544
        6599   6605  6672  6749  6753
ESCAPE  1766#
GETPRI  1766#  8377
GETSWR  1766#  2136#
MULT    1766#
NEWTST  1766#  2228  2259  2290  2313  2336  2370  2406  2441  2477  2515  2554  2576  2598  2622
        2645   2667  2689  2714  2740  2766  2792  2818  2844  2870  2896  2921  2963  3011  3059
        3107   3152  3194  3242  3290  3338  3386  3434  3482  3531  3579  3627  3675  3723  3771
        3852   4010  4148  4251  4350  4493  4664  4835  5006  5156  5306  5456  5565  5686  5804
        5908   5974  6044  6105  6172  6268  6394  6522  6646  6770
NSET    1629#  3869  3920  4028  4069  4558  4599  4639  4729  4770  4810  4900  4941  4981  5072
        5222   5372  5542  5652  5952  6020  6594  6665
PARTAB  1625#  7187
POP     1766#  2243  2274  2297  2320  2351  2385  2418  2453  2491  2527  2561  2583  2605  2629
        2652   2674  2696  2722  2748  2774  2800  2826  2852  2878  2904  2932  2943  2970  2980
        2991   3018  3028  3039  3066  3076  3087  3114  3124  3135  3163  3174  3201  3211  3222
        3249   3259  3270  3297  3307  3318  3345  3355  3366  3393  3403  3414  3441  3451  3'62
        3489   3499  3510  3538  3548  3559  3586  3596  3607  3634  3644  3655  3682  3692  3703
        3730   3740  3751  3778  3788  3799  3867  3879  3893  3905  3918  3938  3959  4026  4038
        4053   4067  4087  4108  4119  4165  4172  4185  4192  4201  4209  4213  4218  4269  4275
        4288   4295  4304  4312  4316  4378  4386  4399  4420  4433  4442  4446  4455  4459  4462
        4516   4524  4537  4547  4556  4574  4589  4597  4609  4613  4626  4637  4687  4695  4708
        4718   4727  4745  4760  4768  4780  4784  4797  4808  4858  4866  4879  4889  4898  4916
        4931   4939  4951  4955  4968  4979  5029  5037  5051  5061  5070  5088  5103  5114  5118
        5129   5179  5187  5201  5211  5220  5238  5253  5264  5268  5279  5329  5337  5351  5361
        5370   5388  5403  5414  5418  5429  5477  5485  5499  5509  5525  5529  5540  5586  5594
        5608   5618  5635  5639  5650  5702  5709  5725  5737  5741  5751  5762  5766  5770  5821
        5827   5844  5856  5860  5870  5881  5934  5938  5950  6002  6006  6018  6070  6073  6076
        6079   6127  6130  6133  6136  6298  6424  6592  6663  7971  8288  8348  8438  8439  8454
        8599
PUSH    1766#  7930  8262  8315  8440  8442  8443  8457
REPORT  1644#  1766#
SCOPE   1661#  2229  2260  2291  2314  2337  2371  2407  2412  2442  2447  2478  2485  2516  2521
        2555   2577  2599  2623  2646  2668  2690  2715  2741  2767  2793  2819  2845  2871  2897
        2922   2927  2964  3012  3060  3108  3153  3158  3195  3243  3291  3339  3387  3435  3483
        3532   3580  3628  3676  3724  3772  3853  4011  4149  4252  4351  4494  4665  4836  5007
        5157   5307  5457  5566  5687  5805  5909  5975  6045  6106  6173  6269  6395  6523  6647
        6771   6785
SET     1629#  3869  3881  3895  3907  3920  3940  3948  4028  4040  4055  4069  4089  4097  4110
```

```
        4220    4464    4558    4599    4628    4639    4729    4770    4799    4810    4900    4941    4970    4981    5072
        5131    5222    5281    5372    5431    5542    5652    5753    5772    5872    5883    5940    5952    6008    6020
        6081    6138    6300    6426    6594    6605
SETPRI  1766#   8166
SETTRA  8638#   8647    8648    8649    8650    8652    8654    8655    8656    8657    8658    8659    8660    8661    8662
        8663    8664
SETUP   1766#   2094
SKIP    1766#   2246    2277    2300    2323    2357    2391    2429    2464    2482    2502    2542    2564    2586    2608
        2633    2655    2677    2699    2726    2752    2778    2804    2830    2856    2882    2908    2946    2995    3043
        3091    3139    3177    3226    3274    3322    3370    3418    3466    3514    3563    3611    3659    3707    3755
        3803    3962    4122    4228    4319    4479    4481    4652    4823    4994    5144    5294    5444    5553    5663
        5783    5893    5960    6028    6089    6146    6228    6350    6486    6618    6763
SLASH   1766#
SPACE   1766#
STARS   1766#   1798    1858    2228    2259    2290    2313    2336    2370    2406    2441    2477    2515    2554    2576
        2598    2622    2645    2667    2689    2714    2740    2766    2792    2818    2844    2870    2896    2921    2963
        3011    3059    3107    3152    3194    3242    3290    3338    3386    3434    3482    3531    3579    3627    3675
        3723    3771    3852    4010    4148    4251    4350    4493    4664    4835    5006    5156    5306    5456    5565
        5686    5804    5908    5974    6044    6105    6148    6172    6232    6268    6365    6394    6493    6522    6621
        6646    6770    6778    7577    7647    7703    7756    7843    7920    7987    8069    8084    8155    8179    8248
        8301    8361    8588    8592    8617
SWRSU   1766#   2113#
TEST    1628#   3866    3878    3892    3904    3917    3937    4025    4037    4052    4066    4086    4107    4217    4461
        4555    4596    4625    4636    4726    4767    4796    4807    4897    4938    4967    4978    5069    5128    5219
        5278    5369    5428    5539    5649    5749    5769    5868    5880    5937    5948    6005    6016    6078    6135
        6297    6423    6591    6662
TRMTRP  8638#
TYPBIN  1766#
TYPDEC  1766#   2209    6796    7687
TYPNAM  1766#   2129
TYPNUM  1766#
TYPOCS  1766#   2201
TYPOCT  1766#   7609    7614    7659    7684    8097    8568    8575
TYPTXT  1766#   2152    2164    2186    2197    2205    2211    7224    7233    7281    7287    7297    7310    7317    7322
        7329    7339    7379    7405    8518    8564    8571
$BEGIN  1628#   2091    2216#
$CRC    1628#   8499
$CRCNE  1637#   6493
$CRCRE  1637#   6365
$CRCTE  1635#   6232
$CRCTG  1640#   6621
$ENDT   1626#   6765
$ERM1   1628#   6813    7187#
$ERTBL  1625#   2005    2091#
$EXTRF  1635#   6148
$FLOAT  1632#   2411    2446    2484    2520
$GETKM  1626#   2178
$GETLI  1626#   2193
$LB01   1633#   3807    3965
$LB02   1633#   4126    4229
$LB03   1633#   4321
$LB04   1633#   4484    4655    4826    4997    5147    5297    5447    5556
$LB05   1633#   5667    5785
$LB06   1633#   5896    5962
$LB07   1633#   6031
$LB08   1633#   6092
```

```
$LINE1  1632#  2395  2430  2503
$LINE2  1633#  2912  3143
$LIN1A  1633#  2465
$LU1    1632#  2218  2249  2326  2360
$LU1A   1632#  2280  2303
$MASK   1628#  8479  8499#
$MSTCL  1626#  8461  8472#
$PARAM  1625#  7445
$READ   1628#  8423  8461#
$ROMCL  1628#  8472  8479#
$ROVAR  1625#  1859  1990#
$SEC0   1633#  2544  2567  2589  2611  2636  2658  2680
$SEC2   1633#  2950  3181  3518
$SEC3   1633#  2703
$TICK   1629#  8588  8615#
$TSTN   1625#  2226  2257  2288  2311  2334  2368  2404  2439  2475  2513  2552  2574  2596  2620
        2643   2665  2687  2712  2738  2764  2790  2816  2842  2868  2894  2919  2961  3009  3057
        3105   3150  3192  3240  3288  3336  3384  3432  3480  3529  3577  3625  3673  3721  3769
        3850   4008  4146  4249  4348  4491  4662  4833  5004  5154  5304  5454  5563  5684  5802
        5906   5972  6042  6103  6170  6266  6392  6520  6644  6768
$WRITE  1628#  8395  8423#
$XZ     1628#  2218  2224  2249  2255  2280  2286  2303  2309  2326  2332  2360  2366  2395  2402
        2430   2437  2465  2473  2503  2511  2544  2550  2567  2572  2589  2594  2611  2618  2636
        2641   2658  2663  2680  2685  2703  2710  2729  2736  2755  2762  2781  2788  2807  2814
        2833   2840  2859  2866  2885  2892  2912  2917  2950  2959  2998  3007  3046  3055  3094
        3103   3143  3148  3181  3190  3229  3238  3277  3286  3325  3334  3373  3382  3421  3430
        3469   3478  3518  3527  3566  3575  3614  3623  3662  3671  3710  3719  3758  3767  3807
        3848   3965  4006  4126  4144  4229  4247  4321  4346  4484  4489  4655  4660  4826  4831
        4997   5002  5147  5152  5297  5302  5447  5452  5556  5561  5667  5682  5785  5800  5896
        5904   5962  5970  6031  6040  6092  6101
$$CMRE  1796#  1835  1836  1837  1838  1839  1840  1841  1842
$$CMTM  1796#  1843  1844  1845  1846  1847  1848  1849  1850  1851  1852
$$ESCA  1766#
$$LRST  1628#  8579  8588#
$$NEWT  1766#  2228  2259  2290  2313  2336  2370  2406  2441  2477  2515  2554  2576  2598  2622
        2645   2667  2689  2714  2740  2766  2792  2818  2844  2870  2896  2921  2963  3011  3059
        3107   3152  3194  3242  3290  3338  3386  3434  3482  3531  3579  3627  3675  3723  3771
        3852   4010  4148  4251  4350  4493  4664  4835  5006  5156  5306  5456  5565  5686  5804
        5908   5974  6044  6105  6172  6268  6394  6522  6646  6770
$$SET   8638#  8647  8648  8649  8650  8652  8654  8655  8656  8657  8658  8659  8660  8661  8662
        8663   8664
$$SKIP  1766#  2246  2277  2300  2323  2357  2391  2429  2464  2482  2502  2542  2564  2586  2608
        2633   2655  2677  2699  2726  2752  2778  2804  2830  2856  2882  2908  2946  2995  3043
        3091   3139  3177  3226  3274  3322  3370  3418  3466  3514  3563  3611  3659  3707  3755
        3803   3962  4122  4228  4319  4479  4482  4653  4824  4995  5145  5295  5445  5554  5664
        5784   5894  5961  6029  6090  6147  6229  6351  6487  6619  6764
.EQUAT  1644#  1656
.HEADE  1644#
.KT11   1644#
.SETUP  1644#  2091
.$CMTA  1644#  1796
.$EOP   1644#  6776
.$ERRO  1644#  7575
.$ERRT  1644#  7645
.$GETK  1625#  7515
.$GETL  1626#  7550
```

```
.$RDDE    1644#    8299
.$RDOC    1644#    8246
.$READ    1644#    7985
.$SCOP    1644#    7701
.$SIZE    1644#    8359
.$TRAP    1644#    8615
.$TYPD    1644#    7918
.$TYPE    1644#    7754
.$TYPO    1644#    7841
.$40CA    1644#    1769


. ABS.  050154     000


 ERRORS DETECTED:  0

 CZKMDA,CZKMDA/CRF/SOL/NL:TOC=CZKMDA
 RUN-TIME: 148 148 14 SECONDS
 RUN-TIME RATIO: 1063/313=3.3
 CORE USED:  45K  (89 PAGES)
```