

DMC11

LOW SPD JUMP FREE/RUN
MD-11-DZDMG-B

EP-DZDMG-B-DL-A

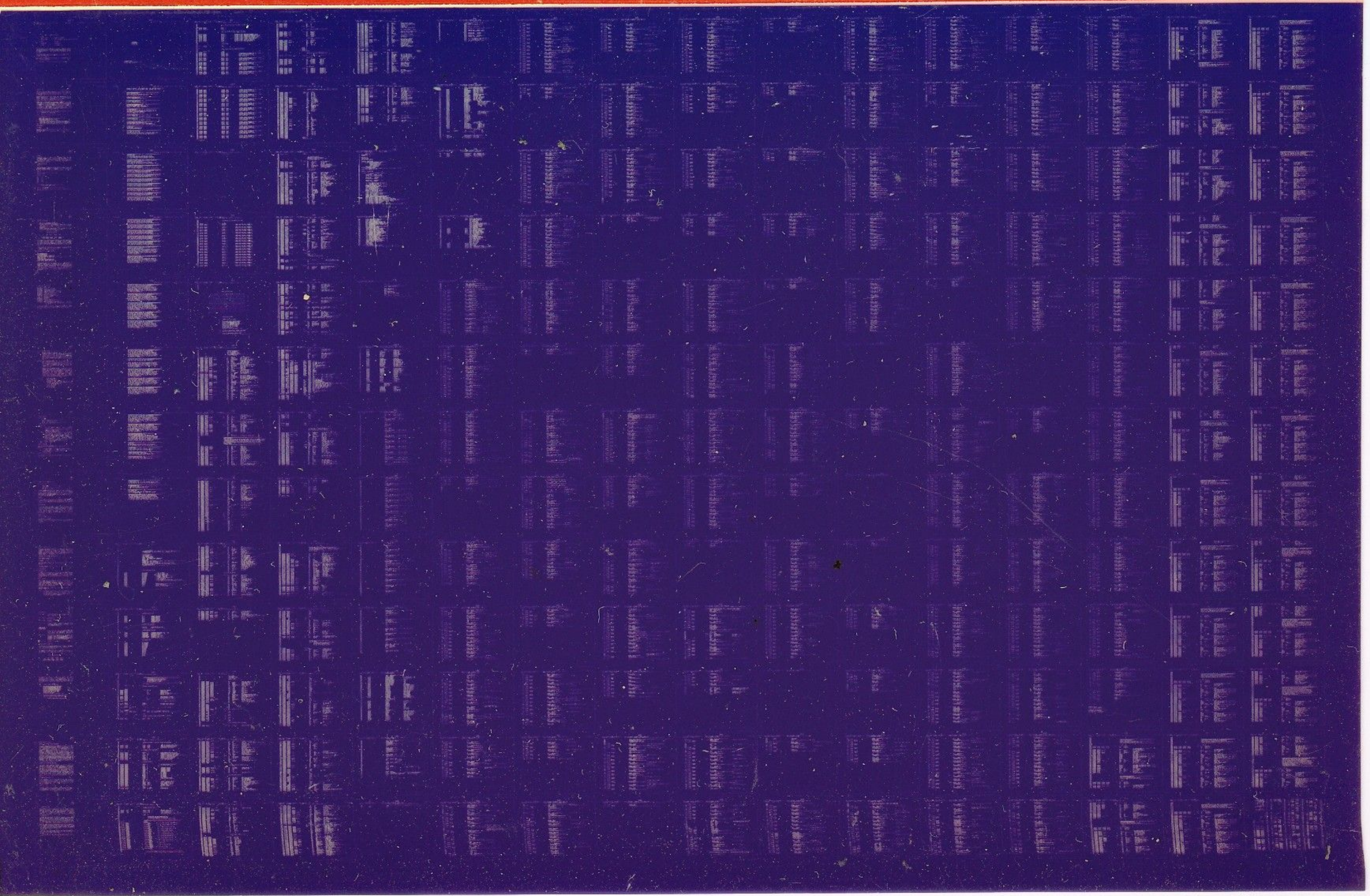
COPYRIGHT © 1977

FICHE 1 OF 2

MAR 1977

digital

MADE IN USA



DMC11

LOW SPD JUMP FREE/RUN
MD-11-DZDMG-B

EP-DZDMG-B-DL-A

MAR 1977

COPYRIGHT © 1977

digital

FICHE 2 OF 2

MADE IN USA

This microfiche card contains a grid of frames, each displaying a different set of data. The data is organized into columns and rows, with some frames containing headers and footers. The frames are arranged in a grid that is approximately 12 frames wide and 15 frames high. The data appears to be a list of records or a table of values, with some frames containing more detailed information than others. The frames are arranged in a grid that is approximately 12 frames wide and 15 frames high. The data appears to be a list of records or a table of values, with some frames containing more detailed information than others.

11

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZDMG-B-D
PRODUCT NAME: DMC11 LOW SPEED JUMP AND FREE RUNNING TESTS
DATE: DEC 1976
MAINTAINER: DIAGNOSTICS
AUTHOR: FAY BASHAW

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

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

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

Copyright (C) 1976 by Digital Equipment Corporation

1. ABSTRACT

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

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

DZDMG tests the DMC11-AR micro-processor (M8200-YA) with low speed crom, or the KMC11 micro-processor (M8204). It performs jump tests on the micro-processor, verifies the control ROM of the M8200-YA, and tests the CRAM and other unique functions of the M8204. If a DMC11-AR (M8200-YA) and line unit (M8201) are present, free-running tests are performed. These tests are skipped if a KMC (M8204) or no line-unit is present. The test test is with a line-unit installed. DZDMG can be used as a Heat Test Diagnostic by Manufacturing.

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

NOTE: Additional diagnostics may be added in the future.

The four diagnostics are:

1. DZDMC [REV] Basic W/R and Micro-processor tests
2. DZDME [REV] DDCMP Line unit tests
3. DZDMF [REV] BITSTUFF Line unit tests
4. DZDMG [REV] Low speed jump and Free-running tests (Heat test tape) NOTE: DZDMG IS RUN ONLY ON A DMC11-AR (M8200-YA).
DZDMH [REV] High speed jump and Free-running tests (Heat test tape) NOTE: DZDMH IS RUN ONLY ON A DMC11-AL (M8200-YB).

2. REQUIREMENTS

2.1 EQUIPMENT

Any PDP11 family CPU (except an LSI-11) with minimum 8k memory
ASR 33 (or equivalent)
DMC11-AR (M8200-YA) or or KMC11-A (M8204) with a DMC11-DA or a
DMC11-FA

2.2 STORAGE

Program will use all 8K of memory except where ABL and BOOTSTRAP LOADER reside. Locations 1500 thru 1640; contain the "STATUS TABLE" information which is generated at start of diagnostics by manual input (questions) or automatically (auto-sizing). This area is an overlay area and should not be altered by the operator.

3. LOADING PROCEEDURE

3.1 METHOD

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

ABSOLUTE LOADER starting address *500

MEMORY * SIZE

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

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

4. STARTING PROCEEDJRE

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

MAP OF DMC11 STATUS

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

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

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

HOW MANY DMC11'S TO BE TESTED?1

01
 CSR ADDRESS?160010
 VECTOR ADDRESS?310
 BR PRIORITY LEVEL? (4,5,6,7)?5
 DOES MICRO-PROCESSOR HAVE CRAM? (Y OR N)N
 WHICH LINE UNIT? IF NONE TYPE "N". IF M8201 TYPE "1". IF M8202 TYPE "2"
 IS THE LOCK BACK CONNECTOR ON?Y
 SWITCH PAC#1 (DDCMP LINE#)?377
 SWITCH PAC#2 (BMB73 BOOT ADD)?377

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

4.1 CONTROL SWITCH SETTINGS

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

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

4.1.2 SWITCH REGISTER OPTIONS (at start up)

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

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

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

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

4.1.3 DYNAMIC SWITCHES

ERROR SWITCHES

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

SCOPE SWITCHES

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

4.2 STARTING ADDRESS

Starting address is at 00020C there are no other starting addresses for the DMC11 diagnostics. (See Section 4.0)

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

E. OPERATING PROCEDURE

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

5.2 PROGRAM AND/OR OPERATOR ACTION

The typical approach should be

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

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

6. ERRORS

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

6.2 ERROR RECOVERY

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

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

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

7.2 OPERATING RESTRICTIONS

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

7.3 HARDWARE CONFIGURATION RESTRICTIONS

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

KMC(MB204)- Jumper W1 must be in.

LINE UNIT(MB201)- Jumpers W1, W2, and W4 must be IN. Jumpers W3 and W5 must be OUT. SW8 of E26 must be in the ON POSITION.

LINE UNIT (MB202)- Jumper W1 must be in. SW8 of E26 must be in the OFF position.

8. MISCELLANEOUS

8.1 EXECUTION TIME

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

8.2 PASS COMPLETE

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

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

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

E.4 KEY LOCATIONS

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

DMCROO-DMCR17
DMST00-DMST17
(1500)-(1640)

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

- DMACTV (1306) Each bit set in this location indicates that the associated DMC11 will be tested in turn. EXAMPLE: (DMACTV) 1276/00000000000011111 means that DMC11 no. 00,01,02,03,04 will be tested. EXAMPLE: (DMACTV) 1276/00000000000010001 Means that DMC11 no. 00,04 will be tested.

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

E.4A 'STATUS TABLE' (1500-1640)

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

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

MAP OF DMC11 STATUS

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

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

CSR: Contains DMC11 CSR address

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

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

STAT3: BIT0=1 PERFORM FREE RUNNING TESTS ON KMC
(must be set manually. SEE TEST 5C)

9.5 METHOD OF AUTO SIZING

9.5.1 FINDING THE CONTROL STATUS REGISTER.

The auto-sizing routine finds a DMC11 as follows: It starts at address 160000 and tests all address in increments of 10 up to and including address 167760. If the address does not time out, the following is done, the first CROM address is written to a 125252 then it is read back. If it contains a -1 or 125252 or 63220 a DMC11 or KMC11 has been found, if not, the address is updated by 10 and the search continues. A -1 indicates a DMC11 with no CROM, a 125252 indicates a KMC11 with CROM and a 63220 indicates a DMC11 with the DD2MP CROM. Further tests are performed at this point to determine which line unit, if any, is installed, if a loop-back connector is installed and various switch settings on the line unit. THIS IS WHY THE STATUS TABLE MUST BE VERIFIED BY THE USER AND IF ANY OF THE INFORMATION DOES NOT AGREE WITH THE HARDWARE THE DIAGNOSTIC MUST BE RESTARTED AND THE QUESTIONS MUST BE ANSWERED. All DMC11's in the system will be found by the auto-sizer. If it does not find a DMC11 the diagnostic must be restarted and the questions answered.

9.5.2 FINDING THE VECTOR AND BR LEVEL

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

9.6 SOFTWARE SWITCH REGISTER

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

Control:

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

SWR=XXXXXX NEW?

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

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

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

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

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

DOCUMENT

DZDMGB SEQ

COPYRIGHT 1977
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

6 MAINDEC-11-DZDMG-B DMC11 REMOTE CROM, JUMP, AND FREE RUNNING TESTS
COPYRIGHT 1976, DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754

1626 ***** TEST 1 *****
TEST OF BR RIGHT SHIFT
VERIFY THAT A DEST OF BR RSH (011) OF A MICRL INSTRUCTION
SHIFTS THE RESULTING BR DATA RIGHT ONCE.

1666 ***** TEST 2 *****
IOP CROM WRITE/READ TEST
FLOAT A 1 THROUGH EACH CROM LOCATION

1700 ***** TEST 3 *****
IOP CROM WRITE/READ TEST
FLOAT A 0 THROUGH EACH CROM LOCATION

1737 ***** TEST 4 *****
IOP CROM DUAL ADDRESSING TEST
WRITE EACH ADDRESS INTO ITSELF, READ EACH
ADDRESS TO VERIFY CORRECT ADDRESSING

1783 ***** TEST 5 *****
IOP CROM READ TEST
THIS TEST WRITES THE CROM WITH THE CROM MICRO-CODE MAP
THEN READS IT BACK AND COMPARES EACH ADDRESS WITH THE
DUPLICATE OF THE CROM MICRO-CODE.

1820 ***** TEST 6 *****
IOP MAIN MEMORY TEST
FLOAT A 1 THROUGH ALL MAIN MEMORY LOCATIONS

1866 ***** TEST 7 *****
IOP MAIN MEMORY TEST
FLOAT A 0 THROUGH ALL MAIN MEMORY LOCATIONS

1914 ***** TEST 10 *****
IOP MAIN MEMORY DUAL ADDRESSING TEST
LOAD EACH MEMORY LOCATION WITH ITS OWN ADDRESS
READ BACK EACH LOCATION TO VERIFY CORRECT ADDRESSING

1982 ***** TEST 11 *****
IOP MAR TEST
PERFORM DUAL ADDRESSING TEST
USING MAR AUTO-INC FEATURE

2022 ***** TEST 12 *****
IOP (CROM) ODT BITS TEST
LOAD MAR WITH A 0 INC MAR UNTIL IT OVERFLOWS (2000 TIMES)
VERIFY THAT IBUS* 10 BITS IS SET ONLY WHEN MAR BIT 8 IS A ONE
AND THAT IBUS* 10 BIT6 IS SET ON MAR OVERFLOW(2000)

2083 ***** TEST 13 *****
CROM READ TEST
THIS TEST READS EACH ROM LOCATION AND COMPARES
IT TO A SOFTWARE DUPLICATE OF THE CROM. THIS TEST
ALSO TESTS THE JUMP(I) MICRO-PROCESSOR INSTRUCTION.

2132 ***** TEST 14 *****
CROM TEST OF JUMP(I) NEVER MICRO-PROCESSOR INSTRUCTION.
PERFORM THE JUMP INSTRUCTION
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC (IT SHOULD INCREMENT BY ONE).

2189 ***** TEST 15 *****
CROM TEST OF JUMP(I) ALWAYS MICRO-PROCESSOR INSTRUCTION.
PERFORM THE JUMP INSTRUCTION
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2242 ***** TEST 16 *****
CROM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
SET THE C BIT, PERFORM THE JUMP INSTRUCTION.
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2299 ***** TEST 17 *****
CROM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
SET THE Z BIT, PERFORM THE JUMP INSTRUCTION.
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2354 ***** TEST 20 *****
CROM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
SET THE BRO BIT, PERFORM THE JUMP INSTRUCTION.
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2410 ***** TEST 21 *****
CROM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR1 BIT, PERFORM THE JUMP INSTRUCTION.
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2466 ***** TEST 22 *****
CROM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR4 BIT, PERFORM THE JUMP INSTRUCTION.
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2522 ***** TEST 23 *****
CROM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR7 BIT, PERFORM THE JUMP INSTRUCTION.
VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

2578 ***** TEST 24 *****
CROM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE C BIT, PERFORM THE JUMP INSTRUCTION.
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC (IT SHOULD INCREMENT BY ONE).

2635 ***** TEST 25 *****
CROM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE Z BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

2692 ***** TEST 26 *****
CROM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE BRO BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

2749 ***** TEST 27 *****
CROM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE BR1 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

2806 ***** TEST 30 *****
CROM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE BR4 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

2863 ***** TEST 31 *****
CROM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE BR7 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THAT THE JUMP DID NOT OCCUR BY READING
THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

2920 ***** TEST 32 *****
CROM TEST OF JUMP(I) NEVER MICRO-PROCESSOR INSTRUCTION.
PERFORM THE JUMP INSTRUCTION
VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CROM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
THE CROM PC IS CORRECT, IF THE CROM PC IS NOT RIGHT,
THEN PORT4 CONTAINS A 37

2982 ***** TEST 33 *****
CROM TEST OF JUMP(I) ALWAYS MICRO-PROCESSOR INSTRUCTION.
PERFORM THE JUMP INSTRUCTION
VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CROM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
THEN PORT4 WILL CONTAIN A 37

3041 ***** TEST 34 *****
CRAM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
SET THE C BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
THEN PORT4 WILL CONTAIN A 37

3103 ***** TEST 35 *****
CRAM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
SET THE Z BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
THEN PORT4 WILL CONTAIN A 37

3165 ***** TEST 36 *****
CRAM TEST OF JUMP(I) ON BR0 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR0 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
THEN PORT4 WILL CONTAIN A 37

3227 ***** TEST 37 *****
CRAM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR1 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
THEN PORT4 WILL CONTAIN A 37

3289 ***** TEST 40 *****
CRAM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
SET THE BR4 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
THEN PORT4 WILL CONTAIN A 37

3351 ***** TEST 41 *****
 CRAM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
 SET THE BR7 BIT, PERFORM THE JUMP INSTRUCTION,
 VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
 IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
 BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
 THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
 THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
 THEN PORT4 WILL CONTAIN A 37

3413 ***** TEST 42 *****
 CRAM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
 CLEAR THE C BIT, PERFORM THE JUMP INSTRUCTION,

3416 VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
 IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
 BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
 THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
 THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
 THEN PORT4 CONTAINS A 37

3475 ***** TEST 43 *****
 CRAM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
 CLEAR THE Z BIT, PERFORM THE JUMP INSTRUCTION,
 VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
 IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
 BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
 THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
 THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
 THEN PORT4 CONTAINS A 37

3537 ***** TEST 44 *****
 CRAM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
 CLEAR THE BRO BIT, PERFORM THE JUMP INSTRUCTION,
 VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
 IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
 BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
 THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
 THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
 THEN PORT4 CONTAINS A 37

3599 ***** TEST 45 *****
 CRAM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
 CLEAR THE BR1 BIT, PERFORM THE JUMP INSTRUCTION,
 VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
 IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
 BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
 THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
 THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
 THEN PORT4 CONTAINS A 37

3661 ***** TEST 46 *****
CRAM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE BR4 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
THEN PORT4 CONTAINS A 37

3723 ***** TEST 47 *****
CRAM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
CLEAR THE BR7 BIT, PERFORM THE JUMP INSTRUCTION,
VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
THEN PORT4 CONTAINS A 37

3795 ***** TEST 50 *****
FREE RUNNING FLAG MODE DATA TEST
TRANSMIT A MESSAGE AND VERIFY THE RECEIVED DATA
IF NO TURNAROUND CONNECTOR IS ON LINE UNIT LOOP IS SET.
ALL FOLLOWING TESTS ARE FREE RUNNING AND ARE PERFORMED
ONLY ON DMC'S WITH LINE UNITS. IF YOU WISH TO PERFORM
THESE FREE RUNNING TESTS ON A KMC (NORMALLY THE FREE RUNNING TESTS
WILL FAIL ON A KMC, THE TIMER IS TOO FAST) THEN YOU MUST
MANUALLY SET BIT0 OF STAT3 IN THE STATUS MAP.

3960 ***** TEST 51 *****
OVERUN TEST
IN FREE RUNNING MODE SEND MESSAGE WITH NO RECEIVE
BUFFER AVAILABLE, VERIFY THAT AN OVERRUN ERROR OCCURS

4018 ***** TEST 52 *****
LOST DATA TEST
IN FREE RUNNING MODE SEND A MESSAGE LONGER THAN THE RECEIVE
BUFFER, VERIFY THAT A LOST DATA ERROR OCCURS.

4065 ***** TEST 53 *****
TRANSMIT NON-EXISTENT MEMORY TEST
IN FREE RUNNING MODE, LOAD A TRANSMIT BA THAT WILL TIME OUT
VERIFY THAT A NON-EXISTENT MEMORY ERROR OCCURS

4111 ***** TEST 54 *****
RECEIVE NON-EXISTENT MEMORY TEST
IN FREE RUNNING MODE, LOAD A RECEIVE BA THAT WILL TIME OUT
VERIFY THAT A NON-EXISTENT MEMORY ERROR OCCURS

- 4160 ***** TEST 55 *****
 PROCESSOR ERROR TEST
 IN FREE RUNNING MODE, DO A BASE TRANSFER REQUEST AFTER A
 BASE HAS BEEN SET UP, VERIFY THAT A PROCESSOR ERROR OCCURS.

- 4204 ***** TEST 56 *****
 PROCESSOR ERROR TEST
 IN FREE RUNNING MODE DO A RQI WITH AN ILLEGAL IO CODE
 VERIFY THAT A PROCESSOR ERROR OCCURS

- 4248 ***** TEST 57 *****
 HALF DUPLEX TEST
 IN FREE RUNNING MODE, SET HALF DUPLEX AND L U LOOP
 SEND A MESSAGE AND VERIFY THAT THERE ARE NO DONES

- 4288 ***** TEST 60 *****
 FREE RUNNING DATA TEST (INTERRUPT DRIVEN EXERCISE^o)
 THIS TEST REPEATEDLY QUEUES UP 7 RECEIVE BUFFERS AND
 7 TRANSMIT BUFFERS AND CHECKS DATA WHEN ALL 7 BUFFERS
 ARE RECEIVED. TRANSMIT COUNTS RANGE FROM 1 TO 104, ALSO
 ODD AND EVEN TRANSMIT AND RECEIVE BA'S ARE USED. DATA
 IS A BINARY COUNT PATTERN. THE RESUME FUNCTION IS CHECKED IN THIS TEST

:*MAINDEC-11-DZDMG-B DMC11 REMOTE CROM, JUMP, AND FREE RUNNING TESTS
:*COPYRIGHT 1976, DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754
:*

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

:SWITCH REGISTER OPTIONS
:-----

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

SW15=100000 : =1, HALT ON ERROR
SW14=40000 : =1, LOOP ON CURRENT TEST
SW13=20000 : =1, INHIBIT ERROR TYPEOUT
SW12=10000 : =1, DELETE TYPEOUT/BELL ON ERROR.
SW11=4000 : =1, INHIBIT ITERATIONS
SW10=2000 : =1, ESCAPE TO NEXT TEST ON ERROR
SW09=1000 : =1, LOOP WITH CURRENT DATA
SW08=400 : =1, LOOP ON ERROR
SW07=200 : =1, USE CURRENT DMC11 PARAMETERS, =0, AUTOSIZE DMC11
SW06=100 : =1, HALT BEFORE CLOCKING MICRO-PROCESSOR INSTRUCTION
SW05=40
SW04=20
SW03=10 : RESELECT DMC11'S TO BE TESTED (ACTIVE)
SW02=4 : LOCK ON TEST SELECT
SW01=2 : RESTART PROGRAM AT SELECTED TEST
SW00=1 : INPUT DMC11 PARAMETERS

0200: MACY11: 07-MAR-77 15:52 PAGE 2
0200: P:1 07-MAR-77 11:21 INTRODUCTION TO DMC11 DIAGNOSTIC

TRAPCATCHER FOR UNEXPECTED INTERRUPTS

98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143

00000C

: TRAPCATCHER FOR ILLEGAL INTERRUPTS
: THE STANDARD "TRAP CATCHER" IS PLACED
: BETWEEN ADDRESS 0 TO ADDRESS 776.
: IT LOOKS LIKE "PC+2 HALT".

.=0
: STANDARD INTERRUPT VECTORS

.=24
 .PFAIL : POWER FAIL HANDLER
 340 : SERVICE AT LEVEL 7
 .HLT : ERROR HANDLER
 340 : SERVICE AT LEVEL 7
 .TRPSRV : GENERAL HANDLER DISPATCH SERVICE
 340 : SERVICE AT LEVEL 7

.=40
 0 : SAVE FOR ACT-11 OR XXDP
 0 : RETURN ADDRESS IF UNDER ACT-11 OR XXDP
 0 : SAVE FOR ACT-11 OR XXDP
 \$ENDAD : FOR USE WITH ACT-11 OR XXDP

.=52
 0 : ACT-11 PROGRAM CHARACTERISTICS

.=174
DISPREG: 0 : SOFTWARE DISPLAY REGISTER
SWREG: 0 : SOFTWARE SWITCH REGISTER

.=200
 JMP .START : GO TO START OF PROGRAM

.=1000
MTITLE: .ASCII <377><12>/MAINDEC-11-DZDMG-B/<377>
 .ASCII /DMC11 REMOTE CROM. JUMP, AND FREE RUNNING TESTS/<377>

.=1200
: INDIRECT POINTERS TO SWITCH REGISTER AND LIGHT DISPLAY

DISPLAY: 177570
SWR: 177570

000024 005240
000026 000340
000030 004656
000032 000340
000034 004624
000036 000240
000040 000000
000042 000000
000044 000000
000046 003432
000052 000000
000174 000000
000176 000000
000200 000200
001000 005377 040515 047111
001025 104 041515 030461
001200 177570
001202 177570

PROGRAM PARAMETERS, VARIABLES, AND TRAP CALLS.

```

144
145
146
147
148 001204 177560      TKCSR: 177560      ;TE ETYPE KEYBOARD CONTROL REGISTER
149 001206 177562      TKDBR: 177562      ;TE ETYPE KEYBOARD DATA BUFFER
150 001210 177564      TPCSR: 177564      ;TELEPRINTER CONTROL REGISTER
151 001212 177566      TPDBR: 177566      ;TELEPRINTER DATA BUFFER
152
153
154
155
156 001214 000000      RETURN: 0          ;SCOPE ADDRESS FOR LOOP ON TEST
157 001216 000000      NEXT: 0            ;ADDRESS OF NEXT TEST TO BE EXECUTED
158 001220 000000      LOCK: 0            ;ADDRESS FOR LOCK ON CURRENT DATA
159 001222 000003      ICOUNT: 3          ;NUMBER OF ITERATIONS THAT CURRENT TEST WILL BE
160 001224 000000      LPCNT: 0           ;NUMBER OF ITERATIONS COMPLETED
161 001226 000000      TSTNO: 0           ;NUMBER OF TEST IN PROGRESS
162 001230 000000      PASCNT: 0          ;NUMBER OF PASSES COMPLETED
163 001232 000000      ERRCNT: 0          ;TOTAL NUMBER OF ERRORS
164 001234 000000      LSTERR: 0          ;PC OF LAST ERROR CALL
165
166
167
168
169 001236 000000      STRTSW: 0          ;SWITCHES AT START OF PROGRAM
170 001240 000000      STAT: 0            ;DM STATUS WORD STORAGE
171 001242 000000      CLKX: 0            ;
172 001244 000000      MASKX: 0            ;
173 001246 000000      TEMP1: 0            ;TEMPORARY STORAGE
174 001250 000000      TEMP2: 0            ;TEMPORARY STORAGE
175 001252 000000      TEMP3: 0            ;TEMPORARY STORAGE
176 001254 000000      TEMP4: 0            ;TEMPORARY STORAGE
177 001256 000000      TEMP5: 0            ;TEMPORARY STORAGE
178 001260 000000      SAVR0: 0            ;R0 STORAGE
179 001262 000000      SAVR1: 0            ;R1 STORAGE
180 001264 000000      SAVR2: 0            ;R2 STORAGE
181 001266 000000      SAVR3: 0            ;R3 STORAGE
182 001270 000000      SAVR4: 0            ;R4 STORAGE
183 001272 000000      SAVR5: 0            ;R5 STORAGE
184 001274 000000      SAVSP: 0            ;STACK POINTER STORAGE
185 001276 000000      SAVPC: 0            ;PROGRAM COUNTER STORAGE
186 001300 000000      ZERO: 0            ;
187 001302 000001      ONE: 1              ;
188 001304 000000      MEMLIM: 0           ;HIGHEST LOCATION FOR NPR'S
189 001306 000001      DMACTV: .BLKW 1     ;DMC11'S SELECTED ACTIVE.
190 001310 000001      DMNUM: .BLKW 1      ;OCTAL NUMBER OF DMC11'S.
191 001312 000001      SAVACT: .BLKW 1     ;ORIGINAL ACTV DEVICES
192 001314 000001      SAVNUM: .BLKW 1     ;WORKABLE NUMBER
193 001316 000000      RUN: 0              ;POINTER TO RUNNING DEVICE.
194
195 001320 001472      .EVEN
196 001322 00167E      CREAM: DM.MAP-6    ;TABLE POINTER.
                          MILK: CNT.MAP-4    ;TABLE POINTER
    
```

PROGRAM PARAMETERS, VARIABLES, AND TRAP CALLS.

197
198
199
200
201 001324 000
202 001325 000
203 001326 000
204 001327 000
205
206
207
208
209
210
211
212
213
214 001330
215 001330 104400
216 001330 003506
217 104401
218 001332 003644
219 104402
220 001334 003674
221 104403
222 001336 003756
223 104404
224 00.340 004062
225 104405
226 001342 004102
227 104406
228 001344 004302
229 104407
230 001346 004342
231 104410
232 001350 004374
233 104411
234 001352 004400
235 104412
236 001354 005370
237 104413
238 001356 005340
239 104414
240 001360 005406
241 104415
242 001362 005454
243 104416
244 001364 005520
245
246
247

PROGRAM CONTROL FLAGS

INIFLG: .BYTE 0 ;PROGRAM INITIALIZATION FLAG
ERRFLG: .BYTE 0 ;ERROR OCCURED FLAG
LOKFLG: .BYTE 0 ;LOCK ON CURRENT TEST FLAG
QV.FLG: .BYTE 0 ;QUICK VERIFY FLAG.
;ON FIRST PASS OF EACH DMC11 ITERATIONS WILL BE
.EVEN

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

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


```

248
249 ;DMC11 CONTROL INDICATORS FOR CURRENT DMC11 UNDER TEST
250 -----
251
252 001366 000000 STAT1: 0
253 001370 000000 STAT2: 0
254 001372 000000 STAT3: 0
255
256 ;DMC11 VECTOR AND REGISTER INDIRECT POINTERS
257 -----
258
259 001374 000000 DMRVEC: 0 ; POINTER TO DMC11 RECEIVER INTERRUPT VECTOR
260 001376 000000 DMRLVL: 0 ; POINTER TO DMC11 RECEIVER INTERRUPT SERVICE PS
261 001400 000000 DMTVEC: 0 ; POINTER TO DMC11 TRANSMITTER INTERRUPT VECTOR
262 001402 000000 DMTLVL: 0 ; POINTER TO DMC11 TRANSMITTER INTERRUPT SERVICE PS
263 001404 000000 DMCSR: 0 ; POINTER TO DMC11 CONTROL STATUS REGISTER
264 001406 000000 DMCSRH: 0 ; POINTER TO DMC11 CONTROL STATUS REGISTER HIGH BYTE.
265 001410 000000 DMCTL: 0 ; POINTER TO DMC11 CONTROL OUT REGISTER
266 001412 000000 DMP04: 0 ; POINTER TO DMC11 PORT REGISTER(SEL 4)
267 001414 000000 DMP06: 0 ; POINTER TO DMC11 PORT REGISTER(SEL 6)
268
269 ;TEMP STORAGE
270 -----
271
272 001416 000000 TEMP: 0
273 001416 001416 . = .+40
274
275 ;DMC11 STATUS TABLE AND ADDRESS ASSIGNMENTS
276 -----
277
278 . = 1500
279 001500 DM.MAP:
280 001500 000001 DMC000: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 00
281 001502 000001 DMS100: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 00
282 001504 000001 DMS200: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 00
283 001506 000001 DMS300: .BLKW 1 ; 3RD STATUS WORD
284
285 001510 000001 DMC001: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 01
286 001512 000001 DMS101: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 01
287 001514 000001 DMS201: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 01
288 001516 000001 DMS301: .BLKW 1 ; 3RD STATUS WORD
289
290 001520 000001 DMC002: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 02
291 001522 000001 DMS102: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 02
292 001524 000001 DMS202: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 02
293 001526 000001 DMS302: .BLKW 1 ; 3RD STATUS WORD
294
295 001530 000001 DMC003: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 03
296 001532 000001 DMS103: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 03
297 001534 000001 DMS203: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 03
298 001536 000001 DMS303: .BLKW 1 ; 3RD STATUS WORD
299
300 001540 000001 DMC004: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 04
301 001542 000001 DMS104: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 04
302 001544 000001 DMS204: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 04
303 001546 000001 DMS304: .BLKW 1 ; 3RD STATUS WORD
    
```

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

D03

CZCMB MACY11 27(1006) 07-MAR-77 15:52 PAGE 9
CZCMB.P:1 07-MAR-77 11:21

PROGRAM PARAMETERS, VARIABLES, AND TRAP CALLS.

PAGE: 0029

360 00:700 000000

DM.END: 000000

PROGRAM PARAMETERS, VARIABLES, AND TRAP CALLS.

361				
362				
363				
364				
365	001702		CNT.MAP:	
366	001702	000000	FACT00: 0	:PASS COUNT FOR DMC11 NUMBER 00
367	001704	000000	ERCT00: 0	:ERROR COUNT FOR DMC11 NUMBER 00
368				
369	001706	000000	PACT01: 0	:PASS COUNT FOR DMC11 NUMBER 01
370	001710	000000	ERCT01: 0	:ERROR COUNT FOR DMC11 NUMBER 01
371				
372	001712	000000	PACT02: 0	:PASS COUNT FOR DMC11 NUMBER 02
373	001714	000000	ERCT02: 0	:ERROR COUNT FOR DMC11 NUMBER 02
374				
375	001716	000000	PACT03: 0	:PASS COUNT FOR DMC11 NUMBER 03
376	001720	000000	ERCT03: 0	:ERROR COUNT FOR DMC11 NUMBER 03
377				
378	001722	000000	PACT04: 0	:PASS COUNT FOR DMC11 NUMBER 04
379	001724	000000	ERCT04: 0	:ERROR COUNT FOR DMC11 NUMBER 04
380				
381	001726	000000	PACT05: 0	:PASS COUNT FOR DMC11 NUMBER 05
382	001730	000000	ERCT05: 0	:ERROR COUNT FOR DMC11 NUMBER 05
383				
384	001732	000000	PACT06: 0	:PASS COUNT FOR DMC11 NUMBER 06
385	001734	000000	ERCT06: 0	:ERROR COUNT FOR DMC11 NUMBER 06
386				
387	001736	000000	PACT07: 0	:PASS COUNT FOR DMC11 NUMBER 07
388	001740	000000	ERCT07: 0	:ERROR COUNT FOR DMC11 NUMBER 07
389				
390	001742	000000	PACT10: 0	:PASS COUNT FOR DMC11 NUMBER 10
391	001744	000000	ERCT10: 0	:ERROR COUNT FOR DMC11 NUMBER 10
392				
393	001746	000000	PACT11: 0	:PASS COUNT FOR DMC11 NUMBER 11
394	001750	000000	ERCT11: 0	:ERROR COUNT FOR DMC11 NUMBER 11
395				
396	001752	000000	PACT12: 0	:PASS COUNT FOR DMC11 NUMBER 12
397	001754	000000	ERCT12: 0	:ERROR COUNT FOR DMC11 NUMBER 12
398				
399	001756	000000	PACT13: 0	:PASS COUNT FOR DMC11 NUMBER 13
400	001760	000000	ERCT13: 0	:ERROR COUNT FOR DMC11 NUMBER 13
401				
402	001762	000000	PACT14: 0	:PASS COUNT FOR DMC11 NUMBER 14
403	001764	000000	ERCT14: 0	:ERROR COUNT FOR DMC11 NUMBER 14
404				
405	001766	000000	PACT15: 0	:PASS COUNT FOR DMC11 NUMBER 15
406	001770	000000	ERCT15: 0	:ERROR COUNT FOR DMC11 NUMBER 15
407				
408	001772	000000	PACT16: 0	:PASS COUNT FOR DMC11 NUMBER 16
409	001774	000000	ERCT16: 0	:ERROR COUNT FOR DMC11 NUMBER 16
410				
411	001776	000000	PACT17: 0	:PASS COUNT FOR DMC11 NUMBER 17
412	002000	000000	ERCT17: 0	:ERROR COUNT FOR DMC11 NUMBER 17
413				

414
415
416
417
418
419

FORMAT OF STATUS TABLE

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

DEFINITION OF FORMAT

- CSR: CONTAINS DMC11 CSR ADDRESS
- STAT1: BITS 00-08 IS DMC11 VECTOR ADDRESS
 BIT15=1 MICRO-PROCESSOR HAS CRAM
 BIT15=0 MICRO-PROCESSOR HAS CROM
 BIT14=1 ???? TURNAROUND CONNECTOR IS ON
 BIT14=0 NO TURNAROUND CONNECTOR
 BIT13=0 LINE UNIT IS AN M8201
 BIT13=1 LINE UNIT IS AN M8202
 BIT12=1 NO LINE UNIT
 BITS 09-11 IS DMC11 BR PRIORITY LEVEL
- STAT2: LOW BYTE IS SWITCH PAC#1 (DDCMP LINE NUMBER)
 HIGH BYTE IS SWITCH PAC#2 (BM873 BOOT ADD)
- STAT3: BIT0=1 DO FREE RUNNING TESTS ON KMC
 (MUST BE SET TO A ONE MANUALLY [PROGRAMS G AND H ONLY]).

470											
471											:PROGRAM INITIALIZATION
472											:LOCK OUT INTERRUPTS
473											:SET UP PROCESSOR STACK
474											:SET UP POWER FAIL VECTOR
475											:CLEAR PROGRAM CONTROL FLAGS AND COUNTS
476											:TYPE TITLE MESSAGE
477											
478	002002	012737	000340	17777E		.START:	MOV	#340,PS			:LOCK OUT INTERRUPTS
479	002010	012706	001200				MOV	#STACK,SP			:SET UP STACK
480	002014	012737	005240	000024			MOV	#PFAIL,2#24			:SET UP POWER FAIL VECTOR
481	002022	013737	001310	001314			MOV	DMNUM,SAVNUM			:SAVE NUMBER OF DEVICES IN SYSTEM.
482	002030	005037	007556				CLR	SWFLG			:CLEAR SOFT TIMEOUT FLAG
483	002034	105037	001325				CLRB	ERRFLG			:CLEAR ERROR FLAG
484	002040	105037	001327				CLRB	QV.FLG			:ZERO QUICK VERIFY FLAG
485	002044	012737	001470	00132C			MOV	#DM.MAP-10,CREAM			:GET MAP POINTER.
486	002052	012737	001676	001322			MOV	#CNT.MAP-4,MILK			:GET PASS COUNT MAP POINTER
487	002060	012737	100000	00131E			MOV	#BIT15,RUN			:POINT POINTER TO FIRST DEVICE.
488	002066	012700	001702				MOV	#CNT.MAP,RO			:PASS COUNT POINTER TO RO
489	002072	005020				23\$:	CLR	(RO)+			:CLEAR TABLE
490	002074	022700	002002				CMP	#CNT.MAP+100,RO			:DONE YET?
491	002100	001374					BNE	23\$:KEEP GOING
492	002102	005037	001234				CLR	LASTERR			:CLEAR LAST ERROR POINTER
493	002106	012737	000001	001226			MOV	#1,TSTNO			:SET UP FOR TEST 1
494	002114	012737	002002	001214			MOV	#.START,RETURN			:SET UP FOR POWER FAIL BEFORE
495											:TESTING STARTS
496	002122	013746	000006				MOV	2#6,-(SP)			:SAVE CURRENT VECTORS
497	002126	013746	000004				MOV	2#4,-(SP)			
498	002132	012737	002166	000004			MOV	6\$,2#4			:SET UP FOR TIMEOUT
499	002140	012737	177570	001202			MOV	#177570,SWR			:SET SWR TO HARD SWR ADDRESS
500	002146	012737	177570	001200			MOV	#177570,DISPLAY			:SET DISPLAY TO HARD SWR ADDRESS
501	002154	022777	177777	177020			CMP	#-1,2SWR			:REFERENCE HARDWARE SWITCH REGISTER
502	002162	001402					BEQ	6\$+2			:IF = -1 USE SOFT SWR ANYWAY
503	002164	000407					BR	7\$:IF IT EXISTS AND NOT = -1 USE HARD SWR
504	002166	022626				6\$:	CMP	(SP)+,(SP)+			:ADJUST STACK
505	002170	012737	000176	001202			MOV	#SWREG,SWR			:POINTER TO SOFT SWR
506	002176	012737	000174	001200			MOV	#DISPREG,DISPLAY			:POINTER TO SOFT DISPLAY REG
507	002204	012637	000004			7\$:	MOV	(SP)+,2#4			:RESTORE VECTORS
508	002210	012637	000006				MOV	(SP)+,2#6			
509	002214	105737	001324				TSTB	INIFLG			:HAS INITIALIZATION BEEN PERFORMED
510	002220	001006					BNE	20\$:BR IF YES
511	002222	022737	003432	000042			CMP	#SENDAD,2#42			:IF ACT-11 AUTOMATIC MODE. DON'T TYPE ID
512	002230	001402					BEQ	20\$			
513	002232	104402	001000				TYPE	MTITLE			:TYPE TITLE MESSAGE
514	002236	004737	007362			20\$:	JSR	PC,CKSWR			:CHECK FOR SOFT SWR
515	002242	017737	176734	001236			MOV	2SWR,STRTSW			:STORE STARTING SWITCHES
516	002250	005737	000042				TST	2#42			:IS IT RUNNING IN AUTO MODE?
517	002254	001402					BEQ	+6			:BR IF NO
518	002256	005037	001236				CLR	STRTSW			:IF YES, CLEAR SWITCHES
519	002262	032737	000001	001236			BIT	#SW00,STRTSW			:IF SW00=1, QUESTIONS ARE ASKED.
520	002270	001012					BNE	17\$:BR IF SW00=1
521	002272	105737	001236				TSTB	STRTSW			:BIT7=1??
522	002276	100007					BPL	17\$:BR IF SW07=0
523	002300	005737	001306				TST	DMACTV			:ARE ANY DEVICES SELECTED?
524	002304	001006					BNE	16\$:BR IF YES
525	002306	104402	007056				TYPE,	NDACT			:NO DEVICES SELECTED.

PROGRAM INITIALIZATION AND START UP.

```

526 002312 000000 HALT ;STOP THE SHOW
527 002314 000776 BR -2 ;DISQUALIFY CONTINUE SWITCH
528 002316 004737 013252 17$: JSR PC,AUTO.SIZE ;GO DO THE AUTO SIZE
529 002322 105737 001324 16$: TSTB INIFLG ;FIRST TIME?
530 002326 001410 BEQ 21$ ;BR IF YES
531 002330 105737 001236 TSTB STRTSW ;IF USING SAME PARAMETERS DONT TYPE MAP
532 002334 100421 BMI 1$
533 002336 032737 000006 001236 BIT #BIT1!BIT2,STRTSW ;IS TEST NO. OR LOCK SELECTED
534 002344 001403 BEQ 24$ ;IF NO THEN TYPE STATUS
535 002346 000424 SR 1$ ;IF YES DO NOT TYPE STATUS
536 002350 005137 001324 21$: COM INIFLG ;SET FLAG
537 002354 104402 006126 24$: TYPE XHEAD ;TYPE HEADER
538 002360 012704 001500 MOV #DM.MAP,R4 ;SET POINTER
539 002364 010437 001246 5$: MOV R4,TEMP1 ;SET ADDRESS
540 002370 012437 001250 MOV (R4)+,TEMP2 ;SET CSR
541 002374 001411 BEQ 1$ ;ALL DONE IF ZERO
542 002376 012437 001252 MOV (R4)+,TEMP3 ;SET STAT1
543 002402 012437 001254 MOV (R4)+,TEMP4 ;SET STAT2
544 002406 012437 001256 MOV (R4)+,TEMP5 ;SET STAT3
545 002412 104410 CONVRT ;TYPE OUT STATUS MAP
546 002414 007230 XSTATQ ;
547 002416 000762 BR 5$
548 002420 012700 1$: MOV #DM.MAP,R0 ;R0 POINTS TO STATUS TABLE

```

```

*****
;*AUTO SIZE TEST
;*THIS TEST VERIFYS THAT THE DMC11S AND/OR KMC11S ARE AT THE CORRECT FLOATING
;*ADDRESSES FOR YOUR SYSTEM. IF THIS TEST FAILS, IT IS NOT A HARDWARE ERROR.
;*CHECK THE ADDRESSES OF ALL FLOATING DEVICES (DJ,DH,DQ,DU,DUP,LK,DMC,DZ,KMC).
;*IF THERE ARE NO OTHER FLOATING DEVICES BEFORE THE DMC11, THE FIRST
;*DMC11 ADDRESS IS 760070, KMC11 IS 760110. NO DEVICE SHOULD EVER BE AT
;*ADDRESS 760000.
*****

```

```

560 002424 013746 000004 MOV @#4,-(SP) ;SAVE LOC 4
561 002430 013746 000006 MOV @#6,-(SP) ;SAVE LOC 6
562 002434 005037 000006 CLR @#6 ;CLEAR VEC+2
563 002440 005037 001252 CLR TEMP3 ;CLEAR FLAG
564 002444 005005 CLR R5 ;R5=0=DMC, R5=-1=KMC
565 002446 011037 001404 AJSTRT: MOV (R0),DMCSR ;GET NEXT DMC CSR
566 002452 001530 BEQ AUDONE ;BR IF DONE
567 002454 005705 TST R5 ;DMC OR KMC?
568 002456 001005 BNE 1$ ;BR IF KMC
569 002460 032760 100000 000002 BIT #BIT15,2(R0) ;CHECK FOR DMC CSR
570 002466 001044 BNE OK ;SKIP IF NOT DMC
571 002470 000404 BR 2$ ;ITS A DMC SO CONTINUE
572 002472 032760 100000 000002 1$: BIT #BIT15,2(R0) ;CHECK FOR KMC CSR
573 002500 001437 BEQ OK ;SKIP IF NOT KMC
574 002502 012737 002606 000004 2$: MOV #NODEV,@#4 ;SET UP FOR TIMEOUT
575 002510 005705 TST R5 ;DMC OR KMC?
576 002512 001003 BNE 3$ ;BR IF KMC
577 002514 012703 000006 MOV #6,R3 ;R3 IS COUNT OF DEVICES BEFORE DMC
578 002520 000402 BR 4$ ;GO ON
579 002522 012703 000010 3$: MOV #10,R3 ;R3 IS COUNT OF DEVICES BEFORE KMC
580 002526 012702 002722 4$: MOV #DEVTAB,R2 ;R2 IS DEVICE TABLE PONTER
581 002532 012701 160010 MOV #160010,R1 ;START WITH ADDRESS 160010

```

PROGRAM INITIALIZATION AND START UP.

582	002536	005711			FLOAT:	TST	(R1)	:CHECK ADDRESS IN R1
583	002540	111204				MOVB	(R2),R4	:IF NO TIMEOUT, GET NEXT ADDRESS
584	002542	060401				ADD	R4,R1	:IN R1
585	002544	005201				INC	R1	:
586	002546	040401				BIC	R4,R1	:
587	002550	005703				TST	R3	:ANY MORE DEVICES TO CHECK FOR?
588	002552	001371				BNE	FLOAT	:BR IF YES
589	002554	012737	002612	000004		MOV	#ERR,2#4	:OK ONLY DMC'S ARE LEFT. SET JP FOR TIMEOUT
590	002562	005711			FY:	TST	(R1)	:CHECK DMC ADDRESS
591	002564	020137	001404			CMP	R1,DMCSR	:DOES IT MATCH
592	002570	001403				BEQ	OK	:BR IF YES
593	002572	062701	000010			ADD	#10,R1	:GET NEXT DMC ADDRESS
594	002576	000771				BR	FY	:DO IT AGAIN
595	002600	062700	000010		OK:	ADD	#10,R0	:SKIP TO NEXT DMC CSR
596	002604	000720				BR	AUSTR	:CONTINUE
597	002606	122243			NODEV:	CMPB	(R2)+,-(R2)	:ON TIMEOUT, INC R2, DEC R3
598	002610	000002				RTI		:RETURN
599	002612	005737	001252		ERR:	TST	TEMP3	:CHECK FLAG IF = 0 TYPE HEADER
600	002616	001014				BNE	1\$:SKIP HEADER
601	002620	104402				TYPE		:TYPEOUT HEADER MESSAGE
602	002622	007125				CONERR		:CONFIGURATION ERROR!!!!
603	002624	012737	002612	001276		MOV	#ERR,SAVPC	:SAVE PC FOR TYPEOUT
604	002632	104411				CNVRT		:TYPE OUT ERROR PC
605	002634	002702				ERRPC		:
606	002636	104402				TYPE		:TYPE REST OF HEADER
607	002640	007167				CNERR		:
608	002642	012737	177777	001252		MOV	#-1,TEMP3	:SET FLAG SO IT ONLY GETS TYPED ONCE
609	002650	010137	001252		1\$:	MOV	R1,SAVR1	:SAVE R1 FOR TYPEOUT
610	002654	104410				CNVRT		:
611	002656	002710				CONTAB		:TYPE CSR VALUES
612	002660	005705				TST	RE	:DMC OR KMC ?
613	002662	001003				BNE	3\$:BR IF KMC
614	002664	104402				TYPE		:
615	002666	007210				DMCM		:
616	002670	000402				BR	4\$:CONTINUE
617	002672	104402			3\$:	TYPE		:
618	002674	007220				KMCM		:
619	002676	022626			4\$:	CMP	(SP)+,(SP)+	:ADJUST STACK
620	002700	000737				BR	OK	:BR TO GET OUT
621	002702	000001			ERRPC:	1		:
622	002704	006	002			.BYTE	6.2	:
623	002706	001276				SAVPC		:
624	002710	000002			CONTAB:	2		:
625	002712	006	004			.BYTE	6.4	:
626	002714	001262				SAVR1		:
627	002716	006	002			.BYTE	6.2	:
628	002720	001404				DMCSR		:
629	002722	007			DEVTAB:	.BYTE	7	:DJ
630	002723	017				.BYTE	17	:DH
631	002724	007				.BYTE	7	:DQ
632	002725	007				.BYTE	7	:DU
633	002726	007				.BYTE	7	:DUP
634	002727	007				.BYTE	7	:LK
635	002730	007				.BYTE	7	:DMC
636	002731	007				.BYTE	7	:DZ
637	002732	007				.BYTE	7	:KMC

638		002734			.EVEN				
639	002734	005705			AUDONE:	TST	R5		:DMC?
640	002736	001005				BNE	1\$:BR IF KMC AND ALL DONE
641	002740	012705	177777			MOV	#-1,R5		:SET R5 TO -1 (KMC)
642	002744	012700	001500			MOV	#DM.MAP,RO		:RESET RO TO START OF TABLE
643	002750	000636				BR	AUSTR		:GO DO KMC'S
644	002752	012637	000006		1\$:	MOV	(S)+,Q#6		:RESTORE LOC 6
645	002756	012637	000004			MOV	(SP)+,Q#4		:RESTORE LOC 4
646	002762	032737	000010	001236		BIT	#SW03,STRSW		:SELECT SPECIFIC DEVICES??
647	002770	001422				SEQ	3\$:BR IF NO.
648	002772	104402	006046			TYPE	MNEW		:TYPE THE MESSAGE.
649	002776	005000				CLR	RO		:ZERO DATA LIGHTS
650	003000	000000				HALT			:WAIT FOR USER TO TELL WHAT DEVICES TO RUN
651	003002	027737	176174	001312		CMP	QSWR,SAVACT		:IS THE NUMBER VALID?
652	003010	101404				BLOS	2\$:BR IF NUMBER IS OK.
653	003012	104402	005707			TYPE	,MERR3		:TELL USER OF INVALID NUMBER.
654	003016	000000				HALT			:STOP EVERY THING.
655	003020	000776				BR	-2		:RESTART THE PROGRAM AGAIN.
656	003022	017737	176154	001306	2\$:	MOV	QSWR,DMACTV		:GET NEW DEVICE PATTERN
657	003030	013700	001306			MOV	DMACTV,RO		:SHOW THE USER WHAT HE SELECTED.
658	003034	000000				HALT			:CONTINUE DYNAMIC SWITCHES.
659	003036	012700	000300		3\$:	MOV	#300,RO		:PREPARE TO CLEAR THE FLOATING
660	003042	012701	000302			MOV	#302,R1		:VECTOR AREA. 300-776
661	003046	010120			4\$:	MOV	R1,(RO)+		:START PUTTING "PC+2 - HALT"
662	003050	005021				CLR	(R1)+		:IN VECTOR AREA.
663	003052	022021				CMP	(RO)+,(R1)+		:POP POINTERS
664	003054	022700	001000			CMP	#1000,RO		:ALL DONE??
665	003060	001372				BNE	4\$:BR IF NO.
666									
667									
668									
669									
670	003062	012706	001200		.BEGIN:	MOV	#STACK,SP		:SET UP STACK
671	003066	013746	000006			MOV	Q#6,-(SP)		:SAVE LOC 6
672	003072	013746	000004			MOV	Q#4,-(SP)		:SAVE LOC 4
673	003076	005000				CLR	RO		:START AT 0
674	003100	012737	003144	000004		MOV	#2\$,Q#4		:SET UP FOR TIME OUT
675	003106	005037	000006			CLR	Q#6		:TO AUTOSIZE MEMORY
676	003112	005720			6\$:	TST	(RO)+		:CHECK ADDRESS IN RO
677	003114	022700	157776			CMP	#157776,RO		:IS IT AT LEAST 29K
678	003120	001374				BNE	6\$:BR IF NO
679	003122	162700	007776			SUB	#7776,RO		:SAVE 2K FOR MONITORS
680	003126	010037	001304		7\$:	MOV	RO,MEMLIM		:STORE MEMORY LIMIT
681	003132	012637	000004			MOV	(SP)+,Q#4		:RESTORE LOC 4
682	003136	012637	000006			MOV	(SP)+,Q#6		:RESTORE LOC 6
683	003142	000413				BR	10\$:CONTINUE
684	003144	022626			2\$:	CMP	(SP)+,(SP)+		:ADJUST STACK
685	003146	162700	000004			SUB	#4,RO		:GET LAST GOOD ADDRESS
686	003152	162700	007776			SUB	#7776,RO		:SAVE 2K FOR MONITORS
687	003156	022700	030000			CMP	#30000,RO		:IS IT BK?
688	003162	001361				BNE	7\$:BR IF NO
689	003164	012700	037400			MOV	#37400,RO		:IF BK DON'T SAVE 2K
690	003170	000756				BR	7\$		
691	003172	012737	000340	177776	10\$:	MOV	#340,PS		:LOCK OUT INTERRUPTS
692	003200	032737	000004	001236		BIT	#BIT2,STRSW		:CHECK FOR LOCK ON TEST
693	003206	001411				BEG	1\$:BR IF NO LOCK DESIRED.

:TEST START AND RESTART

K03

020M1 MACV11 2711006, 07-MAR-77 15:52 PAGE 16
 020M32.P11 07-MAR-77 11:21

PAGE: 0036

PROGRAM INITIALIZATION AND START UP.

694	003210	104402	005745			TYPE	.MLOCK	;TYPE LOCK SELECTED.
695	003214	012737	000240	003522		MOV	#NOP,TTST	;ADJUST SCOPE ROUTINE.
696	003222	012737	000240	003524		MOV	#NOP,TTST+2	;SET UP TO LOCK
697	003230	000406				BR	3\$;CONTINUE ALONG.
698	003232	013737	003640	003522	1\$:	MOV	BRW,TTST	;PREPARE NORMAL SCOPE ROUTINE
699	003240	013737	003642	003524		MOV	BRX,TTST+2	;LOCK NOT SELECTED, SET UP FOR NORMAL SCOPE LOOP
700	003246	012737	007620	001214	3\$:	MOV	#CYCLE,RETURN	;START AT "CYCLE" FIND WHICH DEVICE TO TEST
701	003254	032737	000002	001236	4\$:	BIT	#SW01,STATSW	;IS TEST NO. SELECTED?
702	003262	001002				BNE	5\$;BR IF YES
703	003264	104402	005657			TYPE	MR	;TYPE R
704	003270	000177	175720		5\$:	JMP	JRETURN	;START TESTING

M03

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

761	003514	032777	040000	175460					
762	003522	001407			TTST:	BIT	#BIT14, @SWR	;	"LOOP ON THIS TEST"
763	003524	000437				BEQ	1\$;	BR IF NO. (IF LOCK SW01=1; THIS LOC =240)
764	003526	105777	175452			BR	3\$;	GOTO 3\$ (IF LOCK SW01=1; THIS LOC =240)
765	003532	100034				TSTB	@TKCSR	;	KEYBOARD DONE?
766	003534	017700	175446			BPL	3\$;	BR IF NO. (LOCK: HIT KEY TO GOTO NEXT TEST)
767	003540	000415				MOV	@TKDBR, R0	;	CLEAR DONE BIT
768	003542	032777	004000	175432	1\$:	BR	2\$;	CONTINUE
769	003550	001011				BIT	#SW11, @SWR	;	DELETE ITERATION? (QUICK PASS)
770	003552	105737	001327			BNE	2\$;	BR IF YES
771	003556	001406				TSTB	QV.FLG	;	HAVE PASSES BEECOMPLETED?
772	003560	005237	001224			BEQ	2\$;	BR IF QUICK PASS.
773	003564	023737	001224	001222		INC	LPCNT	;	UPDATE ITERATION COUNTER
774	003572	101414				CMP	LPCNT, ICOUNT	;	ARE ALL ITERATIONS DONE?"
775	003574	105037	001325		2\$:	BLOS	3\$;	BR IF NOT YET
776	003600	005037	001224			CLRB	ERRFLG	;	PREPARE FOR NEW TEST
777	003604	005037	001220			CLR	LPCNT	;	START ICOUNTER AT 0
778	003610	012737	000020	00:222		CLR	LOCK	;	
779	003616	013737	001216	001214		MOV	#20, ICOUNT	;	RESET ITERATIONS
780	003624	011600			3\$:	MOV	NEXT, RETURN	;	GET NEXT TEST
781	003626	022626				MOV	(SP), R0	;	POP R0 OFF OF THE STACK
782	003630	013701	001404			POP2SP		;	FAKE AN "RTI"
783	003634	000177	175354			MOV	DMCSR, R1	;	R1 CONTAINS BASE DMC ADDRESS
784	003640	001407			BRW:	JMP	@RETURN	;	GO DO THE TEST
785	003642	000437			BRX:	1407			
786						437			
787									
788									
789									
790	003644	004737	007362		.SCOPI:	JSR	PC, CKSWR	;	CHECK FOR SOFT SWR
791	003650	032777	001000	175324		BIT	#SW09, @SWR	;	IS SW09=1 (SET)?
792	003656	001405				BEQ	1\$;	BR IF NOT SET.
793	003660	005737	001220			TST	LOCK	;	
794	003664	001402				BEQ	1\$;	
795	003666	013716	001220		1\$:	MOV	LOCK, (SP)	;	GOTO THE ADDRESS IN LOCK.
796	003672	000002				RTI		;	GO BACK.
797									
798									
799									
800									
801	003674	010546			.TYPE:	MOV	R5 - (SP)	;	SAVE R5 ON THE STACK.
802	003676	017605	000002			MOV	@2(SP), R5	;	GET ADDRESS OF MESSAGE.
803	003702	062766	000002	000002		ADD	#2, 2(SP)	;	POP OVER ADDRESS.
804	003710	005737	007556		4\$:	TST	SWFLG	;	SOFT SWR MESSAGE?
805	003714	001004				BNE	1\$;	IF YES TYPE IT OUT REGARDLESS OF SW12
806	003716	032777	010000	175256		BIT	#SW12, @SWR	;	INHIBIT ALL PRINT OUT?"
807	003724	001012				BNE	3\$;	BR IF NO PRINT OUT WANTED (SW12=1)
808	003726	105715			1\$:	TSTB	(R5)	;	IS NUMBER MINUS? (MSB=1 (BIT?))
809	003730	100002				BPL	2\$;	BR IF NUMBER IS PLUS
810	003732	104402	005574			TYPE	MCRLF	;	TYPE A CR/LF!
811	003736	105777	175246		2\$:	TSTB	@TPCSR	;	TTY READY?
812	003742	100375				BPL	2\$;	BR IF NO.
813	003744	112577	175242			MOV	(R5)+, @TPDBR	;	PRINT CURRENT CHAR.
814	003750	001357				BNE	4\$;	IF NOT ZERO KEEP PRINTING!
815	003752	012605			3\$:	MOV	(SP)+, R5	;	END OF OUTPUT. RESTORE R5
816	003754	000002				RTI		;	GO HOME

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

817
818
819 003756 010346 .INSTR: MOV R3,-(SP) :SAVE R3 ON STACK
820 003760 010446 MOV R4,-(SP) :SAVE R4 ON STACK
821 003762 017637 000004 004000 MOV 4(SP),MSG
822 003770 062766 000002 000004 ADD #2,4(SP)
823 003776 104402 .INST1: TYPE
824 004000 000000 .MSG: 0
825 004002 012704 007256 MOV #INBUF,R4
826 004006 012703 000007 MOV #7,R3
827 004012 105777 175166 1$: TSTB @TKCSR
828 004016 100375 BPL 1$
829 004020 117714 175162 MOVB @TKDBR,(R4)
830 004024 142714 000200 BICB #200,(R4)
831 004030 122427 000015 CMPB (R4),#15
832 004034 001417 BEQ 1
833 004036 105777 175146 2$: TSTB @TPCSR
834 004042 100375 BPL 2$
835 004044 017777 175136 175140 MOV @TKDBR,@TPDBR
836 004052 005303 DEC R3
837 004054 001356 BNE 1$
838 004056 012604 MOV (SP)+,R4
839 004060 012603 MOV (SP)+,R3
840 004062 104402 005570 .INSTE: TYPE MQM
841 004066 010346 MOV R3,-(SP)
842 004070 010446 MOV R4,-(SP)
843 004072 000741 BR .INST1
844 004074 012604 INSTR2: MOV (SP)+,R4 :RESTORE R4
845 004076 012603 MOV (SP)+,R3 :RESTORE R3
846 004100 000002 RTI
847
848 ;CONVERT ASCII STRING TO OCTAL
849
850
851 004102 010546 .PARAM: MOV R5,-(SP)
852 004104 010446 MOV R4,-(SP)
853 004106 016605 000004 MOV 4(SP),R5
854 004112 012537 004272 MOV (R5)+,LOLIM
855 004116 012537 004274 MOV (R5)+,HILIM
856 004122 012537 004276 MOV (R5)+,DEVADR
857 004126 112537 004300 MOVB (R5)+,LOBITS
858 004132 112537 004301 MOVB (R5)+,ADRCNT
859 004136 010566 000004 MOV R5,4(SP)
860 004142 005005 PARAM1: CLR R5
861 004144 012704 007256 MOV #INBUF,R4
862 004150 122714 000015 CMPB #15,(R4)
863 004154 001420 1$: BEQ PARERR
864 004156 121427 000060 CMPB (R4),#60
865 004162 002415 BLT PARERR
866 004164 121427 000067 CMPB (R4),#67
867 004170 003012 BGT PARERR
868 004172 142714 000060 BICB #60,(R4)
869 004176 152405 BISB (R4)+,R5
870 004200 122714 000015 CMPB #15,(R4)
871 004204 001406 BEQ LIMITS
872 004206 006305 ASL R5
    
```

873	004210	006305			ASL	R5	
874	004212	006305			ASL	R5	
875	004214	0007E0			BR	1\$	
876	004216	104404			PARERR: INSTER		
877	004220	000750			BR	PARAM1	
878							
879							
880							
881							
882	004222	020537	004274		LIMITS: CMP	R5, HILIM	
883	004226	101373			BHI	PARERR	
884	004230	020537	004272		CMP	R5, LOLIM	
885	004234	103770			BLO	PARERR	
886	004236	133705	004300		BITB	LOBITS, R5	
887	004242	001365			BNE	PARERR	
888							
889							
890							
891	004244	013704	004276				
892	004250	010524			1\$: MOV	DEVADR, R4	
893	004252	062705	000002		MOV	R5, (R4)+	
894	004256	105337	004301		ADD	#2, R5	
895	004262	001372			DECB	ADRCNT	
896	004264	012604			BNE	1\$	
897	004266	012605			MOV	(SP)+, R4	
898	004270	000002			MOV	(SP)+, R5	
899	004272	000000			RTI		
900	004274	000000			LOLIM: 0		
901	004276	000000			HILIM: 0		
902	004300	000000			DEVADR: 0		
903		004301			LOBITS: 0		
904					ADRCNT=LOBITS+1		
905							
906							
907							
908	004302	016637	000004	001276	.SAV05: MOV	4(SP), SAVPC	;SAVE R7 (PC)
909							
910							
911							
912	004310	010537	001272		SV05: MOV	R5, SAVR5	;SAVE R5
913	004314	010437	001270		MOV	R4, SAVR4	;SAVE R4
914	004320	010337	001266		MOV	R3, SAVR3	;SAVE R3
915	004324	010237	001264		MOV	R2, SAVR2	;SAVE R2
916	004330	010137	001262		MOV	R1, SAVR1	;SAVE R1
917	004334	010037	001260		MOV	R0, SAVR0	;SAVE R0
918	004340	000002			RTI		;LEAVE.
919							
920							
921							
922	004342	013700	001260		.RES05: MOV	SAVR0, R0	;RESTORE R0
923	004346	013701	001262		MOV	SAVR1, R1	;RESTORE R1
924	004352	013702	001264		MOV	SAVR2, R2	;RESTORE R2
925	004356	013703	001266		MOV	SAVR3, R3	;RESTORE R3
926	004362	013704	001270		MOV	SAVR4, R4	;RESTORE R4
927	004366	013705	001272		MOV	SAVR5, R5	;RESTORE R5
928	004372	000002			RTI		;LEAVE

929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984

004374 104402 005574
004400 010046
004402 010146
004404 010346
004406 010446
004410 010546
004412 017601 000012
004416 062766 000002 000012
004424 012137 004616
004430 112137 004620
004434 112137 004621
004440 013137 004622
004444 122737 000003 004620
004452 001003
004454 042737 177400 004622
004462 013704 004622
004466 113705 004620
004472 012700 001416
004476 010403
004500 042703 177770
004504 062703 000060
004510 110320
004512 000241
004514 006004
004516 000241
004520 006004
004522 000241
004524 006004
004526 005305
004530 001362
004532 012703 007320
004536 114023
004540 105337 004620
004544 001374
004546 105737 004621
004552 001405
004554 112723 000040
004560 105337 004621
004564 001373
004566 105013
004570 104402 007320
004574 005337 004616
004600 001313
004602 012605
004604 012604
004606 012603
004610 012601
004612 012600
004614 000002
004616 000000
004620 000000
004621 004621

```
; CONVERT OCTAL NUMBER TO ASCII AND OUTPUT TO TELEPRINTER
-----
.CONVR: TYPE      MCRLF
.CNVRT: MOV       R0,-(SP)
        MOV       R1,-(SP)
        MOV       R3,-(SP)
        MOV       R4,-(SP)
        MOV       R5,-(SP)
        MOV       @12(SP),R1
        ADD       #2,12(SP)
        MOV       (R1)+,WRDCNT
1$:     MOVB      (R1)+,CHRCNT
        MOVB      (R1)+,SPACNT
        MOV       @2(R1)+,BINWRD
        CMPB     #3,CHRCNT
        BNE      2$
        BIC      #177400,BINWRD
2$:     MOV       BINWRD,R4
        MOVB     CHRCNT,R5
        MOV       #TEMP,R0
3$:     MOV       R4,R3
        BIC      #177770,R3
        ADD      #060,R3
        MOVB     R3,(R0)+
        CLC
        ROR      R4
        CLC
        ROR      R4
        CLC
        ROR      R4
        DEC      R5
        BNE      3$
4$:     MOV       #MDATA,R3
        MOVB     -(R0),(R3)+
        DECB    CHRCNT
        BNE      4$
        TSTB    SPACNT
        BEQ     5$
5$:     MOVB     #040,(R3)+
        DECB    SPACNT
        BNE      5$
6$:     CLRB     (R3)
        TYPE    ,MDATA
        DEC     WRDCNT
        BNE     1$
        MOV     (SP)+,R5
        MOV     (SP)+,R4
        MOV     (SP)+,R3
        MOV     (SP)+,R1
        MOV     (SP)+,R0
        RTI
WRDCNT: 0
CHRCNT: 0
SPACNT=CHRCNT+1
```


GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

985 004622 000000

BINWRD: 0

986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040

004624 011646
004626 162716 000002
004632 017616 000000
004636 006316
004640 042716 177001
004644 062716 001330
004650 01761E 000000
004654 00013E

004737 007362
004662 032777 010000 174312
004670 001406
004672 105777 174312
004676 100003
004700 112777 000207 174304
004706 032777 020000 174266
004714 001105
004716 021637 001234
004722 001404
004724 011637 001234
004730 105037 001325
004734 104406
004736 011605
004740 162705 000002
004744 011504
004746 006304
004750 061504
004752 006304
004754 042704 177001
004760 062704 037250
004764 012437 005100
004770 012437 005112
004774 011437 005124
005000 105737 001325
005004 001403
005006 005737 005124
005012 001040
005014 104402 005574
005020 104402 005574
005024 005737 001220
005030 001402
005032 104402 006044
005036 104402 006032
005042 104411 005232
005046 104402 006121

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

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

: ERROR HANDLER
: -----

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

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1041	005052	104411	005224		CNVRT	,ERTABO	:SHOW IT
1042	005056	104402	005574		TYPE	MCRLF	:GIVE A CR/LF
1043	005062	112737	177777	001325	MOV8	#-1,ERRFLG	:NO MORE HEADER UNLESS NO DATA TABLE.
1044	005070	005737	005100		TST	ERRMSG	:IS THERE AN ERROR MESSAGE?
1045	005074	001402			BEQ	WRKO.FM	:BR IF NO.
1046	005076	104402			TYPE		:TYPE
1047	005100	000000			ERRMSG: 0		:ERROR MESSAGE
1048	005102				WRKO.FM:		
1049	005102	005737	005112		TST	DATAHD	:DATA HEADER?
1050	005106	001402			SEQ	TYPDAT	:BR IF NO
1051	005110	104402			TYPE		:TYPE
1052	005112	000000			DATAHD: 0		:DATA HEADER
1053	005114	005737	005124		TYPDAT: TST	DATABP	:DATA TABLE?
1054	005120	001402			BEQ	RESREG	:BR IF NO.
1055	005122	104410			CONVRT		:SHOW
1056	005124	000000			DATABP: 0		:DATA TABLE
1057	005126	104407			RESREG: RES05		:RESTORE PROC REGISTERS
1058	005130	022737	003432	000042	HALTS: CMP	#SENDAD, 2#42	:IF ACT-11 AUTOMATIC MODE, HALT!!
1059	005136	001403			BEQ	1\$	
1060	005140	005777	174036		TST	2\$WR	:HALT ON ERROR?
1061	005144	100005			BPL	EXITER	:BR IF NO HALT ON ERROR
1062	005146	010046			1\$: PUSHRO		:SAVE R0
1063	005150	016600	000002		MOV	2(SP),R0	:SHOW ERROR PC IN DATA LIGHTS
1064	005154	000000			HALT		:HALT
1065	005156	012600			POPPO		:GET R0
1066	005160	005237	001232		EXITER: INC	ERRCNT	:UPDATE ERROR COUNT
1067	005164	032777	000400	174010	BIT	#SW08, 2\$WR	:GOTO TOP OF TEST?
1068	005172	001007			BNE	1\$:BR IF YES
1069	005174	032777	002000	174000	BIT	#SW10, 2\$WR	:GOTO NEXT TEST?
1070	005202	001407			BEQ	2\$:BR IF NO
1071	005204	013737	001216	001214	MOV	NEXT, RETURN	:SET FOR NEXT TEST
1072	005212	012706	001200		1\$: MOV	#STACK, SP	:RESET SP
1073	005216	000177	173772		JMP	2\$RETURN	:GOTO SPECIFIED TEST
1074	005222	000002			2\$: RTI		:RETURN
1075	005224	000001			ERTABO: 1		
1076	005226	006	002		.BYTE	6.2	
1077	005230	001276			SAVPC		
1078	005232	000001			XTSTN: 1		
1079	005234	003	002		.BYTE	3.2	
1080	005236	001226			TSTNO		
1081							:ENTER HERE ON POWER FAILURE
1082							-----
1083							
1084							
1085	005240				.PFAIL:		
1086	005240	012737	005252	000024	MOV	#RESTART, 24	:SET UP FOR POWER UP TRAP
1087	005246	000000			HALT		:HALT ON PCWER DOWN NORMAL
1088	005250	000777			BR	.	
1089							
1090							:PROCESSOR WILL TRAP HERE WHEN POWER IS RESTORED
1091							
1092	005252				RESTAR:		
1093	005252	012737	005240	000024	MOV	#.PFAIL, 24	:SET UP FOR POWER FAILURE
1094	005260	012706	001200		MOV	#STACK, SP	:RESET THE STACK POINTER
1095	005264	013701	001404		MOV	CMCSR, R1	:RESTORE R1
1096	005270	005037	001416		CLR	TEMP	:READY FOR TIMMER

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1097 005274 005237 001416      INC      TEMP      ;PLUS ONE TO THE TIMER!
1098 005300 001375      BNE      .-4       ;BR IF MORE TO GO
1099 005302 104402 005577      TYPE     ,MPFAIL   ;TYPE THE MESSAGE
1100 005306 104411 005332      CNVRT    ,PFTAB    ;TELL WHAT TEST TO RETURN TO.
1101 005312 105037 001325      CLRB     ERRFLG    ;START CLEAN
1102 005316 005037 001234      CLR      LSTERR    ;.....
1103 005322 005011      CLR      (R1)     ;CLEAR MAINT BITS
1104 005324 104412      MSTCLR  ;START CLEAN UP OF DEVICE
1105 005326 000177 173662      JMP      @RETURN  ;START DOING THAT TEST AGAIN.
1106 005332 000001      PFTAB: 1
1107 005334 003      .BYTE   3,2
1108 005336 001226      TSTNO
1109
1110
1111 005340      .DELAY:
1112 005340 012777 000020 174044      MOV      #20,@DMP04
1113 005346 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1114 005350 121111      121111  ;POKE CLOCK DELAY BIT
1115
1116 005352      1$:
1117 005352 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1118 005354 121224      121224  ;PORT4+IBUS*11
1119 005356 032777 000020 174026      BIT      #BIT4,@DMP04 ;IS CLOCK BIT SET?
1120 005364 001772      BEQ     1$       ;BR IF NO
1121 005366 000002      RTI
1122
1123 005370      .MSTCLR:
1124 005370 152777 000100 174010      BISB    #BIT6,@DMCSRH ;SET MASTER CLEAR
1125 005376 142777 000300 174002      BICB    #BIT6!BIT7,@DMCSRH ;CLEAR MASTER CLEAR AND RUN
1126 005404 000002      RTI      ;RETURN
1127
1128 005406      .ROMCLK:
1129 005406 152777 000002 173772      BISB    #BIT1,@DMCSRH ;SET ROMI
1130 005414 013677 173774      MOV     @ (SP)+,@DMP06 ;LOAD INSTRUCTION IN SEL5
1131 005420 062746 000002      ADD     #2,-(SP)     ;ADJUST STACK
1132 005424 032777 000100 173550      BIT     #SW06,@SWR   ;HALT IF SW06 =1
1133 005432 001401      BEQ     1$         ;BR IF SW06 =0
1134 005434 000000      HALT    ;HALT BEFORE CLOCKING INSTRUCTION
1135 005436 152777 000003 173742      BISB    #BIT1!BIT0,@DMCSRH ;CLOCK INSTRUCTION
1136 005444 142777 000007 173734      BICB    #BIT2!BIT1!BIT0,@DMCSRH ;CLEAR ROMO, ROMI, STEP
1137 005452 000002      RTI
1138
1139 005454      .DATACLK:
1140 005454 013637 001416      MOV     @ (SP)+,TEMP  ;PUT TICK COUNT IN TEMP
1141 005460 062746 000002      ADD     #2,-(SP)     ;ADJUST STACK
1142 005464 152777 000020 173714      BISB    #BIT4,@DMCSRH ;SET STEP LU
1143 005472 027777 173706 173704      CMP     @DMCSR,@DMCSR ;WASTE TIME
1144 005500 142777 000020 173700      BICB    #BIT4,@DMCSRH ;CLEAR STEP LL
1145 005506 005337 001416      DEC     TEMP        ;DEC TICK COUNT
1146 005512 001364      BNE     1$         ;BR IF NOT DONE
1147 005514 000002      RTI      ;RETURN
1148 005516 000001      3$:      .BLKW 1
1149
1150 005520      .TIMER:
1151 005520 013637 001416      MOV     @ (SP)+,TEMP  ;MOVE COUNT TO TEMP
1152 005524 062746 000002      ADD     #2,-(SP)     ;ADJUST STACK
1153 005530      1$:
1154 005530 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
    
```

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1153 005532 021364 000002 173650 021364 ;PORT4+IBUS* REG11
1154 005534 032777 000002 173650 BIT #2,JDMP04 ;IS PGM CLOCK BIT CLEAR?
1155 005542 001772 29: BEQ 19 ;BR IF YES
1156 005544 104414 ROMCLK ;NEXT WORD IS INSTRUCTION. ROMCLK PC=530-
1157 005544 021364 021364 ;PORT4+IBUS* REG11
1158 005546 032777 000002 173634 BIT #2,JDMP04 ;IS PGM CLOCK BIT SET?
1159 005550 001372 29: BNE 29 ;BR IF YES
1160 005556 005337 0C1416 DEC TEMP ;DEC COUNT
1161 005560 001361 19: SNE 19 ;BR IF NOT DONE
1162 005564 000002 RTI ;RETURN
1163
1164
1165 005570 020040 000077 MQM: .ASCIZ / ? /
(2) 005574 005015 000 MCRLF: .ASCIZ <15><12>
(2) 005577 377 053520 020122 MPFAIL: .ASCIZ <377>/PWR FAILED. RESTART AT TEST /
(2) 005635 377 047105 020104 MEPASS: .ASCIZ <377>/END PASS DZDMG /
(2) 005657 377 000122 MR: .ASCIZ <377>/R/
(2) 005662 047377 020117 042504 MERR2: .ASCIZ <377>/NO DEVICES PRESENT./
(2) 005707 377 047111 052523 MERR3: .ASCIZ <377>/INSUFFICIENT DATA!/
(2) 005733 377 042524 052123 MTSTPC: .ASCIZ <377>/TEST PC-/
(2) 005745 377 047514 045503 MLOCK: .ASCIZ <377>/LOCK ON SELECTED TEST/
(2) 005774 051503 035122 000040 MCSRX: .ASCIZ /CSR: /
(2) 006002 042526 035103 000040 MVECX: .ASCIZ /VEC: /
(2) 006010 040520 051523 051505 MPASSX: .ASCIZ /PASSES: /
(2) 006021 105 051122 051117 MERRX: .ASCIZ /ERRORS: /
(2) 006032 042524 052123 047040 MTSTN: .ASCIZ /TEST NO: /
(2) 006044 000052 MASTEK: .ASCIZ /*/
(2) 006046 051777 052105 051440 MNEW: .ASCIZ <377>/SET SWITCH REG TO DMC11'S DESIRED ACTIVE..
(2) 006121 120 035103 000040 MERRP: .ASCIZ /PC: /
(2) 006126 020212 020040 020040 XHEAD: .ASCII <212>/ MAP OF DMC11 STATUS/
(2) 006165 377 020040 020040 .ASCII <377>/
(2) 006224 020212 050040 020103 .ASCII <212>/ PC CSR STAT1 STAT2 STAT3/
(2) 006276 026777 026455 026455 .ASCII <377>/-----
(2) 006352 044377 053517 046440 NUM: .ASCIZ <377>/HOW MANY DMC11'S TO BE TESTED?/
(2) 006412 041777 051123 040440 CSR: .ASCIZ <377>/CSR ADDRESS?/
(2) 006430 053377 041505 047524 VEC: .ASCIZ <377>/VECTOR ADDRESS?/
(2) 006451 377 051102 050040 PRIO: .ASCIZ <377>/BR PRIORITY LEVEL? (4,5,6,7)?/
(2) 006510 044777 020106 046504 CRAM: .ASCIZ <377>/IF DMC HAS CRAM (M8204) TYPE "Y", IF CROM (M8200) TYPE "N"
(2) 006606 053777 044510 044103 MODU: .ASCIZ <377>/WHICH LINE UNIT? IF NONE TYPE "N", IF M8201 TYPE "1", IF M
(2) 006720 051777 044527 041524 LINE: .ASCIZ <377>/SWITCH PAC#1 (D0CMP LINE #)?/
(2) 006756 051777 044527 041524 BM: .ASCIZ <377>/SWITCH PAC#2 (BM873 BOOT ADD)?/
(2) 007016 044777 020123 044124 CONN: .ASCIZ <377>/IS THE LOOP BACK CONNECTOR ON?
(2) 007056 047377 020117 042504 NOACT: .ASCIZ <377>/NO DEVICES ARE SELECTED/
(2) 007107 377 051412 051127 SWMES: .ASCIZ <377><12>/SWR= /
(2) 007117 116 053505 020077 SWMES1: .ASCIZ /NEW? /
(2) 007125 377 042377 041515 CONERR: .ASCIZ <377><377>/DMC11 CONFIGURATION ERROR PC: /
(2) 007167 377 054105 042520 CNERR: .ASCIZ <377>/EXPECTED FOUND/
(2) 007210 024040 046504 024503 DMCM: .ASCIZ / (DMC) /
(2) 007220 024040 046513 024503 KMCM: .ASCIZ / (KMC) /
(2) .EVEN
(2) 007230 000005 XSTATQ: 5
1166 007232 006 003 .BYTE 6,3
1167 007234 001246 TEMP1
1168 007236 006 003 .BYTE 6,3
1169 007240 001250 TEMP2
1170 007242 006 003 .BYTE 6,

```

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1171 007244 001252          TEMP3
1172 007246 006          003      .BYTE 6.3
1173 007250 001254          TEMP4
1174 007252 006          002      .BYTE 6.2
1175 007254 001256          TEMP5
1176                                     .EVEN
1177
1178                                     ;BUFFERS FOR INPUT-OUTPUT
1179
1180 007256 000000          INBUF: 3
1181                                     .+.40
1182 007320 000000          MDATA: 0
1183                                     .+.40
1184
1185
1186                                     ;ROUTINE USED TO CHANGE SOFTWARE SWITCH
1187                                     ;REGISTER USING THE CONSOLE TERMINAL
1188 -----
1189
1190 007362 022737 000176 001202          CKSWR: CMP      #SWREG, SWR      ; IS THE SOFT SWR BEING USED?
1191 007370 001071          BNE      CKSWR5          ; BR IF NO
1192 007372 022777 000007 171606          CMP      #7, @TKDBR      ; WAS CTRL G TYPED? (7 BIT ASCII)
1193 007400 001404          BEQ      1$              ; BR IF YES
1194 007402 022777 000207 171576          CMP      #207, @TKDBR    ; WAS CTRL G TYPED? (8 BIT ASCII)
1195 007410 001061          BNE      CKSWR5          ; BR IF NO
1196 007412 010246          1$: MOV     R2, -(SP)        ; STORE R2
1197 007414 010346          MOV     R3, -(SP)        ; STORE R3
1198 007416 010446          MOV     R4, -(SP)        ; STORE R4
1199 007420 012737 177777 007556          MOV     #-1, SWFLG      ; SET SOFT TYPE OUT FLAG
1200 007426 005002          CKSWR1: CLR     R2        ; CLEAR NEW SWR CONTENTS
1201 007430 012704 177777          MOV     #-1, R4         ; SET FLAG TO ALL ONES
1202 007434 104402 007107          TYPE   , SWMES         ; TYPE "SWR="
1203 007440 104411          CKSWR2: CNVRT          ; TYPE OUT PRESENT CONTENTS
1204 007442 007612          SOFTSW          ; OF SOFT SWITCH REGISTER
1205 007444 104402 007117          CKSWR3: TYPE   SWMES1   ; TYPE "NEW"
1206 007450 004737 007560          CKSWR4: JSR     PC, INCHAR ; GET RESPONSE
1207 007454 022703 000015          CMP     #15, R3         ; WAS IT A CR?
1208 007460 001424          BEQ     5$              ; BR IF YES
1209 007462 022703 000012          CMP     #12, R3         ; WAS IT A LF?
1210 007466 001416          BEQ     4$              ; BR IF YES
1211 007470 022703 000025          CMP     #25, R3         ; WAS IT CTRL U?
1212 007474 001754          BEQ     CKSWR1          ; BR IF YES (START OVER)
1213 007476 022703 000007          CMP     #7, R3          ; IF CNTL G GET NEXT CHAR
1214 007502 001762          BEQ     CKSWR4
1215 007504 005004          CLR     R4              ; IT MUST BE A DIGIT SO CLR FLAG
1216 007506 042703 177770          BIC     #177770, R3     ; ONLY 0-7 ARE LEGAL SO MASK OFF BITS
1217 007512 006302          ASL     R2              ; SHIFT R2 3 TIMES
1218 007514 006302          ASL     R2
1219 007516 006302          ASL     R2
1220 007520 050302          BIS     R3, R2          ; ADD LAST DIGIT
1221 007522 000752          BR      CKSWR4          ; GET NEXT CHARACTER
1222 007524 012766 002002 000006          4$: MOV     #.START.6(SP) ; LF WAS TYPED SO GO TO START
1223 007532 005704          5$: TST     R4           ; IS FLAG CLEAR?
1224 007534 001002          BNE     6$              ; IF NOT DON'T CHANGE SOFT SWR
1225 007536 010277 171440          MOV     R2, @SWR        ; IF YES THEN WRITE NEW CONTENTS TO SOFT SWR
1226 007542 005037 007556          6$: CLR     SWFLG       ; CLEAR TYPEOUT FLAG

```

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1227	007546	012604		MOV	(SP)+,R4	:RESTORE R4
1228	007550	012603		MOV	(SP)+,R3	:RESTORE R3
1229	007552	012602		MOV	(SP)+,R2	:RESTORE R2
1230	007554	000207		CKSWRS: RTS	PC	:RETURN
1231						
1232	007556	000000		SWFLG:	0	
1233						
1234	007560	105777	171420	INCHAR: TSTB	@TKCSR	
1235	007564	100375		BPL	.-4	
1236	007566	017703	171414	MOV	@TKDBR,R3	
1237	007572	105777	171412	TSTB	@TPCSR	
1238	007576	100375		BPL	.-4	
1239	007600	010377	17:406	MOV	R3,@TPDBR	
1240	007604	042703	003200	BIC	#BIT7,R3	
1241	007610	000207		RTS	PC	
1242						
1243	007612	000001		SOFTSW:	!	
1244	007614	000006	002	.BYTE	5,2	
1245	007616	000006		SWREG		

:ROUTINE USED TO "CYCLE" THROUGH UP TO 16 DMC11'S
:THIS ROUTINE SETS UP THE CONTROL ADDRESS FOR THE DIAGNOSTIC
:AND RUNS THE SPECIFIED DMC11'S. THIS ROUTINE *MUST*
:BE RUN FIRST BEFORE ENTERING THE DIAGNOSTIC FOR THE
:SETUP NECESSARY.
:

1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301

007620 005737 001306
007624 001004
007626 104402 007056
007632 000000
007634 000776
007636 000241
007640 006137 001316
007644 005537 001316
007650 062737 000004 001322
007656 062737 000010 001320
007664 022737 001700 001320
007672 001006
007674 012737 001500 001320
007702 012737 001702 001322
007710 033737 001316 001306
007716 001747
007720 013700 001320
007724 013702 001322
007730 012037 001404
007734 011037 001374
007740 042737 177000 001374
007746 012037 001366
007752 012037 001370
007756 012037 001372
007762 012237 001230
007766 012237 001232
007772 012700 000002
007776 013737 001404 001406
010004 005237 001406
010010 013737 001406 001410
010016 005237 001410
010022 013737 001410 001412
010030 060037 001412
010034 013737 001412 001414
010042 060037 001414
010046 013737 001374 001376
010054 060037 001376
010060 013737 001376 001400
010066 060037 001400
010072 013737 001400 001402
010100 060037 001402
010104 032737 000002 001236
010112 001450
010114
010114 005737 000042

CYCLE:
1S:
2S:
4S:

TST DMACTV
BNE 1S
TYPE .NOACT
HALT
BR -2
CLC
ROL RUN
ADC RUN
ADD #4,MILK
ADD #10,CREAM
CMP #DM.MAP+200,CREAM
BNE 2S
MOV #DM.MAP,CREAM
MOV #CNT.MAP,MILK
BIT RUN,DMACTV
BEQ 1S
MOV CREAM,R0
MOV MILK,R2
MOV (R0)+,DMCSR
MOV (R0),DMRVEC
BIC #177000,DMRVEC
MOV (R0)+,STAT1
MOV (R0)+,STAT2
MOV (R0)+,STAT3
MOV (R2)+,PASCNT
MOV (R2)+,ERRCNT
MOV #2,R0
MOV DMCSR,DMCSRH
INC DMCSRH
MOV DMCSRH,DMCTL
INC DMCTL
MOV DMCTL,DMP04
ADD R0,DMP04
MOV DMP04,DMP06
ADD R0,DMP06
MOV DMRVEC,DMRLVL
ADC R0,DMRLVL
MOV DMRLVL,DMTVEC
ADD R0,DMTVEC
MOV DMTVEC,DMTLVL
ADD R0,DMTLVL
BIT #SW01,STATSW
BEQ 7S
TST #42

:ARE ANY DMC11'S TO BE TESTED?
:BR IF OK.
:NO DMC11'S SELECTED!!
:STOP THE SHOW.
:DISQUALIFY CONT. SW.
:CLEAR PROC. CARRY BIT.
:UPDATE POINTER
:CATCH CARRY FROM RUN
:UPDATE POINTER
:UPDATE ADDRESS POINTER.
:KEEP GOING; NOT ALL TESTED FOR.
:RESET ADDRESS POINTER.
:RESET PASS COUNT POINTER
:IS THIS ONE ACTIVE?
:BR IF NO
:GET ADDRESS POINTER
:GET PASS COUNT POINTER
:LOAD SYSTEM CTRL. REG
:LOAD VECTOR
:CLEAR UNWANTED BITS
:LOAD STAT1
:LOAD STAT2
:LOAD STAT3
:LOAD PASS COUNT
:LOAD ERROR COUNT
:SAVE CORE THIS WAY!
:PTY LVL
:TX VEC
:TX LVL
:IS TEST NO. SELECTED
:BR IF NO
:RUNNING IN AUTO MODE"

K04

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1302	010120	001045			BNE	7\$:BR IF YES	
1303	010122	104402	005574		TYPE	.MCR.LF		
1304	010126	104403			INSTR		:GET TEST NO.	
1305	010130	006032			MTSTN			
1306	010132	104405			PARAM			
1307	010134	000001			1			
1308	010136	001000			1000			
1309	010140	001226			TSTNO			
1310	010142	000			.BYTE	0		
1311	010143	001			.BYTE	1		
1312	010144	012700	015766		MOV	#TST1,R0		
1313	010150	022710		5\$:	CMR	(PC)+,(R0)	:CMP FIRST WORD TO 12737	
1314	010152	012737			MOV	(PC)+,@(PC)+		
1315	010154	001020			BNE	6\$:BR IF NOT SAME	
1316	010156	023760	001226	000002	CMR	TSTNO.2(R0)	:DOES TSTNO MATCH?	
1317	010164	001014			BNE	6\$:BR IF NO	
1318	010166	022760	001226	000004	CMR	#TSTNO.4(R0)	:IS LAST WORD OK?	
1319	010174	001010			BNE	6\$:BR IF NO	
1320	010176	010037	001214		MOV	R0,RETURN	:IT IS A LEGAL TEST SO DO IT	
1321	010202	104402	005657		TYPE	.MR		
1322	010206	042737	000002	001236	BIC	#SW01,STRTSW		
1323	010214	000412			BR	8\$		
1324	010216	005720		6\$:	TST	(R0)+	:POP R0	
1325	010220	020027	031442		CMR	R0,#TLAST+10	:AT END YET?	
1326	010224	001351			BNE	5\$:BR IF NO	
1327	010226	104402	005570		TYPE	.MQM	:YES ILLEGAL TEST NO.	
1328	010232	000730			BR	4\$:TRY AGAIN	
1329								
1330	010234	012737	015766	001214	7\$:	MOV	#TST1,RETURN	:PREPARE RETURN ADDRESS
1331	010242	013701	001404		8\$:	MOV	DMCSR,R1	:R1 = BASE DMC11 ADDRESS
1332	010246	000177	170742		JMP	@RETURN	:GO START TESTING.	
1333								
1334								
1335								
1336								
1337								
1338								
1339								
1340								
1341								
1342								
1343	010252							
1344	010252	000005			AUTO.SIZE:	RESET	:INSURE A BUS INIT.	
1345	010254	012702	001500		CSRMAP:	MOV	#DM.MAP,R2	:LOAD MAP POINTER.
1346	010260	005022		1\$:	CLR	(R2)+	:ZERO ENTIRE MAP	
1347	010262	022702	001700		CMR	#DM.END,R2	:ALL DONE?	
1348	010266	001374			BNE	1\$:BR IF NO	
1349	010270	005037	001310		CLR	DMNUM	:SET OCTAL NUMBER OF DMC11'S TO 3	
1350	010274	012702	001500		MOV	#DM.MAP,R2	:R2 POINTS TO DMC MAP	
1351	010300	005037	001306		CLR	DMACTV	:CLEAR ACTIVE	
1352	010304	032737	000001	001236	BIT	#SW00,STRTSW	:QUESTIONS?	
1353	010312	001002			BNE	.+6	:BR IF YES	
1354	010314	000137	010744		JMP	7\$:IF NO SKIP QUESTIONS	
1355	010320	012737	000001	001256	MOV	#1,TEMP5	:START WITH 1	
1356	010326	104403			INSTR			
1357	010330	006352			NUM			

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1358	010332	104405			PARAM		
1359	010334	0000C1			1		
1360	010336	000020			16.		
1361	010340	001252			TEMP3		
1362	010342	000			.BYTE	0	
1363	010343	001			.BYTE	1	
1364	010344	013737	001252	001310	MOV	TEMP3, DMNUM	: DMNUM = HOW MANY
1365	010352	104402	005574		TYPE	.MCRLF	
1366	010356	104410		12\$:	CONVRT		: TYPE WHICH DMC IS BEING DONE
1367	010360	011450			WHICH		: TEMPS IS WHICH DMC
1368	010362	005237	001256		INC	TEMPS	
1369	010366	104403			INSTR		
1370	010370	006412			CSR		
1371	010372	104405			PARAM		
1372	010374	160000			160000		
1373	010376	164000			164000		
1374	010400	001254			TEMP4		
1375	010402	000			.BYTE	0	
1376	010403	001			.BYTE	1	
1377	010404	013722	001254		MOV	TEMP4, (R2)+	: STORE CSR IN MAP
1378	010410	104403			INSTR		
1379	010412	006430			VEC		
1380	010414	104405			PARAM		
1381	010416	000000			0		
1382	010420	000776			776		
1383	010422	001254			TEMP4		
1384	010424	000			.BYTE	0	
1385	010425	001			.BYTE	1	
1386	010426	013712	001254		MOV	TEMP4, (R2)	: STORE VECTOR IN MAP
1387	010432	104402		10\$:	TYPE		
1388	010434	006451			PRIO		: ASK WHAT BR LEVEL
1389	010436	004737	011734		JSR	PC, INTTY	: GET RESPONSE
1390	010442	022703	000024		CMP	#24, R3	
1391	010446	101014			BHI	50\$: BR IF LESS THAN 4
1392	010450	022703	000027		CMP	#27, R3	
1393	010454	103411			BLO	50\$: BR IF GREATER THAN 7
1394	010456	012704	000011		MOV	#11, R4	: R4 = NUMBER OF SHIFTS
1395	010462	006303			ASL	R3	: SHIFT R3 LEFT
1396	010464	005304			DEC	R4	: DEC SHIFT COUNT
1397	010466	001375			BNE	.-4	: BR IF NOT DONE
1398	010470	042703	170777		BIC	#170777, R3	: BIC UNWANTED BITS
1399	010474	050312			BIS	R3, (R2)	: PUT BR LEVEL IN STATUS MAP
1400	010476	000403			BR	8\$: CONTINUE
1401	010500	104402		50\$:	TYPE		
1402	010502	005570			MQM		: RESPONSE IS OUT OF LIMITS
1403	010504	000752			BR	10\$: TRY AGAIN
1404	010506	104402		8\$:	TYPE		
1405	010510	006510			CRAM		: DOES DMC HAVE CRAM?
1406	010512	004737	011734		JSR	PC, INTTY	: GET REPLY
1407	010516	022703	000131		CMP	#131, R3	
1408	010522	001406			BEQ	9\$: YES
1409	010524	022703	000116		CMP	#116, R3	: NO
1410	010530	001405			BEQ	16\$: NOT A Y OR N
1411	010532	104402			TYPE		
1412	010534	005570			MQM		: TYPE ""
1413	010536	000763			BR	8\$: ASK AGAIN

M04

DZDMG MACY11 27(1006) 07-MAR-77 15:52 PAGE 31
 DZDMGB.P11 07-MAR-77 11:21

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

PAGE: 0051

1414	010540	052712	100000		9\$: BIS	#BIT15,(R2)	;SET BIT 15 IF CRAM
1415	010544	104402			15\$: TYPE		
1416	010546	006606			MODU		;ASK WHICH LINE UNIT
1417	010550	004737	011734		JSR	PC,INTTY	;GET REPLY
1418	010554	022703	000021		CMP	#21,R3	;"1"
1419	010560	001417			BEQ	30\$	
1420	010562	022703	000022		CMP	#22,R3	;"2"
1421	010566	001412			BEQ	31\$	
1422	010570	022703	000116		CMP	#11b,R3	;"N"
1423	010574	001403			BEQ	32\$	
1424	010576	104402			TYPE		
1425	010600	005570			MQM		;IF NOT A 1,2 OR N TYPE "?"
1426	010602	000760			BR	16\$;TRY AGAIN
1427	010604	052722	010000		32\$: BIS	#BIT12,(R2)+	;SET BIT 12 IN STAT2 IF NO LU
1428	010610	022222			CMP	(R2)+,(R2)+	;POP OVER STAT2 AND STAT3
1429	010612	000447			BR	33\$	
1430	010614	052712	020000		31\$: BIS	#BIT13,(R2)	;SET BIT 13 IN STAT2 IF M8202
1431	010620	104402			30\$: TYPE		
1432	010622	007016			CONN		;ASK IF LOOP-BACK IS ON
1433	010624	004737	011734		JSR	PC,INTTY	;GET REPLY
1434	010630	022703	000131		CMP	#131,R3	;Y
1435	010634	001406			BEQ	17\$	
1436	010636	022703	000116		CMP	#116,R3	;N
1437	010642	001406			BEQ	18\$	
1438	010644	104402			TYPE		
1439	010646	005570			MQM		;IF NOT Y OR N TYPE "?"
1440	010650	000763			BR	30\$;TRY AGAIN
1441	010652	052722	040000		17\$: BIS	#BIT14,(R2)+	;TURNAROUND IS CONNECTED
1442	010656	000402			BR	19\$	
1443	010660	042722	040000		18\$: BIC	#BIT14,(R2)+	;NO TURNAROUND
1444	010664				19\$: INSTR		
1445	010664	104403			LINE		
1446	010666	006720			PARAM		
1447	010670	104405			0		
1448	010672	000000			377		
1449	010674	000377			TEMP4		
1450	010676	001254			.BYTE	0	
1451	010700	000			.BYTE	1	
1452	010701	001			MOV	TEMP4,(R2)+	;STORE SWITCH PAC IN MAP
1453	010702	113722	001254		INSTR		
1454	010706	104403			BM		
1455	010710	006756			PARAM		
1456	010712	104405			0		
1457	010714	000020			377		
1458	010716	000377			TEMP4		
1459	010720	001254			.BYTE	0	
1460	010722	000			.BYTE	1	
1461	010723	001			MOV	TEMP4,(R2)+	;STORE SWITCH PAC IN MAP
1462	010724	113722	001254		TST	(R2)+	;POP OVER STAT3
1463	010730	005722			33\$: DEC	TEMP3	;DEC DMC COUNT
1464	010732	005337	001252		BR	12\$;BR IF MORE TO DO
1465	010736	001205			CONT	13\$;CONTINUE
1466	010740	000137	011350		7\$: MOV	#160000,R1	;SET FOR FIRST ADDRESS TO BE TESTED
1467	010744	012701	160000		MOV	#6\$,R4	;SET FOR NON-EXISTANT DEVICE TIME OUT
1468	010750	012737	011442	000004	2\$: CLR	(R1)	;CLEAR SEL0
1469	010756	005011					

1470	010760	005711			TST	(R1)	; IF DMC11 DMCSR S/B C
1471	010762	001162			BNE	3\$; IF NO DEV ; TRAP TO 4. IF NO BIT 9 THEN NO DMC1
1472	010764	005061	000006		CLR	6(R1)	; CLEAR SEL6
1473	010770	005761	000006		TST	6(R1)	; IF DMC11 THEN DMPIC S/B =0!
1474	010774	001155			BNE	3\$; BR IF NOT DMC11
1475	010776	012711	002000		MOV	#BIT10,(R1)	; SET ROMO
1476	011002	005061	000004		CLR	4(R1)	; CLEAR SEL4
1477	011006	012761	125252	000006	MOV	#125252,6(R1)	; WRITE THIS TO SEL6
1478	011014	052711	020000		BIS	#BIT13,(R1)	; WRITE IT!
1479	011020	022761	125252	000004	CMP	#125252,4(R1)	; WAS IT WRITTEN?
1480	011026	001004			BNE	21\$; IF NO IT IS NOT CRAM
1481	011030	052762	100000	000002	BIS	#BIT15,2(R2)	; SET BIT15 IF CRAM
1482	011036	000421			BR	22\$	
1483	011040	012711	001000		MOV	#BIT9,(R1)	; SET ROMI
1484	011044	012761	100400	000006	MOV	#100400,6(R1)	; PUT INSTRUCTION IN SEL6
1485	011052	012711	001400		MOV	#BIT9:BIT9,(R1)	; CLOCK INSTRUCTION (MICRO PROC PC TO 0)
1486	011056	012711	002000		MOV	#BIT10,(R1)	; SET ROMO
1487	011062	022761	063220	000006	CMP	#63220,6(R1)	; IS IT CROM
1488	011070	001404			BEQ	22\$; BR IF YES
1489	011072	022761	177777	000006	CMP	#-1,6(R1)	; IF = -1 IT HAS NO CROM
1490	011100	001113			BNE	3\$; BR IF NOT DMC11
1491							; AT THIS POINT IT IS ASSUMED THAT R1 HOLDS A DMC11 CSR ADDRESS.
1492	011102	010122			MOV	R1,(R2)+	; STORE CSR IN CORE TABLE.
1493	011104	012711	001000		MOV	#BIT9,(R1)	; CLEAR LINE UNIT LOOP
1494	011110	005061	000004		CLR	4(R1)	; CLEAR PORT4
1495	011114	012761	122113	000006	MOV	#122113,6(R1)	; LOAD INSTRUCTION (CLR DTR)
1496	011122	052711	000400		BIS	#BIT8,(R1)	; CLOCK INSTRUCTION
1497	011126	012761	021264	000006	MOV	#021264,6(R1)	; LOAD INSTRUCTION
1498	011134	052711	000400		BIS	#BIT8,(R1)	; CLOCK INSTRUCTION
1499	011140	122761	000377	000004	CMPB	#377,4(R1)	; IS IT ALL ONES?
1500	011146	001003			BNE	.+10	; BR IF NO
1501	011150	052712	010000		BIS	#BIT12,(R2)	; IF YES, NO LINE UNIT. SET STATUS BIT
1502	011154	000436			BR	20\$	
1503	011156	032761	000002	000004	BIT	#BIT1,4(R1)	; IS SWITCH A ONE?
1504	011164	001403			BEQ	.+10	; BR IF M8201
1505	011166	052712	060000		BIS	#BIT13:BIT14,(R2)	; M8202 ASSUME CONNECTOR
1506	011172	000427			BR	20\$; CONNECTOR ON)
1507	011174	032761	000010	000004	BIT	#BIT3,4(R1)	; IS MRDY SET
1508	011202	001023			BNE	20\$; BR IF M8201 NO CONNECTOR (ON LINE)
1509	011204	012761	000100	000004	MOV	#BIT6,4(R1)	; LOAD PORT4
1510	011212	012761	122113	000006	MOV	#122113,6(R1)	; LOAD INSTRUCTION
1511	011220	052711	000400		BIS	#BIT8,(R1)	; CLOCK INSTRUCTION (SET DTR)
1512	011224	012761	021264	000006	MOV	#021264,6(R1)	; LOAD INSTRUCTION
1513	011232	052711	000400		BIS	#BIT8,(R1)	; CLOCK INSTRUCTION (READ MODEM REG.
1514	011236	032761	000010	000004	BIT	#BIT3,4(R1)	; IS MRDY SET NOW?
1515	011244	001402			BEQ	20\$; BR IF NO CONNECTOR
1516	011246	052712	040000		BIS	#BIT14,(R2)	; SET STATUS BIT FOR CONNECTOR
1517	011252	005722			TST	(R2)+	; POP POINTER
1518	011254	012761	021324	000006	MOV	#021324,6(R1)	; PUT INSTRUCTION IN PORT6
1519	011262	012711	001400		MOV	#BIT9:BIT8,(R1)	; PORT6+LU 15
1520	011266	156122	000004		BISB	4(R1),(R2)+	; STORE DDCMP LINE # IN TABLE
1521	011272	012761	021344	000006	MOV	#021344,6(R1)	; PORT6+INSTRUCTION
1522	011300	012711	001400		MOV	#BIT8:BIT9,(R1)	; CLOCK INSTR.
1523	011304	156122	000004		BISB	4(R1),(R2)+	; STORE BM873 ADD IN TABLE
1524	011310	005722			TST	(R2)+	; POP OVER STAT3
1525	011312	005011			CLR	(R1)	; CLEAR ROMI

```

1526 011314 005237 001310      INC      DMNUM      ;UPDATE DEVICE COUNTER
1527 011320 022737 000020 001310    CMP      #20,DMNUM  ;ARE MAX. NO. OF DEV FOUND?
1528 011326 001410      BEQ      13$      ;YES DON'T LOOK FOR ANY MORE.
1529 011330 005011      3$:     CLR      (R1)   ;CLEAR BIT 10
1530 011332 005061 000006    CLR      6(R1)   ;CLEAR SEL 6
1531 011336 062701 000010    14$:   ADD      #10,R1 ;UPDATE CSR POINTER ADDRESS
1532 011342 022701 164000    CMP      #164000,R1
1533 011346 001203      BNE      2$      ;BR IF MORE ADDRESS TO CHECK.
1534 011350 005037 001306    13$:   CLR      DMACTV
1535 011354 005737 001310    TST      DMNUM
1536 011360 001423      BEQ      5$      ;WERE ANY DMC11'S FOUND AT ALL?
1537 011362 013701 001310    MOV      DMNUM,R1 ;ERROR AUTO SIZER FOUND NO DMC11'S IN THIS SYS.
1538 011366 010137 001314    MOV      R1,SAVNUM ;SAVE NUMBER OF DEVICES
1539 011372 000241      4$:   CLC
1540 011374 006137 001306    ROL      DMACTV   ;GENERATE ACTIVE REGISTER OF DEVICES.
1541 011400 005237 001306    INC      DMACTV   ;SET THE BIT
1542 011404 005301      DEC      R1
1543 011406 001371      BNE      4$      ;BR IF MORE TO GENERATE
1544 011410 012737 000006 000004    MOV      #6,2#4  ;RESTORE TRAP VECTOR
1545 011416 013737 001306 001312    MOV      DMACTV,SAVACT ;SAVE ACTIVE REGISTER
1546 011424 000137 011456    JMP      VECMAP   ;GO FIND THE VECTOR NOW.
1547 011430 104402 005662    5$:   TYPE      MERR2 ;NOTIFY OPR THAT NO DMC11'S FOUND.
1548 011434 005000      CLR      R0      ;MAKE DATA LIGHTS ZERO
1549 011436 000000      HALT
1550 011440 000776      BR      -2       ;STOP THE SHOW
1551 011442 012716 011336    6$:   MOV      #-2,SP ;DISABLE CONT. SW.
1552 011446 000002      RTI            ;ENTERED BY NON-EXISTANT TIME-OUT.
1553
1554 011450 000001      WHICH: 1       ;RETURN TO MAINSTREAM
1555 011452 002 002      .BYTE 2.2
1556 011454 001256      TEMPS
1557
1558 011456 032737 000001 001236    VECMAP: BIT      #SW00,STRTSW
1559 011464 001114      BNE      5$
1560 011466 012737 000340 000022    MOV      #340,2#22 ;SET IOT TRAP PRIO TO 7
1561 011474 012737 011650 000020    MOV      #45,2#20  ;SET IOT TRAP VECTOR
1562 011502 012702 001500      MOV      #DM.MAP,R2 ;SET SOFTWARE POINTER
1563 011506 012700 000300      MOV      #300,R0   ;FLOATING VECTORS START HERE.
1564 011512 012701 000302      MOV      #302,R1   ;PC OF IOT INSTR.
1565 011516 010120    1$:   MOV      R1,(R0)+ ;START FILLING VECTOR AREA
1566 011520 012721 000004      MOV      #4,(R1)+ ;WITH .+2; IOT
1567 011524 022021      CMP      (R0)+,(R1)+ ;ADD 2 TO R0 +R1
1568 011526 020127 001000      CMP      R1,#1000
1569 011532 101771      BLOS    1$      ;BR IF MORE TO FILL
1570 011534 013737 001306 001246    MOV      DMACTV,TEMP1 ;STORE TEMPORALLY
1571 011542 006037 001246    2$:   ROR      TEMP1  ;BRING OUT A BIT
1572 011546 103063      BCC      5$      ;BR IF ALL DONE
1573 011550 012704 000012      MOV      #12,R4   ;R4 IS INDEX REGISTER
1574 011554 016437 011720 177776    MOV      BRLVL(R4),PS ;SET PS TO 7
1575 011562 011201      MOV      (R2),R1
1576 011564 012761 000200 000004    MOV      #200,4(R1)
1577 011572 012711 001000      MOV      #BIT9,(R1) ;SET ROMI
1578 011576 012761 121111 000006    MOV      #121111,6(R1) ;PUT INSTRUCTION IN PORT6
1579 011604 012711 001400      MOV      #BIT9!BIT8,(R1) ;FORCE AN INTERRUPT
1580 011610 105200    7$:   INCB    R0      ;STALL
1581 011612 001376      BNE      -2     ;FOR TIME TO INTERUPT

```


GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1582 011614 162704 000002          SUB      #2,R4          ;GET NEXT LOWEST PS LEVEL
1583 011620 001404          BEQ      6$           ;BR IF R4 = 0
1584 011622 016437 011720 177776    MOV      BRLVL(R4),PS ;MOVE NEXT LOWER LEVEL IN PS
1585 011630 000767          BR       7$           ;BR TO DELAY
1586 011632 052762 005300 000002    6$:     BIS      #5300,2(R2) ;NO INTERRUPT ASSUME 300 AT LEVEL 5 AND FIX DMC11
1587 011640 005011          3$:     CLR      (R1)      ;CLEAR ROMI
1588 011642 062702 000010          ADD      #10,R2       ;POP SOFTWARE POINTER
1589 011646 000735          BR       2$           ;KEEP GOING
1590 011650 051662 000002          4$:     BIS      (SP),2(R2) ;GET VECTOR ADDRESS
1591 011654 042762 000007 000002    BIC      #7,2(R2)     ;CLEAR JUNK
1592 011662 016405 011722          MOV      BRLVL+2(R4),R5 ;GET BR LEVEL OF DMC11
1593 011666 006305          ASL      R5           ;SHIFT LEVEL 4 PLACES
1594 011670 006305          ASL      R5           ;TO THE LEFT FOR THE
1595 011672 006305          ASL      R5           ;STATUS TABLE
1596 011674 006305          ASL      R5
1597 011676 042705 170777    BIC      #170777,R5   ;CLEAR UNWANTED BITS
1598 011702 050562 000002    BIS      R5,2(R2)     ;PUT BR LEVEL IN STATUS TABLE
1599 011706 022626          CMP      (SP)+,(SP)+ ;POP IOT JUNK OFF STACK
1600 011710 012716 011640    MOV      #3$, (SP)   ;SET FOR RETURN
1601 011714 000002          RTI
1602 011716 000207          5$:     RTS      PC      ;ALL DONE WITH "AUTO SIZING"
1603
1604 011720 000000          BRLVL:  0             ;LEVEL 0
1605 011722 000000          0             ;LEVEL 0
1606 011724 000200          200            ;LEVEL 4
1607 011726 000240          240            ;LEVEL 5
1608 011730 000300          300            ;LEVEL 6
1609 011732 000340          340            ;LEVEL 7
1610
1611
1612 011734 105777 167244          INTTY:  TSTB     @TKCSR ;WAIT FOR DONE
1613 011740 100375          BPL     -4
1614 011742 017703 167240          MOV     @TKDBR,R3    ;PUT CHAR IN R3
1615 011746 105777 167236          TSTB   @TPCSR       ;WAIT UNTIL PRINTER IS READY
1616 011752 100375          BPL     -4
1617 011754 010377 167232          MOV     R3,@TPDBR   ;ECHO CHAR
1618 011760 042703 000240          BIC     #BIT7!BITS,R3 ;MASK OFF LOWER CASE
1619 011764 000207          RTS     PC           ;RETURN
1620
1621
1622 011766          15300
          15400  ROMMAP:

```

4	MACRO DEFINITIONS
6	REVISION 00
7	FEBRUARY 25, 1975
8	
9	REVISION 01
10	MARCH 18, 1975
11	NEW CSR BOARD CHANGES
12	
13	HARVEY M. SCHLESINGER
15	COPYRIGHT 1975 DIGITAL EQUIPMENT CORPORATION
67	MICRO INSTRUCTION DEFINITIONS
68	BRANCH INSTRUCTIONS
119	INDEXED BRANCH INSTRUCTIONS
162	MOVE INSTRUCTIONS
284	INPUT/OUTPUT ASSIGNMENTS
336	PROTOCOL DEPENDANT MACROS
379	DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT
387	VERSION 00A FEBRUARY 26, 1975
388	
389	HARVEY M. SCHLESINGER
390	
391	COPYRIGHT 1975, DIGITAL EQUIPMENT CORPORATION
392	
393	VERSION 00B MARCH 17, 1975
394	CSR AND MICROPROCESSOR CHANGES
395	
396	VERSION 00C NOVEMBER 6, 1975
397	RETRANSMISSION CHANGES
398	
399	VERSION 00D DECEMBER 3, 1975
400	TRANSMIT DONE CHANGES
401	
402	THE LATEST MODIFICATIONS WERE ADDED ON:
403	OCTOBER 13, 1976
404	THIS VERSION WAS USED TO BLAST THE FIRST
405	RELEASE ON OCTOBER 13, 1975
407	MICROPROCESSOR MAIN MEMORY ASSIGNMENTS
469	SCRATCH PAD ASSIGNMENTS
504	INIT--INITIALIZATION ROUTINE
559	IDLE--PROGRAM IDLE LOOP
575	BASSRV---- BASE SERVICE ROUTINE
614	NIDLE2---NO CSR ACTIVITY STATE
663	INWAIT---WAIT FOR RQI TO CLEAR
718	OUTINT---SET UP OUTPUT INTERRUPT [RDY0]
766	OUTWAI---WAIT FOR RDY0 TO GO AWAY
773	CTLSRV--CNTL I SERVICE
794	TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE
816	RBASRV--RECEIVE BUFFER ADDRESS SERVICE
898	RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER
9.8	RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD
557	RCVC--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD, SELECT AND FINAL
976	RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES
1000	RCVE--ROUTINE TO HANDLE N FIELD OF NUMBERED MESSAGE
1013	RCVF--ROUTINE TO IGNORE ADDRESS
1018	RCVG--ROUTINE TO IGNORE CRC1
1023	RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYPES

E05

DMS11 DDMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 06-DEC-76 10:14

MACY11 27(1006) 17-MAR-77 16:24
TABLE OF CONTENT

PAGE: 0056

1082 RCVK01--ROUTINE TO HANDLE FIRST BYTE ODD RECEIVE
1089 RCVK0--PROCESS ODD CHARACTER
1119 RCVKE--HANDLE EVEN BYTES
1166 RCVI--STORE UNNUMBERED MESSAGE TYPE
1172 RCVJ--ROUTINE TO HANDLE SUBTYPE FIELD, SELECT AND FINAL
1177 RCVR--UNNUMBERED MESSAGE RESPONSE FIELD
1187 RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD
1193 RCVL--PROCESS CRC3
1218 RCVM--PROCESS CRC4--END OF DATA MESSAGE
1240 EM2--PROCESS RL MESSAGE
1277 TMTDA--TRANSMITTER DISPATCH ROUTINE
1283 TMTA--FIRST CHARACTER OF HEADER
1330 TMTB--OUTPUT FIRST CHAR OF COUNT
1356 TMTC--OUTPUT SECOND CHAR OF COUNT
1373 TMTD--RESPONSE FIELD-NUMBERED MESSAGE
1393 TMT E--NUMBER FIELD--NUMBERED MESSAGE
1402 TMTF--NUMBERED MSG ADDRESS FIELD
1415 TF1-NUMBERED MSG HEADER EOM
1425 TMT H--ROUTINE TO OUTPUT DATA CHARACTERS
1480 TMTI--SEND UNNUMBERED TYPE FIELD
1486 TMTJ--SEND SUB-TYPE FIELD
1493 TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG)
1501 TMTL--UNNUMB MSG NUMBER FIELD
1519 TMTM--UNNUMB MSG--STATION ADDRESS
1535 TIMSRV--TIMEOUT ROUTINE--SENDS REP
1596 SNDACK--ROUTINE TO SEND AN ACK
1653 REP HANDLER
1662 START HANDLER
1675 STACK HANDLER
1711 NXMERR ---NON EXISTANT MEMORY HANDLER
1719 SELQSY--ROUTINE TO CHECK SELECT AND QSYNC AND DIDDLE LINE STATUS WORD

DMC-11 DCOMP PROTOCOL IMPLEMENTATION
DMCNEW.MAC 02-DEC-76 09:40

```

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

```

G05

000000
000001
000002
000003
000004
000005
000006
000007
000008
000009
000010
000011
000012
000013
000014
000015
000016
000017
000018
000019
000020
000021
000022
000023
000024
000025
000026
000027
000028
000029
000030
000031
000032
000033
000034
000035
000036
000037
000038
000039
000040
000041
000042
000043
000044
000045
000046
000047
000048
000049
000050
000051
000052
000053
000054
000055
000056
000057
000058
000059
000060
000061
000062
000063
000064
000065
000066
000067
000068
000069
000070
000071
000072
000073
000074
000075
000076
000077
000078
000079
000080
000081
000082
000083
000084
000085
000086
000087
000088
000089
000090
000091
000092
000093
000094
000095
000096
000097
000098
000099

```
000000
000001
100000
020000
000000
040000
060000
060000
060000
010000
014000
000400
001000
001400
002000
002400
003000
003400
000200
000220
000240
000260
000300
000320
000340
000360
000000
000020
000040
000060
000100
000120
000140
000160
004000
010000
014000
001000
001400
000400
002000
002400
003000
003400
```

```

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

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

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

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

CCOND=1000      : CONDITION C
ZCOND=1400      : CONDITION Z
ALCOND=400      : ALWAYS
BRGCON=2000     : CONDITION BR0
BR1CON=2400     : CONDITION BR1
BR4CON=3000     : CONDITION BR4
BR7CON=3400     : CONDITION BR7

```

67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122

100000

```
.SBTTL MICRO INSTRUCTION DEFINITIONS
.SBTTL BRANCH INSTRUCTIONS

:
JUMP=100000 ;JUMP OP CODE

:
.MACRO ALWAYS ADDRES ;JUMP ALWAYS
MICPC=MICPC+1
<JUMP!ALCOND!<ADDRES-INIT&3000*4>!<ADDRES-INIT&777/2>
.ENDM

:
.MACRO $ZERO
MICPC=MICPC+1
000000
.ENDM $ZERO

:
.MACRO BRO ADDRES ;JUMP IF BRO SET
MICPC=MICPC+1
<JUMP!BROCON!<ADDRES-INIT&3000*4>!<ADDRES-INIT&777/2>
.ENDM

:
.MACRO BR1 ADDRES ;JUMP IF BR1 SET
MICPC=MICPC+1
<JUMP!BR1CON!<ADDRES-INIT&3000*4>!<ADDRES-INIT&777/2>
.ENDM

:
.MACRO BR4 ADDRES ;JUMP IF BR4 SET
MICPC=MICPC+1
<JUMP!BR4CON!<ADDRES-INIT&3000*4>!<ADDRES-INIT&777/2>
.ENDM

:
.MACRO BR7 ADDRES ;JUMP IF BR7 SET
MICPC=MICPC+1
<JUMP!BR7CON!<ADDRES-INIT&3000*4>!<ADDRES-INIT&777/2>
.ENDM

:
.MACRO Z ADDRES ;JUMP IF Z BIT SET
MICPC=MICPC+1
<JUMP!ZCOND!<ADDRES-INIT&3000*4>!<ADDRES-INIT&777/2>
.ENDM

:
.MACRO C ADDRES ;JUMP IF C BIT SET
MICPC=MICPC+1
<JUMP!CCOND!<ADDRES-INIT&3000*4>!<ADDRES-INIT&777/2>
.ENDM

:
.SBTTL INDEXED BRANCH INSTRUCTIONS

:
.MACRO ,ALWAY SRC, FUNC, SPLOC ;INDEXED JUMP ALWAYS
MICPC=MICPC+1
```


123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178

000000

```

< JUMP!ALCOND!SRC!FUNC!SPLOC >
.ENDM
:
.MACRO .BR0 SRC, FUNC, SPLOC ; INDEXED JUMP ON BR0 SET
MICPC=MICPC+1
< JUMP!BR0CON!SRC!FUNC!SPLOC >
.ENDM
:
.MACRO .BR1 SRC, FUNC, SPLOC ; INDEXED JUMP ON BR1 SET
MICPC=MICPC+1
< JUMP!BR1CON!SRC!FUNC!SPLOC >
.ENDM
:
.MACRO .BR4 SRC, FUNC, SPLOC ; INDEXED JUMP ON BR4 SET
MICPC=MICPC+1
< JUMP!BR4CON!SRC!FUNC!SPLOC >
.ENDM
:
.MACRO .BR7 SRC, FUNC, SPLOC ; INDEXED JUMP ON BR7 SET
MICPC=MICPC+1
< JUMP!BR7CON!SRC!FUNC!SPLOC >
.ENDM
:
.MACRO .Z SRC, FUNC, SPLOC ; INDEXED JUMP ON Z BIT SET
MICPC=MICPC+1
< JUMP!ZCOND!SRC!FUNC!SPLOC >
.ENDM
:
.MACRO .C SRC, FUNC, SPLOC ; INDEXED JUMP ON C BIT SET
MICPC=MICPC+1
< JUMP!CCOND!SRC!FUNC!SPLOC >
.ENDM
:
.SBTTL MOVE INSTRUCTIONS
:
MOVE=C ; MOVE OPCODE
:
.MACRO BRSHFT ; BR SHIFT RIGHT
MICPC=MICPC+1
< MOVE!SHFTBR!WRTEBR!SELB >
.ENDM
:
.MACRO BSHFTB ; BR ROTATE
MICPC=MICPC+1
< MOVE!SHFTBR!SELB!BR >
.ENDM
:
.MACRO SP SRC, FUNC, SPLOC ; LOAD SCRATCH-PAD

```

179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234

```

MICPC=MICPC+1
<MOVE!SPX!SRC!FUNC!SPLOC>

.ENDM

:
.MACRO SPBR SRC, FUNC, SPLOC ;LOAD SP AND BR
MICPC=MICPC+1
<MOVE!SPBRX!SRC!FUNC!SPLOC>

.ENDM

:
.MACRO MEM SRC, DATA ;MOVE TO MEMORY
MICPC=MICPC+1
<MOVE!WRMEM!SRC!<DATA>>

.ENDM

:
.MACRO MEMADR ADDRES, FUNC ;WRITE ADDRESS TO MEMORY
MICPC=MICPC+1
.IF B FUNC
<MOVE!WRMEM!<ADDRES-INIT&777/2>>
.IFF
<MOVE!WRMEM!FUNC!<ADDRES-INIT&777/2>>
.ENDC
.ENDM

:
.MACRO MEMINC SRC, DATA ;MOVE TO MEM, INCR MAR
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!SRC!<DATA>>

.ENDM

:
.MACRO BRWRTE SRC, DATA ;MOVE TO BR
MICPC=MICPC+1
<MOVE!WRTEBR!SRC!<DATA>>

.ENDM

:
.MACRO BRADDR ADDRES ;PUT RETURN ADDR (1 BYTE) IN BR
MICPC=MICPC+1
<MOVE!WRTEBR!<ADDRES-INIT&777/2>>

.ENDM

:
.MACRO OUTPUT SRC, DATA ;WRITE OUTPUT
MICPC=MICPC+1
<MOVE!WRROUT!SRC!<DATA>>

.ENDM

:
.MACRO OUT SRC, DATA
MICPC=MICPC+1
<MOVE!WRROUTX!SRC!<DATA>>

.ENDM

```

2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200

```

.MACRO LDMA SRC,DATA ;LOAD MEMORY ADDRESS REG
MICPC=MICPC+1
.IF IDN SRC,IMM
<MOVE!LDMAR!IMM!<DATA&377>>
.IFF
<MOVE!LDMAR!SRC!<DATA>>
.ENDC
.ENDM

```

```

.MACRO LDMAP SRC,DATA ;LOAD MEMORY PAGE NUMBER
MICPC=MICPC+1
.IF IDN SRC,IMM
<MOVE!LDMAPG!IMM!<DATA/400>>
.IFF
<MOVE!LDMAPG!SRC!<DATA>>
.ENDC
.ENDM

```

```

.MACRO LDADDR DATA ;LOAD A LINE TABLE ADDRESS
BRWRT IMM,DATA
LDMA BR,<ADD!SP.RMD>
.ENDM

```

```

.MACRO CMP SRC,SPADDR ;COMPARE SOURCE AND SP
MICPC=MICPC+1
<SUBTC!SRC!SPADDR>
.ENDM

```

```

.MACRO NOP SRC,FUNC,SPADDR ;NOP-SOURCE. FUNC. NO DEST
MICPC=MICPC+1
<SRC!FUNC!SPADDR>
.ENDM

```

```

.MACRO CALL REG,ADDRES ;SUBROUTINE CALL
DISP=<MICPC+1>&377
BRWRT IMM,DISP+3
SP BR,SELB,REG
ALWAYS ADDRES
.ENDM

```

```

.MACRO RETURN REG,PAGE ;SUBROUTINE RETURN
.ALWAY BR,SELA,<REG!PAGE>
.ENDM

```

284		.SBTTL INPUT/OUTPUT ASSIGNMENTS	
285		:IBUS ASSIGNMENTS	
286	100000	INCON=0+100000	:IN CONTROL CSR
287	100020	MAIN=20+100000	:MAINTAINENCE REGISTER
288	100040	OCON=40+100000	:OUT CONTROL CSR
289	100060	UBADDR=60+100000	:UNUSED
290	100100	PORT1=100+100000	:CSR4
291	100120	PORT2=120+100000	:CSR5
292	100140	PORT3=140+100000	:CSR6
293	100160	PORT4=160+100000	:CSR7
294	100200	NPR=200+100000	:NPR CONTROL
295	100220	UBBR=220+100000	:BR(INTERRUPT)CONTROL
296	000000	INDAT1=0	:INPUT DATA LOW BYTE
297	000020	INDAT2=20	:INPUT DATA HIGH BYTE
298	000140	IOBA1=140	:OUTPUT BA LOW BYTE
299	000160	IOBA2=160	:OUTPUT BA HIGH BYTE
300	000100	IIBA1=100	:INPUT BA LOW BYTE
301	000120	IIBA2=120	:INPUT BA HIGH BYTE
302	000200	RCVDAT=200	:RECEIVE DATA
303	000220	TMTCON=220	:TMTR CONTROL
304	000240	RCVCON=240	:RCVR CONTROL
305	000260	MODEM=260	:MODEM CONTROL
306	000300	SYNREG=300	:SYN REGISTER
307	000320	LNOSW=320	:LINE NUMBER SWITCH
308	000340	BMB73=340	:BMB73 ADDRESS
309	000360	LUMAIN=360	:LINE UNIT MAINTAINENCE
310		:OBUS ASSIGNMENTS	
311		:EXTENDED OBUS	
312	000000	OINCON=0	:IN CONTROL CSR
313	000001	OMAIN=1	:MAINT
314	000002	OCON=2	:OUT CONTROL CSR
315	000003	OUBADD=3	:UNUSED
316	000004	OPORT1=4	:CSR4
317	000005	OPORT2=5	:CSR5
318	000006	OPORT3=6	:CSR6
319	000007	OPORT4=7	:CSR7
320	000010	ONPR=10	:NPR CONTROL
321	000011	OBR=11	:BR CONTROL
322		:UNEXTENDED OBUS	
323	000002	OUTDA1=2	:OUTPUT DATA LOW BYTE
324	000003	OUTDA2=3	:OUTPUT DATA HIGH BYTE
325	000006	OBA1=6	:OUTPUT BA LOW BYTE
326	000007	OBA2=7	:OUTPUT BA HIGH BYTE
327	000004	IBA1=4	:INPUT BA LOW BYTE
328	000005	IBA2=5	:INPUT BA HIGH BYTE
329	000010	TMTDAT=10	:TMTR DATA
330	000011	OTMTCO=11	:TMTR CONTROL
331	000012	ORCVCO=12	:RCVR CONTROL
332	000013	OMODEM=13	:MODEM CONTROL
333	000014	SYNC=14	:SYN REGISTER
334	000017	OLUMAN=17	:LINE UNIT MAINT.

336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377

```

.SBTTL PROTOCOL DEPENDANT MACROS
.MACRO RSTATE STATE ;UPDATE RECEIVE STATE POINTER
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<STATE-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>
.ENDM

.MACRO TSTATE STATE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<STATE-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
.ENDM

.MACRO STATE ADDR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<ADDR-INIT&777/2>>
.ENDM

.MACRO PSTATE STATE
MEM IMM,<<STATE-INIT&777/2>>
.ENDM

.MACRO PSTATI STATE
MEMINC IMM,<<STATE-INIT&777/2>>
.ENDM

.MACRO SYNMAC
SP BR,SELB,SP2 ;UPDATE STATE POINTER FROM BR
SYNOUT
ALWAYS IDLE
.ENDM

.MACRO SYNOUT
LDMA IMM,UNMSG ;LOAD PTR TO UNNUMB MESSAGE SKELETON
OUTPUT <MEMX!INCMAR>,<SELB!OTMTCO> ;SOM TO TMTR CONTROL
OUTPUT <MEMX!INCMAR>,<SELB!TMDAT> ;SYNC TO TMTR SILO
.ENDM

```

177777

MICPC=177777 ;INIT MICRO PC

N05

DMC-1: MICROPROCESSOR INSTRUCTIONS
LOW.MAC 13-OCT-76 14:33

MACY11 27(1006) 07-MAR-77 16:24 PAGE 5
DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT

PAGE: 0065

379
380 000000
381 000000

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

385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405

.TITLE DMC11 DDCMP PROTOCOL IMPLEMENTATION
.IDENT /V0001/
.SBTTL VERSION 00A FEBRUARY 26,1975
.SBTTL
.SBTTL HARVEY M. SCHLESINGER
.SBTTL
.SBTTL COPYRIGHT 1975, DIGITAL EQUIPMENT CORPORATION
.SBTTL
.SBTTL VERSION 00B MARCH 17,1975
.SBTTL CSR AND MICROPROCESSOR CHANGES
.SBTTL
.SBTTL VERSION 00C NOVEMBER 6, 1975
.SBTTL RETRANSMISSION CHANGES
.SBTTL
.SBTTL VERSION 00D DECEMBER 3,1975
.SBTTL TRANSMIT DONE CHANGES
.SBTTL
.SBTTL THE LATEST MODIFICATIONS WERE ADDED ON:
.SBTTL OCTOBER 13, 1976
.SBTTL THIS VERSION WAS USED TO BLAST THE FIRST
.SBYTL RELEASE ON OCTOBER 13, 1976

COB

```

407 .SBTTL MICROPROCESSOR MAIN MEMORY ASSIGNMENTS
408 ;ALLOCATION OF MICROPROCESSOR MAIN MEMORY
409 000000 NAKSR=0 ;NAKS RECD--DYNAMIC
410 000001 NAKST=NAKSR+1 ;NAKS TMTE--DYNAMIC
411 000002 REPSR=NAKST+1 ;REPS RECD--DYNAMIC
412 000003 REPST=REPSR+1 ;REPS TMTE--DYNAMIC
413 000006 NP=REPST+3 ;CONSTANT 0
414 000007 NTLR=NP+1 ;NAKS-MSG NO BUFFERS CUMUL.
415 000010 NHDR=NTLR+1 ;NAKS-MSG HEADER BAD
416 000011 NDATA=NHDR+1 ;NAKS-DATA BAD
417 000012 NTLS=NDATA+1 ;NAK SENT --NO BUFFERS
418 000013 NHDS=NTLS+1 ;NAK SENT BAD HEADER
419 000014 NDATS=NHDS+1 ;NAK SENT BAD DATA
420 000015 REPCS=NDATS+1 ;REPS SENT CUMUL
421 000016 REPCR=REPCS+1 ;REPS RECD CUMUL
422 000017 BASE=REPCR+1 ;ORE TABLE BASE ADDRESS
423 000022 SRC=BASE+C ;START OF INPUT CHAIN--NEXT RECV DCNE
424 000023 ERC=SRC+1 ;END OF INPUT CHAIN
425 000024 LRC=ERC+1 ;LAST POINTER RECD
426 000025 RCL1=LRC+1 ;RECEIVE LINK #1
427 000032 RCL2=RCL1+5 ; " " #2
428 000037 RCL3=RCL2+5 ; " " #3
429 000044 RCL4=RCL3+5
430 000051 RCL5=RCL4+5
431 000056 RCL6=RCL5+5
432 000063 RCL7=RCL6+5
433 000070 STC=RCL7+5 ;START OF OUTPUT CHAIN---NEXT TMT DONE
434 000071 LTC=STC+1 ;LAST TRANSMITTED
435 000072 ETC=LTC+1 ;END OF TRANSMIT CHAIN
436 000073 TML1=ETC+1 ;TRANSMIT LINK #1
437 000101 TML2=TML1+6 ; " " #2
438 000107 TML3=TML2+6 ; " " #3
439 000115 TML4=TML3+6
440 000123 TML5=TML4+6
441 000131 TML6=TML5+6
442 000137 TML7=TML6+6
443 000145 TML8=TML7+6
444 000153 T=TML8+6 ;TYPE FIELD
445 000154 ST=T+1 ;SUBTYPE FIELD
446 000155 ISP17=ST+1 ;MSG ACKED IMAGE
447 000156 IMG10=ISP17+1 ;IMAGE OF BIT 1 OF SP10
448 000157 IMG11=IMG10+1 ;IMAGE OF SP11
449 000160 IMG12=IMG11+1 ;IMAGE OF SP12
450 000161 IMG17=IMG12+1 ;IMAGE OF SP17
451 000162 PRTST=IMG17+1 ;PORT STATE
452 000163 TYPTAB=PRTST+1 ;TYPE TABLE---
453 ;72 TYPE TABLE REP
454 ;73 " " NAK
455 000165 TYPSTT=TYPTAB+2 ;74 " " START
456 ;75 " " STACK
457
458
459 000170 OSTATE=TYPSTT+3 ;OLD STATE POINTER
460 000171 ISP11=OSTATE+1 ;SP11 IMAGE
461 000172 ISP12=ISP11+1 ;SP12 IMAGE
462 000173 INCONS=ISP12+1 ;IN CONTROL CSR IMAGE
  
```

463 000174
464 000240
465 000241
466 000242
467 000400

RTHRS=INCONS+1 ;RECV THRESHOLD LINK
NXTINT=240 ;NEXT INTERRUPT POSITION
NXTSP=NXTINT+1 ;END OF INTERRUPT CHAIN
INTSTK=NXTSP+1 ;STACK OF INTERRUPTS
MMEND=400 ;MAIN MEMORY END

469		.SBTTL	SCRATCH PAD ASSIGNMENTS
470	000000	SP0=0	:SP0---SCRATCH REGISTER
471	000001	SP1=1	:SP1---PORT STATUS WORD
472			:BIT ASSIGNMENTS
473			:BIT0---INIT MODE
474			:BIT1---SEC STATION SELECT(UNUSED)
475			:BIT2---NO BUFFER ASSIGNED IN BOOT MODE
476			:BIT3---DLE RECEIVED WHILE NOT IN MAINT MODE
477			:BIT4---INTERRUPT PENDING
478			:BIT6---DISCONNECT ERROR
479			:BIT7---BOOT MODE
480	000002	SP2=2	:SP2---TRANSMIT STATE POINTER
481	000003	SP3=3	:SP3---RECEIVE STATE POINTER
482	000004	SP4=4	:SP4---END RECV ADDRESS
483	000005	SP5=5	:SP5---END RECEIVE ADDRESS
484	000006	SP6=6	:SP6---END TRANSMIT ADDRESS
485	000007	SP7=7	:SP7---END TRANSMIT ADDRESS
486	000010	SP10=10	:SP10---LINE STATUS WORD
487			:BIT ASSIGNMENTS
488			:BIT0---UNNUMB PENDING
489			:BIT1---MESSAGE IN PROGRESS
490			:BIT2---LINE HAS GONE IDLE
491			:BIT3---START RECEIVED
492			:BIT4---CLEAR ACTIVE ON END
493			:BIT5---START MODE
494			:BIT6---HALF DUPLEX
495			:BIT7---OK TO SEND
496	000011	SP11=11	:SP11---R FIELD
497	000012	SP12=12	:SP12---N FIELD
498	000013	SP13=13	:SP13---TYPE
499	000014	SP14=14	:SP14---RECEIVE LINK IMAGE
500	000015	SP15=15	:SP15---TIMER ENTRY---NUMBER OF ONE SECOND TICKS
501	000016	SP16=16	:SP16---POINTER TO TMT LINK COPY IN MAIN MEM
502	000017	SP17=17	:SP17---LAST MESSAGE ACKNOWLEDGED

```

S04          .SBTTL INIT--INITIALIZATION ROUTINE
S05          ;ZEROS MAIN MEMORY
S06          ;LOOPS WAITING FOR RECEIVE DATA(BOOT?)
S07          ;OR FOR RQI TO BE SET
S08          ;WILL ACCEPT ONLY BASE FORMAT. ALL OTHERS WILL RETURN A PROCEDURE ERROR
S09          ;
S10          ;AT INITIALIZATION --- THE HARDWARE CLEARS THE BR AND MAR
S11          =11766
S12 011766   011766   INIT:  SP      BR,SELB,SP0          ;CLEAR SP0
          (1)          000000   MICPC=MICPC+1
          (1) 011766   063220   <MOVE!SPX!BR!SELB!SP0>
          (1)
S13 011770   000001   SP      BR,SELB,SP3          ;PAGE ONE TRANSFER ADDRESS
          (1)          000001   MICPC=MICPC+1
          (1) 011770   063223   <MOVE!SPX!BR!SELB!SP3>
          (1)
S14 011772   000002   SP      BR,SELB,SP17         ;CLEAR SP17
          (1)          000002   MICPC=MICPC+1
          (1) 011772   063237   <MOVE!SPX!BR!SELB!SP17>
          (1)
S15 011774   000003   OUT     BR,<SELA!OINCON>      ;ZERO THE IN CONTROL CSR
          (1)          000003   MICPC=MICPC+1
          (1) 011774   061200   <MOVE!WROUTX!BR!<SELA!OINCON>>
          (1)
S16 011776   000004   OUT     BR,<SELA!OOCON>      ;ZERO THE OUT CONTROL CSR
          (1)          000004   MICPC=MICPC+1
          (1) 011776   061202   <MOVE!WROUTX!BR!<SELA!OOCON>>
          (1)
S17 012000   000005   SP      IMM,370,SP10        ;WRITE 5 ONE BITS TO THE HIGH ORDER
          (1)          000005   MICPC=MICPC+1
          (1) 012000   003370   <MOVE!SPX!IMM!370!SP10>
          (1)
S18          ;BITS OF SP10
S19 012002   000006   S$:    SP      BR,AA,SP10    ;SHIFT SP10 LEFT SETTING CARRY THE
          (1)          000006   MICPC=MICPC+1
          (1) 012002   06313C   <MOVE!SPX!BR!AA!SP10>
          (1)
S20          ;FIRST 5 TIMES THRU THE LOOP
S21 012004   000007   MEMINC BR,ADDC!SP3         ;WRITE A ONE TO THE FIRST 5 MEMORY
          (1)          000007   MICPC=MICPC+1
          (1) 012004   076423   <MOVE!WRMEM!INCMAR!BR!<ADDC!SP3>>
          (1)
S22          ;LOCATIONS AND ZERO THE REST
S23 012006   000010   SP      BR,INCA,SP0        ;INCREMENT COUNTER
          (1)          000010   MICPC=MICPC+1
          (1) 012006   063060   <MOVE!SPX!BR!INCA!SP0>
          (1)
S24 012010   000011   Z      10$                ;ALL DONE
          (1)          000011   MICPC=MICPC+1
          (1) 012010   101413   <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
          (1)
S25 012012   000012   ALWAYS 5$                 ;KEEP GOING
          (1)          000012   MICPC=MICPC+1
          (1) 012012   100406   <JUMP!ALCOND!<5$-INIT&3000*4>!<5$-INIT&777/2>>
          (1)
S26 012014   10$:  SPBR     IMM,1,SP1          ;WRITE A 1 TO THE BR AND SP1
    
```

```

(1) 012014 000013      MICPC=MICPC+1
(1) 012014 003401      <MOVE!SPBRX!IMM!1!SP1>
527 012016 000014      SP      BR,SELB,SP11      ;WRITE A 1 TO SP11
(1) 012016 000014      MICPC=MICPC+1
(1) 012016 003231      <MOVE!SPX!BR!SELB!SP11>
528 012020 000015      SP      BR,SELB,SP12      ;WRITE A 1 TO SP12
(1) 012020 000015      MICPC=MICPC+1
(1) 012020 003232      <MOVE!SPX!BR!SELB!SP12>
529 012022 000016      LDMA   IMM,PRST      ;POINT MAR TO UNNUM MSG SKELETON
(1) 012022 000016      MICPC=MICPC+1
(1) 012022 010152      .IF IDN IMM,IMM
(1) 012022 010152      <MOVE!LDMAR!IMM!<PRTST&377>>
(1) 012022 010152      .IFF
(1) 012022 010152      <MOVE!LDMAR!IMM!<PRTST>>
(1) 012022 010152      .ENDC
(1) 012022 000016      JOC
530 012024 000017      PSTATI NIDLE2      ;WRITE PORT IDLE STATE
(1) 012024 000017      MEMINC IMM,<<NIDLE2-INIT&777/2>>
(1) 012024 016520      MICPC=MICPC+1
(1) 012024 016520      <MOVE!WRMEM!INCMAR!IMM!<<NIDLE2-INIT&777/2>>>
531 012026 000020      BRWRTE IMM,226      ;WRITE SYNC TO MEMORY
(1) 012026 000020      MICPC=MICPC+1
(1) 012026 000020      <MOVE!WRTEBR!IMM!<226>>
532 012030 000021      OUTPUT BR,SELB!SYNC ;LOAD THE SYNC REGISTER
(1) 012030 000021      MICPC=MICPC+1
(1) 012030 002234      <MOVE!WROUT!BR!<SELB!SYNC>>
533 012032 000022      MEMINC IMM,3      ;REP
(1) 012032 000022      MICPC=MICPC+1
(1) 012032 016403      <MOVE!WRMEM!INCMAR!IMM!<3>>
534 012034 000023      MEM    IMM,2      ;NAK
(1) 012034 000023      MICPC=MICPC+1
(1) 012034 002402      <MOVE!WRMEM!IMM!<2>>
535 012036 000024      SP      MEMX!INCMAR,SELB,SP15 ;SET STARTING COUNT
(1) 012036 000024      MICPC=MICPC+1
(1) 012036 057235      <MOVE!SPX!MEMX!INCMAR!SELB!SP15>
536 012040 000025      MEMINC IMM,6      ;START
(1) 012040 000025      MICPC=MICPC+1
(1) 012040 016406      <MOVE!WRMEM!INCMAR!IMM!<6>>
537 012042 000026      MEMINC IMM,7      ;STACK
(1) 012042 000026      MICPC=MICPC+1
(1) 012042 016407      <MOVE!WRMEM!INCMAR!IMM!<7>>
538 012044 000027      MEMINC IMM,1      ;ACK
(1) 012044 000027      MICPC=MICPC+1
(1) 012044 016401      <MOVE!WRMEM!INCMAR!IMM!<1>>

```

```

539 012046 000030          LDMA IMM,STC          :LOAD ADDRESS OF LAST TMT CHAIN
      (1)          001          MICPC=MICPC+1
      (1)          012046 010070          .IF IDN IMM,IMM
      (1)          001          <MOVE!LDMAR!IMM!<STC&377>
      (1)          001          .IFF
      (1)          000          <MOVE!LDMAR!IMM!<STC>>
      (1)          003          .ENDC

540 012050 000031          MEMINC IMM,TML1         :STORE ADDRESS OF FIRST TMT LINK
      (1)          012050 016473          MICPC=MICPC+1
      (1)          001          <MOVE!WRMEM!INCMAR!IMM!<TML1>>

541 012052 000032          MEMINC IMM,TML1         :STORE ADDRESS OF FIRST TMT LINK
      (1)          012052 016473          MICPC=MICPC+1
      (1)          001          <MOVE!WRMEM!INCMAR!IMM!<TML1>>

542 012054 000033          MEMINC IMM,TML1         :STORE ADDRESS OF FIRST TMT LINK
      (1)          012054 016473          MICPC=MICPC+1
      (1)          001          <MOVE!WRMEM!INCMAR!IMM!<TML1>>

543 012056 000034          LDMA IMM,SRC          :LOAD ADDRESS OF LAST RECV CHAIN
      (1)          001          MICPC=MICPC+1
      (1)          012056 010022          .IF IDN IMM,IMM
      (1)          001          <MOVE!LDMAR!IMM!<SRC&377>>
      (1)          000          .IFF
      (1)          000          <MOVE!LDMAR!IMM!<SRC>>
      (1)          000          .ENDC

544 012060 000035          MEMINC IMM,RCL1         :SET UP ADDRESS OF FIRST RECV LINK
      (1)          012060 016425          MICPC=MICPC+1
      (1)          001          <MOVE!WRMEM!INCMAR!IMM!<RCL1>>

545 012062 000036          MEMINC IMM,RCL1         :SET UP ADDRESS OF FIRST RECV LINK
      (1)          012062 016425          MICPC=MICPC+1
      (1)          001          <MOVE!WRMEM!INCMAR!IMM!<RCL1>>

546 012064 000037          MEMINC IMM,RCL1         :SET UP ADDRESS OF FIRST RECV LINK
      (1)          012064 016425          MICPC=MICPC+1
      (1)          001          <MOVE!WRMEM!INCMAR!IMM!<RCL1>>

547 012066 000040          LDMA IMM,NXTINT        :ADDRESS OF NEXT INTERRUPT POINTER TO MAR
      (1)          001          MICPC=MICPC+1
      (1)          012066 010240          .IF IDN IMM,IMM
      (1)          001          <MOVE!LDMAR!IMM!<NXTINT&377>>
      (1)          000          .IFF
      (1)          000          <MOVE!LDMAR!IMM!<NXTINT>>
      (1)          000          .ENDC

548 012070 000041          MEMINC IMM,INTSTK       :INITIALIZE NEXT INTERRUPT POINTER
      (1)          012070 016642          MICPC=MICPC+1
      (1)          001          <MOVE!WRMEM!INCMAR!IMM!<INTSTK>>

549 012072 000042          MEM IMM,INTSTK        :INITIALIZE INSERTION POINTER
      (1)          012072 002642          MICPC=MICPC+1
      (1)          001          <MOVE!WRMEM!IMM!<INTSTK>>
    
```

```

550 012074      BRWRT  IMM,200          :WRITE THE RUN BIT TO THE BR
(1)          MICPC=MICPC+1
(1) 012074 000043 <MOVE!WRTEBR!IMM!<200>>
(1)
551 012076      OUT      BR,<SELB!OMAIN> :WRITE THE RUN BIT TO MAINT CSR
(1)          MICPC=MICPC+1
(1) 012076 000044 <MOVE!WROUTX!BR!<SELB!OMAIN>>
(1)
552          :FALL INTO IDLE LOOP
553          .IF NDF LOW
554          ALWAYS IDLE
555
556          REXIT: SP      BR,SELB.SP3
557          .ENDC
001
000

```

```

559 .SBTTL IDLE--PROGRAM IDLE LOOP
560 ;PROGRAM IDLE LOOP
561 ;DISPATCHES TO APPROPRIATE SERVICE ROUTINES
562 ;USES STATE POINTERS FOR TMT,RCV,CSR ACTIVITY
563
564 012100 IDLE: BRWRT BR, <SELA!SP10> ;READ TRANSMIT STATUS WORD FROM SP10 TO 3F
(1) MICPC=MICPC+1
(1) 012100 000045 <MOVE!WRTEBR!BR!<SELA!SP10>>
(1) 06361C
565 012102 BR1 TMTDA ;IF DATA TO SEND-- BRANCH
(1) MICPC=MICPC+1
(1) 012102 000046 <JUMP!BR1CON!<TMTDA-INIT&3000*4>!<TMTDA-INIT&777/2>>
(1) 11240C
566 012104 BR0 TMTDA ;IF DATA TO SEND-- BRANCH
(1) MICPC=MICPC+1
(1) 012104 000047 <JUMP!BR0CON!<TMTDA-INIT&3000*4>!<TMTDA-INIT&777/2>>
(1) 11200C
567 012106 ALWAYS I1
(1) MICPC=MICPC+1
(1) 012106 000050 <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
(1) 100452
568
569 012110 XEXIT: SP BR, SELB, SP2
(1) MICPC=MICPC+1
(1) 012110 000051 <MOVE!SPX!BR!SELB!SP2>
(1) 063222
570 012112 11: BRWRT IBUS, RCVCON ;READ LINE UNIT RECEIVE CONTROL WORD
(1) MICPC=MICPC+1
(1) 012112 000052 <MOVE!WRTEBR!IBUS!<RCVCON>>
(1) 020640
571 012114 .BR4 BR, SELA, SP3!PAGE1 ;BRANCH BASED UPON RCV STATE
(1) MICPC=MICPC+1
(1) 012114 000053 <JUMP!BR4CON!BR!SELA!SP3!PAGE1>
(1) 167203
572 012116 12: LDMA IMM, PRTST ;ADDRESS PORT STATE
(1) MICPC=MICPC+1
(1) 012116 000054 .IF IDN IMM, IMM
(1) 001 <MOVE!LDMAR!IMM!<PRTST&377>>
(1) 0101E2 .IFF
(1) <MOVE!LDMAR!IMM!<PRTST>>
(1) .ENDC
(1) 000
573 012120 .ALWAY MEMX, SELB, 0 ;INDEX
(1) MICPC=MICPC+1
(1) 012120 000055 <JUMP!ALCOND!MEMX!SELB!0>
(1) 140620

```



```

575          .SBTTL BASSRV---- BASE SERVICE ROUTINE
576 012122 BASSRV: PSTATE NIDLE2
(1) 012122 MEM IMM, <<NIDLE2-INIT&777/2>>
(2) 000056 MICPC=MICPC+1
(2) 012122 002520 <MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>
(2)
577 012124 LDMA IMM, BASE ;CLEAR TO MAR SO IT POINTS TO BASE POINT
(1) 000057 MICPC=MICPC+1
(1) 00: .IF IDN IMM, IMM
(1) 012124 010017 <MOVE!LDMAR!IMM!<BASE&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<BASE>>
(1) .ENDC
(1) 000
578 012126 MEMINC IBUS, PORT1 ;READ CSR4
(1) 000060 MICPC=MICPC+1
(1) 012126 136500 <MOVE!WRMEM!INCMAR!IBUS!<PORT1>>
(1)
579 012130 MEMINC IBUS, PORT2 ;READ CSR5
(1) 000061 MICPC=MICPC+1
(1) 012:30 136520 <MOVE!WRMEM!INCMAR!IBUS!<PORT2>>
(1)
580 012132 MEM IBUS, PORT4
(1) 000062 MICPC=MICPC+1
(1) 012132 122560 <MOVE!WRMEM!IBUS!<PORT4>>
(1)
581 012134 SP IBUS, INCON, SPO ;READ INPUT CONTROL CSR
(1) 000063 MICPC=MICPC+1
(1) 012134 123000 <MOVE!SPX!IBUS!INCON!SPO>
(1)
582 012136 BRWRTE IMM, 100 ;CLEAR THE BR
(1) 000064 MICPC=MICPC+1
(1) 012136 000500 <MOVE!WRTEBR!IMM!<100>>
(1)
583 012140 OUT BR, <AANDB!OINCON> ;CLEAR THE INCONTROL CSR
(1) 000065 MICPC=MICPC+1
(1) 012140 061260 <MOVE!WROUTX!BR!<AANDB!OINCON>>
(1)
584 012142 BRWRTE IMM, 120 ; MASK FOR HDX AND DTR
(1) 000066 MICPC=MICPC+1
(1) 012142 000520 <MOVE!WRTEBR!IMM!<120>>
(1)
585 012144 OUTPUT BR, <SELB!OMODEM>
(1) 000067 MICPC=MICPC+1
(1) 012144 062233 <MOVE!WROUT!BR!<SELB!OMODEM>>
(1)
586 012146 BRWRTE MEMX, SELB ;READ SEL6
(1) 000070 MICPC=MICPC+1
(1) 012146 040620 <MOVE!WRTEBR!MEMX!<SELB>>
(1)
587 012150 BR4 RESUME ;IF SET RESUME
(1) 000071 MICPC=MICPC+1
(1) 012150 103100 <JUMP!BR4CON!<RESUME-INIT&3000*4>!<RESUME-INIT&777/2>>
(1)
588 012152 LDMA IMM, T ;LOAD ADDRESS OF TYPE FIELD
(1) 000072 MICPC=MICPC+1

```

```

(1)      001
(1) 012152 010153      .IF IDN IMM,IMM
(1)      .IFF
(1)      <MOVE!LDMAR!IMM!<T&377>>
(1)      .IFF
(1)      <MOVE!LDMAR!IMM!<T>>
(1)      .ENDC
(1)      000
589 012154      MEMINC IMM,6      ;WRITE START TYPE TO MEMORY
(1)      000073      MICPC=MICPC+1
(1) 012154 016406      <MOVE!WRMEM!INCMAR!IMM!.6
(1)
590 012156      MEM      IMM,300      ;WRITE SELECT AND FINAL TO MEMORY
(1)      000074      MICPC=MICPC+1
(1) 012156 002700      <MOVE!WRMEM!IMM!<300>>
(1)
591 012160      SP      BR,DECA,SP1      ;TURN OFF INIT MODE
(1)      000075      MICPC=MICPC+1
(1) 012160 063161      <MOVE!SPX!BR!DECA!SP1>
(1)
592 012162      BS1:  BRWRTE IMM,241      ;SET OK TO SEND,STARTMODE AND UNNUM PENDING
(1)      000076      MICPC=MICPC+1
(1) 012162 000641      <MOVE!WRTEBR!IMM!<241>>
(1)
593 012164      ALWAYS SA3
(1)      000077      MICPC=MICPC+1
(1) 012164 110733      <JUMP!ALCOND!<SA3-INIT&3000*4>!<SA3-INIT&777/2>>
(1)
594 012166      RESUME: SP      IMM,SP4,4      ;SET UP SP4 FOR COUNTING NPRS
(1)      000100      MICPC=MICPC+1
(1) 012166 003004      <MOVE!SPX!IMM!SP4!4>
(1)
595 012170      SP      BR,INCA,SP10      ;SET UNNUMB MESSAGE PENDING TO
(1)      000101      MICPC=MICPC+1
(1) 012170 063070      <MOVE!SPX!BR!INCA!SP10>
(1)
596
597 012172      LDMA      IMM,BASE      ;TRICK TRANSMITTER CODE
(1)      000102      MICPC=MICPC+1      ;ADDRESS BASE TABLE ADDRESS
(1)      001
(1) 012172 010017      .IF IDN IMM,IMM
(1)      <MOVE!LDMAR!IMM!<BASE&377>>
(1)      .IFF
(1)      <MOVE!LDMAR!IMM!<BASE>>
(1)      .ENDC
(1)      000
598 012174      STATE      FUDGE      ;SET TMTR STATE TO ENTER TABLE UPDATE
(1)      000103      MICPC=MICPC+1
(1) 012174 000736      <MOVE!WRTEBR!IMM!<FUDGE-INIT&777/2>>
599 012176      ALWAYS T80      ;GO SET UP MXT BITS AND ADRESS OF BASE FOR NPRS
(1)      000104      MICPC=MICPC+1
(1) 012176 110462      <JUMP!ALCOND!<T80-INIT&3000*4>!<T80-INIT&777/2>>
(1)
600 012200      BS2:  LDMA      IMM,IMG10
(1)      000105      MICPC=MICPC+1
(1)      001
(1) 012200 010156      .IF IDN IMM,IMM
(1)      <MOVE!LDMAR!IMM!<IMG10&377>>
(1)      .IFF
(1)      <MOVE!LDMAR!IMM!<IMG10>>
(1)

```

```

(1)          000          .ENDC
(1)
601 012202          SP      MEMX,AORB,SP10          ;RESTORE BIT 1 OF SP10
(1)          000106          MICPC=MICPC+1
(1) 012202          043310          <MOVE!SPX!MEMX!AORB!SP10>
(1)
602 012204          SP      MEMX!INCMAR,SELB,SP11          ;RESTORE SP11
(1)          000107          MICPC=MICPC+1
(1) 012204          057231          <MOVE!SPX!MEMX!INCMAR!SELB!SP11>
(1)
603 012206          SP      MEMX!INCMAR,SELB,SP12          ;RESTORE SP12
(1)          000110          MICPC=MICPC+1
(1) 012206          057232          <MOVE!SPX!MEMX!INCMAR!SELB!SP12>
(1)
604 012210          SP      MEMX,SELB,SP17          ;RESTORE          SP17
(1)          000111          MICPC=MICPC+1
(1) 012210          043237          <MOVE!SPX!MEMX!SELB!SP17>
(1)
605 012212          SP      BR,DECA,SP10          ;TURN OFF UNNUM MESSAGE PENDING AND
(1)          000112          MICPC=MICPC+1
(1) 012212          063170          <MOVE!SPX!BR!DECA!SP10>
(1)
606
607 012214          STATE  NIDLE2          ;ZERO THE BRG
(1)          000113          MICPC=MICPC+1          ;PORT STATUS
(1) 012214          000520          <MOVE!WRTEBR!IMM!<NIDLE2-INIT&777/2>>
608 012216          SP      BR,SELB,SP13          ;SAVE IN SP13 - NOTE THAT STATUS READ INTO
(1)          000114          MICPC=MICPC+1
(1) 012216          063233          <MOVE!SPX!BR!SELB!SP13>
(1)
609
610 012220          SP      BR,DECA,SP1          ;RAM WAS TABLE UPDATE WHICH SAVED STATUS IN SP13
(1)          000115          MICPC=MICPC+1          ;CLEAR INIT MODE
(1) 012220          063161          <MOVE!SPX!BR!DECA!SP1>
(1)
611 012222          BRWRTE IMM,200          ;SET OK TO SEND
(1)          000116          MICPC=MICPC+1
(1) 012222          000600          <MOVE!WRTEBR!IMM!<200>>
(1)
612 012224          ALWAYS SA3
(1)          000117          MICPC=MICPC+1
(1) 012224          110733          <JUMP!ALCOND!<SA3-INIT&3000*4>!<SA3-INIT&777/2>>
(1)

```

```

614
615 012226 000120
(1) 012226 060601
(1)
(1)
616 001
617
618 000
619 001
620 012230 000121
(1) 012230 103220
(1)
(1)
621 000
622 012232 000122
(1) 012232 123400
(1)
(1)
623 012234 000123
(1) 012234 001620
(1)
(1)
624 012236 000124
(1) 012236 103155
(1)
(1)
625
626
627 012240 000125
(1) 012240 060601
(1)
(1)
628 001
629
630 000
631 001
632 012242 000126
(1) 012242 102131
(1)
(1)
633 000
634 012244 000127
(1) 012244 123620
(1)
(1)
635 012246 000130
(1) 012246 113245
(1)
(1)
636 001
637
638 000
639 001
640 012250 000131
(1) 012250 023660
(1)

```

```

.SBTTL NIDLE2---NO CSR ACTIVITY STATE
NIDLE2: BRWRT BR SELA!SPI ;READ PORT STATUS WORC
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SPI>>

.IF NDF $LOW
BR4 NIDLES ;INTERRUPT PENDING?---BRANCH
.ENDC
.IF DF $LOW
BR4 OUTINT
MICPC=MICPC+1
<JUMP!BR4CON!<OUTINT-INIT&3000*4>!<OUTINT-INIT&777/2>>

.ENDC
SPBR IBUS,INCON,SPO ;READ INPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!INCON!SPO>

BRSHFT ;SHIFT IT RIGHT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>

BR4 INWAT1 ;IF RQI SET -- BRANCH
MICPC=MICPC+1
<JUMP!BR4CON!<INWAT1-INIT&3000*4>!<INWAT1-INIT&777/2>>

;TO RE-READ THE IN CNTRL REGISTER TO AVOID
;A RACE IN MICRO-P READ/UNIBUS WRITE
;READ PORT STATUS
NIDLE6: BRWRT BR SELA!SPI
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SPI>>

.IF NDF $LOW
BR0 IDLE
.ENDC
.IF DF $LOW
BR0 10S
MICPC=MICPC+1
<JUMP!BR0CON!<10S-INIT&3000*4>!<10S-INIT&777/2>>

.ENDC
SPBR IBUS,UBBR,SPO ;TIMER EXPIRES?
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!UBBR!SPO>

BR4 TIMSRV
MICPC=MICPC+1
<JUMP!BR4CON!<TIMSRV-INIT&3000*4>!<TIMSRV-INIT&777/2>>

.IF NDF $LOW
ALWAYS IDLE
.ENDC
.IF DF $LOW
10S: SPBR IBUS,MODEM,SPO ;READ MODEM CONTROL CSR
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!MODEM!SPO>

```

```

(1)
641 012252          BRWRTE BR,AA!SPO          ;SHIFT IT LEFT
(1)                MICPC=MICPC+1
(1) 012252 060520  <MOVE!WRTEBR!BR!<AA!SPO>>
(1)
642 012254          BR4      SETDSR          ;IF DSR SET, CLEAR FLAG
(1)                MICPC=MICPC+1
(1) 012254 000133  <JUMP!BR4CON!<SETDSR-INIT&3000*4>!<SETDSR-INIT&777/2>>
(1)                103150
(1)
643 012256          BRWRTE BR,SELA!SP10       ;READ LINE STATUS WORD
(1)                MICPC=MICPC+1
(1) 012256 000134  <MOVE!WRTEBR!BR!<SELA!SP10>>
(1)                060510
(1)
644 012260          BRSHFT
(1)                MICPC=MICPC+1
(1) 012260 000135  <MOVE!SHFTBR!WRTEBR!SELB>
(1)                001620
(1)
645 012262          BR4      IDLE            ;START MODE
(1)                MICPC=MICPC+1
(1) 012262 000136  <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)                103045
(1)
646 012264          BRWRTE BR,AA!SP1         ;READ PORT STATUS WORD
(1)                MICPC=MICPC+1
(1) 012264 000137  <MOVE!WRTEBR!BR!<AA!SP1>>
(1)                060521
(1)
647 012266          BR1      IDLE            ;INIT MODE
(1)                MICPC=MICPC+1
(1) 012266 000140  <JUMP!BR1CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)                102445
(1)
648 012270          BR7      IDLE            ;DISCONNECT ERROR ALREADY SENT
(1)                MICPC=MICPC+1
(1) 012270 000141  <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)                103445
(1)
649 012272          SPBR     IBUS,MAIN,SPO    ;READ THE MAINT REGISTER
(1)                MICPC=MICPC+1
(1) 012272 000142  <MOVE!SPBRX!IBUS!MAIN!SPO>
(1)                123420
(1)
650 012274          BRWRTE BR,ADD!SPO        ;SHIFT LEFT
(1)                MICPC=MICPC+1
(1) 012274 000143  <MOVE!WRTEBR!BR!<ADD!SPO>>
(1)                060400
(1)
651 012276          BR4      IDLE            ;LU LOOP -- EXIT
(1)                MICPC=MICPC+1
(1) 012276 000144  <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)                103045
(1)
652 012300          BRWRTE IMM,100          ;WRITE DISCONNECT ERROR
(1)                MICPC=MICPC+1
(1) 012300 000145  <MOVE!WRTEBR!IMM!<100>>
(1)                000500
(1)
653 012302          SP      BR,AORB,SP1      ;FLAG ERROR RECORDED
(1)                MICPC=MICPC+1
(1) 012302 000146  <MOVE!SPX!BR!AORB!SP1>
(1)                063301
(1)
654 012304          ALWAYS  ERRXX           ;MAKE A CONTROL OUT
(1)                MICPC=MICPC+1
(1) 012304 000147  <JUMP!ALCOND!<ERRXX-INIT&3000*4>!<ERRXX-INIT&777/2>>
(1)                114671

```

(1)
655 012306
(1) 000150
(1) 012306 000677
(1)
656 012310
(1) 000151
(1) 012310 100652
(1)
657 000
658 001
659
660
661 000

```

SETDSR: BRWRT  IMM,277          ;CLEAR DISCONNECT ERROR FLAG
          MICPC=MICPC+1
          <MOVE!WRTEBR!IMM!<277>>

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

          .ENDC
          .IF NDF $LOW
NIDLES:  PSTATE OUTINT          ;SET STATE FOR INTERRUPT PROCESSING
          ALWAYS IDLE
          .ENDC

```

```

663
664 012312          .SBTTL INWAIT---WAIT FOR ROI TO CLEAR
(1)                SPBR      IBUS,INCON,SPO          ;READ INPUT CONTROL CSR
(1)                MICPC=MICPC+1
(1)                <MOVE!SPBRX!IBUS!INCON!SPO>
665 012314          BRWRT  BR,<AA!SPO>              ;SHIFT IT LEFT
(1)                MICPC=MICPC+1
(1)                <MOVE!WRTEBR!BR!<AA!SPO>>
666 012316          BR7     NIDLE3                    ;INTERRUPT ENABLE HAS BEEN SET
(1)                MICPC=MICPC+1
(1)                <JUMP!BR7CON!<NIDLE3-INIT&3000*4>!<NIDLE3-INIT&777/2>>
667
668 012320          INWAT1: SPBR      IBUS,INCON,SPO          ;READ THE INPUT CONTROL CSR
(1)                MICPC=MICPC+1
(1)                <MOVE!SPBRX!IBUS!INCON!SPO>
(1)
669 012322          BR7     INWAT2                    ;READY IN STILL SET
(1)                MICPC=MICPC+1
(1)                <JUMP!BR7CON!<INWAT2-INIT&3000*4>!<INWAT2-INIT&777/2>>
(1)
670 012324          NIDLE3: PSTATE INWAT1              ;UPDATE STATE TO INPUT
(1)                MEM      IMM,<<INWAT1-INIT&777/2>>
(2)                MICPC=MICPC+1
(2)                <MOVE!WRMEM!IMM!<<INWAT1-INIT&777/2>>>
(2)
671 012326          BRWRT  BR,AA!SPO                  ;SHIFT CSR LEFT
(1)                MICPC=MICPC+1
(1)                <MOVE!WRTEBR!BR!<AA!SPO>>
(1)
672 012330          BR7     ININT
(1)                MICPC=MICPC+1
(1)                <JUMP!BR7CON!<ININT-INIT&3000*4>!<ININT-INIT&777/2>>
(1)
673 012332          PSTATE INWAIT                    ;UPDATE STATE POINTER TO NO INTERRUPT GENERATED
(1)                MEM      IMM,<<INWAIT-INIT&777/2>>
(2)                MICPC=MICPC+1
(2)                <MOVE!WRMEM!IMM!<<INWAIT-INIT&777/2>>>
(2)
674 012334          NIDLE4: BRWRT  IMM,200
(1)                MICPC=MICPC+1
(1)                <MOVE!WRTEBR!IMM!<200>>
(1)
675 012336          OUT     BR,AORB!OINCON            ;SET THE ROYI
(1)                MICPC=MICPC+1
(1)                <MOVE!WROUTX!BR!<AORB!OINCON>>
(1)
676 012340          ALWAYS IDLE
(1)                MICPC=MICPC+1
(1)                <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
677
678 012342          INWAT2: BRSHFT                      ;SHIFT THE BR RIGHT
(1)                MICPC=MICPC+1
(1)                <MOVE!SHFTBR!WRTEBR!SELB>

```

```

(1)
679 012344          BR4   NIDLE6          ;RQI SET--- GO AWAY
(1)          0001E7          MICPC=MICPC+1
(1) 012344 103125          <JUMP!BR4CON!<NIDLE6-INIT&3000*4>!<NIDLE6-INIT&777/2>>
(1)
680          001          .IF NDF $LOW
681          PSTATE INSRV          ;SET NEXT STATE TO INPUT SERVICE
682          ALWAYS IDLE
683          .ENDC
(1)          000
684 012346          INSRV: SPBR   IBUS,INCON,SPD          ;READ THE INPUT CONTROL CSR
(1)          000170          MICPC=MICPC+1
(1) 012346 123400          <MOVE!SPBRX!IBUS!INCON!SPD>
(1)
685 012350          BR1   30$          ;--SENSE OR BASE
(1)          000171          MICPC=MICPC+1
(1) 012350 102605          <JUMP!BR1CON!<30$-INIT&3000*4>!<30$-INIT&777/2>>
(1)
686 012352          BR0   10$          ;CNTL 'I
(1)          000172          MICPC=MICPC+1
(1) 012352 102177          <JUMP!BR0CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
(1)
687 012354          BRSHFT          ;MUST BE BA/CC-SHIFT FOR IN OR OUT
(1)          000173          MICPC=MICPC+1
(1) 012354 001620          <MOVE!SHFTBR!WRTEBR!SELB>
(1)
688 012356          BR1   15$          ;
(1)          000174          MICPC=MICPC+1
(1) 012356 102601          <JUMP!BR1CON!<15$-INIT&3000*4>!<15$-INIT&777/2>>
(1)
689 012360          PSTATE TBASRV          ;TRANSMITTER
(1) 012360          MEM   IMM, <<TBASRV-INIT&777/2>>
(2)          000175          MICPC=MICPC+1
(2) 012360 002703          <MOVE!WRMEM!IMM!<<TBASRV-INIT&777/2>>>
(2)
690 012362          ALWAYS 20$
(1)          00017E          MICPC=MICPC+1
(1) 012362 100602          <JUMP!ALCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
(1)
691 012364          10$: PSTATE CTLSRV
(1) 012364          MEM   IMM, <<CTLSRV-INIT&777/2>>
(2)          000177          MICPC=MICPC+1
(2) 012364 002661          <MOVE!WRMEM!IMM!<<CTLSRV-INIT&777/2>>>
(2)
692 012366          ALWAYS 20$
(1)          000200          MICPC=MICPC+1
(1) 012366 100602          <JUMP!ALCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
(1)
693 012370          15$: PSTATE RBASRV
(1) 012370          MEM   IMM, <<RBASRV-INIT&777/2>>
(2)          000201          MICPC=MICPC+1
(2) 012370 00272E          <MOVE!WRMEM!IMM!<<RBASRV-INIT&777/2>>>
(2)
694 012372          20$: BRWRTE BR SELA!SP1          ;INIT MODE
(1)          000202          MICPC=MICPC+1
(1) 012372 060601          <MOVE!WRTEBR!BR!<SELA!SP1>>
(1)

```



```

695 012374      BRO      PROCER      ; IF INIT MODE--ERROR
(1) (1) 012374 000203      MICPC=MICPC+1
(1) (1) 012374 102211      < JUMP! BROCON! < PROCER-INIT&3000*4>! < PROCER-INIT&777/2>>
(1)
696 012376      ALWAYS  IDLE
(1) (1) 012376 000204      MICPC=MICPC+1
(1) (1) 012376 100445      < JUMP! ALCOND! < IDLE-INIT&3000*4>! < IDLE-INIT&777/2>>
(1)
697          001
698          30$: .IF NDF $LOW
699          PSTATE  BASSRV
700          BRO      35$      ; IF BASE -PROCESS
701          ALWAYS  PROCER
702          35$: BRWRT  BR, SELA! SP1      ; INIT MODE
703          BRO      IDLE
704          .ENDC
705          .IF DF  $LOW
(1) 012400 000205      30$: BRO      35$      ; IF BASE---PROCESS
(1) 012400 102207      MICPC=MICPC+1
(1) 012400 102207      < JUMP! BROCON! < 35$-INIT&3000*4>! < 35$-INIT&777/2>>
(1)
706 012402      ALWAYS  PROCER
(1) (1) 012402 000206      MICPC=MICPC+1
(1) (1) 012402 100611      < JUMP! ALCOND! < PROCER-INIT&3000*4>! < PROCER-INIT&777/2>>
(1)
707 012404      35$: BRWRT  BR, SELA! SP1      ; INIT MODE?
(1) (1) 012404 000207      MICPC=MICPC+1
(1) (1) 012404 060601      < MOVE! WRTEBR! BR! < SELA! SP1>>
(1)
708 012406      BRO      BASSRV
(1) (1) 012406 000210      MICPC=MICPC+1
(1) (1) 012406 102056      < JUMP! BROCON! < BASSRV-INIT&3000*4>! < BASSRV-INIT&777/2>>
(1)
709          000
710 012410      PROCER: .ENDC
(1) (1) 012410 000211      BRWRT  IMM, 100      ; CLEAR INPUT CONTROL CSR
(1) (1) 012410 000500      MICPC=MICPC+1
(1) (1) 012410 000500      < MOVE! WRTEBR! IMM! < 100>>
(1)
711 012412      OUT      BR, AANDB! OINCON      ::
(1) (1) 012412 000212      MICPC=MICPC+1
(1) (1) 012412 061260      < MOVE! WROUTX! BR! < AANDB! OINCON>>
(1)
712 012414      LDMA   IMM, << RTHRS+3>>      : ADDRESS ERROR LIM
(1) (1) 012414 000213      MICPC=MICPC+1
(1) (1) 012414 010177      .IF IDN IMM, IMM
(1) (1) 012414 010177      < MOVE! LDMA! IMM! << RTHRS+3>> < 377>>
(1) (1) 012414 010177      .IFF
(1) (1) 012414 010177      < MOVE! LDMA! IMM! << RTHRS+3>>>
(1) (1) 012414 010177      .ENDC
(1)
713 012416      MEMINC IMM, 2
(1) (1) 012416 000214      MICPC=MICPC+1
(1) (1) 012416 015402      < MOVE! WRMEM! INCMAR! IMM! < 2>>
(1)
714 012420      MEM     IMM, 0
(1) (1) 012420 000215      MICPC=MICPC+1
(1) (1) 012420 002400      < MOVE! WRMEM! IMM! < 0>>
    
```

G07

MAC: DDUMP PROTOCOL IMPLEMENTATION
DDNEW.MAC 06-DEC-76 10:14

MACY11 27 (1006) 07-MAR-77 16:24 PAGE 6-18
INWAIT---WAIT FOR RQI TO CLEAR

PAGE: 0024

(1)	715	012422	000216	OUTPUT MEMX SELB!OMODEM	:CLEAR DATA TERMINAL READY
(1)			042233	MICPC=MICPC+1	
(1)	716	012424	000217	<MOVE!WROUT!MEMX!<SELB!OMODEM>	
(1)			114527	ALWAYS RCEXX	:POST THE ERROR - FATAL
(1)				MICPC=MICPC+1	
(1)				<JUMP!ALCOND!<RCEXX-INIT\$3000*4>!<RCEXX-INIT\$777/2>	

```

718
719 012426      001
720
721      000
722      001
723 012426
(1) 012426      000220
(2) 012426      002654
(2)
725      000
726
727 012430      000221
(1)      001
(1) 012430      010240
(1)
(1)
(1)      000
(1)
728 012432      000222
(1)      001
(1)
(1)
(1) 012432      050220
(1)      000
(1)
729 012434      000223
(1)      123040
(1)
(1)
730 012436      000224
(1)      055302
(1)
(1)
731 012440      000225
(1)      001
(1)
(1)
(1) 012440      050220
(1)      000
(1)
732 012442      000226
(1)      074520
(1)
(1)
733
734
735 012444      000227
(1)      055224
(1)
(1)
736 012446

```

```

.SBTTL OUTINT---SET UP OUTPUT INTERRUPT [RDY0]
OUTINT:
  .IF NDF $LOW
  PSTATE PINT2
  .ENDC
  .IF DF $LOW
  PSTATE OUTWAIT          :PORT STATUS TO WAITING FOR OUT
  MEM IMM,<<OUTWAIT-INIT&777/2>>
  MICPC=MICPC+1
  <MOVE!WRMEM!IMM!<<OUTWAIT-INIT&777/2>>>
  .ENDC
                                :COMPLETION
  LDMA IMM,NXTINT          :ADDRESS OF NEXT INTERRUPT POINTER
  MICPC=MICPC+1
  .IF IDN IMM,IMM
  <MOVE!LDMAR!IMM!<NXTINT&377>>
  .IFF
  <MOVE!LDMAR!IMM!<NXTINT>>
  .ENDC
  LDMA MEMX,SELB          :NEXT INTERRUPT
  MICPC=MICPC+1
  .IF IDN MEMX,IMM
  <MOVE!LDMAR!IMM!<SELB&377>>
  .IFF
  <MOVE!LDMAR!MEMX!<SELB>>
  .ENDC
  SP IBUS,OCON,SPO        :READ THE OUTPUT CONTROL CSR
  MICPC=MICPC+1
  <MOVE!SPX!IBUS!OCON!SPO>
  OUT <MEMX!INCMAR>,<AORB!OCON> :WRITE THE OUT CONTROL CSR
  MICPC=MICPC+1
  <MOVE!WROUTX!MEMX!INCMAR!<AORB!OCON>>
  LDMA MEMX,SELB          :ADDRESS LINK
  MICPC=MICPC+1
  .IF IDN MEMX,IMM
  <MOVE!LDMAR!IMM!<SELB&377>>
  .IFF
  <MOVE!LDMAR!MEMX!<SELB>>
  .ENDC
  BRWRTE <BR!INCMAR>,<AA!SPO> :KICK PAST LINK STATUS BYTE
  MICPC=MICPC+1
  <MOVE!WRTEBR!BR!INCMAR!<AA!SPO>>
                                :SHIFT CSRO IMAGE LEFT
                                :***DO NOT CHANGE BR UNTIL BR7***
  OUT <MEMX!INCMAR>,<SELB!OPORT1> :WRITE LOW BYTE OF BA TO CSR
  MICPC=MICPC+1
  <MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT1>>
  OUT <MEMX!INCMAR>,<SELB!OPORT2> :WRITE HIGH BYTE OF BA TO CSR

```

```

(1) 012446 000230 MICPC=MICPC+1
(1) 012446 055225 <MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT2>>
(1) 737 012450 OUT <MEMX!INCMAR>,<SELB!OPORT4> ;WRITE HIGH BYTE OF COUNT TO CSR
(1) 012450 000231 MICPC=MICPC+1
(1) 012450 055227 <MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT4>>
(1) 738 012452 OUT <MEMX!INCMAR>,<SELB!OPORT3> ;WRITE THE LOW BYTE OF COUNT
(1) 012452 000232 MICPC=MICPC+1
(1) 012452 05522E <MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT3>>
(1) 739 ;***HERE IS BR7***
(1) 012454 BR7 PE1 ;INTERRUPT ENABLE IS SET
(1) 012454 000233 MICPC=MICPC+1
(1) 012454 103750 <JUMP!BR7CON!<PE1-INIT&3000*4>!<PE1-INIT&777/2>>
(1) 741 ;GENERATE AN INTERRUPT
(1) 742 001 .IF NDF $LOW
(1) 743 .ALWAYS $IDLE
(1) 744 FINT2: PSTATE $OUTWAIT
(1) 745 000 .ENDC
(1) 746 001 .IF DF $LOW
(1) 747 012456 PINT2: .ENDC
(1) 748 000 LDMA IMM,NXTINT ;ADDRESS NEXT INTERRUPT GUEJE
(1) 012456 000234 MICPC=MICPC+1
(1) 012456 001 .IF IDN IMM,IMM
(1) 012456 010240 <MOVE!LDMAR!IMM!<NXTINT&377>>
(1) 012456 000 <MOVE!LDMAR!IMM!<NXTINT>>
(1) 012456 000 .ENCC
(1) 750 012460 SP MEMX,SELB,SPO ;COPY ADDRESS FOR NEXT INT TO SPO
(1) 012460 000235 MICPC=MICPC+1
(1) 012460 043220 <MOVE!SPX!MEMX!SELB!SPO>
(1) 751 012462 MEM IMM,INTSTK ;ASSUME WRAP AROUND CASE
(1) 012462 000236 MICPC=MICPC+1
(1) 012462 002642 <MOVE!WRMEM!IMM!<INTSTK>>
(1) 752 012464 BRWRTE IMM,<<MMEND-2>> ;ADDRESS OF LAST INT IN STACK
(1) 012464 000237 MICPC=MICPC+1
(1) 012464 000776 <MOVE!WRTEBR!IMM!<<MMEND-2>>>
(1) 753 012466 CMP BR,SPO ;SHOULD WE WRAP
(1) 012466 000240 MICPC=MICPC+1
(1) 012466 060360 <SUBTC!BR!SPO>
(1) 754 012470 Z SS ;YES--BRANCH
(1) 012470 000241 MICPC=MICPC+1
(1) 012470 101644 <JUMP!ZCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
(1) 755 012472 BRWRTE IMM,2 ;OFFSET FOR NEXT POINTER
(1) 012472 000242 MICPC=MICPC+1
(1) 012472 000402 <MOVE!WRTEBR!IMM!<2>>
    
```

```

756 012474 MEM BR,ADD!SPO ;UPDATE POINTER
(1) 000243 MICPC=MICPC+1
(1) 012474 062400 <MOVE!WRMEM!BR!<ADD!SPC>>
(1)
757 012476 5$: SP MEMX,SELB,SPO ;COPY POINTER TO SPO
(1) 000244 MICPC=MICPC+1
(1) 012476 043220 <MOVE!SPX!MEMX!SELB!SPO>
(1)
758 012500 LDMA IMM,NXTSP ;PICK UP START OF IN QUEUE
(1) 000245 MICPC=MICPC+1
(1) 001 .IF IDN IMM,IMM
(1) 012500 010241 <MOVE!LDMAR!IMM!<NXTSP&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<NXTSP>>
(1) 000 .ENDC
(1)
759 012502 CMP MEMX,SPO ;COMPARE TO END
(1) 000246 MICPC=MICPC+1
(1) 012502 040360 <SUBTC!MEMX!SPO>
(1)
760 012504 Z 10$ ;IF EQUAL--CLEAR INT PENDING
(1) 000247 MICPC=MICPC+1
(1) 012504 101651 <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
(1)
761 012506 ALWAYS IDLE
(1) 000250 MICPC=MICPC+1
(1) 012506 100445 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
762 012510 10$: BRWRT IMM,357 ;MASK TO CLEAR INT PENDING
(1) 000251 MICPC=MICPC+1
(1) 012510 000757 <MOVE!WRTEBR!IMM!<357>>
(1)
763 012512 CLRIDL: SP BR,AANDB,SP1
(1) 000252 MICPC=MICPC+1
(1) 012512 063261 <MOVE!SPX!BR!AANDB!SP1>
(1)
764 012514 ALWAYS IDLE
(1) 000253 MICPC=MICPC+1
(1) 012514 100445 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
    
```

K07

```

766
767 012516 000254
(1) 012516 123440
(1)
(1)
768 012520 000255
(1) 012520 103525
(1)
(1)
769 012522 000256
(1) 012522 00050C
(1)
(1)
770 012524 000257
(1) 012524 0E1262
(1)
(1)
771 012526 000260
(1) 012526 100674
(1)
(1)
(1)

```

```

.SBTTL OUTWAI--WAIT FOR RDYO TO GO AWAY
OUTWAI: SPBR IBUS, OCON, SPD ;READ OUTPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!OCON!SPD>

BR7 NIDLE6 ;RDYO SET --GET OUT
MICPC=MICPC+1
<JUMP!BR7CON!<NIDLE6-INIT&3000*4>!<NIDLE6-INIT&777/2>>

BRWRTE IMM,100 ;CLEAR CONTROL BITS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<100>>

OUT BR, OCON!AANDB
MICPC=MICPC+1
<MOVE!WROUTX!BR!<OCON!AANDB>>

ALWAYS INS13
MICPC=MICPC+1
<JUMP!ALCOND!<INS13-INIT&3000*4>!<INS13-INIT&777/2>>

```

```

773
774 012530          .SBTTL  CTLSRV--CNTL I SERVICE
(1)                SPBR    IBUS,PORT4,SPO          ;TC SFC
(1) 012530          MICPC=MICPC+1
(1)                <MOVE!SPBRX!IBUS!PORT4!SPO>
775 012532          BRSHFT
(1)                MICPC=MICPC+1
(1) 012532          <MOVE!SHFTBR!WRTEBR!SELB>
776 012534          BR1     HDSEL                    ;IF SET IS HALF DUPLEX
(1)                MICPC=MICPC+1
(1) 012534          <JUMP!BR1CON!<HDSEL-INIT&3000*4>!<HDSEL-INIT&777/2>>
(1)
777 012536          BRWRTE IMM,100                  ;MASK FOR DTR
(1)                MICPC=MICPC+1
(1) 012536          <MOVE!WRTEBR!IMM!<100>>
(1)
778 012540          OUTPUT BR,SELB!OMODEM           ;TURN OFF HALF-DUPLEX
(1)                MICPC=MICPC+1
(1) 012540          <MOVE!WRROUT!BR!<SELB!OMODEM>>
(1)
779 012542          INS11: BRWRTE DP,<SELA!SPO>      ;RESTORE THE CNTL WORD
(1)                MICPC=MICPC+1
(1) 012542          <MOVE!WRTEBR!DP!<SELA!SPO>>
(1)
780 012544          BRO     CBOOT                    ;IF SET IS BOOT
(1)                MICPC=MICPC+1
(1) 012544          <JUMP!BROCON!<CBOOT-INIT&3000*4>!<CBOOT-INIT&777/2>>
(1)
781 012546          INS12: SP     IBUS,INCON,SPO      ;READ THE INPUT CONTROL CSR
(1)                MICPC=MICPC+1
(1) 012546          <MOVE!SPX!IBUS!INCON!SPO>
(1)
782 012550          BRWRTE IMM,100                  ;ZERO THE BR REGISTER EXCEPT INT ENABLE
(1)                MICPC=MICPC+1
(1) 012550          <MOVE!WRTEBR!IMM!<100>>
(1)
783 012552          OUT     BR,<AANDB!OINCON>         ;CLEAR IN CONTROL CSR
(1)                MICPC=MICPC+1
(1) 012552          <MOVE!WRROUTX!BR!<AANDB!OINCON>>
(1)
784 012554          LDMA   IMM,PRST                  ;ADDRESS PORT STATE
(1)                MICPC=MICPC+1
(1)                .IF IDN IMM,IMM
(1) 012554          <MOVE!LDMAR!IMM!<PRST&377>>
(1)                .IFF
(1)                <MOVE!LDMAR!IMM!<PRST>>
(1)                .ENDC
(1)                000
785 012556          INS13: PSTATE NIDLE2
(1) 012556          MEM     IMM,<<NIDLE2-INIT&777/2>>
(2)                MICPC=MICPC+1
(2) 012556          <MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>
(2)
786 012560          ALWAYS IDLE
(1)                MICPC=MICPC+1

```

```

(1) 012560 100445          <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
787
788 012562                CBOOT: BRWRT  IMM,200          :MASK FOR BOOT MODE
(1) 000276                MICPC=MICPC+1
(1) 012562 000600        <MOVE!WRTEBR!IMM!<200>>
(1)
789 012564                SP      BR,AORB,SP1          ;IN PORT STATUS WORD
(1) 000277                MICPC=MICPC+1
(1) 012564 063301        <MOVE!SPX!BR!AORB!SP1>
(1)
790 012566                BRWRT  IMM,204          :MASI FOR OK TO SEND AND LINE IDLE
(1) 000300                MICPC=MICPC+1
(1) 012566 000604        <MOVE!WRTEBR!IMM!<204>>
(1)
791 012570                SP      BR,SELB,SP10         ;IN LINE STATUS
(1) 000301                MICPC=MICPC+1
(1) 012570 063230        <MOVE!SPX!BR!SELB!SP10>
(1)
792 012572                ALWAYS INS12
(1) 000302                MICPC=MICPC+1
(1) 012572 100670        <JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
(1)

```



```

794
795 012574 000303          .SBTTL TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE
(1)          001          TBASRV: LDMA IMM,ETC          ;GET PCINTER TO END OF TMT CHAIN
(1) 012574 010072          MICPC=MICPC+1
(1)          000          .IF IDN IMM,IMM
(1)          000          <MOVE!LDMAR!IMM!<ETC&377>>
(1)          000          .IFF
(1)          000          <MOVE!LDMAR!IMM!<ETC>>
(1)          000          .ENDC

796 012576 000304          LDMA MEMX,<SELB!SPX!SPD>          ;FIND THE LINK
(1)          001          MICPC=MICPC+1
(1)          001          .IF IDN MEMX,IMM
(1)          001          <MOVE!LDMAR!IMM!<SELB!SPX!SPD&377>>
(1)          001          .IFF
(1) 012576 053220          <MOVE!LDMAR!MEMX!<SELB!SPX!SPD>>
(1)          000          .ENDC

797 012600 000305          MEMINC IMM,1          ;BUFFER ASSIGNED IN IN LINK FLAGS
(1)          016401          MICPC=MICPC+1
(1) 012600 016401          <MOVE!WRMEM!INCMAR!IMM!<1>>

798 012602 000306          BRWRT <IMM!INCMAR>,6          ;POINT PAST NUMBER FIELD
(1)          014406          MICPC=MICPC+1
(1) 012602 014406          <MOVE!WRTEBR!IMM!INCMAR!<6>>

799
800 012604 000307          MEMINC IBUS,PORT1          ;SET BR FOR ADDITION TO SPD
(1)          136500          MICPC=MICPC+1
(1) 012604 136500          <MOVE!WRMEM!INCMAR!IBUS!<PORT1>>

801 012606 000310          MEMINC IBUS,PORT2
(1)          136520          MICPC=MICPC+1
(1) 012606 136520          <MOVE!WRMEM!INCMAR!IBUS!<PORT2>>

802 012610 000311          MEMINC IBUS,PORT4
(1)          136560          MICPC=MICPC+1
(1) 012610 136560          <MOVE!WRMEM!INCMAR!IBUS!<PORT4>>

803 012612 000312          MEMINC IBUS,PORT3
(1)          136540          MICPC=MICPC+1
(1) 012612 136540          <MOVE!WRMEM!INCMAR!IBUS!<PORT3>>

804 012614 000313          SP BR,ADD,SPD          ;OFFSET TO NEXT TRANSMIT LINK
(1)          063000          MICPC=MICPC+1
(1) 012614 063000          <MOVE!SPX!BR!ADD!SPD>

805 012616 000314          BRWRT IMM,T          ;LOAD BR WITH ADDRESS OF CHAIN END
(1)          000553          MICPC=MICPC+1
(1) 012616 000553          <MOVE!WRTEBR!IMM!<T>>

806 012620 000315          LDMA IMM,ETC
(1)          001          MICPC=MICPC+1
(1)          001          .IF IDN IMM,IMM
(1) 012620 010072          <MOVE!LDMAR!IMM!<ETC&377>>
(1)          001          .IFF
(1)          001          <MOVE!LDMAR!IMM!<ETC>>
    
```

```

(1)          000          .ENDC
(1)
807 012622      CMP      BR,SPO          ;END OF CHAIN?
(1)          000316      MICPC=MICPC+1
(1) 012622      060360      <SUBTC!BR!SPO>
(1)
908 012624      Z        20$          ;IF YES--BRANCH
(1)          000317      MICPC=MICPC+1
(1) 012624      101724      <JUMP!ZCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
(1)
809 012626      MEM      BR,SELA!SPO      ;UPDATE THE END POINTER IN MEMORY
(1)          000320      MICPC=MICPC+1
(1) 012626      062600      <MOVE!WRMEM!BR!<SELA!SPO>>
(1)
810 012630      10$:     BRWRT  IMM,2          ;NUMBERED MSG PENDING MASK
(1)          000321      MICPC=MICPC+1
(1) 012630      000402      <MOVE!WRTBR!IMM!<2>>
(1)
811 012632      SP      BR,AORB,SP10      ;UPDATE LINE STATUS
(1)          000322      MICPC=MICPC+1
(1) 012632      063310      <MOVE!SPX!BR!AORB!SP10>
(1)
812 012634      ALWAYS  INS12
(1)          000323      MICPC=MICPC+1
(1) 012634      100670      <JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
(1)
813 012636      20$:     MEM      IMM,TML1          ;WRAP IT AROUND
(1)          000324      MICPC=MICPC+1
(1) 012636      002473      <MOVE!WRMEM!IMM!<TML1>>
(1)
814 012640      ALWAYS  10$          ;CONTINUE PROCESSING
(1)          000325      MICPC=MICPC+1
(1) 012640      100721      <JUMP!ALCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
(1)

```

```

816
817 012642 000326
(1) 001
(1) 012642 010023
(1)
(1)
(1) 000
(1)
818 012644 000327
(1) 001
(1)
(1)
(1) 012644 053220
(1) 000
(1)
819 012646 000330
(1) 012646 016401
(1)
820 012650 000331
(1) 012650 136500
(1)
821 012652 000332
(1) 012652 136520
(1)
822 012654 000333
(1) 012654 136560
(1)
823 012656 000334
(1) 012656 136540
(1)
824
825 012660 000335
(1) 001
(1) 012660 010023
(1)
(1)
(1) 000
(1)
826 012662 000336
(1) 012662 002425
(1)
827 012664 000337
(1) 012664 000463
(1)
828 012666 000340
(1)

```

```

.SBTTL RBASRV--RECEIVE BUFFER ADDRESS SERVICE
RBASRV: LDMA IMM,ERC ;ADDRES END OF RECEIVE CHAIN
MICPC=MICPC+1
.IF IDN IMM IMM
<MOVE!LDMAR!IMM!<ERC&377>>
.IFF
<MOVE!LDMAR!IMM!<ERC>>
.ENDC

LDMA MEMX,<SELB!SPX!SPO> ;GET THE POINTER TO LINK
MICPC=MICPC+1
.IF IDN MEMX IMM
<MOVE!LDMAR!IMM!<SELB!SPX!SPO&377>>
.IFF
<MOVE!LDMAR!MEMX!<SELB!SPX!SPO>>
.ENDC

MEMINC IMM,1
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<1>>

MEMINC IBUS,PORT1
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT1>>

MEMINC IBUS,PORT2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT2>>

MEMINC IBUS,PORT4
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT4>>

MEMINC IBUS,PORT3
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT3>>

:::NOTE INVERTED ORDER OF PORT 3 AND PORT4
LDMA IMM,ERC
MICPC=MICPC+1
.IF IDN IMM IMM
<MOVE!LDMAR!IMM!<ERC&377>>
.IFF
<MOVE!LDMAR!IMM!<ERC>>
.ENDC

MEM IMM,RCL1 ;ASSUME WRAP AROUND CASE
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<RCL1>>

BRWRITE IMM,RCL7 ;GET ADDRESS OF END OF CHAIN AREA
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCL7>>

CMP BR,SPO ;CHECK FOR END
MICPC=MICPC+1

```

008

```

(1) 012666 060363          <SUBTC!BR!SPO>
(1)
829 012670                Z      INS12          ;IF EQUAL BRANCH
(1) 012670 000341          MICPC=MICPC+1
(1) 012670 101670          <JUMP!ZCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
(1)
830 012672                BRWRTE IMM,5          ;CALCULATE ADDRESS OF NEXT LINK
(1) 012672 000342          MICPC=MICPC+1
(1) 012672 000405          <MOVE!WRTEBR!IMM!<5>>
(1)
931 012674                MEM      BR,ADD!SPO          ;...
(1) 012674 000343          MICPC=MICPC+1
(1) 012674 062400          <MOVE!WRMEM!BR!<ADD!SPO>>
(1)
832 012676                ALWAYS INS12          ;EXIT
(1) 012676 000344          MICPC=MICPC+1
(1) 012676 100670          <JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
(1)
833 012700                RA1:  BRWRTE IMM,317          ;MASK TO CLEAR START MODE AND CLR ACTIVE
(1) 012700 000345          MICPC=MICPC+1
(1) 012700 000717          <MOVE!WRTEBR!IMM!<317>>
(1)
834 012702                SPBR    BR,AANDB,SP10          ;CLEAR BIT IN LINE STATUS WORD
(1) 012702 000346          MICPC=MICPC+1
(1) 012702 063670          <MOVE!SPBRX!BR!AANDB!SP10>
(1)
835 012704                RA3:  BRWRTE IMM,0          ;CLEAR BR
(1) 012704 000347          MICPC=MICPC+1
(1) 012704 000400          <MOVE!WRTEBR!IMM!<0>>
(1)
836 012706                SP      BR,SELB,SP13          ;SET NUMB MESSAGE TYPE IN SP13
(1) 012706 000350          MICPC=MICPC+1
(1) 012706 063233          <MOVE!SPX!BR!SELB!SP13>
(1)
837 012710                STATE  RCVB          ;CHANGE RECEIVE STATE POINTER TO STATE B
(1) 012710 000351          MICPC=MICPC+1
(1) 012710 000424          <MOVE!WRTEBR!IMM!<RCVB-INIT&777/2>>
838 012712                ALWAYS REXIT
(1) 012712 000352          MICPC=MICPC+1
(1) 012712 104422          <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
(1)
839
840 012714                RTHRES: BRWRTE IMM,2
(1) 012714 000353          MICPC=MICPC+1
(1) 012714 000402          <MOVE!WRTEBR!IMM!<2>>
(1)
841 012716                ALWAYS ERRXX
(1) 012716 000354          MICPC=MICPC+1
(1) 012716 114671          <JUMP!ALCOND!<ERRXX-INIT&3000*4>!<ERRXX-INIT&777/2>>
(1)
842
843                001
844                RCVCX:  ;IF NDF LOW
(1) 012716 114671          SPBR    IBUS,RCVCON,SPO          ;READ RCVR CONTROL CSR
(1) 012716 114671          BRWRTE BR,ADD!SPO          ;SHIFT LEFT
(1) 012716 114671          BR7    I1
(1) 012716 114671          ALWAYS TA1
  
```

E08

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 06-DEC-76 10:14

MACY11 27(1006) 07-MAR-77 16:24 PAGE 6-29
RBASRV--RECEIVE BUFFER ADDRESS SERVICE

PAGE: 0095

048
049
050
051
052
053
054

000

ACK:	BRWRT	BR,AA!SP10	:READ LINE STATUS SHIFTING LEFT
	BR4	SS	:IF START RECD--CLEAR START MODE
	ALWAYS	IDLE	
SS:	BRWRT	IMM,327	:CLEAR START MODE
	SP	BR.AANDB,SP10	:IN LINE STATUS
	ALWAYS	IDLE	
	.ENDC		

```

856 012720      000355      MOSEL: BRWRT  IMM,100      :HC MASK TC BR
      (1)      000500      MICPC=MICPC+1
      (1) 012720      000500      <MOVE!WRTEBR!IMM!<100>>
      (1)
857 012722      000356      SP      BR,AORB,SP10      :UPDATE PORT STATUS WORD
      (1)      063310      MICPC=MICPC+1
      (1) 012722      063310      <MOVE!SPX!BR!AORB!SP10>
      (1)
858 012724      000357      ALWAYS  INS11
      (1)      100666      MICPC=MICPC+1
      (1) 012724      100666      <JUMP!ALCOND!<INS11-INIT&3000*4>!<INS11-INIT&777/2>>
      (1)
859
860 012726      000360      PE1:  BRWRT  IMM,300      :MASK FOR INTERRUPT AND VECTOR THROUGH XCH
      (1)      000700      MICPC=MICPC+1
      (1) 012726      000700      <MOVE!WRTEBR!IMM!<300>>
      (1)
861 012730      000361      SP      IBUS,UBBR,SPO      :READ BR CONTROL REG
      (1)      123220      MICPC=MICPC+1
      (1) 012730      123220      <MOVE!SPX!IBUS!UBBR!SPO>
      (1)
862 012732      000362      OUT    BR,<AORB!OBR>      :INTERRUPT
      (1)      061311      MICPC=MICPC+1
      (1) 012732      061311      <MOVE!WROUTX!BR!<AORB!OBR>>
      (1)
863      001      .IF NDF SLOW
864      000      ALWAYS  IDLE
865      000      .ENDC
866      001      .IF DF  SLOW
867 012734      000363      ALWAYS  PINT2
      (1)      100634      MICPC=MICPC+1
      (1) 012734      100634      <JUMP!ALCOND!<PINT2-INIT&3000*4>!<PINT2-INIT&777/2>>
      (1)
868      000      .ENDC
869
870 012736      000364      HALTED: MEMADR  EM6
      (1)      001      MICPC=MICPC+1
      (1) 012736      002730      .IF B
      (1)      000      <MOVE!WRMEM!<EM6-INIT&777/2>>
      (1)      000      .IFF
      (1)      000      <MOVE!WRMEM!!<EM6-INIT&777/2>>
      (1)      000      .ENDC
871
872 012740      000365      ACLOW: BRWRT  IMM,2      :FALL INTO ACLOW
      (1)      000402      MICPC=MICPC+1      :CAUSE AN AC LOW
      (1) 012740      000402      <MOVE!WRTEBR!IMM!<2>>
      (1)
873 012742      000366      OUT    BR,<SELB!OBR>
      (1)      061231      MICPC=MICPC+1
      (1) 012742      061231      <MOVE!WROUTX!BR!<SELB!OBR>>
      (1)
874 012744      000367      SS:  BRWRT  IBUS,UBBR      :WAIT FOR IT TO COMPLETE
      (1)      120620      MICPC=MICPC+1
      (1) 012744      120620      <MOVE!WRTEBR!IBUS!<UBBR>>
      (1)
875 012746      BR1      55

```

```

(1) 012746 000370 MICPC=MICPC+1
(1) 012746 102757 <JUMP!BR1CON!<SS-INIT&3000*4>!<SS-INIT&777/2.>
(1) 012750 000371 .ALWAY MEMX SELB.PAGE3
(1) 012750 154620 MICPC=MICPC+1
(1) 012750 154620 <JUMP!ALCOND!MEMX!SELB!PAGE3>
(1) 012752 000372 CKTIME: BRWRTE IBUS UBBR ;READ BR CONTROL REG
(1) 012752 120620 MICPC=MICPC+1
(1) 012752 120620 <MOVE!WRTEBR!IBUS!<UBBR>>
(1) 012754 000373 BR4 HALTED
(1) 012754 103364 MICPC=MICPC+1
(1) 012754 103364 <JUMP!BR4CON!<HALTED-INIT&3000*4>!<HALTED-INIT&777/2.>
(1) 012756 000374 ALWAYS EM1
(1) 012756 114733 MICPC=MICPC+1
(1) 012756 114733 <JUMP!ALCOND!<EM1-INIT&3000*4>!<EM1-INIT&777/2.>
(1) 012760 000375 tbus: BRWRTE IBUS NPR
(1) 012760 120600 MICPC=MICPC+1
(1) 012760 120600 <MOVE!WRTEBR!IBUS!<NPR>>
(1) 012762 000376 BRO IDLE
(1) 012762 102045 MICPC=MICPC+1
(1) 012762 102045 <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2.>
(1) 012764 000377 ALWAYS EC2
(1) 012764 114760 MICPC=MICPC+1
(1) 012764 114760 <JUMP!ALCOND!<EC2-INIT&3000*4>!<EC2-INIT&777/2.>
(1) 012764 114760
884

```

```

886      012766      .=INIT+1000
887      000377      MICPC=377
888      .SBTTL RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER
889      :ENTERED FROM IDLE LOOP
890      :DETERMINES IF MESSAGE TYPE IS NUMBERED, UNNUMBERED OR BOOT
891      :SETS UP APROPRIATE STATES FOR REST OF MESSAGE.
892 012766      RCVA: SP      IBUS,RCVDAT,SPD      :READ RECEIVE CHARACTE R TO SPD
(1)      00043C      MICPC=MICPC+1
(1) 012766      02320C      <MOVE!SPX!IBUS!RCVDAT!SPD
(1)
893 012770      BRWRTE BR,SELA!SP1      :READ PORT STATUS WORD
(1)      000401      MICPC=MICPC+1
(1) 012770      060601      <MOVE!WRTEBR!BR!<SELA!SP1>>
(1)
894 012772      BRO      S$      :IF INIT MODE---ONLY BOOT OK
(1)      000402      MICPC=MICPC+1
(1) 012772      106012      <JUMP!BROCON!<S$-INIT&3000*4>!<S$-INIT&777/2>>
(1)
895 012774      BR7      S$      ;IF BOOT MODE---ONLY BOOT OK
(1)      000403      MICPC=MICPC+1
(1) 012774      107412      <JUMP!BR7CON!<S$-INIT&3000*4>!<S$-INIT&777/2>>
(1)
896 012776      BRWRTE IMM,201      :SOH TO BR
(1)      000404      MICPC=MICPC+1
(1) 012776      000601      <MOVE!WRTEBR!IMM!<201>>
(1)
897 013000      CMP      BR,SPD      :COMPARE BR TO SPD
(1)      000405      MICPC=MICPC+1
(1) 013000      060360      <SUBTC!BR!SPD>
(1)
898 013002      Z      RA1      :IF EQUAL-IS NUMBERED MESSAGE
(1)      000406      MICPC=MICPC+1
(1) 013002      101745      <JUMP!ZCOND!<RA1-INIT&3000*4>!<RA1-INIT&777/2>>
(1)
899 013004      BRWRTE IMM,5      :ENQ TO BR
(1)      000407      MICPC=MICPC+1
(1) 013004      000405      <MOVE!WRTEBR!IMM!<5>>
(1)
900 013006      CMP      BR,SPD      :COMPARE ENQ TO SPD
(1)      000410      MICPC=MICPC+1
(1) 013006      060360      <SUBTC!BR!SPD>
(1)
901 013010      Z      RA2      ;IF EQUAL-IS UNNUMBERED MESSAGE
(1)      000411      MICPC=MICPC+1
(1) 013010      105421      <JUMP!ZCOND!<RA2-INIT&3000*4>!<RA2-INIT&777/2>>
(1)
902 013012      S$: BRWRTE IMM,220      ;DLE TO BR
(1)      000412      MICPC=MICPC+1
(1) 013012      000620      <MOVE!WRTEBR!IMM!<220>>
(1)
903 013014      CMP      BR,SPD      :COMPARE DLE TO SPD
(1)      000413      MICPC=MICPC+1
(1) 013014      060360      <SUBTC!BR!SPD>
(1)
904 013016      Z      BOOT      :IF EQUAL IS BOOT
(1)      000414      MICPC=MICPC+1
  
```



```

(1) 013016 105762          <JUMP!ZCOND!<BOOT-INIT&3000*4>!<BOOT-INIT&777/2>>
(1)
905 013020 000415          FLUSH:  OUTPUT IMM,<200!ORCVCC>          ;FLUSH THE INPUT SILC
(1) 013020 002212          MICPC=MICPC+1
(1) 013020 002212          <MOVE!WR0UT!IMM!<200!ORCVCC>>
(1)
906 013022 000416          BRWRTE IMM,357          ;MASK TO CLEAR--CLEAR ACTIVE
(1) 013022 000757          MICPC=MICPC+1
(1) 013022 000757          <MOVE!WRTEBR!IMM!<357>>
(1)
907 013024 000417          SP      BR,AANDB,SP10          ;IN LINE STATUS WORD
(1) 013024 06327C          MICPC=MICPC+1
(1) 013024 06327C          <MOVE!SPX!BR!AANDB!SP10>
(1)
908 013026 00042C          ALWAYS RM1          ;SET STATE TO RCVA AND RETURN TO IDLE
(1) 013026 114662          MICPC=MICPC+1
(1) 013026 114662          <JUMP!ALCOND!<RM1-INIT&3000*4>!<RM1-INIT&777/2>>
(1)
909 013030 000421          RA2:  STATE  RCVI          ;CHANGE RECEIVE STATE TO I
(1) 013030 000700          MICPC=MICPC+1
(1) 013030 000700          <MOVE!WRTEBR!IMM!<RCVI-INIT&777/2>>
(1) 013030 001          .IF NDF LOW
(1) 013030 000          ALWAYS REXIT
(1) 013030 001          .ENDC
(1) 013030 001          .IF DF  LOW
914 013032 000422          REXIT: SP      BR,SELB,SP3
(1) 013032 063223          MICPC=MICPC+1
(1) 013032 063223          <MOVE!SPX!BR!SELB!SP3>
(1)
915 013034 000423          ALWAYS IDLE
(1) 013034 10044E          MICPC=MICPC+1
(1) 013034 10044E          <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
916 013034 000          .ENDC
  
```

```

918 .SBTTL RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD
919 :ENTERED FROM IDLE LOOP
920 :STORES COUNT FIELD AND SETS UP RCVB AS NEXT STATE
921 :
922 013036 RCVB: SP IBUS,RCVDAT,SP4 :READ CHARACTER TO SP4
923 013036 MICPC=MICPC+1
(1) 013036 000424 <MOVE!SPX!IBUS!RCVDAT!SP4>
(1) 013036 023204
924 013040 LDMA IMM,LRC ;LOAD ADDRESS OF START OF RECV CHAIN
(1) 013040 070425 MICPC=MICPC+1
(1) 013040 001 .IF IDN IMM,IMM
(1) 013040 010024 <MOVE!LDMAR!IMM!<LRC&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<LRC>>
(1) .ENDC
925 013042 SP MEMX!LDMAR,SELB,SP14 ;COPY POINTER TO SP14
(1) 013042 000426 MICPC=MICPC+1
(1) 013042 053234 <MOVE!SPX!MEMX!LDMAR!SELB!SP14>
926 013044 :AND LOAD MAR WITH ADDRESS OF CURRENT BA
927 013044 BRWRT MEMX,INCMAR!SELB ;READ FLAGS BYTE
(1) 013044 000427 MICPC=MICPC+1
(1) 013044 054620 <MOVE!WRTEBR!MEMX!<INCMAR!SELB>>
928 013046 BR0 RB1 ;RECV BUFFER ASSIGNED---CONTINUE
(1) 013046 000430 MICPC=MICPC+1
(1) 013046 106042 <JUMP!BR0CON!<RB1-INIT&3000*4>!<RB1-INIT&777/2>>
929 013050 BRWRT BR,SELA!SP1 ;READ STATUS BYTE
(1) 013050 000431 MICPC=MICPC+1
(1) 013050 060601 <MOVE!WRTEBR!BR!<SELA!SP1>>
930 013052 BR7 RB3 ;MAINT MODE
(1) 013052 000432 MICPC=MICPC+1
(1) 013052 107440 <JUMP!BR7CON!<RB3-INIT&3000*4>!<RB3-INIT&777/2>>
931 013054 LDMA IMM,T ;ERROR--LOAD TYPE FIELD ADDRESS IN MAR
(1) 013054 000433 MICPC=MICPC+1
(1) 013054 001 .IF IDN IMM,IMM
(1) 013054 010153 <MOVE!LDMAR!IMM!<T&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<T>>
(1) .ENDC
932 013056 MEMINC IMM,2 ;LOAD NAK TYPE
(1) 013056 000434 MICPC=MICPC+1
(1) 013056 016402 <MOVE!WRMEM!INCMAR!IMM!<2>>
933 013060 MEM IMM,310 ;LOAD SUB-TYPE NO BUFFERS
(1) 013060 000435 MICPC=MICPC+1
(1) 013060 002710 <MOVE!WRMEM!IMM!<310>>
934 013062 LDMA IMM,NTLS
(1) 013062 000436 MICPC=MICPC+1

```

K08

```

(1)          001
(1) 013062 013012      .IF IDN IMM,IMM
(1)          000      <MOVE!LDMAR!IMM!<NTLS&377>>
(1)          000      .IFF
(1)          000      <MOVE!LDMAR!IMM!<NTLS>>
(1)          000      .ENDC

935 013064      ALWAYS RHS          :BRANCH TO SEND NAK ROUTINE
(1)          000437      MICPC=MICPC+1
(1) 013064 104557      <JUMP!ALCOND!<RHS-INIT&3000*4>!<RHS-INIT&777/2>>
(1)
936 013066      RB3: BRWRTE IMM,4      :MASK FOR NO BUFFER AVAILABLE
(1)          000440      MICPC=MICPC+1
(1) 013066 000404      <MOVE!WRTEBR!IMM!<4>>
(1)
937 013070      SP BR,AORB,SP1        :SET THE FLAG
(1)          000441      MICPC=MICPC+1
(1) 013070 063301      <MOVE!SPX!BR!AORB!SP1>
(1)
938 013072      RB1: STATE RCVC
(1)          000442      MICPC=MICPC+1
(1) 013072 000462      <MOVE!WRTEBR!IMM!<RCVC-INIT&777/2>>
939 013074      RBC: SP BR,SELB,SP3
(1)          000443      MICPC=MICPC+1
(1) 013074 063223      <MOVE!SPX!BR!SELB!SP3>
(1)
940 013076      OUTPUT <MEMX!INCMAR>,<SELB!OBA1> ;OUTPUT LOW ORDER BYTE OF ADDRESS
(1)          000444      MICPC=MICPC+1
(1) 013076 056226      <MOVE!WROUT!MEMX!INCMAR!<SELB!OBA1>>
(1)
941 013100      . OUTPUT MEMX!INCMAR,<SELB!OBA2> ;OUTPUT HIGH BYTE OF ADDRESS
(1)          000445      MICPC=MICPC+1
(1) 013100 056227      <MOVE!WROUT!MEMX!INCMAR!<SELB!OBA2>>
(1)
942 013102      SP IBUS,UBBR,SPO      :READ THE BUS REQ REGISTER
(1)          000446      MICPC=MICPC+1
(1) 013102 123220      <MOVE!SPX!IBUS!UBBR!SPO>
(1)
943 013104      BRWRTE IMM,101        :MASK OFF ALL BUT NXM AND VEC4 BITS
(1)          000447      MICPC=MICPC+1
(1) 013104 000501      <MOVE!WRTEBR!IMM!<101>>
(1)
944 013106      SP BR,AANDB,SPO      :AND SAVE IN SPO
(1)          000450      MICPC=MICPC+1
(1) 013106 063260      <MOVE!SPX!BR!AANDB!SPO>
(1)
945 013110      SP IMM,300,SP5        :MASK TO ISOLATE EX. MEM BITS
(1)          000451      MICPC=MICPC+1
(1) 013110 003305      <MOVE!SPX!IMM!300!SP5>
(1)
946          :NOTE THIS REALLY WRITES A 305 BUT THE
947          :S GETS SHIFTED OUT
948 013112      BRWRTE MEMX,AANDB!SP5 ;MASK ALL BUT EX. MEM BITS
(1)          000452      MICPC=MICPC+1
(1) 013112 040665      <MOVE!WRTEBR!MEMX!<AANDB!SP5>>
(1)
949 013114      BRSHFT                :SHIFT THEM INTO THE CORRECT POSITION

```

L08

DMS1: DDCMP PROTOCOL IMPLEMENTATION
 DCCNEW.MAC 06-DEC-76 10:14

MACY11 27(1006) 07-MAR-77 16:24 PAGE 6-36
 RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD

PAGE: 0102

```

(1)          000453          MICPC=MICPC+1
(1) 013114 001620          <MOVE!SHFTBR!WRTEBR!SELB>
(1)
950 013116          BRSHFT
(1)          000454          MICPC=MICPC+1
(1) 013116 001620          <MOVE!SHFTBR!WRTEBR!SELB>
(1)
951 013120          BRSHFT
(1)          000455          MICPC=MICPC+1
(1) 013120 001620          <MOVE!SHFTBR!WRTEBR!SELB>
(1)
952 013122          BRSHFT
(1)          000456          MICPC=MICPC+1
(1) 013122 001620          <MOVE!SHFTBR!WRTEBR!SELB>
(1)
953 013124          OUT      BR,AORB!OBR          ;WRITE EX MEM BITS OUT
(1)          000457          MICPC=MICPC+1
(1) 013124 061311          <MOVE!WROUTX!BR!<AORB!OBR>>
(1)
954 013126          ALWAYS IDLE
(1)          000460          MICPC=MICPC+1
(1) 013126 100445          <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
955 013130          RB2:    ALWAYS I2
(1)          000461          MICPC=MICPC+1
(1) 013130 100454          <JUMP!ALCOND!<I2-INIT&3000*4>!<I2-INIT&777/2>>
(1)
(1)
(1)

```

M08

```

957          .SBTTL RCVC--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD, SELECT AND FINA
958          ;ENTERED FROM IDLE LOOP
959          ;INTERPRETS SELECT AND FINAL
960          ;CHECKS FOR COUNT TOO LARGE
961
962 013132    RCVC: ALWAYS SELQSY          ;"CALL" SELECT/QSYNC SUBROUTINE
(1)         000462 MICPC=MICPC+1
(1) 013132  11447C <JUMP!ALCOND!<SELQSY-INIT&3000*4>!<SELQSY-INIT&777/2>>
(1)
963 013134    LDMA BR <SELA!SP14>          ;LOAD ADDRESS OF CURRENT COUNT
(1)         000463 MICPC=MICPC+1
(1)         001 .IF IDN BR IMM
(1)         <MOVE!LDMAR!IMM!<SELA!SP14&377>>
(1)         .IFF
(1) 013134  070214 <MOVE!LDMAR!BR!<SELA!SP14>>
(1)         000 .ENDC
(1)
964 013136    BRWRT BR!INCMAR,SELA!SP1     ;READ STATUS BYTE
(1)         000464 MICPC=MICPC+1
(1) 013136  074601 <MOVE!WRTBR!BR!INCMAR!<SELA!SP1>>
(1)
965 013140    BRWRT SHFTBR!INCMAR,SELB     ;SHIFT IT RIGHT
(1)         000465 MICPC=MICPC+1
(1) 013140  015620 <MOVE!WRTBR!SHFTBR!INCMAR!<SELB>>
(1)
966 013142    BR1 RCS                      ;NO BUFFER ASSIGNED IN MAINT MODE
(1)         000466 MICPC=MICPC+1
(1) 013142  106475 <JUMP!BRCON!<RCS-INIT&3000*4>!<RCS-INIT&777/2>>
(1)
967 013144    BRWRT IMM!INCMAR,77         ;MASK FOR COUNT BITS
(1)         000467 MICPC=MICPC+1
(1) 013144  014477 <MOVE!WRTBR!IMM!INCMAR!<77>>
(1)
968 013146    SP MEMX!INCMAR,SELB,SPD     ;COPY HIGH BYTE OF COUNT TO SPD
(1)         000470 MICPC=MICPC+1
(1) 013146  057220 <MOVE!SPX!MEMX!INCMAR!SELB!SPD>
(1)
969 013150    BRWRT BR,AANDB!SPD          ;MASK TO BR
(1)         000471 MICPC=MICPC+1
(1) 013150  060660 <MOVE!WRTBR!BR!<AANDB!SPD>>
(1)
970 013152    CMP BR,SP5                  ;COMPARE HIGH ORDER BITS OF COUNT
(1)         000472 MICPC=MICPC+1
(1) 013152  060365 <SUBTC!BR!SP5>
(1)
971 013154    C RCFATL                    ;IF CARRY--TOO BIG ERROR
(1)         000473 MICPC=MICPC+1
(1) 013154  115116 <JUMP!CCOND!<RCFATL-INIT&3000*4>!<RCFATL-INIT&777/2>>
(1)
972 013156    Z RCLW                      ;IF EQUAL COMPARE LOW ORDER BITS OF COUNT
(1)         000474 MICPC=MICPC+1
(1) 013156  115513 <JUMP!ZCOND!<RCLW-INIT&3000*4>!<RCLW-INIT&777/2>>
(1)
973 013160    RCS: STATE RCVD            ;SET NEXT STATE TO D
(1)         000475 MICPC=MICPC+1
(1) 013160  000477 <MOVE!WRTBR!IMM!<RCVD-INIT&777/2>>

```

N08

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 06-DEC-76 10:14

MACY11 27(1006) 07-MAR-77 16:24 PAGE 6-38
RCVC--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD, SELECT AND FINAL

PAGE: 0104

974 013162
(1) 000476
(1) 013162 104422
(1)

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

```

976 .SBTTL RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES
977
978 013164 RCVD: SP BR,DECA,SP4 ;DECREMENTLOW BYTE OF COUNT
(1) 000477 MICPC=MICPC+1
(1) 013164 063164 <MOVE!SPX!BR!DECA!SP4>
(1)
979 013166 C RD3 ;NO OVERFLOW
(1) 000500 MICPC=MICPC+1
(1) 013166 105102 <JUMP!CCOND!<RD3-INIT&3000*4>!<RD3-INIT&777/2>>
(1)
980 013170 SP BR,DECA,SP5 ;OVERFLOW - DECREMENT HIGH BYTE
(1) 000501 MICPC=MICPC+1
(1) 013170 063165 <MOVE!SPX!BR!DECA!SP5>
(1)
981 013172 RD3: STATE RCVE
(1) 000502 MICPC=MICPC+1
(1) 013172 000523 <MOVE!WRTEBR!IMM!<RCVE-INIT&777/2>>
982 013174 RD2: SP BR,SELB,SP3 ;SAVE THE STATE
(1) 000503 MICPC=MICPC+1
(1) 013174 063223 <MOVE!SPX!BR!SELB!SP3>
(1)
983 013176 SPBR IBUS,RCVDAT,SPO ;INPUT THE CHARACTER
(1) 000504 MICPC=MICPC+1
(1) 013176 023600 <MOVE!SPBRX!IBUS!RCVDAT!SPO>
(1)
984 013200 BRWRTE BR,SUB!SP17 ;COMPARE NEW R TO LAST R
(1) 000505 MICPC=MICPC+1
(1) 013200 060757 <MOVE!WRTEBR!BR!<SUB!SP17>>
(1)
985 013202 BR7 10$ ;IF NEW IS GREATER---PROCESS
(1) 000506 MICPC=MICPC+1
(1) 013202 107510 <JUMP!BR7CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
(1)
986 013204 ALWAYS IDLE
(1) 000507 MICPC=MICPC+1
(1) 013204 100445 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
987 013206 10$: BRWRTE BR,SELA!SP1 ;READ STATUS BYTE
(1) 000510 MICPC=MICPC+1
(1) 013206 060601 <MOVE!WRTEBR!BR!<SELA!SP1>>
(1)
988 013210 BR7 IDLE ;MAINT. MODE - GET JUT
(1) 000511 MICPC=MICPC+1
(1) 013210 103445 <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
989 013212 BRWRTE BR,SELA!SP10
(1) 000512 MICPC=MICPC+1
(1) 013212 060610 <MOVE!WRTEBR!BR!<SELA!SP10>>
(1)
990 013214 BRSHFT
(1) 000513 MICPC=MICPC+1
(1) 013214 0C1620 <MOVE!SHFTBR!WRTEBR!SELB>
(1)
991 013216 BR4 IDLE
(1) 000514 MICPC=MICPC+1
(1) 013216 103045 <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

```

```

(1)
992 013220          LDMA  IMM,ISP17          ;ADDRESS LAST ACKED IMAGE
(1)                MICPC=MICPC+1
(1)                .IF IDN IMM,IMM
(1) 013220 010155  <MOVE!LDMAR!IMM!<ISP17&377>>
(1)                .IFF
(1)                <MOVE!LDMAR!IMM!<ISP17>>
(1)                .ENDC
(1)                000
(1)
993 013222          MEM  BR,SELA!SPO          ;COPY THE CHAR
(1)                MICPC=MICPC+1
(1) 013222 062600  <MOVE!WRMEM!BR!<SELA!SPO>>
(1)
994 013224          RDS: BRWRTE IMM,REPST!LDMAR ;SET UP COUNT FOR TIMER
(1)                MICPC=MICPC+1
(1) 013224 010403  <MOVE!WRTEBR!IMM!<REPST!LDMAR>>
(1)
995                ;****DEPENDENT ON REPST = 2
996 013226          MEM  IMM,1              ;RESET REP THRESHOLD
(1)                MICPC=MICPC+1
(1) 013226 002401  <MOVE!WRMEM!IMM!<1>>
(1)
997 013230          SP  BR,SELB,SP15         ;RESET THE COUNT
(1)                MICPC=MICPC+1
(1) 013230 063235  <MOVE!SPX!BR!SELB!SP15>
(1)
998 013232          ALWAYS IDLE
(1)                MICPC=MICPC+1
(1) 013232 100445  <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
    
```



```

1000 .SBTTL RCVE--ROUTINE TO HANDLE N FIELD OF NUMBERED MESSAGE
1001
1002 013234 000523 RCVE: BRWRT BR, SELA!SP1 ;READ THE STATUS BYTE
      (1) 060601 MICPC=MICPC+1
      (1) 013234 060601 <MOVE!WRTEBR!BR!<SELA!SP1>>
      (1)
1003 013236 000524 BR7 RCVQ
      (1) 000524 MICPC=MICPC+1
      (1) 013236 107713 <JUMP!BR7CON!<RCVQ-INIT&3000*4>!<RCVQ-INIT&777/2>>
      (1)
1004 013240 000525 BRWRT IBUS,RCV DAT ;INPUT THE CHARACTER
      (1) 000525 MICPC=MICPC+1
      (1) 013240 020600 <MOVE!WRTEBR!IBUS!<RCV DAT>>
      (1)
1005 013242 000526 CMP BR,SP11
      (1) 000526 MICPC=MICPC+1
      (1) 013242 060371 <SUBTC!BR!SP11>
      (1)
1006 013244 000527 Z S$
      (1) 000527 MICPC=MICPC+1
      (1) 013244 105532 <JUMP!ZCOND!<S$-INIT&3000*4>!<S$-INIT&777/2>>
      (1)
1007 013246 000530 SP BR,DECA,SP13 ;FORCE MSG TYPE TO -1
      (1) 000530 MICPC=MICPC+1
      (1) 013246 063173 <MOVE!SPX!BR!DECA!SP13>
      (1)
1008 013250 000531 ALWAYS RE2
      (1) 000531 MICPC=MICPC+1
      (1) 013250 104533 <JUMP!ALCOND!<RE2-INIT&3000*4>!<RE2-INIT&777/2>>
      (1)
1009 013252 000532 S$: SP BR,INCA,SP11 ;UPDATE R FIELD
      (1) 000532 MICPC=MICPC+1
      (1) 013252 063071 <MOVE!SPX!BR!INCA!SP11>
      (1)
1010 013254 000533 RE2: STATE RCVF ;NEXT RECEIVE STATE IS F
      (1) 000533 MICPC=MICPC+1
      (1) 013254 000535 <MOVE!WRTEBR!IMM!<RCVF-INIT&777/2>>
      (1) 013256 000534 ALWAYS REXIT
      (1) 000534 MICPC=MICPC+1
      (1) 013256 104422 <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
      (1)

```

1013
 1014 013260
 (1) 000535
 (1) 013260 000540
 1015 013262
 (1) 000536
 (1) 013262 020200
 (1)
 1016 013264
 (1) 000537
 (1) 013264 104422
 (1)
 1017
 1018
 1019
 1020 013266
 (1) 000540
 (1) 013266 000542
 1021 013270
 (1) 000541
 (1) 013270 104536
 (1)

```

.SBTTL RCVF--ROUTINE TO IGNORE ADDRESS
RCVF: STATE RCVG
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVG-INIT&777/2>>
RCVF1: NOP IBUS RCVDAT,0 ;INPUT CHARACTER - AND DISCARD
      MICPC=MICPC+1
      <IBUS!RCVDAT!0>

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

.SBTTL RCVG--ROUTINE TO IGNORE CRC1
RCVG: STATE RCVH ;NEXT STATE IS RCVH
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVH-INIT&777/2>>
ALWAYS RCVF1
MICPC=MICPC+1
<JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>

```

```

1023 .SBTTL RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYP
1024 :
1025 013272 RCVH: SP IBUS,RCVDAT,SPC ;GET CHAR IN SPD
1026 013272 MICPC=MICPC+1
      <MOVE!SPX!IBUS!RCVDAT!SPD>
      (1) 000542
      (1) 013272 023200
1027 013274 BRWRT IBUS,RCVCON ;READ RECVR CONTROL REGISTER
      MICPC=MICPC+1
      (1) 000543
      (1) 013274 020640 <MOVE!WRTEBR!IBUS!<RCVCON>>
1029 013276 BRD TDON1 ;IF BCC MATCH SET CRC IS GOOD
      MICPC=MICPC+1
      (1) 000544
      (1) 013276 116167 <JUMP!BRCON!<TDON1-INIT&3000*4>!<TDON1-INIT&777/2>>
1029 013300 BRWRT BR,SELA!SP1 ;READ STATUS BYTE
      MICPC=MICPC+1
      (1) 000545
      (1) 013300 060601 <MOVE!WRTEBR!BR!<SELA!SP1>>
1030 013302 BR7 RHX ;MAINT MODE
      MICPC=MICPC+1
      (1) 000546
      (1) 013302 107751 <JUMP!BR7CON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
1031 013304 BRWRT DP,<SELA!SP10> ;READ PORT STATUS WORD TO BR
      MICPC=MICPC+1
      (1) 000547
      (1) 013304 060610 <MOVE!WRTEBR!DP!<SELA!SP10>>
1032 013306 BRSHFT
      MICPC=MICPC+1
      (1) 000550
      (1) 013306 001620 <MOVE!SHFTBR!WRTEBR!SELB>
1033 013310 BR4 SNAK1 ;IF START MODE--PROCEED TO RESEND START
      MICPC=MICPC+1
      (1) 000551
      (1) 013310 117315 <JUMP!BR4CON!<SNAK1-INIT&3000*4>!<SNAK1-INIT&777/2>>
1034 013312 LDMA IMM,T ;ELSE BCC ERROR--LOAD ADDRESS OF TYPE FI
      MICPC=MICPC+1
      (1) 000552
      (1) 013312 010153 .IF IDN IMM,IMM
      <MOVE!LDMAR!IMM!<T&377>>
      .IFF
      <MOVE!LDMAR!IMM!<T>>
      .ENDC
      (1) 000
1035 013314 MEMINC IMM,2 ;WRITE NAK TYPE
      MICPC=MICPC+1
      (1) 000553
      (1) 013314 016402 <MOVE!WRMEM!INCMAR!IMM!<2>>
1036 013316 MEMINC IMM,301 ;WRITE HEADER BCC ERROR SUBTYPE
      MICPC=MICPC+1
      (1) 000554
      (1) 013316 016701 <MOVE!WRMEM!INCMAR!IMM!<301>>
1037 013320 MEM BR,SELA!SP17 ;RESTORE LAST ACKED IMAGE
      MICPC=MICPC+1
      (1) 000555
      (1) 013320 062617 <MOVE!WRMEM!BR!<SELA!SP17>>
1038 013322 LDMA IMM,NHDS ;ADDRESS CUM ERROR COUNTER

```

(1)		000556	MICPC=MICPC+1	
(1)		001	.IF IDN IMM IMM	
(1)	013322	010013	<MOVE!LDMAR!IMM!<NHDS&377>>	
(1)			.IFF	
(1)			<MOVE!LDMAR!IMM!<NHDS>>	
(1)		000	.ENDC	
1039	013324		RH5: SP MEMX,SELB,SP0	:WRITE IT TO SP0
(1)		000557	MICPC=MICPC+1	
(1)	013324	043220	<MOVE!SPX!MEMX!SELB!SP0>	
1040	013326		MEM BR,INCA!SP0	:INCREMENT IT
(1)		000560	MICPC=MICPC+1	
(1)	013326	062460	<MOVE!WRMEM!BR!<INCA!SP0>>	
1041	013330		LDMA IMM,NAKST	:ADDRESS NAKS TMED DYNAMIC
(1)		000561	MICPC=MICPC+1	
(1)		001	.IF IDN IMM IMM	
(1)	013330	010001	<MOVE!LDMAR!IMM!<NAKST&377>>	
(1)			.IFF	
(1)			<MOVE!LDMAR!IMM!<NAKST>>	
(1)		000	.ENDC	
1042	013332		BRWRITE MEMX,SELB	:WRITE IT TO BR
(1)		000562	MICPC=MICPC+1	
(1)	013332	040620	<MOVE!WRITEBR!MEMX!<SELB>>	
1043	013334		BSHFTB	:SHIFT IT RIGHT
(1)		000563	MICPC=MICPC+1	
(1)	013334	061620	<MOVE!SHFTBR!SELB!BR>	
1044	013336		MEM BR,SELB	:UPDATE IT
(1)		000564	MICPC=MICPC+1	
(1)	013336	062620	<MOVE!WRMEM!BR!<SELB>>	
1045	013340		BRO NTHRES	:BRANCH IF THRESHOLD EXCEEDED
(1)		000565	MICPC=MICPC+1	
(1)	013340	116264	<JUMP!BROCON!<NTHRES-INIT&3000*4>!<NTHRES-INIT&777/2>>	
1046	013342		ALWAYS SNAK	
(1)		000566	MICPC=MICPC+1	
(1)	013342	114712	<JUMP!ALCOND!<SNAK-INIT&3000*4>!<SNAK-INIT&777/2>>	
1047	013344		RH3: BRWRITE DP,<DECA!SP13>	:LOAD TYPE RECEIVED--DECREMENTING
(1)		000567	MICPC=MICPC+1	
(1)	013344	060573	<MOVE!WRITEBR!DP!<DECA!SP13>>	
1048	013346		Z RH1	:IF ALUOUT IS ALL ONES IS NUMBERED MSG
(1)		000570	MICPC=MICPC+1	
(1)	013346	115505	<JUMP!ZCOND!<RH1-INIT&3000*4>!<RH1-INIT&777/2>>	
1049	013350		RSTATE RCVA	
(1)		000571	MICPC=MICPC+1	
(1)	013350	000400	<MOVE!WRITEBR!IMM!<RCVA-INIT&777/2>>	
(1)		000572	MICPC=MICPC+1	
(1)	013352	063223	<MOVE!SPX!BR!SELB!SP3>	

```

1050 013354          BRWRT DP <SELA!SP10>          ;LOAD LINE STATUS WORD IN BR
      (1)           000573
      (1) 013354 060E1C          MICPC=MICPC+1
      (1)           <MOVE!WRTEBR!DP!<SELA!SP10>>

1051 013356          BR4      FLUSH1
      (1)           000574
      (1) 013356 117040          MICPC=MICPC+1
      (1)           <JUMP!BR4CON!<FLUSH1-INIT&3000*4>! FLUSH1-INIT&777/2>

1052 013360          001:    BRSHFT          ;SHIFT RIGHT
      (1)           000575
      (1) 013360 00162C          MICPC=MICPC+1
      (1)           <MOVE!SHFTBR!WRTEBR!SELB>

1053 013362          BR4      105
      (1)           000576
      (1) 013362 107204          MICPC=MICPC+1
      (1)           <JUMP!BR4CON!<105-INIT&3000*4>! \105-INIT&777/2>>

1054 013364          LDMA     IMM, TYPTAB          ;ADDRESS TYPE TABLE
      (1)           000577
      (1)           001
      (1) 013364 010163          MICPC=MICPC+1
      (1)           .IF IDN IMM, IMM
      (1)           <MOVE!LDMAR!IMM!<TYPTAB&377>>
      (1)           .IFF
      (1)           <MOVE!LDMAR!IMM!<TYPTAB>>
      (1)           .ENDC
      (1)           000

1055 013366          CMP      <MEMX!INCMAR>, SP13
      (1)           000600
      (1) 013366 054373          MICPC=MICPC+1
      (1)           <SUBTC!MEMX!INCMAR!SP13>

1056 013370          Z      REP
      (1)           000601
      (1) 013370 115404          MICPC=MICPC+1
      (1)           <JUMP!ZCOND!<REP-INIT&3000*4>!<REP-INIT&777/2>>

1057 013372          CMP      <MEMX!INCMAR>, SP13
      (1)           000602
      (1) 013372 054373          MICPC=MICPC+1
      (1)           <SUBTC!MEMX!INCMAR!SP13>

1058 013374          Z      NAK
      (1)           000603
      (1) 013374 115443          MICPC=MICPC+1
      (1)           <JUMP!ZCOND!<NAK-INIT&3000*4>!<NAK-INIT&777/2>>

1059 013376          105:    LDMA     IMM, TYPSTT          ;SET POINTER TO START TYPE
      (1)           000604
      (1)           001
      (1) 013376 0101E5          MICPC=MICPC+1
      (1)           .IF IDN IMM, IMM
      (1)           <MOVE!LDMAR!IMM!<TYPSTT&377>>
      (1)           .IFF
      (1)           <MOVE!LDMAR!IMM!<TYPSTT>>
      (1)           .ENDC
      (1)           000

1060 013400          CMP      <MEMX!INCMAR>, SP13
      (1)           000605
      (1) 013400 054373          MICPC=MICPC+1
      (1)           <SUBTC!MEMX!INCMAR!SP13>

1061 013402          Z      START
      (1)           000606
      (1) 013402 115413          MICPC=MICPC+1
      (1)           <JUMP!ZCOND!<START-INIT&3000*4>!<START-INIT&777/2>>
  
```

```

1062      ;STACK TYPE
1063 013404 CMP      <MEMX!INCMAR>,SP13
      (1)      MICPC=MICPC+1
      (1) 013404 000607 <SUBTC!MEMX!INCMAR!SP13>
      (1)      054373
1064      Z      STACK
      (1)      MICPC=MICPC+1
      (1) 013406 000610 <JUMP!ZCOND!<STACK-INIT&3000*4>!<STACK-INIT&777/2>>
      (1)      115425
1065      CMP      <MEMX!INCMAR>,SP13      ;ACK TYPE
      (1)      MICPC=MICPC+1
      (1) 013410 000611 <SUBTC!MEMX!INCMAR!SP13>
      (1)      054373
1066      Z      ACK
      (1)      MICPC=MICPC+1
      (1) 013412 000612 <JUMP!ZCOND!<ACK-INIT&3000*4>!<ACK-INIT&777/2>>
      (1)      105620
1067      ALWAYS IDLE      ;OTHERWISE IGNORE--MUST BE OBS MSG
      (1)      MICPC=MICPC+1
      (1) 013414 000613 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      (1)      100445
1068      001
1069      RCVCK: .IF DF LOW
      (1)      SPBR      IBUS,RCVCON,SP0      ;READ RCVR CONTROL CSR
      (1) 013416 000614 MICPC=MICPC+1
      (1)      023640 <MOVE!SPBRX!IBUS!RCVCON!SP0>
1070      BRWRT BR,ADD!SP0      ;SHIFT LEFT
      (1)      MICPC=MICPC+1
      (1) 013420 000615 <MOVE!WRTEBR!BR!<ADD!SP0>>
      (1)      060400
1071      BR7      I1
      (1)      MICPC=MICPC+1
      (1) 013422 000616 <JUMP!BR7CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
      (1)      103452
1072      ALWAYS TA1
      (1)      MICPC=MICPC+1
      (1) 013424 000617 <JUMP!ALCOND!<TA1-INIT&3000*4>!<TA1-INIT&777/2>>
      (1)      110405
1073      ACK: BRWRT BR,AA!SP10      ;READ LINE STATUS-SHIFTING LEFT
      (1)      MICPC=MICPC+1
      (1) 013426 000620 <MOVE!WRTEBR!BR!<AA!SP10>>
      (1)      060530
1074      BR4      5$      ;IF START RECD -- CLEAR START MODE
      (1)      MICPC=MICPC+1
      (1) 013430 000621 <JUMP!BR4CON!<5$-INIT&3000*4>!<5$-INIT&777/2>>
      (1)      107223
1075      ALWAYS IDLE
      (1)      MICPC=MICPC+1
      (1) 013432 000622 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      (1)      100445
1076      5$: BRWRT IMM,327      ;CLEAR START MODE
      (1)      MICPC=MICPC+1
      (1) 013434 000623 <MOVE!WRTEBR!IMM!<327>>
      (1)      000727
1077      SP      BR,AA&DB,SP10      ;IN LINE STATUS
      (1)      MICPC=MICPC+1
      (1) 013436 000624

```

J09

MAC: DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 06-DEC-76 10:14

MACY11 27(1006) 07-MAR-77 16:24 PAGE 6-47
RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYPES

PAGE: 0113

(1) 013436 063270
1078 013440
(1) 000625
(1) 013440 104517
1079 000

<MOVE!SPX!BR!AANDB!SPIC>
ALWAYS RDS
MICPC=MICPC+1
<JUMP!ALCOND!<RDS-INIT\$3000*4>!<RDS-INIT\$777/2>>
.ENDC

:*****TIME CRITICAL CODE-- CHANGE WITH GREAT CARE*****

1081
1082
1083 013442
(1) 000626
(1) 013442 123600
(1)
1084 013444
(1) 000627
(1) 013444 102045
(1)
1085 013446
(1) 000630
(1) 013446 000600
(1)
1086 013450
(1) 000631
(1) 013450 061310
(1)
1087 013452
(1) 000632
(1) 013452 000670
1088 013454
(1) 000633
(1) 013454 104637
(1)
1089
1090 001
1091
1092
1093
1094
1095
1096 000
1097 001
1098 013456
(1) 000634
(1) 013456 123600
(1)
1099 013460
(1) 000635
(1) 013460 102045
(1)
1100 013462
(1) 000636
(1) 013462 000645
1101 013464
(1) 000637
(1) 013464 063223
(1)
1102 013466
(1) 000640
(1) 013466 022203
(1)
1103 013470
1104 000
1105 013470

RCVK01: .SBTTL RCVK01--ROUTINE TO HANDLE FIRST BYTE CCD RECEIVE
SPBR IBUS,NPR,SPD ;READ NPR REGISTER
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!NPR!SPD>

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

BRWRT IMM,200 ;MASK FOR CO(BYTE TRANSFER)
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<200>>

OUT BR,<AORB!ONPR> ;TURN ON CO
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AORB!ONPR>>

STATE RKE1
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RKE1-INIT&777/2>>
ALWAYS RCVK02
MICPC=MICPC+1
<JUMP!ALCOND!<RCVK02-INIT&3000*4>!<RCVK02-INIT&777/2>>

.SBTTL RCVK0--PROCESS ODD CHARACTER
.IF NDF LOW
RCVK0: STATE RCVKE
RCVK02: SP BR,SELB,SP3 ;SET STATE
OUTPUT IBUS,RCVDAT!OUTDA2 ;OUT CHAR TO BUFFER
RK2: SPBR IBUS,NPR,SPD ;NPR BUSY
BRO RK1 ;IF SO GET READY TO GO
.ENDC
.IF DF LOW
RCVK0: SPBR IBUS,NPR,SPD ;IS AN NPR GOING
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!NPR!SPD>

BRO IDLE ;IF SO --- GET OUT
MICPC=MICPC+1
<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

STATE RCVKE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVKE-INIT&777/2>>
RCVK02: SP BR,SELB,SP3 ;SET STATE
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>

OUTPUT IBUS,RCVDAT!OUTDA2 ;OUTPUT A CHAR
MICPC=MICPC+1
<MOVE!WROUT!IBUS!<RCVDAT!OUTDA2>>

RK2: .ENDC
RK2: BRWRT IMM,21 ;SET OUT NPR (01) AND NPR REG


```

(1) 000641
(1) 013470 000421 MICPC=MICPC+1
(1) <MOVE!WRTEBR!IMM!<21>>
1106 013472 SP IBUS,NPR,SP0 ;READ NPR REGISTER
(1) 000642 MICPC=MICPC+1
(1) 013472 123200 <MOVE!SPX!IBUS!NPR!SP0>
(1)
1107 013474 RK7: OUT BR,<AORB!ONPR> ;WRITE NPR REGISTER
(1) 000643 MICPC=MICPC+1
(1) 013474 06131C <MOVE!WR0UTX!BR!<AORB!ONPR>>
(1)
1108 013476 ALWAYS IDLE
(1) 000644 MICPC=MICPC+1
(1) 013476 100445 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
1109 001 .IF NDF LOW
1110 LDMA IMM,OSTATE ;LOAD POINTER TO OLD STATE STACK
1111 MEM DP,<SELA!SP3> ;STORE CURRENT NEXT STATE POINTER
1112 STATE RHBLOC ;SET STATE FOR NPR BLOCKED
1113 ALWAYS REXIT
1114 RHBLOC: LDMA IMM,OSTATE ;ADDRESS OLD STATE POITER
1115 SP MEMX,SELB,SP3 ;RESTORE THE STATE
1116 ALWAYS RK2 ;TRY THE NPR NOW
1117 .ENDC
000

```

```

1119
1120 013500 000645
(1) 013500 120600
(1)
1121 001
1122 BR4 RK4 ;IF RECV NPR--BRANCH
1123 000
1124 001
1125 013502 000646
(1) 013502 102045
(1)
1126 000
1127 013504 000647
(1) 013504 023140
(1)
1128 013506 000650
(1) 013506 062066
(1)
1129 013510 000651
(1) 013510 063164
(1)
1130 013512 000652
(1) 013512 105255
(1)
1131 013514 000653
(1) 013514 063165
(1)
1132 013516 000654
(1) 013516 105721
(1)
1133 013520 000655
(1) 013520 023140
(1)
1134 013522 000656
(1) 013522 062066
(1)
1135 013524 000657
(1) 013524 115030
(1)
1136 013526 000660
(1) 013526 023200
(1)
1137 013530 000661
(1)

```

.SBTTL RCVKE--HANDLE EVEN BYTES
RCVKE: BRWRT IBUS,NPR ;READ NPR CONTROL REGISTER
MICPC=MICPC+1
<MOVE!WRTEBR!IBUS!<NPR>>

.IF NDF LOW
BR4 RK4 ;IF RECV NPR--BRANCH
.ENDC
.IF DF LOW
BR0 IDLE
MICPC=MICPC+1
<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

.ENDC
RK5: SP IBUS,IOBA1,SPO ;READ LOW BYTE OF BA TO SP
MICPC=MICPC+1
<MOVE!SPX!IBUS!IOBA1!SPO>

OUTPUT DP,<INCA!OBA1> ;WRITE INCREMENTED BA
MICPC=MICPC+1
<MOVE!WROUT!DP!<INCA!OBA1>>

RK50: SP BR,DECA,SP4 ;DECREMENT CHARACTER COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP4>

C 10\$;NO OVERFLOW
MICPC=MICPC+1
<JUMP!CCOND!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>

SP BR,DECA,SP5 ;OVERFLOW - DECREMENT HIGH BYTE
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP5>

Z RL3 ;BYTE COUNT ZERO
MICPC=MICPC+1
<JUMP!ZCOND!<RL3-INIT&3000*4>!<RL3-INIT&777/2>>

10\$: SP IBUS,IOBA1,SPO ;READ INCREMENTED BA
MICPC=MICPC+1
<MOVE!SPX!IBUS!IOBA1!SPO>

OUTPUT DP,<INCA!OBA1> ;WRITE INCREMENTED BA
MICPC=MICPC+1
<MOVE!WROUT!DP!<INCA!OBA1>>

C ICBA22 ;IF CARRY INC BA HIGH
MICPC=MICPC+1
<JUMP!CCOND!<ICBA22-INIT&3000*4>!<ICBA22-INIT&777/2>>

RK9: SP IBUS,RCVDAT,SPO ;READ CHAR AND SAVE IN SPO
MICPC=MICPC+1
<MOVE!SPX!IBUS!RCVDAT!SPO>

RK3: OUTPUT BR,SELA!OUTDA1 ;WRITE THE CHARACTER
MICPC=MICPC+1

```

(1) 013530 062202          <MOVE!WROUT!BR!<SELA!OUTDA1>>
(1)
1138 013532                SP      BR,DECA,SP4          :DECREMENT THE COUNTOF BYTES
(1) 013532 000562          MICPC=MICPC+1
(1) 013532 063164          <MOVE!SPX!BR!DECA!SP4>
(1)
1139 013534                C      RK6              :NO OVERFLOW
(1) 013534 000663          MICPC=MICPC+1
(1) 013534 105266          <JUMP!CCOND!<RK6-INIT&3000*4>!<RK6-INIT&777/2>>
(1)
1140 013536                SP      BR,DECA,SP5          :DECREMENT HIGH BYTE OF COUNT
(1) 013536 000564          MICPC=MICPC+1
(1) 013536 063165          <MOVE!SPX!BR!DECA!SP5>
(1)
1141 013540                Z      RL4              ;BYTE COUNT ZERO
(1) 013540 000665          MICPC=MICPC+1
(1) 013540 111765          <JUMP!ZCOND!<RL4-INIT&3000*4>!<RL4-INIT&777/2>>
(1)
1142          001
1143          001          RK6:  .IF NDF LOW
1144          001          BRWRT  IBUS,RCVCON          :READ RECEIVER CONTROL REGISTER
1145          001          BR4     RCVK0          ;IF ANOTHER CHARACTER--PROCESS
1146          001          STATE  RCVK0
1147          001          ALWAYS REXIT
1148          001          BR0     IDLE
1149          000          ALWAYS  RK5              ;IF NO NPR --PROCESS
1150          001          .ENDC
1151 013542                .IF DF  LOW
(1) 013542 000666          STATE  RCVK0
(1) 013542 000634          MICPC=MICPC+1
1152 013544                <MOVE!WRTEBR!IMM!<RCVK0-INIT&777/2>>
(1) 013544 000667          ALWAYS  REXIT
(1) 013544 104422          MICPC=MICPC+1
(1) 013544 104422          <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
1153          000          .ENDC
1154          000
1155 013546                .
(1) 013546 000670          SP      IBUS,NPR,SPO          :READ NPR REGISTER
(1) 013546 123200          MICPC=MICPC+1
(1) 013546 123200          <MOVE!SPX!IBUS!NPR!SPO>
1156 013550                BRWRT  IMM,177          ;MASK FOR ALL BUT CC
(1) 013550 000671          MICPC=MICPC+1
(1) 013550 000577          <MOVE!WRTEBR!IMM!<177>>
1157 013552                OUT     BR,<AANDB!ONPR>          ;TURN OFF ALL BUT CC
(1) 013552 000672          MICPC=MICPC+1
(1) 013552 061270          <MOVE!WROUTX!BR!<AANDB!ONPR>>
1158 013554                ALWAYS  RK50
(1) 013554 000673          MICPC=MICPC+1
(1) 013554 104651          <JUMP!ALCOND!<RK50-INIT&3000*4>!<RK50-INIT&777/2>>
1159          000          ;*****END OF TIME CRITICAL PATH*****
1160          000
1161 013556                RCVKED: SP      IBUS,RCVDAT,SPO          :READ CHARACTER AND SAVE IN SPO
(1) 013556 000674          MICPC=MICPC+1

```

```

(1) 013556 02320C      <MOVE!SPX!IBUS!RCVDAT!SP0>
(1)
1162 013560           BRWRTE BR SELA!SP1          ;READ STATUS BYTE
(1) 013560 000675     MICPC=MICPC+1
(1) 013560 060601     <MOVE!WRTEBR!BR!<SELA!SP1>>
(1)
1163 013562           BR7 PASWRD              ;MAINT MODE - SEE IF RLD MESSAGE
(1) 013562 000676     MICPC=MICPC+1
(1) 013562 11760C     <JUMP!BR7CON!<PASWRD-INIT&3000*4>!<PASWRD-INIT&777/2>>
(1)
1164 013564           ALWAYS RK3                ;OTHERWISE PROCESS NORMALLY
(1) 013564 000677     MICPC=MICPC+1
(1) 013564 104661     <JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>
(1)

```

1166
 1167 013566 000700
 (1) 013566 023213
 (1)
 1168 013570 000701
 (1) 013570 000703
 1169 013572 000702
 (1) 013572 104422
 (1)
 1:70

```

.SBTTL RCVI--STORE UNNUMBERED MESSAGE TYPE
RCVI: SP IBUS,RCVDAT,SP13 :STORE UNNUMBERED TYPE
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!RCVDAT!SP13>

STATE RCVJ ;NEXT STATE IS J
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVJ-INIT&777/2>>
      ALWAYS REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
:

```

1172
 1173 013574
 (1) 000703
 (1) 013574 114470
 (1)
 1174 013576
 (1) 000704
 (1) 013576 000706
 1175 013600
 (1) 000705
 (1) 013600 104422
 (1)

RCVJ: .SBTTL RCVJ--ROUTINE TO HANDLE SUBTYPE FIELD,SELECT AND FINAL
 ALWAYS SELQSY ;"CALL" SELECT AND QSYNC SUBROUTINE
 MICPC=MICPC+1
 <JUMP!ALCOND!<SELQSY-INIT&3000*4>!<SELQSY-INIT&777/2>>
 STATE RCVR ;NEXT STATE IS N
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<RCVR-INIT&777/2>>
 ALWAYS REXIT
 MICPC=MICPC+1
 <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>

E10

```

1177 .SBTTL RCVR--UNNUMBERED MESSAGE RESPONSE FIELD
1178 :ENTERED FROM IDLE LOOP
1179 :
1180 RCVR: BRWRTE IMM,3 ;REP MESSAGE TYPE TO BR
(1) MICPC=MICPC+1
(1) 013602 000706 <MOVE!WRTEBR!IMM!<3>>
(1) 013602 000403
1181 NOP BR,SUB,SP13 ;IS TYPE ACK OR NAK
(1) MICPC=MICPC+1
(1) 013604 000707 <BR!SUB!SP13>
(1) 013604 060353
1182 STATE RCVG ;NEXT STATE IS RCVG
(1) MICPC=MICPC+1
(1) 013606 000710 <MOVE!WRTEBR!IMM!<RCVG-INIT&777/2>>
(1) 013606 000713 :***NOTE THIS INSTR DOES NOT CLOCK "C"
1183 :IF NOT IGNORE
1184 C RCVF1
(1) MICPC=MICPC+1
(1) 013610 000711 <JUMP!CCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>
(1) 013610 105136
1185 ALWAYS RD2 ;DO RANGE CHECKS
(1) MICPC=MICPC+1
(1) 013612 000712 <JUMP!ALCOND!<RD2-INIT&3000*4>!<RD2-INIT&777/2>>
(1) 013612 104503
(1)
(1)

```

F10

MAC: DDMP PROTOCOL IMPLEMENTATION
DDNEW.MAC 06-DEC-76 10:14

MACY11 27(1006) 07-MAR-77 16:24 PAGE 6-56
RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD

PAGE: 0122

1187
1188
1189
1190 013614 000713
1191 013614 000535
1192 013616 000714
1193 013616 104536

.SBTTL RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD
:ENTER FROM IDLE

RCVQ: STATE RCVF ;NEXT STATE IS ADDRESS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVF-INIT\$777/2>>
ALWAYS RCVF1
MICPC=MICPC+1
<JUMP!ALCOND!<RCVF1-INIT\$3000*4>!<RCVF1-INIT\$777/2>>

1193			.SBTTL RCVL--PROCESS CRC3	
1194			:ENTERED FROM IDLE _OOP	
1195	013620		RCVL: SPBR IBUS,NPR,SPC	:READ NPR CONTROL
(1)		000715	MICPC=MICPC+1	
(1)	013620	123600	<MOVE!SPBRX!IBUS!NPR!SPC>	
(1)				
1196		001	.IF NDF LOW	
1197			BR4 RL1	:RCV NPR BRANCH
1198		000	.ENDC	
1199		001	.IF DF _OW	
1200	013622		BRO IDLE	
(1)		000716	MICPC=MICPC+1	
(1)	013622	102045	<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
(1)				
1201		000	.ENDC	
1202	013624		RL1: BRWRT IMM,177	:MASK TO TURN OFF CC
(1)		000717	MICPC=MICPC+1	
(1)	013624	000577	<MOVE!WRTBR!IMM!<177>>	
(1)				
1203	013626		OUT BR,ANDB!ONPR	
(1)		000720	MICPC=MICPC+1	
(1)	013626	061270	<MOVE!WROUTX!BR!<ANDB!ONPR>>	
(1)				
1204	013630		RL3: SPBR IBUS,IOBA1,SPD	:READ LOW ORDER BYTE OF ADDRESS
(1)		000721	MICPC=MICPC+1	
(1)	013630	023540	<MOVE!SPBRX!IBUS!IOBA1!SPD>	
(1)				
1205	013632		OUTPUT BR,INCA!OBA1	:INCREMENT THE LOW ORDER BYTE
(1)		000722	MICPC=MICPC+1	
(1)	013632	062066	<MOVE!WROUT!BR!<INCA!OBA1>>	
(1)				
1206	013634		SP IBUS,IOBA2,SPD	:READ HIGH BYTE
(1)		000723	MICPC=MICPC+1	
(1)	013634	023160	<MOVE!SPX!IBUS!IOBA2!SPD>	
(1)				
1207	013636		OUTPUT BR,AC!OBA2	:ADD CARRY TO HIGH BYTE
(1)		000724	MICPC=MICPC+1	
(1)	013636	062107	<MOVE!WROUT!BR!<AC!OBA2>>	
(1)				
1208			.STATE RCVM	
1209	013640		MICPC=MICPC+1	
(1)		000725	<MOVE!WRTBR!IMM!<RCVM-INIT&777/2>>	
(1)	013640	000727		
1210	013642		ALWAYS RCVF1	
(1)		000726	MICPC=MICPC+1	
(1)	013642	104536	<JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>	
(1)				
1211		001	.IF NDF LOW	
1212			RL1: BRO IDLE	:NPR GOING --GET OUT
1213			ALWAYS RL2	
1214		000	.ENDC	
1215				
1216				

H10

```

1220      .SBTTL RCVM--PROCESS CRC4--END OF DATA MESSAGE
1221      :ENTERED FROM IDLE LOOP
1222      :IF CRC CORRECT -- QUEUE INTERRUPT AND UPDATE RESPONSE
1223      :
1223 013644 RCVM:      :IF CRC WRONG SEND NAK
      BRWRT  IBUS,UBBR          :READ UNIBUS BR REGISTER
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<UBBR>>
1224      BR0      NXMERR          :NON-EXISTANT MEMORY
      MICPC=MICPC+1
      <JUMP!BROCON!<NXMERR-INIT&3000*4>!<NXMERR-INIT&777/2>>
1225 013646      SP      IBUS,RCVDAT,SPD          :READ CRC CHARACTER
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!RCVDAT!SPC
1226 013652      BRWRT  IBUS,RCVCON          :READ RECEIVER CONTROL REGISTER
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<RCVCON>>
1227 013654      BR0      RCVM1          :IF CRC GOOD -- PROCESS
      MICPC=MICPC+1
      <JUMP!BROCON!<RCVM1-INIT&3000*4>!<RCVM1-INIT&777/2>>
1228 013656      BRWRT  BR,SELA!SP1          :READ STATUS BYTE
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
1229 013660      BR7      RHX          :CRC ERROR IN BOOT MODE - FLUSH
      MICPC=MICPC+1
      <JUMP!BR7CON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
1230 013662      LDMA    IMM,T          :ELSE SEND NAK --DATA ERROR
      MICPC=MICPC+1
      .IF IDN IMM,IMM
      <MOVE!LDMAR!IMM!<T&377>>
      .IFF
      <MOVE!LDMAR!IMM!<T>>
      .ENDC
1231 013664      MEMINC  IMM,2          :NAK TYPE
      MICPC=MICPC+1
      <MOVE!WRMEM!INCMAR!IMM!<2>>
1232 013666      MEMINC  IMM,302        :DATA ERROR SUBTYPE
      MICPC=MICPC+1
      <MOVE!WRMEM!INCMAR!IMM!<302>>
1233 013670      LDMA    IMM,NDATS
      MICPC=MICPC+1
      .IF IDN IMM,IMM
      <MOVE!LDMAR!IMM!<NDATS&377>>
      .IFF
      <MOVE!LDMAR!IMM!<NDATS>>
      .ENDC
  
```

```

(1)
1234 013672          ALWAYS RNS                :SEND NAK
(1)          000742  MICPC=MICPC+1
(1) 013672 104557  <JUMP!ALCOND!<RHS-INIT&3000*4>!<RHS-INIT&777/2>>
(1)
1235
1236 013674          RCVMQ: LDMA IMM,<<RTHRS+3>>          :POINT TO ERROR WORD
(1)          000743  MICPC=MICPC+1
(1)          001      .IF IDN IMM,IMM
(1) 013674 010177  <MOVE!LDMAR!IMM!<<RTHRS+3>&377>>
(1)          .IFF
(1)          <MOVE!LDMAR!IMM!<<RTHRS+3>>>
(1)          .ENDC
(1)          000
(1)
1237 013676          BRWRTE IMM,10                :MAINT MESSAGE ERROR
(1)          000744  MICPC=MICPC+1
(1) 013676 000410  <MOVE!WRTEBR!IMM!<10>>
(1)
1238 013700          ALWAYS RCEXY                ;GIVE FATAL ERROR
(1)          000745  MICPC=MICPC+1
(1) 013700 114525  <JUMP!ALCOND!<RCEXY-INIT&3000*4>!<RCEXY-INIT&777/2>>
(1)

```

```

1240 .SBTTL EM2--PROCESS RLD MESSAGE
1241 ;ENTERED FROM IDLE LOOP
1242 ;IF RLD PASSWORD CHECKS TRIGGER THE BOOT ROM
1243
1244 EM2: BRWRT IBUS,RCVDAT ;READ THE CHAR
(1) MICPC=MICPC+1
(1) 013702 000746 <MOVE!WRTEBR!IBUS!<RCVDAT>>
(1) 013702 020600
1245 CMP BR,SP16 ;IS IT A MATCH
(1) MICPC=MICPC+1
(1) 013704 000747 <SUBTC!BR!SP16>
(1) 013704 060376
1246 Z EM3
(1) MICPC=MICPC+1
(1) 013706 000750 <JUMP!ZCOND!<EM3-INIT&3000*4>!<EM3-INIT&777/2>>
(1) 013706 105757
1247 ;FALL INTO RHX
1248 RHX: BRWRT BR,AA!SP1 ;READ STATUS BYTE SHIFTED LEFT
(1) MICPC=MICPC+1
(1) 013710 000751 <MOVE!WRTEBR!BR!<AA!SP1>>
(1) 013710 060521
1249 BR4 105 ;DLE RECEIVED IN NORMAL MODE
(1) MICPC=MICPC+1
(1) 013712 000752 <JUMP!BR4CON!<105-INIT&3000*4>!<105-INIT&777/2>>
(1) 013712 107354
1250 ALWAYS FLUSH ;ALREADY IN MAINT MODE
(1) MICPC=MICPC+1
(1) 013714 000753 <JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
(1) 013714 104415
1251 105: BRWRT IMM,163 ;MASK TO CLEAR ALL MAINT RELATED BITS
(1) MICPC=MICPC+1
(1) 013716 000754 <MOVE!WRTEBR!IMM!<163>>
(1) 013716 000563
1252 SP BR,AANDB,SP1 ;CLEAR THEM
(1) MICPC=MICPC+1
(1) 013720 000755 <MOVE!SPX!BR!AANDB!SP1>
(1) 013720 063261
1253 ALWAYS FLUSH
(1) MICPC=MICPC+1
(1) 013722 000756 <JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
(1) 013722 104415
1254
1255 EM3: SP BR,DECA,SP4 ;DECREMENT CHARACTER COUNT BY ONE
(1) MICPC=MICPC+1
(1) 013724 000757 <MOVE!SPX!BR!DECA!SP4>
(1) 013724 063164
1256 Z EMTRIG ;TRIGGER AC LOW
(1) MICPC=MICPC+1
(1) 013726 000760 <JUMP!ZCOND!<EMTRIG-INIT&3000*4>!<EMTRIG-INIT&777/2>>
(1) 013726 115720
1257 ALWAYS IDLE
(1) MICPC=MICPC+1
(1) 013730 000761 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1) 013730 100445
(1)

```

```

1259
1260 013732 000762      BOOT:  BRWRT  BR, SELA!SP1      ;SEE IF IN MAINT. MODE
      (1) 013732 060501      MICPC=MICPC+1
      (1) 013732 060501      <MOVE!WRTEBR!BR!<SELA!SP1>>
1261 013734 000763      BR7   RA3      ;BRANCH IF SC AND TREAT DLE LIKE NUM. MSG.
      (1) 013734 103747      MICPC=MICPC+1
      (1) 013734 103747      <JUMP!BR7CON!<RA3-INIT&3000*4>!<RA3-INIT&777/2>>
1262 013736 000764      BRWRT  IMM,210  ;MASK TO SET MAINT MODE AND DLE RECV'D
      (1) 013736 000610      MICPC=MICPC+1
      (1) 013736 000610      <MOVE!WRTEBR!IMM!<210>>
1263 013740 000765      SP    BR, AORB,SP1 ;SET THE BITS
      (1) 013740 063301      MICPC=MICPC+1
      (1) 013740 063301      <MOVE!SPX!BR!AORB!SP1>
1264 013742 000766      ALWAYS RA3      ;TREAT LIKE NUMBERED MESSAGE
      (1) 013742 100747      MICPC=MICPC+1
      (1) 013742 100747      <JUMP!ALCOND!<RA3-INIT&3000*4>!<RA3-INIT&777/2>>
1265 013744 000767      RESEXT: BRWRT  IMM,4      ;ADD TO MXT BITS
      (1) 013744 000404      MICPC=MICPC+1
      (1) 013744 000404      <MOVE!WRTEBR!IMM!<4>>
1266 013746 000770      SP    IBUS, NPR, SPO
      (1) 013746 123200      MICPC=MICPC+1
      (1) 013746 123200      <MOVE!SPX!IBUS!NPR!SPO>
1267 013750 000771      OUT   BR, ADD!ONPR ;DO IT
      (1) 013750 061010      MICPC=MICPC+1
      (1) 013750 061010      <MOVE!WROUTX!BR!<ADD!ONPR>>
1268 013752 000772      ALWAYS RES1
      (1) 013752 110753      MICPC=MICPC+1
      (1) 013752 110753      <JUMP!ALCOND!<RES1-INIT&3000*4>!<RES1-INIT&777/2>>
1269 013754 000773      TABMXT: BRWRT  IMM,4      ;INCREMENT MXT
      (1) 013754 000404      MICPC=MICPC+1
      (1) 013754 000404      <MOVE!WRTEBR!IMM!<4>>
1270 013756 000774      SP    IBUS, UBBR, SPO ;READ BR CONTROL
      (1) 013756 123220      MICPC=MICPC+1
      (1) 013756 123220      <MOVE!SPX!IBUS!UBBR!SPO>
1271 013760 000775      OUT   BR, ADD!OBR
      (1) 013760 061011      MICPC=MICPC+1
      (1) 013760 061011      <MOVE!WROUTX!BR!<ADD!OBR>>
1272 013762 000776      ALWAYS ECX
      (1) 013762 114767      MICPC=MICPC+1
      (1) 013762 114767      <JUMP!ALCOND!<ECX-INIT&3000*4>!<ECX-INIT&777/2>>
1273 013764 000777      $ZERO
      (1) 013764 000000      MICPC=MICPC+1
      (1) 013764 000000      000000

```

r

L10

EM2--PROCESS RLD MESSAGE
DE-DEC-76 10:14
PROY000: IMPLEMENTATION

MAR-77 16:24 PAGE 6-62
EM2--PROCESS RLD MESSAGE

PAGE: 0129

M10

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 06-DEC-76 13:14

MACY11 27(1006) 07-MAR-77 16:24 PAGE 6-63
EM2--PROCESS RLD MESSAGE

PAGE: 0129

1275		013766	.=INIT+2000	
1276		000777	MICPC=777	
1277			.SBTTL TMTDA--TRANSMITTER DISPATCH ROUTINE	
1278			.	
1279	013766		TMTDA: BRWRITE IBUS,TMTCON	;READ TRANSMITTER CONTROL REGISTER
(1)		001000	MICPC=MICPC+1	
(1)	013766	020620	<MOVE!WRITEBR!IBUS!<TMTCON>>	
(1)				
1280	013770		.BR4 DP SELA,<2!PAGE2>	;IF READY PROCEED
(1)		001001	MICPC=MICPC+1	
(1)	013770	173202	<JUMP! BR4CON!DP!SELA!2!PAGE2>	
(1)				
1281	013772		ALWAYS I1	;ELSE IDLE
(1)		001002	MICPC=MICPC+1	
(1)	013772	100452	<JUMP!ALCOND!<I1-INIT\$3000*4>!<I1-INIT\$777/2>>	
(1)				

```

1283          .SBTTL TMTA--FIRST CHARACTER OF HEADER
1284
1285 013774    BRWRT  BR,AA!SP10          :SHIFT LEFT
(1)          MICPC=MICPC+1
(1) 013774    <MOVE!WRTEBR!BR!<AA!SP10>>
(1)
1286 013776    BR7    RCVCK
(1)          MICPC=MICPC+1
(1) 013776    <JUMP!BR7CON!<RCVCK-INIT&3000*4>!<RCVCK-INIT&777/2>>
(1)
1287 014000    TAI:   BRWRT  BR,SELA!SP10      :REREAD STATUS
(1)          MICPC=MICPC+1
(1) 014000    <MOVE!WRTEBR!BR!<SELA!SP10>>
(1)
1288 014002    BRO    NUMSYN                  ;IF UNNUMBPENDING -- SEND IT
(1)          MICPC=MICPC+1
(1) 014002    <JUMP!BROCON!<NUMSYN-INIT&3000*4>!<NUMSYN-INIT&777/2>>
(1)
1289 014004    BRSHFT
(1)          MICPC=MICPC+1
(1) 014004    <MOVE!SHFTBR!WRTEBR!SELB>
(1)
1290 014006    BR4    I1                      ;IF START MODE--EXIT
(1)          MICPC=MICPC+1
(1) 014006    <JUMP!BR4CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
(1)
1291 014010    BR1    NUMSYN                  ;IF LINE HAS GONE IDLE SEND SYN
(1)          MICPC=MICPC+1
(1) 014010    <JUMP!BR1CON!<NUMSYN-INIT&3000*4>!<NUMSYN-INIT&777/2>>
(1)
1292          :ELSE--START TO SEND MESSAGE
1293 014012    TMTXT: TSTATE TMTB
(1)          MICPC=MICPC+1
(1) 014012    <MOVE!WRTEBR!IMM!<TMTB-INIT&777/2>>
(1)          MICPC=MICPC+1
(1) 014014    <MOVE!SPX!BR!SELB!SP2>
1294 014016    BRWRT  BR,SELA!SP1            ;ARE WE IN BOOT MODE
(1)          MICPC=MICPC+1
(1) 014016    <MOVE!WRTEBR!BR!<SELA!SP1>>
(1)
1295 014020    BR7    TMTBT                  ;IF SO SEND DLE
(1)          MICPC=MICPC+1
(1) 014020    <JUMP!BR7CON!<TMTBT-INIT&3000*4>!<TMTBT-INIT&777/2>>
(1)
1296 014022    BRWRT  BR,<SELA!SP10>         ;UNNUMB MESSGE?
(1)          MICPC=MICPC+1
(1) 014022    <MOVE!WRTEBR!BR!<SELA!SP10>>
(1)
1297 014024    BRO    TMTUN                  ;IF SO --BRANCH
(1)          MICPC=MICPC+1
(1) 014024    <JUMP!BROCON!<TMTUN-INIT&3000*4>!<TMTUN-INIT&777/2>>
(1)
1298 014026    BRWRT  IMM,201                 ;ELSE STORE SOH
(1)          MICPC=MICPC+1
(1) 014026    <MOVE!WRTEBR!IMM!<201>>
(1)

```


1299 014030
 (1) 001021
 (1) 014030 062220
 (1)
 1300 014032
 (1) 001022
 (1) 014032 100452
 (1)
 1301 014034
 (1) 001023
 (1) 014034 000600
 (1) 001024
 (1) 014036 063222
 1302 014040
 (1) 001025
 (1) 014040 000405
 (1)
 1303 014042
 (1) 001026
 (1) 014042 115421
 (1)
 1304 014044
 (1) 001027
 (1) 014044 000620
 (1)
 1305 014046
 (1) 001030
 (1) 014046 110421
 (1)
 1306
 1307 014050
 (1) 001031
 (1) 014050 060610
 (1)
 1308 014052
 (1) 001032
 (1) 014052 113434
 (1)
 1309 014054
 (1) 001033
 (1) 014054 100452
 (1)
 1310 014056
 (1) 001034
 (1) 014056 020660
 (1)
 1311 014060
 (1) 001035
 (1) 014060 001620
 (1)
 1312 014062
 (1) 001036
 (1) 014062 103052
 (1)
 1313 014064
 (1) 001037

TMTAS: OUTPUT BR, <SELB!TMTDAT> ; IN TMT SILC
 MICPC=MICPC+1
 <MOVE!WROUT!BR!<SELB!TMTDAT>>
 ALWAYS I1
 MICPC=MICPC+1
 <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
 TMTUN: TSTATE TMTI
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<TMTI-INIT&777/2>>
 MICPC=MICPC+1
 <MOVE!SPX!BR!SELB!SP2>
 BRWRT IMM, 5 ; ENG TO BR
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<5>>
 ALWAYS TMTAS
 MICPC=MICPC+1
 <JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>
 TMTBT: BRWRT IMM, 220 ; WRITE A DLE TO BR
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<220>>
 ALWAYS TMTAS ; SEND IT
 MICPC=MICPC+1
 <JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>
 NUMSYN: BRWRT BR, <SELA!SP10> ; READ LINE STATUS WORD
 MICPC=MICPC+1
 <MOVE!WRTEBR!BR!<SELA!SP10>>
 BR7 5\$; IF OK TO SEND--PROCEED
 MICPC=MICPC+1
 <JUMP!BR7CON!<5\$-INIT&3000*4>!<5\$-INIT&777/2>>
 ALWAYS I1 ; ELSE--IDLE
 MICPC=MICPC+1
 <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
 5\$: BRWRT IBUS, MODEM ; ARE WE STILL SENDING?
 MICPC=MICPC+1
 <MOVE!WRTEBR!IBUS!<MODEM>>
 BRSHFT
 MICPC=MICPC+1
 <MOVE!SHFTBR!WRTEBR!SELB>
 BR4 I1 ; RTS SET? IF SO WE ARE--STALL
 MICPC=MICPC+1
 <JUMP!BR4CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
 BRWRT IMM, 373 ; MASK TO TURN OFFLINE IDLE
 MICPC=MICPC+1

```

( ) 014064 000773      <MOVE!WRTEBR!IMM!<373>>
(1)
1314 014066           SP      BR,AANDB,SP10      ;IN LINE STATUS WORD
(1) 001040      MICPC=MICPC+1
(1) 014066 063270      <MOVE!SPX!BR!AANDB!SP10>
(1)
1315 014070           TSTATE TMTA1
(1) 001041      MICPC=MICPC+1
(1) 014070 000445      <MOVE!WRTEBR!IMM!<TMTA1-INIT&777/2>>
(1) 001042      MICPC=MICPC+1
(1) 014072 053222      <MOVE!SPX!BR!SELB!SP2>
1316           001      .IF NDF $LOW
1317           000      BRWRTE IMM,12      ;SET UP NUMBER OF SYNS
1318           001      .ENDC
1319           001      .IF DF $LOW
1320 014074           BRWRTE IMM,10
(1) 001043      MICPC=MICPC+1
(1) 014074 000410      <MOVE!WRTEBR!IMM!<10>>
(1)
1321           000      .ENDC
1322 014076           SP      BR,SELB,SP6      ;STORE IN SP6
(1) 001044      MICPC=MICPC+1
(1) 014076 063226      <MOVE!SPX!BR!SELB!SP6>
(1)
1323 014100           TMTA1: SP      BR,DECA,SP6      ;DECREMENT SYN COUNT
(1) 001045      MICPC=MICPC+1
(1) 014100 063166      <MOVE!SPX!BR!DECA!SP6>
(1)
1324 014102           Z      TMTEXT
(1) 001046      MICPC=MICPC+1
(1) 014102 111412      <JUMP!ZCOND!<TMTEXT-INIT&3000*4>!<TMTEXT-INIT&777/2>>
(1)
1325 014104           BRWRTE IMM,1      ;MASK FOR SOM
(1) 001047      MICPC=MICPC+1
(1) 014104 000401      <MOVE!WRTEBR!IMM!<1>>
(1)
1326 014106           OUTPUT BR,SELB!OTMTCO      ;WRITE IT TO TMTR CONTROL
(1) 001050      MICPC=MICPC+1
(1) 014106 062231      <MOVE!WROUT!BR!<SELB!OTMTCO>>
(1)
1327 014110           BRWRTE IMM,226      ;SYNC CHAR
(1) 001051      MICPC=MICPC+1
(1) 014110 000626      <MOVE!WRTEBR!IMM!<226>>
(1)
1328 014112           ALWAYS TMTA5
(1) 001052      MICPC=MICPC+1
(1) 014112 110421      <JUMP!ALCOND!<TMTA5-INIT&3000*4>!<TMTA5-INIT&777/2>>
(1)

```

```

1330          .SBTTL TMTB--OUTPUT FIRST CHAR OF COUNT
1331
1332 014114    001053    TMTB:  SPBR  IBUS,NPR,SPD          ;READ BR CONTRCL REG
      (1)      123600    <MOVE!SPBRX!IBUS!NPR!SPD>
      (1)
1333 014116    001054    BRO  IDLE          ;NPR GOING--GET OUT
      (1)      102045    MICPC=MICPC+1
      (1)      <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      (1)
1334 014120    001055    LDMA  IMM,LTC          ;GETPOINTER TO NEXT TMT LINK
      (1)      001      MICPC=MICPC+1
      (1)      010071    .IF IDN IMM,IMM
      (1)      <MOVE!LDMAR!IMM!<LTC&377>>
      (1)      .IFF
      (1)      <MOVE!LDMAR!IMM!<LTC>>
      (1)      .ENDC
1335 014122    001056    LDMA  MEMX,SELB!SPX!SP16 ;POINT TO THE LINK
      (1)      001      MICPC=MICPC+1
      (1)      .IF IDN MEMX,IMM
      (1)      <MOVE!LDMAR!IMM!<SELB!SPX!SP16&377>>
      (1)      .IFF
      (1)      053236    <MOVE!LDMAR!MEMX!<SELB!SPX!SP16>>
      (1)      000      .ENDC
1336 014124    001057    MEMINC IMM,3          ;WRITE MSG TMTED TO FLAGS
      (1)      016403    MICPC=MICPC+1
      (1)      <MOVE!WRMEM!INCMAR!IMM!<3>>
      (1)
1337 014126    001060    MEMINC DP,SELA!SP12   ;PICK UP MSGNO
      (1)      076612    MICPC=MICPC+1
      (1)      <MOVE!WRMEM!INCMAR!DP!<SELA!SP12>>
      (1)
1338 014130    001061    STATE TMTC          ;ADDRESS TMTR STATE
      (1)      000500    MICPC=MICPC+1
      (1)      <MOVE!WRTEBR!IMM!<TMTC-INIT&777/2>>
1339 014132    001062    TBO:  SP  BR,SELB,SP2   ;UPDATE IT
      (1)      063222    MICPC=MICPC+1
      (1)      <MOVE!SPX!BR!SELB!SP2>
      (1)
1340 014134    001063    OUTPUT <MEMX!INCMAR>,SELB!IBA1 ;WRITE LOW BYTE OF ADDRESS
      (1)      056224    MICPC=MICPC+1
      (1)      <MOVE!WROUT!MEMX!INCMAR!<SELB!IBA1>>
      (1)
1341 014136    001064    OUTPUT <MEMX!INCMAR>,SELB!IBA2 ;WRITE HIGH BYTE OF ADDRESS
      (1)      056225    MICPC=MICPC+1
      (1)      <MOVE!WROUT!MEMX!INCMAR!<SELB!IBA2>>
      (1)
1342 014140    001065    SP  MEMX,SELB,SP7   ;HIGH BYTE OF COUNT TO SP7
      (1)      043227    MICPC=MICPC+1
      (1)      <MOVE!SPX!MEMX!SELB!SP7>
      (1)
1343
1344 014142    001066    SP  IMM,300,SPD     ;WAIT TO MASK OFF MEM EXT. BITS
      (1)      MICPC=MICPC+1          ;MASK FOR MXT

```

```

(1) 014142 003300 <MOVE!SPX!IMM!300!SP0>
(1)
1345 014144 BRWRTE MEMX!INCMAR,AANDB!SP0 :TURN OFF CC2
(1) 001067 MICPC=MICPC+1
(1) 014144 054660 <MOVE!WRTEBR!MEMX!INCMAR!<AANDB!SP0>>
(1)
1346 014146 BRSHFT :SHIFT BITS INTO CORRECT POSITION
(1) 001070 MICPC=MICPC+1
(1) 014146 001620 <MOVE!SHFTBR!WRTEBR!SELB>
(1)
1347 014150 BRSHFT
(1) 001071 MICPC=MICPC+1
(1) 014150 001620 <MOVE!SHFTBR!WRTEBR!SELB>
(1)
1348 014152 BRSHFT
(1) 001072 MICPC=MICPC+1
(1) 014152 001620 <MOVE!SHFTBR!WRTEBR!SELB>
(1)
1349 014154 BRSHFT
(1) 001073 MICPC=MICPC+1
(1) 014154 001620 <MOVE!SHFTBR!WRTEBR!SELB>
(1)
1350 014156 OUT BR,SELB!ONPR 4
(1) 001074 MICPC=MICPC+1
(1) 014156 061230 <MOVE!WROUTX!BR!<SELB!ONPR>>
(1)
1351 014160 SPBR MEMX!INCMAR,SELB,SP6 :LOWBYTE OF COUNT TO SP6
(1) 001075 MICPC=MICPC+1
(1) 014160 057626 <MOVE!SPBRX!MEMX!INCMAR!SELB!SP6>
(1)
1352 014162 OUTPUT BR,SELB!TMTDAT :WRITE IT TO TMR SILO
(1) 001076 MICPC=MICPC+1
(1) 014162 062230 <MOVE!WROUT!BR!<SELB!TMTDAT>>
(1)
1353 014164 ALWAYS IDLE
(1) 001077 MICPC=MICPC+1
(1) 014164 100445 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1)
1354 :
```

```

1356 .SBTTL TMTD--OUTPUT SECOND CHAR OF COUNT
1357
1358 014166 001100 TMTD: BRWRT IMM,77 ;MASK TO CLEAR MXT BITS
      (1) 014166 000477 MICPC=MICPC+1
      (1) <MOVE!WRTEBR!IMM!<77>>
1359 014170 001101 SPBR BR,AANDB,SP7 ;CLEAR THEM
      (1) 014170 063667 MICPC=MICPC+1
      (1) <MOVE!SPBRX!BR!AANDB!SP7>
1360 014172 001102 OUTPUT DP,<SELB!TMTDAT> ;WRITE TO TMT SILC
      (1) 014172 062230 MICPC=MICPC+1
      (1) <MOVE!WROUT!DP!<SELB!TMTDAT>>
1361 014174 001103 LDMA IMM,LTC ;POINT TO TMT BUFFER
      (1) 014174 001001 MICPC=MICPC+1
      (1) .IF IDN IMM,IMM
      (1) <MOVE!LDMAR!IMM!<LTC&377>>
      (1) .IFF
      (1) <MOVE!LDMAR!IMM!<LTC>>
      (1) .ENDC
1362 014176 001104 BRWRT IMM,6 ;OFFSET TO NEXT LINK
      (1) 014176 000406 MICPC=MICPC+1
      (1) <MOVE!WRTEBR!IMM!<6>>
1363 014200 001105 MEM BR,ADD!SP16 ;UPDATE THE PCINTER
      (1) 014200 062416 MICPC=MICPC+1
      (1) <MOVE!WRMEM!BR!<ADD!SP16>>
1364 014202 001106 BRWRT IMM,TMLB ;GET WRAPAROUND ADDRESS
      (1) 014202 000545 MICPC=MICPC+1
      (1) <MOVE!WRTEBR!IMM!<TMLB>>
1365 014204 001107 CMP BR,SP16 ;WRAPAROUND
      (1) 014204 060375 MICPC=MICPC+1
      (1) <SUBTC!BR!SP16>
1366 014206 001110 Z 105
      (1) 014206 111513 MICPC=MICPC+1
      (1) <JUMP!ZCOND!<105-INIT&3000*4>!<105-INIT&777/2>>
1367 014210 001111 SS: STATE TMTD
      (1) 014210 000515 MICPC=MICPC+1
      (1) <MOVE!WRTEBR!IMM!<TMTD-INIT&777/2>>
1368 014212 001112 ALWAYS XEXIT
      (1) 014212 100451 MICPC=MICPC+1
      (1) <JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
1369 014214 001113 105: MEM IMM,TML1 ;GO BACK TO FIRST LINK
      (1) 014214 002473 MICPC=MICPC+1
      (1) <MOVE!WRMEM!IMM!<TML1>>
1370 014216 001114 ALWAYS SS
      (1) 014216 110511 MICPC=MICPC+1
      (1) <JUMP!ALCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
    
```

G11

MACY: 00000 PROTOCOL IMPLEMENTATION
0000000000 06-DEC-76 10:14

MACY11 27(1006) 07-MAR-77 16:24 PAGE 6-70
*MFC--OUTPUT SECOND CHAR OF COUNT

PAGE: 0136

1371

:

H11

EMUL: DDUMP PROTOCOL IMPLEMENTATION
DDNEW.MAC 06-DEC-76 10:17

MACY11 27.1006) 07-MAR-77 16:24 PAGE 6-71
TMD--RESPONSE FIELD-NUMBERED MESSAGE

PAGE: 0137

```

1373
1374 014220 001115
      (1) 000523
1375 014222 001116
      (1) 06316E
      (1)
1376 014224 001117
      (1) 111121
      (1)
1377 014226 001120
      (1) 063157
      (1)
1378 014230 001121
      (1) 001
      (1) 010171
      (1)
      (1) 000
1379 001
1380

```

```

.SBTTL TMD--RESPONSE FIELD-NUMBERED MESSAGE
TMD: STATE TMT
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMT-INIT$777/2>>
      SP BR,DECA,SP6 :ADJUST COUNT FOR TWO'S COMPLEMENT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP6>

      C TD2 :NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<TD2-INIT$3000*4>!<TD2-INIT$777/2>>

      SP BR,DECA,SP7 :DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP7>

T02: LDMA IMM,ISP11 :RESP FIELD ADDR TO MAR
      MICPC=MICPC+1
      .IF IDN IMM,IMM
      <MOVE!LDMAR!IMM!<ISP11&377>>
      .IFF
      <MOVE!LDMAR!IMM!<ISP11>>
      .ENDC

      .IF NDF LOW
      OUTPUT MEMX,SELB!TMDAT :WRITE IT TO SILC

```

MAC: 30000 PROTOCOL IMPLEMENTATION
30000.MAC 06-DEC-76 10:14

1385

I11

MACY11 27(1006) 07-MAR-77 16:24 PAGE 10
TMD--RESPONSE FIELD-NUMBERED MESSAGE

ALWAYS KEXIT

PAGE: 0125

J11

DMC: DDMP PROTOCOL IMPLEMENTATION
DDNEW.MAC 06-DEC-76 10:14

MACY11 27(1006) 07-MAR-77 16:24 PAGE 11
TMTD--RESPONSE FIELD-NUMBERED MESSAGE

PAGE: 0129

1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400

014232
014232

001122
110636

000

.ENDC
.IF DF LOW
ALWAYS TJI
MICPC=MICPC+1
JUMP!ALCOND! TJI-INIT\$3000+4! TJI-INIT\$777.E
.ENDC

K11

EMC: DOOMB PROTOCOL IMPLEMENTATION
DOONEW.MAC 06-DEC-76 10:14

MACY11 27(1006) 07-MAR-77 16.24 PAGE 12
TMTE--NUMBER FIELD--NUMBERED MESSAGE

PAGE: 0140

1393
 1394 014234
 1395 014234
 (1) 001123
 (1) 014234 123600
 (1)
 1396 014236
 (1) 001124
 (1) 014236 102052
 (1)
 1397 014240
 (1) 001125
 (1) 014240 060612
 (1)
 1398 014242
 (1) 001126
 (1) 014242 062230
 (1)
 1399 014244
 (1) 001127
 (1) 014244 000531
 1400 014246
 (1) 001130
 (1) 014246 110570
 (1)

.SBTTL TMTE--NUMBER FIELD--NUMBERED MESSAGE
 TMTE: SPBR IBUS,NPR,SPO ;READ NPR CONTROL REGISTER
 MICPC=MICPC+1
 <MOVE!SPBRX!IBUS!NPR!SPO>
 BR0 I1 ;BLSY - GET OUT
 MICPC=MICPC+1
 <JUMP!BR0CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
 BRWRT BR,SELA!SP12
 MICPC=MICPC+1
 <MOVE!WRTEBR!BR!<SELA!SP12>>
 OUTPUT BR,<SELB!TMTDAT> ;WRITE IT TO THE SILO
 MICPC=MICPC+1
 <MOVE!WR0UT!BR!<SELB!TMTDAT>>
 STATE TMTF
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<TMTF-INIT&777/2>>
 ALWAYS TH3
 MICPC=MICPC+1
 <JUMP!ALCOND!<TH3-INIT&3000*4>!<TH3-INIT&777/2>>

1402
 1403
 1404 014250
 (1) 001131
 (1) 014250 000535
 1405 014252
 (1) 001132
 (1) 014252 063222
 (1)
 1406 014254
 (1) 001133
 (1) 014254 000401
 (1)
 1407 001
 1408
 1409
 1410 000
 1411 001
 1412 014256
 (1) 001134
 (1) 014256 113421
 (1)
 1413 000

```

.SBTTL TMTF--NUMBERED MSG ADDRESS FIELD
TMTF: STATE TF1
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TF1-INIT&777/2>>
TF2:  SP BR SELB,SP2
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>

BRWRTE IMM,1          :LOAD ADDRESS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<1>>

  .IF NDF LOW
  OUTPUT BR,<SELB!TMTDAT>
  ALWAYS I1
  .ENDC
  .IF DF LOW
  ALWAYS TMTAS
  MICPC=MICPC+1
  <JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>
  .ENDC

```

M11

DMC1: DDUMP PROTOCOL IMPLEMENTATION
 DDNEW.MAC 06-DEC-76 10:14

MACY11 27(1006) 07-MAR-77 16:24 PAGE 12-2
 TF1-NUMBERED MSG HEADER EOM

PAGE: 0142

1415			.SBTTL TF1-NUMBERED MSG HEADER EOM	
1416	014260		TF1: BRWRT IMM,2	;EOM MASK TO BR
(1)		001135	MICPC=MICPC+1	
(1)	014260	000402	<MOVE!WRTEBR!IMM!<2>>	
(1)				
1417	014262		OUTPUT BR,<SELB!OTMTCO>	;UPDATE TMTR CONTROL REGISTER
(1)		001136	MICPC=MICPC+1	
(1)	014262	062231	<MOVE!WROUT!BR!<SELB!OTMTCO>>	
(1)				
1418	014264		OUTPUT BR,<SELB!TMTDAT>	;OUTPUT A GARBAGE CHAR
(1)		001137	MICPC=MICPC+1	
(1)	014264	062230	<MOVE!WROUT!BR!<SELB!TMTDAT>>	
(1)				
1419	014266		BRWRT IBUS,IIBA1	;READ LOW ORDER FROM INBA
(1)		001140	MICPC=MICPC+1	
(1)	014266	020500	<MOVE!WRTEBR!IBUS!<IIBA1>>	
(1)				
1420	014270		BRO TMTF1	;IF ODD BYTE--BRANCH
(1)		001141	MICPC=MICPC+1	
(1)	014270	112155	<JUMP!BROCON!<TMTF1-INIT&3000*4>!<TMTF1-INIT&777/2>>	
(1)				
1421	014272		STATE TMTH	
(1)		001142	MICPC=MICPC+1	
(1)	014272	000544	<MOVE!WRTEBR!IMM!<TMTH-INIT&777/2>>	
1422	014274		ALWAYS XEXIT	
(1)		001143	MICPC=MICPC+1	
(1)	014274	100451	<JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>	
(1)				

1424
 1425
 1426
 1427 014276 001144
 (1) 014276 123600
 (1)
 (1)
 1428 001
 1429
 1430 000
 1431 014300 001145
 (1) 014300 102052
 (1)
 (1)
 1432 014302 001146
 (1) 014302 022010
 (1)
 (1)
 1433 014304 001147
 (1) 014304 023100
 (1)
 (1)
 1434 014306 001150
 (1) 014306 062064
 (1)
 (1)
 1435 014310 001151
 (1) 014310 063166
 (1)
 (1)
 1436 014312 001152
 (1) 014312 111155
 (1)
 (1)
 1437 014314 001153
 (1) 014314 063167
 (1)
 (1)
 1438 014316 001154
 (1) 014316 115402
 (1)
 (1)
 1439 014320 001
 1440 001
 1441
 1442
 1443 000
 1444 014320 001155
 (1) 014320 000557
 (1)
 1445 014322 001156
 (1) 014322 100451
 (1)
 (1)
 1446 014324 001
 1447

```

:*****TIME CRITICAL PATH--MODIFY WITH GREAT CARE
.SBTTL TMTM--ROUTINE TO OUTPUT DATA CHARACTERS
TMTM:  SPBR  IBUS,NPR,SPO          ;READ NPR CONTROL
      MICPC=MICPC+1
      <MOVE!SPBRX!IBUS!NPR!SPO>

      .IF NDF LOW
      BR4  5$                      ;IF RECV NPR --PROCESS
      .ENDC
      BRO  I1                      ;IF NPR IN PROGRESS --BRANCH
      MICPC=MICPC+1
      <JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>

5$:  OUTPUT IBUS,<INDAT1!TMTDAT>    ;WRITE THE EVEN CHAR TO TMT-SILO
      MICPC=MICPC+1
      <MOVE!WROUT!IBUS!<INDAT1!TMTDAT>>

      SP  IBUS,IIBA1,SPO          ;READ LOW BYTE OF BA TO SP
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IIBA1!SPO>

      OUTPUT BR,<INCA!IBA1>        ;OUTPUT INCREMENTED BA
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<INCA!IBA1>>

      SP  BR,DECA,SP6            ;DECREMENT CHARACTER COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP6>

      C  TH6                      ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<TH6-INIT&3000*4>!<TH6-INIT&777/2>>

      SP  BR,DECA,SP7            ;DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP7>

      Z  HEH1                      ;BYTE COUNT ZERO
      MICPC=MICPC+1
      <JUMP!ZCOND!<HEH1-INIT&3000*4>!<HEH1-INIT&777/2>>

TH6:  .IF NDF LOW
      BRWRT IBUS,TMTCON          ;READ TMTM CONTROL CSR
      BR4  TH9                    ;IF MORE ROOM IN SILO--BRANCH
      .ENDC

TMTM1: STATE TMTM0
      MICPC=MICPC+1
      <MOVE!WRTBR!IMM!<TMTM0-INIT&777/2>>
      ALWAYS XEXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>

TMTM0: .IF DF LOW
    
```

1448	014324		SPBR IBUS,NPR,SPO	:NPR BUSY
(1)		001157	MICPC=MICPC+1	
(1)	014324	123600	<MOVE!SPBRX!IBUS!NPR!SPO>	
(1)				
1449	014326		BRO I1	
(1)		001160	MICPC=MICPC+1	
(1)	014326	102052	<JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>	
(1)				
1450		000	.ENDC	
1451	014330		TH9: OUTPUT IBUS,<INDAT2!TMTDAT>	;ODD CHAR TO SILC
(1)		001161	MICPC=MICPC+1	
(1)	014330	022030	<MOVE!WROUT!IBUS!<INDAT2!TMTDAT>>	
(1)				
1452	014332		SP IBUS,IIBA1,SPO	:READ LOW BYTE TO BA
(1)		001152	MICPC=MICPC+1	
(1)	014332	023100	<MOVE!SPX!IBUS!IIBA1!SPO>	
(1)				
1453	014334		OUTPUT BR,<INCA!IBA1>	;OUTPUT THE INCREMENTED BA
(1)		001163	MICPC=MICPC+1	
(1)	014334	062064	<MOVE!WROUT!BR!<INCA!IBA1>>	
(1)				
1454	014336		C HOINCH	
(1)		001164	MICPC=MICPC+1	
(1)	014336	111372	<JUMP!CCOND!<HOINCH-INIT&3000*4>!<HOINCH-INIT&777/2>>	
(1)				
1455	014340		TH8: SP BR,DECA,SP6	;DECREMENT CHARACTERCOUNT
(1)		001165	MICPC=MICPC+1	
(1)	014340	063166	<MOVE!SPX!BR!DECA!SP6>	
(1)				
1456	014342		C TH7	:NO OVERFLOW
(1)		001166	MICPC=MICPC+1	
(1)	014342	111171	<JUMP!CCOND!<TH7-INIT&3000*4>!<TH7-INIT&777/2>>	
(1)				
1457	014344		SP BR,DECA,SP7	:DECREMENT HIGH BYTE OF COUNT
(1)		001167	MICPC=MICPC+1	
(1)	014344	063167	<MOVE!SPX!BR!DECA!SP7>	
(1)				
1458	014346		Z HEH1	:BYTE COUNT ZERO
(1)		001170	MICPC=MICPC+1	
(1)	014346	115402	<JUMP!ZCOND!<HEH1-INIT&3000*4>!<HEH1-INIT&777/2>>	
(1)				
1459	014350		TH7: SPBR IBUS,NPR,SPO	:READ NPR REGISTER
(1)		001171	MICPC=MICPC+1	
(1)	014350	123600	<MOVE!SPBRX!IBUS!NPR!SPO>	
(1)				
1460		001	.IF NDF LOW	
1461			BRO TH2	:IF NPR BUSY WAIT TO GO
1462		000	.ENDC	
1463	014352		STATE TMTH	
(1)		001172	MICPC=MICPC+1	
(1)	014352	000544	<MOVE!WRTBR!IMM!<TMTH-INIT&777/2>>	
(1)				
1464	014354		TH3: SP BR,SELB,SP2	:SAVE ISTATE
(1)		001173	MICPC=MICPC+1	
(1)	014354	063222	<MOVE!SPX!BR!SELB!SP2>	
(1)				
1465	014356		TH3X: BRWRT IMM,157	:CLEAR CO AND C1

```

(1)
(1) 014356 000557
(1)
1466 014360
(1) 001175
(1) 014360 063260
(1)
1467 014362
(1) 001176
(1) 014362 000401
(1)
1468          001
1469
1470
1471
1472
1473          000
1474          001
1475 014364
(1) 001177
(1) 014364 104642
(1)
1476          000
1477
1478

```

```

MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<157>>

SP      BR,AANDB,SP0          :CLEAR THE BITS
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SP0>

BRWRTE IMM,1                  :WRITE NPR BITS TO BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<1>>

      .IF NDF LOW
      OUT   BR,<AORB!CNPR>
      ALWAYS I1
      STATE TH7
      ALWAYS XEXIT
      .ENDC
      .IF DF LOW
      ALWAYS RK7
      MICPC=MICPC+1
      <JUMP!ALCOND!<RK7-INIT&3000*4>!<RK7-INIT&777/2>>
      .ENDC
:*****END TIME CRITICAL PATH*****
:

```

TH2:

1480
 1481 014366 001200
 (1) 001
 (1) 014366 010153
 (1)
 (1) 000
 (1)
 1482 014370 001201
 (1) 043226
 (1)
 1483 014372 001202
 (1) 014372 000604
 1484 014374 001203
 (1) 014374 110606
 (1)
 1495
 1486
 1487 014376 001204
 (1) 001
 (1) 014376 010154
 (1)
 (1) 000
 (1)
 1488 014400 001205
 (1) 014400 000610
 1489 014402 001206
 (1) 014402 042230
 (1)
 1490 014404 001207
 (1) 014404 100451
 (1)
 1491

```

TMTI: .SBTTL TMTI--SEND UNNUMBERED TYPE FIELD
      LDMA IMM,T ;ADDRESS OF TYPE FIELD TO MAR
      MICPC=MICPC+1
      .IF IDN IMM,IMM
      <MOVE!LDMAR!IMM!<T&377>>
      .IFF
      <MOVE!LDMAR!IMM!<T>>
      .ENDC

      SP MEMX SELB,SP6 ;COPY IT TO SP6
      MICPC=MICPC+1
      <MOVE!SPX!MEMX!SELB!SP6>

STATE TMTJ
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTJ-INIT&777/2>>
ALWAYS TJ1
MICPC=MICPC+1
<JUMP!ALCOND!<TJ1-INIT&3000*4>!<TJ1-INIT&777/2>>

TMTJ: .SBTTL TMTJ--SEND SUB-TYPE FIELD
      LDMA IMM,ST ;ADDRESS OF SUB-TYPE FIELD TO MAR
      MICPC=MICPC+1
      .IF IDN IMM,IMM
      <MOVE!LDMAR!IMM!<ST&377>>
      .IFF
      <MOVE!LDMAR!IMM!<ST>>
      .ENDC

STATE TMTK
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTK-INIT&777/2>>
TJ1: OUTPUT MEMX SELB!TMTDAT
      MICPC=MICPC+1
      <MOVE!WROUT!MEMX!<SELB!TMTDAT>>

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

```



```

1493 .SBTTL TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG.)
1494
1495 014406 TMTK: BRWRTE IMM,3 ;WRITE A 3 TO BR
(1) 001210 MICPC=MICPC+1
(1) 014406 000403 <MOVE!WRTEBR!IMM!<3>>
(1)
1496 014410 NOP BR,SP6 ;IF TYPE LESS THAN 3
(1) 001211 MICPC=MICPC+1
(1) 014410 060346 <BR!SUB!SP6>
(1)
1497 014412 TSTATE TMTL
(1) 001212 MICPC=MICPC+1
(1) 014412 000616 <MOVE!WRTEBR!IMM!<TMTL-INIT&777/2>>
(1) 001213 MICPC=MICPC+1
(1) 014414 063222 <MOVE!SPX!BR!SELB!SP2>
1498 014416 C TMTLO
(1) 001214 MICPC=MICPC+1
(1) 014416 111223 <JUMP!CCOND!<TMTLO-INIT&3000*4>!<TMTLO-INIT&777/2>>
(1)
1499 014420 ALWAYS TD2
(1) 001215 MICPC=MICPC+1
(1) 014420 110521 <JUMP!ALCOND!<TD2-INIT&3000*4>!<TD2-INIT&777/2>>
(1)
1500
1501
1502 014422 TMTL: .SETTL TMTL--UNNUMB MSG NUMBER FIELD
(1) 001216 TSTATE TMTM
(1) 014422 000627 MICPC=MICPC+1
(1) 001217 <MOVE!WRTEBR!IMM!<TMTM-INIT&777/2>>
(1) 014424 063222 MICPC=MICPC+1
(1) <MOVE!SPX!BR!SELB!SP2>
1503 014426 BRWRTE IMM,3
(1) 001220 MICPC=MICPC+1
(1) 014426 000403 <MOVE!WRTEBR!IMM!<3>>
(1)
1504 014430 CMP BR,SP6 ;IS MESSAGE REP
(1) 001221 MICPC=MICPC+1
(1) 014430 060366 <SUBTC!BR!SP6>
(1)
1505 014432 Z TMTL1 ;YES
(1) 001222 MICPC=MICPC+1
(1) 014432 111625 <JUMP!ZCOND!<TMTL1-INIT&3000*4>!<TMTL1-INIT&777/2>>
(1)
1506 014434 TMTLO: BRWRTE IMM,0 ;ADDRESS CONTNAT OF ZERO
(1) 001223 MICPC=MICPC+1
(1) 014434 000400 <MOVE!WRTEBR!IMM!<0>>
(1)
1507 014436 ALWAYS TMTA5
(1) 001224 MICPC=MICPC+1
(1) 014436 110421 <JUMP!ALCOND!<TMTA5-INIT&3000*4>!<TMTA5-INIT&777/2>>
(1)
1508
1509 014440 TMTL1: BRWRTE BR,DECA!SP12 ;WRITE A RESPONSE
(1) 001225 MICPC=MICPC+1
(1) 014440 060572 <MOVE!WRTEBR!BR!<DECA!SP12>>
(1)
1510 001 .IF NDF LOW

```

1511
1512
1513
1514
1515 014442
1516
1517 001226
1518 014442 110421
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600

OUTPUT BR,SELB!TMTDAT
ALWAYS I1
.ENDC
.IF DF LOW
ALWAYS TMTAS
MICPC=MICPC+!
.JUMP!ALCOND!<TMTAS-INIT\$2000*4>!<TMTAS-INIT\$777.2>
.ENDC
:

```

1519
1520 014444 (1) 001227
(1) 014444 000531
1521 014446 (1) 001230
(1) 014446 110532
(1)
1522 014450 (1) 001231
(1) 014450 000402
(1)
1523 014452 (1) 001232
(1) 014452 062231
(1)
1524 014454 (1) 001233
(1) 014454 062230
(1)
1525 014456 (1) 001234
(1) 014456 000404
(1)
1526 014460 (1) 001235
(1) 014460 063710
(1)
1527 014462 (1) 001236
(1) 014462 060530
(1)
1528 014464 (1) 001237
(1) 014464 113643
(1)
1529 014466 (1) 001240
(1) 014466 000776
(1)
1530 014470 (1) 001241
(1) 014470 063270
(1)
1531 014472 (1) 001242
(1) 014472 110734
(1)
1532 014474 (1) 001243
(1) 014474 000576
(1)
1533 014476 (1) 001244
(1) 014476 110541

```

```

.SBTTL TMTM--UNNUMB MSG--STATION ADDRESS
TMTM: STATE TNEOM
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TNEOM-INIT&777/2>>
ALWAYS TF2
MICPC=MICPC+1
<JUMP!ALCOND!<TF2-INIT&3000*4>!<TF2-INIT&777/2>>

TNEOM: BRWRTE IMM,2 ;END OF MESSAGE TO BF
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>>

OUTPUT BR,<SELB!OTMTCO>
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELB!OTMTCO>>

OUTPUT BR,<SELB!TMTDAT> ;OUTPUT A GARBAGE CHARACTER
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELB!TMTDAT>>

BRWRTE IMM,4 ;SET UP LINE HAS GONE IDLE MASK
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<4>>

SPBR BR,AORB,SP10 ;UPDATE LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPBRX!BR!AORB!SP10>

BRWRTE BR,AA!SP10 ;SHIFT STATUS LEFT
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AA!SP10>>

BR7 105 ;IF HDX SET---BRANCH TO CLEAR OK TO SEND
MICPC=MICPC+1
<JUMP!BR7CON!<105-INIT&3000*4>!<105-INIT&777/2>>

BRWRTE IMM,376 ;MASK TO TURN OFF UNNUMB PENDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<376>>

SS: SP BR,AANDB,SP10 ;MASK TO LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SP10>

ALWAYS TEOM2
MICPC=MICPC+1
<JUMP!ALCOND!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>

105: BRWRTE IMM,176 ;CLEAR OK TO SEND AND UNNUMB PENDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<176>>

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

```

```

1535 .SBTTL TIMSRV--TIMEOUT ROUTINE--SENDS REP
1536 :
1537 :ENABLE LSB
1538 014500 001245 TIMSRV: SP IBUS,UBBR,SPC :READ UNIBUS BR REGISTER
1539 014500 123220 MICPC=MICPC+1
1540 :MOVE!SPX!IBUS!UBBR!SPC)
1541 :
1542 BRWRTE IMM,177 :MASK OFF BR REG
1543 014502 001246 MICPC=MICPC+1
1544 014502 000577 <MOVE!WRTEBR!IMM!<177>>
1545 :
1546 :RESET TIMER---SLICK MOVE
1547 :SINCE TIMER IS RESET BY WRITING
1548 :A 1 AND THE EXPIRATION LOOKS
1549 :LIKE 1--VCILA
1550 014504 OUT BR,<AANDB!OBR> :AND THE BIT ON
1551 014504 001247 MICPC=MICPC+1
1552 014504 061271 <MOVE!WROUTX!BR!<AANDB!OBR>>
1553 :
1554 BRWRTE BR,SELA!SP1 :READ STATUS BYTE
1555 014506 001250 MICPC=MICPC+1
1556 014506 060601 <MOVE!WRTEBR!BR!<SELA!SP1>>
1557 :
1558 BR7 IDLE :IF IN MAINT. MODE DISABLE TIMER
1559 014510 001251 MICPC=MICPC+1
1560 014510 103445 <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1561 :
1562 SP BR,DECA,SP15 :DECREMENT THE COUNTER
1563 014512 001252 MICPC=MICPC+1
1564 014512 063175 <MOVE!SPX!BR!DECA!SP15>
1565 :
1566 Z 20$ :IF ALL ONES HAS EXPIRED
1567 014514 001253 MICPC=MICPC+1
1568 014514 111662 <JUMP!ZCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
1569 :
1570 10$: BRWRTE BR,SELA!SP10 :READ LINE STATUS
1571 014516 001254 MICPC=MICPC+1
1572 014516 060610 <MOVE!WRTEBR!BR!<SELA!SP10>>
1573 :
1574 BR1 TABUPD ;NUMBERED MESSAGE IN PROGRESS
1575 014520 001255 MICPC=MICPC+1
1576 014520 116737 <JUMP!BR1CON!<TABUPD-INIT&3000*4>!<TABUPD-INIT&777/2>>
1577 :
1578 BR0 TABUPD ;UNNUMBMSGIN PROGRESS
1579 014522 001256 MICPC=MICPC+1
1580 014522 116337 <JUMP!BR0CON!<TABUPD-INIT&3000*4>!<TABUPD-INIT&777/2>>
1581 :
1582 BRSHFT
1583 014524 001257 MICPC=MICPC+1
1584 014524 001620 <MOVE!SHFTBR!WRTEBR!SELB>
1585 :
1586 BR4 IDLE :START MODE
1587 014526 001260 MICPC=MICPC+1
1588 014526 103045 <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1589 :
1590 ALWAYS SNDACK ;ELSE SEND ACK

```

```

(1) 014530 001261 MICPC=MICPC+1
(1) 014530 000727 <JUMP!ALCOND!<SNDACK-INIT&3000*4!<SNDACK-INIT&777/2>>

1555 014532 TIME1:
1556 014532 205: BRWRTE IMM,2
(1) 014532 001262 MICPC=MICPC+1
(1) 014532 000402 <MOVE!WRTEBR!IMM!<2>>

1557 014534 SP BR,SELB,SP15 ;RESET THE TIMER TICK COUNT
(1) 014534 001262 MICPC=MICPC+1
(1) 014534 06323E <MOVE!SPX!BR!SELB!SP15>

1558 014536 BRWRTE DP,<SELA!SP10> ;READ LINE STATUS WORD
(1) 014536 001264 MICPC=MICPC+1
(1) 014536 060E10 <MOVE!WRTEBR!DP!<SELA!SP10>>

1559 014540 BRSHFT
(1) 014540 001265 MICPC=MICPC+1
(1) 014540 001620 <MOVE!SHFTBR!WRTEBR!SELB>

1560 014542 BR4 BS1 ;IF IN START MODE--BRANCH
(1) 014542 001266 MICPC=MICPC+1
(1) 014542 103076 <JUMP!BR4CON!<BS1-INIT&3000*4!<BS1-INIT&777/2>>

1561 014544 BRWRTE BR,DECA!SP12 ;GET LAST NUMBER SENT
(1) 014544 001267 MICPC=MICPC+1
(1) 014544 060572 <MOVE!WRTEBR!BR!<DECA!SP12>>

1562 014546 CMP BR,SP17 ;COMPARE TO LAST ACKED
(1) 014546 001270 MICPC=MICPC+1
(1) 014546 060377 <SUBTC!BR!SP17>

1563 014550 Z 105 ;IF EQ --SEND ACK
(1) 014550 001271 MICPC=MICPC+1
(1) 014550 111654 <JUMP!ZCOND!<105-INIT&3000*4!<105-INIT&777/2>>

1564 014552 TIME2: LDMA IMM,T ;LOAD ADDRESS OF TYPE FIELD IN UNNUMB SK
(1) 014552 001272 MICPC=MICPC+1
(1) 014552 001 001 .IF IDN IMM,IMM
(1) 014552 010153 <MOVE!LDMAR!IMM!<T&377>>
(1) 014552 000 .IFF
(1) 014552 000 <MOVE!LDMAR!IMM!<T>>
(1) 014552 000 .ENDC

1565 014554 MEMINC IMM,3 ;LOAD REP TYPE
(1) 014554 001273 MICPC=MICPC+1
(1) 014554 016403 <MOVE!WRMEM!INCMAR!IMM!<3>>

1566 014556 MEMINC IMM,300 ;ZERO THE SUB-TYPE
(1) 014556 001274 MICPC=MICPC+1
(1) 014556 01E700 <MOVE!WRMEM!INCMAR!IMM!<300>>

1567 014560 LDMA IMM,REPCS ;CUMULATIVE REPS RECD
(1) 014560 001275 MICPC=MICPC+1
(1) 014560 001 001 .IF IDN IMM,IMM
(1) 014560 01001E <MOVE!LDMAR!IMM!<REPCS&377>>

```

(1)			.IFF		
(1)			<MOVE!LDMAR!IMM!<REPCS>>		
(1)			.ENDC		
1568	014562		SP MEMX SELB SPD	:COPY IT TO SPD	
(1)		001276	MICPC=MICPC+1		
(1)	014562	043220	<MOVE!SPX!MEMX!SELB!SPD>		
1569	014564		MEM BR INCA!SPD	:INCREMENT IT	
(1)		001277	MICPC=MICPC+1		
(1)	014564	062460	<MOVE!WRMEM!BR!<INCA!SPD>>		
1570	014566		LDMAR IMM REPST	:ADDRESS DYNAMIC REP COUNTER	
(1)		001300	MICPC=MICPC+1		
(1)		001	.IF IDN IMM IMM		
(1)	014566	010003	<MOVE!LDMAR!IMM!<REPST&377>>		
(1)			.IFF		
(1)			<MOVE!LDMAR!IMM!<REPST>>		
(1)			.ENDC		
1571	014570		BRWRTE MEMX SELB	:COPY IT TO THE BR	
(1)		001301	MICPC=MICPC+1		
(1)	014570	040620	<MOVE!WRTEBR!MEMX!<SELB>>		
1572	014572		BSHFTB		
(1)		001302	MICPC=MICPC+1		
(1)	014572	061620	<MOVE!SHFTBR!SELB!BR>		
1573	014574		MEM BR SELB		
(1)		001303	MICPC=MICPC+1		
(1)	014574	062620	<MOVE!WRMEM!BR!<SELB>>		
1574	014576		BRO RTHRES		
(1)		001304	MICPC=MICPC+1		
(1)	014576	102353	<JUMP!BROCON!<RTHRES-INIT&3000*4>!<RTHRES-INIT&777/2>>		
1575	014600		BRWRTE IMM 201	:MASK FOR OK TO SEND	
(1)		001305	MICPC=MICPC+1		
(1)	014600	000601	<MOVE!WRTEBR!IMM!<201>>		
1576	014602		SP BR AORB SP10	:OR IT IN	
(1)		001306	MICPC=MICPC+1		
(1)	014602	063310	<MOVE!SPX!BR!AORB!SP10>		
1577	014604		ALWAYS IDLE		
(1)		001307	MICPC=MICPC+1		
(1)	014604	100445	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>		
(1)			.DSABLE LSB		
1578			:		
1579			:		

```

1581 014506 001310
(1) 014606 120620
(1)
(1)
1582 014610 001311 ;NON-EXISTANT MEMORY
(1) 014610 116063 <JUMP!BROCON!<NXMERR-INIT&3000*4!<NXMERR-INIT&777/2>>
(1)
(1)
1583 014612 001312 :EOM TO BR
(1) 014612 090402 <MOVE!WRTEBR!IMM!<2>>
(1)
(1)
1584 014614 001313 :WRITE TMT CONTROL
(1) 014614 062231 <MOVE!WROUT!BR!<SELB!OTMTCO>>
(1)
(1)
1585 014616 001314 :WRITE GARBAGE DATA
(1) 014616 062230 <MOVE!WROUT!BR!<SELB!TMTDAT>>
(1)
(1)
1586 014620 001315 :CHECK FOR BOOT MODE
(1) 014620 060601 <MOVE!WRTEBR!BR!<SELA!SP1>>
(1)
(1)
1587 014622 001316 :---IF SET IS MAINT MSG
(1) 014622 113755 <JUMP!BR7CON!<BTEOM-INIT&3000*4!<BTEOM-INIT&777/2>>
(1)
(1)
1588 014624 001317 :INCREMENT THE MESSAGE NUMBER
(1) 014624 063072 <MOVE!SPX!BR!INCA!SP12>
(1)
(1)
1589 014626 001318 :ADDRESS LAST TMT LINK
(1) 014626 010071 LDMA IMM,LTC ;ADDRESS LAST TMT LINK
(1) 001 .IF IDN IMM,IMM
(1) <MOVE!LDMAR!IMM!<LTC&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<LTC>>
(1) .ENDC
(1) 000
(1)
1590 014630 001321
(1) 001 LDMA MEMX,SELB
(1) MICPC=MICPC+1
(1) .IF IDN MEMX,IMM
(1) <MOVE!LDMAR!IMM!<SELB&377>>
(1) .IFF
(1) <MOVE!LDMAR!MEMX!<SELB>>
(1) .ENDC
(1) 014630 050220
(1) 000
(1)
1591 014632 001322
(1) 014632 040620 BRWRTE MEMX,SELB
(1) MICPC=MICPC+1
(1) <MOVE!WRTEBR!MEMX!<SELB>>
(1)
(1)
1592 014634 001323
(1) 014634 112334 BRO TEOM2
(1) MICPC=MICPC+1
(1) <JUMP!BROCON!<TEOM2-INIT&3000*4!<TEOM2-INIT&777/2>>

```

1593	014636	001324	TEOM3: BRWRTE IMM,375	;TURN OFF MESSAGE PENDING
(1)		000775	MICPC=MICPC+1	
(1)	014636		<MOVE!WRTEBR!IMM!<375>>	
1594	014640	001325	SPBR BR,AANDB,SP10	
(1)		063670	MICPC=MICPC+1	
(1)	014640		<MOVE!SPBRX!BR!AANDB!SP10>	
1595	014642	001326	BRO TEOM2	;IF UNNUMB PENDING--GO AWAY
(1)		112334	MICPC=MICPC+1	
(1)	014642		<JUMP!BROCON!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>	
1596			.SBTTL SNDACK--ROUTINE TO SEND AN ACK	
1597	014644	001327	LDMA IMM,T	
(1)		001	MICPC=MICPC+1	
(1)	014644	010153	.IF IDN IMM,IMM	
(1)			<MOVE!LDMAR!IMM!<T&377>>	
(1)			.IFF	
(1)			<MOVE!LDMAR!IMM!<T>>	
(1)		000	.ENDC	
1598	014646	001330	MEMINC IMM,1	
(1)		016401	MICPC=MICPC+1	
(1)	014646		<MOVE!WRMEM!INCMAR!IMM!<1>>	
1599	014650	001331	BRWRTE IMM,5	
(1)		000405	MICPC=MICPC+1	
(1)	014650		<MOVE!WRTEBR!IMM!<5>>	
1600	014652	001332	SA2: MEMINC IMM,300	
(1)		016700	MICPC=MICPC+1	
(1)	014652		<MOVE!WRMEM!INCMAR!IMM!<300>>	
1601	014654	001333	SA3: SP BR,AORB,SP10	
(1)		063310	MICPC=MICPC+1	
(1)	014654		<MOVE!SPX!BR!AORB!SP10>	
1602			.	
1603	014656	001334	TEOM2: STATE TMTA	
(1)		000403	MICPC=MICPC+1	
(1)	014656		<MOVE!WRTEBR!IMM!<TMTA-INIT&777/2>>	
1604	014660	001335	ALWAYS XEXIT	
(1)		100451	MICPC=MICPC+1	
(1)	014660		<JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>	
1605	014662	001336	FUDGE: BRWRTE IBUS,NPR	;READ NPR CONTROL
(1)		120600	MICPC=MICPC+1	
(1)	014662		<MOVE!WRTEBR!IBUS!<NPR>>	
1606	014664	001337	BRO IDLE	;IF NPR GOING---LEAVE
(1)		102045	MICPC=MICPC+1	
(1)	014664		<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
1607	014666	001340	BRWRTE BR!LDMAR,SELA!SP4	;LOAD THE MAR
(1)		070604	MICPC=MICPC+1	
(1)	014666		<MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>	


```

(1)
1608 014670          BR7   BS2           ;IF SET - READ BACK ALL 200
(1)          001341   MICPC=MICPC+1
(1) 014670 103505   <JUMP!BR7CON!<BS2-INIT&3000*4>!<BS2-INIT&777/2>>
(1)
1609 014672          MEMINC IBUS,INDAT1      ;OTHERWISE RESTORE TWO BYTES
(1)          001342   MICPC=MICPC+1
(1) 014672 036400   <MOVE!WRMEM!INCMAR!IBUS!<INDAT1>>
(1)
1610 014674          MEMINC IBUS,INDAT2      ;...
(1)          001343   MICPC=MICPC+1
(1) 014674 036420   <MOVE!WRMEM!INCMAR!IBUS!<INDAT2>>
(1)
1611 014676          BRWRTE IMM,2           ;UPDATE---UNIBUS ADDRESS
(1)          001344   MICPC=MICPC+1
(1) 014676 000402   <MOVE!WRTEBR!IMM!<2>>
(1)
1612 014700          SP      BR,ADD,SP4       ;UPDATE NPR COUNTER
(1)          001345   MICPC=MICPC+1
(1) 014700 063004   <MOVE!SPX!BR!ADD!SP4>
(1)
1613 014702          SP      IBUS,IIBA1,SPO    ;UPDATE ADDRESS LOW
(1)          001346   MICPC=MICPC+1
(1) 014702 023100   <MOVE!SPX!IBUS!IIBA1!SPO>
(1)
1614 014704          OUTPUT BR,ADD!IBA1
(1)          001347   MICPC=MICPC+1
(1) 014704 062004   <MOVE!WROUT!BR!<ADD!IBA1>>
(1)
1615 014706          SP      IBUS,IIBA2,SPO    ;READ HIGH ADDRESS
(1)          001350   MICPC=MICPC+1
(1) 014706 023120   <MOVE!SPX!IBUS!IIBA2!SPO>
(1)
1616 014710          OUTPUT BR,AC!IBA2        ;UPDATE HIGH
(1)          001351   MICPC=MICPC+1
(1) 014710 062105   <MOVE!WROUT!BR!<AC!IBA2>>
(1)
1617 014712          C      RESEXT           ;IF CARRY---UPDATE MXT
(1)          001352   MICPC=MICPC+1
(1) 014712 105367   <JUMP!CCOND!<RESEXT-INIT&3000*4>!<RESEXT-INIT&777/2>>
(1)
1618 014714          RES1: SP      IBUS,NPR,SPO  ;READ NPR REGISTER
(1)          001353   MICPC=MICPC+1
(1) 014714 123200   <MOVE!SPX!IBUS!NPR!SPO>
(1)
1619 014716          ALWAYS TH3X           ;GO DO ANOTHER NPR
(1)          001354   MICPC=MICPC+1
(1) 014716 110574   <JUMP!ALCOND!<TH3X-INIT&3000*4>!<TH3X-INIT&777/2>>
(1)
1620 014720          BTEOM: BRWRTE IMM,374     ;MASK FOR CLEAR MSG PENDING
(1)          001355   MICPC=MICPC+1
(1) 014720 000774   <MOVE!WRTEBR!IMM!<374>>
(1)
1621 014722          SP      BR,AANDB,SP10     ;TURN THEM OFF IN LINE STATUS WORD
(1)          001356   MICPC=MICPC+1
(1) 014722 063270   <MOVE!SPX!BR!AANDB!SP10>

```

```

(1)
1622 014724          SP      BR,SELB,SP13          ;STORE UNRECOGNIZABLE VALUE INTO SP13
(1)          001357          MICPC=MICPC+1
(1) 014724 063233          <MOVE!SPX!BR!SELB!SP13>
(1)
1623          ;SO "RH3" WILL EXIT BACK TO IDLE LOOP
1624 014726          LDMA   IMM,STC          ;ADDRESS START OF TMT CHAIN
(1)          001360          MICPC=MICPC+1
(1)          001          .IF IDN IMM IMM
(1) 014726 010070          <MOVE!LDMAR!IMM!<STC&377>>
(1)          .IFF
(1)          <MOVE!LDMAR!IMM!<STC>>
(1)          000          .ENDC
(1)
1625 014730          SP      MEMX,SELB,SPO          ;COPY LINK ADDRESS
(1)          001361          MICPC=MICPC+1
(1) 014730 043220          <MOVE!SPX!MEMX!SELB!SPO>
(1)
1626 014732          TSTATE NUMSYN          ;CHANGE XMIT STATE TO LINE IS IDLE
(1)          001362          MICPC=MICPC+1
(1) 014732 000431          <MOVE!WRTEBR!IMM!<NUMSYN-INIT&777/2>>
(1)          001363          MICPC=MICPC+1
(1) 014734 063222          <MOVE!SPX!BR!SELB!SP2>
1627 014736          ALWAYS TDON2          ;POST A DONE
(1)          001364          MICPC=MICPC+1
(1) 014736 114534          <JUMP!ALCOND!<TDON2-INIT&3000*4>!<TDON2-INIT&777/2>>
(1)
1628 014740          R_L4: RSTATE RCVL
(1)          001365          MICPC=MICPC+1
(1) 014740 000715          <MOVE!WRTEBR!IMM!<RCVL-INIT&777/2>>
(1)          001366          MICPC=MICPC+1
(1) 014742 063223          <MOVE!SPX!BR!SELB!SP3>
1629 014744          SP      IBUS,NPR,SPO          ;READ NPR CONTROL REGISTER
(1)          001367          MICPC=MICPC+1
(1) 014744 123200          <MOVE!SPX!IBUS!NPR!SPO>
(1)
1630          001          .IF NDF LOW
1631          BRWRTE IMM,200          ;MASK FOR CO
1632          OUT      BR,<AORB!ONPR>          ;SET CO
1633          ALWAYS RK2
1634          .ENDC
1635          001          .IF DF LOW
1636 014746          BRWRTE IMM,221
(1)          001370          MICPC=MICPC+1
(1) 014746 000621          <MOVE!WRTEBR!IMM!<221>>
(1)
1637 014750          ALWAYS RK7
(1)          001371          MICPC=MICPC+1
(1) 014750 104643          <JUMP!ALCOND!<RK7-INIT&3000*4>!<RK7-INIT&777/2>>
(1)
1638          000          .ENDC
1639 014752          HOINCH: SP      IBUS,IIBA2,SPO
(1)          001372          MICPC=MICPC+1
(1) 014752 023120          <MOVE!SPX!IBUS!IIBA2!SPO>
(1)
1640 014754          OUTPUT BR,INCA:IIBA2          ;OLTPLT INCREMENTED BA

```

```

(1) 014754 001373 MICPC=MICPC+1
(1) 014754 062065 <MOVE!WROUT!BR!<INCA!IBA2>>
(1) 1641 014756 C SS ;INCREMENT BYTEW COUNT
(1) 014756 001374 MICPC=MICPC+1
(1) 014756 111376 <JUMP!CCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
(1) 1642 014760 ALWAYS TH8
(1) 014760 001375 MICPC=MICPC+1
(1) 014760 110565 <JUMP!ALCOND!<TH8-INIT&3000*4>!<TH8-INIT&777/2>>
(1) 1643 ;INCREMENT MXT BITS
1644 014762 5$: SP IBUS,NPR,SPO ;READ NPR REG IWTH CURRENT MXT BITS
(1) 014762 001376 MICPC=MICPC+1
(1) 014762 123200 <MOVE!SPX!IBUS!NPR!SPO>
(1) 1645 014764 BRWRTE IMM,4 ;WRITE BIT TO ADD
(1) 014764 001377 MICPC=MICPC+1
(1) 014764 000404 <MOVE!WRTEBR!IMM!<4>>
(1) 1646 014766 OUT BR,<ADD!ONPR> ;TURN ON PROPER MXT BITS
(1) 014766 001403 MICPC=MICPC+1
(1) 014766 061010 <MOVE!WROUTX!BR!<ADD!ONPR>>
(1) 1647 014770 ALWAYS TH8
(1) 014770 001401 MICPC=MICPC+1
(1) 014770 110565 <JUMP!ALCOND!<TH8-INIT&3000*4>!<TH8-INIT&777/2>>
(1) 1648 ;
1649 014772 HEH1: STATE TEOM
(1) 014772 001402 MICPC=MICPC+1
(1) 014772 000710 <MOVE!WRTEBR!IMM!<TEOM-INIT&777/2>>
1650 014774 ALWAYS XEXIT
(1) 014774 001403 MICPC=MICPC+1
(1) 014774 100451 <JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
1651

```

1653			.SBTTL REP HANDLER	
1654	014776		LDMA IMM,REPCR ;LOAD MAR ADDRESS WITH POINTER TO REPS RECD	
(1)		001404	MICPC=MICPC+1	
(1)		001	.IF IDN IMM,IMM	
(1)	014776	010016	<MOVE!LDMAR!IMM!<REPCR&377>>	
(1)			.IFF	
(1)			<MOVE!LDMAR!IMM!<REPCR>>	
(1)		000	.ENDC	
1655	015000		SP MEMX,SELB,SPO ;READ NUMBER OF REPS RECD	
(1)		001405	MICPC=MICPC+1	
(1)	015000	043220	<MOVE!SPX!MEMX!SELB!SPO>	
1656	015002		MEM DP,<INCA!SPO> ;INCREMNT REPS RECD	
(1)		001406	MICPC=MICPC+1	
(1)	015002	062460	<MOVE!WRMEM!DP!<INCA!SPO>>	
1657	015004		LDMA IMM,T ;LOAD ADDRESS OF TYPE FIELD	
(1)		001407	MICPC=MICPC+1	
(1)		001	.IF IDN IMM,IMM	
(1)	015004	010153	<MOVE!LDMAR!IMM!<T&377>>	
(1)			.IFF	
(1)			<MOVE!LDMAR!IMM!<T>>	
(1)		000	.ENDC	
1658	015006		MEMINC IMM,2 ;LOAD NAK TYPE	
(1)		001410	MICPC=MICPC+1	
(1)	015006	016402	<MOVE!WRMEM!INCMAR!IMM!<2>>	
1659	015010		MEMINC IMM,303 ;LOAD REP RESPONSE SUB-TYPE	
(1)		001411	MICPC=MICPC+1	
(1)	015010	016703	<MOVE!WRMEM!INCMAR!IMM!<303>>	
1660	015012		ALWAYS SNAK ;SEND AN UNNUMB MSG	
(1)		001412	MICPC=MICPC+1	
(1)	015012	114712	<JUMP!ALCOND!<SNAK-INIT&3000*4>!<SNAK-INIT&777/2>>	
1661			.SBTTL START HANDLER	
1662			BRWRT DP,<SELA!SP10> ;READ LINE STATUS WORD	
1663	015014		MICPC=MICPC+1	
(1)		001413	<MOVE!WRTEBR!DP!<SELA!SP10>>	
(1)	015014	060610		
1664	015016		BRSHFT ;GET START MODE BIT IN TESTABLE POSITION	
(1)		001414	MICPC=MICPC+1	
(1)	015016	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
1665	015020		BR4 10\$;IF IN START MODE SET STACK	
(1)		001415	MICPC=MICPC+1	
(1)	015020	117021	<JUMP!BR4CON!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>	
1666			;ELSE SET UP START ERROR	
1667	015022		LDMA IMM,<<RTHRS+3>>	
(1)		001416	MICPC=MICPC+1	
(1)		001	.IF IDN IMM,IMM	
(1)	015022	010177	<MOVE!LDMAR!IMM!<<RTHRS+3>&377>>	

```

(1)          .IFF
(1)          <MOVE!LDMAR!IMM!<<RTHRS+3>>>
(1)          .ENDC
(1)          000
1668 015024  BRWRT  IMM,200
(1)          001417  MICPC=MICPC+1
(1) 015024 000600  <MOVE!WRTEBR!IMM!<200>>
(1)
1669 015026  ALWAYS RCEXY
(1)          001420  MICPC=MICPC+1
(1) 015026 114525  <JUMP!ALCOND!<RCEXY-INIT&3000*4>!<RCEXY-INIT&777/2>>
(1)
1670 015030  108:  LDMA  IMM,T          ;SET UP ADDRESS OF TYPE FIELD
(1)          001421  MICPC=MICPC+1
(1)          001      .IF IDN IMM,IMM
(1) 015030 010153  <MOVE!LDMAR!IMM!<T&377>>
(1)          .IFF
(1)          <MOVE!LDMAR!IMM!<T>>
(1)          .ENDC
(1)          000
1671 015032  MEMINC IMM,7          ;WRITE STACK TYPE
(1)          001422  MICPC=MICPC+1
(1) 015032 016407  <MOVE!WRMEM!INCMAR!IMM!<7>>
(1)
1672 015034  BRWRT  IMM,11        ;SET START RECD AND UNNUMB PENDING
(1)          001423  MICPC=MICPC+1
(1) 015034 000411  <MOVE!WRTEBR!IMM!<11>>
(1)
1673 015036  ALWAYS SA2          ;SEND THE UNNUMBERED MESSAGE
(1)          001424  MICPC=MICPC+1
(1) 015036 110732  <JUMP!ALCOND!<SA2-INIT&3000*4>!<SA2-INIT&777/2>>
(1)
1674
1675          ;SBTTL STACK HANDLER
1676 015040  STACK: BRWRT  IMM,327      ;MASK TO CLEAR START MODE
(1)          001425  MICPC=MICPC+1
(1) 015040 000727  <MOVE!WRTEBR!IMM!<327>>
(1)
1677 015042  SP      BR,AANDB,SP10    ;CLEAR START MODE
(1)          001426  MICPC=MICPC+1
(1) 015042 063270  <MOVE!SPX!BR!AANDB!SP10>
(1)
1678 015044  ALWAYS TIME1        ;RESET TIMER AND IDLE
(1)          001427  MICPC=MICPC+1
(1) 015044 110662  <JUMP!ALCOND!<TIME1-INIT&3000*4>!<TIME1-INIT&777/2>>
(1)

```

```

1680 015046 001430      ICBA22: SP      IBUS, IOBA2, SPD      :READTHEHIGH ORDERBITS OF BA TO SPD
      (1)      023160      MICPC=MICPC+1
      (1)      015046      <MOVE!SPX!IBUS!IOBA2!SPD>
      (1)
1681 015050 001431      OUTPUT DP <INCA!OBA2>      :OUTPUT THE INCREMENTED COUNT
      (1)      062067      MICPC=MICPC+1
      (1)      015050      <MOVE!WROUT!DP!<INCA!OBA2>>
      (1)
1682 015052 001432      C      SS      :IF CARRY SET INCREMENT THE MXTBITS
      (1)      115034      MICPC=MICPC+1
      (1)      015052      <JUMP!CCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
      (1)
1683 015054 001433      ALWAYS RK9
      (1)      104660      MICPC=MICPC+1
      (1)      015054      <JUMP!ALCOND!<RK9-INIT&3000*4>!<RK9-INIT&777/2>>
      (1)
1684
1685 015056 001434      SS:      SP      IBUS, UBBR, SPD
      (1)      123220      MICPC=MICPC+1
      (1)      015056      <MOVE!SPX!IBUS!UBBR!SPD>
      (1)
1686 015060 001435      BRWRT IMM, 4
      (1)      030404      MICPC=MICPC+1
      (1)      015060      <MOVE!WRTEBR!IMM!<4>>
      (1)
1687 015062 001436      OUT      BR <ADD!OBR>
      (1)      061011      MICPC=MICPC+1
      (1)      015062      <MOVE!WROUTX!BR!<ADD!OBR>>
      (1)
1688 015064 001437      ALWAYS RK9
      (1)      104660      MICPC=MICPC+1
      (1)      015064      <JUMP!ALCOND!<RK9-INIT&3000*4>!<RK9-INIT&777/2>>
      (1)
1689 015066 001440      FLUSH1: SP      IMM, 200, SPD      :FLUSH THE RECVR
      (1)      003200      MICPC=MICPC+1
      (1)      015066      <MOVE!SPX!IMM!200!SPD>
      (1)
1690 015070 001441      OUTPUT BR <SELA!ORCVCC>
      (1)      062212      MICPC=MICPC+1
      (1)      015070      <MOVE!WROUT!BR!<SELA!ORCVCC>>
      (1)
1691 015072 001442      ALWAYS CG1
      (1)      104575      MICPC=MICPC+1
      (1)      015072      <JUMP!ALCOND!<CG1-INIT&3000*4>!<CG1-INIT&777/2>>
      (1)
1692 015074 001443      NAK:      LDMA      IMM, STC      :ADDRESS START OF TMT CHAIN
      (1)      001      MICPC=MICPC+1
      (1)      015074      .IF IDN IMM, IMM
      (1)      010070      <MOVE!LDMAR!IMM!<STC&377>>
      (1)      .IFF
      (1)      <MOVE!LDMAR!IMM!<STC>>
      (1)      .ENDC
      (1)      000
1693 015076 001444      SP      MEMX! INCMAR, SELB, SPD      :COPY IT TO SPD
      (1)      057220      MICPC=MICPC+1
      (1)      015076      <MOVE!SPX!MEMX!INCMAR!SELB!SPD>

```

```

(1) 1694 015100 001445 MEM BR SELA!SP0 ;COPY START OF CHAIN
(1) 015100 062600 MICPC=MICPC+1
(1) <MOVE!WRMEM!BR!<SELA!SP0>>

(1) 1695 015102 001446 BRWRT BR INCA!SP17 ;GETLASTMESSAGE ACKED
(1) 015102 060477 MICPC=MICPC+1
(1) <MOVE!WRTEBR!BR!<INCA!SP17>>

(1) 1696 015104 001447 SP BR SELB,SP12 ;COPY TO CURRENT NUMBER
(1) 015104 063232 MICPC=MICPC+1
(1) <MOVE!SPX!BR!SELB!SP12>

(1) 1697 015106 001450 BRWRT IMM,6 ;WRITE NUMBERED MSG PENDING
(1) 015106 00040E MICPC=MICPC+1
(1) <MOVE!WRTEBR!IMM!<6>>

(1) 1698 015110 SP BR AORB,SP10 ;AND LINE HAS GONE IDLE
(1) 015110 001451 MICPC=MICPC+1 ;SET IT IN LINE STATUS WORD
(1) 015110 063310 <MOVE!SPX!BR!AORB!SP10>

(1) 1700 015112 SP BR SELB,SP15 ;RESET TIMER COUNT
(1) 015112 001452 MICPC=MICPC+1
(1) 015112 063235 <MOVE!SPX!BR!SELB!SP15>

(1) 1701 015114 ALWAYS TEOM1
(1) 015114 001453 MICPC=MICPC+1
(1) 015114 110720 <JUMP!ALCOND!<TEOM1-INIT&3000*4>!<TEOM1-INIT&777/2>>

(1) 1702 015116 ININT: BRWRT IMM,15 ;MASK FOR TURN OFF ALL BUT EXT MEM BITS + NXM
(1) 015116 001454 MICPC=MICPC+1
(1) 015116 000415 <MOVE!WRTEBR!IMM!<15>>

(1) 1703 015120 SP IBUS,UBBR,SP0 ;READ BR CONTROL REGISTER
(1) 015120 001455 MICPC=MICPC+1
(1) 015120 123220 <MOVE!SPX!IBUS!JBBR!SP0>

(1) 1704 015122 SP BR AANDB,SP0 ;MASK OFF VECTOR TO XCH
(1) 015122 001456 MICPC=MICPC+1
(1) 015122 0632E0 <MOVE!SPX!BR!AANDB!SP0>

(1) 1705 015124 BRWRT IMM,200 ;MASK FOR INTERRUPT
(1) 015124 001457 MICPC=MICPC+1
(1) 015124 000E00 <MOVE!WRTEBR!IMM!<200>>

(1) 1706 015126 OUT BR AORB!OBR ;INTERRUPT
(1) 015126 001460 MICPC=MICPC+1
(1) 015126 0E1311 <MOVE!WRROUTX!BR!<AORB!OBR>>

(1) 1707 015130 SP IBUS,INCON,SP0 ;RESTORE INPUT CONTROL CSP
(1) 015130 001461 MICPC=MICPC+1
(1) 015130 123000 <MOVE!SPX!IBUS!INCON!SP0>

(1) 1708 015132 ALWAYS NIDLE4
(1) 015132 001462 MICPC=MICPC+1
    
```

G13

EMUL: DDCNF PROTOCOL IMPLEMENTATION
DDNEW.MAC 26-DEC-76 10:14

MACY11 27,1006) 07-MAR-77 16:24 PAGE 12-22
STACK HANDLER

PAGE: 0162

015:32 100563
1705

JUMP!ALCOND!<NIDLE4-INIT33000*4>!<NIDLE4-INIT3777/2>

:

H13

MAC: CDCMS PROTOCOL IMPLEMENTATION
CDCNEW.MAC 06-DEC-76 10:14

MACY11 27 (1006) 07-MAR-77 16:24 PAGE 12-23
NXMERR ---NON EXISTANT MEMORY HANDLER

PAGE: 0183

1711			.SBTTL NXMERR ---NON EXISTANT MEMORY HANDLER
1712	015134	001463	NXMERR: LDMA IMM,<<RTHRS+3>> ;ADDRESS ERROR LINK
(1)		001464	MICPC=MICPC+1
(1)		001465	.IF IDN IMM,IMM
(1)	015134	010177	<MOVE!LDMAR!IMM!<<RTHRS+3 3377>>
(1)			.IFF
(1)		000	<MOVE!LDMAR!IMM!<<RTHRS+3>>
(1)			.ENDC
1713	015136		MEMINC IMM,1
(1)		001464	MICPC=MICPC+1
(1)	015136	016401	<MOVE!WRMEM!INCMAR!IMM! 1>>
1714	015140		MEM IMM,0 ;NXM ERROR BIT
(1)		001465	MICPC=MICPC+1
(1)	015140	002400	<MOVE!WRMEM!IMM!<0>>
1715	015142		SP MEMX,SELB,SP10 ;CLEAR STATUS
(1)		001466	MICPC=MICPC+1
(1)	015142	043230	<MOVE!SPX!MEMX!SELB!SP10>
1716	015144		ALWAYS RCEXX
(1)		001467	MICPC=MICPC+1
(1)	015144	114827	<JUMP!ALCOND!<RCEXX-INIT\$3000+4>! RCEXX-INIT\$777 2

↑
↓

```

1718 .SBTTL SELQSY--ROUTINETOCHECK SELECT AND QSYNC AND DIDDLE LINE STATUS WORD
1719 :USES SP5. ALWAYS CALLED BY FIRST INSTR IN A RSTATE
1720 015146 SPBR IBUS,RCVDAT,SP5 :READCHARACTERINTO SP5 AND THE 2P
      001470 MICPC=MICPC+1
      023605 <MOVE!SPBRX!IBUS!RCVDAT!SP5>
1721 015150 BR7 155 :SELECT SET?--BRANCH
      001471 MICPC=MICPC+1
      117477 <JUMP!BR7CON!<155-INIT&3000*4>!<155-INIT&777/2>>
1722 015152 5$: BRWRTE BR,AA!SP5 :SHIFT BR LEFT
      001472 MICPC=MICPC+1
      060525 <MOVE!WRTEBR!BR!<AA!SP5>>
1723 015154 BR7 205 :FINAL SET?
      001473 MICPC=MICPC+1
      117502 <JUMP!BR7CON!<205-INIT&3000*4>!<205-INIT&777/2>>
1724 015156 10$: BRWRTE IMM,77 :MASK TO BR
      001474 MICPC=MICPC+1
      003477 <MOVE!WRTEBR!IMM!<77>>
1725 015160 SPBR BR,AANDB,SP5 :TURN OFF SELECTANDFINAL
      001475 MICPC=MICPC+1
      063665 <MOVE!SPBRX!BR!AANDB!SP5>
1726 015162 .ALWAY BR,INCA,SP3!PAGE1
      001476 MICPC=MICPC+1
      164463 <JUMP!ALCOND!BR!INCA!SP3!PAGE1>
1727 015164 15$: BRWRTE IMM,200 :SET OK TO SEND
      001477 MICPC=MICPC+1
      000600 <MOVE!WRTEBR!IMM!<200>>
1728 015166 SP BR,AORB,SP10 :IN LINE STATUS WORD
      001500 MICPC=MICPC+1
      063310 <MOVE!SPX!BR!AORB!SP10>
1729 015170 ALWAYS 55
      001501 MICPC=MICPC+1
      114472 <JUMP!ALCOND!<55-INIT&3000*4>!<55-INIT&777/2>>
1730 015172 20$: BRWRTE IMM,20 :SETCLEARACTIVE
      001502 MICPC=MICPC+1
      000420 <MOVE!WRTEBR!IMM!<20>>
1731 015174 SP BR,AORB,SP10 :IN LINE STATUS WORD
      001503 MICPC=MICPC+1
      063310 <MOVE!SPX!BR!AORB!SP10>
1732 015176 ALWAYS 105
      001504 MICPC=MICPC+1
      114474 <JUMP!ALCOND!<105-INIT&3000*4>!<105-INIT&777/2>>

```

```

1735
1736 015200 001505
(1) 015200 020540
(1)
(1)
1737 015202 001506
(1) 015202 116111
(1)
(1)
1738 001
1739
1740
1741 000
1742 015204 001507
(1) 015204 000674
(1)
1743 015206 001510
(1) 015206 104422
(1)
(1)
1744
1745 015210 001511
(1) 015210 000626
(1)
1746 015212 001512
(1) 015212 104422
(1)
(1)
1747
1748 015214 001513
(1) 015214 040364
(1)
(1)
1749 015216 001514
(1) 015216 11511E
(1)
(1)
1750 015220 001515
(1) 015220 104475
(1)
(1)
1751 015222 001516
(1) 001
(1) 015222 010153
(1)
(1)
(1) 000
(1)
1752 015224 001517
(1) 015224 016402
(1)
(1)
1753 015226 001520
(1) 015226 002711
(1)

```

```

;FUGITIVE RECEIVE ROUTINES---DON'T FIT IN PAGE
R41: BRWRT IBUS,IOBA1 ;READ LOW BYTE OF IN 2A
MICPC=MICPC+1
<MOVE!WRTEBR!IBUS!<IOBA1>>

BR0 RCVODD ;IF SET IS ODD TRANSFER
MICPC=MICPC+1
<JUMP!BROCON!<RCVODD-INIT&3000*4>!<RCVODD-INIT&777/2>>

;IF NDF LOW
BRWRT IBUS,RCVCON ;IS THE RECEIVER READY?
BR4 RCVKED ;YES--GO PROCESS
.ENDC
STATE RCVKED
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVKED-INIT&777/2>>
ALWAYS REXIT
MICPC=MICPC+1
<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>

RCVODD: STATE RCVK01
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVK01-INIT&777/2>>
ALWAYS REXIT
MICPC=MICPC+1
<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>

RCLOW: CMP MEMX,SP4 ;COMPARE LOW ORDER BITS OF COUNT
MICPC=MICPC+1
<SUBTC!MEMX!SP4>

C RCFATL ;CARRY--TOO BIG
MICPC=MICPC+1
<JUMP!CCOND!<RCFATL-INIT&3000*4>!<RCFATL-INIT&777/2>>

ALWAYS RCS ;ELSE CONTINUE
MICPC=MICPC+1
<JUMP!ALCOND!<RCS-INIT&3000*4>!<RCS-INIT&777/2>>

RCFATL: LDMA IMM,T
MICPC=MICPC+1
;IF IDN IMM,IMM
<MOVE!LDMAR!IMM!<T&377>>
;IFF
<MOVE!LDMAR!IMM!<T>>
.ENDC

MEMINC IMM,2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<2>>

MEM IMM,311
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<311>>

```

```

(1) 1754 015230          LDMA   IMM,<<RTHRS+1>>          ;ADDRESS ERROR LINK
(1)          001521          MICPC=MICPC+1
(1)          001          .IF IDN IMM IMM
(1) 015230 010175          <MOVE!LDMAR!IMM!<<RTHRS+1>>377>>
(1)          .IFF
(1)          <MOVE!LDMAR!IMM!<<RTHRS+1>>>
(1)          .ENDC
(1)          000
(1) 1755 015232          MEMINC IBUS,IOBA1
(1)          001522          MICPC=MICPC+1
(1) 015232 03654C          <MOVE!WRMEM!INCMAR!IBUS!<IOBA1>>
(1)
(1) 1756 015234          MEMINC IBUS,IOBA2
(1)          001523          MICPC=MICPC+1
(1) 015234 03656C          <MOVE!WRMEM!INCMAR!IBUS!<IOBA2>>
(1)
(1) 1757 015236          BRWRT  IMM,20
(1)          001524          MICPC=MICPC+1
(1) 015236 00042C          <MOVE!WRTEBR!IMM!<20>>
(1)
(1) 1758 015240          RCEXY: MEMINC IMM,0
(1)          001525          MICPC=MICPC+1
(1) 015240 016400          <MOVE!WRMEM!INCMAR!IMM!<0>>
(1)
(1) 1759 015242          MEM     BR,SELB
(1)          001526          MICPC=MICPC+1
(1) 015242 06262C          <MOVE!WRMEM!BR!<SELB>>
(1)
(1) 1760 015244          RCEXX: OUTPUT IMM,<200!ORCVCO>          ;FLUSH THE INPUT SILO
(1)          001527          MICPC=MICPC+1
(1) 015244 002212          <MOVE!WROUT!IMM!<200!ORCVCO>>
(1)
(1) 1761 015246          SP     IMM,1,SP1          ;SET INIT MODE IN PCRT STATUS WORD
(1)          001530          MICPC=MICPC+1
(1) 015246 003001          <MOVE!SPX!IMM!1!SP1>
(1)
(1) 1762 015250          ALWAYS NTRS1
(1)          001531          MICPC=MICPC+1
(1) 015250 114674          <JUMP!ALCOND!<NTRS1-INIT&3000*4>!<NTRS1-INIT&777/2>>
(1)
(1) 1763
(1) 1764 015252          TCON3: BRWRT  MEMX,SUB!SP17          ;COMPARE RESPONSE TO MSG NO
(1)          001532          MICPC=MICPC+1
(1) 015252 040757          <MOVE!WRTEBR!MEMX!<SUB!SP17>>
(1)
(1) 1765 015254          BR7     RH3          ;IF NEGATIVE EXIT
(1)          001533          MICPC=MICPC+1
(1) 015254 107567          <JUMP!BR7CON!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>
(1)
(1) 1766 015256          TCON2: LDMA   BR,SELA!SPO          ;ADDRESS THE TRANSMITLINK
(1)          001534          MICPC=MICPC+1
(1)          001          .IF IDN BR IMM
(1)          <MOVE!LDMAR!IMM!<SELA!SPO&377>>
(1)          .IFF
(1) 015256 070200          <MOVE!LDMAR!BR!<SELA!SPO>>

```

```

(1)          000          .ENDC
1767 015260          MEM      IMM,0          ;TURN OF ASSIGNEDAND TMTED BITS IN FLAG
(1)          001535      MICPC=MICPC+1
(1) 015260 002400      <MOVE!WRMEM!IMM!<0>>
(1)
1769 015262          LDMA     IMM,STC
(1)          001536      MICPC=MICPC+1
(1)          001          .IF IDN IMM,IMM
(1) 015262 010070      <MOVE!LDMAR!IMM!<STC&377>>
(1)          .IFF
(1)          <MOVE!LDMAR!IMM!<STC>>
(1)          000          .ENDC
(1)
1769 015264          MEM      IMM,TML1          ;ASSUME WRAPAROUND
(1)          001537      MICPC=MICPC+1
(1) 015264 002473      <MOVE!WRMEM!IMM!<TML1>>
(1)
1770 015266          BRWRTE  IMM,TML8          ;WRAPAROUND?
(1)          001540      MICPC=MICPC+1
(1) 015266 000545      <MOVE!WRTEBR!IMM!<TML8>>
(1)
1771 015270          CMP      BR,SPO
(1)          001541      MICPC=MICPC+1
(1) 015270 060360      <SUBTC!BR!SPO>
(1)
1772 015272          Z        TDON4          ;YES
(1)          001542      MICPC=MICPC+1
(1) 015272 115545      <JUMP!ZCOND!<TDON4-INIT&3000*4>!<TDON4-INIT&777/2>>
(1)
1773 015274          BRWRTE  IMM,6          ;OFFSET FOR NEXT TMT LINK
(1)          001543      MICPC=MICPC+1
(1) 015274 000406      <MOVE!WRTEBR!IMM!<6>>
(1)
1774 015276          MEM      BR,ADD!SPO          ;UPDATE THE POINTER
(1)          001544      MICPC=MICPC+1
(1) 015276 062400      <MOVE!WRMEM!BR!<ADD!SPO>>
(1)
1775 015300          TDON4: LDMA     IMM,NXTSP          ;ADDRESS DONE LINK
(1)          001545      MICPC=MICPC+1
(1)          001          .IF IDN IMM,IMM
(1) 015300 010241      <MOVE!LDMAR!IMM!<NXTSP&377>>
(1)          .IFF
(1)          <MOVE!LDMAR!IMM!<NXTSP>>
(1)          000          .ENDC
(1)
1776 015302          LDMA     MEMX,SELB!SPX!SP3          ;ADDRESS THE LINK,COPYING
(1)          001546      MICPC=MICPC+1
(1)          001          .IF IDN MEMX,IMM
(1)          <MOVE!LDMAR!IMM!<SELB!SPX!SP3&377>>
(1)          .IFF
(1) 015302 053223      <MOVE!LDMAR!MEMX!<SELB!SPX!SP3>>
(1)          000          .ENDC
(1)
1777          MEMINC  IMM,200          ;ITS ADDRESS TO SPO
1778 015304          ;WRITE THE INTERRUPT TYPE

```

M13

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 06-DEC-76 10:14

MACY11 27(1006) 07-MAR-77 16:24 PAGE 12-28
SELQSY--ROUTINETOCHECK SELECT AND QSYNC AND DIDDLE LINE STATUS WORD

PAGE: 016E

```

(1)          001547          MICPC=MICPC+1
(1) 015304 016600          <MOVE!WRMEM!INCMAR!IMM!<200>>
(1)
1779 015306          MEM      BR,INCA!SP0          ;COPY ACTUAL LINK ADDRESS
(1)          001550          MICPC=MICPC+1
(1) 015306 062460          <MOVE!WRMEM!BR!<INCA!SP0>>
(1)
1780 015310          LDMA     IMM,NXTSP          ;ADDRESS PTR INT STACK
(1)          001551          MICPC=MICPC+1
(1)          001          .IF IDN IMM,IMM
(1) 015310 010241          <MOVE!LDMAR!IMM!<NXTSP&377>>
(1)          001          .IFF
(1)          000          <MOVE!LDMAR!IMM!<NXTSP>>
(1)          000          .ENDC
(1)          000
1781 015312          MEM      IMM,INTSTK          ;ASSUME WRAP AROUND
(1)          001552          MICPC=MICPC+1
(1) 015312 002642          <MOVE!WRMEM!IMM!<INTSTK>>
(1)
1782 015314          BRWRTE  IMM,<<MMEND-2>>          ;ADDRESS ENDOFINT STACK
(1)          001553          MICPC=MICPC+1
(1) 015314 000776          <MOVE!WRTEBR!IMM!<<MMEND-2>>>
(1)
1783 015316          CMP      BR,SP3          ;WRAPAROUND?
(1)          001554          MICPC=MICPC+1
(1) 015316 060363          <SUBTC!BR!SP3>
(1)
1784 015320          Z        TDON40          ;YES---BRANCH
(1)          001555          MICPC=MICPC+1
(1) 015320 1155E0          <JUMP!ZCOND!<TDON40-INIT&3000*4>!<TDON40-INIT&777/2>>
(1)
1785 015322          BRWRTE  IMM,2          ;OFFSET TO NEXT PAIR
(1)          001556          MICPC=MICPC+1
(1) 015322 000402          <MOVE!WRTEBR!IMM!<2>>
(1)
1786 015324          MEM      BR,ADD!SP3          ;UPDATE POINTER
(1)          001557          MICPC=MICPC+1
(1) 015324 062403          <MOVE!WRMEM!BR!<ADD!SP3>>
(1)
1787 015326          TDON40: BRWRTE  IMM,20          ;WRITE INTERUPT PENDING
(1)          001560          MICPC=MICPC+1
(1) 015326 000420          <MOVE!WRTEBR!IMM!<20>>
(1)
1788 015330          SP      BR,AORB,SP1          ;IN PORT STATUS WORD
(1)          001561          MICPC=MICPC+1
(1) 015330 063301          <MOVE!SPX!BR!AORB!SP1>
(1)
1789 015332          LDMA     IMM,ETC          ;ADDRESS NEXT EMPTY PTR
(1)          001562          MICPC=MICPC+1
(1)          001          .IF IDN IMM,IMM
(1) 015332 010072          <MOVE!LDMAR!IMM!<ETC&377>>
(1)          001          .IFF
(1)          000          <MOVE!LDMAR!IMM!<ETC>>
(1)          000          .ENDC
(1)
1790 015334          SP      MEMX,SELB,SP0          ;COPY IT TO SP0

```

```

(1) 015334 001563 MICPC=MICPC+1
(1) 015334 043220 <MOVE!SPX!MEMX!SELB!SPD>
(1)
1791 015336 LDMA IMM,STC ;GET NEXT DONE PTR
(1) 001564 MICPC=MICPC+1
(1) 001 .IF IDN IMM,IMM
(1) 015336 010070 <MOVE!LDMAR!IMM!<STC&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<STC>>
(1) 000 .ENDC
(1)
1792 015340 CMP MEMX,SPD ;IDENTICAL?
(1) 001565 MICPC=MICPC+1
(1) 015340 040360 <SUBTC!MEMX!SPD>
(1)
1793 015342 Z RH3 ;FINISH PROCESSING HEADER
(1) 001566 MICPC=MICPC+1
(1) 015342 105567 <JUMP!ZCOND!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>
(1)
1794
1795 015344 TDON1: LDMA IMM,ISP17 ;GET LAST ACKED
(1) 001567 MICPC=MICPC+1
(1) 001 .IF IDN IMM,IMM
(1) 015344 010155 <MOVE!LDMAR!IMM!<ISP17&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<ISP17>>
(1) 000 .ENDC
(1)
1796 015346 SP MEMX,SELB,SP17 ;STORE IT IN SP17
(1) 001570 MICPC=MICPC+1
(1) 015346 043237 <MOVE!SPX!MEMX!SELB!SP17>
(1)
1797 015350 LDMA IMM,STC ;GET START OF TMT CHAIN
(1) 001571 MICPC=MICPC+1
(1) 001 .IF IDN IMM,IMM
(1) 015350 010070 <MOVE!LDMAR!IMM!<STC&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<STC>>
(1) 000 .ENDC
(1)
1798 015352 LDMA MEMX,SELB!SPBRX!SPD ;ADDRESS THE LINK
(1) 001572 MICPC=MICPC+1
(1) 001 .IF IDN MEMX,IMM
(1) <MOVE!LDMAR!IMM!<SELB!SPBRX!SPD&377>>
(1) .IFF
(1) 015352 053620 <MOVE!LDMAR!MEMX!<SELB!SPBRX!SPD>>
(1) 000 .ENDC
(1)
1799 015354 BRWRT MEMX!INCMAR,SELB ;GET THE FLAGS
(1) 001573 MICPC=MICPC+1
(1) 015354 054620 <MOVE!WRTEBR!MEMX!INCMAR!<SELB>>
(1)
1800 015356 BR1 TDON3 ;IFBUFFER ASSIGNED PROCEED
(1) 001574 MICPC=MICPC+1
(1) 015356 116532 <JUMP!BR1CON!<TDON3-INIT&3000*4>!<TDON3-INIT&777/2>>
(1)

```

```

1801 015360          ALWAYS RH3                ;ELSE---EXIT
(1)                MICPC=MICPC+1
(1) 015360 001575  <JUMP!ALCOND!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>
(1)
1802
1803 015362          OVRUN: BRWRT IMM,4
(1)                MICPC=MICPC+1
(1) 015362 001576  <MOVE!WRTEBR!IMM!<4>>
(1)                000404
1804 015364          ALWAYS NTRSO
(1)                MICPC=MICPC+1
(1) 015364 001577  <JUMP!ALCOND!<NTRSO-INIT&3000*4>!<NTRSO-INIT&777/2>>
(1)                114E71
1805
1806                : INPUTS:
1807                :
1808                SP0 = RECEIVE CHARACTER
1809 015366          PASWRD: SP      IBUS, LNOSW, SP16      :READ PASSWD SWITCH
(1)                MICPC=MICPC+1
(1) 015366 001600  <MOVE!SPX!IBUS!LNOSW!SP16>
(1)                02333E
1810 015370          Z      10$                ;IF ALL ONES NO RLD ENABLED
(1)                MICPC=MICPC+1
(1) 015370 001601  <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
(1)                115605
1811 015372          BRWRT IMM,6                ;CHECK FOR ENTER MOP MODE
(1)                MICPC=MICPC+1
(1) 015372 001602  <MOVE!WRTEBR!IMM!<6>>
(1)                000406
1812 015374          CMP      BR, SP0
(1)                MICPC=MICPC+1
(1) 015374 001603  <SUBTC!BR!SP0>
(1)                060360
1813 015376          Z      20$                ;IF EQUAL ENTER MOP
(1)                MICPC=MICPC+1
(1) 015376 001604  <JUMP!ZCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
(1)                115612
1814 015400          10$: BRWRT BR, SELA!SP1      :READ STATUS BYTE
(1)                MICPC=MICPC+1
(1) 015400 001605  <MOVE!WRTEBR!BR!<SELA!SP1>>
(1)                060601
1815 015402          BRSHFT                    ;SHIFT IT RIGHT
(1)                MICPC=MICPC+1
(1) 015402 001606  <MOVE!SHFTBR!WRTEBR!SELB>
(1)                001620
1816 015404          BR1      RHX                ;MESSAGE WITH NC BUFFER ASSIGNED
(1)                MICPC=MICPC+1
(1) 015404 001607  <JUMP!BR1CON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
(1)                106751
1817 015406          BRSHFT                    ;SHIFT RIGHT AGAIN
(1)                MICPC=MICPC+1
(1) 015406 001610  <MOVE!SHFTBR!WRTEBR!SELB>
(1)                001620
1818 015410          BR1      RCVMO                ;DLE RECEIVED IN NORMAL MODE
(1)                MICPC=MICPC+1
(1) 015410 001611  <JUMP!BR1CON!<RCVMO-INIT&3000*4>!<RCVMO-INIT&777/2>>
(1)                106743

```



```

(1) 1819 015412          ALWAYS RK3                ;HANDLE MAINT MODE MESSAGE
(1)          001612    MICPC=MICPC+1
(1) 015412 104661      <JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>
(1)
(1) 1820 015414          208: SP      BR,DECA,SP4                ;COUNT FOR NUMB OF COMPARES
(1)          001613    MICPC=MICPC+1
(1) 015414 063164      <MOVE!SPX!BR!DECA!SP4>
(1)
(1) 1821 015416          STATE EM2
(1)          001614    MICPC=MICPC+1
(1) 015416 000746      <MOVE!WRTEBR!IMM!<EM2-INIT&777/2>>
1822 015420          ALWAYS REXIT
(1)          001615    MICPC=MICPC+1
(1) 015420 104422      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
(1)
1823 015422          RCVM1: LDMA  IMM,ISP11                ;ADDRESS SP11 IMAGE
(1)          001616    MICPC=MICPC+1
(1)          001      .IF IDN IMM,IMM
(1) 015422 010171      <MOVE!LDMAR!IMM!<ISP11&377>>
(1)          .IFF
(1)          <MOVE!LDMAR!IMM!<ISP11>>
(1)          .ENDC
(1)          000
(1)
1824 015424          MEM      BR,DECA!SP11                ;COPY SP11
(1)          001617    MICPC=MICPC+1
(1) 015424 062571      <MOVE!WRMEM!BR!<DECA!SP11>>
(1)
1825 015426          LDMA  IMM,NXTSP
(1)          001620    MICPC=MICPC+1
(1)          001      .IF IDN IMM,IMM
(1) 015426 010241      <MOVE!LDMAR!IMM!<NXTSP&377>>
(1)          .IFF
(1)          <MOVE!LDMAR!IMM!<NXTSP>>
(1)          .ENDC
(1)          000
(1)
1826 015430          SP      MEMX!LDMAR,SELB,SP3                ;COPY TO SP3
(1)          001621    MICPC=MICPC+1
(1) 015430 053223      <MOVE!SPX!MEMX!LDMAR!SELB!SP3>
(1)
1827 015432          MEMINC IMM,204                ;RECEIVE DONE IMAGE
(1)          001622    MICPC=MICPC+1
(1) 015432 016604      <MOVE!WRMEM!INCMAR!IMM!<204>>
(1)
1828 015434          MEM      BR!LDMAR,SELA!SP14                ;COPY LINK ADDRRESS TO NEXT INTE
(1)          001623    MICPC=MICPC+1
(1) 015434 072614      <MOVE!WRMEM!BR!LDMAR!<SELA!SP14>>
(1)
1829 015436          MEMINC IMM,0                ;ZERO THE FLAGS
(1)          001624    MICPC=MICPC+1
(1) 015436 016400      <MOVE!WRMEM!INCMAR!IMM!<0>>
(1)
1830 015440          SP      IBUS,IOBA1,SP4                ;GET BEGIN ADDRESS LOW
(1)          001625    MICPC=MICPC+1
(1) 015440 023144      <MOVE!SPX!IBUS!IOBA1!SP4>
(1)

```

1831	015442		SP	IBUS, IOBA2, SP5	;AND HIGH BYTE
(1)		001626	MICPC=MICPC+1		
(1)	015442	0231E5	<MOVE!SPX!IBUS!IOBA2!SP5>		
(1)					
1832	015444		SP	MEMX! INCMAR, SUB, SP4	;SUBTRACT TO GET COUNT
(1)		001627	MICPC=MICPC+1		
(1)	015444	057344	<MOVE!SPX!MEMX!INCMAR!SUB!SP4>		
(1)					
1833	015446		C	105	;IF CARRY SET THEN NO CARRY!
(1)		001630	MICPC=MICPC+1		
(1)	015446	11523E	<JUMP!CCOND!<105-INIT&3000*4>!<105-INIT&777/2>>		
(1)					
1834	015450		SP	BR, DECA, SP5	;DECREMENT HIGH BYTE OF ADDRESS
(1)		001631	MICPC=MICPC+1		
(1)	015450	0631E5	<MOVE!SPX!BR!DECA!SP5>		
(1)					
1835	015452		105: SP	MEMX! INCMAR, SUB, SP5	;SUBTRACT FOR COUNT
(1)		001632	MICPC=MICPC+1		
(1)	015452	057345	<MOVE!SPX!MEMX!INCMAR!SUB!SP5>		
(1)					
1836	015454		MEMINC	BR, SELA! SP5	;COPY TO MEMORY LINK
(1)		001633	MICPC=MICPC+1		
(1)	015454	076605	<MOVE!WRMEM!INCMAR!BR!<SELA!SP5>>		
(1)					
1837	015456		MEMINC	BR, SELA! SP4	
(1)		001634	MICPC=MICPC+1		
(1)	015456	076604	<MOVE!WRMEM!INCMAR!BR!<SELA!SP4>>		
(1)					
1838	015460		BRWRT	IMM, 2	
(1)		001635	MICPC=MICPC+1		
(1)	015460	000402	<MOVE!WRTEBR!IMM!<2>>		
(1)					
1839	015462		LDMA	IMM, NXTSP	;ADDRESS NEXT INT STACK
(1)		001636	MICPC=MICPC+1		
(1)		001	.IF IDN IMM, IMM		
(1)	015462	010241	<MOVE!LDMAR!IMM!<NXTSP&377>>		
(1)			.IFF		
(1)			<MOVE!LDMAR!IMM!<NXTSP>>		
(1)		000	.ENDC		
(1)					
1840	015464		MEM	BR, ADD! SP3	
(1)		001637	MICPC=MICPC+1		
(1)	015464	062403	<MOVE!WRMEM!BR!<ADD!SP3>>		
(1)					
1841	015466		BRWRT	IMM, <<MMEND-2>>	;ADDRESS END OF INT STACK
(1)		001640	MICPC=MICPC+1		
(1)	015466	00077E	<MOVE!WRTEBR!IMM!<<MMEND-2>>>		
(1)					
1842	015470		OMP	BR, SP3	;WRAP AROUND
(1)		001641	MICPC=MICPC+1		
(1)	015470	0603E3	<SUBTC!BR!SP3>		
(1)					
1843	015472		Z	RMIFLP	;IFYES-- BRANCH
(1)		001642	MICPC=MICPC+1		
(1)	015472	115E44	<JUMP!ZCOND!<RMIFLP-INIT&3000*4>!<RMIFLP-INIT&777/2>>		
(1)					

```

1844 015474          ALWAYS RMX1
      (1)             MICPC=MICPC+1
      (1) 015474     001643
      (1)             114645
      (1)             <JUMP!ALCOND!<RMX1-INIT&3000*4!<RMX1-INIT&777/2>>
1845 015476          RMIFLIP: MEM IMM,INTSTK
      (1)             MICPC=MICPC+1
      (1) 015476     001644
      (1)             002642
      (1)             <MOVE!WRMEM!IMM!<INTSTK>>
1846 015500          RMX1:
1847 015500          LDMA IMM,LRC
      (1)             MICPC=MICPC+1
      (1)             .IF IDN IMM,IMM
      (1) 015500     010024
      (1)             <MOVE!LDMAR!IMM!<LRC&377>>
      (1)             .IFF
      (1)             <MOVE!LDMAR!IMM!<LRC>>
      (1)             .ENDC
      (1)             JOC
1848 015502          BRWRT IMM,5
      (1)             MICPC=MICPC+1
      (1) 015502     001646
      (1)             000405
      (1)             <MOVE!WRTEBR!IMM!<5>>
      (1)             ;INDEX TO NEXT BUFFER
1849 015504          SP BR,ADD,SP14
      (1)             MICPC=MICPC+1
      (1) 015504     001647
      (1)             063014
      (1)             <MOVE!SPX!BR!ADD!SP14>
      (1)             ;JDATE COPY OF POINTER
1850 015506          BRWRT IMM,STC
      (1)             MICPC=MICPC+1
      (1) 015506     001650
      (1)             000470
      (1)             <MOVE!WRTEBR!IMM!<STC>>
      (1)             ;ADDRESS OF WRAP AROUND POINT
1851 015510          CMP BR,SP14
      (1)             MICPC=MICPC+1
      (1) 015510     001651
      (1)             060374
      (1)             <SUBTC!BR!SP14>
      (1)             ;WRAPAROUND?
1852 015512          Z RMFLIP
      (1)             MICPC=MICPC+1
      (1) 015512     001652
      (1)             115655
      (1)             <JUMP!ZCOND!<RMFLIP-INIT&3000*4!<RMFLIP-INIT&777/2>>
      (1)             ;IF YES---BRANCH
1853 015514          MEM BR,SELA!SP14
      (1)             MICPC=MICPC+1
      (1) 015514     001653
      (1)             062614
      (1)             <MOVE!WRMEM!BR!<SELA!SP14>>
      (1)             ;ELSE UPDATE THE POINTER
1854 015516          ALWAYS RMX
      (1)             MICPC=MICPC+1
      (1) 015516     001654
      (1)             114656
      (1)             <JUMP!ALCOND!<RMX-INIT&3000*4!<RMX-INIT&777/2>>
1855 015520          RMFLIP: MEM IMM,RCL1
      (1)             MICPC=MICPC+1
      (1) 015520     001655
      (1)             002425
      (1)             <MOVE!WRMEM!IMM!<RCL1>>
      (1)             ;POINT TO FIRST LINK
1856 015522          RMX: BRWRT IMM,20
      (1)             MICPC=MICPC+1
      (1) 015522     001656
      (1)             000420
      (1)             <MOVE!WRTEBR!IMM!<20>>
      (1)             ;MASK FOR INTERUPT PENDING
1857 015524          SP DP,AORB,SP1
      (1)             MICPC=MICPC+1
      (1) 015524     001657
      (1)             063301
      (1)             <MOVE!SPX!DP!AORB!SP1>
      (1)             ;UPDATE PORT STATUS WORD

```

```

(1) 1858 015526 001660 BRWRTE DP,<SELA!SP10> :READ LINE STATUS WORD
(1) 015526 063810 MICPC=MICPC+1
(1) <MOVE!WRTEBR!DP!<SELA!SP10>
(1)
(1) 1859 015530 001661 BR4 FLUSH :IF CLEAR ACTIVE SET---FLUSH
(1) 015530 107015 MICPC=MICPC+1
(1) <JUMP!BR4CON!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
(1)
(1) 1860 015532 001662 RM1: STATE RCVA
(1) 015532 000400 MICPC=MICPC+1
(1) <MOVE!WRTEBR!IMM!<RCVA-INIT&777/2>>
(1) 1861 015534 001663 ALWAYS REXIT
(1) 015534 104422 MICPC=MICPC+1
(1) <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
(1)
(1) 1862 015536 001664 NTHRS: LDMA IMM,ST
(1) 015536 010154 MICPC=MICPC+1
(1) .IF IDN IMM,IMM
(1) <MOVE!LDMAR!IMM!<ST&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<ST>>
(1) .ENDC
(1) 000
(1)
(1) 1863 015540 001665 SPBR MEMX,SELB,SPO
(1) 015540 043820 MICPC=MICPC+1
(1) <MOVE!SPBRX!MEMX!SELB!SPO>
(1)
(1) 1864 015542 001666 BRWRTE BR,ADD!SPO :SHIFT LEFT
(1) 015542 060400 MICPC=MICPC+1
(1) <MOVE!WRTEBR!BR!<ADD!SPO>>
(1)
(1) 1865 015544 001667 BR4 OVRUN
(1) 015544 117176 MICPC=MICPC+1
(1) <JUMP!BR4CON!<OVRUN-INIT&3000*4>!<OVRUN-INIT&777/2>>
(1)
(1) 1866 015546 001670 BRWRTE IMM,1
(1) 015546 000401 MICPC=MICPC+1
(1) <MOVE!WRTEBR!IMM!<1>>
(1)
(1) 1867 015550 001671 ERRXX: LDMA IMM,<<RTHRS+3>>
(1) 1868 015550 001 NTRSC: MICPC=MICPC+1
(1) 015550 010177 .IF IDN IMM,IMM
(1) <MOVE!LDMAR!IMM!<<RTHRS+3>&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<<RTHRS+3>>>
(1) .ENDC
(1) 000
(1)
(1) 1869 015552 001672 MEMINC IMM,0
(1) 015552 016400 MICPC=MICPC+1
(1) <MOVE!WRMEM!INCMAR!IMM!<0>>
(1)
(1) 1870 015554 001673 MEM BR,SELB
(1) 015554 062620 MICPC=MICPC+1
(1) <MOVE!WRMEM!BR!<SELB>>

```

```

1871 015556 001674
(1) 001
(1) 015556 010241
(1)
(1) 000
1872 015560
(1) 001675
(1) 001
(1)
(1) 015560 053220
(1) 000
(1)
1873 015562
(1) 001676
(1) 016601
(1)
1874 015564
(1) 001677
(1) 015564 002574
(1)
1875 015566
(1) 001700
(1) 001
(1) 015566 010241
(1)
(1) 000
1876 015570
(1) 001701
(1) 015570 002642
(1)
1877 015572
(1) 001702
(1) 015572 000776
(1)
1878 015574
(1) 001703
(1) 015574 06036C
(1)
1879 015576
(1) 001704
(1) 015576 115707
(1)
1880 015600
(1) 001705
(1) 015600 000402
(1)
1881 015602
(1) 001706
(1) 015602 06240C
(1)

```

```

NTRS1: LDMA IMM,NXTSP
MICPC=MICPC+1
.IF IDN IMM IMM
<MOVE!LDMAR!IMM!<NXTSP&377>>
.IFF
<MOVE!LDMAR!IMM!<NXTSP>>
.ENDC

LDMA MEMX SELB!SPX!SPC
MICPC=MICPC+1
.IF IDN MEMX IMM
<MOVE!LDMAR!IMM!<SELB!SPX!SPC&377>>
.IFF
<MOVE!LDMAR!MEMX!<SELB!SPX!SPC>>
.ENDC

MEMINC IMM,201
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<201>>

MEM IMM,<<RTHRS>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<RTHRS>>>

LDMA IMM,NXTSP
MICPC=MICPC+1
.IF IDN IMM IMM
<MOVE!LDMAR!IMM!<NXTSP&377>>
.IFF
<MOVE!LDMAR!IMM!<NXTSP>>
.ENDC

MEM IMM,INTSTK ;ASSUME QUEUE WRAP AROUND
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<INTSTK>>

BRWRTE IMM,<<MMEND-2>>
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<<MMEND-2>>>

CMP BR,SPO
MICPC=MICPC+1
<SUBTC!BR!SPO>

Z NTRS2 ;IT DID WRAP AROUND
MICPC=MICPC+1
<JUMP!ZCOND!<NTRS2-INIT&3000*4>!<NTRS2-INIT&777/2>>

BRWRTE IMM,2 ;OFFSET TO NEXT PAIR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>>

MEM BR,ADD!SPO ;UPDATE QUEUE POINTER
MICPC=MICPC+1
<MOVE!WRMEM!BR!<ADD!SPO>>

```

```

1882 015604 001707 NTRR2: BRWRT IMM,20
      (1) 015604 000420 MICPC=MICPC+1
      (1) <MOVE!WRTEBR!IMM!<20>>

1883 015606 001710 SPBR BR,AORB,SP:
      (1) 015606 063701 MICPC=MICPC+1
      (1) <MOVE!SPBRX!BR!AORB!SP!>

1884 015610 001711 BR0 TAB1 :FLAGGED BY ERROR TYPE
      (1) 015610 116342 MICPC=MICPC+1
      (1) <JUMP!BRCON!<TAB1-INIT&3000*4>!<TAB1-INIT&777/2>>

1885 015612 001712 SNAK: LDMA IMM,ISP11
      (1) 015612 010171 MICPC=MICPC+1
      (1) .IF IDN IMM,IMM
      (1) <MOVE!LDMAR!IMM!<ISP11&377>>
      (1) .IFF
      (1) <MOVE!LDMAR!IMM!<ISP11>>
      (1) .ENDC
      (1) 000

1886 015614 001713 SP MEMX,SELB,SP11
      (1) 015614 043231 MICPC=MICPC+1
      (1) <MOVE!SPX!MEMX!SELB!SP11>

1887 015616 001714 SP BR,INCA,SP11 :INCREMENT MSG EXPECTED
      (1) 015616 063071 MICPC=MICPC+1
      (1) <MOVE!SPX!BR!INCA!SP11>

1888 015620 001715 SNAK1: BRWRT IMM,1 :JNNUMB PENING BIT TO BR
      (1) 015620 000401 MICPC=MICPC+1
      (1) <MOVE!WRTEBR!IMM!<1>>

1889 015622 001716 SNAK2: SP BR,AORB,SP10 :UPDATE LINE STATUS WORD
      (1) 015622 063310 MICPC=MICPC+1
      (1) <MOVE!SPX!BR!AORB!SP10>

1890 015624 001717 ALWAYS FLUSH
      (1) 015624 104415 MICPC=MICPC+1
      (1) <JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>

1891 015626 001720 EMTRIG: BRWRT IMM,24
      (1) 015626 000424 MICPC=MICPC+1
      (1) <MOVE!WRTEBR!IMM!<24>>

1892 015630 001721 OUTPUT BR,<SELB!CBA1>
      (1) 015630 062226 MICPC=MICPC+1
      (1) <MOVE!WRROUT!BR!<SELB!CBA1>>

1894 015632 001722 BRWRT IMM,0
      (1) 015632 000400 MICPC=MICPC+1
      (1) <MOVE!WRTEBR!IMM!<0>>

1895 015634 001723 OUTPUT BR,<SELB!CBA2>
      MICPC=MICPC+1
    
```

```

(1) 015634 062227 <MOVE!WROUT!BR!<SELB!0BA2>>
(1)
1896 015636 SPBR IBUS,BM873,SPD :READ BM873 ADDRESS---
(1) 001724 MICPC=MICPC+1
(1) 015636 023740 <MOVE!SPBRX!IBUS!BM873!SPD>
(1)
1897 015640 OUTPUT BR,SELB!OUTDA1 :SET JP LOW BYTE OF ADDRESS
(1) 001725 MICPC=MICPC+1
(1) 015640 062222 <MOVE!WROUT!BR!<SELB!OUTDA1>>
(1)
1898 015642 BRWTE IMM,366 :HIGH BYTE BASE FOR ROM BOOT
(1) 001726 MICPC=MICPC+1
(1) 015642 000762 <MOVE!WRTEBR!IMM!<366>>
(1)
1899 015644 OUTPUT BR,SELB!OUTDA2 :
(1) 001727 MICPC=MICPC+1
(1) 015644 062223 <MOVE!WROUT!BR!<SELB!OUTDA2>>
(1)
1900 015646 EMB: BRWTE IMM,21 :MASK FOR TIMER AND ALSO TO START NPR
(1) 001730 MICPC=MICPC+1
(1) 015646 000421 <MOVE!WRTEBR!IMM!<21>>
(1)
1901 015650 OUT BR,<SELB!OBR>
(1) 001731 MICPC=MICPC+1
(1) 015650 061231 <MOVE!WROUTX!BR!<SELB!OBR>>
(1)
1902 015652 OUT BR,<SELB!ONPR>
(1) 001732 MICPC=MICPC+1
(1) 015652 061230 <MOVE!WROUTX!BR!<SELB!ONPR>>
(1)
1903 015654 EM1: BRWTE IBUS,NPR :READ NPR CONTROL
(1) 001733 MICPC=MICPC+1
(1) 015654 120600 <MOVE!WRTEBR!IBUS!<NPR>>
(1)
1904 015656 BRO CKTIME
(1) 001734 MICPC=MICPC+1
(1) 015656 102372 <JUMP!BROCON!<CKTIME-INIT&3000*4>!<CKTIME-INIT&777/2>>
(1)
1905 015660 MEMADR RM1 :IF NPR DONE
(1) 001735 MICPC=MICPC+1
(1) 001 .IF B
(1) 015660 002662 <MOVE!WRMEM!<RM1-INIT&777/2>>
(1) .IFF
(1) <MOVE!WRMEM!!<RM1-INIT&777/2>>
(1) .ENDC
(1) 000
1906 015662 ALWAYS ACLOW
(1) 001736 MICPC=MICPC+1
(1) 015662 100765 <JUMP!ALCOND!<ACLOW-INIT&3000*4>!<ACLOW-INIT&777/2>>
(1)
1907 015664 TABUPD: SPBR IBUS,RCVCON,SPD :READ RECEIVER CONTROL REG
(1) 001737 MICPC=MICPC+1
(1) 015664 023640 <MOVE!SPBRX!IBUS!RCVCON!SPD>
(1)
1908 015666 BRWTE BR,ADD!SPD :SHIFT LEFT
(1) 001740 MICPC=MICPC+1
(1) 015666 060400 <MOVE!WRTEBR!BR!<ADD!SPD>>

```

J14

```

1909 015670          BR7      IDLE          :RECEIVE ACTIVE--IDLE
      (1) 015670 001741 MICPC=MICPC+1
      (1) 015670 103445 <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

1910 015672          TAB1:  LDMA      IMM,IMG10
      (1) 015672 001742 MICPC=MICPC+1
      (1) 015672 001743 .IF IDN IMM,IMM
      (1) 015672 010156 <MOVE!LDMAR!IMM!<IMG10&377>
      (1) 015672 001744 .IFF
      (1) 015672 000402 <MOVE!LDMAR!IMM!<IMG10>>
      (1) 015672 000403 .ENDC

1911 015674          BRWRTE IMM,2
      (1) 015674 001743 MICPC=MICPC+1
      (1) 015674 000402 <MOVE!WRTEBR!IMM!<2>>

1912 015676          MEMINC  BR,AANDB!SP10          ;SAVE BIT 1 OF SP10
      (1) 015676 001744 MICPC=MICPC+1
      (1) 015676 076670 <MOVE!WRMEM!INCMAR!BR!<AANDB!SP10>>

1913 015700          MEMINC  BR,SELA!SP11          ;SAVE SP11
      (1) 015700 001745 MICPC=MICPC+1
      (1) 015700 076611 <MOVE!WRMEM!INCMAR!BR!<SELA!SP11>>

1914 015702          MEMINC  BR,SELA!SP12          ;SAVE SP12
      (1) 015702 001746 MICPC=MICPC+1
      (1) 015702 076612 <MOVE!WRMEM!INCMAR!BR!<SELA!SP12>>

1915 015704          MEMINC  BR,SELA!SP17          ;SAVE SP17
      (1) 015704 001747 MICPC=MICPC+1
      (1) 015704 076617 <MOVE!WRMEM!INCMAR!BR!<SELA!SP17>>

1916 015706          STATE  RB2          ;NOTE: THE BRG CANNOT BE CHANGED FROM THIS
      (1) 015706 001750 MICPC=MICPC+1
      (1) 015706 000461 <MOVE!WRTEBR!IMM!<RB2-INIT&777/2>>
      (1) 015706 000462 .POINT UNTIL THE BRANCH TO RB0

1918 015710          CMP      BR,SP3
      (1) 015710 001751 MICPC=MICPC+1
      (1) 015710 060363 <SUBTC!BR!SP3>

1920 015712          Z        IDLE
      (1) 015712 001752 MICPC=MICPC+1
      (1) 015712 101445 <JUMP!ZCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

1921 015714          SP      MEMX,SELB,SP13
      (1) 015714 001753 MICPC=MICPC+1
      (1) 015714 043233 <MOVE!SPX!MEMX!SELB!SP13>

1923 015716          PSTATE TBUI          ;NEW PORT STATE ADDRESS
      (1) 015716 001754 MEM      IMM,<<TBUI-INIT&777/2>>
      (2) 015716 002775 MICPC=MICPC+1
      (2) 015716 002775 <MOVE!WRMEM!IMM!<<TBUI-INIT&777/2>>>

```



```

1924 015720          SP      IMM,4,SP4          :INITIALIZE COUNT
(1) 015720 001755  MICPC=MICPC+1
(1) 015720 003004  <MOVE!SPX!IMM!4!SP4>

1925                                     :NOTE: FIRST 6 RAM LOCATIONS ARE NOT WRITTEN
1926                                     :TO CORE TABLE.
1927 015722          LDMA     IMM,BASE          :MAR NOW POINTS TO BASE
(1) 015722 001756  MICPC=MICPC+1
(1) 015722 001  .IF IDN IMM,IMM
(1) 015722 010017  <MOVE!LDMAR!IMM!<BASE&377>
(1)  .IFF
(1)  <MOVE!LDMAR!IMM!<BASE>
(1)  .ENDC
(1) 000

1928 015724          ALWAYS  RBO              ;THE BRG MUST BE PRESET TO STATE RB2
(1) 015724 001757  MICPC=MICPC+1
(1) 015724 104443  <JUMP!ALCOND!<RBO-INIT&3000*4>!<RBO-INIT&777/2>>

1929 015726          EC2:   BRWRTE IMM,2          :INCREMENT COUNT/TEST
(1) 015726 001760  MICPC=MICPC+1
(1) 015726 000402  <MOVE!WRTEBR!IMM!<2>>

1930 015730          SP      BR,ADD,SP4
(1) 015730 001761  MICPC=MICPC+1
(1) 015730 063004  <MOVE!SPX!BR!ADD!SP4>

1931 015732          SP      IBUS,I0BA1,SP0      :POINT TO NEXT ADDRESS
(1) 015732 001762  MICPC=MICPC+1
(1) 015732 023140  <MOVE!SPX!IBUS!I0BA1!SP0>

1932 015734          OUTPUT  BR,ACD!OBA1
(1) 015734 001763  MICPC=MICPC+1
(1) 015734 062006  <MOVE!WROUT!BR!<ADD!OBA1>>

1933 015736          SP      IBUS,I0BA2,SP0
(1) 015736 001764  MICPC=MICPC+1
(1) 015736 023160  <MOVE!SPX!IBUS!I0BA2!SP0>

1934 015740          OUTPUT  BR,AC!OBA2
(1) 015740 001765  MICPC=MICPC+1
(1) 015740 062107  <MOVE!WROUT!BR!<AC!OBA2>>

1935 015742          C      TABMXT
(1) 015742 001766  MICPC=MICPC+1
(1) 015742 105373  <JUMP!CCOND!<TABMXT-INIT&3000*4>!<TABMXT-INIT&777/2>>

1936                                     :
1937 015744          ECX:   BRWRTE BR!LDMAR,SELA!SP4 :READ COUNTER
(1) 015744 001767  MICPC=MICPC+1
(1) 015744 070604  <MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>

1938 015746          BR7     20$              :ALL DONE
(1) 015746 001770  MICPC=MICPC+1
(1) 015746 117774  <JUMP!BR7CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>

1939 015750          OUTPUT  MEMX!INCMAR,SELB!OUTCA1 :STORE COUNTS OF ERRORS
  
```

```

(1) 015750 001771 MICPC=MICPC+1
(1) 056222 <MOVE!WROUT!MEMX!INCMAR!<SELB!OLTDA1>>
(1)
1940 015752 OUTPUT MEMX!INCMAR,SELB!OUTDA2
(1) 001772 MICPC=MICPC+1
(1) 015752 056223 <MOVE!WROUT!MEMX!INCMAR!<SELB!OUTDA2>>
(1)
1941 015754 ALWAYS RKB
(1) 001773 MICPC=MICPC+1
(1) 015754 104641 <JUMP!ALCOND!<RKB-INIT&3000*4>!<RKB-INIT&777/2>>
(1)
1942
1943 015756 205: LDMA IMM,PRST
(1) 001774 MICPC=MICPC+1
(1) 001 .IF IDN IMM,IMM
(1) 015756 01C162 <MOVE!LDMAR!IMM!<PRST&377>>
(1) .IFF
(1) <MOVE!LDMAR!IMM!<PRST>>
(1) .ENDC
(1) 000
1944 015760 MEM BR,SELA!SP13
(1) 001775 MICPC=MICPC+1
(1) 015760 062613 <MOVE!WRMEM!BR!<SELA!SP13>>
(1)
1945 015762 ALWAYS RM1
(1) 001776 MICPC=MICPC+1
(1) 015762 114662 <JUMP!ALCOND!<RM1-INIT&3000*4>!<RM1-INIT&777/2>>
(1)
1946 015764 $ZERO
(1) 001777 MICPC=MICPC+1
(1) 015764 000000 000000
(1)
1947 000001 .END
ABS. 01576E 000

```

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DDCMP/CRF/DS:CRF+DMCNEW,LOW,DDCNEW
RUN-TIME: 5 9 0 SECONDS
RUN-TIME RATIO: 118/14=8.0
CORE USED: 7K (13 PAGES)

M14

1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679

015766	012737	000001	001226
015774	012737	016100	001216
016002	104412		
016004	013701	001404	
016010	005011		
016012	012705	052525	
016016	010561	000004	
016022	104414		
016024	120500		
016026	104414		
016030	061620		
016032	104414		
016034	061225		
016036	006005		
016040	116104	000005	
016044	120504		
016046	001401		
016050	104012		
016052	104414		
016054	061620		
016056	104414		
016060	061225		
016062	006005		
016064	116104	000005	
016070	120504		
016072	001401		
016074	104012		
016076	104400		

TST1:
1\$:
2\$:

```

***** TEST 1 *****
*TEST OF BR RIGHT SHIFT
*VERIFY THAT A DEST OF BR RSH (011) OF A MICRO-INSTRUCTION
*SHIFTS THE RESULTING BR DATA RIGHT ONCE.
*****

: TEST 1
-----
MOV #1,TSTNO
MOV #TST2,NEXT

MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
MOV DMCSR,R1 ;MASTER CLEAR DMC11
CLR (R1) ;R1 = DMC BASE ADDRESS
MOV #52525,R5 ;CLEAR SEL0
MOV R5,4(R1) ;START WITH 125
ROMCLK PORT4+125
120500 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
ROMCLK BR + PORT4
061620 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
ROMCLK BR RSH+BR, SHIFT BR RIGHT
061225 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
ROR R5 ;PORT5+BR
MOV 5(R1),R4 ;R5 = "EXPECTED"
CMPB R5,R4 ;R4 = "FOUND"
BEQ 1$ ;DID BR SHIFT RIGHT ONCE?
HLT 12 ;BR IF YES
;BR RIGHT SHIFT ERROR

ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
061620 ;BR RSH+BR, SHFT BR RIGHT AGAIN
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
061225 ;PORT5+BR
ROR R5 ;R5 = "EXPECTED"
MOV 5(R1),R4 ;R4 = "FOUND"
CMPB R5,R4 ;DID BR SHIFT RIGHT?
BEQ 2$ ;BR IF YES
HLT 12 ;BR RIGHT SHIFT ERROR
SCOPE ;SCOPE THIS TEST
  
```

```

***** TEST 2 *****
*IOP CRAM WRITE/READ TEST
*FLOAT A 1 THROUGH EACH CRAM LOCATION
*****

```

```

: TEST 2
-----
MOV #2,TSTNO
MOV #TST3,NEXT
MOV #3$,LOCK

BIT #BIT15,STAT1
BEQ 5$
CLR R0
  
```

```

;R1 CONTAINS BASE DMC11 ADDRESS
;DOES DMC HAVE CRAM?
;SKIP TEST IF NO CRAM
;R0 = CRAM ADDRESS
  
```

```

1680 016134 012702 000001      1$:  MOV      #1,R2      ;R2 = WRITE DATA
1681 016140                      2$:
1682 016140 012711 00200C      3$:  MOV      #BIT10,(R1)  ;SET ROMO
1683 016144 010061 000004      MOV      R0,4(R1)      ;WRITE ADDRESS TO SEL4
1684 016150 010261 000006      MOV      R2,6(R1)      ;LOAD SEL6 WITH WRITE DATA
1685 016154 052711 020000      BIS      #BIT13,(R1)  ;WRITE SEL6 INTO CRAM
1686 016160 016104 000004      MOV      4(R1),R4      ;READ CRAM INTO "FOUND"
1687 016164 020204                      CMP      R2,R4          ;IS DATA CORRECT?
1688 016166 001401                      BEQ      4$             ;BR IF OK
1689 016170 104001                      HLT      1              ;ERROR
1690 016172 104401      4$:  SCOPE1
1691 016174 000241                      CLC                          ;CLEAR CARRY
1692 016176 006102                      ROL      R2              ;SHIFT WRITE DATA
1693 016200 001357                      BNE      2$             ;BR IF NOT DONE THIS ADDRESS
1694 016202 00520C                      INC      R0              ;BUMP TO NEXT CRAM ADDRESS
1695 016204 02270C 0020CC      CMP      #2000,R0      ;DONE YET?
1696 016210 001351                      BNE      1$             ;BR IF NO
1697 016212 10440C      5$:  SCOPE      ;SCOPE THIS TEST

```

```

;***** TEST 3 *****
;*IOP CRAM WRITE/READ TEST
;*FLOAT A 0 THROUGH EACH CRAM LOCATION
;*****

```

; TEST 3

```

1707 016214 012737 000003 001226  TST3:  MOV      #3,TSTNO
1708 016222 012737 01633E 001216      MOV      #TST4,NEXT
1709 016230 012737 016260 001220      MOV      #3$,LOCK
1710
1711 016236 104412                      MSTCLR                      ;R1 CONTAINS BASE DMC11 ADDRESS
1712 016240 032737 100000 001366      BIT      #BIT15,STAT1  ;MASTER CLEAR DMC11
1713 016246 001432                      BEQ      5$             ;DOES DMC HAVE CRAM?
1714 016250 00500C                      CLR      R0              ;SKIP TEST IF NO CRAM
1715 016252 012702 000001      1$:  MOV      #1,R2      ;R0 = CRAM ADDRESS
1716 016256                      2$:  MOV      #1,R2      ;R2 = WRITE DATA
1717 016256 005102      4:  COM      R2          ;MAKE IT A FLOATING ZERO
1718 016260 012711 0C2000      3$:  MOV      #BIT10,(R1)  ;SET ROMO
1719 016264 010061 000004      MOV      R0,4(R1)      ;WRITE ADDRESS TO SEL4
1720 016270 010261 00000E      MOV      R2,6(R1)      ;LOAD SEL6 WITH WRITE DATA
1721 016274 052711 020000      BIS      #BIT13,(R1)  ;WRITE SEL6 INTO CRAM
1722 016300 016104 000004      MOV      4(R1),R4      ;READ CRAM INTO "FOUND"
1723 016304 020204                      CMP      R2,R4          ;IS DATA CORRECT?
1724 016306 001401                      BEQ      4$             ;BR IF OK
1725 016310 104001                      HLT      1              ;ERROR
1726 016312 104401      4$:  SCOPE1
1727 016314 005102                      COM      R2              ;BACK TO FLOATING ONE
1728 016316 000241                      CLC                          ;CLEAR CARRY
1729 016320 006102                      ROL      R2              ;SHIFT WRITE DATA
1730 016322 001355                      BNE      2$             ;BR IF NOT DONE THIS ADDRESS
1731 016324 00520C                      INC      R0              ;BUMP TO NEXT CRAM ADDRESS
1732 016326 02270C 0020CC      CMP      #2000,R0      ;DONE YET?
1733 016332 001347                      BNE      1$             ;BR IF NO
1734 016334 10440C      5$:  SCOPE      ;SCOPE THIS TEST

```

1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791

016336 012737 000004 001226
016344 012737 016516 001216
016352 012737 016374 001220

016360 104412
016362 032737 100000 001366

016370 001451
016372 005000
016374 010002
016376 012711 002000
016402 010061 000004
016406 010061 000006
016412 052711 020000
016416 005061 000006
016422 016104 000006
016426 020004
016430 001401
016432 104001
016434 104401
016436 005200
016440 022700 002000
016444 001353
016446 005000
016450 012737 016456 001220
016456 0100
016460 012 002000
016464 010061 000004
016470 016104 000006
016474 020004
016476 001401
016500 104002
016502 104401
016504 005200
016506 022700 002000
016512 001361
016514 104400

TST4:

1\$:

2\$:

3\$:

4\$:

5\$:

***** TEST 4 *****
*IOP CRAM DUAL ADDRESSING TEST
*WRITE EACH ADDRESS INTO ITSELF READ EACH
*ADDRESS TO VERIFY CORRECT ADDRESSING

: TEST 4

```
MOV #4,TSTNO  
MOV #TSTS,NEXT  
MOV #15,LOCK  
  
MSTCLR  
BIT #BIT15,STAT1  
BEQ 5$  
CLR R0  
MOV R0,R2  
MOV #BIT10,(R1)  
MOV R0,4(R1)  
MOV R0,6(R1)  
BIS #BIT13,(R1)  
CLR 6(R1)  
MOV 6(R1),R4  
CMP R0,R4  
BEQ 2$  
HLT 1  
SCOPI  
INC R0  
CMP #2000,R0  
BNE 1$  
CLR R0  
MOV #3$,LOCK  
MOV R0,R2  
MOV #BIT10,(R1)  
MOV R0,4(R1)  
MOV 6(R1),R4  
CMP R0,R4  
BEQ 4$  
HLT 2  
SCOPI  
INC R0  
CMP #2000,R0  
BNE 3$  
SCOPE
```

:R1 CONTAINS BASE DMC11 ADDRESS
:MASTER CLEAR DMC11
:DOES DMC HAVE CRAM?
:SKIP TEST IF NO CRAM
:R0 =CRAM ADDRESS
:SAVE R2 FOR TYPEOUT
:SET ROMO
:WRITE ADDRESS TO SEL4
:LOAD SEL6 WITH WRITE DATA
:WRITE CRAM
:CLEAR SEL 6
:SHOULD READ BACK OWN ADDRESS
:IS DATA CORRECT?
:BR IF YES
:DATA ERROR
:LOOP TO 1\$ IF SW09=1
:BUMP TO NEXT ADDRESS
:DONE WRITING YET?
:BR IF NO
:RESTART AT ADDRESS 0
:NEW SCOPI
:SAVE R2 FOR TYPEOUT
:SET ROMO
:SEL4 = CRAM ADDRESS?
:READ CRAM INTO "FOUND"
:IS DATA CORRECT?
:BR IF YES
:DUAL ADDRESSING ERROR
:LOOP TO 3\$ IF SWC9=1
:BUMP TO NEXT ADDRESS
:DONE WRITING YET?
:BR IF NO
:SCOPE THIS TEST

***** TEST 5 *****
*IOP CRAM READ TEST
*THIS TEST WRITES THE CRAM WITH THE CRAM MICRO-CODE MAP
*THEN READS IT BACK AND COMPARES EACH ADDRESS WITH THE
*DUPLICATE OF THE CRAM MICRO-CODE.

: TEST 5

```

1792 016516 012737 000005 001226      TST5:  MOV      #5,TSTNO
1793 016524 012737 016636 001216      MOV      #TST5,NEXT
1794 016532 012737 016566 001220      MOV      #1$,LOCK
1795                                     ;R1 CONTAINS BASE DMC11 ADDRESS
1796 016540 104412      MSTCLR   ;MASTER CLEAR DMC11
1797 016542 032737 100000 001366      BIT      #BIT15,STAT1 ;IS IT RAM OR ROM
1798 016550 001431      BEQ      3$      ;SKIP TEST IF CROM
1799 016552 005011      CLR      (R1)    ;CLEAR RUN
1800 016554 004737 035562      JSR      PC,WROM ;WRITE CRAM WITH MAP
1801 016560 01270C 0117E6      MOV      #ROMMAP,R0 ;SOFTWARE POINTER TO CROM DUPLICATE
1802 016564 005002      CLR      R2      ;R2 = CROM ADDRESS
1803 016566 010261 000904      1$: MOV      R2,4(R1) ;WRITE CROM ADDRESS TO SEL4
1804 016572 012711 002000      MOV      #BIT10,(R1) ;SET CROMO
1805 016576 011005      MOV      (R0),R5 ;PUT "EXPECTED" IN R5
1806 016600 016104 000006      MOV      6(R1),R4 ;PUT "FOUND" IN R4
1807 016604 020504      CMP      R5,R4   ;COMPARE HARD ROM TO SOFT DUPLICATE
1808 016606 001401      BEQ      2$      ;BR IF OK
1809 016610 104003      HLT      3       ;CRAM READ ERROR!
1810 016612 005011      2$: CLR      (R1)    ;CLR BIT10
1811 016614 005061 000006      CLR      6(R1)   ;CLEAR SEL6
1812 016620 104401      SCOPE1  ;LOOP TO 1$ IF SW09=1
1813 016622 005202      INC      R2      ;INC TO NEXT CROM ADDRESS
1814 016624 005720      TST      (R0)+   ;POP R0 BY 2
1815 016626 022702 002000      CMP      #2000,R2 ;DONE 1K YET?
1816 016632 001355      BNE      1$      ;BR IF NO
1817 016634 10440C      3$: SCOPE   ;SCOPE THIS TEST

```

```

:***** TEST 6 *****
:*IOP MAIN MEMORY TEST
:*FLOAT A 1 THROUGH ALL MAIN MEMORY LOCATIONS
:*****

```

: TEST 6

```

1827 016636 012737 000006 001226      TST6:  MOV      #6,TSTNO
1828 016644 012737 017024 001216      MOV      #TST7,NEXT
1829 016652 012737 016702 001220      MOV      #65$,LOCK
1830                                     ;R1 CONTAINS BASE DMC11 ADDRESS
1831 016660 104412      MSTCLR   ;MASTER CLEAR DMC11
1832 016662 032737 100000 001366      BIT      #BIT15,STAT1 ;IS THIS AN IOP?
1833 016670 001454      BEQ      2$      ;SKIP TEST IF NO
1834 016672 005037 034664      CLR      FLAG    ;START WITH ADDRESS 0
1835 016676 012700 000001      1$: MOV      #1,R0  ;START WITH BIT 0
1836 016702 042737 000377 016734      65$: BIC      #377,66$ ;CLEAR ADDRESS FIELD OF INSTRUCTION
1837 016710 042737 000003 016740      BIC      #3,68$   ;CLEAR ADDRESS FIELD OF INSTRUCTION
1838 016716 53737 034664 016734      BISB    FLAG,66$ ;ADD ADDRESS TO INSTRUCTION
1839 016724 53737 034665 016740      BISB    FLAG+1,68$ ;ADD ADDRESS TO INSTRUCTION
1840 016732 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1841 016734 010000      66$: 010000 ;LOAD MAR LO WITH ADDRESS IN FLAG
1842 016736 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1843 016740 004000      68$: 004000 ;LOAD MAR HI
1844 016742 010061 000004      MOV      R0,4(R1) ;WRITE PATTERN IN PORT4
1845 016746 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1846 016750 122500      122500 ;MOVE PORT4 TO MEMORY
1847 016752 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

1848	016754	040620			040620		: MOVE MEMORY TO BR
1849	016756	104414			ROMCLK		: NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1850	016760	061225			61225		: MOVE BR TO PORT5
1851	016762	010005			MOV	RO,R5	: PUT "EXPECTED" IN R5
1852	016764	116104	000005		MOVB	5(R1),R4	: PUT "FOUND" IN R4
1853	016770	120504			CMPB	R5,R4	: DATA CORRECT?
1854	016772	001401			BEQ	67\$: BR IF YES
1855	016774	104010			HLT	10	: DATA ERROR
1856	016776	104401		67\$:	SCOP1		: SW09=1?
1857	017000	000241			CLC		: CLEAR CARRY
1858	017002	106100			ROLB	RO	: SHIFT BIT IN RO
1859	017004	001336			BNE	65\$: DONE IF RO=0
1860	017006	005237	034664		INC	FLAG	: NEXT ADDRESS
1861	017012	022737	002000	034664	CMP	#2000,FLAG	: LAST ADDRESS?
1862	017020	001326			BNE	1\$: BR IF NO
1863	017022	104400		2\$:	SCOPE		: SCOPE THIS TEST

:***** TEST 7 *****
 :*IOP MAIN MEMORY TEST
 :*FLOAT A 0 THROUGH ALL MAIN MEMORY LOCATIONS
 :*****

: TEST 7

1873	017024	012737	000007	001226	TST7:	MOV	#7,TSTNO		
1874	017032	012737	017216	001216		MOV	#TST10,NEXT		
1875	017040	012737	017072	001220		MOV	#65\$,LOCK		
1876									:R1 CONTAINS BASE DMC11 ADDRESS
1877	017046	104412			MSTCLR				:MASTER CLEAR DMC11
1878	017050	032737	100000	001366	BIT	#BIT15,STAT1			: IS THIS AN IOP?
1879	017056	001456			BEQ	2\$: SKIP TEST IF NO
1880	017060	005037	034664		CLR	FLAG			: START WITH ADDRESS 0
1881	017064	012700	000001		1\$:	MOV	#1,RO		: START WITH BIT 0
1882	017070	005100			64\$:	COM	RO		: CHANGE TO FLOATING 0
1883	017072	042737	000377	017124	65\$:	BIC	#377,66\$: CLEAR ADDRESS FIELD OF INSTRUCTION
1884	017100	042737	000003	017130		BIC	#3,68\$: CLEAR ADDRESS FIELD OF INSTRUCTION
1885	017106	153737	034664	017124		BISB	FLAG,66\$: ADD ADDRESS TO INSTRUCTION
1886	017114	153737	034665	017130		BISB	FLAG+1,68\$: ADD ADDRESS TO INSTRUCTION
1887	017122	104414			ROMCLK				: NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1888	017124	010000		66\$:	010000				: LOAD MAR LO WITH ADDRESS IN FLAG
1889	017126	104414			ROMCLK				: NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1890	017130	004000		68\$:	004000				: LOAD MAR HI
1891	017132	010061	000004		MOV	RO,4(R1)			: WRITE PATTERN IN PORT4
1892	017136	104414			ROMCLK				: NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1893	017140	122500			122500				: MOVE PORT4 TO MEMORY
1894	017142	104414			ROMCLK				: NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1895	017144	040620			040620				: MOVE MEMORY TO BR
1896	017146	104414			ROMCLK				: NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1897	017150	061225			61225				: MOVE BR TO PORT5
1898	017152	010005			MOV	RO,R5			: PUT "EXPECTED" IN R5
1899	017154	116104	000005		MOVB	5(R1),R4			: PUT "FOUND" IN R4
1900	017160	120504			CMPB	R5,R4			: DATA CORRECT?
1901	017162	001401			BEQ	67\$: BR IF YES
1902	017164	104010			HLT	10			: DATA ERROR
1903	017166	104401		67\$:	SCOP1				: SW09=1?

E15

```

1904 017170 005100
1905 017172 000241
1906 017174 106100
1907 017176 001334
1908 017200 005237 034664
1909 017204 022737 002000 034664
1910 017212 001324
1911 017214 104400 2$:
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922 017216 012737 000010 001226 TST10:
1923 017224 012737 017516 001226
1924 017232 012737 017256 001220
1925
1926 017240 104412
1927 017242 032737 100000 001366
1928 017250 001521
1929 017252 005037 034664
1930 017256 013702 034664 1$:
1931 017262 042737 000377 017314
1932 017270 042737 000003 017320
1933 017276 153737 034664 017314
1934 017304 153737 034665 017320
1935 017312 104414
1936 017314 010000 2$:
1937 017316 104414
1938 017320 004000 7$:
1939 017322 010261 000004
1940 017326 104414
1941 017330 122500
1942 017332 104414
1943 017334 040620
1944 017336 104414
1945 017340 061225
1946 017342 010205
1947 017344 116104 000005
1948 017350 120504
1949 017352 001401
1950 017354 104010
1951 017356 104401 3$:
1952 017360 005237 034664
1953 017364 022737 002000 034664
1954 017372 001331
1955 017374 012737 017406 001220
1956 017402 005037 034664
1957 017406 013702 034664 4$:
1958 017412 042737 000377 017444
1959 017420 042737 000003 017450

```

```

COM RC :CHANGE TO FLOATING I
CLC :CLEAR CARRY
ROLB RO :SHIFT BIT IN RO
BNE 64$ :DONE IF RO=0
INC FLAG :NEXT ADDRESS
CMP #2000,FLAG :LAST ADDRESS?
BNE 1$ :BR IF NO
SCOPE :SCOPE THIS TEST

```

```

:***** TEST 10 *****
:*IOP MAIN MEMORY DUAL ADDRESSING TEST
:*LOAD EACH MEMORY LOCATION WITH ITS OWN ADDRESS
:*READ BACK EACH LOCATION TO VERIFY CORRECT ADDRESSING
:*****

```

: TEST 10

```

-----
TST10: MOV #10,TSTNO
MOV #TST11,NEXT
MOV #1$,LOCK

MSTCLR :R1 CONTAINS BASE DMC11 ADDRESS
BIT #BIT15,STAT1 :MASTER CLEAR DMC11
BEQ 9$ :IS THIS AN IOP?
CLR FLAG :SKIP TEST IF NO
MOV FLAG,R2 :START AT ADDRESS 0
BIC #377,2$ :PUT DATA IN R2
BIC #3,7$ :CLEAR ADDRESS FIELD OF INSTRUCTION
BISB FLAG,2$ :CLEAR ADDRESS FIELD OF INSTRUCTION
BISB FLAG+1,7$ :ADD ADDRESS TO INSTRUCTION
ROMCLK :ADD ADDRESS TO INSTRUCTION
:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1$: 010000 :LOAD MAR LO
2$: ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
7$: 004000 :LOAD MAR HI
MOV R2,4(R1)
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
122500 :MOVE PORT4 TO MEMORY
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
040620 :MOVE MEMORY TO THE BR
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
61225 :MOV BR TO PORTS
MOV R2,R5 :PUT "EXPECTED" IN R5
MOVB 5(R1),R4 :PUT "FOUND" IN R4
CMPB R5,R4 :DATA CORRECT?
BEQ 3$ :BR IF YES
HLT 10 :DATA ERROR
SCOPE :SW09=1?
INC FLAG :NEXT ADDRESS
CMP #2000,FLAG :LAST ADDRESS
BNE 1$ :BR IF NO
MOV #4$,LOCK :NEW SCOPE 1
CLR FLAG :RESTART AT ADDRESS 0
MOV FLAG,R2 :PUT DATA IN R2
BIC #377,5$ :CLEAR ADDRESS FIELD OF INSTRUCTION
BIC #3,8$ :CLEAR ADDRESS FIELD OF INSTRUCTION

```


F15

1960	017426	153737	034664	017444	BISB	F_FLAG,5\$:ADD ADDRESS TO INSTRUCTION
1961	017434	153737	034665	017450	BISB	F_FLAG+1.9\$:ADD ADDRESS TO INSTRUCTION
1962	017442	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1963	017444	010000			010000		:LOAD THE MAR LO
1964	017446	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1965	017450	004000			004000		:LOAD MAR HI
1966	017452	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1967	017454	040620			040620		:MOVE MEMORY TO THE BR
1968	017456	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1969	017460	061225			61225		:MOV BR TO PORTS
1970	017462	010205			MOV	R2,R5	:PUT "EXPECTED" IN R5
1971	017464	116104	000005		MOVB	5(R1),R4	:PUT "FOUND" IN R4
1972	017470	120504			CMPB	R5,R4	:DATA CORRECT?
1973	017472	001401			BEQ	6\$:BR IF YES
1974	017474	104010			HLT	10	:ADDRESSING ERROR
1975	017476	104401			SCOP1		:SW09=1?
1976	017500	005237	034664		INC	F_FLAG	:NEXT ADDRESS
1977	017504	022737	002000	034664	CMP	#2000,F_FLAG	:IS IT THE LAST
1978	017512	001335			BNE	4\$:BR IF NO
1979	017514	104400			SCOPE		:SCOPE THIS TEST

```

:***** TEST 11 *****
:*IOP MAR TEST
:*PERFORM DUAL ADDRESSING TEST
:*USING MAR AUTO-INC FEATURE
:*****

```

: TEST 11

1990	017516	012737	000011	001226	TST11:	MOV	#11,TSTNO	
1991	017524	012737	017632	001216		MOV	#TST12,NEXT	
1992								:R1 CONTAINS BASE DMC11 ADDRESS
1993	017532	104412			MSTCLR		:MASTER CLEAR DMC11	
1994	017534	032737	100000	001366	BIT	#BIT15,STAT1	:IS THIS AN IOP?	
1995	017542	001432			BEQ	4\$:SKIP TEST IF NO	
1996	017544	005002			CLR	R2	:START WITH A ZERO	
1997	017546	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304	
1998	017550	010000			010000		:LOAD MAR WITH A ZERO	
1999	017552	010261	000004		MOV	R2,4(R1)	:WRITE DATA TO PORT4	
2000	017556	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304	
2001	017560	136500			136500		:MEM+PORT4, AUTO-INC MAR	
2002	017562	005202			INC	R2	:INCREMENT DATA	
2003	017564	022702	002000		CMP	#2000,R2	:DONE YET?	
2004	017570	001370			BNE	1\$:BR IF NO	
2005	017572	005002			CLR	R2	:RESTART WITH A ZERO	
2006	017574	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304	
2007	017576	010000			010000		:LOAD MAR WITH A ZERO	
2008	017600							
2009	017600	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304	
2010	017602	055224			055224		:MOVE MEM TO PORT4	
2011	017604	010205			MOV	R2,R5	:PUT "EXPECTED" IN R5	
2012	017606	016104	000004		MOV	4(R1),R4	:PUT "FOUND" IN R4	
2013	017612	120504			CMPB	R5,R4	:DATA CORRECT?	
2014	017614	001401			BEQ	3\$:BR IF YES	
2015	017616	104011			HLT	11	:MAR ERROR	

```

2016 017620 005202 35:
2017 017622 022702 002000
2018 017626 001354
2019 017630 104400 45:
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031 017632 012737 000012 001226
2032 017640 012737 020040 001216
2033 017646 012737 017674 001220
2034
2035 017654 104412
2036 017656 032737 100000 001366
2037 017664 001464
2038 017666 005002
2039 017670 104414
2040 017672 010000
2041 017674
2042 017674 104414
2043 017676 121204
2044 017700 005005
2045 017702 032702 000400
2046 017706 001402
2047 017710 012705 000040
2048 017714 016104 000004
2049 017720 042704 177637
2050 017724 020504
2051 017726 001401
2052 017730 104007
2053 017732 104401
2054 017734 104414
2055 017736 014000
2056 017740 005202 002000
2057 017742 022702
2058 017746 001352 001220
2059 017750 005037
2060 017754 104414
2061 017756 121204
2062 017760 012705 000100
2063 017764 016104 000004
2064 017770 042704 177637
2065 017774 020504
2066 017776 001401
2067 020000 104007
2068 020002 104414
2069 020004 010000
2070 020006 104414
2071 020010 004000

```

```

INC R2 :NEXT ADDRESS
CMP #2000,R2 :DONE YET?
BNE 25 :BR IF NO
SCOPE :SCOPE THIS TEST

```

```

:***** TEST 12 *****
:*IOP (CRAM) ODT BITS TEST
:*LOAD MAR WITH A 0 INC MAR UNTIL IT OVERFLOWS (2000 TIMES,
:*VERIFY THAT IBUS* 10 BITS IS SET ONLY WHEN MAR BIT 8 IS A ONE
:*AND THAT IBUS* 10 BIT6 IS SET ON MAR OVERFLOW(2000,
:*****

```

TEST 12

```

TST12: MOV #12,TSTNC :R1 CONTAINS BASE DMC11 ADDRESS
MOV #TST13,NEXT :MASTER CLEAR DMC11
MOV #15,LOCK :IS THIS AN IOP?
MSTCLR :SKIP TEST IF NO
BIT #BIT15,STAT1 :R2=SAME AS MAR CONTENTS
BEQ 25 :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
CLR R2 :MAR+0
ROMCLK 010000
IS: ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
121204 :PORT4=IBUS* 10
CLR R5 :R5="EXPECTED"
BIT #BIT8,R2 :IS BIT8 SET IN MAR?
BEQ +6 :BR IF NO
MOV #BIT5,R5 :IF YES THEN SET BITS
MOV 4(R1),R4 :R4="FOUND"
BIC #177637,R4 :CLEAR UNWANTED BITS
CMP R5,R4 :BITS 5&6 SHOULD BE CLEAR
BEQ +4 :BR IF OK
HLT 7 :ERROR BITS 5&6 NOT CLEAR
SCOPE1 :LOOP TO 11$ IF SW09=1
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
014000 :INC MAR
INC R2 :BUMP MEM ADDRESS
CMP #2000,R2 :OVERFLOWED YET?
BNE 15 :BR IF NO
CLR LOCK :NO MORE SCOPE1
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
121204 :PART4+IBUS* 10
MOV #BIT6,R5 :R5="EXPECTED"
MOV 4(R1),R4 :R4="FOUND"
BIC #177637,R4 :CLEAR UNWANTED BITS
CMP R5,R4 :BIT6 SHOULD BE SET
BEQ +4 :BR IF OK
HLT 7 :ERROR, BIT6 NOT SET
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
010000 :MAR+0
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
004000 :MAR HI+0

```

```

2072 020012 104414
2073 020014 121204
2074 020016 005005
2075 020020 016104 000004
2076 020024 042704 177537
2077 020030 020504
2078 020032 001401
2079 020034 104007
2080 020036 104400 2$:

```

```

ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
121204 :PORT4+IBUS* 10
CLR R5 :R5="EXPECTED"
MOV 4(R1),R4 :R4="FOUND"
BIC #177537,R4 :CLEAR UNWANTED BITS
CMP R5,R4 :BITS 5&6 SHOULD BE CLEAR
BEQ +4 :BR IF OK
HLT 7 :ERROR 5&6 NOT BOTH CLEAR
SCOPE :SCOPE THIS TEST

```

```

:***** TEST 13 *****
:*CROM READ TEST
:*THIS TEST READS EACH ROM LOCATION AND COMPARES
:*IT TO A SOFTWARE DUPLICATE OF THE CROM. THIS TEST
:*ALSO TESTS THE JUMP(I) MICRO-PROCESSOR INSTRUCTION.
:*****

```

: TEST 13

```

2092 020040 012737 000013 001226
2093 020046 012737 020230 001216
2094 020054 012737 020106 001220
2095
2096 020062 104412
2097 020064 032737 100000 001366
2098 020072 001055
2099 020074 005011
2100 020076 012700 011766
2101 020102 005002
2102 020104 005003
2103 020106 042737 014377 020126
2104 020114 050237 020126
2105 020120 050337 020126
2106 020124 104414
2107 020126 100400 2$:
2108 020130 012711 002000
2109 020134 011005
2110 020136 016104 000006
2111 020142 020504
2112 020144 001414
2113 020146 010337 001252
2114 020152 000241
2115 020154 006037 001252
2116 020160 006037 001252
2117 020164 006037 001252
2118 020170 050237 001252
2119 020174 104004
2120 020176 104401 3$:
2121 020200 005720
2122 020202 005202
2123 020204 022702 000400
2124 020210 001336
2125 020212 005002
2126 020214 062703 004000
2127 020220 022703 020000

```

```

TST13: MOV #13,TSTNO
MOV #TST14,NEXT
MOV #15,LOCK
MSTCLR :R1 CONTAINS BASE DMC11 ADDRESS
BIT #BIT15,STAT1 :MASTER CLEAR DMC11
BNE 4$ :IS IT RAM OR ROM
CLR (R1) :SKIP TEST IF CROM
MOV #ROMMAP,RC :CLEAR RUN
CLR R2 :R0 POINTS TO SOFTWARE ROM MAP
CLR R3 :R2 CONTAINS ROM ADDRESS BITS 0-7
BIC #14377,2$ :R3 CONTAINS ROM ADDRESS BITS 8&9 IN BITS 11&12
BIS R2,2$ :CLEAR ADDRESS FIELDS OF INSTRUCTION
BIS R3,2$ :ADD BITS 0-7 TO INSTRUCTION
ROMCLK :ADD BITS 11&12 TO INSTRUCTION
100400 :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2$: JUMP(I) TO ROM ADDRESS IN R2 & R3
MOV #BIT10,(R1) :SET ROM0
MOV (R0),R5 :PUT "EXPECTED" IN R5
MOV 6(R1),R4 :PUT "FOUND" IN R4
CMP R5,R4 :COMPARE ROM CONTENTS TO SOFT DUP
BEQ 3$ :BR IF OK
MOV R3,TEMP3 :PUT ROM ADDRESS IN TEMP3
CLC :FOR ERROR TYPEOUT
ROR TEMP3
ROR TEMP3
ROR TEMP3
BIS R2,TEMP3 :TEMP3 NOW CONTAINS CORRECT ADDRESS
HLT 4 :ROM READ ERROR
SCOPE1 :LOOP TO 1$ IF SW09=1
TST (R0)+ :BUMP SOFT POINTER
INC R2 :BUMP ROM ADDRESS
CMP #400,R2 :IS R2 TO MAX YET?
BNE 1$ :BR IF NO
CLR R2 :YES. RESET R2 TO 0
ADC #4000,R3 :INC TO NEXT PAGE OF ROM
CMP #20000,R3 :DONE YET?

```

```

2128 020224 001330
2129 020226 104400
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141 020230 012737 000014 001226
2142 020236 012737 020420 001216
2143 020244 012737 020264 001220
2144
2145 020252 104412
2146 020254 032737 100000 001366
2147 020262 001055
2148 020264
2149 020264 004737 035410
2150 020270 104414
2151 020272 100400
2152 020274 104414
2153 020276 114377
2154 020300 004737 035502
2155 020304 000002
2156 020306 020504
2157 020310 001401
2158 020312 104006
2159 020314 104401
2160 020316 012737 020324 001220
2161 020324
2162 020324 004737 035410
2163 020330 104414
2164 020332 100403
2165 020334 104414
2166 020336 100000
2167 020340 004737 035502
2168 020344 000010
2169 020346 020504
2170 020350 001401
2171 020352 104006
2172 020354 104401
2173 020356 012737 020364 001220
2174 020364
2175 020364 004737 035410
2176 020370 104414
2177 020372 100406
2178 020374 104414
2179 020376 104125
2180 020400 004737 035502
2181 020404 000016
2182 020406 020504
2183 020410 001401

```

```

4$: BNE 1$ :BR IF NO
SCOPE :SCOPE THIS TEST

***** TEST 14 *****
*CROM TEST OF JUMP(I) NEVER MICRO-PROCESSOR INSTRUCTION.
*PERFORM THE JUMP INSTRUCTION
*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).
*****

; TEST 14
-----
TST14: MOV #14,TSTNO
MOV #TST15,NEXT
MOV #1$,LOCK
;R1 CONTAINS BASE DMC11 ADDRESS
;MASTER CLEAR DMC11
;IS IT CROM?
;SKIP TEST IF YES

1$: JSR PC,CLRALL :CLEAR ALL CONDITIONS
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 :START AT ROM PC=0
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
114377!<400*0> :JUMP TO ROM PC OF 1777
JSR PC,ROMDAT :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2 :INDEX
CMP R5,R4 :ARE NEW PC CONTENTS CORRECT?
BEQ 2$ :BR IF YES
HLT 6 :ERROR, CROM PC IS WRONG
2$: SCOP1 :LOOP TO 1$ IF SW09=1
MOV #3$,LOCK :NEW SCOP1

3$: JSR PC,CLRALL :CLEAR ALL CONDITIONS
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100403 :START AT ROM PC=3
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100000!<400*0> :JUMP TO ROM PC OF 0
JSR PC,ROMDAT :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
10 :INDEX
CMP R5,R4 :ARE NEW PC CONTENTS CORRECT?
BEQ 4$ :BR IF YES
HLT 6 :ERROR, CROM PC IS WRONG
4$: SCOP1 :LOOP TO 3$ IF SW09=1
MOV #5$,LOCK :NEW SCOP1

5$: JSR PC,CLRALL :CLEAR ALL CONDITIONS
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100406 :START AT ROM PC=6
ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
104125!<400*0> :JUMP TO ROM PC OF 525
JSR PC,ROMDAT :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
16 :INDEX
CMP R5,R4 :ARE NEW ROM PC CONTENTS CORRECT?
BEQ 6$ :BR IF YES

```

```

2184 020412 104006
2185 020414 104401
2186 020416 104400
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197 020420 012737 000015 001226
2198 020426 012737 020574 001216
2199 020434 012737 020454 001220
2200
2201 020442 104412
2202 020444 032737 100000 001366
2203 020452 001047
2204 020454
2205 020454 104414
2206 020456 100400
2207 020460 104414
2208 020462 114777
2209 020464 004737 035502
2210 020470 003776
2211 020472 020504
2212 020474 001401
2213 020476 104006
2214 020500 104401
2215 020502 012737 020510 001220
2216 020510
2217 020510 104414
2218 020512 100403
2219 020514 104414
2220 020516 100400
2221 020520 004737 035502
2222 020524 000000
2223 020526 020504
2224 020530 001401
2225 020532 104006
2226 020534 104401
2227 020536 012737 020544 001220
2228 020544
2229 020544 104414
2230 020546 100406
2231 020550 104414
2232 020552 104525
2233 020554 004737 035502
2234 020560 001252
2235 020562 020504
2236 020564 001401
2237 020566 104006
2238 020570 104401
2239 020572 104400

;***** TEST 15 *****
;*CROM TEST OF JUMP(I) ALWAYS MICRO-PROCESSOR INSTRUCTION.
;*PERFORM THE JUMP INSTRUCTION
;*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
;*****

; TEST 15
-----
TST15: MOV #15,TSTNO
MOV #TST16,NEXT
MOV #1$,LOCK

MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
BIT #BIT15,STAT1 ;MASTER CLEAR DMC11
BNE 6$+2 ;IS IT CROM?
;SKIP TEST IF YES

1$: ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 ;START AT ROM PC=0
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
114377!<400*1> ;JUMP TO ROM PC OF 1777
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
3776 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 2$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
2$: SCOPI ;LOOP TO 1$ IF SW09=1
MOV #3$,LOCK ;NEW SCOPI

3$: ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100403 ;START AT ROM PC=3
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100000!<400*1> ;JUMP TO ROM PC OF 0
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
0 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 4$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
4$: SCOPI ;LOOP TO 3$ IF SW09=1
MOV #5$,LOCK ;NEW SCOPI

5$: ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100406 ;START AT ROM PC=6
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
104125!<400*1> ;JUMP TO ROM PC OF 525
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1252 ;INDEX
CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
BEQ 6$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
6$: SCOPI ;LOOP TO 5$ IF SW59=1
SCOPE ;SCOPE THIS TEST

```

***** TEST 16 *****
*CROM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
*SET THE C BIT, PERFORM THE JUMP INSTRUCTION.
*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

TEST 16

020574 012737 000016 001226
020602 012737 020764 001216
020610 012737 020630 001220
020616 104412
020620 032737 100000 001366
020626 001355
020630
020630 004737 035456
020634 104414
020636 100400
020640 104414
020642 115377
020644 004737 035502
020650 003776
020652 020504
020654 001401
020656 104006
020660 104401
020662 012737 020670 001220
020670
020670 004737 035456
020674 104414
020676 100403
020700 104414
020702 101000
020704 004737 035502
020710 000000
020712 020504
020714 001401
020716 104006
020720 104401
020722 012737 020730 001220
020730
020730 004737 035456
020734 104414
020736 100406
020740 104414
020742 105125
020744 004737 035502
020750 001252
020752 020504
020754 001401
020756 104006
020760 104401
020762 104400

TST16: MOV #16,TSTNC
MOV #TST17,NEXT
MOV #1\$,LOCK
MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
BIT #BIT15,STAT1 ;MASTER CLEAR DMC11
BNE 6\$+2 ;IS IT CROM?
 ;SKIP TEST IF YES
1\$: JSR PC,SETC ;SET THE C BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 ;START AT ROM PC=0
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
114377! <400*2> ;JUMP TO ROM PC OF 1777
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
3776 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 2\$;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
2\$: SCOP1 ;LOOP TO 1\$ IF SW09=1
MOV #3\$,LOCK ;NEW SCOP1
3\$: JSR PC,SETC ;SET THE C BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100403 ;START AT ROM PC=3
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100000! <400*2> ;JUMP TO ROM PC OF 0
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
0 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 4\$;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
4\$: SCOP1 ;LOOP TO 3\$ IF SW09=1
MOV #5\$,LOCK ;NEW SCOP1
5\$: JSR PC,SETC ;SET THE C BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100406 ;START AT ROM PC=6
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
104125! <400*2> ;JUMP TO ROM PC OF 525
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1252 ;INDEX
CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
BEQ 6\$;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
6\$: SCOP1 ;LOOP TO 5\$ IF SW59=1
SCOP2 ;SCOPE THIS TEST

2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351

020764 012737 C00017 001226
020772 012737 021154 001216
021000 012737 021020 001220
021006 104412
021010 032737 100000 001366
021016 031055
021020
021024 004737 035474
021026 104414
021030 100400
021032 104414
021034 004737 035502
021040 003776
021042 020504
021044 001401
021046 104006
021050 104401
021052 012737 021060 001220
021060
021060 004737 035474
021064 104414
021066 100403
021070 104414
021072 101400
021074 004737 035502
021100 000000
021102 020504
021104 001401
021106 104006
021110 104401
021112 012737 021120 001220
021120
021120 004737 035474
021124 104414
021126 100406
021130 104414
021132 105525
021134 004737 035502
021140 001252
021142 020504
021144 001401
021146 104006
021150 104401
021152 104400

```

:***** TEST 17 *****
:*CROM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
:*SET THE Z BIT, PERFORM THE JUMP INSTRUCTION.
:*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
:*****

: TEST 17
-----
TST17: MOV #17,TSTNO
MOV #TST20,NEXT
MOV #1$,LOCK

MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
;MASTER CLEAR DMC11
BIT #BIT15,STAT1 ;IS IT CROM?
BNE 6$,+2 ;SKIP TEST IF YES

1$: JSR PC,SETZ ;SET THE Z BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 ;START AT ROM PC=0
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
114377! <400*3> ;JUMP TO ROM PC OF 1777
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
3776 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 2$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
2$: SCOP1 ;LOOP TO 1$ IF SW09=1
MOV #3$,LOCK ;NEW SCOP1

3$: JSR PC,SETZ ;SET THE Z BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100403 ;START AT ROM PC=3
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100000! <400*3> ;JUMP TO ROM PC OF 0
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
0 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 4$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
4$: SCOP1 ;LOOP TO 3$ IF SW09=1
MOV #5$,LOCK ;NEW SCOP1

5$: JSR PC,SETZ ;SET THE Z BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100406 ;START AT ROM PC=6
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
104125! <400*3> ;JUMP TO ROM PC OF 525
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1252 ;INDEX
CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
BEQ 6$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
6$: SCOP1 ;LOOP TO 5$ IF SW59=1
SCOPE ;SCOPE THIS TEST

```

```

2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362 021154 012737 000020 001226
2363 021162 012737 021344 001216
2364 021170 012737 021210 001220
2365
2366 021176 104412
2367 021200 032737 100000 001366
2368 021206 001055
2369 021210
2370 021210 004737 035426
2371 021214 104414
2372 021216 100400
2373 021220 104414
2374 021222 116377
2375 021224 004737 035502
2376 021230 003776
2377 021232 020504
2378 021234 001401
2379 021236 104006
2380 021240 104401
2381 021242 012737 021250 001220
2382 021250
2383 021250 004737 035426
2384 021254 104414
2385 021256 100403
2386 021260 104414
2387 021262 102000
2388 021264 004737 035502
2389 021270 000000
2390 021272 020504
2391 021274 001401
2392 021276 104006
2393 021300 104401
2394 021302 012737 021310 001220
2395 021310
2396 021310 004737 035426
2397 021314 104414
2398 021316 100406
2399 021320 104414
2400 021322 106125
2401 021324 004737 035502
2402 021330 001252
2403 021332 020504
2404 021334 001401
2405 021336 104006
2406 021340 104401
2407 021342 104400

```

```

:***** TEST 20 *****
:*CROM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
:*SET THE BRO BIT, PERFORM THE JUMP INSTRUCTION.
:*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
:*****

```

: TEST 20

```

-----
TST20: MOV #20,TSTNO
MOV #TST21,NEXT
MOV #1$,LOCK

MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
BIT #BIT15,STAT1 ;MASTER CLEAR DMC11
BNE 6$+2 ;IS IT CROM?
;SKIP TEST IF YES

1$: JSR PC,SETBRO ;SET THE BRO BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 ;START AT ROM PC=0
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
114377! <400*4> ;JUMP TO ROM PC OF 1777
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
3776 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 2$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG

2$: SCOP1 ;LOOP TO 1$ IF SW09=1
MOV #3$,LOCK ;NEW SCOP1

3$: JSR PC,SETBRO ;SET THE BRO BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100403 ;START AT ROM PC=3
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100000! <400*4> ;JUMP TO ROM PC OF 0
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
0 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 4$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG

4$: SCOP1 ;LOOP TO 3$ IF SW09=1
MOV #5$,LOCK ;NEW SCOP1

5$: JSR PC,SETBRO ;SET THE BRO BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100406 ;START AT ROM PC=6
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
104125! <400*4> ;JUMP TO ROM PC OF 525
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1252 ;INDEX
CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
BEQ 6$ ;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG

6$: SCOP1 ;LOOP TO 5$ IF SW59=1
SCOPE ;SCOPE THIS TEST

```


2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463

021344 012737 000021 001226
021352 012737 021534 001216
021360 012737 021400 001220

021366 104412
021370 032737 100000 001366
021376 001055
021400
021400 004737 035434
021404 104414
021406 100400
021410 104414
021412 116777
021414 004737 035502
021420 003776
021422 020504
021424 001401
021426 104006
021430 104401
021432 012737 021440 001220
021440
021440 004737 035434
021444 104414
021446 100403
021450 104414
021452 102400
021454 004737 035502
021460 000000
021462 020504
021464 001401
021466 104006
021470 104401
021472 012737 021500 001220
021500
021500 004737 035434
021504 104414
021506 100406
021510 104414
021512 106525
021514 004737 035502
021520 001252
021522 020504
021524 001401
021526 104006
021530 104401
021532 104400

```
***** TEST 21 *****  
*CROM TEST OF JUMP(I) ON BRI SET MICRO-PROCESSOR INSTRUCTION.  
*SET THE BRI BIT, PERFORM THE JUMP INSTRUCTION.  
*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC  
*****  
  
; TEST 21  
-----  
TST21: MOV #21,TSTNO  
MOV #TST22,NEXT  
MOV #1$,LOCK  
  
MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS  
;MASTER CLEAR DMC11  
BIT #BIT15,STAT1 ;IS IT CROM?  
BNE 6$+2 ;SKIP TEST IF YES  
  
1$: JSR PC,SETBRI ;SET THE BRI BIT'  
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
100400 ;START AT ROM PC=0  
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
114377! <400*5> ;JUMP TO ROM PC OF 1777  
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA  
3776 ;INDEX  
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?  
BEQ 2$ ;BR IF YES  
HLT 6 ;ERROR, CROM PC IS WRONG  
2$: SCOP1 ;LOOP TO 1$ IF SW09=1  
MOV #3$,LOCK ;NEW SCOP1  
  
3$: JSR PC,SETBRI ;SET THE BRI BIT'  
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
100403 ;START AT ROM PC=3  
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
100000! <400*5> ;JUMP TO ROM PC OF 0  
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA  
0 ;INDEX  
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?  
BEQ 4$ ;BR IF YES  
HLT 6 ;ERROR, CROM PC IS WRONG  
4$: SCOP1 ;LOOP TO 3$ IF SW09=1  
MOV #5$,LOCK ;NEW SCOP1  
  
5$: JSR PC,SETBRI ;SET THE BRI BIT'  
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
100406 ;START AT ROM PC=6  
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304  
104125! <400*5> ;JUMP TO ROM PC OF 525  
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA  
1252 ;INDEX  
CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?  
BEQ 6$ ;BR IF YES  
HLT 6 ;ERROR, CROM PC IS WRONG  
6$: SCOP1 ;LOOP TO 5$ IF SW59=1  
SCOPE ;SCOPE THIS TEST
```

***** TEST 22 *****
*CROM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
*SET THE BR4 BIT, PERFORM THE JUMP INSTRUCTION,
*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

TEST 22

TST22: MOV #22,TSTNO ;R1 CONTAINS BASE DMC11 ADDRESS
MOV #TST23,NEXT ;MASTER CLEAR DMC11
MOV #1\$,LOCK ;IS IT CROM?
;SKIP TEST IF YES
MSTCLR ;SET THE BR4 BIT'
BIT #BIT15,STAT1 ;NEXT WORD IS INSTRUCTION, ROMCLK PC=53C4
BNE 6\$+2 ;START AT ROM PC=0
;SKIP TEST IF YES ;NEXT WORD IS INSTRUCTION, ROMCLK PC=53C4
1\$: JSR PC,SETBR4 ;SET THE BR4 BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=53C4
100400 ;START AT ROM PC=0
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=53C4
114377! <400*6> ;JUMP TO ROM PC OF 1777
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
3776 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 2\$;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
2\$: SCOPI ;LOOP TO 1\$ IF SW09=1
MOV #3\$,LOCK ;NEW SCOPI
3\$: JSR PC,SETBR4 ;SET THE BR4 BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=53C4
100403 ;START AT ROM PC=3
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=53C4
100000! <400*6> ;JUMP TO ROM PC OF 0
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
0 ;INDEX
CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
BEQ 4\$;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
4\$: SCOPI ;LOOP TO 3\$ IF SW09=1
MOV #5\$,LOCK ;NEW SCOPI
5\$: JSR PC,SETBR4 ;SET THE BR4 BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=53C4
100406 ;START AT ROM PC=6
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=53C4
104125! <400*6> ;JUMP TO ROM PC OF 525
JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1252 ;INDEX
CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
BEQ 6\$;BR IF YES
HLT 6 ;ERROR, CROM PC IS WRONG
6\$: SCOPI ;LOOP TO 5\$ IF SW59=1
SCOPE ;SCOPE THIS TEST

2474 021534 012737 000022 001226
2475 021542 012737 021724 001216
2476 021550 012737 021570 001220
2477
2478 021556 104412
2479 021560 032737 100000 001366
2480 021566 001055
2481 021570
2482 021570 004737 035442
2483 021574 104414
2484 021576 100400
2485 021600 104414
2486 021602 117377
2487 021604 004737 035502
2488 021610 003776
2489 021612 020504
2490 021614 001401
2491 021616 104006
2492 021620 104401
2493 021622 012737 021630 001220
2494 021630
2495 021630 004737 035442
2496 021634 104414
2497 021636 100403
2498 021640 104414
2499 021642 103000
2500 021644 004737 035502
2501 021650 000000
2502 021652 020504
2503 021654 001401
2504 021656 104006
2505 021660 104401
2506 021662 012737 021670 001220
2507 021670
2508 021670 004737 035442
2509 021674 104414
2510 021676 100406
2511 021700 104414
2512 021702 107125
2513 021704 004737 035502
2514 021710 001252
2515 021712 020504
2516 021714 001401
2517 021716 104006
2518 021720 104401
2519 021722 104400

***** TEST 23 *****
*CROM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
*SET THE BR7 BIT, PERFORM THE JUMP INSTRUCTION.
*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC

TEST 23

2520										
2521										
2522										
2523										
2524										
2525										
2526										
2527										
2528										
2529										
2530	021724	012737	000023	001226	TST23:	MOV	#23,TSTNO			
2531	021732	012737	022114	001216		MOV	#TST24,NEXT			
2532	021740	012737	021760	001220		MOV	#1\$,LOCK			
2533										
2534	021746	104412				MSTCLR				:R1 CONTAINS BASE DMC11 ADDRESS
2535	021750	032737	100000	001366		BIT	#BIT15,STAT1			:MASTER CLEAR DMC11
2536	021756	001055				BNE	6\$+2			:IS IT CROM?
2537	021760				1\$:					:SKIP TEST IF YES
2538	021760	004737	035450			JSR	PC,SETBR7			:SET THE BR7 BIT'
2539	021764	104414				ROMCLK				:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2540	021766	100400				100400				:START AT ROM PC=0
2541	021770	104414				ROMCLK				:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2542	021772	117777				114377!<400*7>				:JUMP TO ROM PC OF 1777
2543	021774	004737	035502			JSR	PC,ROMDAT			:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2544	022000	003776				3776				:INDEX
2545	022002	020504				CMP	R5,R4			:ARE NEW PC CONTENTS CORRECT?
2546	022004	001401				BEQ	2\$:BR IF YES
2547	022006	104006				HLT	6			:ERROR, CROM PC IS WRONG
2548	022010	104401			2\$:	SCOPI				:LOOP TO 1\$ IF SW03=1
2549	022012	012737	022020	001220		MOV	#3\$,LOCK			:NEW SCOPI
2550	022020				3\$:					
2551	022020	004737	035450			JSR	PC,SETBR7			:SET THE BR7 BIT'
2552	022024	104414				ROMCLK				:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2553	022026	100403				100403				:START AT ROM PC=3
2554	022030	104414				ROMCLK				:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2555	022032	103400				100000!<400*7>				:JUMP TO ROM PC OF 0
2556	022034	004737	035502			JSR	PC,ROMDAT			:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2557	022040	000000				0				:INDEX
2558	022042	020504				CMP	R5,R4			:ARE NEW PC CONTENTS CORRECT?
2559	022044	001401				BEQ	4\$:BR IF YES
2560	022046	104006				HLT	6			:ERROR, CROM PC IS WRONG
2561	022050	104401			4\$:	SCOPI				:LOOP TO 3\$ IF SW09=1
2562	022052	012737	022060	001220		MOV	#5\$,LOCK			:NEW SCOPI
2563	022060				5\$:					
2564	022060	004737	035450			JSR	PC,SETBR7			:SET THE BR7 BIT'
2565	022064	104414				ROMCLK				:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2566	022066	100406				100406				:START AT ROM PC=6
2567	022070	104414				ROMCLK				:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2568	022072	107525				104125!<400*7>				:JUMP TO ROM PC OF 525
2569	022074	004737	035502			JSR	PC,ROMDAT			:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2570	022100	001252				1252				:INDEX
2571	022102	020504				CMP	R5,R4			:ARE NEW ROM PC CONTENTS CORRECT?
2572	022104	001401				BEQ	6\$:BR IF YES
2573	022106	104006				HLT	6			:ERROR, CROM PC IS WRONG
2574	022110	104401			6\$:	SCOPI				:LOOP TO 5\$ IF SW59=1
2575	022112	104400				SCOPE				:SCOPE THIS TEST

***** TEST 24 *****
*CROM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
*CLEAR THE C BIT, PERFORM THE JUMP INSTRUCTION,
*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).

TEST 24

2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631

022114 012737 000024 001226
022122 012737 022304 001216
022130 012737 022150 001220

022136 104412
022140 032737 100000 001366
022146 001055
022150
022150 004737 035410
022154 104414
022156 100400
022160 104414
022162 115377
022164 004737 035502
022170 000002
022172 020504
022174 001401
022176 104006
022200 104401
022202 012737 022210 001220
022210
022210 004737 035410
022214 104414
022216 100403
022220 104414
022222 101000
022224 004737 035502
022230 000010
022232 020504
022234 001401
022236 104006
022240 104401
022242 012737 022250 001220
022250
022250 004737 035410
022254 104414
022256 100406
022260 104414
022262 105125
022264 004737 035502
022270 000016
022272 020504
022274 001401
022276 104006
022300 104401

TST24:

1\$:

2\$:
3\$:

4\$:
5\$:
6\$:

MOV #24,TSTNO
MOV #TST25,NEXT
MOV #1\$,LOCK

MSTCLR
BIT #BIT15,STAT1
BNE 6\$+2

JSR PC,CLRALL
ROMCLK
100400
ROMCLK
114377! <400*2>
JSR PC,ROMDAT
2
CMP R5,R4
BEQ 2\$
HLT 6

SCOPI
MOV #3\$,LOCK

JSR PC,CLRALL
ROMCLK
100403
ROMCLK
100000! <400*2>
JSR PC,ROMDAT
10
CMP R5,R4
BEQ 4\$
HLT 6

SCOPI
MOV #5\$,LOCK

JSR PC,CLRALL
ROMCLK
100406
ROMCLK
104125! <400*2>
JSR PC,ROMDAT
16
CMP R5,R4
BEQ 6\$
HLT 6
SCOPI

;R1 CONTAINS BASE DMC11 ADDRESS
;MASTER CLEAR DMC11
;IS IT CROM?
;SKIP TEST IF YES

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

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

;CLEAR ALL CONDITIONS
;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
;START AT ROM PC=6
;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
;JUMP TO ROM PC OF 525
;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
;INDEX
;ARE NEW ROM PC CONTENTS CORRECT?
;BR IF YES
;ERROR, CROM PC IS WRONG
;LOOP TO 5\$ IF SW59=1

Address	OpCode	Op1	Op2	Op3	Op4	Scope	Comments
2632	022302	104400				SCOPE	:SCOPE THIS TEST
2633							
2634							
2635							:***** TEST 25 *****
2636							:*CROM TEST OF JUMP('I' ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
2637							:*CLEAR THE Z BIT, PERFORM THE JUMP INSTRUCTION,
2638							:*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
2639							:*THE CONTENTS OF THE NEW ROM PC (IT SHOULD INCREMENT BY ONE).
2640							:*****
2641							
2642							
2643							: TEST 25
2644	022304	012737	000025	001226		TST25:	MOV #25,TSTNC
2645	022312	012737	022474	001216			MOV #TST26,NEXT
2646	022320	012737	022340	001220			MOV #15,LOCK
2647							
2648	022326	104412					:R1 CONTAINS BASE DMC11 ADDRESS
2649	022330	032737	100000	001366			:MASTER CLEAR DMC11
2650	022336	001055					:IS IT CROM?
2651	022340					15:	:SKIP TEST IF YES
2652	022340	004737	035410				:CLEAR ALL CONDITIONS
2653	022344	104414					:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2654	022346	100400					:START AT ROM PC=0
2655	022350	104414					:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2656	022352	115777					:JUMP TO ROM PC OF 1777
2657	022354	004737	035502				:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2658	022360	000002					:INDEX
2659	022362	020504					:ARE NEW PC CONTENTS CORRECT?
2660	022364	001401					:BR IF YES
2661	022366	104006					:ERROR, CROM PC IS WRONG
2662	022370	104401				25:	:LOOP TO 15 IF SW09=1
2663	022372	012737	022400	001220			:NEW SCOPI
2664	022400					35:	
2665	022400	004737	035410				:CLEAR ALL CONDITIONS
2666	022404	104414					:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2667	022406	100403					:START AT ROM PC=3
2668	022410	104414					:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2669	022412	101400					:JUMP TO ROM PC OF 0
2670	022414	004737	035502				:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2671	022420	000010					:INDEX
2672	022422	020504					:ARE NEW PC CONTENTS CORRECT?
2673	022424	001401					:BR IF YES
2674	022426	104006					:ERROR, CROM PC IS WRONG
2675	022430	104401				45:	:LOOP TO 35 IF SW09=1
2676	022432	012737	022440	001220			:NEW SCOPI
2677	022440					55:	
2678	022440	004737	035410				:CLEAR ALL CONDITIONS
2679	022444	104414					:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2680	022446	100406					:START AT ROM PC=6
2681	022450	104414					:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2682	022452	105525					:JUMP TO ROM PC OF 525
2683	022454	004737	035502				:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2684	022460	000016					:INDEX
2685	022462	020504					:ARE NEW ROM PC CONTENTS CORRECT?
2686	022464	001401					:BR IF YES
2687	022466	104006					:ERROR, CROM PC IS WRONG

```

2688 022472 104401      65:  SCOPI      :LOOP TO 55 IF SW59=1
2689 022472 104401      SCOPE      :SCOPE THIS TEST
2690
2691
2692 :***** TEST 26 *****
2693 :*CROM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
2694 :*CLEAR THE BRO BIT, PERFORM THE JUMP INSTRUCTION,
2695 :*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
2696 :*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).
2697 :*****
2698
2699 : TEST 26
2700 :-----
2701 022474 012737 000026 001226  TST26:  MOV      #26,TSTNO
2702 022502 012737 022664 001216      MOV      #TST27,NEXT
2703 022510 012737 022530 001220      MOV      #15,LOCK
2704
2705 022516 104412      MSTCLR      :R1 CONTAINS BASE DMC11 ADDRESS
2706 022520 032737 100000 001366      BIT      #BIT15,STAT1 :MASTER CLEAR DMC11
2707 022526 001055      BNE      65+2         :IS IT CROM?
2708 022530      :SKIP TEST IF YES
2709 022530 004737 035410  15:  JSR      PC,CLRALL   :CLEAR ALL CONDITIONS
2710 022534 104414      ROMCLK     :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2711 022536 100400      100400     :START AT ROM PC=0
2712 022540 104414      ROMCLK     :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2713 022542 116377      114377! <400*4> :JUMP TO ROM PC OF 1777
2714 022544 004737 035502  JSR      PC,ROMDAT  :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2715 022550 000002      2          :INDEX
2716 022552 020504      CMP      R5,R4      :ARE NEW PC CONTENTS CORRECT?
2717 022554 001401      BEQ      25        :BR IF YES
2718 022556 104006      HLT      6         :ERROR, CROM PC IS WRONG
2719 022560 104401      25:  SCOPI      :LOOP TO 15 IF SW09=1
2720 022562 012737 022570 001220  MOV      #35,LOCK  :NEW SCOPI
2721 022570
2722 022570 004737 035410  35:  JSR      PC,CLRALL   :CLEAR ALL CONDITIONS
2723 022574 104414      ROMCLK     :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2724 022576 100403      100403     :START AT ROM PC=3
2725 022600 104414      ROMCLK     :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2726 022602 102000      100000! <400*4> :JUMP TO ROM PC OF 0
2727 022604 004737 035502  JSR      PC,ROMDAT  :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2728 022610 000010      10         :INDEX
2729 022612 020504      CMP      R5,R4      :ARE NEW PC CONTENTS CORRECT?
2730 022614 001401      BEQ      45        :BR IF YES
2731 022616 104006      HLT      6         :ERROR, CROM PC IS WRONG
2732 022620 104401      45:  SCOPI      :LOOP TO 35 IF SW09=1
2733 022622 012737 022630 001220  MOV      #55,LOCK  :NEW SCOPI
2734 022630
2735 022630 004737 035410  55:  JSR      PC,CLRALL   :CLEAR ALL CONDITIONS
2736 022634 104414      ROMCLK     :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2737 022636 100406      100406     :START AT ROM PC=6
2738 022640 104414      ROMCLK     :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2739 022642 106125      104125! <400*4> :JUMP TO ROM PC OF 525
2740 022644 004737 035502  JSR      PC,ROMDAT  :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2741 022650 000016      16         :INDEX
2742 022652 020504      CMP      R5,R4      :ARE NEW ROM PC CONTENTS CORRECT?
2743 022654 001401      BEQ      65        :BR IF YES

```

```

2744 022656 104006 HLT 6 :ERROR, CROM PC IS WRONG
2745 022660 104401 6$: SCOP1 :LOOP TO 5$ IF SW59=1
2746 022662 104400 SCOP2 :SCOPE THIS TEST
2747
2748
2749
2750
2751 :***** TEST 27 *****
2752 :*CROM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.*
2753 :*CLEAR THE BR1 BIT, PERFORM THE JUMP INSTRUCTION,*
2754 :*VERIFY THAT THE JUMP DID NOT OCCUR BY READING*
2755 :*THE CONTENTS OF THE NEW ROM PC (IT SHOULD INCREMENT BY ONE).*
2756 :*****
2757 : TEST 27
2758 022664 012737 000027 001226 TST27: MOV #27,TSTNO
2759 022672 012737 023054 001216 MOV #TST3C,NEWT
2760 022700 012737 022720 001220 MOV #1$,LOCK
2761 :R1 CONTAINS BASE DMC11 ADDRESS
2762 022706 104412 MSTCLR :MASTER CLEAR DMC11
2763 022710 032737 100000 001366 BIT #BIT15,STAT1 :IS IT CROM?
2764 022716 001055 SNE 6$+2 :SKIP TEST IF YES
2765 022720
2766 022720 004737 035410 1$: JSR PC,CLRALL :CLEAR ALL CONDITIONS
2767 022724 104414 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2768 022726 100400 100400 :START AT ROM PC=0
2769 022730 104414 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2770 022732 116777 114377! <400*5> :JUMP TO ROM PC OF 1777
2771 022734 004737 035502 JSR PC,ROMDAT :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2772 022740 000002 2 :INDEX
2773 022742 020504 CMP R5,R4 :ARE NEW PC CONTENTS CORRECT?
2774 022744 001401 BEQ 2$ :BR IF YES
2775 022746 104006 HLT 6 :ERROR, CROM PC IS WRONG
2776 022750 104401 2$: SCOP1 :LOOP TO 1$ IF SW09=1
2777 022752 012737 022760 001220 MOV #3$,LOCK :NEW SCOP1
2778 022760 3$:
2779 022760 004737 035410 JSR PC,CLRALL :CLEAR ALL CONDITIONS
2780 022764 104414 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2781 022766 100403 100403 :START AT ROM PC=3
2782 022770 104414 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2783 022772 102400 100000! <400*5> :JUMP TO ROM PC OF 0
2784 022774 004737 035502 JSR PC,ROMDAT :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2785 023000 000010 10 :INDEX
2786 023002 020504 CMP R5,R4 :ARE NEW PC CONTENTS CORRECT?
2787 023004 001401 BEQ 4$ :BR IF YES
2788 023006 104006 HLT 6 :ERROR, CROM PC IS WRONG
2789 023010 104401 4$: SCOP1 :LOOP TO 3$ IF SW09=1
2790 023012 012737 023020 001220 MOV #5$,LOCK :NEW SCOP1
2791 023020 5$:
2792 023020 004737 035410 JSR PC,CLRALL :CLEAR ALL CONDITIONS
2793 023024 104414 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2794 023026 100406 100406 :START AT ROM PC=6
2795 023030 104414 ROMCLK :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2796 023032 106525 104125! <400*5> :JUMP TO ROM PC OF 525
2797 023034 004737 035502 JSR PC,ROMDAT :R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2798 023040 000016 16 :INDEX
2799 023042 020504 CMP R5,R4 :ARE NEW ROM PC CONTENTS CORRECT?

```

```

023044 001401
023046 104006
023050 104401
023052 104401
023054 001401
023056 104401
023058 104401
023060 104401
023062 012737
023064 012737
023066 012737
023068 012737
023070 012737
023072 104412
023074 032737
023076 001055
023078 023110
023080 004737
023082 035410
023084 023114
023086 104414
023088 100400
023090 023116
023092 100400
023094 023120
023096 104414
023098 023122
023100 117377
023102 004737
023104 035502
023106 023130
023108 000002
023110 023132
023112 020504
023114 023134
023116 001401
023118 023136
023120 104006
023122 023140
023124 104401
023126 023142
023128 012737
023130 023150
023132 001220
023134 023150
023136 004737
023138 035410
023140 023154
023142 104414
023144 023156
023146 100403
023148 023160
023150 104414
023152 023162
023154 103000
023156 023164
023158 004737
023160 035502
023162 023170
023164 000010
023166 023172
023168 020504
023170 023174
023172 001401
023174 023176
023178 104006
023180 023200
023182 104401
023184 023202
023186 012737
023188 023210
023190 001220
023192 023210
023194 004737
023196 035410
023198 023214
023200 104414
023202 023216
023204 100406
023206 023220
023208 104414
023210 023222
023212 107125
023214 023224
023216 004737
023218 035502
023220 023230
023222 000016

```

```

6$: BEQ 6$ :BR IF YES
HLT 6$ :ERROR, CROM PC IS WRONG
SCOPE :LOOP TO 5$ IF SW59=1
SCOPE :SCOPE THIS TEST

***** TEST 30 *****
* CROM TEST OF JUMP(!) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
* CLEAR THE BR4 BIT, PERFORM THE JUMP INSTRUCTION.
* VERIFY THAT THE JUMP DID NOT OCCUR BY READING
* THE CONTENTS OF THE NEW ROM PC (IT SHOULD INCREMENT BY ONE).
*****

: TEST 30
-----
TST30: MOV #30, TSTNC
MOV #TST31, NEXT
MOV #1$, LOCK
: R1 CONTAINS BASE DMC11 ADDRESS
: MASTER CLEAR DMC11
MSTCLR : IS IT CROM?
BIT #BIT15, STAT1 : SKIP TEST IF YES
BNE 6$+2

1$: JSR PC, CLRALL : CLEAR ALL CONDITIONS
ROMCLK : NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 : START AT ROM PC=0
ROMCLK : NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
114377! <400*6> : JUMP TO ROM PC OF 1777
JSR PC, ROMDAT : R5=EXPECTED ROM DATA, R4=ACTUAL ROM DATA
2 : INDEX
CMP R5, R4 : ARE NEW PC CONTENTS CORRECT?
BEQ 2$ : BR IF YES
HLT 6$ : ERROR, CROM PC IS WRONG
SCOPE : LOOP TO 1$ IF SW09=1
MOV #3$, LOCK : NEW SCOPE

3$: JSR PC, CLRALL : CLEAR ALL CONDITIONS
ROMCLK : NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100403 : START AT ROM PC=3
ROMCLK : NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100000! <400*6> : JUMP TO ROM PC OF 0
JSR PC, ROMDAT : R5=EXPECTED ROM DATA, R4=ACTUAL ROM DATA
10 : INDEX
CMP R5, R4 : ARE NEW PC CONTENTS CORRECT?
BEQ 4$ : BR IF YES
HLT 6$ : ERROR, CROM PC IS WRONG
SCCP1 : LOOP TO 3$ IF SW09=1
MOV #5$, LOCK : NEW SCOPE

5$: JSR PC, CLRALL : CLEAR ALL CONDITIONS
ROMCLK : NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100406 : START AT ROM PC=6
ROMCLK : NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
104125! <400*6> : JUMP TO ROM PC OF 525
JSR PC, ROMDAT : R5=EXPECTED ROM DATA, R4=ACTUAL ROM DATA
16 : INDEX

```


2856	023232	020504				CMP	R5,R4	:ARE NEW ROM PC CONTENTS CORRECT?
2857	023234	001401				BEG	5\$:BR IF YES
2858	023236	104006				HLT	6	:ERROR, CROM PC IS WRONG
2859	023240	104401			6\$:	SCOPI		:LOOP TO 5\$ IF SW59=1
2860	023242	104400				SCOPE		:SCOPE THIS TEST
2861								
2862								
2863								
2864								:***** TEST 31 *****
2865								:*CROM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
2866								:*CLEAR THE BR7 BIT, PERFORM THE JUMP INSTRUCTION.
2867								:*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
2868								:*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).
2869								:*****
2870								
2871								: TEST 31
2872	023244	012737	000031	001226	TST31:	MOV	#31,TSTNO	
2873	023252	012737	023434	001216		MOV	#TST32,NEXT	
2874	023260	012737	023300	001220		MOV	#1\$,LOCK	
2875								:R1 CONTAINS BASE DMC11 ADDRESS
2876	023266	104412				MSTCLR		:MASTER CLEAR DMC11
2877	023270	032737	100000	001366		BIT	#BIT15,STAT1	:IS IT CROM?
2878	023276	001055				BNE	6\$+2	:SKIP TEST IF YES
2879	023300				1\$:			
2880	023300	004737	035410			JSR	PC,CLRALL	:CLEAR ALL CONDITIONS
2881	023304	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2882	023306	100400				100400		:START AT ROM PC=0
2883	023310	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2884	023312	117777				114377! <400*7>		:JUMP TO ROM PC OF 1777
2885	023314	004737	035502			JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2886	023320	000002				2		:INDEX
2887	023322	020504				CMP	R5,R4	:ARE NEW PC CONTENTS CORRECT?
2888	023324	001401				BEG	2\$:BR IF YES
2889	023326	104006				HLT	6	:ERROR, CROM PC IS WRONG
2890	023330	104401			2\$:	SCOPI		:LOOP TO 1\$ IF SW09=1
2891	023332	012737	023340	001220		MOV	#3\$,LOCK	:NEW SCOPI
2892	023340				3\$:			
2893	023340	004737	035410			JSR	PC,CLRALL	:CLEAR ALL CONDITIONS
2894	023344	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2895	023346	100403				100403		:START AT ROM PC=3
2896	023350	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2897	023352	103400				100800! <400*7>	:JUMP TO	:ROM PC OF 0
2898	023354	004737	035502			JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2899	023360	000010				10		:INDEX
2900	023362	020504				CMP	R5,R4	:ARE NEW PC CONTENTS CORRECT?
2901	023364	001401				BEG	4\$:BR IF YES
2902	023366	104006				HLT	6	:ERROR, CROM PC IS WRONG
2903	023370	104401			4\$:	SCOPI		:LOOP TO 3\$ IF SW09=1
2904	023372	012737	023400	001220		MOV	#5\$,LOCK	:NEW SCOPI
2905	023400				5\$:			
2906	023400	004737	035410			JSR	PC,CLRALL	:CLEAR ALL CONDITIONS
2907	023404	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2908	023406	100406				100406		:START AT ROM PC=6
2909	023410	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2910	023412	107525				104125! <400*7>	:JUMP TO	:ROM PC OF 525
2911	023414	004737	035502			JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA

```

29:02 023420 000016
29:03 023422 020504
29:04 023424 001401
29:05 023426 104006
29:06 023430 104401
29:07 023432 104400
29:08
29:09
29:10
29:11
29:12
29:13
29:14
29:15
29:16
29:17
29:18
29:19
29:20
29:21
29:22
29:23
29:24
29:25
29:26
29:27
29:28
29:29
29:30
29:31
29:32
29:33 023434 012737 000032 001226
29:34 023442 012737 023630 001216
29:35 023450 012737 023474 001220
29:36
29:37 023456 104412
29:38 023460 032737 100000 001366
29:39 023466 001457
29:40 023470 004737 035634
29:41 023474
29:42 023474 004737 035410
29:43 023500 104414
29:44 023502 100400
29:45 023504 104414
29:46 023506 114377
29:47 023510 004737 035530
29:48 023514 000001
29:49 023516 120504
29:50 023520 001401
29:51 023522 104005
29:52 023524 104401
29:53 023526 012737 023534 001220
29:54 023534
29:55 023534 004737 035410
29:56 023540 104414
29:57 023542 100403
29:58 023544 104414
29:59 023546 100000
29:60 023550 004737 035530
29:61 023554 000004
29:62 023556 120504
29:63 023560 001401
29:64 023562 104005
29:65 023564 104401
29:66 023566 012737 023574 001220
29:67 023574

```

```

:16 :INDEX
CMP R5,R4 :ARE NEW ROM PC CONTENTS CORRECT?
BEQ E5 :BR IF YES
HLT E :ERROR, CROM PC IS WRONG
5$: SCOPI :LOOP TO 5$ IF SW59=1
SCOPE :SCOPE THIS TEST

:***** TEST 32 *****
:*CRAM TEST OF JUMP(I) NEVER MICRO-PROCESSOR INSTRUCTION.
:*PERFORM THE JUMP INSTRUCTION
:*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
:*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
:*BR WITH THE LOWEST 8 BITS OF THE CROM PC. AT THIS POINT
:*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
:*THE CROM PC IS CORRECT, IF THE CROM -C IS NOT RIGHT,
:*THEN PORT4 CONTAINS A 37
:*****

: TEST 32
-----
TST32: MOV #32,TSTNO ;R1 CONTAINS BASE DMC11 ADDRESS
MOV #TST33,NEXT ;MASTER CLEAR DMC1!
MOV #1$,LOCK ;IS IT CROM?

MSTCLR ;SKIP TEST IF NO
BIT #BIT15,STAT1 ;SET MEM AND RAM
BEQ 6$+2
JSR PC,MEMSET

1$: JSR PC,CLRALL ;CLEAR ALL CONDITIONS
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 ;START AT ROM PC=0
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
114377!<400*0> ;JUMP TO ROM PC OF 1777
JSR PC,RAMDAT ;R4=CROM PC (LSB 8 BITS)
1 ;EXPECTED DATA
CMPB R5,R4 ;IS ROM PC CORRECT?
BEQ 2$ ;BR IF YES
HLT 5 ;ERROR, CROM PC IS WRONG
2$: SCOPI ;LOOP TO 1$ IF SW09=1
MOV #3$,LOCK ;NEW SCOPI

3$: JSR PC,CLRALL ;CLEAR ALL CONDITIONS
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100403 ;START AT ROM PC=3
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100000!<400*0> ;JUMP TO ROM PC OF 0
JSR PC,RAMDAT ;R4=CROM PC (LSB 8 BITS)
4 ;EXPECTED DATA
CMPB R5,R4 ;IS ROM PC CORRECT?
BEQ 4$ ;BR IF YES
HLT 5 ;ERROR, CROM PC IS WRONG
4$: SCOPI ;LOOP TO 3$ IF SW09=1
MOV #5$,LOCK ;NEW SCOPI

5$:

```

K16

2968	023574	004737	035410		JSR	PC,CLEAR	:CLEAR ALL CONDITIONS
2969	023600	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2970	023602	100406			100406		:START AT ROM PC=6
2971	023604	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2972	023606	104125			104125!<400*0>		:JUMP TO ROM PC OF 525
2973	023610	004737	035530		JSR	PC,RAMDAT	:R4=CRAM PC (LSB 8 BITS)
2974	023614	000007			7		:EXPECTED DATA
2975	023616	120504			CMPB	R5,R4	:IS ROM PC CORRECT?
2976	023620	001401			BEQ	6\$:BR IF YES
2977	023622	104005			HLT	5	:ERROR, CRAM PC IS WRONG
2978	023624	104401	6\$:		SCOPI		:LOOP TO 5\$ IF SW59=1
2979	023626	104400			SCOPE		:SCOPE THIS TEST

```

:***** TEST 33 *****
:*CRAM TEST OF JUMP(I) ALWAYS MICRO-PROCESSOR INSTRUCTION.
:*PERFORM THE JUMP INSTRUCTION
:*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
:*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
:*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
:*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
:*THE JUMP WAS SUCCESSFUL. IF THE JUMP WAS UNSUCCESSFUL
:*THEN PORT4 WILL CONTAIN A 37
:*****

```

: TEST 33

2995	023630	012737	000033	001226	TST33:	MOV	#33,TSTNO	
2996	023636	012737	024010	001216		MOV	#TST34,NEXT	
2997	023644	012737	023670	001220		MOV	#1\$,LOCK	
2998								:R1 CONTAINS BASE DMC11 ADDRESS
2999	023652	104412				MSTCLR		:MASTER CLEAR DMC11
3000	023654	032737	100000	001366		BIT	#BIT15,STAT1	:IS IT CRAM?
3001	023662	001451				BEQ	6\$+2	:SKIP TEST IF NO
3002	023664	004737	035634			JSR	PC,MEMSET	:SET MEM AND RAM
3003	023670				1\$:			
3004	023670	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3005	023672	100400				100400		:START AT ROM PC=0
3006	023674	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3007	023676	114777				114777!<400*1>		:JUMP TO ROM PC OF 1777
3008	023700	004737	035530			JSR	PC,RAMDAT	:R4=CRAM PC (LSB 8 BITS)
3009	023704	000377				377		:EXPECTED DATA
3010	023706	120504				CMPB	R5,R4	:IS ROM PC CORRECT?
3011	023710	001401				BEQ	2\$:BR IF YES
3012	023712	104005				HLT	5	:ERROR, CRAM PC IS WRONG
3013	023714	104401	2\$:			SCOPI		:LOOP TO 1\$ IF SW09=1
3014	023716	012737	023724	001220		MOV	#3\$,LOCK	:NEW SCOPI
3015	023724				3\$:			
3016	023724	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3017	023726	100403				100403		:START AT ROM PC=3
3018	023730	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3019	023732	100400				100000!<400*1>	:JUMP TO	:JUMP TO ROM PC OF 0
3020	023734	004737	035530			JSR	PC,RAMDAT	:R4=CRAM PC (LSB 8 BITS)
3021	023740	000000				0		:EXPECTED DATA
3022	023742	120504				CMPB	R5,R4	:IS ROM PC CORRECT?
3023	023744	001401				BEQ	4\$:BR IF YES

```

3024 023746 104005          HLT      5          :ERROR, CRAM PC IS WRONG
3025 023750 104401          SCOPI                    :LOOP TO 3$ IF SW09=1
3026 023752 012737 023760 001220 4$:      MOV      #5$,LOCK    :NEW SCOPI
3027 023760          5$:
3028 023760 104414          ROMCLK                    :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3029 023762 100406          100406                    :START AT ROM PC=6
3030 023764 104414          ROMCLK                    :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3031 023766 104525          104125! <400*1>          :JUMP TO ROM PC OF 525
3032 023770 004737 035530  JSR      PC,RAMDAT        :R4=CRAM PC (LSB 8 BITS)
3033 023774 000125          125                        :EXPECTED DATA
3034 023776 120504          CMPB     R5,R4            :IS ROM PC CORRECT?
3035 024000 001401          BEQ      6$              :BR IF YES
3036 024002 104005          HLT      5              :ERROR, CRAM PC IS WRONG
3037 024004 104401          5$:      SCOPI                    :LOOP TO 5$ IF SW59=1
3038 024006 104400          SCOPE                    :SCOPE THIS TEST

```

```

3039
3040
3041 :***** TEST 34 *****
3042 :*CRAM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
3043 :*SET THE C BIT, PERFORM THE JUMP INSTRUCTION.
3044 :*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
3045 :*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
3046 :*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3047 :*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
3048 :*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
3049 :*THEN PORT4 WILL CONTAIN A 37
3050 :*****

```

```

3051 :
3052 : TEST 34
3053 :-----
3054 024010 012737 000034 001226  TST34:  MOV      #34,TSTNO
3055 024016 012737 024204 001216      MOV      #TST35,NEXT
3056 024024 012737 024050 001220      MOV      #1$,LOCK
3057 :
3058 024032 104412          MSTCLR                    :R1 CONTAINS BASE DMC11 ADDRESS
3059 024034 032737 100000 001366      BIT      #BIT15,STAT1    :MASTER CLEAR DMC11
3060 024042 001457          BEQ      6$+2            :IS IT CRAM?
3061 024044 004737 035634          JSR      PC,MEMSET        :SKIP TEST IF NO
3062 024050          1$:
3063 024050 004737 035456          JSR      PC,SETC ;SET THE C BIT' :SET MEM AND RAM
3064 024054 104414          ROMCLK                    :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3065 024056 100400          100400                    :START AT ROM PC=0
3066 024060 104414          ROMCLK                    :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3067 024062 115377          114377! <400*2>          :JUMP TO ROM PC OF 1777
3068 024064 004737 035530  JSR      PC,RAMDAT        :R4=CRAM PC (LSB 8 BITS)
3069 024070 000377          377                        :EXPECTED DATA
3070 024072 120504          CMPB     R5,R4            :IS ROM PC CORRECT?
3071 024074 001401          BEQ      2$              :BR IF YES
3072 024076 104005          HLT      5              :ERROR, CRAM PC IS WRONG
3073 024100 104401          2$:      SCOPI                    :LOOP TO 1$ IF SW09=1
3074 024102 012737 024110 001220 1$:      MOV      #3$,LOCK    :NEW SCOPI
3075 024110          3$:
3076 024110 004737 035456          JSR      PC,SETC ;SET THE C BIT' :SET THE C BIT'
3077 024114 104414          ROMCLK                    :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3078 024116 100403          100403                    :START AT ROM PC=3
3079 024120 104414          ROMCLK                    :NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

M16

3080	024122	101000				100000! <400*2>	: JUMP TO ROM PC OF 0
3081	024124	004737	035530			JSR	PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3082	024130	000000				0	: EXPECTED DATA
3083	024132	120504				CMPB	R5,R4 ; IS ROM PC CORRECT?
3084	024134	001401				BEQ	4\$; BR IF YES
3085	024136	104005				HLT	5 ; ERROR, CRAM PC IS WRONG
3086	024140	104401			4\$:	SCOPI	: LOOP TO 3\$ IF SW09=1
3087	024142	012737	024150	001220		MOV	#5\$,LOCK ; NEW SCOPI
3088	024150				5\$:		
3089	024150	004737	035456			JSR	PC,SETC ; SET THE C BIT'
3090	024154	104414				ROMCLK	: NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3091	024156	100406				100406	: START AT ROM PC=6
3092	024160	104414				ROMCLK	: NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3093	024162	105125				104125! <400*2>	: JUMP TO ROM PC OF 525
3094	024164	004737	035530			JSR	PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3095	024170	000125				125	: EXPECTED DATA
3096	024172	120504				CMPB	R5,R4 ; IS ROM PC CORRECT?
3097	024174	001401				BEQ	6\$; BR IF YES
3098	024176	104005				HLT	5 ; ERROR, CRAM PC IS WRONG
3099	024200	104401			6\$:	SCOPI	: LOOP TO 5\$ IF SW59=1
3100	024202	104400				SCOPE	: SCOPE THIS TEST

3101
 3102
 3103
 3104
 3105
 3106
 3107
 3108
 3109
 3110
 3111
 3112
 3113
 3114
 3115

```

:***** TEST 35 *****
:*CRAM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
:*SET THE Z BIT, PERFORM THE JUMP INSTRUCTION.
:*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
:*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
:*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
:*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
:*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
:*THEN PORT4 WILL CONTAIN A 37
:*****
  
```

3116
 3117
 3118
 3119
 3120
 3121
 3122
 3123
 3124
 3125
 3126
 3127
 3128
 3129
 3130
 3131
 3132
 3133
 3134
 3135

```

: TEST 35
:-----
TST35. MOV #35,TSTNO
MOV #TST36,NEXT
MOV #1$,LOCK
MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
;MASTER CLEAR DMC11
BIT #BIT15,STAT1 ;IS IT CRAM?
BEQ 6$+2 ;SKIP TEST IF NO
JSR PC,MEMSET ;SET MEM AND RAM
1$: JSR PC,SETZ ;SET THE Z BIT'
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 ;START AT ROM PC=0
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
114377! <400*3> ;JUMP TO ROM PC OF 1777
JSR PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
377 ;EXPECTED DATA
CMPB R5,R4 ;IS ROM PC CORRECT?
BEQ 2$ ;BR IF YES
HLT 5 ;ERROR, CRAM PC IS WRONG
2$: SCOPI ;LOOP TO 1$ IF SW09=1
  
```

B01

3136	024276	012737	024304	001220		MOV	#3\$,LOCK	;NEW SCOPI
3137	024304				3\$:			
3138	024304	004737	035474			JSR	PC,SETZ	;SET THE Z BIT'
3139	024310	104414				ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3140	024312	100403				100403		;START AT ROM PC=3
3141	024314	104414				ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3142	024316	101400				100000!	<400*3>	;JUMP TO ROM PC OF 0
3143	024320	004737	035530			JSR	PC,RAMDAT	;R4=CRAM PC (LSB 8 BITS)
3144	024324	000000				0		;EXPECTED DATA
3145	024326	120504				CMPB	R5,R4	;IS ROM PC CORRECT?
3146	024330	001401				BEQ	4\$;BR IF YES
3147	024332	104005				HLT	5	;ERROR, CRAM PC IS WRONG
3148	024334	104401			4\$:	SCOPI		;LOOP TO 3\$ IF SW09=1
3149	024336	012737	024344	001220		MOV	#5\$,LOCK	;NEW SCOPI
3150	024344				5\$:			
3151	024344	004737	035474			JSR	PC,SETZ	;SET THE Z BIT'
3152	024350	104414				ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3153	024352	100406				100406		;START AT ROM PC=6
3154	024354	104414				ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3155	024356	105525				104125!	<400*3>	;JUMP TO ROM PC OF 525
3156	024360	004737	035530			JSR	PC,RAMDAT	;R4=CRAM PC (LSB 8 BITS)
3157	024364	000125				125		;EXPECTED DATA
3158	024366	120504				CMPB	R5,R4	;IS ROM PC CORRECT?
3159	024370	001401				BEQ	6\$;BR IF YES
3160	024372	104005				HLT	5	;ERROR, CRAM PC IS WRONG
3161	024374	104401			6\$:	SCOPI		;LOOP TO 5\$ IF SW59=1
3162	024376	104400				SCOPE		;SCOPE THIS TEST
3163								
3164								
3165								
3166								
3167								
3168								
3169								
3170								
3171								
3172								
3173								
3174								
3175								
3176								
3177								
3178	024400	012737	000036	001226	TST36:	MOV	#36,TSTNO	
3179	024406	012737	024574	001216		MOV	#TST37,NEXT	
3180	024414	012737	024440	001220		MOV	#1\$,LOCK	
3181								
3182	024422	104412				MSTCLR		;R1 CONTAINS BASE DMC11 ADDRESS
3183	024424	032737	100000	001366		BIT	#BIT15,STAT1	;MASTER CLEAR DMC11
3184	024432	001457				BEQ	6\$+2	;IS IT CRAM?
3185	024434	004737	035634			JSR	PC,MEMSET	;SKIP TEST IF NO
3186	024440				1\$:			;SET MEM AND RAM
3187	024440	004737	035426			JSR	PC,SETBRO	;SET THE BRO BIT'
3188	024444	104414				ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3189	024446	100400				100400		;START AT ROM PC=0
3190	024450	104414				ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3191	024452	116377				114377!	<400*4>	;JUMP TO ROM PC OF 1777

```

:***** TEST 36 *****
;*CRAM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
;*SET THE BRO BIT, PERFORM THE JUMP INSTRUCTION.
;*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
;*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
;*THEN PORT4 WILL CONTAIN A 37
:*****

```

: TEST 36

```

-----
TST36: MOV #36,TSTNO
      MOV #TST37,NEXT
      MOV #1$,LOCK

```

```

      MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
      BIT #BIT15,STAT1 ;MASTER CLEAR DMC11
      BEQ 6$+2 ;IS IT CRAM?
      JSR PC,MEMSET ;SKIP TEST IF NO
                    ;SET MEM AND RAM

```

```

1$: JSR PC,SETBRO ;SET THE BRO BIT'
    ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
    100400 ;START AT ROM PC=0
    ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
    114377!<400*4> ;JUMP TO ROM PC OF 1777

```

3192	024454	004737	035530		JSR	PC,RAMDAT	;R4=CRAM PC (LSB 8 BITS)
3193	024460	000377			377		;EXPECTED DATA
3194	024462	120504			CMPB	R5,R4	;IS ROM PC CORRECT?
3195	024464	001401			BEQ	2\$;BR IF YES
3196	024466	104005			HLT	5	;ERROR, CRAM PC IS WRONG
3197	024470	104401		2\$:	SCOP1		;LOOP TO 1\$ IF SW09=1
3198	024472	012737	024500	001220	MOV	#3\$,LOCK	;NEW SCOP1
3199	024500			3\$:			
3200	024500	004737	035426		JSR	PC,SETBRO	;SET THE BRO BIT'
3201	024504	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3202	024506	100403			100403		;START AT ROM PC=3
3203	024510	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3204	024512	1020C0			100000!	<400*4>	;JUMP TO ROM PC OF 0
3205	024514	004737	035530		JSR	PC,RAMDAT	;R4=CRAM PC (LSB 8 BITS)
3206	024520	000000			0		;EXPECTED DATA
3207	024522	120504			CMPB	R5,R4	;IS ROM PC CORRECT?
3208	024524	001401			BEQ	4\$;BR IF YES
3209	024526	104005			HLT	5	;ERROR, CRAM PC IS WRONG
3210	024530	104401		4\$:	SCOP1		;LOOP TO 3\$ IF SW09=1
3211	024532	012737	024540	001220	MOV	#5\$,LOCK	;NEW SCOP1
3212	024540			5\$:			
3213	024540	004737	035426		JSR	PC,SETBRO	;SET THE BRO BIT'
3214	024544	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3215	024546	100406			100406		;START AT ROM PC=6
3216	024550	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3217	024552	106125			104125!	<400*4>	;JUMP TO ROM PC OF 525
3218	024554	004737	035530		JSR	PC,RAMDAT	;R4=CRAM PC (LSB 8 BITS)
3219	024560	000125			125		;EXPECTED DATA
3220	024562	120504			CMPB	R5,R4	;IS ROM PC CORRECT?
3221	024564	001401			BEQ	6\$;BR IF YES
3222	024566	104005			HLT	5	;ERROR, CRAM PC IS WRONG
3223	024570	104401		6\$:	SCOP1		;LOOP TO 5\$ IF SW59=1
3224	024572	104400			SCOPE		;SCOPE THIS TEST

```

3225
3226
3227
3228 ;***** TEST 37 *****
3229 ;*CRAM TEST OF JUMP(I) ON BR1 SET MICRO-PROCESSOR INSTRUCTION.
3230 ;*SET THE BR1 BIT, PERFORM THE JUMP INSTRUCTION.
3231 ;*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
3232 ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
3233 ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3234 ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
3235 ;*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
3236 ;*THEN PORT4 WILL CONTAIN A 37
3237 ;*****

```

```

3238 ; TEST 37
3239 ;-----
3240 024574 012737 000037 001226 TST37: MOV #37,TSTNO
3241 024602 012737 024770 001216 MOV #TST40,NEXT
3242 024610 012737 024634 001220 MOV #1$,LOCK
3243
3244 024616 104412 MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
3245 024620 032737 100000 071366 BIT #BIT15,STAT1 ;MASTER CLEAR DMC11
3246 024626 001457 BEQ 6$+2 ;IS IT CRAM?
3247 024630 004737 035634 JSR PC,MEMSET ;SKIP TEST IF NO
;SET MEM AND RAM

```

D01

3248	024634				1\$:				
3249	024634	004737	035434			JSR	PC,SETBR1		;SET THE BR1 BIT'
3250	024640	104414				ROMCLK			;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3251	024642	100400				100400			;START AT ROM PC=0
3252	024644	104414				ROMCLK			;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3253	024646	116777				114377!	<400*5>		;JUMP TO ROM PC OF 1777
3254	024650	004737	035530			JSR	PC,RAMDAT		;R4=CRAM PC (LSB 8 BITS)
3255	024654	000377				377			;EXPECTED DATA
3256	024656	120504				CMPB	R5,R4		;IS ROM PC CORRECT?
3257	024660	001401				BEQ	2\$;BR IF YES
3258	024662	104005				HLT	5		;ERROR, CRAM PC IS WRONG
3259	024664	104401			2\$:	SCOPI			;LOOP TO 1\$ IF SW09=1
3260	024666	012737	024674	001220		MOV	#3\$,LOCK		;NEW SCOPI
3261	024674				3\$:				
3262	024674	004737	035434			JSR	PC,SETBR1		;SET THE BR1 BIT'
3263	024700	104414				ROMCLK			;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3264	024702	100403				100403			;START AT ROM PC=3
3265	024704	104414				ROMCLK			;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3266	024706	102400				100000!	<400*5>	: JUMP TO	ROM PC OF 0
3267	024710	004737	035530			JSR	PC,RAMDAT		;R4=CRAM PC (LSB 8 BITS)
3268	024714	000000				0			;EXPECTED DATA
3269	024716	120504				CMPB	R5,R4		;IS ROM PC CORRECT?
3270	024720	001401				BEQ	4\$;BR IF YES
3271	024722	104005				HLT	5		;ERROR, CRAM PC IS WRONG
3272	024724	104401			4\$:	SCOPI			;LOOP TO 3\$ IF SW09=1
3273	024726	012737	024734	001220		MOV	#5\$,LOCK		;NEW SCOPI
3274	024734				5\$:				
3275	024734	004737	035434			JSR	PC,SETBR1		;SET THE BR1 BIT'
3276	024740	104414				ROMCLK			;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3277	024742	100406				100406			;START AT ROM PC=6
3278	024744	104414				ROMCLK			;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3279	024746	106525				104125!	<400*5>		;JUMP TO ROM PC OF 525
3280	024750	004737	035530			JSR	PC,RAMDAT		;R4=CRAM PC (LSB 8 BITS)
3281	024754	000125				125			;EXPECTED DATA
3282	024756	120504				CMPB	R5,R4		;IS ROM PC CORRECT?
3283	024760	001401				BEQ	6\$;BR IF YES
3284	024762	104005				HLT	5		;ERROR, CRAM PC IS WRONG
3285	024764	104401			6\$:	SCOPI			;LOOP TO 5\$ IF SW59=1
3286	024766	104400				SCOPE			;SCOPE THIS TEST

3287
 3288
 3289
 3290
 3291
 3292
 3293
 3294
 3295
 3296
 3297
 3298
 3299
 3300
 3301

```

;***** TEST 40 *****
;CRAM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
;SET THE BR4 BIT, PERFORM THE JUMP INSTRUCTION.
;VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
;IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
;BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
;THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
;THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
;THEN PORT4 WILL CONTAIN A 37
;*****

```

```

; TEST 40
;-----
TST40: MOV #40,TSTNO
MOV #TST41,NEXT

```

3302 024770 012737 000040 001226
 3303 024776 012737 025164 001216

E01

```

3304 025004 012737 025030 001220      MOV      #1$,LOCK
3305                                     ;R1 CONTAINS BASE DMC11 ADDRESS
3306 025012 104412                                     ;MASTER CLEAR DMC11
3307 025014 032737 100000 001366      MSTCLR
3308 025022 001457                                     ;IS IT CRAM?
3309 025024 004737 035634      BEQ      6$+2
3310 025030                                     ;SKIP TEST IF NO
3311 025030 004737 035442      JSR      PC,MEMSET
3312 025034 104414                                     ;SET THE BR4 BIT'
3313 025036 100400                                     ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3314 025040 104414                                     ;START AT ROM PC=0
3315 025042 117377                                     ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3316 025044 004737 035530      JSR      PC,SETBR4
3317 025044 004737 035530      JSR      PC,RAMDAT
3318 025044 120504      377
3319 025044 120504      CMPB     R5,R4
3320 025054 001401      BEQ      2$
3321 025056 104005      HLT      5
3322 025060 104401      SCOPE1
3323 025062 012737 025070 001220      MOV      #3$,LOCK
3324 025070 004737 035442      JSR      PC,SETBR4
3325 025074 104414                                     ;SET THE BR4 BIT'
3326 025076 100403                                     ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3327 025100 104414                                     ;START AT ROM PC=3
3328 025102 103000                                     ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3329 025104 004737 035530      JSR      PC,RAMDAT
3330 025110 000000      0
3331 025112 120504      CMPB     R5,R4
3332 025114 001401      BEQ      4$
3333 025116 104005      HLT      5
3334 025120 104401      SCOPE1
3335 025122 012737 025130 001220      MOV      #5$,LOCK
3336 025130 004737 035442      JSR      PC,SETBR4
3337 025134 104414                                     ;SET THE BR4 BIT'
3338 025136 100406                                     ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3339 025140 104414                                     ;START AT ROM PC=6
3340 025142 107125                                     ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3341 025144 004737 035530      JSR      PC,RAMDAT
3342 025150 000125      125
3343 025152 120504      CMPB     R5,R4
3344 025154 001401      BEQ      6$
3345 025156 104005      HLT      5
3346 025160 104401      SCOPE1
3347 025162 104400      SCOPE
  
```

```

3348
3349
3350
3351 ;***** TEST 41 *****
3352 ;*CRAM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
3353 ;*SET THE BR7 BIT, PERFORM THE JUMP INSTRUCTION.
3354 ;*VERIFY THE JUMP DID OCCUR BY CLOCKING THE INSTRUCTION
3355 ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
3356 ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3357 ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT,
3358 ;*THE JUMP WAS SUCCESSFUL, IF THE JUMP WAS UNSUCCESSFUL
3359 ;*THEN PORT4 WILL CONTAIN A 37
  
```

F01

```

3360 ;:*****
3361 ;
3362 ; TEST 41
3363 ;-----
3364 025164 012737 000041 001226 TST41: MOV #41,TSTNO
3365 025172 012737 025360 001216 MOV #TST42,NEXT
3366 025200 012737 025224 001220 MOV #1$,LOCK
3367 ;R1 CONTAINS BASE DMC11 ADDRESS
3368 025206 104412 MSTCLR ;MASTER CLEAR DMC11
3369 025210 032737 100000 001366 BIT #BIT15,STAT1 ;IS IT CRAM?
3370 025216 001457 BEQ 6$+2 ;SKIP TEST IF NO
3371 025220 004737 035634 JSR PC,MEMSET ;SET MEM AND RAM
3372 025224 1$:
3373 025224 004737 035450 JSR PC,SETBR7 ;SET THE BR7 BIT'
3374 025230 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3375 025232 100400 100400 ;START AT ROM PC=0
3376 025234 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3377 025236 117777 114377! <400*7> ;JUMP TO ROM PC OF 1777
3378 025240 004737 035530 JSR PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3379 025244 000377 377 ;EXPECTED DATA
3380 025246 120504 CMPB R5,R4 ;IS ROM PC CORRECT?
3381 025250 001401 BEQ 2$ ;BR IF YES
3382 025252 104005 HLT 5 ;ERROR, CRAM PC IS WRONG
3383 025254 104401 2$: SCOP1 ;LOOP TO 1$ IF SW09=1
3384 025256 012737 025264 001220 MOV #3$,LOCK ;NEW SCOP1
3385 025264 3$:
3386 025264 004737 035450 JSR PC,SETBR7 ;SET THE BR7 BIT'
3387 025270 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3388 025272 100403 100403 ;START AT ROM PC=3
3389 025274 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3390 025276 103400 100000! <400*7> ;JUMP TO ROM PC OF 0
3391 025300 004737 035530 JSR PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3392 025304 000000 0 ;EXPECTED DATA
3393 025306 120504 CMPB R5,R4 ;IS ROM PC CORRECT?
3394 025310 001401 BEQ 4$ ;BR IF YES
3395 025312 104005 HLT 5 ;ERROR, CRAM PC IS WRONG
3396 025314 104401 4$: SCOP1 ;LOOP TO 3$ IF SW09=1
3397 025316 012737 025324 001220 MOV #5$,LOCK ;NEW SCOP1
3398 025324 5$:
3399 025324 004737 035450 JSR PC,SETBR7 ;SET THE BR7 BIT'
3400 025330 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3401 025332 100406 100406 ;START AT ROM PC=6
3402 025334 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3403 025336 107525 104125! <400*7> ;JUMP TO ROM PC OF 525
3404 025340 004737 035530 JSR PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3405 025344 000125 125 ;EXPECTED DATA
3406 025346 120504 CMPB R5,R4 ;IS ROM PC CORRECT?
3407 025350 001401 BEQ 6$ ;BR IF YES
3408 025352 104005 HLT 5 ;ERROR, CRAM PC IS WRONG
3409 025354 104401 6$: SCOP1 ;LOOP TO 5$ IF SW59=1
3410 025356 104400 SCOPE ;SCOPE THIS TEST
3411 ;:*****
3412 ;
3413 ;***** TEST 42 *****
3414 ;*CRAM TEST OF JUMP(I) ON C BIT SET MICRO-PROCESSOR INSTRUCTION.
3415 ;*CLEAR THE C BIT, PERFORM THE JUMP INSTRUCTION,

```

```

3416 ;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
3417 ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
3418 ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3419 ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
3420 ;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT.
3421 ;*THEN PORT4 CONTAINS A 37
3422 ;*****
3423
3424 ; TEST 42
3425 ;-----
3426 025360 012737 000042 001226 TST42: MOV #42,TSTNO
3427 025366 012737 025554 001216 MOV #TST43,NEXT
3428 025374 012737 025420 001220 MOV #1$,LOCK
3429 ;R1 CONTAINS BASE DMC11 ADDRESS
3430 025402 104412 MSTCLR ;MASTER CLEAR DMC11
3431 025404 032737 100000 001366 BIT #BIT15,STAT1 ;IS IT CRAM?
3432 025412 001457 BEQ 6$+2 ;SKIP TEST IF NO
3433 025414 004737 035634 JSR PC,MEMSET ;SET MEM AND RAM
3434 025420
3435 025420 004737 035410 1$: JSR PC,CLRALL ;CLEAR ALL CONDITIONS
3436 025424 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3437 025426 100400 100400 ;START AT ROM PC=0
3438 025430 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3439 025432 115377 114377! <400*2> ;JUMP TO ROM PC OF 1777
3440 025434 004737 035530 JSR PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3441 025440 000001 1 ;EXPECTED DATA
3442 025442 120504 CMPB R5,R4 ;IS ROM PC CORRECT?
3443 025444 001401 BEQ 2$ ;BR IF YES
3444 025446 104005 HLT 5 ;ERROR, CRAM PC IS WRONG
3445 025450 104401 2$: SCOP1 ;LOOP TO 1$ IF SW09=1
3446 025452 012737 025460 001220 MOV #3$,LOCK ;NEW SCOP1
3447 025460
3448 025460 004737 035410 3$: JSR PC,CLRALL ;CLEAR ALL CONDITIONS
3449 025464 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3450 025466 100403 100403 ;START AT ROM PC=3
3451 025470 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3452 025472 101000 100000! <400*2> ;JUMP TO ROM PC OF 0
3453 025474 004737 035530 JSR PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3454 025500 000004 4 ;EXPECTED DATA
3455 025502 120504 CMPB R5,R4 ;IS ROM PC CORRECT?
3456 025504 001401 BEQ 4$ ;BR IF YES
3457 025506 104005 HLT 5 ;ERROR, CRAM PC IS WRONG
3458 025510 104401 4$: SCOP1 ;LOOP TO 3$ IF SW09=1
3459 025512 012737 025520 001220 MOV #5$,LOCK ;NEW SCOP1
3460 025520
3461 025520 004737 035410 5$: JSR PC,CLRALL ;CLEAR ALL CONDITIONS
3462 025524 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3463 025526 100406 100406 ;START AT ROM PC=6
3464 025530 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3465 025532 105125 104125! <400*2> ;JUMP TO ROM PC OF 525
3466 025534 004737 035530 JSR PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3467 025540 000007 7 ;EXPECTED DATA
3468 025542 120504 CMPB R5,R4 ;IS ROM PC CORRECT?
3469 025544 001401 BEQ 6$ ;BR IF YES
3470 025546 104005 HLT 5 ;ERROR, CRAM PC IS WRONG
3471 025550 104401 6$: SCOP1 ;LOOP TO 5$ IF SW59=1
    
```

HO1

```

3472 025552 104400          SCOPE          ;SCOPE THIS TEST
3473
3474
3475          ;***** TEST 43 *****
3476          ;*CRAM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
3477          ;*CLEAR THE Z BIT, PERFORM THE JUMP INSTRUCTION,
3478          ;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
3479          ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
3480          ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3481          ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
3482          ;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
3483          ;*THEN PORT4 CONTAINS A 37
3484          ;*****
3485
3486          ; TEST 43
3487          ;-----
3488 025554 012737 000043 001226      TST43: MOV      #43,TSTNO
3489 025562 012737 025750 001216      MOV      #TST44,NEXT
3490 025570 012737 025614 001220      MOV      #1$,LOCK
3491
3492 025576 104412          MSTCLR          ;R1 CONTAINS BASE DMC11 ADDRESS
3493 025600 032737 100000 001366      BIT      #BIT15,STAT1 ;MASTER CLEAR DMC11
3494 025606 001457          BEQ          ;IS IT CRAM?
3495 025610 004737 035634          JSR      PC, MEMSET ;SKIP TEST IF NO
3496 025614          ;SET MEM AND RAM
3497 025614 004737 035410      1$: JSR      PC, CLRALL ;CLEAR ALL CONDITIONS
3498 025620 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3499 025622 100400          100400        ;START AT ROM PC=0
3500 025624 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3501 025626 115777          114377! <400*3> ;JUMP TO ROM PC OF 1777
3502 025630 004737 035530      JSR      PC, RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3503 025634 000001          1             ;EXPECTED DATA
3504 025636 120504          CMPB     R5,R4    ;IS ROM PC CORRECT?
3505 025640 001401          BEQ      2$      ;BR IF YES
3506 025642 104005          HLT      5       ;ERROR, CRAM PC IS WRONG
3507 025644 104401          2$: SCOP1          ;LOOP TO 1$ IF SW09=1
3508 025646 012737 025654 001220      MOV      #3$,LOCK ;NEW SCOP1
3509 025654          3$:
3510 025654 004737 035410      JSR      PC, CLRALL ;CLEAR ALL CONDITIONS
3511 025660 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3512 025662 100403          100403        ;START AT ROM PC=3
3513 025664 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3514 025666 101400          100000! <400*3> ;JUMP TO ROM PC OF 0
3515 025670 004737 035530      JSR      PC, RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3516 025674 000004          4             ;EXPECTED DATA
3517 025676 120504          CMPB     R5,R4    ;IS ROM PC CORRECT?
3518 025700 001401          BEQ      4$      ;BR IF YES
3519 025702 104005          HLT      5       ;ERROR, CRAM PC IS WRONG
3520 025704 104401          4$: SCOP1          ;LOOP TO 3$ IF SW09=1
3521 025706 012737 025714 001220      MOV      #5$,LOCK ;NEW SCOP1
3522 025714          5$:
3523 025714 004737 035410      JSR      PC, CLRALL ;CLEAR ALL CONDITIONS
3524 025720 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3525 025722 100406          100406        ;START AT ROM PC=6
3526 025724 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3527 025726 105525          104125! <400*3> ;JUMP TO ROM PC OF 525
  
```

```

3528 025730 004737 035530 JSR PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3529 025734 000007 7 ;EXPECTED DATA
3530 025736 120504 CMPB R5,R4 ;IS ROM PC CORRECT?
3531 025740 001401 BEQ 6$ ;BR IF YES
3532 025742 104005 HLT 5 ;ERROR, CRAM PC IS WRONG
3533 025744 104401 6$: SCOP1 ;LOOP TO 5$ IF SW59=1
3534 025746 104400 SCOPE ;SCOPE THIS TEST
3535
3536
3537 ;***** TEST 44 *****
3538 ;*CRAM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
3539 ;*CLEAR THE BRO BIT, PERFORM THE JUMP INSTRUCTION,
3540 ;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
3541 ;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
3542 ;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
3543 ;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
3544 ;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
3545 ;*THEN PORT4 CONTAINS A 37
3546 ;*****
3547
3548 ; TEST 44
3549 ;-----
3550 025750 012737 000044 001226 TST44: MOV #44,TSTNO
3551 025756 012737 026144 001216 MOV #TST45,NEXT
3552 025764 012737 026010 001220 MOV #15,LOCK
3553 ;R1 CONTAINS BASE DMC11 ADDRESS
3554 025772 104412 MSTCLR ;MASTER CLEAR DMC11
3555 025774 032737 100000 001366 BIT #BIT15,STAT1 ;IS IT CRAM?
3556 026002 001457 BEQ 6$+2 ;SKIP TEST IF NO
3557 026004 004737 035634 JSR PC,MEMSET ;SET MEM AND RAM
3558 026010 1$:
3559 026010 004737 035410 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
3560 026014 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3561 026016 100400 100400 ;START AT ROM PC=0
3562 026020 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3563 026022 116377 114377! <400*4> ;JUMP TO ROM PC OF 1777
3564 026024 004737 035530 JSR PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3565 026030 000001 1 ;EXPECTED DATA
3566 026032 120504 CMPB R5,R4 ;IS ROM PC CORRECT?
3567 026034 001401 BEQ 2$ ;BR IF YES
3568 026036 104005 HLT 5 ;ERROR, CRAM PC IS WRONG
3569 026040 104401 2$: SCOP1 ;LOOP TO 1$ IF SW09=1
3570 026042 012737 026050 001220 MOV #3$,LOCK ;NEW SCOP1
3571 026050 3$:
3572 026050 004737 035410 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
3573 026054 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3574 026056 100403 100403 ;START AT ROM PC=3
3575 026060 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3576 026062 102000 100000! <400*4> ;JUMP TO ROM PC OF 0
3577 026064 004737 035530 JSR PC,RAMDAT ;R4=CRAM PC (LSB 8 BITS)
3578 026070 000004 4 ;EXPECTED DATA
3579 026072 120504 CMPB R5,R4 ;IS ROM PC CORRECT?
3580 026074 001401 BEQ 4$ ;BR IF YES
3581 026076 104005 HLT 5 ;ERROR, CRAM PC IS WRONG
3582 026100 104401 4$: SCOP1 ;LOOP TO 3$ IF SW09=1
3583 026102 012737 026110 001220 MOV #5$,LOCK ;NEW SCOP1

```

```

3584 026110          5$: JSR    PC,CLRALL      ;CLEAR ALL CONDITIONS
3585 026110 004737 035410 ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3586 026114 104414      100406      ;START AT ROM PC=6
3587 026116 100406      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3588 026120 104414      104125! <400*4> ;JUMP TO ROM PC OF 525
3589 026122 106125      JSR    PC,RAMDAT    ;R4=CRAM PC (LSB 8 BITS)
3590 026124 004737 035530 7          ;EXPECTED DATA
3591 026130 000007      CMPB   R5,R4      ;IS ROM PC CORRECT?
3592 026132 120504      BEQ    6$          ;BR IF YES
3593 026134 001401      HLT    5          ;ERROR, CRAM PC IS WRONG
3594 026136 104005      6$: SCOP1      ;LOOP TO 5$ IF SW59=1
3595 026140 104401      SCOPE     ;SCOPE THIS TEST
3596 026142 104400
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611

```

```

:***** TEST 45 *****
: *CRAM TEST OF JUMP(I) ON BRI SET MICRO-PROCESSOR INSTRUCTION.
: *CLEAR THE BRI BIT, PERFORM THE JUMP INSTRUCTION,
: *VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
: *IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
: *BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
: *THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
: *THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
: *THEN PORT4 CONTAINS A 37
:*****

```

: TEST 45

```

3612 026144 012737 300045 001226 TST45: MOV    #45,TSTNO
3613 026152 012737 026343 001216      MOV    #TST46,NEXT
3614 026160 012737 026204 001220      MOV    #1$,LOCK
3615
3616 026166 104412      MSTCLR      ;R1 CONTAINS BASE DMC11 ADDRESS
3617 026170 032737 100000 001366 BIT    #BIT15,STAT1 ;MASTER CLEAR DMC11
3618 026176 001457      BEQ    6$+2      ;IS IT CRAM?
3619 026200 004737 035634      JSR    PC,MEMSET ;SKIP TEST IF NO
3620 026204          ;SET MEM AND RAM
3621 026204 004737 035410 1$: JSR    PC,CLRALL      ;CLEAR ALL CONDITIONS
3622 026210 104414      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3623 026212 100400      100400      ;START AT ROM PC=0
3624 026214 104414      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3625 026216 116777      114377! <400*5> ;JUMP TO ROM PC OF 1777
3626 026220 004737 035530 JSR    PC,RAMDAT    ;R4=CRAM PC (LSB 8 BITS)
3627 026224 000001      1          ;EXPECTED DATA
3628 026226 120504      CMPB   R5,R4      ;IS ROM PC CORRECT?
3629 026230 001401      BEQ    2$          ;BR IF YES
3630 026232 104005      HLT    5          ;ERROR, CRAM PC IS WRONG
3631 026234 104401      2$: SCOP1      ;LOOP TO 1$ IF SWC9=1
3632 026236 012737 026244 001220 MOV    #3$,LOCK    ;NEW SCOPE
3633 026244          3$:
3634 026244 004737 035410 JSR    PC,CLRALL      ;CLEAR ALL CONDITIONS
3635 026250 104414      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3636 026252 100403      100403      ;START AT ROM PC=3
3637 026254 104414      ROMCLK      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3638 026256 102400      100000! <400*5> ;JUMP TO ROM PC OF 0
3639 026260 004737 035530 JSR    PC,RAMDAT    ;R4=CRAM PC (LSB 8 BITS)

```

```

3640 026264 000004          4          ; EXPECTED DATA
3641 026266 120504          CMPB      R5,R4          ; IS ROM PC CORRECT?
3642 026270 001401          BEQ       4$            ; BR IF YES
3643 026272 104005          HLT      5             ; ERROR, CRAM PC IS WRONG
3644 026274 104401          SCOPI    ; LOOP TO 3$ IF SW09=1
3645 026276 012737 026304 001220 4$:      MOV      #5$,LOCK      ; NEW SCOPI
3646 026304          5$:
3647 026304 004737 035410          JSR      PC,CLRALL      ; CLEAR ALL CONDITIONS
3648 026310 104414          ROMCLK   ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3649 026312 100406          100406   ; START AT ROM PC=6
3650 026314 104414          ROMCLK   ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3651 026316 106525          104125! <400*5>      ; JUMP TO ROM PC OF 525
3652 026320 004737 035530          JSR      PC,RAMDAT      ; R4=CRAM PC (LSB 8 BITS)
3653 026324 000007          7          ; EXPECTED DATA
3654 026326 120504          CMPB      R5,R4          ; IS ROM PC CORRECT?
3655 026330 001401          BEQ       6$            ; BR IF YES
3656 026332 104005          HLT      5             ; ERROR, CRAM PC IS WRONG
3657 026334 104401          SCOPI    ; LOOP TO 5$ IF SW59=1
3658 026336 104400          SCOPE    ; SCOPE THIS TEST
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673

```

```

;***** TEST 46 *****
;*CRAM TEST OF JUMP(I) ON BR4 SET MICRO-PROCESSOR INSTRUCTION.
;*CLEAR THE BR4 BIT, PERFORM THE JUMP INSTRUCTION,
;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
;*THEN PORT4 CONTAINS A 37
;*****

```

TEST 46

```

3674 026340 012737 000046 001226  TST46:  MOV      #46,TSTNO
3675 026346 012737 026534 001216      MOV      #TST47,NEXT
3676 026354 012737 026400 001220      MOV      #1$,LOCK
3677
3678 026362 104412          MSTCLR   ; R1 CONTAINS BASE DMC11 ADDRESS
3679 026364 032737 100000 001366      BIT      #BIT15,STAT1  ; MASTER CLEAR DMC11
3680 026372 001457          BEQ      6$+2          ; IS IT CRAM?
3681 026374 004737 035634          JSR      PC,MEMSET     ; SKIP TEST IF NO
3682 026400          1$:      ; SET MEM AND RAM
3683 026400 004737 035410          JSR      PC,CLRALL      ; CLEAR ALL CONDITIONS
3684 026404 104414          ROMCLK   ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3685 026406 100400          100400   ; START AT ROM PC=0
3686 026410 104414          ROMCLK   ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3687 026412 117377          114377! <400*6>      ; JUMP TO ROM PC OF 1777
3688 026414 004737 035530          JSR      PC,RAMDAT      ; R4=CRAM PC (LSB 8 BITS)
3689 026420 000001          1          ; EXPECTED DATA
3690 026422 120504          CMPB      R5,R4          ; IS ROM PC CORRECT?
3691 026424 001401          BEQ      2$            ; BR IF YES
3692 026426 104005          HLT      5             ; ERROR, CRAM PC IS WRONG
3693 026430 104401          SCOPI    ; LOOP TO 1$ IF SW09=1
3694 026432 012737 026440 001220 2$:      MOV      #3$,LOCK      ; NEW SCOPI
3695 026440          3$:

```

L01

3696	026440	004737	035410		JSR	PC,CLRALL	;CLEAR ALL CONDITIONS
3697	026444	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3698	026446	100403			100403		;START AT ROM PC=3
3699	026450	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3700	026452	103000			100000!	<400*6>	;JUMP TO ROM PC OF 0
3701	026454	004737	035530		JSR	PC,RAMDAT	;R4=CRAM PC (LSB 8 BITS)
3702	026460	000004			4		;EXPECTED DATA
3703	026462	120504			CMPE	RE,R4	;IS ROM PC CORRECT?
3704	026464	001401			BEQ	4\$;BR IF YES
3705	026466	104005			HLT	5	;ERROR, CRAM PC IS WRONG
3706	026470	104401		4\$:	SCOPI		;LOOP TO 3\$ IF SW09=1
3707	026472	012737	026500	001220	MOV	#5\$,LOCK	;NEW SCOPI
3708	026500			5\$:			
3709	026500	004737	035410		JSR	PC,CLRALL	;CLEAR ALL CONDITIONS
3710	026504	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3711	026506	100406			100406		;START AT ROM PC=6
3712	026510	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3713	026512	107125			104125!	<400*6>	;JUMP TO ROM PC OF 525
3714	026514	004737	035530		JSR	PC,RAMDAT	;R4=CRAM PC (LSB 8 BITS)
3715	026520	000007			7		;EXPECTED DATA
3716	026522	120504			CMPE	R5,R4	;IS ROM PC CORRECT?
3717	026524	001401			BEQ	6\$;BR IF YES
3718	026526	104005			HLT	5	;ERROR, CRAM PC IS WRONG
3719	026530	104401		6\$:	SCOPI		;LOOP TO 5\$ IF SW59=1
3720	026532	104400			SCOPE		;SCOPE THIS TEST
3721							
3722							
3723							
3724							
3725							
3726							
3727							
3728							
3729							
3730							
3731							
3732							
3733							
3734							
3735							
3736	026534	012737	000047	001226	TST47:	MOV #47,ISTNO	
3737	026542	012737	026730	001216		MOV #TST50,NEXT	
3738	026550	012737	026574	001220		MOV #1\$,LOCK	
3739							
3740	026556	104412			MSTCLR		;R1 CONTAINS BASE DMC11 ADDRESS
3741	026560	032737	100000	001366	BIT	#BIT15,STAT1	;MASTER CLEAR DMC11
3742	026566	001457			BEQ	6\$+2	;IS IT CRAM?
3743	026570	004737	035634		JSR	PC,MEMSET	;SKIP TEST IF NO
3744	026574						;SET MEM AND RAM
3745	026574	004737	035410		1\$:	JSR	PC,CLRALL
3746	026600	104414			ROMCLK		;CLEAR ALL CONDITIONS
3747	026602	100400			100400		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3748	026604	104414			ROMCLK		;START AT ROM PC=0
3749	026606	117777			114377!	<400*7>	;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3750	026610	004737	035530		JSR	PC,RAMDAT	;JUMP TO ROM PC OF 1777
3751	026614	000001			1		;R4=CRAM PC (LSB 8 BITS)
							;EXPECTED DATA

```

;***** TEST 47 *****
;*CRAM TEST OF JUMP(I) ON BR7 SET MICRO-PROCESSOR INSTRUCTION.
;*CLEAR THE BR7 BIT, PERFORM THE JUMP INSTRUCTION,
;*VERIFY THE JUMP DID NOT OCCUR BY CLOCKING THE INSTRUCTION
;*IN THE LOCATION IT IS AT. THIS INSTRUCTION LOADS THE
;*BR WITH THE LOWEST 8 BITS OF THE CRAM PC. AT THIS POINT
;*THE BR DATA IS MOVED TO PORT4. IF THIS DATA IS CORRECT
;*THE CRAM PC IS CORRECT, IF THE CRAM PC IS NOT RIGHT,
;*THEN PORT4 CONTAINS A 37
;:*****

```

```

; TEST 47
;-----
TST47: MOV #47,ISTNO
        MOV #TST50,NEXT
        MOV #1$,LOCK

MSTCLR
BIT #BIT15,STAT1
BEQ 6$+2
JSR PC,MEMSET

1$: JSR PC,CLRALL
    ROMCLK
    100400
    ROMCLK
    114377! <400*7>
    JSR PC,RAMDAT
    1

```


MO1

3752	026616	120504				CMPB	R5,R4		; IS ROM PC CORRECT?
3753	026620	001401				BEQ	2\$; BR IF YES
3754	026622	104005				HLT	5		; ERROR, CRAM PC IS WRONG
3755	026624	104401			2\$:	SCOP1			; LOOP TO 1\$ IF SW09=1
3756	026626	012737	026634	001220		MOV	#3\$,LOCK		; NEW SCOP1
3757	026634				3\$:				
3758	026634	004737	035410			JSR	PC,CLRALL		; CLEAR ALL CONDITIONS
3759	026640	104414				ROMCLK			; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3760	026642	100403				100403			; START AT ROM PC=3
3761	026644	104414				ROMCLK			; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3762	026646	103400				100000!	<400*7>	JUMP TO	ROM PC OF 0
3763	026650	004737	035530			JSR	PC,RAMDAT		; R4=CRAM PC (LSB 8 BITS)
3764	026654	000004				4			; EXPECTED DATA
3765	026656	120504				CMPB	R5,R4		; IS ROM PC CORRECT?
3766	026660	001401				BEQ	4\$; BR IF YES
3767	026662	104005				HLT	5		; ERROR, CRAM PC IS WRONG
3768	026664	104401			4\$:	SCOP1			; LOOP TO 3\$ IF SW09=1
3769	026666	012737	026674	001220		MOV	#5\$,LOCK		; NEW SCOP1
3770	026674				5\$:				
3771	026674	004737	035410			JSR	PC,CLRALL		; CLEAR ALL CONDITIONS
3772	026700	104414				ROMCLK			; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3773	026702	100406				100406			; START AT ROM PC=6
3774	026704	104414				ROMCLK			; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
3775	026706	107525				104125!	<400*7>	JUMP TO	ROM PC OF 525
3776	026710	004737	035530			JSR	PC,RAMDAT		; R4=CRAM PC (LSB 8 BITS)
3777	026714	000007				7			; EXPECTED DATA
3778	026716	120504				CMPB	R5,R4		; IS ROM PC CORRECT?
3779	026720	001401				BEQ	6\$; BR IF YES
3780	026722	104005				HLT	5		; ERROR, CRAM PC IS WRONG
3781	026724	104401			6\$:	SCOP1			; LOOP TO 5\$ IF SW59=1
3782	026726	104400				SCOPE			; SCOPE THIS TEST

```

;***** TEST 50 *****
;FREE RUNNING FLAG MODE DATA TEST
;TRANSMIT A MESSAGE AND VERIFY THE RECEIVED DATA
;IF NO TURNAROUND CONNECTOR IS ON LINE UNIT LOOP IS SET.
;ALL FOLLOWING TESTS ARE FREE RUNNING AND ARE PERFORMED
;ONLY ON DMC'S WITH LINE UNITS. IF YOU WISH TO PERFORM
;THESE FREE RUNNING TESTS ON A KMC (NORMALLY THE FREE RUNNING TESTS
;WILL FAIL ON A KMC, THE TIMER IS TOO FAST) THEN YOU MUST
;MANUALLY SET BIT0 OF STAT3 IN THE STATUS MAP.
;*****

```

TEST 50

3798	026730	012737	000050	001226	TST50:	MOV	#50,TSTNO		
3799	026736	012737	027742	001216		MOV	#TST51,NEXT		
3800									; R1 CONTAINS BASE DMC11 ADDRESS
3801	026744	104412				MSTCLR			; MASTER CLEAR DMC11
3802	026746	032737	100000	001366		BIT	#BIT15,STAT1		; IS IT A DMC?
3803	026754	001406				BEQ	+.16		; BR IF YES
3804	026756	032737	000001	001372		BIT	#BIT0,STAT3		; KMC WITH BIT0 SET?
3805	026764	001002				BNE	+.6		; BR IF YES
3806	026766	000137	027740			JMP	14\$; SKIP TEST
3807	026772	032737	010000	001366		BIT	#BIT12,STAT1		; LU PRESENT?

3808	027000	001372			BNE	.-12	;BR IF NO
3809	027002	004737	035562		JSR	PC.WROM	;WRITE MAP IN CRAM
3810	027006	013700	034740		MOV	RCOUNT,R0	;CLEAR RECEIVER BUFFER
3811	027012	062700	000002		ADD	#2,R0	;CLEAR 2 MORE LOCATIONS
3812	027016	012702	034742		MOV	#RBUF,R2	;CLEAR OUT RECEIVE BUFFER
3813	027022	105022		10\$:	CLRB	(R2)+	;CLEAR BUFFER
3814	027024	005300			DEC	R0	;DONE YET!
3815	027026	001375			BNE	10\$;NO
3816	027030	005037	034666		CLR	TFLAG	;SET TFLAG TO 0
3817	027034	005037	034670		CLR	RFLAG	;SET RFLAG TO 0
3818	027040	012711	040000		MOV	#BIT14,(R1)	;MASTER CLEAR
3819	027044	032737	100000	001366	BIT	#BIT15,STAT1	;CRAM?
3820	027052	001402			BEQ	.+6	;BR IF NO
3821	027054	012711	100000		MOV	#BIT15,(R1)	;IF CRAM SET RUN
3822	027060	105227	000000		INCB	#0	;DELAY
3823	027064	001375			BNE	.-4	;DELAY
3824	027066	005037	001416		CLR	TEMP	;GET SET TO DELAY
3825	027072	005711		1\$:	TST	(R1)	;RUN SET?
3826	027074	100405			BMI	.+14	;BR IF YES
3827	027076	005237	001416		INC	TEMP	;INC DELAY
3828	027102	001373			BNE	1\$;BR IF NOT DONE
3829	027104	104014			HLT	14	;ERROR RUN NOT SET
3830	027106	000771			BR	1\$;TRY AGAIN
3831	027110	032737	040000	001366	BIT	#BIT14,STAT1	;TURNAROUND CONNECTOR?
3832	027116	001002			BNE	.+6	;BR IF YES
3833	027120	052711	004000		BIS	#BIT11,(R1)	;SET LINE UNIT LOOP
3834	027124	152711	000043		BISB#43,(R1)		;BASE I
3835	027130	005037	001416		CLR	TEMP	;GET SET TO DELAY
3836	027134	105711		2\$:	TSTB	(R1)	;RDI SET?
3837	027136	100404			BMI	.+12	;BR IF YES
3838	027140	005237	001416		INC	TEMP	;INC DELAY
3839	027144	001373			BNE	2\$;BR IF NOT DONE
3840	027146	104014			HLT	14	;ERROR RDI NOT SET
3841	027150	012761	035010	000004	MOV	#BASE,4(R1)	;SET UP BASE ADDRESS
3842	027156	005061	000006		CLR	6(R1)	;CLEAR COUNT
3843	027162	142711	000040		BICB	#40,(R1)	;CLEAR RQI
3844	027166	005037	001416		CLR	TEMP	;GET SET TO DELAY
3845	027172	105711		3\$:	TSTB	(R1)	;IS RDI GONE?
3846	027174	100020			BPL	8\$;BR IF YES
3847	027176	005237	001416		INC	TEMP	;INC DELAY
3848	027202	001373			BNE	3\$;BR IF NOT DONE
3849	027204	105761	000002		TSTB	2(R1)	;IS THERE A CNTL 0 ERROR
3850	027210	100011			BPL	18\$;BR IF NO
3851	027212	016137	000004	001252	MOV	4(R1),TEMP3	;SAVE SEL4 FOR TYPEOUT
3852	027220	016137	000006	001254	MOV	6(R1),TEMP4	;SAVE SEL6 FOR TYPEOUT
3853	027226	104016			HLT	16	;CNTL 0 ERROR
3854	027230	000137	027740		JMP	14\$;FATAL ERROR STOP
3855	027234	104014		18\$:	HLT	14	;ERROR RDI STILL SET
3856	027236			8\$:			
3857	027236	152711	000041		BISB	#41,(R1)	;ASK FOR CNTL I
3858	027242	105711		64\$:	TSTB	(R1)	;WAIT FOR RDI
3859	027244	100376			BPL	64\$;BR IF NOT SETY
3860	027246	005061	000006		CLR	6(R1)	;SET FULL DUPLEX
3861	027252	142711	000040		BICB	#40,(R1)	;CLEAR RQI
3862	027256	105711		65\$:	TSTB	(R1)	;RDI UP?
3863	027260	100776			BMI	65\$;BR IF YES

3864	027262	152711	000044		BISB	#44, (R1)	: REC BA/CC
3865	027266	005037	001416		CLR	TEMP	: GET SET TO DELAY
3866	027272	105711		4\$:	TSTB	(R1)	: IS RDI SET?
3867	027274	100404			BMI	.+12	: BR IF YES
3868	027276	005237	001416		INC	TEMP	: INC DELAY
3869	027302	001373			BNE	4\$: BR IF DELAY NOT DONE
3870	027304	104014			HLT	14	: ERROR RDI NOT SET
3871	027306	012761	034742	000004	MOV	#RBUF, 4(R1)	: LOAD REC BP
3872	027314	013761	034740	000006	MOV	RCOUNT, 6(R1)	: LOAD REC COUNT
3873	027322	142711	000040		BICB	#40, (R1)	: CLEAR RQI
3874	027326	005037	001416		CLR	TEMP	: GET SET TO DELAY
3875	027332	105711		5\$:	TSTB	(R1)	: RDI GONE?
3876	027334	100004			BPL	.+12	: BR IF YES
3877	027336	005237	001416		INC	TEMP	: INC DELAY
3878	027342	001373			BNE	5\$: BR IF NOT DONE
3879	027344	104014			HLT	14	: ERROR RDI STILL SET
3880	027346	152711	000040		BISB	#40, (R1)	: XMIT BA/CC
3881	027352	005037	001416		CLR	TEMP	: GET SET TO DELAY
3882	027356	105711		6\$:	TSTB	(R1)	: RDI SET?
3883	027360	100404			BMI	.+12	: BR IF YES
3884	027362	005237	001416		INC	TEMP	: INC DELAY
3885	027366	001373			BNE	6\$: BR IF NOT DONE
3886	027370	104014			HLT	14	: ERROR RDI NOT SET
3887	027372	012761	034674	000004	MOV	#TBUF, 4(R1)	: LOAD XMIT BUFFER
3888	027400	013761	034672	000006	MOV	TOUNT, 6(R1)	: LOAD COUNT
3889	027406	142711	000040		BICB	#40, (R1)	: CLEAR RQI
3890	027412	005037	001416		CLR	TEMP	: GET SET TO DELAY
3891	027416	105711		7\$:	TSTB	(R1)	: RDI GONE?
3892	027420	100004			BPL	.+12	: BR IF YES
3893	027422	005237	001416		INC	TEMP	: INC DELAY
3894	027426	001373			BNE	7\$: BR IF NOT DONE DELAY
3895	027430	104014			HLT	14	: ERROR RDI STILL SET
3896	027432	005037	001416		CLR	TEMP	: GET SET TO DELAY
3897	027436	012737	000022	001246	MOV	#22, TEMP1	: GET SET FOR LONG DELAY
3898	027444	105761	000002		11\$:	TSTB	2(R1),
3899	027450	100407			BMI	17\$: BR IF YES
3900	027452	005237	001416		INC	TEMP	: INC DELAY
3901	027456	001372			BNE	11\$: BR IF DELAY NOT DONE
3902	027460	005337	001246		DEC	TEMP1	: DEC DELAY COUNT
3903	027464	001367			BNE	11\$: BR IF NOT DONE DELAY
3904	027466	104014			HLT	14	: ERROR RDI NOT SET
3905	0274	016137	000002	001250	17\$:	MOV	2(R1), TEMP2
3906	027476	001001			BNE	.+4	: BR IF OK
3907	027500	104014			HLT	14	: ERROR!!! SEL2 = 0!!!!!!
3908	027502	032761	000004	000002	BIT	#BIT2, 2(R1)	: REC OR XMIT?
3909	027510	001032			BNE	13\$: BR IF REC
3910	027512	005737	034666		12\$:	TST	TFLAG
3911	027516	001401			BEQ	.+4	: BR IF YES
3912	027520	104014			HLT	14	: ERROR MULTIPLE XMIT DONES
3913	027522	012737	177777	034666	MOV	#-1, TFLAG	: SET TFLAG TO -1
3914	027530	132761	000001	000002	BITB	#BIT0, 2(R1)	: IS IT CONTROL 0
3915	027536	001401			BEQ	.+4	: BR IF NO
3916	027540	104014			HLT	14	: XMIT ERROR
3917	027542	022761	034674	000004	CMP	#TBUF, 4(R1)	: XMIT BA CORRECT?
3918	027550	001401			BEQ	.+4	: BR IF YES
3919	027552	104014			HLT	14	: XMIT BA ERROR

3920	027554	023761	034672	000006		CMP	TCount,6(R1)	;CJUNT OK?
3921	027562	001401				BEQ	.+4	;BR IF YES
3922	027564	104014				HLT	14	;XMIT COUNT ERROR
3923	027566	142761	000207	000002		BICB	#207,2(R1)	;CLEAR RDO AND BITS 0-2
3924	027574	000453				BR	15\$;CONTINUE
3925	027576	005737	034670		13\$:	TST	RFLAG	;FIRST TIME HERE?
3926	027602	001401				BEQ	.+4	;BR IF YES
3927	027604	104014				HLT	14	;ERROR MULTIPLE REC DONES
3928	027606	012737	177777	034670		MOV	#-1,RFLAG	;SET RFLAG TO -1
3929	027614	132761	000001	000002		BITB	#BIT0,2(R1)	;IS IT CNTL 0
3930	027622	001401				BEQ	.+4	;BR IF NO
3931	027624	104014				HLT	14	;RECEIVE ERROR
3932	027626	022761	034742	000004		CMP	#RBUF,4(R1)	;REC BA CORRECT?
3933	027634	001401				BEQ	.+4	;BR IF YES
3934	027636	104014				HLT	14	;REC BA ERROR
3935	027640	023761	034740	000006		CMP	RCOUNT,6(R1)	;COUNT OK?
3936	027646	001401				BEQ	.+4	;BR IF YES
3937	027650	104014				HLT	14	;REC COUNT ERROR
3938	027652	013700	034740			MOV	RCOUNT,R0	;GET SET TO CHECK DATA
3939	027656	012702	034674			MOV	#TBUF,R2	;R2 POINTS TO GOOD DATA
3940	027662	012703	034742			MOV	#RBUF,R3	;R3 POINTS TO RECEIVE DATA
3941	027666	010337	001252		9\$:	MOV	R3,TEMP3	;SAVE ADDRESS FOR TYPEOUT
3942	027672	112205				MOVB	(R2)+,R5	;R5 = XMIT DATA
3943	027674	112304				MOVB	(R3)+,R4	;R4 = RECEIVE DATA
3944	027676	120504				CMPB	R5,R4	;CHECK DATA
3945	027700	001401				BEQ	.+4	;BR IF OK
3946	027702	104013				HLT	13	;DATA ERROR
3947	027704	005300				DEC	R0	;DEC COUNT
3948	027706	001367				BNE	9\$;BR IF NOT DONE
3949	027710	005713				TST	(R3)	;THIS SHOULD BE 0, ELSE
3950	027712	001401				BEQ	.+4	;IT RECEIVED TO MUCH!!
3951	027714	104014				HLT	14	;ERROR
3952	027716	142761	000207	000002		BICB	#207,2(R1)	;CLEAR RDO AND BITS 0-2
3953	027724	005737	034670		15\$:	TST	RFLAG	;REC DONE?
3954	027730	001640				BEQ	16\$;BR IF NO
3955	027732	005737	034666			TST	TFLAG	;XMIT DONE?
3956	027736	001635				BEQ	16\$;BR IF NO
3957	027740	104400			14\$:	SCOPE		;SCOPE THIS TEST

```

;***** TEST 51 *****
; *OVERUN TEST
; *IN FREE RUNNING MODE SEND MESSAGE WITH NO RECEIVE
; *BUFFER AVAILABLE, VERIFY THAT AN OVERRUN ERROR OCCURS
;*****

```

; TEST 51

3967								
3968	027742	012737	000051	001226	TST51:	MOV	#51,TSTNO	
3969	027750	012737	030170	001216		MOV	#TST52,NEXT	
3970								
3971	027756	104412				MSTCLR		;R1 CONTAINS BASE DMC11 ADDRESS
3972	027760	032737	100000	001366		BIT	#BIT15,STAT1	;MASTER CLEAR DMC11
3973	027766	001406				BEQ	.+16	;IS IT A DMC?
3974	027770	032737	000001	001372		BIT	#BIT0,STAT3	;BR IF YES
3975	027776	001002				BNE	.+6	;KMC WITH BIT0 SET?

DZDMG MACY11 27(1006) 07-MAR-77 15:52 PAGE 77
DZDMGB.P11 07-MAR-77 11:21 FREE RUNNING TESTS

3976	030000	000137	030152		JMP	10\$;SKIP TEST
3977	030004	032737	010000	001366	BIT	#BIT12,STAT1	;LU PRESENT?
3978	030012	001372			BNE	.-12	;BR IF NO
3979	030014	004737	035562		JSR	PC,WROM	;WRITE MICRO-CODE IN CRAM
3980	030020	004737	035762		JSR	PC,BASELD	;LOAD DMC BASE ADDRESS
3981	030024	004537	036252		JSR	R5,XFRELD	;LOAD XMIT BA/CC
3982	030030	034674			TBUF		;BA
3983	030032	000044			44		;CC
3984	030034	012700	000010		MOV	#10,R0	;RO = RETRANSMISSION COUNT
3985	030040	012703	000010		MOV	#10,R3	;DELAY COUNT
3986	030044	005037	001416		CLR	TEMP	;CLEAR DELAY COUNTER
3987	030050	105761	000002		1\$: TSTB	2(R1)	;IS RDY 0 SET?
3988	030054	100407			BMI	.-+20	;BR IF SET
3989	030056	005237	001416		INC	TEMP	;INC DELAY COUNTER
3990	030062	001372			BNE	1\$;BR IF NOT DONE DELAY
3991	030064	005303			DEC	R3	;DEC DELAY COUNT
3992	030066	001370			BNE	1\$;BR IF DELAY NOT DONE
3993	030070	104014			HLT	14	;ERROR, RDY 0 NOT SET
3994	030072	000427			BR	10\$;GET OUT
3995	030074	132761	000001	000002	BITB	#BIT0,2(R1)	;IS IT CNTL 0?
3996	030102	001002			BNE	11\$;BR IF YES
3997	030104	104014			HLT	14	;ERROR, NOT CNTL 0
3998	030106	000421			BR	10\$;CONTINUE
3999	030110	012705	000004		11\$: MOV	#BIT2,R5	;PUT "EXPECTED" IN R5
4000	030114	016104	000006		MOV	6(R1),R4	;PUT "FOUND" IN R4
4001	030120	020504			CMP	R5,R4	;IS ORUN SET?
4002	030122	001404			BEQ	12\$;BR IF YES
4003	030124	022704	000001		CMP	#1,R4	;DATA CK ERROR?
4004	030130	001411			BEQ	13\$;BR IF YES
4005	030132	104015			HLT	15	;ERROR, ORUN NOT SET
4006	030134	042761	000207	000002	12\$: BIC	#207,2(R1)	;CLEAR RDO
4007	030142	005037	001416		CLR	TEMP	;RESET DELAY
4008	030146	005300			DEC	RO	;DEC RETRANS COUNT
4009	030150	001337			BNE	1\$;COUNTINUE
4010	030152	104400			10\$: SCOPE		;SCOPE THIS TEST
4011	030154	042761	000207	000002	13\$: BIC	#207,2(R1)	;IGNOR THIS ERROR
4012	030162	005037	001416		CLR	TEMP	;RESET DELAY
4013	030166	000730			BR	1\$;CONTINUE
4014							
4015							
4016							
4017							
4018							
4019							
4020							
4021							
4022							
4023							
4024	030170	012737	011052	001226	TST52: MOV	#52,TSTNO	
4025	030176	012737	031362	001216	MOV	#TST53,NEXT	
4026							
4027	030204	104412			MSTCLR		;R1 CONTAINS BASE DMC11 ADDRESS
4028	030206	032737	110000	001366	BIT	#BIT15,STAT1	;MASTER CLEAR DMC11
4029	030214	001406			BEQ	.-+16	;IS IT A DMC?
4030	030216	032737	000001	001372	BIT	#BIT0,STAT3	;BR IF YES
4031	030224	001102			BNE	.-+6	;KMC WITH BIT0 SET?

```

;***** TEST 52 *****
; *LOST DATA TEST
; *IN FREE RUNNING MODE SEND A MESSAGE LONGER THAN THE RECEIVE
; *BUFFER, VERIFY THAT A LOST DATA ERROR OCCURS.
;*****

```

```

; TEST 52
;-----

```

```

TST52: MOV #52,TSTNO
MOV #TST53,NEXT

MSTCLR
BIT #BIT15,STAT1
BEQ .+16
BIT #BIT0,STAT3
BNE .+6
;R1 CONTAINS BASE DMC11 ADDRESS
;MASTER CLEAR DMC11
;IS IT A DMC?
;BR IF YES
;KMC WITH BIT0 SET?
;BR IF YES

```

4032	030226	000137	030360		JMP	10\$:SKIP TEST
4033	030232	032737	010000	001366	BIT	#BIT12,STAT1	:LU PRESENT?
4034	030240	001372			BNE	-12	:BR IF NO
4035	030242	004737	035562		JSR	PC,WROM	:WRITE MICRO-CODE IN CRAM
4036	030246	004737	035762		JSR	PC,BASELD	:LOAD DMC BASE ADDRESS
4037	030252	004537	036220		JSR	R5,XFRELD	:LOAD RECEIVE BA/CC
4038	030256	034742			RBUF		:BA
4039	030260	000020			20		:CC
4040	030262	004537	036252		JSR	R5,XFRELD	:LOAD XMIT BA/CC
4041	030266	034674			TBUF		:BA
4042	030270	000044			44		:CC
4043	030272	012703	000010		MOV	#10,R3	:DELAY COUNT
4044	030276	005037	001416		CLR	TEMP	:CLEAR DELAY COUNTER
4045	030302	105761	000002	1\$:	TSTB	2(R1)	:IS RDY 0 SET?
4046	030306	100407			BMI	+20	:BR IF SET
4047	030310	005237	001416		INC	TEMP	:INC DELAY COUNTER
4048	030314	001372			BNE	1\$:BR IF NOT DONE DELAY
4049	030316	005303			DEC	R3	:DEC DELAY COUNT
4050	030320	001370			BNE	1\$:BR IF DELAY NOT DONE
4051	030322	104014			HLT	14	:RDY 0 NOT SET
4052	030324	000415			BR	10\$:CONTINUE
4053	030326	132761	000001	000002	BITB	#BIT0,2(R1)	:IS CNTL 0?
4054	030334	001002			BNE	11\$:BR IF YES
4055	030336	104014			HLT	14	:ERROR NOT CNTL 0
4056	030340	000407			BR	10\$:CONTINUE
4057	030342	012705	000020	11\$:	MOV	#BIT4,R5	:PUT "EXPECTED" IN R5
4058	030346	016104	000006		MOV	6(R1),R4	:PUT "FOUND" IN R4
4059	030352	020504			CMP	R5,R4	:IS LOST DATA SET?
4060	030354	001401			BEQ	10\$:BR IF YES
4061	030356	104015			HLT	15	:ERROR, LOST DATA NOT SET
4062	030360	104400		10\$:	SCOPE		:SCOPE THIS TEST

```

:***** TEST 53 *****
: *TRANSMIT NON-EXISTENT MEMORY TEST
: *IN FREE RUNNING MODE, LOAD A TRANSMIT BA THAT WILL TIME OUT
: *VERIFY THAT A NON-EXISTENT MEMORY ERROR OCCURS
:*****

```

TEST 53

4073	030362	012737	000053	001226	TST53:	MOV	#53,TSTNO	
4074	030370	012737	030544	001216		MOV	#TST54,NEXT	
4075								:R1 CONTAINS BASE DMC11 ADDRESS
4076	030376	104412			MSTCLR			:MASTER CLEAR DMC11
4077	030400	032737	100000	001366	BIT	#BIT15,STAT1		:IS IT A DMC?
4078	030406	001406			BEQ	+16		:BR IF YES
4079	030410	032737	000001	001372	BIT	#BIT0,STAT3		:KMC WITH BIT0 SET?
4080	030416	001002			BNE	+6		:BR IF YES
4081	030420	000137	030542		JMP	10\$:SKIP TEST
4082	030424	032737	010000	001366	BIT	#BIT12,STAT1		:LU PRESENT?
4083	030432	001372			BNE	-12		:BR IF NO
4084	030434	004737	035562		JSR	PC,WROM		:WRITE MICRO-CODE IN CRAM
4085	030440	004737	035762		JSR	PC,BASELD		:LOAD DMC BASE ADDRESS
4086	030444	004537	036252		JSR	R5,XFRELD		:LOAD XMIT BA/CC
4087	030450	177320			177320			:BA

4088	030452	140044			140044				:CC
4089	030454	012703	000010		MOV	#10,R3			:DELAY COUNT
4090	030460	005037	001416		CLR	TEMP			:CLEAR DELAY COUNTER
4091	030464	105761	000002		TSTB	2(R1)			:IS RDY 0 SET?
4092	030470	100407		15:	BMI	.+20			:BR IF SET
4093	030472	005237	001416		INC	TEMP			:INC DELAY COUNTER
4094	030476	001372			BNE	15			:BR IF NOT DONE DELAY
4095	030500	005303			DEC	R3			:DEC DELAY COUNT
4096	030502	001370			BNE	15			:BR IF DELAY NOT DONE
4097	030504	104014			HLT	14			:ERROR, RDY 0 NOT SET
4098	030506	000415			BR	105			:GET OUT
4099	030510	132761	000001	000002	BITB	#BIT0,2(R1)			:IS IT CNTL 0?
4100	030516	001002			BNE	115			:BR IF YES
4101	030520	104014			HLT	14			:ERROR, NOT CNTL 0
4102	030522	000407			BR	105			:CONTINUE
4103	030524	012705	000400		MOV	#BIT8,R5			:PUT "EXPECTED" IN R5
4104	030530	016104	000006		MOV	6(R1),R4			:PUT "FOUND" IN R4
4105	030534	020504		115:	CMP	R5,R4			:IS NON-EX-MEM SET?
4106	030536	001401			BEQ	.+4			:BR IF YES
4107	030540	104015			HLT	15			:ERROR NON-EX-MEM NOT SET
4108	030542	104400		105:	SCOPE				:SCOPE THIS TEST

```

:***** TEST 54 *****
:*RECEIVE NON-EXISTENT MEMORY TEST
:*IN FREE RUNNING MODE, LOAD A RECEIVE BA THAT WILL TIME OUT
:*VERIFY THAT A NON-EXISTENT MEMORY ERROR OCCURS
:*****

```

: TEST 54

4118									
4119	030544	012737	000054	001226	TST54:	MOV	#54,TSTNO		
4120	030552	012737	030736	001216		MOV	#TST55,NEXT		
4121									:R1 CONTAINS BASE DMC11 ADDRESS
4122	030560	104412			MSTCLR				:MASTER CLEAR DMC11
4123	030562	032737	100000	001366	BIT	#BIT15,STAT1			:IS IT A DMC?
4124	030570	001406			BEQ	.+16			:BR IF YES
4125	030572	032737	000001	001372	BIT	#BIT0,STAT3			:KMC WITH BIT0 SET?
4126	030600	001002			BNE	.+6			:BR IF YES
4127	030602	000137	030734		JMP	105			:SKIP TEST
4128	030606	032737	010000	001366	BIT	#BIT12,STAT1			:LU PRESENT?
4129	030614	001372			BNE	.-12			:BR IF NO
4130	030616	004737	035562		JSR	PC,WROM			:WRITE MICRO-CODE IN CRAM
4131	030622	004737	035762		JSR	PC,BASEL			:LOAD DMC BASE ADDRESS
4132	030626	004537	036220		JSR	R5,RFRELD			:LOAD RECEIVE BA/CC
4133	030632	177320				177320			:BA
4134	030634	140044				140044			:CC
4135	030636	004537	036252		JSR	R5,XFRELD			:LOAD XMIT BA/CC
4136	030642	034674			TBUF				:BA
4137	030644	000044			44				:CC
4138	030646	012703	000010		MOV	#10,R3			:DELAY COUNT
4139	030652	005037	001416		CLR	TEMP			:CLEAR DELAY COUNTER
4140	030656	105761	000002		TSTB	2(R1)			:IS RDY 0 SET?
4141	030662	100407		15:	BMI	.+20			:BR IF SET
4142	030664	005237	001416		INC	TEMP			:INC DELAY COUNTER
4143	030670	001372			BNE	15			:BR IF NOT DONE DELAY

4144	030672	005303			DEC	R3	:DEC DELAY COUNT
4145	030674	001370			BNE	1\$:BR IF DELAY NOT DONE
4146	030676	104014			HLT	14	:ERROR, RDY 0 NOT SET
4147	030700	000415			PR	10\$:GET OUT
4148	030702	132761	000001	000002	BITB	#BIT0,2(R1)	:IS IT CNTL 0?
4149	030710	001002			3NE	11\$:BR IF YES
4150	030712	104014			HLT	14	:ERROR, NOT CNTL 0
4151	030714	000407			BR	10\$:CONTINUE
4152	030716	012705	000400		MOV	#BIT8,R5	:PUT "EXPECTED" IN R5
4153	030722	016104	000006		MOV	6(R1),R4	:PUT "FOUND" IN R4
4154	030726	020504			CMP	R5,R4	:IS NON-EX-MEM SET?
4155	030730	001401			BEQ	.+4	:BR IF YES
4156	030732	104015			HLT	15	:ERROR NON-EX-MEM NOT SET
4157	030734	104400			SCOPE		:SCOPE THIS TEST

```

:***** TEST 55 *****
:*PROCESSOR ERROR TEST
:*IN FREE RUNNING MODE, DO A BASE TRANSFER REQUEST AFTER A
:*BASE HAS BEEN SET UP, VERIFY THAT A PROCESSOR ERROR OCCURS.
:*****

```

: TEST 55

4168	030736	012737	000055	001226	TST55:	MOV	#55,TSTNO	
4169	030744	012737	031114	001216		MOV	#TST56,NEXT	
4171	030752	104412			MSTCLR		:R1 CONTAINS BASE DMC11 ADDRESS	
4172	030754	032737	100000	001366	BIT	#BIT15,STAT1	:MASTER CLEAR DMC11	
4173	030762	001406			BEQ	.+16	:IS IT A DMC?	
4174	030764	032737	000001	001372	BIT	#BIT0,STAT3	:BR IF YES	
4175	030772	001002			BNE	.+6	:KMC WITH BIT0 SET?	
4176	030774	000137	031112		JMP	10\$:BR IF YES	
4177	031000	032737	010000	001366	BIT	#BIT12,STAT1	:SKIP TEST	
4178	031006	001372			BNE	.-12	:LU PRESENT?	
4179	031010	004737	035562		JSR	PC,WROM	:BR IF NO	
4180	031014	004737	035762		JSR	PC,BASELD	:WRITE MICRO-CODE IN CRAM	
4181	031020	152711	000043		BISB	#43,(R1)	:LOAD BASE ADDRESS	
4182	031024	105711			TSTB	(R1)	:2ND BASE REQUEST	
4183	031026	100376			BPL	.-2	:RDI SET?	
4184	031030	142711	000040		BICB	#40,(R1)	:BR IF NO	
4185	031034	005037	001416		CLR	TEMP	:CLEAR RQI	
4186	031040	105761	000002		TSTB	2(R1)	:GET SET TO DELAY	
4187	031044	100405			BMI	14\$:RDO SET?	
4188	031046	005237	001416		INC	TEMP	:BR IF YES	
4189	031052	001372			BNE	13\$:INC DELAY	
4190	031054	104014			HLT	14	:BR IF NOT DONE DELAY	
4191	031056	000770			BR	13\$:ERROR, RDO NOT SET	
4192	031060	132761	000001	000002	BITB	#BIT0,2(R1)	:TRY AGAIN	
4193	031066	001002			BNE	11\$:IS IS CNTL 0?	
4194	031070	104014			HLT	14	:BR IF YES	
4195	031072	000407			BR	10\$:ERROR NOT CNTL 0	
4196	031074	012705	001000		MOV	#BIT9,R5	:CONTINUE	
4197	031100	016104	000006		MOV	6(R1),R4	:PUT "EXPECTED" IN R5	
4198	031104	020504			CMP	R5,R4	:PUT "FOUND" IN R4	
4199	031106	001401			BEQ	.+4	:IS PROC ERROR SET?	

:BR IF YES

4200 031110 104015
4201 031112 104400
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212 031114 012737 000056 001226
4213 031122 012737 031272 001216
4214
4215 031130 104412
4216 031132 032737 100000 001366
4217 031140 001406
4218 031142 032737 000001 001372
4219 031150 001002
4220 031152 000137 031270
4221 031156 032737 010000 001366
4222 031164 001372
4223 031166 004737 035562
4224 031172 004737 035762
4225 031176 152711 000046
4226 031202 105711
4227 031204 100376
4228 031206 142711 000040
4229 031212 005037 001416
4230 031216 105761 000002
4231 031222 100405
4232 031224 005237 001416
4233 031230 001372
4234 031232 104014
4235 031234 000770
4236 031236 132761 000001 000002
4237 031244 001002
4238 031246 104014
4239 031250 000407
4240 031252 012705 001000
4241 031256 016104 000006
4242 031262 020504
4243 031264 001401
4244 031266 104015
4245 031270 104400
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255

10\$: HLT 15
SCOPE

;ERROR, PROC ERROR NOT SET
;SCOPE THIS TEST

***** TEST 56 *****
*PROCESSOR ERROR TEST
*IN FREE RUNNING MODE DO A RQI WITH AN ILLEGAL IO CODE
*VERIFY THAT A PROCESSOR ERROR OCCURS

: TEST 56

TST56:

MOV #56,TSTNO
MOV #TST57,NEXT
MSTCLR
BIT #BIT15,STAT1
BEQ .+16
BIT #BIT0,STAT3
BNE .+6
JMP 10\$
BIT #BIT12,STAT1
BNE .-12
JSR PC,WROM
JSR PC,BASELD
BISB #46,(R1)
TSTB (R1)
BPL .-2
BICB #40,(R1)
CLR TEMP
TSTB 2(R1)
BMI .+14
INC TEMP
BNE 1\$
HLT 14
BR 1\$
BITB #BIT0,2(R1)
BNE 11\$
HLT 14
BR 10\$
MOV #BIT9,R5
MOV 6(R1),R4
CMP R5,R4
BEQ .+4
HLT 15
SCOPE

;R1 CONTAINS BASE DMC11 ADDRESS
;MASTER CLEAR DMC11
;IS IT A DMC?
;BR IF YES
;KMC WITH BIT0 SET?
;BR IF YES
;SKIP TEST
;LU PRESENT?
;BR IF NO
;WRITE MICRO-CODE IN CRAM
;LOAD DMC BASE ADDRESS
;RQI AND ILLEGAL CODE
;WAIT FOR RQI
;BR IF NO RQI
;CLEAR RQI
;CLEAR COUNTER
;RDY 0 SET?
;BR IF YES
;BUMP COUNTER DELAY
;BR IF NOT DONE
;ERROR NO RDY 0
;TRY AGAIN
;IS IT CNTL 0
;BR IF YES
;ERROR, NOT CNTL 0
;CONTINUE
;PUT "EXPECTED" IN R5
;PUT "FOUND" IN R4
;IS PROC ERROR SET?
;BR IF YES
;ERROR PROC ERROR NOT SET
;SCOPE THIS TEST

1\$:

11\$:

10\$:

***** TEST 57 *****
*HALF DUPLEX TEST
*IN FREE RUNNING MODE, SET HALF DUPLEX AND L U LOOP
*SEND A MESSAGE AND VERIFY THAT THERE ARE NO DONES

: TEST 57

4256 031272 012737 000057 001226
 4257 031300 012737 031432 001216
 4258
 4259 031306 104412
 4260 031310 032737 100000 001366
 4261 031316 001406
 4262 031320 032737 000001 001372
 4263 031326 001002
 4264 031330 000137 031424
 4265 031334 032737 010000 001366
 4266 031342 001372
 4267 031344 004737 035562
 4268 031350 004737 036100
 4269 031354 004537 036220
 4270 031360 034742
 4271 031362 000044
 4272 031364 004537 036252
 4273 031370 034674
 4274 031372 000044
 4275 031374 012703 000003
 4276 031400 005037 001416
 4277 031404 105761 000002
 4278 031410 100406
 4279 031412 005237 001416
 4280 031416 001372
 4281 031420 005303
 4282 031422 001370
 4283 031424 104400
 4284 031426 104014
 4285 031430 000775

TST57: MOV #57,TSTNO
 MOV #TST60,NEXT
 MSTCLR
 BIT #BIT15,STAT1
 BEQ .+16
 BIT #BIT0,STAT3
 BNE .+6
 JMP 10\$
 BIT #BIT12,STAT1
 BNE .-12
 JSR PC,WROM
 JSR PC,BASELH
 JSR R5,RFRELD
 RBUF
 44
 JSR R5,XFRELD
 TBUF
 44
 MOV #3,R3
 CLR TEMP
 4\$: TSTB 2(R1)
 BMI 5\$
 INC TEMP
 BNE 4\$
 DEC R3
 BNE 4\$
 10\$: SCOPE
 5\$: HLT 14
 BR 10\$

;R1 CONTAINS BASE DMC11 ADDRESS
 ;MASTER CLEAR DMC11
 ;IS IT A DMC?
 ;BR IF YES
 ;KMC WITH BIT0 SET?
 ;BR IF YES
 ;SKIP TEST
 ;LU PRESENT?
 ;BR IF NO
 ;WRITE MICRO-CODE
 ;LOAD BASE AND HALF DUPLEX
 ;LOAD RECEIVE BUFFER
 ;BA
 ;CC
 ;LOAD TRANSMIT BUFFER
 ;BA
 ;CC
 ;LOAD DELAY COUNT
 ;CLEAR DELAY
 ;IS DONE SET?
 ;BR IF YES (ERROR)
 ;INC DELAY
 ;BR IF DELAY NOT DONE
 ;DEC DELAY COUNT
 ;BR IF DELAY NOT DONE
 ;SCOPE THIS TEST
 ;ERROR DONE WITH HALF-DUPLEX
 ;GET OUT

4286
 4287
 4288
 4289
 4290
 4291
 4292
 4293
 4294
 4295
 4296
 4297
 4298
 4299 031432 012737 000060 001226
 4300 031440 012737 003274 001216
 4301
 4302 031446 104412
 4303 031450 032737 100000 001366
 4304 031456 001406
 4305 031460 032737 000001 001372
 4306 031466 001002
 4307 031470 000137 032402
 4308 031474 032737 010000 001366
 4309 031502 001372
 4310 031504 004737 035562
 4311 031510 012737 000340 177776

TST60: MOV #60,TSTNO
 MOV #.EOP,NEXT
 MSTCLR
 BIT #BIT15,STAT1
 BEQ .+16
 BIT #BIT0,STAT3
 BNE .+6
 JMP ENDEX1
 BIT #BIT12,STAT1
 BNE .-12
 JSR PC,WROM
 MOV #340,PS

;R1 CONTAINS BASE DMC11 ADDRESS
 ;MASTER CLEAR DMC11
 ;IS IT A DMC?
 ;BR IF YES
 ;KMC WITH BIT0 SET?
 ;BR IF YES
 ;SKIP TEST
 ;LU PRESENT?
 ;BR IF NO
 ;WRITE MICR-CODE
 ;LOCK OUT INTERRUPTS

***** TEST 60 *****
 ;*FREE RUNNING DATA TEST (INTERRUPT DRIVEN EXERCISER)
 ;*THIS TEST REPEATEDLY QUEUES UP 7 RECEIVE BUFFERS AND
 ;*7 TRANSMIT BUFFERS AND CHECKS DATA WHEN ALL 7 BUFFERS
 ;*ARE RECEIVED. TRANSMIT COUNTS RANGE FROM 1 TO 104. ALSO
 ;*ODD AND EVEN TRANSMIT AND RECEIVE BA'S ARE USED. DATA
 ;*IS A BINARY COUNT PATTERN. THE RESUME FUNCTION IS CHECKED IN THIS TEST
 ;*****
 ; TEST 60
 ;-----

4312	031516	013700	001366		MOV	STAT1,RO	;GET BR LEVEL
4313	031522	006200			ASR	RO	;SHIFT RIGHT 4 TIMES
4314	031524	006200			ASR	RO	
4315	031526	006200			ASR	RO	
4316	031530	006200			ASR	RO	
4317	031532	042700	177437		BIC	#177437,RO	;PUT BR LEVEL IN RO
4318	031536	012777	032476	147630	MOV	#IISR,DMRVEC	;LOAD INPUT VECTOR
4319	031544	010077	147626		MOV	RO,DMRLVL	;LOAD LEVEL
4320	031550	012777	033002	147622	MOV	#OISR,DMTVEC	;LOAD OUTPUT VECTOR
4321	031556	010077	147620		MOV	RO,DMTLVL	;LOAD LEVEL
4322							
4323							
4324							
4325	031562	012737	000104	034666	MOV	#104,TFLAG	;TFLAG CONTAINS COUNT
4326	031570	012700	033416		MOV	#XMITBA+2,RO	;RO POINTS TO BA LIST
4327	031574	012703	033710		MOV	#RBUF,R3	;R3 CONTAINS BUFFER ADDRESS
4328	031600	010320		1\$:	MOV	R3,(R0)+	;LOAD BA LIST WITH REC BA
4329	031602	062703	000106		ADD	#106,R3	;UPDATE BUFFER ADDRESS
4330	031606	022700	033434		CMP	#XMITBA+20,RO	;END OF REC BUFFERS?
4331	031612	001372			BNE	1\$;NO LOAD NEXT ONE
4332	031614	012720	033452	2\$:	MOV	#TBUF,(R0)+	;LOAD BA LIST WITH XMIT BA
4333	031620	022700	033452		CMP	#XMITBA+36,RO	;END OF XMIT BUFFERS?
4334	031624	001373			BNE	2\$;NO LOAD NEXT BUFFER
4335	031626	012700	033564		MOV	#RCNTAB+2,RO	;RO POINTS TO COUNT LIST
4336	031632	013720	034666	3\$:	MOV	TFLAG,(R0)+	;LOAD COUNT OF 104
4337	031636	022700	033602		CMP	#RCNTAB+20,RO	;END OF REC COUNT LIST?
4338	031642	001373			BNE	3\$;BR IF NO
4339	031644	012737	000006	034664	MOV	#6,FLAG ;LOOP COUNT	
4340	031652	012711	040000		MOV	#BIT14,(R1)	;SET MASTER CLEAR
4341	031656	032737	100000	001366	BIT	#BIT15,STAT1	;IOP?
4342	031664	001402			BEQ	.+6	;BR IF NO
4343	031666	012711	100000		MOV	#BIT15,(R1)	;SET RUN ON IOP
4344	031672	012700	177777		MOV	#-1,RO	;RO IS INPUT DONE COUNTER
4345	031676	005037	033412	CLRTAB:	CLR	RESUME	;CLEAR RESUME FLAG
4346	031702	012705	033620		MOV	#RONTAB,R5	;GET READY TO CLEAR ALL RECEIVE
4347	031706	005025		2\$:	CLR	(R5)+	;BUFFERS
4348	031710	022705	034662		CMP	#RBUFE,R5	;END OF BUFFER?
4349	031714	001374			BNE	2\$;BR IF NO
4350	031716	005737	034664		TST	FLAG	;VARIABLE COUNTS?
4351	031722	100407			BMI	5\$;BR IF YES(DON'T CHANGE THEM)
4352	031724	012704	033602		MOV	#XCNTAB,R4	;R4 POINTS TO XMIT COUNT LIST
4353	031730	013724	034666	4\$:	MOV	TFLAG,(R4)+	;LOAD XMIT CHAR COUNT
4354	031734	022704	033620		CMP	#XCNTAB+16,R4	;DONE?
4355	031740	001373			BNE	4\$;BR IF NO
4356	031742	005002		5\$:	CLR	R2	;R2 IS OUTPUT DONE COUNTER
4357	031744	005004			CLR	R4	;R4 IS USED AS INDEX IN OISR
4358	031746	005711			TST	(R1)	;IS RUN SET?
4359	031750	100376			BPL	.-2	;WAIT FOR RUN
4360	031752	152761	000100	000002	BISB	#BIT6,2(R1)	;SET IEO
4361	031760	022737	000006	034664	CMP	#6,FLAG ;FIRST TIME?	
4362	031766	001003			BNE	1\$;BR IF NOT
4363	031770	052711	004143		BIS	#4143,(R1)	;SET LU LOOP, IEI,RQI,BASE I
4364	031774	000402			BR	3\$;CONTINUE
4365	031776	052711	004144	1\$:	BIS	#4144,(R1)	;SET LU LOOP, IEI, RQI, REC BA-CC
4366	032002	005037	001416	3\$:	CLR	TEMP	;SET UP FOR DELAY COUNT
4367	032006	012737	000022	001250	MOV	#22,TEMP2	;GET SET FOR DELAY

4368	032014	005037	177776			CLR	PS	; ALLOW INTERRUPTS
4369	032020	022737	000001	034664	SCAN:	CMP	#1, FLAG	; 1 BYTE MESS?
4370	032026	001002				BNE	1\$; BR IF NO
4371	032030	000137	032440			JMP	ENDEX3	; BR IF YES
4372	032034	022700	000020		1\$:	CMP	#20, R0	; INPUT DONE?
4373	032040	001402				BEQ	SCAN2	; BR IF YES
4374	032042	000137	032446			JMP	SCAN1	; BR IF NO
4375	032046	022702	000034		SCAN2:	CMP	#34, R2	; XMIT DONE FOR ALL MESSAGES?
4376	032052	001175				BNE	SCAN1	; BR IF NO
4377	032054	022704	000034			CMP	#34, R4	; REC DONE FOR ALL MESSAGES?
4378	032060	001172				BNE	SCAN1	; BR IF NO
4379	032062	012700	033620			MOV	#RDNTAB, R0	; GET FIRST REC BUFFER
4380	032066	012002			5\$:	MOV	(R0)+, R2	; R2 POINTS TO BUFFER
4381	032070	005005				CLR	R5	; R5=EXPECTED
4382	032072	005003				CLR	R3	; R3 = COUNT
4383	032074	005737	034664			TST	FLAG	; CHECK FOR ODD XMIT BA'S
4384	032100	100012				BPL	6\$; ONLY FOR VARIABLE COUNTS
4385	032102	022710	000027			CMP	#27, (R0)	; IF 27 BUMP DATA BY 1 (ODD XMIT BA)
4386	032106	001406				BEQ	7\$; BR IF YES
4387	032110	022710	000042			CMP	#42, (R0)	; IF 42 THEN ODD XMIT BA ALSO
4388	032114	001403				BEQ	7\$; BR IF YES
4389	032116	022710	000103			CMP	#103, (R0)	; IF 103 THEN ODD XMIT BA ALSO
4390	032122	001001				BNE	6\$; SKIP IF NOT
4391	032124	005205			7\$:	INC	R5	; START DATA AT 1 FOR ODD XMIT BA'S
4392	032126	010237	001252		6\$:	MOV	R2, TEMP3	; SAVE ADDRESS FOR TYPEOUT
4393	032132	112204				MOVB	(R2)+, R4	; GET RECEIVE DATA
4394	032134	120504				CMPB	R5, R4	; IS IT CORRECT?
4395	032136	001401				BEQ	.+4	; BR IF YES
4396	032140	104013				HLT	13	; DATA ERROR
4397	032142	005205				INC	R5	; NEXT CHARACTER
4398	032144	005203				INC	R3	; INC COUNT
4399	032146	021003				CMP	(R0), R3	; DONE YET?
4400	032150	001366				BNE	6\$; BR IF NO
4401	032152	062700	000002			ADD	#2, R0	; GET NEXT REC BUFFER
4402	032156	022700	033654			CMP	#RDNTAB+34, R0	; DONE YET?
4403	032162	001341				BNE	5\$; BR IF NO
4404	032164	012700	000001			MOV	#1, R0	; SET R0 TO 1
4405	032170	005737	034664			TST	FLAG	; VARIABLE COUNTS?
4406	032174	100004				BPL	4\$; BR IF NO
4407	032176	005237	034664			INC	FLAG	; FLAG IS NEGITIVE
4408	032202	001235				BNE	CLRTAB	; BR IF NOT DONE
4409	032204	000447				BR	ENDEX	; ALL DONE
4410	032206	032737	000001	034664	4\$:	BIT	#BIT0, FLAG	; CHANGE CHAR COUNT FOR NEXT LOOP
4411	032214	001003				BNE	1\$; BR TO SUB 40
4412	032216	005337	034666			DEC	TFLAG	; DEC BY ONE
4413	032222	000403				BR	2\$; CONTINUE
4414	032224	162737	000040	034666	1\$:	SUB	#40, TFLAG	; SUBTRACT 40 FROM XMIT COUNT
4415	032232	005337	034664		2\$:	DEC	FLAG	; DEC LOOP COUNT
4416	032236	001217				BNE	CLRTAB	; GO DO IT AGAIN
4417	032240	005004				CLR	R4	; R4 CONTAINS OFFSET
4418	032242	012702	033604			MOV	#XCNTAB+2, R2	; R2 POINTS TO XMIT COUNT LIST
4419	032246	062704	000013		3\$:	ADD	#13, R4	; INCREASE R4 BY 13
4420	032252	060422				ADD	R4, (R2)+	; MAKE COUNTS VARIABLE
4421	032254	022702	033620			CMP	#XCNTAB+16, R2	; DONE ALL ?
4422	032260	001372				BNE	3\$; BR IF NO
4423	032262	012702	033426			MOV	#RECBA+12, R2	; R2 POINTS TO REC BA LIST

```

4424 032266 005222          INC      (R2)+      ;MAKE THIS REC BA ODC
4425 032270 005222          INC      (R2)+      ;MAKE THIS REC BA ODD
4426 032272 005222          INC      (R2)+      ;MAKE THIS REC BA ODD
4427 032274 062702 000004  ADD      #4,R2      ;SKIP TO XMIT BA LIST
4428 032300 005222          INC      (R2)+      ;MAKE THIS XMIT BA ODD
4429 032302 005222          INC      (R2)+      ;MAKE THIS XMIT BA ODD
4430 032304 062702 000004  ADD      #4,R2      ;SKIP TO NEXT ODD BA
4431 032310 005222          INC      (R2)+      ;MAKE THIS XMIT BA ODD
4432 032312 012737 177772 034664  MOV      #-5,FLAG   ;MAKE FLAG NEGITIVE
4433 032320 000137 031676          JMP      CLRTAB     ;LOOP WITH VARIABLE COUNTS
4434 032324 152711 000146          ENDEX: BISB      #146,(R1) ;SHUT DOWN DMC
4435 032330 005737 034664          1$:  TST      FLAG   ;HAS INTERRUPT OCCURED?
4436 032334 001775          BEQ      1$        ;BR IF NO
4437 032336 012700 000003          MOV      #3,R0     ;BASE ADDRESS OFFSET
4438 032342 105760 035010          2$:  TSTB     BASE(R0) ;CHECK ERROR COUNT
4439 032346 001016          BNE      ENDEX2    ;BR IF ERROR
4440 032350 005200          INC      R0        ;BUMB INDEX
4441 032352 022700 000011          CMP      #11,R0    ;DONE ALL ERROR COUNTERS YET?
4442 032356 001371          BNE      2$        ;BR IF NO
4443 032360 122760 000013 035010  CMPB     #13,BASE(R0) ;13 ERRORS L'CAUSE OF RESUME
4444 032366 001006          BNE      ENDEX2    ;BR IF NOT OK
4445 032370 005200          INC      R0        ;NEXT BASE TABLE LOCATION
4446 032372 122760 000013 035010  CMPB     #13,BASE(R0) ;13 ERRORS BECAUSE OF RESUME
4447 032400 001001          BNE      ENDEX2    ;BR IF NOT OK
4448 032402 104400          ENDEX1: SCOPE     ;SCOPE THIS TEST
4449 032404 113737 035013 001250  ENDEX2: MOVB     BASE+3,TEMP2 ;STORE ALL ODD ADDRESSES
4450 032412 113737 035015 001252  MOVB     BASE+5,TEMP3 ;FOR TYPEOUT
4451 032420 113737 035017 001254  MOVB     BASE+7,TEMP4
4452 032426 113737 035021 001256  MOVB     BASE+11,TEMP5
4453 032434 104017          HLT      17        ;NON ZERO ERROR COUNT
4454 032436 000761          BR       ENDEX1    ;GET OUT
4455 032440 022700 000017          ENDEX3: CMP      #17,R0 ;ALL DONE INPUT?
4456 032444 001600          BEQ      SCAN2     ;BR IF YES
4457 032446 005337 001416          SCAN1: DEC      TEMP ;DECREMENT DELAY COUNTER
4458 032452 001402          BEQ      1$        ;BR IF ZERO
4459 032454 000137 032020          JMP      SCAN      ;BR IF NOT DONE DELAY
4460 032460 005337 001250          1$:  DEC      TEMP2 ;DEC DELAY COUNT
4461 032464 001402          BEQ      2$        ;BR IF DONE DELAY
4462 032466 000137 032020          JMP      SCAN      ;BR IF NOT DONE
4463 032472 104014          2$:  HLT      14    ;ERROR HUNG
4464 032474 000742          BR       ENDEX1    ;GET OUT
4465
4466
4467          ;INPUT INTERRUPT SERVICE ROUTINE
4468 032476 022700 000017          IISR:  CMP      #17,R0 ;PROC. ERROR DONE?
4469 032502 001421          BEQ      12$       ;BR IF YES
4470 032504 005737 033412          TST      RESUME    ;IS THIS A RESUME INTERRUPT
4471 032510 001432          BEQ      8$        ;BR IF NO
4472 032512 032711 000002          BIT      #BIT1,(R1) ;CNTL OR BASE?
4473 032516 001407          BEQ      13$       ;BR IF CNTL I
4474 032520 012761 035010 000004  MOV      #BASE,4(R1) ;LOAD BASE ADDRESS
4475 032526 012761 010000 000006  MOV      #BIT12,6(R1) ;WITH RESUME BIT SET
4476 032534 000404          BR       12$       ;CONTINUE
4477 032536 005061 000006          13$:  CLR      6(R1)  ;SELECT FULL DUPLEX
4478 032542 005037 033412          CLR      RESUME    ;CLEAR RESUME FLAG
4479 032546 142711 000040          12$:  BICB     #40,(R1) ;CLEAR RQI

```

4480	032552	105711				TSTB	(R1)	; IS RDI GONE?
4481	032554	100776				BMI	.-2	; BR IF NO
4482	032556	005737	033412			TST	RESUME	; BASE OR CNTL I?
4483	032562	001403				BEQ	14\$; BR IF IT WAS CNTL I
4484	032564	152711	000041			BISB	#41, (R1)	; ASK FOR CNTL I
4485	032570	000002				RTI		; RETURN
4486	032572	105011			14\$:	CLRB	(R1)	; CLEAR BSEL 0
4487	032574	000002				RTI		; RETURN
4488	032576	005700			8\$:	TST	RO	; FIRST TIME HERE?
4489	032600	100006				BPL	7\$; LOAD BASE IF MINUS
4490	032602	012761	035010	000004		MOV	#BASE, 4(R1)	; SET UP BASE ADDRESS
4491	032610	005061	000006			CLR	6(R1)	; CLEAR COUNT
4492	032614	000434				BR	3\$; CONTINUE
4493	032616	001003			7\$:	BNE	1\$; CNTL I FULL DUPLEX IF 0
4494	032620	005061	000006			CLR	6(R1)	; SELECT FULL DUPLEX
4495	032624	000430				BR	3\$; CONTINUE
4496	032626	032700	000010		1\$:	BIT	#BIT3, RO	; XMIT?
4497	032632	001013				BNE	2\$; BR IF YES
4498	032634	000241				CLC		; CLEAR CARRY
4499	032636	006100				ROL	RO	; MAKE RO EVEN
4500	032640	016061	033414	000004		MOV	RECBA(RO), 4(R1)	; LOAD REC BUFFER
4501	032646	016061	033562	000006		MOV	RCNTAB(RO), 6(R1)	; LOAD COUNT
4502	032654	000241				CLC		; CLEAR CARRY
4503	032656	006000				ROR	RO	; GET RO BACK
4504	032660	000412				BR	3\$; CONTINUE
4505	032662	000241			2\$:	CLC		; CLEAR CARRY
4506	032664	006100				ROL	RO	; MAKE IT EVEN
4507	032666	016061	033414	000004		MOV	XMITBA(RO), 4(R1)	; LOAD XMIT BUFFER
4508	032674	016061	033562	000006		MOV	RCNTAB(RO), 6(R1)	; LOAD COUNT
4509	032702	000241				CLC		; CLEAR CARRY
4510	032704	006000				ROR	RO	; PUT IT BACK
4511	032706	142711	000040		3\$:	BICB	#40, (R1)	; CLEAR ROI
4512	032712	105711				TSTB	(R1)	; WAIT FOR
4513	032714	100776				BMI	.-2	; RDI TO GO AWAY
4514	032716	005200				INC	RO	; INC COUNT
4515	032720	001003				BNE	6\$; IF 0 ASK FOR CNTL I
4516	032722	152711	000041			BISB	#41, (R1)	; ASK FOR CNTL I
4517	032726	000002				RTI		; RETURN
4518	032730	022700	000017		6\$:	CMP	#17, RO	; DONE YET?
4519	032734	001411				BEQ	4\$; BR IF YES
4520	032736	032700	000010			BIT	#BIT3, RO	; XMIT?
4521	032742	001003				BNE	5\$; BR IF YES
4522	032744	152711	000044			BISB	#44, (R1)	; ASK FOR REC BA/CC
4523	032750	000002				RTI		; RETURN
4524	032752	152711	000040		5\$:	BISB	#40, (R1)	; ASK FOR XMIT BA/CC
4525	032756	000002				RTI		; RETURN
4526	032760	022737	000001	034664	4\$:	CMP	#1 FLAG	; 1 BYTE MESS?
4527	032766	001403				BEQ	15\$; BR IF YES
4528	032770	152711	000046			BISB	#46, (R1)	; FORCE PROC. ERROR
4529	032774	000002				RTI		; RETURN
4530	032776	105011			15\$:	CLRB	(R1)	; CLR SEL0
4531	033000	000002				RTI		; RETURN
4532								
4533								
4534								
4535	033002	032761	000001	000002	OISR:	BIT	#BIT0, 2(R1)	; IS THIS AN ERROR?

; OUTPUT INTERRUPT SERVICE ROUTINE

4536	033010	001461			BEQ	1\$: BR IF NO
4537	033012	005737	034664		TST	FLAG		: IS THIS SHUT DOWN INTERRUPT?
4538	033016	001006			BNE	9\$: BR IF NO
4539	033020	005237	034664		INC	FLAG		: YES MAKE FLAG NON-ZERO
4540	033024	022761	001000	000006	CMP	#BIT9,6(R1)		: SHUT DOWN BIT SET?
4541	033032	001516			BEQ	10\$: YES ALL IS OK
4542	033034	022700	000017		CMP	#17,R0	9\$:	: RESUME INTERRUPT?
4543	033040	001033			BNE	11\$: BR IF NO
4544	033042	022761	001000	000006	CMP	#BIT9,6(R1)		: PROC. ERROR BIT SET?
4545	033050	001027			BNE	11\$: BR IF NO
4546	033052	005200			INC	R0		: BUMP COUNTER (TO 20)
4547	033054	012711	040000		MOV	#BIT14,(R1)		: MASTER CLEAR DEVICE
4548	033060	032737	100000	001366	BIT	#BIT15,STAT1		: DMC OR KMC?
4549	033066	001405			BEQ	.+14		: BR IF DMC
4550	033070	012711	100000		MOV	#BIT15,(R1)		: SET RUN ON KMC
4551	033074	105227	000000		INCB	#0		: DELAY ON KMC
4552	033100	001375			BNE	.-4		
4553	033102	012737	177777	033412	MOV	#-1,RESUME		: SET RESUME FLAG
4554	033110	005711			TST	(R1)		: RUN SET?
4555	033112	100376			BPL	.-2		: BR IF NO
4556	033114	012761	000100	000002	MOV	#BIT6,2(R1)		: SET IEO
4557	033122	052711	034143		BIS	#4143,(R1)		: ASK FOR PORT(BASE REQ)
4558	033126	000002			RTI			: RETURN
4559	033130	016137	000004	001252	MOV	4(R1),TEMP3	11\$:	: SAVE FOR ERROR TYPEOUT
4560	033136	016137	000006	001254	MOV	6(R1),TEMP4		: SAVE FOR ERROR TYPEOUT
4561	033144	104016			HLT	16		: CNTL 0 ERROR
4562	033146	022626			CMP	(SP)+,(SP)+		: ADJUST STACK
4563	033150	000137	032402		JMP	ENDEX1		: GET OUT
4564	033154	032761	000004	000002	BIT	#BIT2,2(R1)	1\$:	: RECEIVE?
4565	033162	001046			BNE	2\$: BR IF YES
4566	033164	022761	033453	000004	CMP	#TBUF+1,4(R1)		: XMIT BA CORRECT?
4567	033172	001405			BEQ	4\$: BR IF OK
4568	033174	022761	033452	000004	CMP	#TBUF,4(R1)		: XMIT BA CORRECT?
4569	033202	001401			BEQ	4\$: BR IF YES
4570	033204	104014			HLT	14		: XMIT BA ERROR
4571	033206	005005			CLR	R5	4\$:	: R5 IS INDEX REG
4572	033210	026561	033602	000006	CMP	XCNTAB(R5),6(R1)	5\$:	: IS CHAR COUNT OK?
4573	033216	001406			BEQ	6\$: BR IF YES
4574	033220	062705	000002		ADD	#2,R5		: INC INDEX
4575	033224	022705	000016		CMP	#16,R5		: DONE LIST YET?
4576	033230	001367			BNE	5\$: BR IF NO
4577	033232	104014			HLT	14		: XMIT COUNT ERROR
4578	033234	016162	000004	033654	MOV	4(R1),XDNTAB(R2)	6\$:	: STORE XMIT DONE BA
4579	033242	062702	000002		ADD	#2,R2		: INC INDEX
4580	033246	016162	000006	033654	MOV	6(R1),XDNTAB(R2)		: STORE XMIT DONE CC
4581	033254	062702	000002		ADD	#2,R2		: INC INDEX
4582	033260	142761	000207	000002	BICB	#207,2(R1)		: CLEAR RDO
4583	033266	000002			RTI			: RETURN
4584	033270	105011			CLRB	(R1)	10\$:	: CLEAR SEL0
4585	033272	105061	000002		CLRB	2(R1)		: CLEAR SEL2
4586	033276	000002			RTI			: RETURN
4587	033300	012705	000002		MOV	#2,R5	2\$:	: SET UP R5 AS INDEX
4588	033304	026561	033414	000004	CMP	RECBA(R5),4(R1)		: COMPARE WITH LIST OF CORRECT BA'S
4589	033312	001406			BEQ	3\$: BR IF OK?
4590	033314	062705	000002		ADD	#2,R5		: INCREMENT R5
4591	033320	022705	000020		CMP	#20,R5		: END OF LIST?

4592	033324	001367			BNE	2\$+4		;BR IF NO
4593	033326	104014			HLT	14		;REC BA ERROR
4594	033330	005005			CLR	R5		;R5 IS INDEX
4595	033332	026561	033602	000006	3\$: CMP	XCNTAB(R5),6(R1)		;CHECK FOR CORRECT REC COUNT
4596	033340	001406			7\$: BEQ	8\$;BR IF YES
4597	033342	062705	000002		ADD	#2,R5		;INCREMENT R5
4598	033346	022705	000016		CMP	#16,R5		;END OF LIST?
4599	033352	001367			BNE	7\$;BR IF NOT
4600	033354	104014			HLT	14		;REC COUNT ERROR
4601	033356	016164	000004	033620	8\$: MOV	4(R1),RDNTAB(R4)		;STORE REC BA
4602	033364	062704	000002		ADD	#2,R4		;INC INDEX
4603	033370	016164	000006	033620	MOV	6(R1),RDNTAB(R4)		;STORE REC DONE CC
4604	033376	062704	000002		ADD	#2,R4		;INC INDEX
4605	033402	142761	000207	000002	BICB	#207,2(R1)		;CLEAR RDC
4606	033410	000002			RTI			;RETURN
4607								
4608								
4609								
4610								
4611	033412	000000						
4612	033414				RESUME:	0		
4613	033414	000017			RECBA:			
4614					XMITBA:	.BLKW 17		;REC & XMIT BA LIST
4615	033452				TBUFF:			;TRANSMIT DATA
4616	033452	000	001	002	.BYTE	0,1,2,3,4,5,6,7		
4617	033455	003	004	005				
4618	033460	006	007					
4619	033462	010	011	012	.BYTE	10,11,12,13,14,15,16,17		
4620	033465	013	014	015				
4621	033470	016	017					
4622	033472	020	021	022	.BYTE	20,21,22,23,24,25,26,27		
4623	033475	023	024	025				
4624	033500	026	027					
4625	033502	030	031	032	.BYTE	30,31,32,33,34,35,36,37		
4626	033505	033	034	035				
4627	033510	036	037					
4628	033512	040	041	042	.BYTE	40,41,42,43,44,45,46,47		
4629	033515	043	044	045				
4630	033520	046	047					
4631	033522	050	051	052	.BYTE	50,51,52,53,54,55,56,57		
4632	033525	053	054	055				
4633	033530	056	057					
4634	033532	060	061	062	.BYTE	60,61,62,63,64,65,66,67		
4635	033535	063	064	065				
4636	033540	066	067					
4637	033542	070	071	072	.BYTE	70,71,72,73,74,75,76,77		
4638	033545	073	074	075				
4639	033550	076	077					
4640	033552	100	101	102	.BYTE	100,101,102,103,104,105,106,107		
4641	033555	103	104	105				
4642	033560	106	107					
4643								
4644	033562	000010			RCNTAB:	.BLKW 10		;RECEIVE COUNT TABLE
4645	033602	000007			XCNTAB:	.BLKW 7		;TRANSMIT COUNT TABLE
4646								
4647	033620	000016			RDNTAB:	.BLKW 16		;RECEIVE DONE TABLE (BA/CC)


```

4648 033654 000016          XDNTAB: .BLKW 16          ;XMIT DONE TABLE (BA/CC)
4649                                     RBUFF:          ;RECEIVER BUFFERS
4650 033710                                     RBUFF1: .BLKB 106
4651 033710 000106          RBUFF2: .BLKB 106
4652 034016 000106          RBUFF3: .BLKB 106
4653 034124 000106          RBUFF4: .BLKB 106
4654 034232 000106          RBUFF5: .BLKB 106
4655 034340 000106          RBUFF6: .BLKB 106
4656 034446 000106          RBUFF7: .BLKB 106
4657 034554 000106          RBUFFE: 0          ;END OF RECEIVER BUFFERS
4658 034662 000000
4659                                     09500
4660                                     09600
4661                                     09700
4662                                     09800          ;BUFFER AREA
4663                                     09900          ;-----
4664                                     10000
4665 034664 000000          10100          FLAG: 0
4666 034666 000000          10200          TFLAG: 0
4667 034670 000000          10300          RFLAG: 0
4668 034672 000044          10400          TCOUNT: 44
4669 034674 041101 042103 043105          10500          TBUF: .ASCII/ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789/
4670 034702 044107 045111 046113
4671 034710 047115 050117 051121
4672 034716 052123 053125 054127
4673 034724 055131 030460 031462
4674 034732 032464 033466 034470
4675                                     10600          .EVEN
4676 034740 000044          10700          RCOUNT: 44
4677 034742 035010          10800          RBUF: .+.46
4678                                     10900          .EVEN
4679 035010 035410          11000          BASE: .+.256.
4680                                     00300
4681                                     00400
4682                                     00500          ;SUBROUTINES
4683                                     00600          ;-----
4684                                     00700
4685 035410          00800          CLRALL:
4686                                     00900          ;THIS SUBROUTINE CLEARS THE C&Z BITS AND THE BR
4687                                     01000
4688 035410 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4689 035412 000400          000400          ;BR+0
4690 035414 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4691 035416 063220          00400          ;SP(0)+BR
4692 035420 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4693 035422 060400          01600          060400          ;BR+SP(0)+BR
4694 035424 000207          01700          RTS          PC
4695                                     01800
4696                                     01900
4697 035426          02000          SETBRO:
4698                                     02100          ;THIS SUBROUTINE SETS BRO BIT
4699                                     02200
4700 035426 104414          ROMCLK          ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4701 035430 000401          02400          000401          ;BR+001
4702 035432 000207          02500          RTS          PC
4703                                     02600

```

4704			02700		
4705	035434		02800	SETBR1:	
4706			02900		; THIS SUBROUTINE SETS BR1 BIT
4707			03000		
4708	035434	104414		ROMCLK	; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4709	035436	000402	03200	000402	; BR+002
4710	035440	000207	03300	RTS PC	
4711			03400		
4712			03500		
4713	035442		03600	SETBR4:	
4714			03700		; THIS SUBROUTINE SETS BR4 BIT
4715			03800		
4716	035442	104414		ROMCLK	; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4717	035444	000420	04000	000420	; BR+020
4718	035446	000207	04100	RTS PC	
4719			04200		
4720			04300		
4721	035450		04400	SETBR7:	
4722			04500		; THIS SUBROUTINE SETS BR7 BIT
4723			04600		
4724	035450	104414		ROMCLK	; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4725	035452	000600	04800	000600	; BR+200
4726	035454	000207	04900	RTS PC	
4727			05000		
4728			05100		
4729	035456		05200	SETC:	
4730			05300		; THIS SUBROUTINE SETS THE C BIT
4731			05400		
4732	035456	104414		ROMCLK	; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4733	035460	000777	05600	000777	; BR+377
4734	035462	104414		ROMCLK	; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4735	035464	063220	05800	063220	; SP(0)+BR
4736	035466	104414		ROMCLK	; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4737	035470	060400	06000	060400	; BR+SP(0)+BR
4738	035472	000207	06100	RTS PC	
4739			06200		
4740			06300		
4741	035474		06400	SETZ:	
4742			06500		; THIS SUBROUTINE SETS THE Z BIT
4743			06600		
4744	035474	104414		ROMCLK	; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
4745	035476	000777	06800	000777	; BR+377
4746	035500	000207	06900	RTS PC	
4747			07000		
4748			07100		
4749	035502		07200	ROMDAT:	
4750			07300		; THIS SUBROUTINE LOADS R5 WITH EXPECTED ROM CONTENTS
4751			07400		; AND LOADS R4 WITH ACTUAL ROM CONTENTS
4752			07500		
4753	035502	017600	07600	MOV	2(SP), R0 ; INDEX FOR COMPARE
4754	035506	062716	07700	ADD	#2, (SP) ; ADJUST STACK
4755	035512	012711	07800	MOV	#BIT10, (R1) ; SET ROM0
4756	035516	016005	07900	MOV	ROMMAP(R0), R5 ; PUT "EXPECTED" IN R5
4757	035522	016104	08000	MOV	6(R1), R4 ; PUT "FOUND" IN R4
4758	035526	000207	08100	RTS	PC ; RETURN
4759			08200		

4760	035530			08300	RAMDAT:				
4761				08400		; THIS SUBROUTINE LOADS R4 WITH THE LOWEST			
4762				08500		; 8 BITS OF THE CRAM PC.			
4763				08600					
4764	035530	017605	000000	08700	MOV	2(SP), R5	; GOOD DATA		
4765	035534	062716	000002	08800	ADD	#2, (SP)	; ADJUST STACK		
4766	035540	005011		08900	CLR	(R1)	; CLEAR BIT10		
4767	035542	052711	000400	09000	BIS	#BIT8, (R1)	; CLOCK INSTRUCTION IN CRAM THAT WAS		
4768				09100			; JUMPED TO, IT LOADS BR WITH ROM PC		
4769	035546	005011		09200	CLR	(R1)	; CLR BIT8		
4770	035550	104414			ROMCLK		; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304		
4771	035552	061225		09400	061225		; MOV BR TO PORT 5		
4772	035554	116104	000005	09500	MOV	5(R1), R4	; PUT "FOUND" IN R4		
4773	035560	000207		09600	RTS	PC	; RETURN		
4774				09700					
4775	035562			09800	WROM:				
4776				09900		; THIS SUBROUTINE WRITES THE ROMMAP INTO THE CRAM			
4777				10000					
4778	035562	032737	100000	001366	10100	BIT	#BIT15, STAT1	; BE SURE DMC HAS CRAM	
4779	035570	001420			10200	BEQ	2\$; SKIP IF NO CRAM	
4780	035572	005000			10300	CLR	RO	; RO=CRAM ADDRESS	
4781	035574	012702	011766		10400	MOV	#ROMMAP, R2	; R2 POINTS TO ROMMAP	
4782	035600	012711	002000		10500	1\$:	MOV	#BIT10, (R1)	; SET ROMO
4783	035604	010061	000004		10600	MOV	RO, 4(R1)	; LOAD CRAM ADDRESS	
4784	035610	012261	000006		10700	MOV	(R2)+, 6(R1)	; LOAD WORD TO BE WRITTEN	
4785	035614	052711	020000		10800	BIS	#BIT13, (R1)	; WRITE IT!	
4786	035620	005200			10900	INC	RO	; NEXT ADDRESS	
4787	035622	022700	002000		11000	CMF	#2000, RO	; DONE YET?	
4788	035626	001364			11100	BNE	1\$; BR IF NO	
4789	035630	005011			11200	CLR	(R1)	; CLEAR SEL0	
4790	035632	000207			11300	2\$:	RTS	PC	; RETURN
4791					11400				
4792					11500				
4793	035634				11600	MEMSET:			
4794					11700		; THIS SUBROUTINE LOADS CRAM WITH SPECIAL INSTRUCTIONS		
4795					11800		; FOR THE CRAM JUMP TEST. ALL CRAM LOCATIONS ARE LOADED		
4796					11900		; WITH INSTRUCTIONS THAT MOVE A 37 TO THE BR, EXCEPT THE		
4797					12000		; FOLLOWING CRAM ADDRESSES: 0, 1, 4, 7, 525, 1777. THESE LOCATIONS		
4798					12100		; CONTAIN INSTRUCTIONS WHICH LOAD THE BR WITH THE LOWEST		
4799					12200		; 8 BITS OF THAT CRAM ADDRESS.		
4800					12300				
4801	035634	005000			12400	CLR	RO	; RO = CRAM ADDRESS	
4802	035636	012711	002000		12500	1\$:	MOV	#BIT10, (R1)	; SET ROMO
4803	035642	010061	000004		12600	MOV	RO, 4(R1)	; LOAD CRAM ADDRESS	
4804	035646	012761	000437	000006	12700	MOV	#437, 6(R1)	; LOAD INSTRUCTION	
4805	035654	052711	020000		12800	BIS	#BIT13, (R1)	; WRITE INSTRUCTION IN CRAM	
4806	035660	005200			12900	INC	RO	; NEXT ADDRESS	
4807	035662	022700	002000		13000	CMF	#2000, RO	; DONE YET?	
4808	035666	001363			13100	BNE	1\$; BR IF NO	
4809	035670	005000			13200	CLR	RO	; INDEX REGISTER	
4810	035672	012711	002000		13300	2\$:	MOV	#BIT10, (R1)	; SET ROMO
4811	035676	016061	035732	000004	13400	MOV	CRAM(RO), 4(R1)	; LOAD CRAM ADDRESS IN SEL4	
4812	035704	016061	035746	000006	13500	MOV	INSTU(RO), 6(R1)	; LOAD INSTRUCTION TO BE WRITTEN	
4813	035712	052711	020000		13600	BIS	#BIT13, (R1)	; WRITE CRAM!	
4814	035716	005720			13700	TST	(RO)+	; NEXT	
4815	035720	022700	000014		13800	CMF	#14, RO	; DONE YET?	

4816	035724	001362		13900	BNE	2\$:BR IF NO
4817	035726	005011		14000	CLR	(R1)		:CLEAR ALL BITS
4818	035730	000207		14100	RTS	PC		:RETURN
4819				14200				
4820	035732	000000	000001	000004	14300	CRAMA:	.WORD	0,1,4,7,1777,525
4821	035740	000007	001777	000525				
4822	035746	000400		14400	INSTU:	000400		:BR+0
4823	035750	000401		14500		000401		:BR+1
4824	035752	000404		14600		000404		:BR+4
4825	035754	000407		14700		000407		:BR+7
4826	035756	000777		14800		000777		:BR+377
4827	035760	000525		14900		000525		:BR+125
4828				15000				
4829				15100				
4830	035762			15200	BASELD:			
4831				15300				:THIS SUBROUTINE LOADS THE DMC WITH A BASE ADDRESS
4832				15400				:AND PUTS DMC INTO FULL-DUPLEX MODE
4833				15500				
4834	035762	012711	040000	15600	MOV	#BIT14,(R1)		:MASTER CLEAR
4835	035766	032737	100000	001366	15700	BIT	#BIT15,STAT1	:CRAM?
4836	035774	001402		15800	BEG	+.6		:BR IF NO
4837	035776	012711	100000	15900	MOV	#BIT15,(R1)		:IF CRAM SET RUN
4838	036002	105227	000000	16000	INCB	#0		:DELAY
4839	036006	001375		16100	BNE	.-4		:BR IF NOT DONE DELAY
4840	036010	005711		16200	1\$:	TST	(R1)	:IS RUN SET?
4841	036012	100376		16300	BPL	1\$:BR IF NO
4842	036014	052711	004000	16400	BIS	#BIT11,(R1)		:SET LU LOOP
4843	036020	152711	000043	16500	BISB	#43,(R1)		:BASE REQUEST
4844	036024	105711		16600	2\$:	TSTB	(R1)	:RDY I SET?
4845	036026	100376		16700	BPL	2\$:BR IF NO
4846	036030	012761	035010	000004	16800	MOV	#BASE,4(R1)	:LOAD BASE ADDRESS
4847	036036	005061	000006	16900	CLR	6(R1)		:CLEAR CC
4848	036042	142711	000040	17000	BICB	#40,(R1)		:CLEAR RQI
4849	036046	105711		17100	3\$:	TSTB	(R1)	:RDY I CLEAR?
4850	036050	100776		17200	BMI	3\$:BR IF NO
4851	036052	152711	000041		BISB	#41,(R1)		:ASK FOR CNTL I
4852	036056	105711			64\$:	TSTB	(R1)	:WAIT FOR RDI
4853	036060	100376			BPL	64\$:BR IF NOT SETY
4854	036062	005061	000006		CLR	6(R1)		:SET FULL DUP. FX
4855	036066	142711	000040		BICB	#40,(R1)		:CLEAR RQI
4856	036072	105711			65\$:	TSTB	(R1)	:RDI UP?
4857	036074	100776			BMI	65\$:BR IF YES
4858	036076	000207		17400	RTS	PC		:RETURN
4859				17500				
4860	036100			17600	BASELH:			
4861				17700				:THIS SUBROUTINE LOADS THE DMC WITH A BASE ADDRESS
4862				17800				:AND PUTS DMC INTO HALF-DUPLEX MODE
4863				17900				
4864	036100	012711	040000	18000	MOV	#BIT14,(R1)		:MASTER CLEAR
4865	036104	032737	100000	001366	18100	BIT	#BIT15,STAT1	:CRAM?
4866	036112	001402		18200	BEG	+.6		:BR IF NO
4867	036114	012711	100000	18300	MOV	#BIT15,(R1)		:IF CRAM SET RUN
4868	036120	105227	000000	18400	INCB	#0		:DELAY
4869	036124	001375		18500	BNE	.-4		:BR IF NOT DONE DELAY
4870	036126	005711		18600	1\$:	TST	(R1)	:IS RUN SET?
4871	036130	100376		18700	BPL	1\$:BR IF NO

4872	036132	052711	004000		18800	BIS	#BIT11,(R1)	;SET LU LOOP
4873	036136	152711	000043		18900	BISB	#43,(R1)	;BASE REQUEST
4874	036142	105711			19000	2\$: TSTB	(R1)	;RDY I SET?
4875	036144	100376			19100	BPL	2\$;BR IF NO
4876	036146	012761	035010	000004	19200	MOV	#BASE,4(R1)	;LOAD BASE ADDRESS
4877	036154	005061	000006		19300	CLR	6(R1)	;CLEAR CC
4878	036160	142711	000040		19400	BICB	#40,(R1)	;CLEAR RQI
4879	036164	105711			19500	3\$: TSTB	(R1)	;RDY I CLEAR?
4880	036166	100776			19600	BMI	3\$;BR IF NO
4881	036170	152711	000041			BISB	#41,(R1)	;ASK FOR CNTL I
4882	036174	105711				64\$: TSTB	(R1)	;WAIT FOR RDI
4883	036176	100376				BPL	64\$;BR IF NOT SETY
4884	036200	012761	002000	000006		MOV	#BIT10,6(R1)	;SET HALF DUPLEX
4885	036206	142711	000040			BICB	#40,(R1)	;CLEAR RQI
4886	036212	105711				65\$: TSTB	(R1)	;RDI UP?
4887	036214	100776				BMI	65\$;BR IF YES
4888	036216	000207			19800	RTS	PC	;RETURN
4889					19900			
4890	036220				20000	RFRELD:		
4891					20100			;THIS SUBROUTINE LOADS THE DMC WITH A RECEIVE BA/CC
4892					20200			
4893	036220	152711	000044		20300	BISB	#44,(R1)	;REC BA/CC REQUEST
4894	036224	105711			20400	1\$: TSTB	(R1)	;RDY I SET?
4895	036226	100376			20500	BPL	1\$;BR IF NO
4896	036230	012561	000004		20600	MOV	(R5)+,4(R1)	;LOAD REC BA
4897	036234	012561	000006		20700	MOV	(R5)+,6(R1)	;LOAD REC CC
4898	036240	142711	000040		20800	BICB	#40,(R1)	;CLEAR RQI
4899	036244	105711			20900	2\$: TSTB	(R1)	;IS RDY I CLEAR
4900	036246	100776			21000	BMI	2\$;BR IF NO
4901	036250	000205			21100	RTS	R5	;RETURN
4902					21200			
4903	036252				21300	XFRELD:		
4904					21400			;THIS SUBROUTINE LOADS THE DMC WITH A TRANSMIT BA/CC
4905					21500			
4906	036252	152711	000040		21600	BISB	#40,(R1)	;XMIT BA/CC REQUEST
4907	036256	105711			21700	1\$: TSTB	(R1)	;RDY I SET?
4908	036260	100376			21800	BPL	1\$;BR IF NO
4909	036262	012561	000004		21900	MOV	(R5)+,4(R1)	;LOAD XMIT BA
4910	036266	012561	000006		22000	MOV	(R5)+,6(R1)	;LOAD XMIT CC
4911	036272	142711	000040		22100	BICB	#40,(R1)	;CLEAR RQI
4912	036276	105711			22200	2\$: TSTB	(R1)	;IS RDY I CLEAR
4913	036300	100776			22300	BMI	2\$;BR IF NO
4914	036302	000205			22400	RTS	R5	;RETURN
4915					00300			
036304	041777	040522	020115	00400	EM1:	.ASCIZ	<377>/CRAM DATA ERROR/	
036325	377	051103	046501	00500	EM2:	.ASCIZ	<377>/CRAM DUAL ADDRESSING ERROR/	
036361	377	051103	046517	00600	EM3:	.ASCIZ	<377>/CROM DATA ERROR/	
036402	045377	046525	020120	00700	EM4:	.ASCIZ	<377>/JUMP ERROR/	
036416	047777	052104	042440	00800	EM5:	.ASCIZ	<377>/ODT ERROR IN IBUS* REG10/	
036450	044777	050117	046440	00900	EM6:	.ASCIZ	<377>/IOP MAIN MEMORY TEST/	
036476	044777	050117	046440	01000	EM7:	.ASCIZ	<377>/IOP MAR TEST/	
036514	041377	020122	044522	01100	EM10:	.ASCIZ	<377>/BR RIGHT SHIFT TEST/	
036541	377	042522	042503	01200	EM11:	.ASCIZ	<377>/RECEIVE DATA ERROR/	
036565	377	051106	042505	01300	EM12:	.ASCIZ	<377>/FREE RUNNING ERROR/	
036611	377	047503	052116	01400	EM13:	.ASCIZ	<377>/CONTROL OUT ERROR/	
036634	044777	052116	051105	01500	EM14:	.ASCIZ	<377>/INTERNAL DDCMP ERROR COUNTS NON ZERO/	

036702	042777	050130	041505	01600				
036734	042777	050130	041505	01700	DH1:	.ASCIZ	<377>/EXPECTED	FOUND ADDRESS/
036755	377	051440	046105	01800	DH2:	.ASCIZ	<377>/EXPECTED	FOUND/
036776	041377	051501	025505	01900	DH3:	.ASCIZ	<377>/SEL4	SEL6/
				02000	DH4:	.ASCIZ	<377>/BASE+3 THRU	BASE+12 /
				02100				
				02200				
				02300	.EVEN			
037024	000003			02400	DT1:	3		
037026	006	004		02500		.BYTE	6,4	
037030	001264			02600		SAVR2		
037032	006	004		02700		.BYTE	6,4	
037034	001270			02800		SAVR4		
037036	004	002		02900		.BYTE	4,2	
037040	001260			03000		SAVR0		
037042	000003			03100	DT2:	3		
037044	006	004		03200		.BYTE	6,4	
037046	001272			03300		SAVR5		
037050	006	004		03400		.BYTE	6,4	
037052	001270			03500		SAVR4		
037054	004	002		03600		.BYTE	4,2	
037056	001264			03700		SAVR2		
037060	000003			03800	DT3:	3		
037062	006	004		03900		.BYTE	6,4	
037064	001272			04000		SAVR5		
037066	006	004		04100		.BYTE	6,4	
037070	001270			04200		SAVR4		
037072	004	002		04300		.BYTE	4,2	
037074	001252			04400		TEMP3		
037076	000002			04500	DT4:	2		
037100	003	007		04600		.BYTE	3,7	
037102	001272			04700		SAVR5		
037104	003	002		04800		.BYTE	3,2	
037106	001270			04900		SAVR4		
037110	000002			05000	DT5:	2		
037112	006	004		05100		.BYTE	6,4	
037114	001272			05200		SAVR5		
037116	006	002		05300		.BYTE	6,2	
037120	001270			05400		SAVR4		
037122	000003			05500	DT6:	3		
037124	003	010		05600		.BYTE	3,10	
037126	001272			05700		SAVR5		
037130	003	004		05800		.BYTE	3,4	
037132	001270			05900		SAVR4		
037134	004	002		06000		.BYTE	4,2	
037136	034664			06100		FLAG		
037140	000003			06200	DT7:	3		
037142	003	010		06300		.BYTE	3,10	
037144	001272			06400		SAVR5		
037146	003	004		06500		.BYTE	3,4	
037150	001270			06600		SAVR4		
037152	004	002		06700		.BYTE	4,2	
037154	001264			06800		SAVR2		
037156	000003			06900	DT10:	3		
037160	003	007		07000		.BYTE	3,7	
037162	001272			07100		SAVR5		
037164	003	004				.BYTE	3,4	

037166	001270		07200	SAVR4	
037170	006	002	07300	.BYTE	6.2
037172	001252		07400	TEMP3	
037174	000902		07500	DT11: 2	
037176	006	004	07600	.BYTE	6.4
037200	001252		07700	TEMP3	
037202	006	002	07800	.BYTE	6.2
037204	001254		07900	TEMP4	
037206	000010		08000	DT12: 10	
037210	003	002	08100	.BYTE	3.2
037212	001250		08200	TEMP2	
037214	003	002	08300	.BYTE	3.2
037216	035014		08400	BASE+4	
037220	003	002	08500	.BYTE	3.2
037222	001252		08600	TEMP3	
037224	003	002	08700	.BYTE	3.2
037226	035016		08800	BASE+6	
037230	003	002	08900	.BYTE	3.2
037232	001254		09000	TEMP4	
037234	003	002	09100	.BYTE	3.2
037236	035020		09200	BASE+10	
037240	003	002	09300	.BYTE	3.2
037242	001256		09400	TEMP5	
037244	003	002	09500	.BYTE	3.2
037246	035022		09600	BASE+12	
			09700		
037250			09800	.ERRTAB:	
037250	000000		09900	0	
037252	000000		10000	0	
037254	000000		10100	0	
037256	036304		10200	EM1	
037260	036702		10300	DH1 ;HLT	1
037262	037024		10400	DT1	
037264	036325		10500	EM2	
037266	036702		10600	DH1 ;HLT	2
037270	037024		10700	DT1	
037272	036304		10800	EM1	
037274	036702		10900	DH1 ;HLT	3
037276	037042		11000	DT2	
037300	036361		11100	EM3	
037302	036702		11200	DH1 ;HLT	4
037304	037060		11300	DT3	
037306	036402		11400	EM4	
037310	036734		11500	DH2 ;HLT	5
037312	037076		11600	DT4	
037314	036402		11700	EM4	
037316	036734		11800	DH2 ;HLT	6
037320	037110		11900	DT5	
037322	036416		12000	EM5	
037324	036734		12100	DH2 ;HLT	7
037326	037076		12200	DT4	
037330	036450		12300	EM6	
037332	036702		12400	DH1 ;HLT	10
037334	037122		12500	DT6	
037336	036476		12600	EM7	
037340	036702		12700	DH1 ;HLT	11

037342	037140	12800	DT7		
037344	036514	12900	EM10		
037346	036734	13000	DH2	:HLT	12
037350	037076	13100	DT4		
037352	036541	13200	EM11		
037354	036702	13300	DH1	;HLT	13
037356	037156	13400	DT10		
037360	036565	13500	EM12		
037362	000000	13600	0	:HLT	14
037364	000000	13700	0		
037366	036565	13800	EM12		
037370	036734	13900	DH2	:HLT	15
037372	037110	14000	DT5		
037374	036611	14100	EM13		
037376	036755	14200	DH3	:HLT	16
037400	037174	14300	DT11		
037402	036634	14400	EM14		
037404	036776	14500	DH4	:HLT	17
037406	037206	14600	DT12		
		14700			
		14800			
037410		14900	CORMAX:		
	000001	15400	.END		

N03

EM2	036325	4915#																		
EM3	036361	4915#																		
EM4	036402	4915#																		
EM5	036416	4915#																		
EM6	036450	4915#																		
EM7	036476	4915#																		
ENDEX	032324	4409	4434#																	
ENDEX1	032402	4307	4448#	4454	4464	4563														
ENDEX2	032404	4439	4444	4447	4449#															
ENDEX3	032440	4371	4455#																	
ERCT00	001704	367#																		
ERCT01	001710	370#																		
ERCT02	001714	373#																		
ERCT03	001720	376#																		
ERCT04	001724	379#																		
ERCT05	001730	382#																		
ERCT06	001734	385#																		
ERCT07	001740	388#																		
ERCT10	001744	391#																		
ERCT11	001750	394#																		
ERCT12	001754	397#																		
ERCT13	001760	400#																		
ERCT14	001764	403#																		
ERCT15	001770	406#																		
ERCT16	001774	409#																		
ERCT17	002000	412#																		
ERR	002612	589	599#	603																
ERRCNT	001232	163#	727	754	1066*	1280*														
ERRFLG	001325	202#	483#	713#	775*	1016*	1029	1043*	1101*											
ERRMSG	005100	1026#	1044	1047#																
ERRPC	002702	605	621#																	
ERTAB0	005224	1041	1075#																	
EXIT =	000205	96#																		
EXITER	005160	1061	1066#																	
FLAG	034664	1834*	1838	1839	1860*	1861	1880*	1885	1886	1908*	1909	1929*	1930	1933						
		1934	1952*	1953	1956*	1957	1960	1961	1976*	1977	4339*	4350	4361	4369						
		4383	4405	4407*	4410	4415*	4432*	4435	4526	4537	4539*	4665#	4915							
FLOAT	002536	582#	588																	
FY	002562	590#	594																	
HALTS	005130	1012	1058#																	
HILIM	004274	855#	882	900#																
ICOUNT	001222	159#	773	778#																
IISR	032476	4318	4468#																	
INBUF	007256	825	861	1180#																
INCHAR	007560	1206	1234#																	
INIFLG	001324	201#	509	529	536*															
INSTER=	104404	223#	876																	
INSTR =	104403	221#	1304	1356	1369	1378	1445	1454												
INSTR2	004074	832	844#																	
INSTU	035746	4812	4822#																	
INTTY	011734	1389	1406	1417	1433	1612#														
KMCM	007220	618	1165#																	
LIMITS	004222	871	882#																	
LINE	006720	1165#	1446																	
LOBITS	004300	857#	886	902#	903															
LOCK	001220	158#	777#	793	795	1035	1675#	1709#	1747#	1768#	1794#	1829#	1875#	1924#						

CROSS REFERENCE TABLE -- USER SYMBOLS

PACT05	001726	381#																			
PACT06	001732	384#																			
PACT07	001736	387#																			
PACT10	001742	390#																			
PACT11	001746	393#																			
PACT12	001752	396#																			
PACT13	001756	399#																			
PACT14	001762	402#																			
PACT15	001766	405#																			
PACT16	001772	408#																			
PACT17	001776	411#																			
PARAM =	104405	225#	1306	1358	1371	1380	1447	1456													
PARAM1	004142	860#	877																		
PARBIT=	040000	96#																			
PARERR	004216	863	865	867	876#	883	885	887													
PASCNT	001230	162#	714*	715	726	751	1279*														
PERFOR=	004537	96#																			
PFTAB	005332	1100	1106#																		
POPPO =	012600	72#	1065																		
POP1SP=	005726	70#																			
POP2SP=	022626	74#																			
PRI0	006451	1165#	1388																		
PS =	177776	63#	478*	691*	1574*	1584*	4311*	4369*													
PUSHRO=	010046	71#	1062																		
PUSHIS=	005746	69#																			
PUSH2S=	024646	73#																			
QV.FLG	001327	204#	484*	730*	770																
RAMDAT	035530	2947	2960	2973	3008	3020	3032	3068	3081	3094	3130	3143	3156	3192							
		3205	3218	3254	3267	3280	3316	3329	3342	3378	3391	3404	3440	3453							
		3466	3502	3515	3528	3564	3577	3590	3626	3639	3652	3688	3701	3714							
		3750	3763	3776	4760#																
RBUF	034742	382	3871	3932	3940	4038	4270	4677#													
RBUFF	033710	4327	4650#																		
RBUFFE	034662	4348	4658#																		
RBUFF1	033710	4651#																			
RBUFF2	034016	4652#																			
RBUFF3	034117	4653#																			
RBUFF4	034218	4654#																			
RBUFF5	034319	4655#																			
RBUFF6	034420	4656#																			
RBUFF7	034521	4657#																			
RCNTAB	033212	4335	4337	4501	4538	4644#															
RCOUNT	034740	3810	3872	3935	3938	4676#															
RONTAB	033620	4346	4379	4402	4631#	4603*	4647#														
RECBA	033414	4423	4500	4588	4632#																
RESREG	005126	1054	1057#																		
RESTAR	005252	1086	1092#																		
RESTR	003444	729	733	741#																	
RESUME	033412	4345*	4470	4478*	4482	4553*	4611#														
RES05 =	104407	229#	1057																		
RETURN	001214	156#	494*	700*	704	741*	779*	783	1071*	1073	1105	1320*	1330*	1332							
RFLAG	034670	3817*	3925	3928*	3953	4667#															
RFELD	036220	4037	4132	4269	4890#																
ROMCLK=	104414	239#	1112	1115	1152	1157	1642	1644	1646	1654	1656	1840	1842	1845							
		1847	1849	1897	1889	1892	1894	1896	1935	1937	1940	1942	1944	1962							
		1964	1966	1968	1997	2000	2006	2009	2039	2042	2054	2060	2068	2070							

E04

CZDMG MACY11 27(1006) 07-MAR-77 15:52 PAGE 105
 CZDMGB.P11 07-MAR-77 11:21

PAGE: 0250

CROSS REFERENCE TABLE -- USER SYMBOLS

STAT1	001366	252*	1276*	1677	1712	1750	1797	1832	1978	1927	1994	2036	2097	2146
		2202	2255	2311	2367	2423	2479	2535	2592	2649	2706	2763	2820	2877
		2938	3000	3059	3121	3183	3245	3307	3369	3431	3493	3555	3617	3679
		3741	3802	3807	3819	3831	3972	3977	4028	4033	4077	4082	4123	4128
		4172	4177	4216	4221	4260	4265	4303	4308	4312	4341	4548	4778	4835
		4865												
STAT2	001370	253*	1277*											
STAT3	001372	254*	1278*	3804	3974	4030	4079	4125	4174	4218	4262	4305		
STRTSW	001236	169*	515*	518*	519	521	531	533	646	692	701	1298	1322*	1352
		1558												
SVOS	004310	912*												
SWFLG	007556	482*	804	1199*	1226*	1232*								
SWMES	007107	1165*	1202											
SWMES1	007117	1165*	1205											
SWR	001202	143*	499*	501	505*	515	651	656	761	768	791	806	1006	1011
		1060	1067	1069	1130	1190	1225*							
SWREG	000176	129*	505	1190	1245									
SW00	= 000001	45*	519	1352	1558									
SW01	= 000002	44*	701	1298	1322									
SW02	= 000004	43*												
SW03	= 000010	42*	646											
SW04	= 000020	41*												
SW05	= 000040	40*												
SW06	= 000100	39*	1130											
SW07	= 000200	38*												
SW08	= 000400	37*	1067											
SW09	= 001000	36*	791											
SW10	= 002000	35*	1069											
SW11	= 004000	34*	768											
SW12	= 010000	33*	806	1006										
SW13	= 020000	32*	1011											
SW14	= 040000	31*												
SW15	= 100000	30*												
TBUF	034674	3887	3917	3939	3982	4041	4136	4273	4669*					
TBUFF	033452	4332	4566	4568	4615*									
TCOUNT	034672	3888	3920	4668*										
TEMP	001416	272*	950	1096*	1097*	1138*	1143*	1149*	1161*	3824*	3827*	3835*	3838*	3844*
		3847*	3865*	3868*	3874*	3877*	3881*	3884*	3890*	3893*	3896*	3900*	3986*	3989*
		4007*	4012*	4044*	4047*	4090*	4093*	4139*	4142*	4185*	4188*	4229*	4232*	4276*
		4279*	4366*	4457*										
TEMP1	001246	173*	539*	1167	1570*	1571*	3897*	3902*						
TEMP2	001250	174*	540*	1169	3905*	4367*	4449*	4460*	4915					
TEMP3	001252	175*	542*	563*	599	608*	1171	1361	1364	1464*	2113*	2115*	2116*	2117*
		2118*	3851*	3941*	4392*	4450*	4559*	4915						
TEMP4	001254	176*	543*	1173	1374	1377	1383	1386	1450	1453	1459	1462	3852*	4451*
		4560*	4915											
TEMP5	001256	177*	544*	1175	1355*	1368*	1556	4452*	4915					
TFLAG	034666	3816*	3910	3913*	3955	4325*	4336	4353	4412*	4414*	4666*			
TIMER	= 104416	243*												
TKCSR	001204	148*	764	827	1234	1612								
TKDBR	001206	149*	766	829	835	1192	1194	1236	1614					
TLAST	= 031432	1325	4680*											
TPCSR	001210	150*	811	833	1008	1237	1615							
TPDBR	001212	151*	813*	835*	1010*	1239*	1617*							
TRPOK	004636	996*												
TSTNO	001226	161*	493*	1080	1108	1309	1316	1318	1634*	1673*	1707*	1745*	1792*	1827*

CROSS REFERENCE TABLE -- USER SYMBOLS

		1873*	1922*	1990*	2031*	2092*	2141*	2197*	2250*	2306*	2362*	2418*	2474*	2530*
		2587*	2644*	2701*	2758*	2815*	2872*	2933*	2995*	3054*	3116*	3178*	3240*	3302*
		3364*	3426*	3488*	3550*	3612*	3674*	3736*	3798*	3968*	4024*	4073*	4119*	4168*
		4212*	4256*	4299*										
TST1	015766	1312	1330	1634*										
TST10	017216	1874	1922*											
TST11	017516	1923	1990*											
TST12	017632	1991	2031*											
TST13	020040	2032	2092*											
TST14	020230	2093	2141*											
TST15	020420	2142	2197*											
TST16	020574	2198	2250*											
TST17	020764	2251	2306*											
TST2	016100	1635	1673*											
TST20	021154	2307	2362*											
TST21	021344	2363	2418*											
TST22	021534	2419	2474*											
TST23	021724	2475	2530*											
TST24	022114	2531	2587*											
TST25	022304	2588	2644*											
TST26	022474	2645	2701*											
TST27	022664	2702	2758*											
TST3	016214	1674	1707*											
TST30	023054	2759	2815*											
TST31	023244	2816	2872*											
TST32	023434	2873	2933*											
TST33	023630	2934	2995*											
TST34	024010	2996	3054*											
TST35	024204	3055	3116*											
TST36	024400	3117	3178*											
TST37	024574	3179	3240*											
TST4	016336	1708	1745*											
TST40	024770	3241	3302*											
TST41	025164	3303	3364*											
TST42	025360	3365	3426*											
TST43	025554	3427	3488*											
TST44	025750	3489	3550*											
TST45	026144	3551	3612*											
TST46	026340	3613	3674*											
TST47	026534	3675	3736*											
TST5	016516	1746	1792*											
TST50	026730	3737	3798*											
TST51	027742	3799	3968*											
TST52	030170	3969	4024*											
TST53	030362	4025	4073*											
TST54	030544	4074	4119*											
TST55	030736	4120	4168*											
TST56	031114	4169	4212*											
TST57	031272	4213	4256*											
TST6	016636	1793	1827*											
TST60	031432	4257	4299*	4680										
TST7	017024	1828	1873*											
TTST	003522	695*	696*	698*	699*	762*								
TWOSYN=	010000	96*												
TYPDAT	005114	1032	1050	1053*										
TYPE =	104402	219*	513	525	537	601	606	614	617	648	653	694	703	716

CROSS REFERENCE TABLE -- USER SYMBOLS

	717	719	721	723	810	823	840	933	973	1032	1034	1037	1038
	1040	1042	1046	1051	1099	1202	1205	1257	1303	1321	1327	1365	1387
	1401	1404	1411	1415	1424	1431	1438	1547					
TYPMSG	005014												
VEC	006430												
VECMAP	011456												
WHICH	011450												
WRDCNT	004616												
WRKO.F	005102												
WROM	035562												
XBX	004706												
XCNTAB	033602												
XCSR	003456												
XDNTAB	033654												
XERR	003500												
XFRELD	036252												
XHEAD	006126												
XMITBN	033414												
XPASS	003472												
XSTATQ	007230												
XTSTN	005232												
XVEC	003464												
X0 =	000110												
X1 =	000101												
X2 =	000102												
X3 =	000103												
X4 =	000104												
X5 =	000105												
X6 =	000106												
X7 =	000107												
ZERG	001300												
SCOD =	***** U												
SCRAP =	177777												
	186#												
	1#												
	1781#	1624#	1627	1630#	1664#	1667	1669#	1698#	1701	1703#	1735#	1738	1741#
	1983	1784	1788#	1818#	1821	1823#	1864#	1867	1869#	1912#	1915	1918#	1980#
	2193#	1986#	2020#	2023	2027#	2081#	2084	2088#	2130#	2133	2137#	2187#	2190
	2464#	2240#	2243	2246#	2296#	2299	2302#	2352#	2355	2358#	2408#	2411	2414#
	2693	2467	2470#	2520#	2523	2526#	2576#	2579	2583#	2633#	2636	2640#	2690#
	2929#	2697#	2747#	2750	2754#	2804#	2807	2811#	2861#	2864	2868#	2918#	2921
	3225#	2980#	2983	2991#	3039#	3042	3050#	3101#	3104	3112#	3163#	3166	3174#
	3476	3228	3236#	3287#	3290	3298#	3349#	3352	3360#	3411#	3414	3422#	3473#
	3732#	3484#	3535#	3538	3546#	3597#	3600	3608#	3659#	3662	3670#	3721#	3724
	4109#	3793#	3786	3794#	3958#	3961	3964#	4014#	4017	4020#	4063#	4066	4069#
	4289	4112	4115#	4158#	4161	4164#	4202#	4205	4208#	4246#	4249	4252#	4286#
	123	4295#											
SENDAD	003432												
SN =	000060												
	1712#	511	735#	1058									
	1825	1#	1624	1630	1637#	1664	1669	1671	1677#	1698	1703	1705	1711
	1980	1712#	1735	1741	1749	1750#	1781	1788	1790	1796	1797#	1818	1823
	2096	1825	1831	1832#	1864	1869	1871	1877	1878#	1912	1918	1920	1927#
	2246	1980	1986	1988	1993	1994#	2020	2027	2029	2035	2036#	2081	2088
	2357#	2096	2097#	2130	2137	2139	2145	2146#	2187	2193	2195	2201	2202#
	2528	2246	2248	2254	2255#	2296	2302	2304	2310	2311#	2352	2358	2360
	2690	2357#	2408	2414	2416	2422	2423#	2464	2470	2472	2478	2479#	2520
	2819	2528	2534	2535#	2576	2583	2585	2591	2592#	2633	2640	2642	2648
	2991	2690	2697	2699	2705	2706#	2747	2754	2756	2762	2763#	2804	2811
		2819	2820#	2861	2868	2870	2876	2877#	2918	2929	2931	2937	2938#
		2991	2993	2999	3000#	3039	3050	3052	3058	3059#	3101	3112	3114

020M3 MACY1: 27010061 07-MAR-77 15:52 PAGE 109
020MGB.F11 07-MAR-77 :1:21 CROSS REFERENCE TABLE -- USER SYMBOLS

PAGE: 0254

.TRPTA 001330 214# 998
.TYPE 003674 220 801#

DMEND	1#	705													
DMFRNT	1#														
HLT	75#	1652	1662	1689	1725	1762	1775	1809	1855	1902	1950	1974	2015	2052	2067
	2079	2119	2158	2171	2184	2213	2225	2237	2267	2280	2293	2323	2336	2349	2379
	2392	2405	2435	2448	2461	2491	2504	2517	2547	2560	2573	2604	2617	2630	2661
	2674	2687	2718	2731	2744	2775	2788	2801	2832	2845	2858	2889	2902	2915	2951
	2964	2977	3012	3024	3036	3072	3085	3098	3134	3147	3160	3196	3209	3222	3258
	3271	3284	3320	3333	3346	3382	3395	3408	3444	3457	3470	3506	3519	3532	3568
	3581	3594	3630	3643	3656	3692	3705	3718	3754	3767	3780	3829	3840	3853	3855
	3870	3879	3886	3895	3904	3907	3912	3916	3919	3922	3927	3931	3934	3937	3946
	3951	3993	3997	4005	4051	4055	4061	4097	4101	4107	4146	4150	4156	4190	4194
	4200	4234	4238	4244	4284	4396	4453	4463	4561	4570	4577	4593	4600		
\$AUTO	1#	549													
\$BRASH	1#	1624													
\$BUFE	1#	1177													
\$BYTE	1#	4616	4619	4622	4625	4628	4631	4634	4637	4640					
\$CKDAT	1#	4379													
\$COMP	1#														
\$SCRAM	1#	1664	1698												
\$SCRAMD	1#	1735													
\$CYCLE	1#	1246													
\$DATAF	1#	3783													
\$EOP	1#	705													
\$EXER	1#	4286													
\$FD	1#	3857	4851												
\$FINI	1#	4680													
\$GETPA	1#														
\$HALF	1#	4246													
\$HD	1#	4881													
\$HEACE	1#														
\$IOPOD	1#	2020													
\$JUMP	1#	2130	2187	2240	2296	2352	2408	2464	2520	2576	2633	2690	2747	2804	2861
	2918	2980	3039	3101	3163	3225	3287	3349	3411	3473	3535	3597	3659	3721	
\$LSTDA	1#	4014													
\$MARHI	1#														
\$MEMFL	1#	1932	1878												
\$MEMD	1#	1864													
\$MEM1	1#	1818													
\$MEM2	1#	1912													
\$MEM3	1#	1980													
\$MOCK	1#														
\$MSG	1#	1165													
\$NONEX	1#	4063	4109												
\$ORUN	1#	3958													
\$PFAIL	1#	1081													
\$PROC	1#	4158													
\$PROC1	1#	4202													
\$QUEST	1#	1356	1369	1378	1445	1454									
\$RAMCL	1#	1109													
\$RCLK	1#	1112	1115	1152	1157	1642	1644	1646	1653	1656	1840	1842	1845	1847	1849
	1887	1889	1892	1894	1896	1935	1937	1940	1942	1944	1962	1964	1966	1968	1997
	2000	2006	2009	2039	2042	2054	2060	2068	2070	2072	2106	2150	2152	2163	2165
	2176	2178	2205	2207	2217	2219	2229	2231	2259	2261	2272	2274	2285	2287	2315
	2317	2328	2330	2341	2343	2371	2373	2384	2386	2397	2399	2427	2429	2440	2442
	2453	2455	2483	2485	2496	2498	2509	2511	2539	2541	2552	2554	2565	2567	2596
	2598	2609	2611	2622	2624	2653	2655	2666	2668	2679	2681	2710	2712	2723	2725

CROSS REFERENCE TABLE -- MACRO NAMES

	2736	2738	2767	2769	2780	2782	2793	2795	2824	2826	2837	2839	2850	2852	2881
	2883	2894	2896	2907	2909	2943	2945	2956	2958	2969	2971	3004	3006	3016	3018
	3028	3030	3064	3066	3077	3079	3090	3092	3126	3128	3139	3141	3152	3154	3198
	3190	3201	3203	3214	3216	3250	3252	3263	3265	3276	3278	3312	3314	3325	3327
	3338	3340	3374	3376	3387	3389	3400	3402	3436	3438	3449	3451	3462	3464	3498
	3500	3511	3513	3524	3526	3560	3562	3573	3575	3586	3588	3622	3624	3635	3637
	3648	3650	3684	3686	3697	3699	3710	3712	3746	3748	3759	3761	3772	3774	4688
	4690	4692	4700	4708	4716	4724	4732	4734	4736	4744	4770				
\$RDROM	1#	1781													
\$ROMRD	1#	2081													
\$SCOPE	1#	755													
\$SETUP	1#	3972	4028	4077	4123										
\$SIMBC	1#														
\$SKIPT	1#	3802	3972	4028	4077	4123	4172	4216	4260	4303					
\$SOFTC	1#	1185													
\$TRPDE	1#	215	217	219	221	223	225	227	229	231	233	235	237	239	241
	243														
\$TSTN	1#	1632	1671	1705	1743	1790	1825	1871	1920	1988	2029	2090	2139	2195	2248
	2304	2360	2416	2472	2528	2585	2642	2699	2756	2813	2870	2931	2993	3052	3114
	3176	3238	3300	3362	3424	3486	3548	3610	3672	3734	3796	3966	4022	4071	4117
	4166	4210	4254	4297											
\$VARIA	1#	134													
\$XZ	1#	1624	1630	1664	1669	1698	1703	1735	1741	1781	1788	1818	1823	1864	1969
	1912	1918	1980	1986	2020	2027	2081	2088	2130	2137	2187	2193	2240	2246	2296
	2302	2352	2358	2408	2414	2464	2470	2520	2526	2576	2583	2633	2640	2690	2697
	2747	2754	2804	2811	2861	2868	2918	2929	2980	2991	3039	3050	3101	3112	3163
	3174	3225	3236	3287	3298	3349	3360	3411	3422	3473	3484	3535	3546	3597	3608
	3659	3670	3721	3732	3783	3794	3958	3964	4014	4020	4063	4069	4109	4115	4158
	4164	4202	4208	4246	4252	4286	4295								

. ABS. 037410. 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DZDMGB, DZDMGB.SEG/SOL/CRF/NL: TOC+IPLUTL, DZDMGB
RUN-TIME: 16 22 1 SECONDS
RUN-TIME RATIO: 809/40=20.2
CORE USED: 29K (57 PAGES)