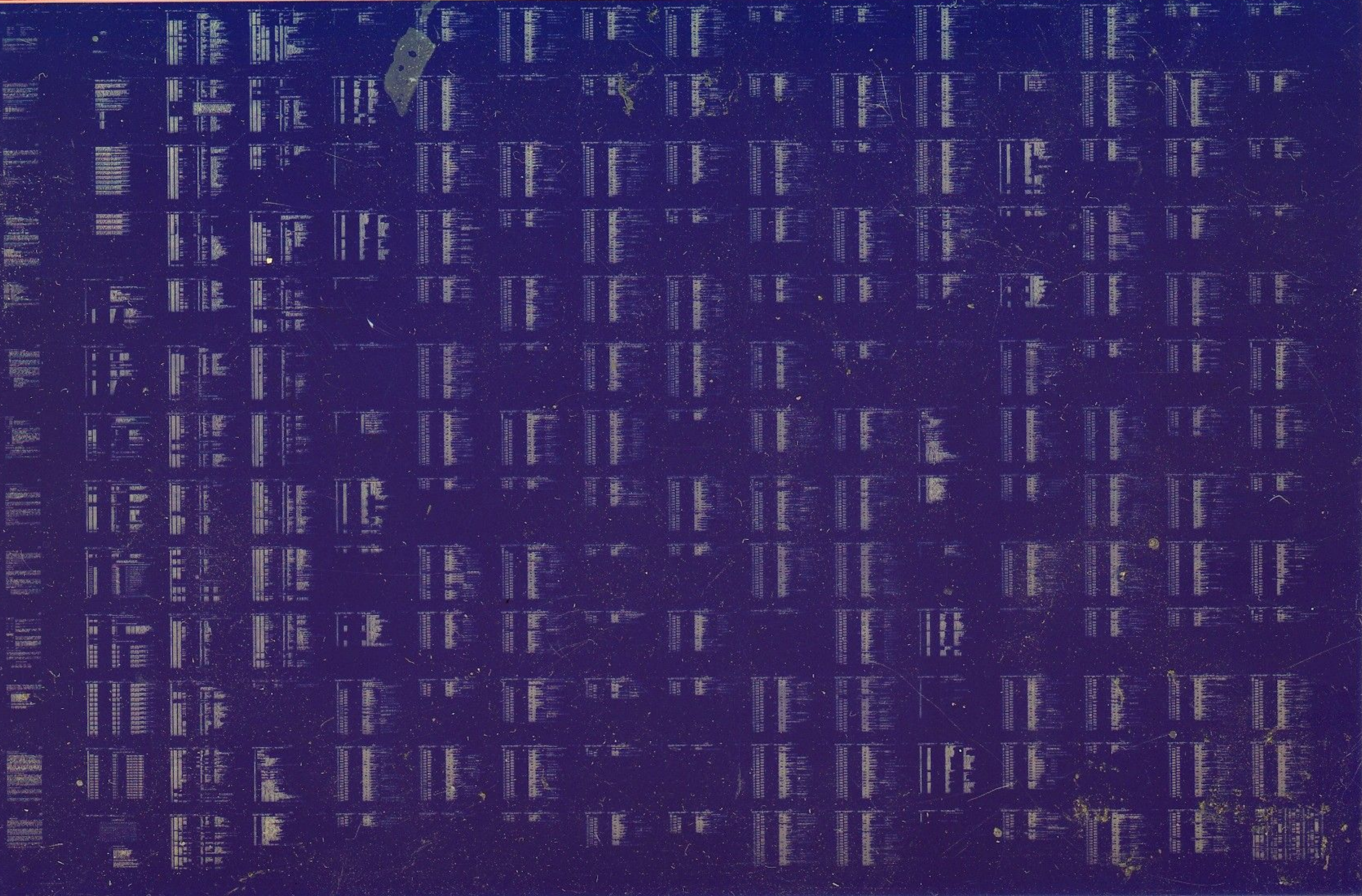


DMC11

CROM AND JUMP TESTS
MD-11-DZDMG-C

EP-DZDMG-C-DL-A
COPYRIGHT © 76-77
FICHE 1 OF 2

AUG 1977
digital
MADE IN USA



DMC11

CROM AND JUMP TESTS
MD-11-DZDMG-C

EP-DZDMG-C-DL-A
COPYRIGHT © 76-77
FICHE 2 OF 2

AUG 1977
digital
MADE IN USA

[Faint, illegible text visible through the paper, likely bleed-through from the reverse side.]

B01

EDF10ZDMFBSEQ
PDP10 PAGE: 0001

00010000

770712

PDP10 411

HDR10ZDMGCSEQ

00010000

770712

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZDMG-C-D
PRODUCT NAME: DMC11 CROM AND JUMP TESTS
DATE: MAY 1977
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, 1977 by Digital Equipment Corporation

1. ABSTRACT

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

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

DZDMG tests the DMC11-AR and DMC11-AL micro-processors (M8200-YA M8200-YB). It performs jump tests on the micro-processor and verifies the control ROM of the M8200. This diagnostic will not run on a KMC (M8204), however it is possible to load the KMC CRAM with the DMC micro-code. See test 2 for details.

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

NOTE: Additional diagnostics may be added in the future.

The five diagnostics are:

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

2. REQUIREMENTS

2.1 EQUIPMENT

Any PDP11 family CPU (except an LSI-11) with minimum 8k memory
ASR 33 (or equivalent)
DMC11-AR (M8200-YA) or an DMC11-AL (M8200-YB)

2.2 STORAGE

Program will use all BK 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 PROCEDURE

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

MAP OF DMC11 STATUS

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

The program will type 'R' and proceed to run the diagnostic. The above is only an example. This would indicate the status table starting at add. 1500 in the program. In this example the table contains the information and status of two DMC11'S. THE STATUS TABLE MUST BE VERIFIED BY THE USER IF AUTO SIZING IS DONE. For information of status table see section B.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 MB201 TYPE "1", IF MB202 TYPE "2"?1
 IS THE LOOP 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
D: Set a switch for each DMC desired active.
EXAMPLE: If you have 4 DMC's but only want to run the first and the last set SWR bits 0 and 3 = 1. PRESS CONTINUE
E: Number (IF VALID) will be in data lights (excluding 11/05)
F: Set with any other switch settings desired. PRESS CONTINUE.

4.1.3 DYNAMIC SWITCHES

ERROR SWITCHES

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

SCOPE SWITCHES

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

4.2 STARTING ADDRESS

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

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

5. OPERATING PROCEDURE

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

5.2 PROGRAM AND OR OPERATOR ACTION

The typical approach should be

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

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

5. ERRORS

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

6.2 ERROR RECOVERY

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

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

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

7.2 OPERATING RESTRICTIONS

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

7.3 HARDWARE CONFIGURATION RESTRICTIONS

DMC11(M8200)- Jumper W1 must be in, and switch 7 of E76 must be in the OFF position.

KMC(M8204)- Jumper W1 must be in.

8. MISCELLANEOUS

8.1 EXECUTION TIME

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

8.2 PASS COMPLETE

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

```
END PASS 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:.

8.4 KEY LOCATIONS

RETURN (1214) Contains the address where program will return when iteration count is reached or if loop on test is asserted.

NEXT (1216) Contains the address of the next test to be performed.

TSTNO (1226) Contains the number of the test now being performed.

RUN (1316) The bit in 'RUN' always points to the DMC11 currently being tested. EXAMPLE: (RUN) 1302/0000000001000000 Means that DMC11 no.06 is the DMC11 now running.

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

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

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

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

8.4A 'STATUS TABLE' (1500-1640)

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

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

MAP OF DMC11 STATUS

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

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

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

8.5 METHOD OF AUTO SIZING

8.5.1 FINDING THE CONTROL STATUS REGISTER.

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

8.5.2 FINDING THE VECTOR AND BR LEVEL

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

8.5 SOFTWARE SWITCH REGISTER

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

Control:

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

SWR=XXXXXX NEW?

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

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

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

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

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

DZDMG LST

B02

DECDOC VER 00.04 11-JUL-77 12:14 PAGE 01 PAGE: 0014

DOCUMENT

DZDMG LST

COPYRIGHT 1977
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

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

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

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

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

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

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

DMC11-AR (M8200-YA)

23-414A9
 23-415A9
 23-416A9
 23-417A9
 23-418A9
 23-419A9
 23-420A9
 23-421A9

DMC11-AL (M8200-YB)

23-392A9
 23-393A9
 23-394A9
 23-395A9
 23-396A9
 23-397A9
 23-398A9
 23-399A9

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

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

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

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

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

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

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

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

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

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

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

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

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

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

GENERAL DEFINATIONS AND EQUIVALENCIES

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

:REGISTER DEFINITIONS
:-----

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

:LOCATION EQUIVALENCIES
:-----

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

:INSTRUCTION DEFINITIONS
:-----

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

:BIT DEFINITIONS
:-----

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

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
(2)
(2)
137
138
139
140
141
142
143

000000

000024 005336
000026 000340
000030 004750
000032 000340
000034 004716
000036 000340

000040 000000
000042 000000
000044 000000
000046 003522
000052 000000

000174 000000
000176 000000

000200 000137 002002

001000 005377 040515 047111
(2) 001025 104 041515 030461

001200

001200 177570
001202 177570

```
*****
-----
; TRAPCATCAER 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
.TRPSR /       ; 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-C/<377>
        .ASCII /DMC11 CROM AND JUMP TESTS/<377>

.=1200
; INDIRECT POINTERS TO SWITCH REGISTER AND LIGHT DISPLAY
-----

DISPLAY: 177570
SWR: 177570
```


PROGRAM PARAMETERS, VARIABLES, AND TRAP CALLS.

```

144
145 ;INDIRECT POINTERS TO TELETYPE VECTORS AND REGISTERS
146 ;-----
147
148 001204 177560 TKCSR: 177560 ; TELETYPE KEYBOARD CONTROL REGISTER
149 001206 177562 TKDBR: 177562 ; TELETYPE KEYBOARD DATA BUFFER
150 001210 177564 TPCSR: 177564 ; TELEPRINTER CONTROL REGISTER
151 001212 177566 TPDBR: 177566 ; TELEPRINTER DATA BUFFER
152
153 ;PROGRAM CONTROL PARAMETERS
154 ;-----
155
156 001214 000000 RETURN: 0 ; SCOPE ADDRESS FOR LOOP ON TEST
157 001216 000000 NEXT: 0 ; ADDRESS OF NEXT TEST TO BE EXECUTED
158 001220 000000 LOCK: 0 ; ADDRESS FOR LOCK ON CURRENT DATA
159 001222 000003 ICOUNT: 3 ; NUMBER OF ITERATIONS THAT CURRENT TEST WILL BE
160 001224 000000 LPCNT: 0 ; NUMBER OF ITERATIONS COMPLETED
161 001226 000000 TSTNO: 0 ; NUMBER OF TEST IN PROGRESS
162 001230 000000 PASCNT: 0 ; NUMBER OF PASSES COMPLETED
163 001232 000000 ERRCNT: 0 ; TOTAL NUMBER OF ERRORS
164 001234 000000 LSTERR: 0 ; PC OF LAST ERROR CALL
165
166 ;PROGRAM VARIABLES
167 ;-----
168
169 001236 000000 STARTSW: 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 .EVEN
195 001320 001472 CREAM: DM.MAP-6 ; TABLE POINTER.
196 001322 001676 MILK: CNT.MAP-4 ; TABLE POINTER

```


PROGRAM PARAMETERS, VARIABLES, AND TRAP CALLS.

```

248 ;DMC11 CONTROL INDICATORS FOR CURRENT DMC11 UNDER TEST
249 ;-----
250
251 001366 000000 STAT1: 0
252 001370 000000 STAT2: 0
253 001372 000000 STAT3: 0
254
255 ;DMC11 VECTOR AND REGISTER INDIRECT POINTERS
256 ;-----
257
258 001374 000000 DMRVEC: 0 ; POINTER TO DMC11 RECEIVER INTERRUPT VECTOR
259 001376 000000 DMRLVL: 0 ; POINTER TO DMC11 RECEIVER INTERRUPT SERVICE PS
260 001400 000000 DMTVEC: 0 ; POINTER TO DMC11 TRANSMITTER INTERRUPT VECTOR
261 001402 000000 DMTLVL: 0 ; POINTER TO DMC11 TRANSMITTER INTERRUPT SERVICE PS
262 001404 000000 DMCSR: 0 ; POINTER TO DMC11 CONTROL STATUS REGISTER
263 001406 000000 DMCSRH: 0 ; POINTER TO DMC11 CONTROL STATUS REGISTER HIGH BYTE.
264 001410 000000 DMCTL: 0 ; POINTER TO DMC11 CONTROL OUT REGISTER
265 001412 000000 DMP04: 0 ; POINTER TO DMC11 PORT REGISTER(SEL 4)
266 001414 000000 DMP06: 0 ; POINTER TO DMC11 PORT REGISTER(SEL 6)
267
268 ;TEMP STORAGE
269 ;-----
270
271 001416 000000 TEMP: 0
272 001416 001460 .=. +40
273
274 ;DMC11 STATUS TABLE AND ADDRESS ASSIGNMENTS
275 ;-----
276
277 . = 1500
278 001500 DM.MAP:
279 001500 000001 DMC00: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 00
280 001502 000001 DMS100: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 00
281 001504 000001 DMS200: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 00
282 001506 000001 DMS300: .BLKW 1 ; 3RD STATUS WORD
283
284 001510 000001 DMC01: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 01
285 001512 000001 DMS101: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 01
286 001514 000001 DMS201: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 01
287 001516 000001 DMS301: .BLKW 1 ; 3RD STATUS WORD
288
289 001520 000001 DMC02: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 02
290 001522 000001 DMS102: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 02
291 001524 000001 DMS202: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 02
292 001526 000001 DMS302: .BLKW 1 ; 3RD STATUS WORD
293
294 001530 000001 DMC03: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 03
295 001532 000001 DMS103: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 03
296 001534 000001 DMS203: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 03
297 001536 000001 DMS303: .BLKW 1 ; 3RD STATUS WORD
298
299 001540 000001 DMC04: .BLKW 1 ; CONTROL STATUS REGISTER FOR DMC11 NUMBER 04
300 001542 000001 DMS104: .BLKW 1 ; VECTOR FOR DMC11 NUMBER 04
301 001544 000001 DMS204: .BLKW 1 ; DDCMP LINE# FOR DMC11 NUMBER 04
302 001546 000001 DMS304: .BLKW 1 ; 3RD STATUS WORD
303

```


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

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

413

 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	I	CSR
I	C	O	N	T	R	O	L		R	E	G	I	S	T	E	R	I
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
I	*	I	*	I	*	I	*	I	*	I	*	I	*	I	*	I	STAT1
I	I	I	I	I	I	I	I	I	I	V	E	C	T	O	R	*	I
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
I	*	B	M		A	D	D	*	I	*	L	I	N	E	*	*	STAT2
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	*	I	STAT3
I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	*	I

 DEFINITION OF FORMAT

- CSR: CONTAINS DMC11 CSR ADDRESS
- STAT1: BITS 00-08 IS DMC11 VECTOR ADDRESS
 BIT15=1 MICRO-PROCESSOR HAS 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 [PROGRAM DZDMI ONLY])
 KMC MUST HAVE MICRO-CODE WRITTEN FROM RUNNING
 DZDMG TEST 2 FIRST
 BIT1=1 DMC11-AL LOCAL HIGH SPEED MICRO-CODE
 BIT1=0 DMC11-AR REMOTE LOW SPEED MICRO-CODE


```

524 002312 000000          HALT          ;STOP THE SHOW
525 002314 000776          BR          ;DISQUALIFY CONTINUE SWITCH
526 002316 004737 010512 17$: JSR          PC,AUTO.SIZE ;GO DO THE AUTO SIZE
527 002322 105737 001324 16$: TSTB         INIFLG      ;FIRST TIME?
528 002326 001410          BEQ          21$      ;BR IF YES
529 002330 105737 001236          TSTB         STATSW     ;IF USING SAME PARAMETERS DONT TYPE MAP
530 002334 100431          BMI          1$
531 002336 032737 000006 001236 BIT          #BIT1!BIT2,STATSW ;IS TEST NO. OR LOCK SELECTED
532 002344 001403          BEQ          24$      ;IF NO THEN TYPE STATUS
533 002346 000424          BR          1$          ;IF YES DO NOT TYPE STATUS
534 002350 005137 001324 21$: COM          INIFLG      ;SET FLAG
535 002354 104402 006224 24$: TYPE         XHEAD       ;TYPE HEADER
536 002360 012704 001500          MOV          #DM.MAP,R4    ;SET POINTER
537 002364 010437 001246 5$:  MOV          R4,TEMP1    ;SET ADDRESS
538 002370 012437 001250          MOV          (R4)+,TEMP2  ;SET CSR
539 002374 001411          BEQ          1$          ;ALL DONE IF ZERO
540 002376 012437 001252          MOV          (R4)+,TEMP3  ;SET STAT1
541 002402 012437 001254          MOV          (R4)+,TEMP4  ;SET STAT2
542 002406 012437 001256          MOV          (R4)+,TEMP5  ;SET STAT3
543 002412 104410          CONVRT        ;TYPE OUT STATUS MAP
544 002414 007454          XSTATQ
545 002416 000762          BR          5$
546 002420 012700 001500 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. THIS TEST MAY REQUIRE 2 OR MORE ATTEMPTS TO GET THE
: *RIGHT ADDRESSES. AFTER YOU HAVE CHANGED THE ADDRESS TO WHAT IT TOLD
: *YOU THE FIRST TIME, IT MAY COME BACK AND TELL YOU A DIFFERENT ADDRESS
: *THE NEXT TIME YOU RUN IT. PLEASE HAVE PATIENCE, THE FINAL ADDRESS
: *WILL BE CORRECT (AS LONG AS ALL DEVICES IN FRONT OF THE DMC'S ARE
: *CORRECT).
*****

```

```

563 002424 013746 000004          MOV          @#4,-(SP)    ;SAVE LOC 4
564 002430 013746 000006          MOV          @#6,-(SP)    ;SAVE LOC 6
565 002434 005037 000006          CLR          @#6         ;CLEAR VEC+2
566 002440 005037 001252          CLR          TEMP3      ;CLEAR FLAG
567 002444 005005          CLR          R5         ;R5=0=DMC, R5=-1=KMC
568 002446 011037 001404  AUSTRT: MOV          (R0),DMCSR    ;GET NEXT DMC CSR
569 002452 001564          BEQ          AUDONE     ;BR IF DONE
570 002454 005705          TST          R5        ;DMC OR KMC?
571 002456 001005          BNE          1$        ;BR IF KMC
572 002460 032760 100000 000002 BIT          #BIT15,2(R0) ;CHECK FOR DMC CSR
573 002466 001061          BNE          SKIP      ;SKIP IF NOT DMC
574 002470 000404          BR          2$        ;ITS A DMC SO CONTINUE
575 002472 032760 100000 000002 1$: BIT          #BIT15,2(R0) ;CHECK FOR KMC CSR
576 002500 001454          BEQ          SKIP      ;SKIP IF NOT KMC
577 002502 012737 002674 000004 2$: MOV          #NODEV,@#4  ;SET UP FOR TIMEOUT
578 002510 005705          TST          R5        ;DMC OR KMC?
579 002512 001003          BNE          3$        ;BR IF KMC

```


E03

636	002760	104402		3\$:	TYPE		
637	002762	007330			KMCM		
638	002764	022626		4\$:	CMP	(SP)+, (SP)+	:ADJUST STACK
639	002766	000727			BR	OK	:BR TO GET OLT
640	002770	000001		ERRPC:	1		
641	002772	006	002		.BYTE	6,2	
642	002774	001276			SAVPC		
643	002776	000002		CONTAB:	2		
644	003000	006	004		.BYTE	6,4	
645	003002	003022			XLOC		
646	003004	006	002		.BYTE	6,2	
647	003006	001404			DMCSR		
648	003010	007		DEVTAB:	.BYTE	7	:DJ
649	003011	017			.BYTE	17	:DH
650	003012	007			.BYTE	7	:DQ
651	003013	007			.BYTE	7	:DU
652	003014	007			.BYTE	7	:DUP
653	003015	007			.BYTE	7	:LK
654	003016	007			.BYTE	7	:DMC
655	003017	007			.BYTE	7	:DZ
656	003020	007			.BYTE	7	:KMC
657		003022			.EVEN		
658	003022	000000		XLOC:	0		
659	003024	005705		AUDONE:	TST	R5	:DMC?
660	003026	001005			BNE	1\$:BR IF KMC AND ALL DONE
661	003030	012705	177777		MOV	#-1, R5	:SET R5 TO -1 (KMC)
662	003034	012700	001500		MOV	#DM.MAP, R0	:RESET R0 TO START OF TABLE
663	003040	000602			BR	AUSTR	:GO DO KMC'S
664	003042	012637	000006	1\$:	MOV	(SP)+, 2#6	:RESTORE LOC 6
665	003046	012637	000004		MOV	(SP)+, 2#4	:RESTORE LOC 4
666	003052	032737	000010	001236	BIT	#SW03, STRTSW	:SELECT SPECIFIC DEVICES??
667	003060	001422			BEQ	3\$:BR IF NO.
668	003062	104402	006144		TYPE	MNEW	:TYPE THE MESSAGE.
669	003066	005000			CLR	R0	:ZERO DATA LIGHTS
670	003070	000000			HALT		:WAIT FOR USER TO TELL WHAT DEVICES TO RUN
671	003072	027737	176104	001312	CMP	2SWR, SAVACT	:IS THE NUMBER VALID?
672	003100	101404			BLOS	2\$:BR IF NUMBER IS OK.
673	003102	104402	006005		TYPE	, MERR3	:TELL USER OF INVALID NUMBER.
674	003106	000000			HALT		:STOP EVERY THING.
675	003110	000776			BR	-2	:RESTART THE PROGRAM AGAIN.
676	003112	017737	176064	001306	2\$:	MOV 2SWR, DMACTV	:GET NEW DEVICE PATTERN
677	003120	013700	001306		MOV	DMACTV, R0	:SHOW THE USER WHAT HE SELECTED.
678	003124	000000			HALT		:CONTINUE DYNAMIC SWITCHES.
679	003126	012700	000300		3\$:	MOV #300, R0	:PREPARE TO CLEAR THE FLOATING
680	003132	012701	000302		MOV	#302, R1	:VECTOR AREA. 300-776
681	003136	010120			4\$:	MOV R1, (R0)+	:START PUTTING "PC+2 - HALT"
682	003140	005021			CLR	(R1)+	:IN VECTOR AREA.
683	003142	022021			CMP	(R0)+, (R1)+	:POP POINTERS
684	003144	022700	001000		CMP	#1000, R0	:ALL DONE??
685	003150	001372			BNE	4\$:BR IF NO.
686							
687							
688							
689							
690	003152	012706	001200		.BEGIN:	MOV #STACK, SP	:SET UP STACK
691	003156	013746	000006		MOV	2#6, -(SP)	:SAVE LOC 6

:TEST START AND RESTART
 :-----

F03

692	003162	013746	000004		MOV	2#4, -(SP)	;SAVE LOC 4	
693	003166	005000			CLR	RO	;START AT 0	
694	003170	012737	003234	000004	MOV	2#5, 2#4	;SET UP FOR TIME OUT	
695	003176	005037	000006		CLR	2#6	;TO AUTOSIZE MEMORY	
696	003202	005720			6\$:	TST	(RO)+	;CHECK ADDRESS IN RO
697	003204	022700	157776		CMP	157776, RO	;IS IT AT LEAST 28K	
698	003210	001374			BNE	6\$;BR IF NO	
699	003212	162700	007776		SUB	7776, RO	;SAVE 2K FOR MONITORS	
700	003216	010037	001304		7\$:	MOV	RO, MEMLIM	;STORE MEMORY LIMIT
701	003222	012637	000004		MOV	(SP)+, 2#4	;RESTORE LOC 4	
702	003226	012637	000006		MOV	(SP)+, 2#6	;RESTORE LOC 6	
703	003232	000413			BR	10\$;CONTINUE	
704	003234	022626			2\$:	CMP	(SP)+, (SP)+	;ADJUST STACK
705	003236	162700	000004		SUB	4, RO	;GET LAST GOOD ADDRESS	
706	003242	162700	007776		SUB	7776, RO	;SAVE 2K FOR MONITORS	
707	003246	022700	030000		CMP	30000, RO	;IS IT 8K?	
708	003252	001361			BNE	7\$;BR IF NO	
709	003254	012700	037400		MOV	37400, RO	;IF 8K DON'T SAVE 2K	
710	003260	000756			BR	7\$		
711	003262	012737	000340	177776	10\$:	MOV	340, PS	;LOCK OUT INTERRUPTS
712	003270	032737	000004	001236	BIT	BIT2, STRTSW	;CHECK FOR LOCK ON TEST	
713	003276	001411			BEQ	1\$;BR IF NO LOCK DESIRED.	
714	003300	104402	006043		TYPE	, MLOCK	;TYPE LOCK SELECTED.	
715	003304	012737	000240	003612	MOV	NOPT, TTST	;ADJUST SCOPE ROUTINE.	
716	003312	012737	000240	003614	MOV	NOPT, TTST+2	;SET UP TO LOCK	
717	003320	000406			BR	3\$;CONTINUE ALONG.	
718	003322	013737	003730	003612	1\$:	MOV	BRW, TTST	;PREPARE NORMAL SCOPE ROUTINE
719	003330	013737	003732	003614	MOV	BRX, TTST+2	;LOCK NOT SELECTED, SET UP FOR NORMAL SCOPE LOOP	
720	003336	012737	010060	001214	3\$:	MOV	CYCLE, RETURN	;START AT "CYCLE" FIND WHICH DEVICE TO TEST
721	003344	032737	000002	001236	4\$:	BIT	SW01, STRTSW	;IS TEST NO. SELECTED?
722	003352	001002			BNE	5\$;BR IF YES	
723	003354	104402	005755		TYPE	, MR	;TYPE R	
724	003360	000177	175630		5\$:	JMP	RETURN	;START TESTING

```

725                                     :END OF PASS
726                                     :TYPE NAME OF TEST
727                                     :UPDATE PASS COUNT
728                                     :CHECK FOR EXIT TO ACT-11
729                                     :RESTART TEST
730
731 003364 000005 .EOP: RESET ;MAKE THE WORLD CLEAN AGAIN.
732 003366 005037 001234 CLR LSTERR ;CLEAR LAST ERROR PC
733 003372 105037 001325 CLRB ERRFLG ;CLEAR EPROR FLAG
734 003376 005237 001230 INC PASCNT ;UPDATE PASS COUNT
735 003402 013777 001230 175570 MOV PASCNT, @DISPLAY ;DISPLAY PASS COUNT
736 003410 104402 005733 TYPE ,MEPASS ;TYPE END PASS
737 003414 104402 006072 TYPE ,MCSRX ;TYPE CSR
738 003420 104411 003546 CNVRT ,XCSR ;SHOW IT
739 003424 104402 006100 TYPE ,MVECX ;TYPE VECTOR
740 003430 104411 003554 CNVRT ,XVEC ;SHOW IT
741 003434 104402 006106 TYPE ,MPASSX ;TYPE PASSES
742 003440 104411 003562 CNVRT ,XPASS ;SHOW IT
743 003444 104402 006117 TYPE ,MERRX ;TYPE ERRORS
744 003450 104411 003570 CNVRT ,XERR ;SHOW IT
745 003454 013700 001322 MOV MILK, RO ;GET POINTER TO PASS COUNT
746 003460 013720 001230 MOV PASCNT, (RO)+ ;STORE PASS COUNT FOR THIS DMC!!
747 003464 013720 001232 MOV ERRCNT, (RO)+ ;STORE ERROR COUNT FOR THIS DMC!!
748 003470 005337 001314 DEC SAVNUM ;ARE ALL DEVICES TESTED?
749 003474 001017 BNE RESTR ;BR IF NO.
750 003476 112737 000377 001327 MOVVB #377, QV.FLG ;SET THE QUICK VERIFY FLAG.
751 003504 013737 001310 001314 MOV DMNUM, SAVNUM ;RESTORE THE COUNT
752 003512 013701 000042 MOV @#42, R1 ;CHECK FOR ACT-11 OR DDP
753 003516 001406 BEQ RESTR ;IF NOT, CONTINUE TESTING
754 003520 000005 RESET ;STOP THE SHOW--CLEAR THE WORLD
755 003522
756 003522 004711 $ENDAC: JSR PC, (R1)
757 003524 000240 NOP
758 003526 000240 NOP
759 003530 000240 NOP
760 003532 000240 NOP
761 003534 012737 010060 001214 RESTR: MOV #CYCLE, RETURN
762 003542 000137 010060 JMP CYCLE
763 003546 000001 XCSR: 1
764 003550 006 002 .BYTE 6,2
765 003552 001404 DMCSR
766 003554 000001 XVEC: 1
767 003556 004 002 .BYTE 4,2
768 003560 001374 DMRVEC
769 003562 000001 XPASS: 1
770 003564 006 002 .BYTE 6,2
771 003566 001230 PASCNT
772 003570 000001 XERR: 1
773 003572 006 002 .BYTE 6,2
774 003574 001232 ERRCNT
775
776                                     :SCOPE LOOP AND INTERATION HANDLER
777 -----
778
779 003576 004737 007606 .SCOPE: JSR PC, CKSWR ;CHECK FOR SOFT SWR
780 003602 010016 MOV RO, (SP) ;SAVE RO ON THE STACK

```


GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

781 003604 032777 040000 175370
782 003612 001407
783 003614 000437
784 003616 005737 003734
785 003622 001434
786 003624 005037 003734
787 003630 000415
788 003632 032777 004000 175342
789 003640 001011
790 003642 105737 001327
791 003646 001406
792 003650 005237 001224
793 003654 023737 001224 001222
794 003662 101414
795 003664 105037 001325
796 003670 005037 001224
797 003674 005037 001220
798 003700 012737 000020 001222
799 003706 013737 001216 001214
800 003714 011600
801 003716 022626
802 003720 013701 001404
803 003724 000177 175264
804 003730 001407
805 003732 000437
806 003734 000000
807
808
809
810
811 003736 004737 007606
812 003742 032777 001000 175232
813 003750 001405
814 003752 005737 001220
815 003756 001402
816 003760 013716 001220
817 003764 000002
818
819
820
821
822 003766 010546
823 003770 017605 000002
824 003774 062766 000002 000002
825 004002 005737 010016
826 004006 001004
827 004010 032777 010000 175164
828 004016 001012
829 004020 105715
830 004022 100002
831 004024 104402 005672
832 004030 105777 175154
833 004034 100375
834 004036 112577 175150
835 004042 001357
836 004044 012605

```

```

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

```

```

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

```

```

;TELETYPE OUTPUT ROUTINE
;-----
.TYPE: MOV R5, -(SP) ;SAVE R5 ON THE STACK.
        MOV @2(SP),R5 ;GET ADDRESS OF MESSAGE.
        ADD #2,2(SP) ;POP OVER ADDRESS.
4$: TST SWFLG ;SOFT SWR MESSAGE?
      BNE 1$ ;IF YES TYPE IT OUT REGARDLESS OF SW12
      BIT #SW12,JSWR ;INHIBIT ALL PRINT OUT??
      BNE 3$ ;BR IF NO PRINT OUT WANTED (SW12=1)
1$: TSTB (R5) ;IS NUMBER MINUS? (MSB=1(BIT7))
      BPL 2$ ;BR IF NUMBER IS PLUS
      TYPE MCRLF ;TYPE A CR/LF!
2$: TSTB @TPCSR ;TTY READY?
      BPL 2$ ;BR IF NO.
      MOVB (R5)+,@TPDDBR ;PRINT CURRENT CHAR.
      BNE 4$ ;IF NOT ZERO KEEP PRINTING!
3$: MOV (SP)+,R5 ;END OF OUTPUT. RESTORE R5

```

```

837 004046 000002          RTI          ;GO HOME
838
839
840 004050 010346          .INSTR: MOV    R3,-(SP)          ;SAVE R3 ON STACK
841 004052 010446          MOV    R4,-(SP)          ;SAVE R4 ON STACK
842 004054 017637 000004 004072  MOV    @4(SP),.MSG
843 004062 062766 000002 000004  ADD    #2,4(SP)
844 004070 104402          .INST1: TYPE
845 004072 000000          .MSG:  0
846 004074 012704 007502  MOV    #INBUF,R4
847 004100 012703 000007  MOV    #7,R3
848 004104 105777 175074  1$:   TSTB  @TKCSR
849 004110 100375          BPL    1$
850 004112 117714 175070  MOVB  @TKDBR,(R4)
851 004116 142714 000200  BICB  #200,(R4)
852 004122 122427 000015  CMPB  (R4)+,#15
853 004126 001417          BEQ   INSTR2
854 004130 105777 175054  2$:   TSTB  @TPCSR
855 004134 100375          BPL    2$
856 004136 017777 175044 175046  MOV    @TKDBR,@TPDBR
857 004144 005303          DEC   R3
858 004146 001356          BNE   1$
859 004150 012604          MOV   (SP)+,R4
860 004152 012603          MOV   (SP)+,R3
861 004154 104402 005666  .INSTE: TYPE  MQM
862 004160 010346          MOV   R3,-(SP)
863 004162 010446          MOV   R4,-(SP)
864 004164 000741          BR    .INST1
865 004166 012604  INSTR2: MOV   (SP)+,R4          ;RESTORE R4
866 004170 012603          MOV   (SP)+,R3          ;RESTORE R3
867 004172 000002          RTI
868
869          ;CONVERT ASCII STRING TO OCTAL
870
871
872 004174 010546          .PARAM: MOV   R5,-(SP)
873 004176 010446          MOV   R4,-(SP)
874 004200 016605 000004  MOV   4(SP),R5
875 004204 012537 004364  MOV   (R5)+,LOLIM
876 004210 012537 004366  MOV   (R5)+,HILIM
877 004214 012537 004370  MOV   (R5)+,DEVADR
878 004220 112537 004372  MOVB  (R5)+,LOBITS
879 004224 112537 004373  MOVB  (R5)+,ADRCNT
880 004230 010566 000004  MOV   R5,4(SP)
881 004234 005005  PARAM1: CLR   R5
882 004236 012704 007502  MOV   #INBUF,R4
883 004242 122714 000015  CMPB  #15,(R4)
884 004246 001420          BEQ   PARERR
885 004250 121427 000060  1$:   CMPB  (R4),#60
886 004254 002415          BLT   PARERR
887 004256 121427 000067  CMPB  (R4),#67
888 004262 003012          BGT   PARERR
889 004264 142714 000060  BICB  #60,(R4)
890 004270 152405          BISB  (R4)+,R5
891 004272 122714 000015  CMPB  #15,(R4)
892 004276 001406          BEQ   LIMITS
    
```

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

893	004300	006305			ASL	R5	
894	004302	006305			ASL	R5	
895	004304	006305			ASL	R5	
896	004306	000760			BR	IS	
897	004310	104404			PARERR: INSTER		
898	004312	000750			BR	PARAM1	
899							
900							: TEST TO SEE IF NUMBER IS WITHIN LIMITS
901							-----
902							
903	004314	020537	004366		LIMITS: CMP	R5, HILIM	
904	004320	101373			BHI	PARERR	
905	004322	020537	004364		CMP	R5, LOLIM	
906	004326	103770			BLO	PARERR	
907	004330	133705	004372		BITB	LOBITS, R5	
908	004334	001365			BNE	PARERR	
909							
910							: STORE NUMBER AT SPECIFIED ADDRESS
911							
912	004336	013704	004370		IS: MOV	DEVADR, R4	
913	004342	010524			MOV	R5, (R4)+	
914	004344	062705	000002		ADD	#2, R5	
915	004350	105337	004373		DECB	ADRCNT	
916	004354	001372			BNE	IS	
917	004356	012604			MOV	(SP)+, R4	
918	004360	012605			MOV	(SP)+, R5	
919	004362	000002			RTI		
920	004364	000000			LOLIM: 0		
921	004366	000000			HILIM: 0		
922	004370	000000			DEVADR: 0		
923	004372	000000			LOBITS: 0		
924		004373			ADRCNT=LOBITS+1		
925							
926							: SAVE PC OF TEST THAT FAILED AND R0-R5
927							-----
928							
929	004374	016637	000004	001276	.SAV05: MOV	4(SP), SAVPC	: SAVE R7 (PC)
930							
931							: SAVE R0-R5
932							
933	004402	010537	001272		SV05: MOV	R5, SAVR5	: SAVE R5
934	004406	010437	001270		MOV	R4, SAVR4	: SAVE R4
935	004412	010337	001266		MOV	R3, SAVR3	: SAVE R3
936	004416	010237	001264		MOV	R2, SAVR2	: SAVE R2
937	004422	010137	001262		MOV	R1, SAVR1	: SAVE R1
938	004426	010037	001260		MOV	R0, SAVR0	: SAVE R0
939	004432	000002			RTI		: LEAVE.
940							
941							: RESTORE R0-R5
942							
943	004434	013700	001260		.RES05: MOV	SAVR0, R0	: RESTORE R0
944	004440	013701	001262		MOV	SAVR1, R1	: RESTORE R1
945	004444	013702	001264		MOV	SAVR2, R2	: RESTORE R2
946	004450	013703	001266		MOV	SAVR3, R3	: RESTORE R3
947	004454	013704	001270		MOV	SAVR4, R4	: RESTORE R4
948	004460	013705	001272		MOV	SAVR5, R5	: RESTORE R5

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

949 004464 000002 RTI ;LEAVE
950
951 ;CONVERT OCTAL NUMBER TO ASCII AND OUTPUT TO TELEPRINTER
952 -----
953
954 004466 104402 005672 .CONVR: TYPE MCRLF
955 004472 010046 .CNVRT: MOV R0,-(SP)
956 004474 010146 MOV R1,-(SP)
957 004476 010346 MOV R3,-(SP)
958 004500 010446 MOV R4,-(SP)
959 004502 010546 MOV R5,-(SP)
960 004504 017601 000012 MOV @12(SP),R1
961 004510 062766 000002 000012 ADD #2,12(SP)
962 004516 012137 004712 MOV (R1)+,WRDCNT
963 004522 112137 004712 1$: MOVB (R1)+,CHRCNT
964 004526 112137 004713 MOVB (R1)+,SPACNT
965 004532 013137 004714 MOV @2(R1)+,BINWRD
966 004536 122737 000003 004712 CMPB #3,CHRCNT
967 004544 001003 BNE 2$
968 004546 042737 177400 004714 BIC #177400,BINWRD
969 004554 013704 004714 2$: MOV BINWRD,R4
970 004560 113705 004712 MOVB CHRCNT,R5
971 004564 012700 001416 MOV #TEMP,R0
972 004570 010403 3$: MOV R4,R3
973 004572 042703 177770 BIC #177770,R3
974 004576 062703 000060 ADD #060,R3
975 004602 110320 MOVB R3,(R0)+
976 004604 000241 CLC
977 004606 006004 ROR R4
978 004610 000241 CLC
979 004612 006004 ROR R4
980 004614 000241 CLC
981 004616 006004 ROR R4
982 004620 005305 DEC R5
983 004622 001362 BNE 3$
984 004624 012703 007544 MOV #MDATA,R3
985 004630 114023 4$: MOVB -(R0),(R3)+
986 004632 105337 004712 DECB CHRCNT
987 004636 001374 BNE 4$
988 004640 105737 004713 TSTB SPACNT
989 004644 001405 BEQ 6$
990 004646 112723 000040 5$: MOVB #040,(R3)+
991 004652 105337 004713 DECB SPACNT
992 004656 001373 BNE 5$
993 004660 105013 6$: CLRB (R3)
994 004662 104402 007544 TYPE ,MDATA
995 004666 005337 004710 DEC WRDCNT
996 004672 001313 BNE 1$
997 004674 012605 MOV (SP)+,R5
998 004676 012604 MOV (SP)+,R4
999 004700 012603 MOV (SP)+,R3
1000 004702 012601 MOV (SP)+,R1
1001 004704 012600 MOV (SP)+,R0
1002 004706 000002 RTI
1003 004710 000000 WRDCNT: 0
1004 004712 000000 CHRCNT: 0

```

```

1005          004713          SPACNT=CHRCNT+1
1006 004714 000000          BINWRD: 0
1007
1008
1009
1010          ; TRAP DISPATCH SERVICE
1011          ; ARGUMENT OF TRAP IS EXTRACTED
1012          ; AND USED AS OFFSET TO OBTAIN POINTER
1013          ; TO SELECTED SUBROUTINE
1014 004716 011646          .TRPSR: MOV      (SP), -(SP)          ; GET PC OF RETURN
1015 004720 162716 000002          SUB      #2, (SP)          ; =PC OF TRAP
1016 004724 017616 000000          MOV      @2(SP), (SP)          ; GET TRP
1017 004730 006316          TRPOK: ASL      (SP)          ; MULTIPLY TRAP ARG BY 2
1018 004732 042716 177001          BIC      #177001, (SP)          ; CLEAR UNWANTED BITS
1019 004736 062716 001330          ADD      #.TRPTAB, (SP)          ; POINTER TO SUBROUTINE ADDRESS
1020 004742 017616 000000          MOV      @2(SP), (SP)          ; SUBROUTINE ADDRESS
1021 004746 000136          JMP      @2(SP)+          ; GO TO SUBROUTINE
1022
1023          ; ERROR HANDLER
1024          ; -----
1025
1026 004750 004737 007606          .HLT: JSR      PC, CKSWR          ; CHECK FOR SOFT SWR
1027 004754 032777 010000 174220          BIT      #SW12, @SWR          ; BELL ON ERROR?
1028 004762 001406          BEQ      XBX          ; BR IF NO BELL
1029 004764 105777 174220          TSTB     @TPCSR          ; TTY READY.
1030 004770 100003          BPL      XBX          ; DON'T WAIT IF TTY NOT READY.
1031 004772 112777 000207 174212          MOVB     #207, @TPDBR          ; PUSH A BELL AT THE TTY.
1032 005000 032777 020000 174174          XBX: BIT      #SW13, @SWR          ; DELETE ERROR PRINT OUT?
1033 005006 001105          BNE      HALTS          ; BR IF NO PRINT OUT WANTED.
1034 005010 021637 001234          CMP      (SP), LSTERR          ; WAS THIS ERROR FOUND LAST TIME?
1035 005014 001404          BEQ      IS          ; BR IF YES
1036 005016 011637 001234          MOV      (SP), LSTERR          ; RECORD BEING HERE
1037 005022 105037 001325          CLRB     ERRFLG          ; PREPARE HEADER
1038 005026 104406          IS: SAVOS          ; SAVE ALL PROC REGISTERS
1039 005030 011605          MOV      (SP), R5          ; GET THE PC OF ERROR
1040 005032 162705 000002          SUB      #2, R5          ; GET ADDRESS OF TRAP CALL
1041 005036 011504          MOV      (R5), R4          ; GET HLT INSTRUCTION
1042 005040 006304          ASL      R4          ; MULT BY TWO
1043 005042 061504          ADD      (R5), R4          ; DOUBLE IT
1044 005044 006304          ASL      R4          ; MULT AGAIN
1045 005046 042704 177001          BIC      #177001, R4          ; CLEAR JUNK
1046 005052 062704 027602          ADD      #.ERRTAB, R4          ; GET POINTER
1047 005056 012437 005172          MOV      (R4)+, ERRMSG          ; GET ERROR MESSAGE
1048 005062 012437 005204          MOV      (R4)+, DATAHD          ; GET DATA HEADER
1049 005066 011437 005216          MOV      (R4), DATABP          ; GET DATA TABLE
1050 005072 105737 001325          TSTB     ERRFLG          ; TYPE HEADREER
1051 005076 001403          BEQ      TYPMSG          ; BR IF YES
1052 005100 005737 005216          TST      DATABP          ; DOES DATA TABLE EXIST?
1053 005104 001040          BNE      TYPDAT          ; BR IF YES.
1054 005106 104402 005672          TYPMSG: TYPE     , MCRLF          ;
1055 005112 104402 005672          TYPE     , MCRLF          ;
1056 005116 005737 001220          TST      LOCK          ;
1057 005122 001402          BEQ      IS          ;
1058 005124 104402 006142          TYPE     , MASTEK          ;
1059 005130 104402 006130          IS: TYPE     , MTSTN          ;
1060 005134 104411 005330          CNVRT    , XTSTN          ; SHOW IT
    
```


GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1061	005140	104402	006217		TYPE	,MERRPC	:TYPE PC.
1062	005144	104411	005322		CNVRT	,ERTABO	:SHOW IT
1063	005150	104402	005672		TYPE	MCRLF	:GIVE A CR/LF
1064	005154	112737	177777	001325	MOVB	4-1,ERRFLG	:NO MORE HEADER UNLESS NO DATA TABLE.
1065	005162	005737	005172		TST	ERRMSG	:IS THERE AN ERROR MESSAGE?
1066	005166	001402			BEQ	WRKO.FM	:BR IF NO.
1067	005170	104402			TYPE		:TYPE
1068	005172	000000			ERRMSG:	0	:ERROR MESSAGE
1069	005174				WRKO.FM:		
1070	005174	005737	005204		TST	DATAHD	:DATA HEADER?
1071	005200	001402			BEQ	TYPDAT	:BR IF NO
1072	005202	104402			TYPE		:TYPE
1073	005204	000000			DATAHD:	0	:DATA HEADER
1074	005206	005737	005216		TYPDAT:	TST	:DATA TABLE?
1075	005212	001402			BEQ	RESREG	:BR IF NO.
1076	005214	104410			CONVRT		:SHOW
1077	005216	000000			DATAHD:	0	:DATA TABLE
1078	005220	104407			RESREG:	RESOS	:RESTORE PROC REGISTERS
1079	005222	022737	003522	000042	HALTS:	CMP	:IF ACT-11 AUTOMATIC MODE, HALT!!
1080	005230	001403			BEQ	1\$	
1081	005232	005777	173744		TST	2\$WR	:HALT ON ERROR?
1082	005236	100005			BPL	EXITER	:BR IF NO HALT ON ERROR
1083	005240	010046			1\$:	PUSHRO	:SAVE RO
1084	005242	016600	000002		MOV	2(SP),RO	:SHOW ERROR PC IN DATA LIGHTS
1085	005246	000000			HALT		:HALT
1086	005250	012600			POPPO		:GET RO
1087	005252	005237	001232		EXITER:	INC	:UPDATE ERROR COUNT
1088	005256	032777	000400	173716	BIT	3\$W08,2\$WR	:GOTO TOP OF TEST?
1089	005264	001007			BNE	1\$:BR IF YES
1090	005266	032777	002000	173706	BIT	3\$W10,2\$WR	:GOTO NEXT TEST?
1091	005274	001411			BEQ	2\$:BR IF NO
1092	005276	013737	001216	001214	MOV	NEXT RETURN	:SET FOR NEXT TEST
1093	005304	012706	001200		1\$:	MOV	:RESET SP
1094	005310	013701	001404		MOV	DMCSR,R1	:SET UP R1
1095	005314	000177	173674		JMP	2\$RETURN	:GOTO SPECIFIED TEST
1096	005320	000002			2\$:	RTI	:RETURN
1097	005322	000001			ERTABO:	1	
1098	005324	006	002		.BYTE	6,2	
1099	005326	001276			SAVPC		
1100	005330	000001			XTSTN:	1	
1101	005332	003	002		.BYTE	3,2	
1102	005334	001226			TSTNO		
1103							:ENTER HERE ON POWER FAILURE
1104							:-----
1105							
1106							
1107	005336				.PFAIL:		
1108	005336	012737	005350	000024	MOV	3\$RESTART,24	:SET UP FOR POWER UP TRAP
1109	005344	000000			HALT		:HALT ON POWER DOWN NORMAL
1110	005346	000777			BR	.	
1111							
1112							:PROCESSOR WILL TRAP HERE WHEN POWER IS RESTORED
1113							
1114	005350				RESTAR:		
1115	005350	012737	005336	000024	MOV	3\$.PFAIL,24	:SET UP FOR POWER FAILURE
1116	005356	012706	001200		MOV	3\$STACK,SP	:RESET THE STACK POINTER

DZDMG MACY11 30(1046) 11-JUL-77 12:11 PAGE 23
 DZDMG.P11 22-APR-77 09:29

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

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

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

```

1173 005626
1174 005626 104414
1175 005630 021364
1176 005632 032777 000002 173552
1177 005640 001772
1178 005642
1179 005642 104414
1180 005644 021364
1181 005646 032777 000002 173536
1182 005654 001372
1183 005656 005337 001416
1184 005662 001361
1185 005664 000002
1186
1187 005666 020040 000077
(2) 005672 005015 000
(2) 005675 377 053520 020122
(2) 005733 377 047105 020104
(2) 005755 377 000122
(2) 005760 047377 020117 042504
(2) 006005 377 047111 052523
(2) 006031 377 042524 052123
(2) 006043 377 047514 045503
(2) 006072 051503 035122 000040
(2) 006100 042526 035103 000040
(2) 006106 040520 051523 051505
(2) 006117 105 051122 051117
(2) 006130 042524 052123 047040
(2) 006142 000052
(2) 006144 051777 052105 051440
(2) 006217 120 035103 000040
(2) 006224 020212 020040 020040
(2) 006263 377 020040 020040
(2) 006322 020212 050040 020103
(2) 006374 026777 026455 026455
(2) 006450 044377 053517 046440
(2) 006510 041777 051123 040440
(2) 006526 053377 041505 047524
(2) 006547 377 051102 050040
(2) 006606 044777 020106 046504
(2) 006704 053777 044510 044103
(2) 007016 051777 044527 041524
(2) 007054 051777 044527 041524
(2) 007114 044777 020123 044124
(2) 007154 047377 020117 042504
(2) 007205 377 051412 051127
(2) 007215 116 053505 020077
(2) 007223 377 042377 041515
(2) 007277 377 054105 042520
(2) 007320 024040 046504 024503
(2) 007330 024040 046513 024503
(2) 007340 042377 041515 030461
(2)
(2) 007454 000005
1188 007456 006 003
1189 007460 001246

```

```

15: ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021364 ;PORT4+IBUS* REG11
BIT #2,ADMP04 ;IS PGM CLOCK BIT CLEAR?
BEQ 15 ;BR IF YES

25: ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
021364 ;PORT4+IBUS* REG11
BIT #2,ADMP04 ;IS PGM CLOCK BIT SET?
BNE 25 ;BR IF YES
DEC TEMP ;DEC COUNT
BNE 15 ;BR IF NOT DONE
RTI ;RETURN

MQM: .ASCIZ / ?/
MCRLF: .ASCIZ <15><12>
MPFAIL: .ASCIZ <377>/PWR FAILED. RESTART AT TEST /
MEPASS: .ASCIZ <377>/END PASS DZDMG /
MR: .ASCIZ <377>/R/
MERR2: .ASCIZ <377>/NO DEVICES PRESENT./
MERR3: .ASCIZ <377>/INSUFFICIENT DATA!/
MTSTPC: .ASCIZ <377>/TEST PC-/
MLOCK: .ASCIZ <377>/LOCK ON SELECTED TEST/
MCSRX: .ASCIZ /CSR: /
MVECX: .ASCIZ /VEC: /
MPASSX: .ASCIZ /PASSES: /
MERRX: .ASCIZ /ERRORS: /
MTSTN: .ASCIZ /TEST NO: /
MASTEK: .ASCIZ /*/
MNEW: .ASCIZ <377>/SET SWITCH REG TO DMC11'S DESIRED ACTIVE./
MERRPC: .ASCIZ /PC: /
XHEAD: .ASCII <212>/ MAP OF DMC11 STATUS/
.ASCII <377>/
.ASCII <212>/ PC CSR STAT1 STAT2 STAT3/
.ASCIZ <377>/-----/
.ASCIZ <377>/HOW MANY DMC11'S TO BE TESTED?/
CSR: .ASCIZ <377>/CSR ADDRESS?/
VEC: .ASCIZ <377>/VECTOR ADDRESS?/
PRIO: .ASCIZ <377>/BR PRIORITY LEVEL? (4,5,6,7)?/
CRAM: .ASCIZ <377>/IF DMC HAS CRAM (M8204) TYPE "Y", IF CROM (M8200) TYPE "N"
MODU: .ASCIZ <377>/WHICH LINE UNIT? IF NONE TYPE "N", IF M8201 TYPE "1", IF M
LINE: .ASCIZ <377>/SWITCH PAC#1 (DDCMP LINE #)?/
BM: .ASCIZ <377>/SWITCH PAC#2 (BM873 BOOT ADD)?/
CONN: .ASCIZ <377>/IS THE LOOP BACK CONNECTOR ON?
NOACT: .ASCIZ <377>/NO DEVICES ARE SELECTED/
SWMES: .ASCIZ <377><12>/SWR= /
SWMES1: .ASCIZ /NEW? /
CONERR: .ASCIZ <377><377>/DMC11 FOUND AT NON-STANDARD ADDRESS PC:
CNERR: .ASCIZ <377>/EXPECTED FOUND/
DMCM: .ASCIZ / (DMC) /
KMCM: .ASCIZ / (KMC) /
SPEED: .ASCIZ <377>/DMC11-AR (REMOTE, LOW SPEED) OR DMC11-AL (LOCAL, HIGH SPEED) ?
.EVEN
XSTATQ: 5
.BYTE 6,3
TEMP1

```

1190	007462	006	003		.BYTE	6,3		
1191	007464	001250			TEMP2			
1192	007466	006	003		.BYTE	6,3		
1193	007470	001252			TEMP3			
1194	007472	006	003		.BYTE	6,3		
1195	007474	001254			TEMP4			
1196	007476	006	002		.BYTE	6,2		
1197	007500	001256			TEMP5			
1198				.EVEN				
1199								
1200					:BUFFERS FOR INPUT-OUTPUT			
1201								
1202	007502	000000		INBUF:	0			
1203		007544		.+.40				
1204	007544	000000		MDATA:	0			
1205		007606		.+.40				
1206								
1207								
1208					:ROUTINE USED TO CHANGE SOFTWARE SWITCH			
1209					:REGISTER USING THE CONSOLE TERMINAL			
1210					-----			
1211								
1212	007606	022737	000176	001202	CKSWR:	CMP	#SWREG,SWR	: IS THE SOFT SWR BEING USED?
1213	007614	001077				BNE	CKSWRS	: BR IF NO
1214	007616	105777	171362			TSTB	@TKCSR	: IS DONE SET?
1215	007622	100003				BPL	25	: GO ON IF NOT SET
1216	007624	012737	177777	003734		MOV	#-1,DONE	: IF DONE SET, SET FLAG
1217	007632	022777	000007	171346	25:	CMP	#7,@TKDBR	: WAS CTRL G TYPED? (7 BIT ASCII
1218	007640	001404				BEQ	15	: BR IF YES
1219	007642	022777	000207	171336		CMP	#207,@TKDBR	: WAS CTRL G TYPED? (8 BIT ASCII
1220	007650	001061				BNE	CKSWRS	: BR IF NO
1221	007652	010246			15:	MOV	R2,-(SP)	: STORE R2
1222	007654	010346				MOV	R3,-(SP)	: STORE R3
1223	007656	010446				MOV	R4,-(SP)	: STORE R4
1224	007660	012737	177777	010016		MOV	#-1,SWFLG	: SET SOFT TYPE OUT FLAG
1225	007666	005002			CKSWR1:	CLR	R2	: CLEAR NEW SWR CONTENTS
1226	007670	012704	177777			MOV	#-1,R4	: SET FLAG TO ALL ONES
1227	007674	104402	007205			TYPE	.SWMES	: TYPE "SWR="
1228	007700	104411			CKSWR2:	CNVRT		: TYPE OUT PRESENT CONTENTS
1229	007702	010052				SOFTSW		: OF SOFT SWITCH REGISTER
1230	007704	104402	007215		CKSWR3:	TYPE	.SWMES1	: TYPE "NEW?"
1231	007710	004737	010020		CKSWR4:	JSR	@C INCHAR	: GET RESPONSE
1232	007714	022703	000015			CMP	#15,R3	: WAS IT A CR?
1233	007720	001424				BEQ	55	: BR IF YES
1234	007722	022703	000012			CMP	#12,R3	: WAS IT A LF?
1235	007726	001416				BEQ	45	: BR IF YES
1236	007730	022703	000025			CMP	#25,R3	: WAS IT CTRL J?
1237	007734	001754				BEQ	CKSWR1	: BR IF YES (START OVER,
1238	007736	022703	000007			CMP	#7,R3	: IF CNTL G GET NEXT CHAR
1239	007742	001762				BEQ	CKSWR4	
1240	007744	005004				CLR	R4	: IT MUST BE A DIGIT SO CLR FLAG
1241	007746	042703	177770			BIC	#177770,R3	: ONLY 0-7 ARE LEGAL SO MASK OFF BITS
1242	007752	006302				ASL	R2	: SHIFT R2 3 TIMES
1243	007754	006302				ASL	R2	
1244	007756	006302				ASL	R2	
1245	007760	050302				BIS	R3,R2	: ADD LAST DIGIT

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1246	007762	000752			BR	CKSWR4	:GET NEXT CHARACTER
1247	007764	012766	002002	000006	4\$: MOV	#.START,6(SP)	:LF WAS TYPED SO GO TO START
1248	007772	005704			5\$: TST	R4	:IS FLAG CLEAR?
1249	007774	001002			BNE	6\$:IF NOT DON'T CHANGE SOFT SWR
1250	007776	010277	171200		MOV	R2,@SWR	:IF YES THEN WRITE NEW CONTENTS TO SCFT SWR
1251	010002	005037	010016		6\$: CLR	SWFLG	:CLEAR TYPEOUT FLAG
1252	010006	012604			MOV	(SP)+,R4	:RESTORE R4
1253	010010	012603			MOV	(SP)+,R3	:RESTORE R3
1254	010012	012602			MOV	(SP)+,R2	:RESTORE R2
1255	010014	000207			CKSWR5: RTS	PC	:RETURN
1256							
1257	010016	000000			SWFLG:	0	
1258							
1259	010020	105777	171160		INCHAR: TSTB	@TKCSR	
1260	010024	100375			BPL	-4	
1261	010026	017703	171154		MOV	@TKDBR,R3	
1262	010032	105777	171152		TSTB	@TPCSR	
1263	010036	100375			BPL	-4	
1264	010040	010377	171146		MOV	R3,@TPOBR	
1265	010044	042703	000200		BIC	#BIT7,R3	
1266	010050	000207			RTS	PC	
1267							
1268	010052	000001			SOFTSW: 1		
1269	010054	006	002		.BYTE	6.2	
1270	010056	000176			SWREG		


```

1271
1272
1273
1274
1275
1276
1277
1278
1279
1280 010060 005737 001306          CYCLE:  TST      DMACTV      ;ARE ANY DMC11'S TO BE TESTED?
1281 010064 001004                BNE      1$           ;BR IF OK.
1282 010066 104402 007154          TYPE     ,NOACT      ;NO DMC11'S SELECTED!!
1283 010072 000000                HALT                    ;STOP THE SHOW.
1284 010074 000776                BR                    ;DISQUALIFY CONT. SW.
1285 010076 000241                1$:  CLC            ;CLEAR PROC. CARRY BIT.
1286 010100 006137 001316          ROL                    ;UPDATE POINTER
1287 010104 005537 001316          ADC      RUN          ;CATCH CARRY FROM RUN
1288 010110 062737 000004 001322  ADD      #4,MILK      ;UPDATE POINTER
1289 010116 062737 000010 001320  ADD      #10,CREAM    ;UPDATE ADDRESS POINTER.
1290 010124 022737 001700 001320  CMP      #DM.MAP+200,CREAM
1291 010132 001006                BNE      2$           ;KEEP GOING; NOT ALL TESTED FOR.
1292 010134 012737 001500 001320  MOV      #DM.MAP,CREAM ;RESET ADDRESS POINTER.
1293 010142 012737 001702 001322  MOV      #CNT.MAP,MILK ;RESET PASS COUNT POINTER
1294 010150 033737 001316 001306  2$:  BIT      RUN,DMACTV ;IS THIS ONE ACTIVE?
1295 010156 001747                BEQ      1$           ;BR IF NO
1296 010160 013700 001320          MOV      CREAM,R0     ;GET ADDRESS POINTER
1297 010164 013702 001322          MOV      MILK,R2      ;GET PASS COUNT POINTER
1298 010170 012037 001404          MOV      (R0)+,DMCSR  ;LOAD SYSTEM CTRL. REG
1299 010174 011037 001374          MOV      (R0),DMRVEC  ;LOAD VECTOR
1300 010200 042737 177000 001374  BIC      #177000,DMRVEC ;CLEAR UNWANTED BITS
1301 010206 012037 001366          MOV      (R0)+,STAT1  ;LOAD STAT1
1302 010212 012037 001370          MOV      (R0)+,STAT2  ;LOAD STAT2
1303 010216 012037 001372          MOV      (R0)+,STAT3  ;LOAD STAT3
1304 010222 012237 001230          MOV      (R2)+,PASCNT ;LOAD PASS COUNT
1305 010226 012237 001232          MOV      (R2)+,ERRCNT ;LOAD ERROR COUNT
1306 010232 012700 000002          MOV      #2,R0        ;SAVE CORE THIS WAY!
1307 010236 013737 001404 001406  MOV      DMCSR,DMCSRH
1308 010244 005237 001406          INC      DMCSRH
1309 010250 013737 001406 001410  MOV      DMCSRH,DMCTL
1310 010256 005237 001410          INC      DMCTL
1311 010262 013737 001410 001412  MOV      DMCTL,DMP04
1312 010270 060037 001412          ADD      R0,DMP04
1313 010274 013737 001412 001414  MOV      DMP04,DMP06
1314 010302 060037 001414          ADD      R0,DMP06
1315
1316 010306 013737 001374 001376  MOV      DMRVEC,DMRLVL ;PTY LVL
1317 010314 060037 001376          ADD      R0,DMRLVL
1318 010320 013737 001376 001400  MOV      DMRLVL,DMTVEC ;TX VEC
1319 010326 060037 001400          ADD      R0,DMTVEC
1320 010332 013737 001400 001402  MOV      DMTVEC,DMTLVL ;TX LVL
1321 010340 060037 001402          ADD      R0,DMTLVL
1322
1323 010344 032737 000002 001236  BIT      #SW01,STARTSW ;IS TEST NO. SELECTED
1324 010352 001450                BEQ      7$           ;BR IF NO
1325 010354
1326 010354 005737 000042          4$:  TST      #42      ;RUNNING IN AUTO MODE?
    
```

1327	010360	001045			BNE	7\$;BR IF YES
1328	010362	104402	005672		TYPE	.MCRLF		
1329	010366	104403			INSTR			;GET TEST NO.
1330	010370	006130			MTSTN			
1331	010372	104405			PARAM			
1332	010374	000001			1			
1333	010376	001000			1000			
1334	010400	001226			TSTNO			
1335	010402	000			0			
1336	010403	001			.BYTE			
1337	010404	012700	022322		.BYTE			
1338	010410	022710			5\$: MOV	#TST1,RO		
1339	010412	012737			CMP	(PC)+,(RO)		;CMP FIRST WORD TO 12737
1340	010414	001020			MOV	(PC)+,(PC)+		
1341	010416	023760	001226	000002	BNE	6\$;BR IF NOT SAME
1342	010424	001014			CMP	TSTNO,2(RO)		;DOES TSTNO MATCH?
1343	010426	022760	001226	000004	BNE	6\$;BR IF NO
1344	010434	001010			CMP	#TSTNO,4(RO)		;IS LAST WORD OK?
1345	010436	010037	001214		BNE	6\$;BR IF NO
1346	010442	104402	005755		MOV	RO,RETURN		;IT IS A LEGAL TEST SO DO IT
1347	010446	042737	000002	001236	TYPE	MA		
1348	010454	000412			BIC	#SW01,STRTSW		
1349	010456	005720			BR	8\$		
1350	010460	020027	026472		6\$: TST	(RO)+		;POP RO
1351	010464	001351			CMP	RO,#TLAST+10		;AT END YET?
1352	010466	104402	005666		BNE	5\$;BR IF NO
1353	010472	000730			TYPE	MQM		;YES ILLEGAL TEST NO.
1354					BR	4\$;TRY AGAIN
1355	010474	012737	022322	001214	7\$: MOV	#TST1,RETURN		;PREPARE RETURN ADDRESS
1356	010502	013701	001404		8\$: MOV	DMCSR,R1		;R1 = BASE DMC11 ADDRESS
1357	010506	000177	170502		JMP	RETURN		;GO START TESTING.
1358								
1359								
1360								
1361								
1362								
1363								
1364								
1365								
1366								
1367								
1368	010512							
1369	010512	000005			AUTO.SIZE:	RESET		;INSURE A BUS INIT.
1370	010514	012702	001500		CSRMAP: MOV	#DM.MAP,R2		;LOAD MAP POINTER.
1371	010520	005022			1\$: CLR	(R2)+		;ZERO ENTIRE MAP
1372	010522	022702	001700		CMP	#DM.END,R2		;ALL DONE?
1373	010526	001374			BNE	1\$;BR IF NO
1374	010530	005037	001310		CLR	DMNUM		;SET OCTAL NUMBER OF DMC11'S TO 0
1375	010534	012702	001500		MOV	#DM.MAP,R2		;R2 POINTS TO DMC MAP
1376	010540	005037	001306		CLR	DMACTV		;CLEAR ACTIVE
1377	010544	032737	000001	001236	BIT	#SW00,STRTSW		;QUESTIONS?
1378	010552	001002			BNE	+6		;BR IF YES
1379	010554	000137	011252		JMP	7\$;IF NO SKIP QUESTIONS
1380	010560	012737	000001	001256	MOV	#1,TEMPS		;START WITH 1
1381	010566	104403			INSTR			
1382	010570	006450			NUM			

;ROUTINE USED TO "AUTO SIZE" THE DMC11
 ;CSR AND VECTOR.
 ;NOTE: THE CSR MAY BE ANY WHERE IN THE FLOATING
 ADDRESS RANGE (160000:164000)
 AND THE VECTOR MAY BE ANY WHERE IN THE
 FLOATING VECTOR RANGE (300:770)

1383	010572	104405			PARAM		
1384	010574	000001			1		
1385	010576	000020			16.		
1386	010600	001252			TEMP3		
1387	010602	000			.BYTE	0	
1388	010603	001			.BYTE	1	
1389	010604	013737	001252	001310	MOV	TEMP3,DMNUM	;DMNUM = HOW MANY
1390	010612	104402	005672		12\$:	TYPE	.MCRLF
1391	010616	104410			CONVRT		;TYPE WHICH DMC IS BEING DONE
1392	010620	012002			WHICH		;TEMPS IS WHICH DMC
1393	010622	005237	001256		INC	TEMPS	
1394	010626	104403			INSTR		
1395	010630	006510			CSR		
1396	010632	104405			PARAM		
1397	010634	160000			160000		
1398	010636	164000			164000		
1399	010640	001254			TEMP4		
1400	010642	000			.BYTE	0	
1401	010643	001			.BYTE	1	
1402	010644	013722	001254		MOV	TEMP4,(R2)+	;STORE CSR IN MAP
1403	010650	104403			INSTR		
1404	010652	006526			VEC		
1405	010654	104405			PARAM		
1406	010656	000000			0		
1407	010660	000776			776		
1408	010662	001254			TEMP4		
1409	010664	000			.BYTE	0	
1410	010665	001			.BYTE	1	
1411	010666	013712	001254		MOV	TEMP4,(R2)	;STORE VECTOR IN MAP
1412	010672	104402			10\$:	TYPE	
1413	010674	006547			PRIC		;ASK WHAT BR LEVEL
1414	010676	004737	012266		JSR	PC,INTTY	;GET RESPONSE
1415	010702	022703	000024		CMP	#24,R3	
1416	010706	101014			BHI	50\$;BR IF LESS THAN 4
1417	010710	022703	000027		CMP	#27,R3	
1418	010714	103411			BLO	50\$;BR IF GREATER THAN 7
1419	010716	012704	000011		MOV	#11,R4	;R4 = NUMBER OF SHIFTS
1420	010722	006303			ASL	R3	;SHIFT R3 LEFT
1421	010724	005304			DEC	R4	;DEC SHIFT COUNT
1422	010726	001375			BNE	-4	;BR IF NOT DONE
1423	010730	042703	170777		BIC	#170777,R3	;BIC UNWANTED BITS
1424	010734	050312			BIS	R3,(R2)	;PUT BR LEVEL IN STATUS MAP
1425	010736	000403			BR	8\$;CONTINUE
1426	010740	104402			50\$:	TYPE	
1427	010742	005666			MQM		;RESPONSE IS OUT OF LIMITS
1428	010744	000752			BR	10\$;TRY AGAIN
1429	010746	104402			9\$:	TYPE	
1430	010750	006606			CRAM		;DOES DMC HAVE CRAM?
1431	010752	004737	012266		JSR	PC,INTTY	;GET REPLY
1432	010756	022703	000131		CMP	#131,R3	
1433	010762	001427			BEQ	9\$;YES
1434	010764	022703	000116		CMP	#116,R3	;NO
1435	010770	001403			BEQ	40\$;NOT A Y OR N
1436	010772	104402			TYPE		;TYPE ""
1437	010774	005666			MQM		
1438	010776	000763			BR	9\$;ASK AGAIN

H04

1439	011000	104402		40\$:	TYPE		
1440	011002	007340			SPEED		:DMC11-AR OR DMC11-AL?
1441	011004	004737	012266		JSR	PC,INTTY	:GET RESPONSE
1442	011010	022703	000122		CMP	#12,R3	:IS IT R
1443	011014	001414			BEQ	16\$:BR IF REMOTE
1444	011016	022703	000114		CMP	#114,R3	:IS IT L
1445	011022	001403			BEQ	41\$:BR IF LOCAL
1446	011024	104402			TYPE		
1447	011026	005666			MQM		
1448	011030	000763			BR	40\$:TRY AGAIN
1449	011032	052762	000002 000004	41\$:	BIS	#BIT1,4(R2)	:SET BIT1 IN STAT3
1450	011040	000402			BR	16\$:CONTINUE
1451	011042	052712	100000	9\$:	BIS	#BIT15,(R2)	:SET BIT 15 IF CRAM
1452	011046	104402		16\$:	TYPE		
1453	011050	006704			MODU		:ASK WHICH LINE UNIT
1454	011052	004737	012266		JSR	PC,INTTY	:GET REPLY
1455	011056	022703	000021		CMP	#21,R3	: "1"
1456	011062	001417			BEQ	30\$	
1457	011064	022703	000022		CMP	#22,R3	: "2"
1458	011070	001412			BEQ	31\$	
1459	011072	022703	000116		CMP	#116,R3	: "N"
1460	011076	001403			BEQ	32\$	
1461	011100	104402			TYPE		
1462	011102	005666			MQM		:IF NOT A 1,2 OR N TYPE "?"
1463	011104	000760			BR	16\$:TRY AGAIN
1464	011106	052722	010000	32\$:	BIS	#BIT12,(R2)+	:SET BIT 12 IN STAT2 IF NO LU
1465	011112	022222			CMP	(R2)+,(R2)+	:POP OVER STAT2 AND STAT3
1466	011114	000447			BR	33\$	
1467	011116	052712	020000	31\$:	BIS	#BIT13,(R2)	:SET BIT 13 IN STAT2 IF M8202
1468	011122	104402		30\$:	TYPE		
1469	011124	007114			CONN		:ASK IF LOOP-BACK IS ON
1470	011126	004737	012266		JSR	PC,INTTY	:GET REPLY
1471	011132	022703	000131		CMP	#131,R3	:Y
1472	011136	001406			BEQ	17\$	
1473	011140	022703	000116		CMP	#116,R3	:N
1474	011144	001406			BEQ	18\$	
1475	011146	104402			TYPE		
1476	011150	005666			MQM		:IF NOT Y OR N TYPE "?"
1477	011152	000763			BR	30\$:TRY AGAIN
1478	011154	052722	040000	17\$:	BIS	#BIT14,(R2)+	:TURNAROUND IS CONNECTED
1479	011160	000402			BR	19\$	
1480	011162	042722	040000	18\$:	BIC	#BIT14,(R2)+	:NO TURNAROUND
1481	011166			19\$:			
1482	011166	104403			INSTR		
1483	011170	007016			LINE		
1484	011172	104405			PARAM		
1485	011174	000000			0		
1486	011176	000377			377		
1487	011200	001254			TEMP4		
1488	011202	000			.BYTE	0	
1489	011203	001			.BYTE	1	
1490	011204	113722	001254		MOV8	TEMP4,(R2)+	:STORE SWITCH PAC IN MAP
1491	011210	104403			INSTR		
1492	011212	007054			BM		
1493	011214	104405			PARAM		
1494	011216	000000			0		

1495	011220	000377			377			
1496	011222	001254			TEMP4			
1497	011224	000			.BYTE	0		
1498	011225	001			.BYTE	1		
1499	011226	113722	001254		MOVB	TEMP4,(R2)+	:	STORE SWITCH PAC IN MAP
1500	011232	005722			TST	(R2)+	:	POP OVER STAT3
1501	011234	005337	001252	33\$:	DEC	TEMP3	:	DEC DMC COUNT
1502	011240	001402			BEQ	34\$:	BR IF DONE
1503	011242	000137	010612		JMP	12\$:	JUMP IF NOT
1504	011246	000137	011702	34\$:	JMP	13\$:	CONTINUE
1505	011252	012701	160000	7\$:	MOV	#160000,R1	:	SET FOR FIRST ADDRESS TO BE TESTED
1506	011256	012737	011774	000004	MOV	#6\$,2#4	:	SET FOR NON-EXISTANT DEVICE TIME CUT
1507	011264	005011		2\$:	CLR	(R1)	:	CLEAR SEL0
1508	011266	005711			TST	(R1)	:	IF DMC11 DMC SR S/B 0
1509	011270	001172			BNE	3\$:	IF NO DEV ; TRAP TO 4. IF NO BIT 8 THEN NO DMC1
1510	011272	005061	000006		CLR	6(R1)	:	CLEAR SEL6
1511	011276	005761	000006		TST	6(R1)	:	IF DMC11 THEN DMRIC S/B =0!
1512	011302	001165			BNE	3\$:	BR IF NOT DMC11
1513	011304	012711	002000		MOV	#BIT10,(R1)	:	SET ROM0
1514	011310	005061	000004		CLR	4(R1)	:	CLEAR SEL4
1515	011314	012761	125252	000006	MOV	#125252,6(R1)	:	WRITE THIS TO SEL6
1516	011322	052711	020000		BIS	#BIT13,(R1)	:	WRITE IT!
1517	011326	022761	125252	000004	CMP	#125252,4(R1)	:	WAS IT WRITTEN?
1518	011334	001004			BNE	21\$:	IF NO IT IS NOT CRAM
1519	011336	052762	100000	000002	BIS	#BIT15,2(R2)	:	SET BIT15 IF CRAM
1520	011344	000431			BR	22\$:	
1521	011346	012711	001000	21\$:	MOV	#BIT9,(R1)	:	SET ROM1
1522	011352	012761	100417	000006	MOV	#100417,6(R1)	:	PUT INSTRUCTION IN SEL6
1523	011360	012711	001400		MOV	#BIT9!BIT8,(R1)	:	CLOCK INSTRUCTION (MICRO PROC PC TO 0.
1524	011364	012711	002000		MOV	#BIT10,(R1)	:	SET ROM0
1525	011370	022761	000626	000006	CMP	#626,6(R1)	:	IS IT LOCAL CROM
1526	011376	001411			BEQ	23\$:	BR IF YES
1527	011400	022761	016520	000006	CMP	#16520,6(R1)	:	IS IT REMOTE CROM?
1528	011406	001410			BEQ	22\$:	BR IF YES
1529	011410	022761	177777	000006	CMP	#-1,6(R1)	:	NO CROM?
1530	011416	001404			BEQ	22\$:	BR IF YES
1531	011420	000516			BR	3\$:	NOT A DMC
1532	011422	052762	000002	000006	23\$:	BIS	#BIT1,6(R2)	SET BIT 1 IN STAT3
1533					: AT THIS POINT IT IS ASSUMED THAT R1 HOLDS A DMC11 CSR ADDRESS.			
1534	011430	010122			22\$:	MOV	R1,(R2)+	STORE CSR IN CORE TABLE.
1535	011432	012711	001000		15\$:	MOV	#BIT9,(R1)	CLEAR LINE UNIT LOOP
1536	011436	005061	000004			CLR	4(R1)	CLEAR PORT4
1537	011442	012761	122113	000006		MOV	#122113,6(R1)	LOAD INSTRUCTION (CLR DTR)
1538	011450	052711	000400			BIS	#BIT8,(R1)	CLOCK INSTRUCTION
1539	011454	012761	021264	000006		MOV	#021264,6(R1)	LOAD INSTRUCTION
1540	011462	052711	000400			BIS	#BIT8,(R1)	CLOCK INSTRUCTION
1541	011466	122761	000377	000004		CMPB	#377,4(R1)	IS IT ALL ONES?
1542	011474	001003				.+10		BR IF NO
1543	011476	052712	010000			BIS	#BIT12,(R2)	IF YES, NO LINE UNIT, SET STATUS BIT
1544	011502	000436				BR	20\$	
1545	011504	032761	000002	000004		BIT	#BIT1,4(R1)	IS SWITCH A ONE?
1546	011512	001403				BEQ	.+10	BR IF M8201
1547	011514	052712	060000			BIS	#BIT13!BIT14,(R2)	M8202 ASSUME CONNECTOR
1548	011520	000427				BR	20\$	CONNECTOR ON)
1549	011522	032761	000010	000004		BIT	#BIT3,4(R1)	IS MRDY SET
1550	011530	001023				BNE	20\$	BR IF M8201 NO CONNECTOR (ON LINE)

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1551	011532	012761	000100	000004		MOV	#BIT6,4(R1)	:LOAD PORT4
1552	011540	012761	122113	000006		MOV	#122113,6(R1)	:LOAD INSTRUCTION
1553	011546	052711	000400			BIS	#BIT8,(R1)	:CLOCK INSTRUCTION(SET DTR)
1554	011552	012761	021264	000006		MOV	#021264,6(R1)	:LOAD INSTRUCTION
1555	011560	052711	000400			BIS	#BIT8,(R1)	:CLOCK INSTRUCTION(PEAD MODEM REG)
1556	011564	032761	000010	000004		BIT	#BIT3,4(R1)	:IS MRDY SET NOW?
1557	011572	001402				BEQ	20\$:BR IF NO CONNECTOR
1558	011574	052712	040000			BIS	#BIT14,(R2)	:SET STATUS BIT FOR CONNECTOR
1559	011600	005722			20\$:	TST	(R2)+	:POP POINTER
1560	011602	012761	021324	000006		MOV	#021324,6(R1)	:PUT INSTRUCTION IN PORT6
1561	011610	012711	001400			MOV	#BIT9:BIT8,(R1)	:PORT4+LU IS
1562	011614	156122	000004			BISB	4(R1),(R2)+	:STORE DDCMP LINE # IN TABLE
1563	011620	012761	021344	000006		MOV	#021344,6(R1)	:PORT6+INSTRUCTION
1564	011626	012711	001400			MOV	#BIT8:BIT9,(R1)	:CLOCK INSTR.
1565	011632	156122	000004			BISB	4(R1),(R2)+	:STORE BM873 ADD IN TABLE
1566	011636	005722				TST	(R2)+	:POP OVER STAT3
1567	011640	005011				CLR	(R1)	:CLEAR ROMI
1568	011642	005237	001310			INC	DMNUM	:UPDATE DEVICE COUNTER
1569	011646	022737	000020	001310		CMP	#20,DMNUM	:ARE MAX. NO. OF DEV FOUND?
1570	011654	001412				BEQ	13\$:YES DON'T LOOK FOR ANY MORE.
1571	011656	005011			3\$:	CLR	(R1)	:CLEAR BIT 10
1572	011660	005061	000006			CLR	6(R1)	:CLEAR SEL 5
1573	011664	062701	000010		14\$:	ADD	#10,R1	:UPDATE CSR POINTER ADDRESS
1574	011670	022701	164000			CMP	#164000,R1	
1575	011674	001402				BEQ	13\$:BR IF DONE
1576	011676	000137	011264			JMP	2\$:JUMP IF NOT
1577	011702	005037	001306		13\$:	CLR	DMACTV	
1578	011706	005737	001310			TST	DMNUM	:WERE ANY DMC11'S FOUND AT ALL?
1579	011712	001423				BEQ	5\$:ERROR AUTO SIZER FOUND NO DMC11'S IN THIS SYS.
1580	011714	013701	001310			MOV	DMNUM,R1	
1581	011720	010137	001314			MOV	R1,SAVNUM	:SAVE NUMBER OF DEVICES
1582	011724	000241			4\$:	CLC		
1583	011726	006137	001306			ROL	DMACTV	:GENERATE ACTIVE REGISTER OF DEVICES.
1584	011732	005237	001306			INC	DMACTV	:SET THE BIT
1585	011736	005301				DEC	R1	
1586	011740	001371				BNE	4\$:BR IF MORE TO GENERATE
1587	011742	012737	000006	000004		MOV	#6,2#4	:RESTORE TRAP VECTOR
1588	011750	013737	001306	001312		MOV	DMACTV,SAVACT	:SAVE ACTIVE REGISTER
1589	011756	000137	012010			JMP	VECMAP	:GO FIND THE VECTOR NOW.
1590	011762	104402	005760		5\$:	TYPE	MERR2	:NOTIFY OPR THAT NO DMC11'S FOUND.
1591	011766	005000				CLR	RO	:MAKE DATA LIGHTS ZERO
1592	011770	000000				HALT		:STOP THE SHOW
1593	011772	000776				BR	.-2	:DISABLE CONT. SW.
1594	011774	012716	011664		6\$:	MOV	#14\$,(SP)	:ENTERED BY NON-EXISTANT TIME-CLT.
1595	012000	000002				RTI		:RETURN TO MAINSTREAM
1596								
1597	012002	000001			WHICH:	1		
1598	012004	002	002			.BYTE	2.2	
1599	012006	001256				TEMPS		
1600								
1601	012010	032737	000001	001236	VECMAP:	BIT	#SW00,STRTSW	
1602	012016	001114				BNE	5\$	
1603	012020	012737	000349	000022		MOV	#340,2#22	:SET IOT TRAP PRIO TO 7
1604	012026	012737	012202	000020		MOV	#4\$,2#20	:SET IOT TRAP VECTOR
1605	012034	012702	001500			MOV	#DM.MAP,R2	:SET SOFTWARE POINTER
1606	012040	012700	000300			MOV	#300,RO	:FLOATING VECTORS START HERE.

1607	012044	012701	000302		MOV	#302,R1	;PC OF IOT INSTR.
1608	012050	010120		15:	MOV	R1,(R0)+	;START FILLING VECTOR AREA
1609	012052	012721	000004		MOV	#4,(R1)+	;WITH +2; IOT
1610	012056	022021			CMP	(R0)+,(R1)+	;ADD 2 TO R0 +R1
1611	012060	020127	001000		CMP	R1,#1000	
1612	012064	101771			BLOS	15	;BR IF MORE TO FILL
1613	012066	013737	001306	001246	MOV	DMACTV,TEMP1	;STORE TEMPORALLY
1614	012074	006037	001246	25:	ROR	TEMP1	;BRING OUT A BIT
1615	012100	103063			BCC	55	;BR IF ALL DONE
1616	012102	012704	000012		MOV	#12,R4	;R4 IS INDEX REGISTER
1617	012106	016437	012252	177776	MOV	BRLVL(R4),PS	;SET PS TO 7
1618	012114	011201			MOV	(R2),R1	
1619	012116	012761	000200	000004	MOV	#200,4(R1)	
1620	012124	012711	001000		MOV	#BIT9,(R1)	;SET ROMI
1621	012130	012761	121111	000006	MOV	#121111,6(R1)	;PUT INSTRUCTION IN PORT6
1622	012136	012711	001400		MOV	#BIT9:BIT8,(R1)	;FORCE AN INTERRUPT
1623	012142	105200		75:	INCB	R0	;STALL
1624	012144	001376			BNE	.-2	;FOR TIME TO INTERUPT
1625	012146	162704	000002		SUB	#2,R4	;GET NEXT LOWEST PS LEVEL
1626	012152	001404			BEQ	65	;BR IF R4 = 0
1627	012154	016437	012252	177776	MOV	BRLVL(R4),PS	;MOVE NEXT LOWER LEVEL IN PS
1628	012162	000767			BR	75	;BR TO DELAY
1629	012164	052762	005300	000002	65:	BIS	#5300,2(R2)
1630	012172	005011		35:	CLR	(R1)	;NO INTERRUPT ASSUME 300 AT LEVEL 5 AND FIX DMC11
1631	012174	062702	000010		CLR	(R1)	;CLEAR ROMI
1632	012200	000735			ADD	#10,R2	;POP SOFTWARE POINTER
1633	012202	051662	000002		BR	25	;KEEP GOING
1634	012206	042762	000007	000002	45:	BIS	(SP),2(R2)
1635	012214	016405	012254		BIC	#7,2(R2)	;GET VECTOR ADDRESS
1636	012220	006305			MOV	BRLVL+2(R4),R5	;CLEAR JUNK
1637	012222	006305			ASL	R5	;GET BR LEVEL OF DMC11
1638	012224	006305			ASL	R5	;SHIFT LEVEL 4 PLACES
1639	012226	006305			ASL	R5	;TO THE LEFT FOR THE
1640	012230	042705	170777		ASL	R5	;STATUS TABLE
1641	012234	050562	000002		BIC	#170777,R5	;CLEAR UNWANTED BITS
1642	012240	022626			BIS	R5,2(R2)	;PUT BR LEVEL IN STATUS TABLE
1643	012242	012716	012172		CMP	(SP)+,(SP)+	;POP IOT JUNK OFF STACK
1644	012246	000002			MOV	#35,(SP)	;SET FOR RETURN
1645	012250	000207		55:	RTI	PC	
1646					RTS	PC	;ALL DONE WITH "AJTO SIZING"
1647	012252	000000			BRLVL:	0	;LEVEL 0
1648	012254	000000				0	;LEVEL 0
1649	012256	000200				200	;LEVEL 4
1650	012260	000240				240	;LEVEL 5
1651	012262	000300				300	;LEVEL 6
1652	012264	000340				340	;LEVEL 7
1653							
1654							
1655	012266	105777	166712		INTTY:	TSTB	@TKCSR
1656	012272	100375				BPL	.-4
1657	012274	017703	166706			MOV	@TKDBR,R3
1658	012300	105777	166704			TSTB	@TPCSR
1659	012304	100375				BPL	.-4
1660	012306	010377	166700			MOV	R3,@TPDBR
1661	012312	042703	000240			BIC	#BIT7:BITS,R3
1662	012316	000207				RTS	PC

L04

DZDMG MACY11 30(1046) 11-JUL-77 12:11 PAGE 34
DZDMG.P11 22-APR-77 09:29

PAGE: 0050

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1663		01000		
1664		01100	ROMMAP: 0	; POINTER TO HI OR LO SPEED MICRO-CODE
1665	012320	01200		
1666		01300	LOMAP:	; LOW SPEED (REMOTE) MICRO-CODE
1667	012322			

7	MACRO DEFINITIONS
9	REVISION 00
10	FEBRUARY 25, 1975
11	
12	REVISION 01
13	MARCH 18, 1975
14	NEW CSR BOARD CHANGES
15	
16	HARVEY M. SCHLESINGER
18	COPYRIGHT 1975 DIGITAL EQUIPMENT CORPORATION
70	MICRO INSTRUCTION DEFINITIONS
71	BRANCH INSTRUCTIO. 5
114	INDEXED BRANCH INSTRUCTIONS
150	MOVE INSTRUCTIONS
258	INPUT/OUTPUT ASSIGNMENTS
310	PROTOCOL DEPENDANT MACROS
353	DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT
361	VERSION 00A FEBRUARY 26, 1975
362	
363	HARVEY M. SCHLESINGER
364	
365	COPYRIGHT 1975, DIGITAL EQUIPMENT CORPORATION
366	
367	VERSION 00B MARCH 17, 1975
368	CSR AND MICROPROCESSOR CHANGES
369	
370	VERSION 00C NOVEMBER 6, 1975
371	RETRANSMISSION CHANGES
372	
373	VERSION 00D DECEMBER 3, 1975
374	TRANSMIT DONE CHANGES
375	
376	THE LATEST MODIFICATIONS WERE ADDED ON:
377	OCTOBER 13, 1976
378	THIS VERSION WAS USED TO BLAST THE FIRST
379	RELEASE ON OCTOBER 13, 1976
381	MICROPROCESSOR MAIN MEMORY ASSIGNMENTS
443	SCRATCH PAD ASSIGNMENTS
478	INIT--INITIALIZATION ROUTINE
533	IDLE--PROGRAM IDLE LOOP
549	BASSRV---- BASE SERVICE ROUTINE
588	NIDLE2---NO CSR ACTIVITY STATE
637	INWAIT---WAIT FOR RQI TO CLEAR
692	OUTINT---SET UP OUTPUT INTERRUPT [RDYO]
740	OUTWAI---WAIT FOR RDYO TO GO AWAY
747	CTLSRV--CNTL I SERVICE
768	TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE
790	RBASRV--RECEIVE BUFFER ADDRESS SERVICE
862	RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER
892	RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD
931	RCVC--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD, SELECT AND FINAL
950	RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES
974	RCVE--ROUTINE TO HANDLE N FIELD OF NUMBERED MESSAGE
997	RCVF--ROUTINE TO IGNORE ADDRESS
992	RCVG--ROUTINE TO IGNORE CRC1
997	RCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYPES

1056	RCVK01--ROUTINE TO HANDLE FIRST BYTE ODD RECEIVE
1063	RCVK0--PROCESS ODD CHARACTER
1093	RCVKE--HANDLE EVEN BYTES
1140	RCVI--STORE UNNUMBERED MESSAGE TYPE
1146	RCVJ--ROUTINE TO HANDLE SUBTYPE FIELD, SELECT AND FINAL
1151	RCVQ--UNNUMBERED MESSAGE RESPONSE FIELD
1161	RCVJ--UNNUMBERED MESSAGE--NUMBER FIELD
1167	RCVL--PROCESS CRC3
1192	RCVM--PROCESS CRC4--END OF DATA MESSAGE
1214	EM2--PROCESS RLD MESSAGE
1251	TMTDA--TRANSMITTER DISPATCH ROUTINE
1257	TMTA--FIRST CHARACTER OF HEADER
1304	TMTB--OUTPUT FIRST CHAR OF COUNT
1330	TMTC--OUTPUT SECOND CHAR OF COUNT
1347	TMTD--RESPONSE FIELD-NUMBERED MESSAGE
1367	TMTE--NUMBER FIELD--NUMBERED MESSAGE
1376	TMTF--NUMBERED MSG ADDRESS FIELD
1389	TF1--NUMBERED MSG HEADER EOM
1399	TMTH--ROUTINE TO OUTPUT DATA CHARACTERS
1454	TMTI--SEND UNNUMBERED TYPE FIELD
1460	TMTJ--SEND SUB-TYPE FIELD
1467	TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG)
1475	TMTL--UNNUMB MSG NUMBER FIELD
1493	TMTM--UNNUMB MSG--STATION ADDRESS
1509	TIMSRV--TIMEOUT ROUTINE--SENDS REP
1570	SNDACK--ROUTINE TO SEND AN ACK
1627	REP HANDLER
1636	START HANDLER
1649	STACK HANDLER
1685	NXMERR ---NON EXISTANT MEMORY HANDLER
1692	SELQSY--ROUTINE TO CHECK SELECT AND QSYNC AND DIDDLE LINE STATUS WORD

21 000000
22
23 000000
24 100000
25 020000
26 000000
27 040000
28 060000
29 060000
30
31 060000
32 010000
33 014000
34 000400
35 001000
36 001400
37 002000
38 002400
39 003000
40 003400
41
42 000200
43 000220
44 000240
45 000260
46 000300
47 000320
48 000340
49 000360
50 000000
51 000020
52 000040
53 000060
54 000100
55 000120
56 000140
57 000160
58
59 004000
60 010000
61 014000
62 001000
63 001400
64 000400
65 002000
66 002400
67 003000
68 003400

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

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

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

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

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

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{|}~

100000

.SBTTL MICRO INSTRUCTION DEFINITIONS
.SBTTL BRANCH INSTRUCTIONS
JUMP=100000 ;JUMP OP CODE

.SBTTL INDEXED BRANCH INSTRUCTIONS

000000

.SBTTL MOVE INSTRUCTIONS
MOVE=0 ;MOVE OPCODE

258		.SBTTL INPUT/OUTPUT ASSIGNMENTS	
259		:IBUS ASSIGNMENTS	
260	100000	INCON=0+100000	: IN CONTROL CSR
261	100020	MAIN=20+100000	: MAINTAINENCE REGISTER
262	100040	OCON=40+100000	: OUT CONTROL CSR
263	100060	UBADDR=60+100000	: UNUSED
264	100100	PORT1=100+100000	: CSR4
265	100120	PORT2=120+100000	: CSR5
266	100140	PORT3=140+100000	: CSR6
267	100160	PORT4=160+100000	: CSR7
268	100200	NPR=200+100000	: NPR CONTROL
269	100220	UBBR=220+100000	: BR(INTERRUPT)CONTROL
270	000000	INDAT1=0	: INPUT DATA LOW BYTE
271	000020	INDAT2=20	: INPUT DATA HIGH BYTE
272	000140	IOBA1=140	: OUTPUT BA LOW BYTE
273	000160	IOBA2=160	: OUTPUT BA HIGH BYTE
274	000100	IIBA1=100	: INPUT BA LOW BYTE
275	000120	IIBA2=120	: INPUT BA HIGH BYTE
276	000200	RCVDAT=200	: RECEIVE DATA
277	000220	TMTCON=220	: TMTR CONTROL
278	000240	RCVCON=240	: RCVR CONTROL
279	000260	MODEM=260	: MODEM CONTROL
280	000300	SYNREG=300	: SYN REGISTER
281	000320	LNOSW=320	: LINE NUMBER SWITCH
282	000340	BM873=340	: BM873 ADDRESS
283	000360	LUMAIN=360	: LINE UNIT MAINTAINENCE
284		:OBUS ASSIGNMENTS	
285		:EXTENDED OBUS	
286	000000	OINCON=0	: IN CONTROL CSR
287	000001	OMAIN=1	: MAINT
288	000002	OCON=2	: OUT CONTROL CSR
289	000003	OUBADD=3	: UNUSED
290	000004	OPORT1=4	: CSR4
291	000005	OPORT2=5	: CSR5
292	000006	OPORT3=6	: CSR6
293	000007	OPORT4=7	: CSR7
294	000010	ONPR=10	: NPR CONTROL
295	000011	OBR=11	: BR CONTROL
296		:UNEXTENDED OBUS	
297	000002	OUTDA1=2	: OUTPUT DATA LOW BYTE
298	000003	OUTDA2=3	: OUTPUT DATA HIGH BYTE
299	000006	OBA1=6	: OUTPUT BA LOW BYTE
300	000007	OBA2=7	: OUTPUT BA HIGH BYTE
301	000004	IIBA1=4	: INPUT BA LOW BYTE
302	000005	IIBA2=5	: INPUT BA HIGH BYTE
303	000010	TMTDAT=10	: TMTR DATA
304	000011	OTMTCO=11	: TMTR CONTROL
305	000012	ORCVC0=12	: RCVR CONTROL
306	000013	OMODEM=13	: MODEM CONTROL
307	000014	SYNC=14	: SYN REGISTER
308	000017	OLUMAN=17	: LINE UNIT MAIN*

DMC-11 MICROPROCESSOR INSTRUCTIONS
DMCNEW.MAC 21-APR-77 10:38

MACY11 30(1046) 11-JUL-77 12:18 PAGE 3-1
PROTOCOL DEPENDANT MACROS

PAGE: 0057

.SBTTL PROTOCOL DEPENDANT MACROS

.....

177777

MICPC=177777 ;INIT MICRO PC

310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
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
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400

G05

DMC-11 MICROPROCESSOR INSTRUCTIONS
LOW.MAC 13-OCT-76 14:33

MACY11 30(1046) 11-JUL-77 12:18 PAGE 5
DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT

PAGE: 0052

353
354 000000
355 000000

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

H05

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10:08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 6
DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT

PAGE: 0059

359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380

.TITLE DMC11 DDCMP PROTOCOL IMPLEMENTATION
.IDENT /V0001/
.SBTTL VERSION LJA FEBRUARY 26, 1975
.SBTTL
.SBTTL HARVEY M. SCHLESINGER
.SBTTL
.SBTTL COPYRIGHT 1975, DIGITAL EQUIPMENT CORPORATION
.SBTTL
.SBTTL VERSION 008 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
.SBTTL RELEASE ON OCTOBER 13, 1976

381
382
383 000000
384 000001
385 000002
386 000003
387 000006
388 000007
389 000010
390 000011
391 000012
392 000013
393 000014
394 000015
395 000016
396 000017
397 000022
398 000023
399 000024
400 000025
401 000032
402 000037
403 000044
404 000051
405 000056
406 000063
407 000070
408 000071
409 000072
410 000073
411 000101
412 000107
413 000115
414 000123
415 000131
416 000137
417 000145
418 000153
419 000154
420 000155
421 000156
422 000157
423 000160
424 000161
425 000162
426 000163
427
428
429 000165
430
431
432
433 000170
434 000171
435 000172
436 000173

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

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10:08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 6-2
MICROPROCESSOR MAIN MEMORY ASSIGNMENTS

PAGE: 0061

437	000174
438	000240
439	000241
440	000242
441	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=40C	;MAIN MEMORY END

443		.SBTTL	SCRATCH PAD ASSIGNMENTS
444	000000	SP0=0	;SP0---SCRATCH REGISTER
445	000001	SP1=1	;SP1---PORT STATUS WORD
446			;BIT ASSIGNMENTS
447			;BIT0---INIT MODE
448			;BIT1---SEC STATION SELECT(UNUSED)
449			;BIT2---NO BUFFER ASSIGNED IN BOOT MODE
450			;BIT3---OLE RECEIVED WHILE NOT IN MAINT MODE
451			;BIT4---INTERRUPT PENDING
452			;BIT5---DISCONNECT ERROR
453			;BIT7---BOOT MODE
454	000002	SP2=2	;SP2---TRANSMIT STATE POINTER
455	000003	SP3=3	;SP3---RECEIVE STATE POINTER
456	000004	SP4=4	;SP4---END RECV ADDRESS
457	000005	SP5=5	;SP5---END RECEIVE ADDRESS
458	000006	SP6=6	;SP6---END TRANSMIT ADDRESS
459	000007	SP7=7	;SP7---END TRANSMIT ADDRESS
460	000010	SP10=10	;SP10---LINE STATUS WORD
461			;BIT ASSIGNMENTS
462			;BIT0---UNNUMB PENDING
463			;BIT1---MESSAGE IN PROGRESS
464			;BIT2---LINE HAS GONE IDLE
465			;BIT3---START RECEIVED
466			;BIT4---CLEAR ACTIVE ON END
467			;BIT5---START MODE
468			;BIT6---HALF DUPLEX
469			;BIT7---OK TO SEND
470	000011	SP11=11	;SP11---R FIELD
471	000012	SP12=12	;SP12---N FIELD
472	000013	SP13=13	;SP13---TYPE
473	000014	SP14=14	;SP14---RECEIVE LINK IMAGE
474	000015	SP15=15	;SP15---TIMER ENTRY---NUMBER OF ONE SECOND TICKS
475	000016	SP16=16	;SP16---POINTER TO TMT LINK COPY IN MAIN MEM
476	000017	SP17=17	;SP17---LAST MESSAGE ACKNOWLEDGED


```

478 .SBTTL INIT--INITIALIZATION ROUTINE
479 ;ZEROS MAIN MEMORY
480 ;LOOPS WAITING FOR RECEIVE DATA(BOOT?)
481 ;OR FOR RQI TO BE SET
482 ;WILL ACCEPT ONLY BASE FORMAT. ALL OTHERS WILL RETURN A PROCEDURE ERROR
483 ;
484 ;AT INITIALIZATION --- THE HARDWARE CLEARS THE BR AND MAR
485 ;=12322
486 012322 012322 INIT: SP BR,SELB,SP0 ;CLEAR SP0
(1) (1) 012322 000000 MICPC=MICPC+1
(1) 012322 063220 <MOVE!SPX!BR!SELB!SP0>
487 012324 012324 SP BR,SELB,SP3 ;PAGE ONE TRANSFER ADDRESS
(1) (1) 012324 000001 MICPC=MICPC+1
(1) 012324 063223 <MOVE!SPX!BR!SELB!SP3>
488 012326 012326 SP BR,SELB,SP17 ;CLEAR SP17
(1) (1) 012326 000002 MICPC=MICPC+1
(1) 012326 063237 <MOVE!SPX!BR!SELB!SP17>
489 012330 012330 OUT BR,<SELA!OINCON> ;ZERO THE IN CONTROL CSR
(1) (1) 012330 000003 MICPC=MICPC+1
(1) 012330 061200 <MOVE!WROUTX!BR!<SELA!OINCON>>
490 012332 012332 OUT BR,<SELA!OOCON> ;ZERO THE OUT CONTROL CSR
(1) (1) 012332 000004 MICPC=MICPC+1
(1) 012332 061202 <MOVE!WROUTX!BR!<SELA!OOCON>>
491 012334 012334 SP IMM,370,SP10 ;WRITE 5 ONE BITS TO THE HIGH ORDER
(1) (1) 012334 000005 MICPC=MICPC+1
(1) 012334 003370 <MOVE!SPX!IMM!370!SP10>
492 ;BITS OF SP10
493 012336 012336 S$: SP BR,AA,SP10 ;SHIFT SP10 LEFT SETTING CARRY THE
(1) (1) 012336 000006 MICPC=MICPC+1
(1) 012336 063130 <MOVE!SPX!BR!AA!SP10>
494 ;FIRST 5 TIMES THRU THE LOOP
495 012340 012340 MEMINC BR,ADDC!SP3 ;WRITE A ONE TO THE FIRST 5 MEMORY
(1) (1) 012340 000007 MICPC=MICPC+1
(1) 012340 076423 <MOVE!WRMEM!INCMAR!BR!<ADDC!SP3>>
496 ;LOCATIONS AND ZERO THE REST
497 012342 012342 SP BR,INCA,SP0 ;INCREMENT COUNTER
(1) (1) 012342 000010 MICPC=MICPC+1
(1) 012342 063060 <MOVE!SPX!BR!INCA!SP0>
498 012344 012344 Z 10$ ;ALL DONE
(1) (1) 012344 000011 MICPC=MICPC+1
(1) 012344 101413 <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
499 012346 012346 ALWAYS S$ ;KEEP GOING
(1) (1) 012346 000012 MICPC=MICPC+1
(1) 012346 100406 <JUMP!ALCOND!<5$-INIT&3000*4>!<5$-INIT&777/2>>
500 012350 012350 10$: SPBR IMM,1,SP1 ;WRITE A 1 TO THE BR AND SP1
(1) (1) 012350 000013 MICPC=MICPC+1
(1) 012350 003401 <MOVE!SPBRX!IMM!1!SP1>
501 012352 012352 SP BR,SELB,SP11 ;WRITE A 1 TO SP11
(1) (1) 012352 000014 MICPC=MICPC+1
(1) 012352 063231 <MOVE!SPX!BR!SELB!SP11>
502 012354 012354 SP BR,SELB,SP12 ;WRITE A 1 TO SP12
(1) (1) 012354 000015 MICPC=MICPC+1
(1) 012354 063232 <MOVE!SPX!BR!SELB!SP12>
503 012356 012356 LDMA IMM,PRTST ;POINT MAR TO UNNUM MSG SKELETON
(1) (1) 012356 000016 MICPC=MICPC+1
(1) 012356 010162 <MOVE!LDMAR!IMM!<PRTST&377>>

```

```

504 012360          PSTATI NIDLE2          ;WRITE PORT IDLE STATE
(1) 012360          MEMINC IMM,<<NIDLE2-INIT&777/2>>
(2) 012360 000017   MICPC=MICPC+1
(2) 012360 016520   <MOVE!WRMEM!INCMAR!IMM!<<NIDLE2-INIT&777/2>>>
505 012362          BRWRT IMM,226          ;WRITE SYNC TO MEMORY
(1) 012362 000020   MICPC=MICPC+1
(1) 012362 000626   <MOVE!WRTEBR!IMM!<226>>
506 012364          OUTPUT BR,SELB!SYNC    ;LOAD THE SYNC REGISTER
(1) 012364 000021   MICPC=MICPC+1
(1) 012364 062234   <MOVE!WROUT!BR!<SELB!SYNC>>
507 012366          MEMINC IMM,3          ;REP
(1) 012366 000022   MICPC=MICPC+1
(1) 012366 016403   <MOVE!WRMEM!INCMAR!IMM!<3>>
508 012370          MEM IMM,2            ;NAK
(1) 012370 000023   MICPC=MICPC+1
(1) 012370 002402   <MOVE!WRMEM!IMM!<2>>
509 012372          SP MEMX!INCMAR,SELB,SP15 ;SET STARTING COUNT
(1) 012372 000024   MICPC=MICPC+1
(1) 012372 057235   <MOVE!SPX!MEMX!INCMAR!SELB!SP15>
510 012374          MEMINC IMM,6          ;START
(1) 012374 000025   MICPC=MICPC+1
(1) 012374 016406   <MOVE!WRMEM!INCMAR!IMM!<6>>
511 012376          MEMINC IMM,7          ;STACK
(1) 012376 000026   MICPC=MICPC+1
(1) 012376 016407   <MOVE!WRMEM!INCMAR!IMM!<7>>
512 012400          MEMINC IMM,1          ;ACK
(1) 012400 000027   MICPC=MICPC+1
(1) 012400 016401   <MOVE!WRMEM!INCMAR!IMM!<1>>
513 012402          LDMA IMM,STC          ;LOAD ADDRESS OF LAST TMT CHAIN
(1) 012402 000030   MICPC=MICPC+1
(1) 012402 010070   <MOVE!LDMAR!IMM!<STC&377>>
514 012404          MEMINC IMM,TML1      ;STORE ADDRESS OF FIRST TMT LINK
(1) 012404 000031   MICPC=MICPC+1
(1) 012404 016473   <MOVE!WRMEM!INCMAR!IMM!<TML1>>
515 012406          MEMINC IMM,TML1
(1) 012406 000032   MICPC=MICPC+1
(1) 012406 016473   <MOVE!WRMEM!INCMAR!IMM!<TML1>>
516 012410          MEMINC IMM,TML1
(1) 012410 000033   MICPC=MICPC+1
(1) 012410 016473   <MOVE!WRMEM!INCMAR!IMM!<TML1>>
517 012412          LDMA IMM,SRC          ;LOAD ADDRESS OF LAST RECV CHAIN
(1) 012412 000034   MICPC=MICPC+1
(1) 012412 010022   <MOVE!LDMAR!IMM!<SRC&377>>
518 012414          MEMINC IMM,RCL1      ;SET UP ADDRESS OF FIRST RECV LINK
(1) 012414 000035   MICPC=MICPC+1
(1) 012414 016425   <MOVE!WRMEM!INCMAR!IMM!<RCL1>>
519 012416          MEMINC IMM,RCL1
(1) 012416 000036   MICPC=MICPC+1
(1) 012416 016425   <MOVE!WRMEM!INCMAR!IMM!<RCL1>>
520 012420          MEMINC IMM,RCL1
(1) 012420 000037   MICPC=MICPC+1
(1) 012420 016425   <MOVE!WRMEM!INCMAR!IMM!<RCL1>>
521 012422          LDMA IMM,NXTINT      ;ADDRESS OF NEXT INTERRUPT POINTER TO MAR
(1) 012422 000040   MICPC=MICPC+1
(1) 012422 010240   <MOVE!LDMAR!IMM!<NXTINT&377>>
522 012424          MEMINC IMM,INTSTK    ;INITIALIZE NEXT INTERRUPT POINTER

```

(1)		000041	MICPC=MICPC+1	
(1)	012424	016642	<MOVE!WRMEM!INCMAR!IMM!<INTSTK>>	
523	012426		MEM IMM INTSTK	;INITIALIZE INSERTION POINTER
(1)		000042	MICPC=MICPC+1	
(1)	012426	002642	<MOVE!WRMEM!IMM!<INTSTK>>	
524	012430		BRWRT IMM,200	;WRITE THE RUN BIT TO THE BR
(1)		000043	MICPC=MICPC+1	
(1)	012430	000600	<MOVZ!WRTEBR!IMM!<2U0>>	
525	012432		OUT BR,<SELB!OMAIN>	;WRITE THE RUN BIT TO MAINT CSR
(1)		000044	MICPC=MICPC+1	
(1)	012432	061221	<MOVE!WROUTX!BR!<SELB!OMAIN>>	
526				;FALL INTO IDLE LOOP

533
 534
 535
 536
 537
 538 012434
 (1) 000045
 (1) 012434 063610
 539 012436
 (1) 300046
 (1) 012436 112400
 540 012440
 (1) 000047
 (1) 012440 112000
 541 012442
 (1) 000050
 (1) 012442 100452
 542
 543 012444
 (1) 000051
 (1) 012444 063222
 544 012446
 (1) 000052
 (1) 012446 020640
 545 012450
 (1) 000053
 (1) 012450 167203
 546 012452
 (1) 000054
 (1) 012452 010162
 547 012454
 (1) 000055
 (1) 012454 140620

```

.SBTTL IDLE--PROGRAM IDLE LOOP
;PROGRAM IDLE LOOP
;DISPATCHES TO APPROPRIATE SERVICE ROUTINES
;USES STATE POINTERS FOR TMT,RCV,CSR ACTIVITY
IDLE:  BRWRT BR,SELB,SP10          ;READ TRANSMIT STATUS WORD FROM SP10 TO BR
      MICPC=MICPC+1
      <MOVE!WRTBR!BR!SELB!SP10>
      BR1  TMTDA                    ;IF DATA TO SEND-- BRANCH
      MICPC=MICPC+1
      <JUMP!BR1CON!TMTDA-INIT&3000*4!TMTDA-INIT&777/2>
      BR0  TMTDA                    ;IF DATA TO SEND-- BRANCH
      MICPC=MICPC+1
      <JUMP!BR0CON!TMTDA-INIT&3000*4!TMTDA-INIT&777/2>
      ALWAYS I1
      MICPC=MICPC+1
      <JUMP!ALCOND!I1-INIT&3000*4!I1-INIT&777/2>
      .
XEXIT: SP  BR,SELB,SP2
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
I1:   BRWRT IBUS,RCVCON            ;READ LINE UNIT RECEIVE CONTROL WORD
      MICPC=MICPC+1
      <MOVE!WRTBR!IBUS!RCVCON>
      BR4  BR,SELB,SP3,PAGE1        ;BRANCH BASED UPON RCV STATE
      MICPC=MICPC+1
      <JUMP!BR4CON!BR!SELB!SP3!PAGE1>
I2:   LDMA  IMM,PRST              ;ADDRESS PORT STATE
      MICPC=MICPC+1
      <MOVE!LDMA!IMM!PRST&377>
      .ALWAY MEMX,SELB,0           ;INDEX
      MICPC=MICPC+1
      <JUMP!ALCOND!MEMX!SELB!C>
  
```

549
550 012456
(1) 012456
(2) 000056
(2) 012456 002520
551 012460
(1) 000057
(1) 012460 010017
552 012462
(1) 000060
(1) 012462 136500
553 012464
(1) 000061
(1) 012464 136520
554 012466
(1) 000062
(1) 012466 122560
555 012470
(1) 000063
(1) 012470 123000
556 012472
(1) 000064
(1) 012472 000500
557 012474
(1) 000065
(1) 012474 061260
558 012476
(1) 000066
(1) 012476 000520
559 012500
(1) 000067
(1) 012500 062233
560 012502
(1) 000070
(1) 012502 040620
561 012504
(1) 000071
(1) 012504 103100
562 012506
(1) 000072
(1) 012506 010153
563 012510
(1) 000073
(1) 012510 016406
564 012512
(1) 000074
(1) 012512 002700
565 012514
(1) 000075
(1) 012514 063161
566 012516
(1) 000076
(1) 012516 000641
567 012520
(1) 000077
(1) 012520 110733

```

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

```

BS1:

568 012522
 (1) 000100
 (1) 012522 003004
 569 012524
 (1) 000101
 (1) 012524 063070
 570
 571 012526
 (1) 000102
 (1) 012526 010017
 572 012530
 (1) 000103
 (1) 012530 000736
 573 012532
 (1) 000104
 (1) 012532 110462
 574 012534
 (1) 000105
 (1) 012534 010156
 575 012536
 (1) 000106
 (1) 012536 043310
 576 012540
 (1) 000107
 (1) 012540 057231
 577 012542
 (1) 000110
 (1) 012542 057232
 578 012544
 (1) 000111
 (1) 012544 043237
 579 012546
 (1) 000112
 (1) 012546 063170
 580
 581 012550
 (1) 000113
 (1) 012550 000520
 582 012552
 (1) 000114
 (1) 012552 063233
 583
 584 012554
 (1) 000115
 (1) 012554 063161
 585 012556
 (1) 000116
 (1) 012556 000600
 586 012560
 (1) 000117
 (1) 012560 110733

```

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

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

BS2:    LDMA    IMM,IMG10
        MICPC=MICPC+1
        <MOVE!LDMA!IMM!<IMG10&377>>
        SP      MEMX,AORB,SP10      ;RESTORE BIT 1 OF SP10
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!AORB!SP10>
        SP      MEMX,INCMAR,SELB,SP11 ;RESTORE SP11
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!INCMAR!SELB!SP11>
        SP      MEMX,INCMAR,SELB,SP12 ;RESTORE SP12
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!INCMAR!SELB!SP12>
        SP      MEMX,SELB,SP17      ;RESTORE      SP17
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!SELB!SP17>
        SP      BR,DECA,SP10       ;TURN OFF UNNUM MESSAGE PENDING AND
        MICPC=MICPC+1
        <MOVE!SPX!BR!DECA!SP10>
        STATE   NIDLE2             ;ZERO THE BRG
        MICPC=MICPC+1              ;PORT STATUS
        <MOVE!WTEBR!IMM!<NIDLE2-INIT&777/2>>
        SP      BR,SELB,SP13       ;SAVE IN SP13 - NOTE THAT STATUS READ INTO
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP13>
        SP      BR,DECA,SP1        ;RAM WAS TABLE UPDATE WHICH SAVED STATUS IN SP13
        MICPC=MICPC+1              ;CLEAR INIT MODE
        <MOVE!SPX!BR!DECA!SP1>
        BRWTE   IMM,200            ;SET OK TO SEND
        MICPC=MICPC+1
        <MOVE!WTEBR!IMM!<200>>
        ALWAYS  SA3
        MICPC=MICPC+1
        <JUMP!ALCOND!<SA3-INIT&3000*4>!<SA3-INIT&777/2>>

```

588
589 012562 000120
(1) 012562 060601
594 012564 000121
(1) 012564 103220
596 012566 000122
(1) 012566 123400
597 012570 000123
(1) 012570 001620
598 012572 000124
(1) 012572 103155
599
600
601 012574 000125
(1) 012574 060601
606 012576 000126
(1) 012576 102131
608 012600 000127
(1) 012600 123620
609 012602 000130
(1) 012602 113245
614 012604 000131
(1) 012604 023660
615 012606 000132
(1) 012606 060520
616 012610 000133
(1) 012610 103150
617 012612 000134
(1) 012612 060610
618 012614 000135
(1) 012614 001620
619 012616 000136
(1) 012616 103045
620 012620 000137
(1) 012620 060521
621 012622 000140
(1) 012622 102445
622 012624 000141
(1)

```

.SBTTL NIDLE2---NO CSR ACTIVITY STATE
NIDLE2: BRWRT BR SELA!SP1 ;READ PORT STATUS WORD
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<SELA!SP1>>
        BR4 OUTINT
        MICPC=MICPC+1
        <JUMP!BR4CON!<OUTINT-INIT&3000*4>!<OUTINT-INIT&777/2>>
        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 AVCID
        ;A RACE IN MICRO-P READ/UNIBUS WRITE
        ;READ PORT STATUS
NIDLE6: BRWRT BR SELA!SP1
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<SELA!SP1>>
        BRO 10$
        MICPC=MICPC+1
        <JUMP!BROCON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
        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>>
10$: SPBR IBUS,MODEM,SPO ;READ MODEM CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!MODEM!SPO>
        BRWRT BR AA!SPO ;SHIFT IT LEFT
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<AA!SPO>>
        BR4 SETDSR ;IF DSR SET, CLEAR FLAG
        MICPC=MICPC+1
        <JUMP!BR4CON!<SETDSR-INIT&3000*4>!<SETDSR-INIT&777/2>>
        BRWRT BR SELA!SP10 ;READ LINE STATUS WORD
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<SELA!SP10>>
        BRSHFT
        MICPC=MICPC+1
        <MOVE!SHFTBR!WRTEBR!SELB>
        BR4 IDLE ;START MODE
        MICPC=MICPC+1
        <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        BRWRT BR AA!SP1 ;READ PORT STATUS WORD
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<AA!SP1>>
        BR1 IDLE ;INIT MODE
        MICPC=MICPC+1
        <JUMP!BR1CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        BR7 IDLE ;DISCONNECT ERROR ALREADY SENT
        MICPC=MICPC+1

```


(1) 012624 103445
 623 012626 000142
 (1) 012626 123420
 624 012630 000143
 (1) 012630 060400
 625 012632 000144
 (1) 012632 103045
 626 012634 000145
 (1) 012634 000500
 627 012636 000146
 (1) 012636 063301
 628 012640 000147
 (1) 012640 114671
 629 012642 000150
 (1) 012642 000677
 630 012644 000151
 (1) 012644 100652

```

<JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
SPBR IBUS MAIN,SPO ;READ THE MAINT REGISTER
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!MAIN!SPO>
BRWRT BR,ADD!SPO ;SHIFT LEFT
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<ADD!SPO>>
BR4 IDLE ;LU LOOP -- EXIT
MICPC=MICPC+1
<JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
BRWRT IMM,100 ;WRITE DISCONNECT ERROR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<100>>
SP BR,AORB,SP1 ;FLAG ERROR RECORDED
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP1>
ALWAYS ERRXX ;MAKE A CONTROL OUT
MICPC=MICPC+1
<JUMP!ALCOND!<ERRXX-INIT&3000*4>!<ERRXX-INIT&777/2>>
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>>
    
```

637
 638 012646 000152
 (1) 012646 123400
 639 012650 000153
 (1) 012650 060520
 640 012652 000154
 (1) 012652 103557
 641
 642 012654 000155
 (1) 012654 123400
 643 012656 000156
 (1) 012656 103566
 644 012660 000157
 (1) 012660 002555
 (2) 012660 000160
 (2) 012662 060520
 645 012662 000161
 (1) 012664 117454
 646 012664 000162
 (1) 012666 002552
 (2) 012666 000163
 (2) 012670 000600
 647 012666 000164
 (1) 012672 061300
 648 012670 000165
 (1) 012674 100445
 649 012672 000166
 (1) 012676 001620
 650 012674 000167
 (1) 012700 103125
 651 012676 000170
 (1) 012702 123400
 652 012700 000171
 (1) 012704 102605
 653 012702 000172
 (1) 012706 102177

```

INWAIT: .SBTTL INWAIT---WAIT FOR RQI TO CLEAR
        SPBR   IBUS,INCON,SPO           ;READ INPLT CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!INCON!SPO>
        BRWRT  BR, <AA!SPO>           ;SHIFT IT LEFT
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<AA!SPO>>
        BR7    NIDLE3                 ;INTERRUPT ENABLE HAS BEEN SET
        MICPC=MICPC+1
        <JUMP!BR7CON!<NIDLE3-INIT&3000*4>!<NIDLE3-INIT&777/2>>

INWAT1: SPBR   IBUS,INCON,SPO           ;READ THE INPUT CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!INCON!SPO>
        BR7    INWAT2                 ;READY IN STILL SET
        MICPC=MICPC+1
        <JUMP!BR7CON!<INWAT2-INIT&3000*4>!<INWAT2-INIT&777/2>>

NIDLE3: PSTATE INWAT1                 ;UPDATE STATE TO INPUT
        MEM    IMM, <<INWAT1-INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<<INWAT1-INIT&777/2>>>
        BRWRT  BR,AA!SPO             ;SHIFT CSR LEFT
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<AA!SPO>>
        BR7    ININT
        MICPC=MICPC+1
        <JUMP!BR7CON!<ININT-INIT&3000*4>!<ININT-INIT&777/2>>

        PSTATE INWAIT                 ;UPDATE STATE POINTER TO NO INTERRUPT GENERATED
        MEM    IMM, <<INWAIT-INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<<INWAIT-INIT&777/2>>>

NIDLE4: BRWRT  IMM,200
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<200>>
        OUT    BR,AORB!OINCON         ;SET THE RDY!
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<AORB!OINCON>>
        ALWAYS IDLE
        MICPC=MICPC+1
        <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

INWAT2: BRSHFT                          ;SHIFT THE BR RIGHT
        MICPC=MICPC+1
        <MOVE!SHFTBR!WRTEBR!SELB>
        BR4    NIDLE6                 ;RQI SET--- GO AWAY
        MICPC=MICPC+1
        <JUMP!BR4CON!<NIDLE6-INIT&3000*4>!<NIDLE6-INIT&777/2>>

INSRV:  SPBR   IBUS,INCON,SPO           ;READ THE INPUT CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!INCON!SPO>
        BR1    30$                    ;---SENSE OR BASE
        MICPC=MICPC+1
        <JUMP!BR1CON!<30$-INIT&3000*4>!<30$-INIT&777/2>>
        BR0    10$                    ;CNTL I
        MICPC=MICPC+1
        <JUMP!BR0CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
  
```

```

661 012710 000173      BRSHFT ;MUST BE BA/CC-SHIFT FOR IN OR OUT
(1) 012710 001620      MICPC=MICPC+1
662 012712 000174      <MOVE!SHFTBR!WRTEBR!SELB>
(1) 012712 102601      BR1 15$
663 012714 000175      MICPC=MICPC+1
(1) 012714 002703      <JUMP!BRCON!<15$-INIT&3000*4>!<15$-INIT&777/2>>
(2) 012714 002703      PSTATE TBASRV ;TRANSMITTER
664 012716 000176      MEM IMM,<<TBASRV-INIT&777/2>>
(1) 012716 100602      MICPC=MICPC+1
665 012720 000177      <MOVE!WRMEM!IMM!<<TBASRV-INIT&777/2>>>
(1) 012720 002661      ALWAYS 20$
666 012722 000200      MICPC=MICPC+1
(1) 012722 100602      <JUMP!ALCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
667 012724 000201      10$: PSTATE CTLSRV
(1) 012724 002726      MEM IMM,<<CTLSRV-INIT&777/2>>
(2) 012724 002726      MICPC=MICPC+1
668 012726 000202      <MOVE!WRMEM!IMM!<<CTLSRV-INIT&777/2>>>
(1) 012726 060601      ALWAYS 20$
669 012730 000203      MICPC=MICPC+1
(1) 012730 102211      <JUMP!ALCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
670 012732 000204      15$: PSTATE RBASRV
(1) 012732 100445      MEM IMM,<<RBASRV-INIT&777/2>>
679 012734 000205      MICPC=MICPC+1
(1) 012734 102207      <MOVE!WRMEM!IMM!<<RBASRV-INIT&777/2>>>
680 012736 000206      20$: BRWRTE BR SELA!SP1 ;INIT MODE
(1) 012736 100611      MICPC=MICPC+1
681 012740 000207      <MOVE!WRTEBR!BR!<SELA!SP1>>
(1) 012740 060601      BRO PROCER ;IF INIT MODE--ERROR
682 012742 000210      MICPC=MICPC+1
(1) 012742 102056      <JUMP!BROCON!<PROCER-INIT&3000*4>!<PROCER-INIT&777/2>>
684 012744 000211      30$: ALWAYS IDLE
(1) 012744 000500      MICPC=MICPC+1
685 012746 000212      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
(1) 012746 061260      35$: BRO 35$ ;IF BASE---PROCESS
686 012750 000213      MICPC=MICPC+1
(1) 012750 010177      <JUMP!BROCON!<35$-INIT&3000*4>!<35$-INIT&777/2>>
687 012752 000214      ALWAYS PROCER
(1) 012752 000207      MICPC=MICPC+1
(1) 012752 060601      <JUMP!ALCOND!<PROCER-INIT&3000*4>!<PROCER-INIT&777/2>>
(1) 012752 000207      35$: BRWRTE BR SELA!SP1 ;INIT MODE?
(1) 012752 060601      MICPC=MICPC+1
(1) 012752 060601      <MOVE!WRTEBR!BR!<SELA!SP1>>
(1) 012752 000210      BRO BASSRV
(1) 012752 102056      MICPC=MICPC+1
(1) 012752 102056      <JUMP!BROCON!<BASSRV-INIT&3000*4>!<BASSRV-INIT&777/2>>
(1) 012752 000211      PROCER: BRWRTE IMM,100 ;CLEAR INPUT CONTROL CSR
(1) 012752 000500      MICPC=MICPC+1
(1) 012752 000500      <MOVE!WRTEBR!IMM!<100>>
(1) 012752 000212      OUT BR AANDB!OINCON ;;
(1) 012752 061260      MICPC=MICPC+1
(1) 012752 061260      <MOVE!WROUTX!BR!<AANDB!OINCON>>
(1) 012752 000213      LDMA IMM,<<RTHRS+3>> ;ADDRESS ERROR LINK
(1) 012752 010177      MICPC=MICPC+1
(1) 012752 010177      <MOVE!LDMA!IMM!<<RTHRS+3>&377>>
(1) 012752 000214      MEMINC IMM,2
(1) 012752 000214      MICPC=MICPC+1

```

```

(1) 012752 016402
688 012754
(1) 000215
(1) 012754 002400
689 012756
(1) 000216
(1) 012756 042233
690 012760
(1) 000217
(1) 012760 114527

```

```

<MOVE!WRMEM!INCMAR!IMM!<2>>
MEM IMM,0
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<0>>
OUTPUT MEMX SELB!OMODEM ;CLEAR DATA TERMINAL READY
MICPC=MICPC+1
<MOVE!WROUT!MEMX!<SELB!OMODEM>>
ALWAYS RCEXX ;POST THE ERROR - FATAL
MICPC=MICPC+1
<JUMP!ALCOND!<RCEXX-INIT&3000*4>!<RCEXX-INIT&777/2>>

```

```

692
693 012762
698 012762
(1) 012762
(2) 000220
(2) 012762 002654
700
701 012764
(1) 000221
(1) 012764 010240
702 012766
(1) 000222
(1) 012766 050220
703 012770
(1) 000223
(1) 012770 123040
704 012772
(1) 000224
(1) 012772 055302
705 012774
(1) 000225
(1) 012774 050220
706 012776
(1) 000226
(1) 012776 074520
707
708
709 013000
(1) 000227
(1) 013000 055224
710 013002
(1) 000230
(1) 013002 055225
711 013004
(1) 000231
(1) 013004 055227
712 013006
(1) 000232
(1) 013006 055226
713
714 013010
(1) 000233
(1) 013010 103760
715
721 013012
723 013012
(1) 000234
(1) 013012 010240
724 013014
(1) 000235
(1) 013014 043220
725 013016
(1) 000236
(1) 013016 002642
726 013020
(1) 000237

```

```

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

```

```

(1) 013020 000776      <MOVE!WRTEBR!IMM!<<MMEND-2>>>
727 013022            CMP      BR,SPO          ;SHOULD WE WRAP
(1) 013022 000240      MICPC=MICPC+1
(1) 013022 060360      <SUBTC!BR!SPO>
728 013024            Z      5$          ;YES--BRANCH
(1) 013024 000241      MICPC=MICPC+1
(1) 013024 101644      <JUMP!ZCOND!<5$-INIT&3000*4>!<5$-INIT&777/2>>
729 013026            BRWRTE IMM,2      ;OFFSET FOR NEXT POINTER
(1) 013026 000242      MICPC=MICPC+1
(1) 013026 000402      <MOVE!WRTEBR!IMM!<2>>
730 013030            MEM      BR,ADD!SPO  ;UPDATE POINTER
(1) 013030 000243      MICPC=MICPC+1
(1) 013030 062400      <MOVE!WRMEM!BR!<ADD!SPO>>
731 013032            5$: SP      MEMX,SELB,SPO ;COPY POINTER TO SPO
(1) 013032 000244      MICPC=MICPC+1
(1) 013032 043220      <MOVE!SPX!MEMX!SELB!SPO>
732 013034            LDMA     IMM,NXTSP  ;PICK UP START OF IN QUEUE
(1) 013034 000245      MICPC=MICPC+1
(1) 013034 010241      <MOVE!LDMAR!IMM!<NXTSP&377>>
733 013036            CMP      MEMX,SPO  ;COMPARE TO END
(1) 013036 000246      MICPC=MICPC+1
(1) 013036 040360      <SUBTC!MEMX!SPO>
734 013040            Z      10$        ;IF EQUAL--CLEAR INT PENDING
(1) 013040 000247      MICPC=MICPC+1
(1) 013040 101651      <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
735 013042            ALWAYS IDLE
(1) 013042 000250      MICPC=MICPC+1
(1) 013042 100445      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
736 013044            10$: BRWRTE IMM,357 ;MASK TO CLEAR INT PENDING
(1) 013044 000251      MICPC=MICPC+1
(1) 013044 000757      <MOVE!WRTEBR!IMM!<357>>
737 013046            CLRIDL: SP      BR,AANDB,SP1
(1) 013046 000252      MICPC=MICPC+1
(1) 013046 063261      <MOVE!SPX!BR!AANDB!SP1>
738 013050            ALWAYS IDLE
(1) 013050 000253      MICPC=MICPC+1
(1) 013050 100445      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

```

740
 741 013052 000254
 (1) 013052 123440
 742 013054 000255
 (1) 013054 103525
 743 013056 000256
 (1) 013056 000500
 744 013060 000257
 (1) 013060 061262
 745 013062 000260
 (1) 013062 100674

```

.SBTTL OUTWAI--WAIT FOR RDYO TO GO AWAY
OUTWAI: SPBR IBUS, OCON, SPO ;READ OUTPUT CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!OCON!SPO>
        BR7 NIDLE6 ;RDYO SET --GET OUT
        MICPC=MICPC+1
        <JUMP!BR7CON!<NIDLE6-INIT&3000*4>!<NIDLE6-INIT&777/2>>
        BRWRT IMM,100 ;CLEAR CONTROL BITS
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<100>>
        OUT BR, OCON!AANDB
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<OCON!AANDB>>
        ALWAYS INS13
        MICPC=MICPC+1
        <JUMP!ALCOND!<INS13-INIT&3000*4>!<INS13-INIT&777/2>>

```


747			.SBTTL CTLSRV--CNTL I SERVICE	
748	013064		CTLSRV: SPBR IBUS,PORT4,SPO	;TO SPO
(1)		000261	MICPC=MICPC+1	
(1)	013064	123560	<MOVE!SPBRX!IBUS!PORT4!SPO>	
749	013066		BRSHFT	
(1)		000262	MICPC=MICPC+1	
(1)	013066	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
750	013070		BR1 HOSEL	;IF SET IS HALF DUPLEX
(1)		000263	MICPC=MICPC+1	
(1)	013070	102755	<JUMP!BR1CON!<HOSEL-INIT&3000*4>!<HOSEL-INIT&777/2>>	
751	013072		BRWRTE IMM,100	;MASK FOR DTR
(1)		000264	MICPC=MICPC+1	
(1)	013072	000500	<MOVE!WRTEBR!IMM!<100>>	
752	013074		OUTPUT BR,SELB!OMODEM	;TURN OFF HALF-DUPLEX
(1)		000265	MICPC=MICPC+1	
(1)	013074	062233	<MOVE!WROUT!BR!<SELB!OMODEM>>	
753	013076		INS11: BRWRTE DP<SELA!SPO>	;RESTORE THE CNTL WORD
(1)		000266	MICPC=MICPC+1	
(1)	013076	060600	<MOVE!WRTEBR!DP!<SELA!SPO>>	
754	013100		BR0 CBOOT	;IF SET IS BOOT
(1)		000267	MICPC=MICPC+1	
(1)	013100	102276	<JUMP!BR0CON!<CBOOT-INIT&3000*4>!<CBOOT-INIT&777/2>>	
755	013102		INS12: SP IBUS,INCON,SPO	;READ THE INPUT CONTROL CSR
(1)		000270	MICPC=MICPC+1	
(1)	013102	123000	<MOVE!SPX!IBUS!INCON!SPO>	
756	013104		BRWRTE IMM,100	;ZERO THE BR REGISTER EXCEPT INT ENABLE
(1)		000271	MICPC=MICPC+1	
(1)	013104	000500	<MOVE!WRTEBR!IMM!<100>>	
757	013106		OUT BR,<AANDB!OINCON>	;CLEAR IN CONTROL CSR
(1)		000272	MICPC=MICPC+1	
(1)	013106	061260	<MOVE!WROUTX!BR!<AANDB!OINCON>>	
758	013110		LDMA IMM,PRST	;ADDRESS PORT STATE
(1)		000273	MICPC=MICPC+1	
(1)	013110	010162	<MOVE!LDMAR!IMM!<PRST&377>>	
759	013112		INS13: PSTATE NIDLE2	
(1)	013112		MEM IMM,<<NIDLE2-INIT&777/2>>	
(2)		000274	MICPC=MICPC+1	
(2)	013112	002520	<MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>	
760	013114		ALWAYS IDLE	
(1)		000275	MICPC=MICPC+1	
(1)	013114	100445	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
761			.	
762	013116		CBOOT: BRWRTE IMM,200	;MASK FOR BOOT MODE
(1)		000276	MICPC=MICPC+1	
(1)	013116	000600	<MOVE!WRTEBR!IMM!<200>>	
763	013120		SP BR,AORB,SP1	;IN PORT STATUS WORD
(1)		000277	MICPC=MICPC+1	
(1)	013120	063301	<MOVE!SPX!BR!AORB!SP1>	
764	013122		BRWRTE IMM,204	;MASK FOR OK TO SEND AND LINE IDLE
(1)		000300	MICPC=MICPC+1	
(1)	013122	000604	<MOVE!WRTEBR!IMM!<204>>	
765	013124		SP BR,SELB,SP10	;IN LINE STATUS
(1)		000301	MICPC=MICPC+1	
(1)	013124	063230	<MOVE!SPX!BR!SELB!SP10>	
766	013126		ALWAYS INS12	
(1)		000302	MICPC=MICPC+1	

N06

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10:08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 6-19
CTLSRV--CNTL I SERVICE

PAGE: 0078

(1) 013126 100670

<JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>

768
769 013130 000303
(1) 013130 010072
770 013132 000304
(1) 013132 053220
771 013134 000305
(1) 013134 016401
772 013136 000306
(1) 013136 014406
773
774 013140 000307
(1) 013140 136500
775 013142 000310
(1) 013142 136520
776 013144 000311
(1) 013144 136560
777 013146 000312
(1) 013146 136540
778 013150 000313
(1) 013150 063000
779 013152 000314
(1) 013152 000553
780 013154 000315
(1) 013154 010072
781 013156 000316
(1) 013156 060360
782 013160 000317
(1) 013160 101724
783 013162 000320
(1) 013162 062600
784 013164 000321
(1) 013164 000402
785 013166 000322
(1) 013166 063310
786 013170 000323
(1) 013170 100670
787 013172 000324
(1) 013172 002473

```
.SBITL TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE
TBASRV: LDMA IMM,ETC ;GET POINTER TO END OF TMT CHAIN
        MICPC=MICPC+1
        <MOVE!LDMA!IMM!<ETC&377>>
        LDMA MEMX,<SELB!SPX!SPO> ;FIND THE LINK
        MICPC=MICPC+1
        <MOVE!LDMA!MEMX!<SELB!SPX!SPO>>
        MEMINC IMM,1 ;BUFFER ASSIGNED IN IN LINK FLAGS
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IMM!<1>>
        BRWRT <IMM!INCMAR>,6 ;POINT PAST NUMBER FIELD
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!INCMAR!<6>> ;SET BR FOR ADDITION TO SPO

        MEMINC IBUS,PORT1
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IBUS!<PORT1>>
        MEMINC IBUS,PORT2
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IBUS!<PORT2>>
        MEMINC IBUS,PORT4
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IBUS!<PORT4>>
        MEMINC IBUS,PORT3
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IBUS!<PORT3>>
        SP BR,ADD,SPO ;OFFSET TO NEXT TRANSMIT LINK
        MICPC=MICPC+1
        <MOVE!SPX!BR!ADD!SPO>
        BRWRT IMM,T ;LOAD BR WITH ADDRESS OF CHAIN END
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<T>>
        LDMA IMM,ETC
        MICPC=MICPC+1
        <MOVE!LDMA!IMM!<ETC&377>>
        CMP BR,SPO ;END OF CHAIN?
        MICPC=MICPC+1
        \SUBTC!BR!SPO>
        Z 20$ ;IF YES--BRANCH
        MICPC=MICPC+1
        <JUMP!ZCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
        MEM BR,SELA!SPO ;UPDATE THE END POINTER IN MEMORY
        MICPC=MICPC+1
        <MOVE!WRMEM!BR!<SELA!SPO>>
10$: BRWRT IMM,2 ;NUMBERED MSG PENDING MASK
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<2>>
        SP BR,AORB,SP10 ;UPDATE LINE STATUS
        MICPC=MICPC+1
        <MOVE!SPX!BR!AORB!SP10>
        ALWAYS INS12
        MICPC=MICPC+1
        <JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
20$: MEM IMM,TML1 ;WRAP IT AROUND
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<TML1>>
```

C07

JMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10:08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 6-21
TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE

PAGE: 0080

798 013174
(1) 000325
: 013174 100721

ALWAYS 10\$;CONTINUE PROCESSING
MICPC=MICPC+1
<JUMP!ALCOND!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>

790
791 013176 000326
(1) 013176 010023
792 013200 000327
(1) 013200 053220
793 013202 000330
(1) 013202 016401
794 013204 000331
(1) 013204 136500
795 013206 000332
(1) 013206 136520
796 013210 000333
(1) 013210 136560
797 013212 000334
(1) 013212 136540
798
799 013214 000335
(1) 013214 010023
800 013216 000336
(1) 013216 002425
801 013220 000337
(1) 013220 000463
802 013222 000340
(1) 013222 060360
803 013224 000341
(1) 013224 101670
804 013226 000342
(1) 013226 000405
805 013230 000343
(1) 013230 062400
806 013232 000344
(1) 013232 100670
807 013234 000345
(1) 013234 000717
808 013236 000346
(1) 013236 063670
809 013240 000347
(1) 013240 000400

```

.SBTTL RBASRV--RECEIVE BUFFER ADDRESS SERVICE
RBASRV: LDMA IMM,ERC ;ADDRESS END OF RECEIVE CHAIN
MICPC=MICPC+1
<MOVE!LDMA!IMM!<ERC&377>>
LDMA MEMX,<SELB!SPX!SPO> ;GET THE POINTER TO LINK
MICPC=MICPC+1
<MOVE!LDMA!MEMX!<SELB!SPX!SPO>>
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
<MOVE!LDMA!IMM!<ERC&377>>
MEM IMM,RCL1 ;ASSUME WRAP AROUND CASE
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<RCL1>>
BRWRT 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
<SUBTC!BR!SPO>
Z INS12 ;IF EQUAL BRANCH
MICPC=MICPC+1
<JUMP!ZCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
BRWRT IMM,5 ;CALCULATE ADDRESS OF NEXT LINK
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<5>>
MEM BR,ADD!SPO ;...
MICPC=MICPC+1
<MOVE!WRMEM!BR!<ADD!SPO>>
ALWAYS INS12 ;EXIT
MICPC=MICPC+1
<JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
RA1: BRWRT IMM,317 ;MASK TO CLEAR START MODE AND CLR ACTIVE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<317>>
SPBR BR,AANDB,SP10 ;CLEAR BIT IN LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPBRX!BR!AANDB!SP10>
RA3: BRWRT IMM,0 ;CLEAR BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<0>>

```

```

810 013242
(1) 000350
(1) 013242 063233
811 013244
(1) 000351
(1) 013244 000424
912 013246
(1) 000352
(1) 013246 104422
813
914 013250
(1) 000353
(1) 013250 000402
915 013252
(1) 000354
(1) 013252 114671
816

```

```

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

```

830 013254 000355
(1) 013254 000500
831 013256 000356
(1) 013256 063310
832 013260 000357
(1) 013260 100666
833
834 013262 000360
(1) 013262 000700
835 013264 000361
(1) 013264 123220
836 013266 000362
(1) 013266 061311
841 013270 000363
(1) 013270 100634
843
844 013272 000364
(1) 013272 002730
845
846 013274 000365
(1) 013274 000402
847 013276 000366
(1) 013276 061231
848 013300 000367
(1) 013300 120620
849 013302 000370
(1) 013302 102767
850 013304 000371
(1) 013304 154620
851 013306 000372
(1) 013306 120620
852 013310 000373
(1) 013310 103364
853 013312 000374
(1) 013312 114733
854
855 013314 000375
(1) 013314 120600
856 013316

```

MOSEL: BRWRT IMM,100 ;HD MASK TO BR
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<100>>
        SP BR,AORB,SP10 ;UPDATE PORT STATUS WORD
        MICPC=MICPC+1
        <MOVE!SPX!BR!AORB!SP10>
        ALWAYS INS11
        MICPC=MICPC+1
        <JUMP!ALCOND!<INS11-INIT&3000*4>!<INS11-INIT&777/2>>

PE1: BRWRT IMM,300 ;MASK FOR INTERUPT AND VECTOR THROUGH X04
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<300>>
      SP IBUS,UBBR,SPO ;READ BR CONTROL REG
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!UBBR!SPO>
      OUT BR,<AORB!OBR> ;INTERRUPT
      MICPC=MICPC+1
      <MOVE!WROUTX!BR!<AORB!OBR>>
      ALWAYS PINT2
      MICPC=MICPC+1
      <JUMP!ALCOND!<PINT2-INIT&3000*4>!<PINT2-INIT&777/2>>

HALTED: MEMADR EM6
        MICPC=MICPC+1
        <MOVE!WRMEM!<EM6-INIT&777/2>>

ACLOW: BRWRT IMM,2 ;FALL INTO ACLOW
        MICPC=MICPC+1 ;CAUSE AN AC LOW
        <MOVE!WRTEBR!IMM!<2>>
        OUT BR,<SELB!OBR>
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<SELB!OBR>>

SS: BRWRT IBUS,UBBR ;WAIT FOR IT TO COMPLETE
     MICPC=MICPC+1
     <MOVE!WRTEBR!IBUS!<UBBR>>
     BR1 SS
     MICPC=MICPC+1
     <JUMP!BR1CON!<SS-INIT&3000*4>!<SS-INIT&777/2>>
     .ALWAY MEMX,SELB,PAGE3
     MICPC=MICPC+1
     <JUMP!ALCOND!MEMX!SELB!PAGE3>

CKTIME: BRWRT IBUS,UBBR ;READ BR CONTROL REG
         MICPC=MICPC+1
         <MOVE!WRTEBR!IBUS!<UBBR>>
         BR4 HALTED
         MICPC=MICPC+1
         <JUMP!BR4CON!<HALTED-INIT&3000*4>!<HALTED-INIT&777/2>>
         ALWAYS EM1
         MICPC=MICPC+1
         <JUMP!ALCOND!<EM1-INIT&3000*4>!<EM1-INIT&777/2>>

IBU1: BRWRT IBUS,NPR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<NPR>>
      BRO IDLE
    
```

G07

JMC11 DDCMP PROTOCOL IMPLEMENTATION
JCCNEW.MAC 21-APR-77 10:08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 5-25
RBASRV--RECEIVE BUFFER ADDRESS SERVICE

PAGE: 0084

(1) 000376
(1) 013316 102045
957 013320
(1) 000377
(1) 013320 114760
959

MICPC=MICPC+1
<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2
ALWAYS EC2
MICPC=MICPC+1
<JUMP!ALCOND!<EC2-INIT&3000*4>!<EC2-INIT&777/2

860	013322	. =INIT+1000
961	000377	MICPC=377
962		.SBTTL RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER
963		:ENTERED FROM IDLE LOOP
964		:DETERMINES IF MESSAGE TYPE IS NUMBERED, UNNUMBERED OR BOOT
965		:SETS UP APPROPRIATE STATES FOR REST OF MESSAGE.
966	013322	RCVA: SP IBUS,RCVDAT,SPO ;READ RECEIVE CHARACTER TO SPC
(1)	000400	MICPC=MICPC+1
(1)	013322	<MOVE!SPX!IBUS!RCVDAT!SPO>
967	013324	BRWRT BR,SELA!SP1 ;FEAD PORT STATUS WORD
(1)	000401	MICPC=MICPC+1
(1)	013324	<MOVE!WRTBR!BR!<SELA!SP1>>
868	013326	BR0 S\$;IF INIT MODE---ONLY BOOT OK
(1)	000402	MICPC=MICPC+1
(1)	013326	<JUMP!BR0CON!<S\$-INIT&3000*4>!<S\$-INIT&777/2>>
969	013330	BR7 S\$;IF BOOT MODE---ONLY BOOT OK
(1)	000403	MICPC=MICPC+1
(1)	013330	<JUMP!BR7CON!<S\$-INIT&3000*4>!<S\$-INIT&777/2>>
870	013332	BRWRT IMM,201 ;SOH TO BR
(1)	000404	MICPC=MICPC+1
(1)	013332	<MOVE!WRTBR!IMM!<201>>
871	013334	CMP BR,SPO ;COMPARE BR TO SPC
(1)	000405	MICPC=MICPC+1
(1)	013334	<SUBTC!BR!SPO>
872	013336	Z RA1 ;IF EQUAL-IS NUMBERED MESSAGE
(1)	000406	MICPC=MICPC+1
(1)	013336	<JUMP!ZCOND!<RA1-INIT&3000*4>!<RA1-INIT&777/2>>
873	013340	BRWRT IMM,5 ;ENG TO BR
(1)	000407	MICPC=MICPC+1
(1)	013340	<MOVE!WRTBR!IMM!<5>>
874	013342	CMP BR,SPO ;COMPARE ENG TO SPC
(1)	000410	MICPC=MICPC+1
(1)	013342	<SUBTC!BR!SPO>
875	013344	Z RA2 ;IF EQUAL-IS UNNUMBERED MESSAGE
(1)	000411	MICPC=MICPC+1
(1)	013344	<JUMP!ZCOND!<RA2-INIT&3000*4>!<RA2-INIT&777/2>>
876	013346	BRWRT IMM,220 ;DLE TO BR
(1)	000412	MICPC=MICPC+1
(1)	013346	<MOVE!WRTBR!IMM!<220>>
877	013350	CMP BR,SPO ;COMPARE DLE TO SPC
(1)	000413	MICPC=MICPC+1
(1)	013350	<SUBTC!BR!SPO>
878	013352	Z BOOT ;IF EQUAL IS BOOT
(1)	000414	MICPC=MICPC+1
(1)	013352	<JUMP!ZCOND!<BOOT-INIT&3000*4>!<BOOT-INIT&777/2>>
879	013354	FLUSH: OUTPUT IMM,<200!ORCVCO> ;FLUSH THE INPUT SILO
(1)	000415	MICPC=MICPC+1
(1)	013354	<MOVE!WROUT!IMM!<200!ORCVCO>>
880	013356	BRWRT IMM,357 ;MASK TO CLEAR--CLEAR ACTIVE
(1)	000416	MICPC=MICPC+1
(1)	013356	<MOVE!WRTBR!IMM!<357>>
881	013360	SP BR,AANDB,SP10 ;IN LINE STATUS WORD
(1)	000417	MICPC=MICPC+1
(1)	013360	<MOVE!SPX!BR!AANDB!SP10>
882	013362	ALWAYS RM1 ;SET STATE TO RCVA AND RETURN TO IDLE
(1)	000420	MICPC=MICPC+1

```

(1) 013362 114662
883 013364 000421
(1) 013364 000700
888 013366 000422
(1) 013366 063223
899 013370 000423
(1) 013370 100445

```

```

RA2: <JUMP!ALCOND!<RM1-INIT&3000*4>!<RM1-INIT&777/2>>
      STATE RCVI :CHANGE RECEIVE STATE TO I
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVI-INIT&777/2>>
REXIT: SP BR SELB,SP3
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP3>
        ALWAYS IDLE
        MICPC=MICPC+1
        <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

```

892
893
894
895
896 013372
897 013372
(1) 000424
(1) 013372 023204
898 013374
(1) 000425
(1) 013374 010024
899 013376
(1) 000426
(1) 013376 053234
900
901 013400
(1) 000427
(1) 013400 054620
902 013402
(1) 000430
(1) 013402 106042
903 013404
(1) 000431
(1) 013404 050601
904 013406
(1) 000432
(1) 013406 107440
905 013410
(1) 000433
(1) 013410 010153
906 013412
(1) 000434
(1) 013412 016402
907 013414
(1) 000435
(1) 013414 002710
908 013416
(1) 000436
(1) 013416 010012
909 013420
(1) 000437
(1) 013420 104557
910 013422
(1) 000440
(1) 013422 000404
911 013424
(1) 000441
(1) 013424 063301
912 013426
(1) 000442
(1) 013426 000462
913 013430
(1) 000443
(1) 013430 063223
914 013432
(1) 000444

```

;SMTL RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD
;ENTERED FROM IDLE LOOP
;STORES COUNT FIELD AND SETS UP RCVC AS NEXT STATE
;
RCVB:
SP      IBUS,RCVDAT,SP4      ;READ CHARACTER TO SP4
MICPC=MICPC+1
<MOVE!SPX!IBUS!RCVDAT!SP4>
LDMA   IMM,LRC              ;LOAD ADDRESS OF START OF RECV CHAIN
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<LRC&377>>
SP      MEMX!LDMAR,SELB,SP14 ;COPY POINTER TO SP14
MICPC=MICPC+1
<MOVE!SPX!MEMX!LDMAR!SELB!SP14>
;AND LOAD MAR WITH ADDRESS OF CURRENT BA
;READ FLAGS BYTE
BRWRT  MEMX,INCMAR!SELB
MICPC=MICPC+1
<MOVE!WRTEBR!MEMX!<INCMAR!SELB>>
;RCV BUFFER ASSIGNED---CONTINUE
BR0    RB1
MICPC=MICPC+1
<JUMP!BR0CON!<RB1-INIT&3000*4>!<RB1-INIT&777/2>>
BRWRT  BR,SELA!SP1         ;READ STATUS BYTE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
;MAINT MODE
BR7    RB3
MICPC=MICPC+1
<JUMP!BR7CON!<RB3-INIT&3000*4>!<RB3-INIT&777/2>>
LDMA   IMM,T               ;ERROR--LOAD TYPE FIELD ADDRESS IN MAR
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<T&377>>
;LOAD NAK TYPE
MEMINC IMM,2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<2>>
;LOAD SUB-TYPE NO BUFFERS
MEM    IMM,310
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<310>>
LDMA   IMM,NTLS
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<NTLS&377>>
;BRANCH TO SEND NAK ROUTINE
ALWAYS RNS
MICPC=MICPC+1
<JUMP!ALCOND!<RNS-INIT&3000*4>!<RNS-INIT&777/2>>
;MASK FOR NO BLFFER AVAILABLE
BR3:   BRWRT  IMM,4
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<4>>
;SET THE FLAG
SP      BR,AORB,SP1
MICPC=MICPC+1
<MOVE!SPX!BR!AORB!SP1>
RB1:   STATE  RCVC
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVC-INIT&777/2>>
RB0:   SP      BR,SEL3,SP3
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>
OUTPUT <MEMX!INCMAR>,<SELB!OBA1> ;OUTPUT LOW ORDER BYTE OF ADDRESS
MICPC=MICPC+1

```

```

(1) 013432 056226 <MOVE!WROUT!MEMX!INCMAR!<SELB!OBA1>>
915 013434 000445 OUTPUT MEMX!INCMAR,<SELB!OBA2> ;OUTPUT HIGH BYTE OF ADDRESS
(1) 013434 056227 MICPC=MICPC+1
(1) 013436 000446 <MOVE!WROUT!MEMX!INCMAR!<SELB!OBA2>>
916 013436 123220 SP IBUS,UBBR,SPO ;READ THE BUS REQ REGISTER
(1) 013436 000446 MICPC=MICPC+1
(1) 013436 123220 <MOVE!SPX!IBUS!UBBR!SPO>
917 013440 000447 BRWRITE IMM,101 ;MASK OFF ALL BUT NXM AND VEC4 BITS
(1) 013440 000501 MICPC=MICPC+1
918 013442 000450 <MOVE!WRTEBR!IMM!<101>>
(1) 013442 063260 SP BR,AANDB,SPO ;AND SAVE IN SPO
919 013444 000451 MICPC=MICPC+1
(1) 013444 003305 <MOVE!SPX!IMM!300!SP5> ;MASK TO ISOLATE EX. MEM BITS
920 321 ;NOTE THIS REALLY WRITES A 305 BUT THE
921 ;5 GETS SHIFTED OUT
922 013446 000452 BRWRITE MEMX,AANDB!SP5 ;MASK ALL BUT EX. MEM BITS
(1) 013446 040665 MICPC=MICPC+1
923 013450 000453 <MOVE!WRTEBR!MEMX!<AANDB!SP5>> ;SHIFT THEM INTO THE CORRECT POSITION
(1) 013450 001620 BRSHFT
924 013452 000454 MICPC=MICPC+1
(1) 013452 001620 <MOVE!SHFTBR!WRTEBR!SELB>
925 013454 000455 BRSHFT
(1) 013454 001620 MICPC=MICPC+1
926 013456 000456 <MOVE!SHFTBR!WRTEBR!SELB>
(1) 013456 001620 BRSHFT
927 013460 000457 OUT BR,AORB!OBR ;WRITE EX MEM BITS OUT
(1) 013460 061311 MICPC=MICPC+1
928 013462 000460 <MOVE!WROUTX!BR!<AORB!OBR>>
(1) 013462 100445 ALWAYS IDLE
929 013464 000461 MICPC=MICPC+1
(1) 013464 100454 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
RB2: ALWAYS I2
MICPC=MICPC+1
<JUMP!ALCOND!<I2-INIT&3000*4>!<I2-INIT&777/2>>

```

```

931 .SBTTL RCVC--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD, SELECT AND FINA
932 ;ENTERED FROM IDLE LOOP
933 ;INTERPRETS SELECT AND FINAL
934 ;CHECKS FOR COUNT TOO LARGE
935 .
936 RCVC: ALWAYS SELQSY ;"CALL" SELECT/QSYNC SUBROUTINE
(1) MICPC=MICPC+1
(1) 013466 114470 <JUMP!ALCOND!<SELQSY-INIT&3000*4>!<SELQSY-INIT&777/2>>
937 013470 LDMA BR,<SELA!SP14> ;LOAD ADDRESS OF CURRENT COUNT
(1) MICPC=MICPC+1
(1) 013470 070214 <MOVE!LDMA!BR!<SELA!SP14>>
939 013472 BRWRT BR!INCMAR,SELA!SP1 ;READ STATUS BYTE
(1) MICPC=MICPC+1
(1) 013472 074601 <MOVE!WRTBR!BR!INCMAR!<SELA!SP1>>
939 013474 BRWRT SHFTBR!INCMAR,SELB ;SHIFT IT RIGHT
(1) MICPC=MICPC+1
(1) 013474 015620 <MOVE!WRTBR!SHFTBR!INCMAR!<SELB>>
940 013476 BR1 RCS ;NO BUFFER ASSIGNED IN MAINT MODE
(1) MICPC=MICPC+1
(1) 013476 106475 <JUMP!BR1CON!<RCS-INIT&3000*4>!<RCS-INIT&777/2>>
941 013500 BRWRT IMM!INCMAR,77 ;MASK FOR COUNT BITS
(1) MICPC=MICPC+1
(1) 013500 014477 <MOVE!WRTBR!IMM!INCMAR!<77>>
942 013502 SP MEMX!INCMAR,SELB,SPO ;COPY HIGH BYTE OF COUNT TO SPO
(1) MICPC=MICPC+1
(1) 013502 057220 <MOVE!SPX!MEMX!INCMAR!SELB!SPO>
943 013504 BRWRT BR,AANDB!SPO ;MASK TO BR
(1) MICPC=MICPC+1
(1) 013504 060660 <MOVE!WRTBR!BR!<AANDB!SPO>>
944 013506 CMP BR,SPS ;COMPARE HIGH ORDER BITS OF COUNT
(1) MICPC=MICPC+1
(1) 013506 060365 <SUBTC!BR!SPS>
945 013510 C RCFATL ;IF CARRY--TOO BIG ERROR
(1) MICPC=MICPC+1
(1) 013510 115116 <JUMP!CCOND!<RCFATL-INIT&3000*4>!<RCFATL-INIT&777/2>>
946 013512 Z RLOW ;IF EQUAL COMPARE LOW ORDER BITS OF COUNT
(1) MICPC=MICPC+1
(1) 013512 115513 <JUMP!ZCOND!<RLOW-INIT&3000*4>!<RLOW-INIT&777/2>>
947 013514 RC5: STATE RCVD ;SET NEXT STATE TO D
(1) MICPC=MICPC+1
(1) 013514 000475 <MOVE!WRTBR!IMM!<RCVD-INIT&777/2>>
948 013516 ALWAYS REXIT
(1) MICPC=MICPC+1
(1) 013516 104422 <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>

```

950
 951
 952 013520 000477
 (1) 013520 063164
 953 013522 000500
 (1) 013522 105102
 954 013524 000501
 (1) 013524 063165
 955 013526 000502
 (1) 013526 000523
 956 013530 000503
 (1) 013530 063223
 957 013532 000504
 (1) 013532 023600
 958 013534 000505
 (1) 013534 060757
 959 013536 000506
 (1) 013536 107510
 960 013540 000507
 (1) 013540 100445
 961 013542 000510
 (1) 013542 060601
 962 013544 000511
 (1) 013544 103445
 963 013546 000512
 (1) 013546 060610
 964 013550 000513
 (1) 013550 001620
 965 013552 000514
 (1) 013552 103045
 966 013554 000515
 (1) 013554 010155
 967 013556 000516
 (1) 013556 062600
 968 013560 000517
 (1) 013560 010403
 969
 970 013562 000520
 (1)

```
.SBTTL RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES
RCVD:  SP      BR,DECA,SP4          ;DECREMENTLOW BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP4>
      C      RD3          ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<RD3-INIT&3000*4>!<RD3-INIT&777/2>>
      SP      BR,DECA,SP5          ;OVERFLOW - DECREMENT HIGH BYTE
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP5>
RD3:   STATE  RCVE
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVE-INIT&777/2>>
RD2:   SP      BR,SELB,SP3          ;SAVE THE STATE
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP3>
      SPBR    IBUS,RCVDAT,SPO          ;INPUT THE CHARACTER
      MICPC=MICPC+1
      <MOVE!SPBRX!IBUS!RCVDAT!SPO>
      BRWRTE BR,SUB!SP17          ;COMPARE NEW R TO LAST R
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SUB!SP17>>
      BR7     10$          ;IF NEW IS GREATER---PROCESS
      MICPC=MICPC+1
      <JUMP!BR7CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
      ALWAYS  IDLE
      MICPC=MICPC+1
      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
10$:   BRWRTE BR,SELA!SP1          ;READ STATUS BYTE
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
      BR7     IDLE          ;MAINT. MODE - GET OUT
      MICPC=MICPC+1
      <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      BRWRTE BR,SELA!SP10
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP10>>
      BRSHFT
      MICPC=MICPC+1
      <MOVE!SHFTBR!WRTEBR!SELB>
      BR4     IDLE
      MICPC=MICPC+1
      <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      LDMA    IMM,ISP17          ;ADDRESS LAST ACKED IMAGE
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<ISP17&J77>>
      MEM     BR,SELA!SPO          ;COPY THE CHAR
      MICPC=MICPC+1
      <MOVE!WRMEM!BR!<SELA!SPO>>
RD5:   BRWRTE IMM,REPST!LDMAR          ;SET UP COUNT FOR TIMER
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<REPST!LDMAR>>
      MEM     IMM,1          ;***DEPENDENT ON REPST = 2
      MICPC=MICPC+1          ;RESET REP THRESHOLD
```

NO7

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10:08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 6-32
RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES

PAGE: 0091

(1) 013562 002401
971 013564
(1) 000521
(1) 013564 063235
972 013566
(1) 000522
(1) 013566 100445

<MOVE!WRMEM!IMM!<1>>
SP BR,SELB,SP15 ;RESET THE COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP15>
ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

```

974
975
976 013570 000523
(1) 013570 060601
977 013572 000524
(1) 013572 107713
978 013574 000525
(1) 013574 020600
979 013576 000526
(1) 013576 060371
980 013600 000527
(1) 013600 105532
981 013602 000530
(1) 013602 063173
982 013604 000531
(1) 013604 104533
983 013606 000532
(1) 013606 063071
984 013610 000533
(1) 013610 000535
985 013612 000534
(1) 013612 104422

```

```

.SBTTL RCVE--ROUTINE TO HANDLE N FIELD OF NUMBERED MESSAGE
RCVE: BRWRITE BR, SELA!SP1 ;READ THE STATUS BYTE
      MICPC=MICPC+1
      <MOVE!WRITEBR!BR!<SELA!SP1>>
      BR7 RCVQ
      MICPC=MICPC+1
      <JUMP!BR7CON!<RCVQ-INIT&3000*4>!<RCVQ-INIT&777/2>>
      BRWRITE IBUS, RCVDAT ;INPUT THE CHARACTER
      MICPC=MICPC+1
      <MOVE!WRITEBR!IBUS!<RCVDAT>>
      CMP BR, SP11
      MICPC=MICPC+1
      <SUBTC!BR!SP11>
      Z SS
      MICPC=MICPC+1
      <JUMP!ZCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
      SP BR, DECA, SP13 ;FORCE MSG TYPE TO -1
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP13>
      ALWAYS RE2
      MICPC=MICPC+1
      <JUMP!ALCOND!<RE2-INIT&3000*4>!<RE2-INIT&777/2>>
SS: SP BR, INCA, SP11 ;UPDATE R FIELD
      MICPC=MICPC+1
      <MOVE!SPX!BR!INCA!SP11>
RE2: STATE RCVF ;NEXT RECEIVE STATE IS F
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<RCVF-INIT&777/2>>
      ALWAYS REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>

```



```

987
988 013614 000535
(1) 013614 000540
989 013616 000536
(1) 013616 020200
990 013620 000537
(1) 013620 104422
991
992
993
994 013622 000540
(1) 013622 000542
995 013624 000541
(1) 013624 104536
  
```

```

.SBTTL RCVF--ROUTINE TO IGNORE ADDRESS
RCVF: STATE RCVG
      MICPC=MICPC+1
      <MOVE!WATEBR!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!WATEBR!IMM!<RCVH-INIT&777/2>>
      ALWAYS RCVF1
      MICPC=MICPC+1
      JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>
  
```

997
998
999 013626
1000 013626
(1) 000542
(1) 013626 023200
1001 013630
(1) 000543
(1) 013630 020640
1002 013632
(1) 000544
(1) 013632 116167
1003 013634
(1) 000545
(1) 013634 060601
1004 013636
(1) 000546
(1) 013636 107751
1005 013640
(1) 000547
(1) 013640 060610
1006 013642
(1) 000550
(1) 013642 001620
1007 013644
(1) 000551
(1) 013644 117315
1008 013646
(1) 000552
(1) 013646 010153
1009 013650
(1) 000553
(1) 013650 016402
1010 013652
(1) 000554
(1) 013652 015701
1011 013654
(1) 000555
(1) 013654 062617
1012 013656
(1) 000556
(1) 013656 010013
1013 013660
(1) 000557
(1) 013660 043220
1014 013662
(1) 000560
(1) 013662 062460
1015 013664
(1) 000561
(1) 013664 010001
1016 013666
(1) 000562
(1) 013666 040620
1017 013670
(1) 000563

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

(1) 013670 061620
1018 013672 000564
(1) 013672 062620
1019 013674 000565
(1) 013674 116264
1020 013676 000566
(1) 013676 114712
1021 013700 000567
(1) 013700 060573
1022 013702 000570
(1) 013702 115505
1023 013704 000571
(1) 013704 000400
(1) 013704 000572
(1) 013706 063223
1024 013710 000573
(1) 013710 060610
1025 013712 000574
(1) 013712 117940
1026 013714 000575
(1) 013714 001620
1027 013716 000576
(1) 013716 107204
1028 013720 000577
(1) 013720 010163
1029 013722 000600
(1) 013722 054373
1030 013724 000601
(1) 013724 115404
1031 013726 000602
(1) 013726 054373
1032 013730 000603
(1) 013730 115443
1033 013732 000604
(1) 013732 010165
1034 013734 000605
(1) 013734 054373
1035 013736 000606

```

<MOVE!SHFTBR!SELB!BR>
MEM BR,SELB ;UPDATE IT
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELB>>
BR0 NTHRES ;BRANCH IF THRESHOLD EXCEEDED
MICPC=MICPC+1
<JUMP!BROCON!<NTHRES-INIT&3000*4>!<NTHRES-INIT&777/2>>
ALWAYS SNAK
MICPC=MICPC+1
<JUMP!ALCOND!<SNAK-INIT&3000*4>!<SNAK-INIT&777/2>>
RH3: BRWRT DP <DECA!SP13> ;LOAD TYPE RECEIVED--DECREMENTING
MICPC=MICPC+1
<MOVE!WRTEBR!DP!<DECA!SP13>>
Z RH1 ;IF ALUOUT IS ALL ONES IS NUMBERED MSG
MICPC=MICPC+1
<JUMP!ZCOND!<RH1-INIT&3000*4>!<RH1-INIT&777/2>>
RSTATE RCVA
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVA-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>
BRWRT DP <SELA!SP10> ;LOAD LINE STATUS WORD IN BR
MICPC=MICPC+1
<MOVE!WRTEBR!DP!<SELA!SP10>>
BR4 FLUSH1
MICPC=MICPC+1
<JUMP!BR4CON!<FLUSH1-INIT&3000*4>!<FLUSH1-INIT&777/2>>
CG1: BRSHFT ;SHIFT RIGHT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR4 105
MICPC=MICPC+1
<JUMP!BR4CON!<105-INIT&3000*4>!<105-INIT&777/2>>
LDMA IMM, TYPTAB ;ADDRESS TYPE TABLE
MICPC=MICPC+1
<MOVE!LDMA!IMM!<TYPTAB&377>>
CMP <MEMX!INCMAR>,SP13
MICPC=MICPC+1
<SUBTC!MEMX!INCMAR!SP13>
Z REP
MICPC=MICPC+1
<JUMP!ZCOND!<REP-INIT&3000*4>!<REP-INIT&777/2>>
105: CMP <MEMX!INCMAR>,SP13
MICPC=MICPC+1
<SUBTC!MEMX!INCMAR!SP13>
Z NAK
MICPC=MICPC+1
<JUMP!ZCOND!<NAK-INIT&3000*4>!<NAK-INIT&777/2>>
LDMA IMM, TYPSTT ;SET POINTER TO START TYPE
MICPC=MICPC+1
<MOVE!LDMA!IMM!<TYPSTT&377>>
CMP <MEMX!INCMAR>,SP13
MICPC=MICPC+1
<SUBTC!MEMX!INCMAR!SP13>
Z START
MICPC=MICPC+1

```

(1)	013736	115413	<JUMP!ZCOND!<START-INIT&3000*4>!<START-INIT&777/2>> ;STACK TYPE
1036			
1037	013740		CMP <MEMX!INCMAR>,SP13
(1)		000607	MICPC=MICPC+1
(1)	013740	054373	<SUBTC!MEMX!INCMAR!SP13>
1039	013742		Z STACK
(1)		000610	MICPC=MICPC+1
(1)	013742	115425	<JUMP!ZCOND!<STACK-INIT&3000*4>!<STACK-INIT&777/2>>
1039	013744		CMP <MEMX!INCMAR>,SP13 ;ACK TYPE
(1)		000611	MICPC=MICPC+1
(1)	013744	054373	<SUBTC!MEMX!INCMAR!SP13>
1040	013746		Z ACK
(1)		000612	MICPC=MICPC+1
(1)	013746	105620	<JUMP!ZCOND!<ACK-INIT&3000*4>!<ACK-INIT&777/2>>
1041	013750		ALWAYS IDLE ;OTHERWISE IGNORE--MUST BE OBS MSG
(1)		000613	MICPC=MICPC+1
(1)	013750	100445	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1043	013752		RCVCK: SPBR IBUS,RCVCON,SPO ;READ RCVR CONTROL CSR
(1)		000614	MICPC=MICPC+1
(1)	013752	023640	<MOVE!SPBRX!IBUS!RCVCON!SPO>
1044	013754		BRWRT BR,ADD!SPO ;SHIFT LEFT
(1)		000615	MICPC=MICPC+1
(1)	013754	060400	<MOVE!WRTEBR!BR!<ADD!SPO>>
1045	013756		BR7 I1
(1)		000616	MICPC=MICPC+1
(1)	013756	103452	<JUMP!BR7CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
1046	013760		ALWAYS TAI
(1)		000617	MICPC=MICPC+1
(1)	013760	110405	<JUMP!ALCOND!<TAI-INIT&3000*4>!<TAI-INIT&777/2>>
1047	013762		ACK: BRWRT BR,AA!SP10 ;READ LINE STATUS-SHIFTING LEFT
(1)		000620	MICPC=MICPC+1
(1)	013762	060530	<MOVE!WRTEBR!BR!<AA!SP10>>
1048	013764		BR4 SS ;IF START RECD -- CLEAR START MODE
(1)		000621	MICPC=MICPC+1
(1)	013764	107223	<JUMP!BR4CON!<SS-INIT&3000*4>!<SS-INIT&777/2>>
1049	013766		ALWAYS IDLE
(1)		000622	MICPC=MICPC+1
(1)	013766	100445	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1050	013770		SS: BRWRT IMM,327 ;CLEAR START MODE
(1)		000623	MICPC=MICPC+1
(1)	013770	000727	<MOVE!WRTEBR!IMM!<327>>
1051	013772		SP BR,AANDB,SP10 ;IN LINE STATUS
(1)		000624	MICPC=MICPC+1
(1)	013772	063270	<MOVE!SPX!BR!AANDB!SP10>
1052	013774		ALWAYS RDS
(1)		000625	MICPC=MICPC+1
(1)	013774	104517	<JUMP!ALCOND!<RDS-INIT&3000*4>!<RDS-INIT&777/2>>

```

1055
1056
1057 013776 000626
      (1) 013776 123600
1058 014000 000627
      (1) 014000 102045
1059 014002 000630
      (1) 014002 000600
1060 014004 000631
      (1) 014004 061310
1061 014006 000632
      (1) 014006 000670
1062 014010 000633
      (1) 014010 104637
1063
1072 014012 000634
      (1) 014012 123600
1073 014014 000635
      (1) 014014 102045
1074 014016 000636
      (1) 014016 000645
1075 014020 000637
      (1) 014020 063223
1076 014022 000640
      (1) 014022 022203
1077 014024
1079 014024 000641
      (1) 014024 000421
1080 014026 000642
      (1) 014026 123200
1081 014030 000643
      (1) 014030 061310
1082 014032 000644
      (1) 014032 100445

```

```

;*****TIME CRITICAL CODE-- CHANGE WITH GREAT CARE*****
.SBTTL RCVK01--ROUTINE TO HANDLE FIRST BYTE ODD RECEIVE
RCVK01: SPBR IBUS,NPR,SPO ;READ NPR REGISTER
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!NPR!SPO>
        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
RCVK0: SPBR IBUS,NPR,SPO ;IS AN NPR GOING
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!NPR!SPO>
        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>>
RKE:
RKE: BRWRT IMM,21 ;SET OUT NPR (C1) AND NPR REG
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<21>>
        SP IBUS,NPR,SPO ;READ NPR REGISTER
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!NPR!SPO>
RKE: OUT BR <AORB!ONPR> ;WRITE NPR REGISTER
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<AORB!ONPR>>
        ALWAYS IDLE
        MICPC=MICPC+1
        <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

```

1093
1094 014034 000645
(1) 014034 120600
1099 014036 000646
(1) 014036 102045
1101 014040 000647
(1) 014040 023140
1102 014042 000650
(1) 014042 062066
1103 014044 000651
(1) 014044 063164
1104 014046 000652
(1) 014046 105255
1105 014050 000653
(1) 014050 063165
1106 014052 000654
(1) 014052 105721
1107 014054 000655
(1) 014054 023140
1108 014056 000656
(1) 014056 062066
1109 014060 000657
(1) 014060 115030
1110 014062 000660
(1) 014062 023200
1111 014064 000661
(1) 014064 062202
1112 014066 000662
(1) 014066 063164
1113 014070 000663
(1) 014070 105266
1114 014072 000664
(1) 014072 063165
1115 014074 000665
(1) 014074 111765
1125 014076 000666
(1) 014076 000634
1126 014100

```

.SBTTL RCVKE--HANDLE EVEN BYTES
RCVKE: BRWRT IBUS,NPR ;READ NPR CONTROL REGISTER
        MICPC=MICPC+1
        <MOVE!WRTEBR!IBUS!<NPR>>
        BRD IDLE
        MICPC=MICPC+1
        <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
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 105 ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<105-INIT&3000*4>!<105-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>>
105: 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
      <MOVE!WROUT!BR!<SELA!OUTDA1>>
      SP BR,DECA,SP4 ;DECREMENT THE COUNT OF BYTES
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP4>
      C RK6 ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<RK6-INIT&3000*4>!<RK6-INIT&777/2>>
      SP BR,DECA,SP5 ;DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP5>
      Z RL4 ;BYTE COUNT ZERO
      MICPC=MICPC+1
      <JUMP!ZCOND!<RL4-INIT&3000*4>!<RL4-INIT&777/2>>
RK6: STATE RCVKO
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVKO-INIT&777/2>>
      ALWAYS REXIT

```

```

(1) 000667
(1) 014100 104422
1128
1129 014102
(1) 000670
(1) 014102 123200
1130 014104
(1) 000671
(1) 014104 000577
1131 014106
(1) 000672
(1) 014106 061270
1132 014110
(1) 000673
(1) 014110 104651
1133
1134
1135 014112
(1) 000674
(1) 014112 023200
1136 014114
(1) 000675
(1) 014114 060601
1137 014116
(1) 000676
(1) 014116 117600
1138 014120
(1) 000677
(1) 014120 104661

```

```

MICPC=MICPC+1
<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
RKE1: SP IBUS,NPR,SPO ;READ NPR REGISTER
MICPC=MICPC+1
<MOVE!SPX!IBUS!NPR!SPO>
BRWRT IMM,177 ;MASK FOR ALL BUT CO
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<177>>
OUT BR,<AANDB!ONPR> ;TURN OFF ALL BUT CO
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AANDB!ONPR>>
ALWAYS RK50
MICPC=MICPC+1
<JUMP!ALCOND!<RK50-INIT&3000*4>!<RK50-INIT&777/2>>
:*****END OF TIME CRITICAL PATH*****
RCVKEO: SP IBUS,RCV DAT,SPO ;READ CHARACTER AND SAVE IN SPO
MICPC=MICPC+1
<MOVE!SPX!IBUS!RCV DAT!SPO>
BRWRT BR,SELA!SP1 ;READ STATUS BYTE
MICPC=MICPC+1
<MOVE!WRTBR!BR!<SELA!SP1>>
BR7 PASWRD ;MAINT MODE - SEE IF RLD MESSAGE
MICPC=MICPC+1
<JUMP!BR7CON!<PASWRD-INIT&3000*4>!<PASWRD-INIT&777/2>>
ALWAYS RK3 ;OTHERWISE PROCESS NORMALLY
MICPC=MICPC+1
<JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>

```

JOB

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10:08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 6-41
RCVI--STORE UNNUMBERED MESSAGE TYPE

PAGE: 0100

1140		
1141	014122	000700
(1)		023213
(1)	014122	
1142	014124	000701
(1)		000703
(1)	014124	
1143	014126	000702
(1)		134422
(1)	014126	
1144		

```

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

```


K08

DMC11 JDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10:08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 6-42
RCVJ--ROUTINE TO HANDLE SUBTYPE FIELD,SELECT AND FINAL

PAGE: 0101

1146
1147 014130
(1) 000703
(1) 014130 114470
1148 014132
(1) 000704
(1) 014132 000706
1149 014134
(1) 000705
(1) 014134 104422

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>>

1151
1152
1153
1154 014136 000706
(1) 014136 000403
1155 014140 000707
(1) 014140 060353
1156 014142 000710
(1) 014142 000713
1157
1158 014144 000711
(1) 014144 105136
1159 014146 000712
(1) 014146 104503

```

.SBTTL RCVR--UNNUMBERED MESSAGE RESPONSE FIELD
;ENTERED FROM IDLE LOOP
RCVR: BRWRT IMM,3 ;REP MESSAGE TYPE TO BR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<3>>
      NOP BR,SUB,SP13 ;IS TYPE ACK OR NAK
      MICPC=MICPC+1
      <BR!SUB!SP13>
      STATE RCVQ ;NEXT STATE IS RCVQ
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVQ-INIT&777/2>>
      C RCVF1 ;***NOTE THIS INSTR DOES NOT CLOCK "C"
      MICPC=MICPC+1 ;IF NOT IGNORE
      <JUMP!CCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>
      ALWAYS RD2 ;DO RANGE CHECKS
      MICPC=MICPC+1
      <JUMP!ALCOND!<RD2-INIT&3000*4>!<RD2-INIT&777/2>>

```

MOS

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10.08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 6-44
RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD

PAGE: 0103

1161
1162
1163
1164 014150
(1) 000713
(1) 014150 000535
1165 014152
(1) 000714
(1) 014152 104536

.SBTTL RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD
;ENTER FROM IDLE
RCVQ: STATE RCVF ;NEXT STATE IS ADDRESS
MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<RCVF-INIT&777/2>>
ALWAYS RCVF1
MICPC=MICPC+1
<JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>

1167			.SBTTL RCVL--PROCESS CRC3	
1168			;ENTERED FROM IDLE LOOP	
1169	014154		RCVL: SPBR IBUS,NPR,SPO	;READ NPR CONTROL
(1)		000715	MICPC=MICPC+1	
(1)	014154	123600	<MOVE!SPBRX!IBUS!NPR!SPO>	
1174	014156		BRO IDLE	
(1)		000716	MICPC=MICPC+1	
(1)	014156	102045	<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
1176	014160		RL2: BRWRT IMM,177	;MASK TO TURN OFF CO
(1)		000717	MICPC=MICPC+1	
(1)	014160	000577	<MOVE!WRTBR!IMM!<177>>	
1177	014162		OUT BR,AANDB!ONPR	
(1)		000720	MICPC=MICPC+1	
(1)	014162	061270	<MOVE!WROUTX!BR!<AANDB!ONPR>>	
1178	014164		RL3: SPBR IBUS,IOBA1,SPO	;READ LOW ORDER BYTE OF ADDRESS
(1)		000721	MICPC=MICPC+1	
(1)	014164	023540	<MOVE!SPBRX!IBUS!IOBA1!SPO>	
1179	014166		OUTPUT BR,INCA!OBA1	;INCREMENT THE LOW ORDER BYTE
(1)		000722	MICPC=MICPC+1	
(1)	014166	062066	<MOVE!WROUT!BR!<INCA!OBA1>>	
1180	014170		SP IBUS,IOBA2,SPO	;READ HIGH BYTE
(1)		000723	MICPC=MICPC+1	
(1)	014170	023160	<MOVE!SPX!IBUS!IOBA2!SPO>	
1181	014172		OUTPUT BR,AC!OBA2	;ADD CARRY TO HIGH BYTE
(1)		000724	MICPC=MICPC+1	
(1)	014172	062107	<MOVE!WROUT!BR!<AC!OBA2>>	
1182			:	
1183	014174		STATE RCVL	
(1)		000725	MICPC=MICPC+1	
(1)	014174	000727	<MOVE!WRTBR!IMM!<RCVM-INIT&777/2>>	
1184	014176		ALWAYS RCVF1	
(1)		000726	MICPC=MICPC+1	
(1)	014176	104536	<JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>	
1185			:	
1190			:	

```

1192 .SBTTL RCVM--PROCESS CRC4--END OF DATA MESSAGE
1193 ;ENTERED FROM IDLE LOOP
1194 ;IF CRC CORRECT -- QUEUE INTERRUPT AND UPDATE RESPONSE
1195 ;
1196 ;IF CRC WRONG SEND NAK
1197 RCVM: BRWRT IBUS,UBBR ;READ UNIBUS BR REGISTER
(1) MICPC=MICPC+1
(1) 014200 120620 <MOVE!WRTEBR!IBUS!<UBBR>>
1198 014202 BRO NXMERR ;NON-EXISTANT MEMORY
(1) MICPC=MICPC+1
(1) 014202 116063 <JUMP!BROCON!<NXMERR-INIT&3000*4>!<NXMERR-INIT&777/2>>
1199 014204 SP IBUS,RCVDAT,SPD ;READ CRC CHARACTER
(1) MICPC=MICPC+1
(1) 014204 023200 <MOVE!SPX!IBUS!RCVDAT!SPD>
1200 014206 BRWRT IBUS,RCVCON ;READ RECEIVER CONTROL REGISTER
(1) MICPC=MICPC+1
(1) 014206 020640 <MOVE!WRTEBR!IBUS!<RCVCON>>
1201 014210 BRO RCVM1 ;IF CRC GOOD -- PROCESS
(1) MICPC=MICPC+1
(1) 014210 116216 <JUMP!BROCON!<RCVM1-INIT&3000*4>!<RCVM1-INIT&777/2>>
1202 014212 BRWRT BR,SELA!SP1 ;READ STATUS BYTE
(1) MICPC=MICPC+1
(1) 014212 060601 <MOVE!WRTEBR!BR!<SELA!SP1>>
1203 014214 BR7 RHX ;CRC ERROR IN BOOT MODE - FLUSH
(1) MICPC=MICPC+1
(1) 014214 107751 <JUMP!BR7CON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
1204 014216 LDMA IMM,T ;ELSE SEND NAK --DATA ERROR
(1) MICPC=MICPC+1
(1) 014216 010153 <MOVE!LDMAR!IMM!<T&377>>
1205 014220 MEMINC IMM,2 ;NAK TYPE
(1) MICPC=MICPC+1
(1) 014220 016402 <MOVE!WRMEM!INCMAR!IMM!<2>>
1206 014222 MEMINC IMM,302 ;DATA ERROR SJBTYP
(1) MICPC=MICPC+1
(1) 014222 016702 <MOVE!WRMEM!INCMAR!IMM!<302>>
1207 014224 LDMA IMM,NDATS
(1) MICPC=MICPC+1
(1) 014224 010014 <MOVE!LDMAR!IMM!<NDATS&377>>
1208 014226 ALWAYS RHS ;SEND NAK
(1) MICPC=MICPC+1
(1) 014226 104557 <JUMP!ALCOND!<RHS-INIT&3000*4>!<RHS-INIT&777/2>>
1209
1210 RCVM0: LDMA IMM,<<RTHRS+3>> ;POINT TO ERROR WORD
(1) MICPC=MICPC+1
(1) 014230 010177 <MOVE!LDMAR!IMM!<<RTHRS+3>&377>>
1211 014232 BRWRT IMM,10 ;MAINT MESSAGE ERROR
(1) MICPC=MICPC+1
(1) 014232 000410 <MOVE!WRTEBR!IMM!<10>>
1212 014234 ALWAYS RCEXY ;GIVE FATAL ERROR
(1) MICPC=MICPC+1
(1) 014234 114525 <JUMP!ALCOND!<RCEXY-INIT&3000*4>!<RCEXY-INIT&777/2>>

```

1214			.SBTTL EM2--PROCESS RLD MESSAGE
1215			;ENTERED FROM IDLE LOOP
1216			;IF RLD PASSWORD CHECKS TRIGGER THE BOOT ROM
1217			
1218	014236		EM2: BRWRITE IBUS,RCVDAT ;READ THE CHAR
(1)		000746	MICPC=MICPC+1
(1)	014236	020600	<MOVE!WRITEBR!IBUS!<RCVDAT>>
1219	014240		CMP BR,SP16 ;IS IT A MATCH
(1)		000747	MICPC=MICPC+1
(1)	014240	060376	<SUBTC!BR!SP16>
1220	014242		Z EM3
(1)		000750	MICPC=MICPC+1
(1)	014242	105757	<JUMP!ZCOND!<EM3-INIT&3000*4>!<EM3-INIT&777/2>>
1221			;FALL INTO RMX
1222	014244		RHX: BRWRITE BR,AA,SP1 ;READ STATUS BYTE SHIFTED LEFT
(1)		000751	MICPC=MICPC+1
(1)	014244	060521	<MOVE!WRITEBR!BR!<AA!SP1>>
1223	014246		BR4 10\$;DLE RECEIVED IN NORMAL MODE
(1)		000752	MICPC=MICPC+1
(1)	014246	107354	<JUMP!BR4CON!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>
1224	014250		ALWAYS FLUSH ;ALREADY IN MAINT MODE
(1)		000753	MICPC=MICPC+1
(1)	014250	104415	<JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
1225	014252		10\$: BRWRITE IMM,163 ;MASK TO CLEAR ALL MAINT RELATED BITS
(1)		000754	MICPC=MICPC+1
(1)	014252	000563	<MOVE!WRITEBR!IMM!<163>>
1226	014254		SP BR,AAND8,SP1 ;CLEAR THEM
(1)		000755	MICPC=MICPC+1
(1)	014254	063261	<MOVE!SPX!BR!AAND8!SP1>
1227	014256		ALWAYS FLUSH
(1)		000756	MICPC=MICPC+1
(1)	014256	104415	<JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
1228			
1229	014260		EM3: SP BR,DECA,SP4 ;DECREMENT CHARACTER COUNT BY ONE
(1)		000757	MICPC=MICPC+1
(1)	014260	063164	<MOVE!SPX!BR!DECA!SP4>
1230	014262		Z EMTRIG ;TRIGGER AC LOW
(1)		000760	MICPC=MICPC+1
(1)	014262	115720	<JUMP!ZCOND!<EMTRIG-INIT&3000*4>!<EMTRIG-INIT&777/2>>
1231	014264		ALWAYS IDLE
(1)		000761	MICPC=MICPC+1
(1)	014264	100445	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

1233
 1234 014266 000762
 (1) 014266 060601
 1235 014270 000763
 (1) 014270 103747
 1236 014272 000764
 (1) 014272 000610
 1237 014274 000765
 (1) 014274 063301
 1238 014276 000766
 (1) 014276 100747
 1239 014300 000767
 (1) 014300 000404
 1240 014302 000770
 (1) 014302 123200
 1241 014304 000771
 (1) 014304 061010
 1242 014306 000772
 (1) 014306 110753
 1243 014310 000773
 (1) 014310 000404
 1244 014312 000774
 (1) 014312 12322C
 1245 014314 000775
 (1) 014314 061011
 1246 014316 000776
 (1) 014316 114767
 1247 014320 000777
 (1) 014320 000000

```

BOOT:  BRWRT  BR SELA!SP1           ;SEE IF IN MAINT. MODE
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<SELA!SP1>>
        BR7   RA3                   ;BRANCH IF SO AND TREAT DLE LIKE NUM. MSG.
        MICPC=MICPC+1
        <JUMP!BR7CON!<RA3-INIT&3000*4>!<RA3-INIT&777/2>>
        BRWRT IMM,210               ;MASK TO SET MAINT MODE AND DLE RECV'D
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<210>>
        SP    BR,AORB,SP1           ;SET THE BITS
        MICPC=MICPC+1
        <MOVE!SPX!BR!AORB!SP1>
        ALWAYS RA3                  ;TREAT LIKE NUMBERED MESSAGE
        MICPC=MICPC+1
        <JUMP!ALCOND!<RA3-INIT&3000*4>!<RA3-INIT&777/2>>
RESEXT: BRWRT  IMM,4                ;ADD TO MXT BITS
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<4>>
        SP    IBUS,NPR,SPO
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!NPR!SPO>
        OUT   BR,ADD!ONPR           ;DO IT
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<ADD!ONPR>>
        ALWAYS RES1
        MICPC=MICPC+1
        <JUMP!ALCOND!<RES1-INIT&3000*4>!<RES1-INIT&777/2>>
TABMXT: BRWRT  IMM,4                ;INCREMENT MXT
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<4>>
        SP    IBUS,UBR,SPO         ;READ BR CONTROL
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!UBR!SPO>
        OUT   BR,ADD!OBR
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<ADD!OBR>>
        ALWAYS ECX
        MICPC=MICPC+1
        <JUMP!ALCOND!<ECX-INIT&3000*4>!<ECX-INIT&777/2>>
        $ZERO
        MICPC=MICPC+1
        000000
  
```

1249		014322
1250		000777
1251		
1252		
1253	014322	
(1)		001000
(1)	014322	020620
1254	014324	
(1)		001001
(1)	014324	173202
1255	014326	
(1)		001002
(1)	014326	100452

```

.=INIT+2000
MICPC=777
.SBTTL TMTDA--TRANSMITTER DISPATCH ROUTINE
TMTDA: BRWRT IBUS,TMTCON ;READ TRANSMITTER CONTROL REGISTER
MICPC=MICPC+1
<MOVE!WRTBR!IBUS!<TMTCON>>
.BR4 DP SELA,<2!PAGE2> ;IF READY PROCEED
MICPC=MICPC+1
<JUMP! BR4CON!DP!SELA!2!PAGE2>
ALWAYS I1 ;ELSE IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>

```



```

1257
1258
1259 014330 001003
      (1) 014330 060530
1260 014332 001004
      (1) 014332 107614
1261 014334 001005
      (1) 014334 060610
1262 014336 001006
      (1) 014336 112031
1263 014340 001007
      (1) 014340 001620
1264 014342 001010
      (1) 014342 103052
1265 014344 001011
      (1) 014344 112431
1266
1267 014346 001012
      (1) 014346 000453
      (1) 014350 001013
      (1) 014350 063222
1268 014352 001014
      (1) 014352 060601
1269 014354 001015
      (1) 014354 113427
1270 014356 001016
      (1) 014356 060610
1271 014360 001017
      (1) 014360 112023
1272 014362 001020
      (1) 014362 000601
1273 014364 001021
      (1) 014364 062230
1274 014366 001022
      (1) 014366 100452
1275 014370 001023
      (1) 014370 000600
      (1) 014372 001024
      (1) 014372 063222
1276 014374

```

```

.SBTTL TMTA--FIRST CHARACTER OF HEADER
TMTA: BRWRT BR,AA!SP10 ;SHIFT LEFT
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<AA!SP10>>
      BR7 RCVCK
      MICPC=MICPC+1
      <JUMP!BR7CON!<RCVCK-INIT&3000*4>!<RCVCK-INIT&777/2>>
TMTA: BRWRT BR,SELA!SP10 ;REREAD STATUS
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP10>>
      BRO NUMSYN ;IF UNNUMBPENDING -- SEND IT
      MICPC=MICPC+1
      <JUMP!BROCON!<NUMSYN-INIT&3000*4>!<NUMSYN-INIT&777/2>>
      BRSHFT
      MICPC=MICPC+1
      <MOVE!SHFTBR!WRTEBR!SELB>
      BR4 I1 ;IF START MODE--EXIT
      MICPC=MICPC+1
      <JUMP!BR4CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
      BR1 NUMSYN ;IF LINE HAS GONE IDLE SEND SYN
      MICPC=MICPC+1
      <JUMP!BR1CON!<NUMSYN-INIT&3000*4>!<NUMSYN-INIT&777/2>>
      ;ELSE--START TO SEND MESSAGE
TMTXT: TSTATE TMTB
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTB-INIT&777/2>>
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      BRWRT BR,SELA!SP1 ;ARE WE IN BOOT MODE
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
      BR7 TMTBT ;IF SO SEND DLE
      MICPC=MICPC+1
      <JUMP!BR7CON!<TMTBT-INIT&3000*4>!<TMTBT-INIT&777/2>>
      BRWRT BR,<SELA!SP10> ;UNNUMB MESSAGE?
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP10>>
      BRO TMTUN ;IF SO --BRANCH
      MICPC=MICPC+1
      <JUMP!BROCON!<TMTUN-INIT&3000*4>!<TMTUN-INIT&777/2>>
      BRWRT IMM,201 ;ELSE STORE SOH
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<201>>
TMTAS: OUTPUT BR,<SELB!TMTDAT> ;IN TMT SILO
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!TMTDAT>>
      ALWAYS I1
      MICPC=MICPC+1
      <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
TMTJN: TSTATE TMTI
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTI-INIT&777/2>>
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      BRWRT IMM,5 ;END TO BR

```

```

(1) 001025
(1) 014374 003405
1277 014376
(1) 001026
(1) 014376 110421
1278 014400
(1) 001027
(1) 014400 000620
1279 014402
(1) 001030
(1) 014402 110421
1280
1281 014404
(1) 001031
(1) 014404 060610
1282 014406
(1) 001032
(1) 014406 113434
1293 014410
(1) 001033
(1) 014410 100452
1284 014412
(1) 001034
(1) 014412 020660
1295 014414
(1) 001035
(1) 014414 001620
1286 014416
(1) 001036
(1) 014416 103052
1287 014420
(1) 001037
(1) 014420 000773
1288 014422
(1) 001040
(1) 014422 063270
1289 014424
(1) 001041
(1) 014424 000445
(1) 001042
(1) 014426 063222
1294 014430
(1) 001043
(1) 014430 000410
1296 014432
(1) 001044
(1) 014432 063226
1297 014434
(1) 001045
(1) 014434 063166
1298 014436
(1) 001046
(1) 014436 111412
1299 014440
(1) 001047
(1) 014440 000401

```

```

MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<5>>
ALWAYS TMTA5
MICPC=MICPC+1
<JUMP!ALCOND!<TMTA5-INIT&3000*4>!<TMTA5-INIT&777/2>>
TMTBT: BRWRT IMM,220 ;WRITE A DLE TO BR
MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<220>>
ALWAYS TMTA5 ;SEND IT
MICPC=MICPC+1
<JUMP!ALCOND!<TMTA5-INIT&3000*4>!<TMTA5-INIT&777/2>>
NUMSYN: BRWRT BR,<SELA!SP10> ;READ LINE STATUS WORD
MICPC=MICPC+1
<MOVE!WRITEBR!BR!<SELA!SP10>>
BR7 S5 ;IF OK TO SEND--PROCEED
MICPC=MICPC+1
<JUMP!BR7CON!<S5-INIT&3000*4>!<S5-INIT&777/2>>
ALWAYS I1 ;ELSE--IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
S5: BRWRT IBUS,MODEM ;ARE WE STILL SENDING?
MICPC=MICPC+1
<MOVE!WRITEBR!IBUS!<MODEM>>
BRSHFT
MICPC=MICPC+1
<MOVE!SHFTBR!WRITEBR!SELB>
BR4 I1 ;RTS SET? IF SO WE ARE--STALL
MICPC=MICPC+1
<JUMP!BR4CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
BRWRT IMM,373 ;MASK TO TURN OFFLINE IDLE
MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<373>>
SP BR,AANDB,SP10 ;IN LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SP10>
TSTATE TMTA1
MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<TMTA1-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
BRWRT IMM,10
MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<10>>
SP BR,SELB,SP6 ;STORE IN SP6
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP6>
TMTA1: SP BR,DECA,SP6 ;DECREMENT SYN COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP6>
Z TMTXT
MICPC=MICPC+1
<JUMP!ZCOND!<TMTXT-INIT&3000*4>!<TMTXT-INIT&777/2>>
BRWRT IMM,1 ;MASK FOR SOM
MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<1>>

```

1300	014442	
(1)		001050
(1)	014442	062231
1301	014444	
(1)		001051
(1)	014444	000626
1302	014446	
(1)		001052
(1)	014446	110421

```

OUTPUT BR,SELB!OTMTCO           ;WRITE IT TO TMTA CONTROL
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELB!OTMTCO>>
BRWRT IMM,226                   ;SYNC CHAR
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<226>>
ALWAYS TMTAS
MICPC=MICPC+1
<JUMP!ALCOND!<TMTAS-INIT$3000+4 !TMTAS-INIT&777/2>>

```

1304			.SBTTL TMTB--OUTPUT FIRST CHAR OF COUNT
1305			
1306	014450	001053	TMTB: SPBR IBUS,NPR,SPO ;READ BR CONTROL REG
(1)		12360C	MICPC=MICPC+1
(1)	014450		<MOVE!SPBRX!IBUS!NPR!SPO>
1307	014452		BRJ IDLE ;NPR GOING--GET OUT
(1)		001054	MICPC=MICPC+1
(1)	014452	102045	<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1308	014454		LDMA IMM,LTC ;GET POINTER TO NEXT TMT LINK
(1)		001055	MICPC=MICPC+1
(1)	014454	010071	<MOVE!LDMAR!IMM!<LTC&377>>
1309	014456		LDMA MEMX,SELB!SPX!SP16 ;POINT TO THE LINK
(1)		001056	MICPC=MICPC+1
(1)	014456	053236	<MOVE!LDMAR!MEMX!<SELB!SPX!SP16>>
1310	014460		MEMINC IMM,3 ;WRITE MSG TMTED TO FLAGS
(1)		001057	MICPC=MICPC+1
(1)	014460	016403	<MOVE!WRMEM!INCMAR!IMM!<3>>
1311	014462		MEMINC DP,SELA!SP12 ;PICK UP MSGNO
(1)		001060	MICPC=MICPC+1
(1)	014462	076612	<MOVE!WRMEM!INCMAR!DP!<SELA!SP12>>
1312	014464		STATE TMT ;ADDRESS TMT STATE
(1)		001061	MICPC=MICPC+1
(1)	014464	000500	<MOVE!WRTEBR!IMM!<TMT-INIT&777/2>>
1313	014466		TBO: SP BR,SELB,SP2 ;UPDATE IT
(1)		001062	MICPC=MICPC+1
(1)	014466	063222	<MOVE!SPX!BR!SELB!SP2>
1314	014470		OUTPUT <MEMX!INCMAR>,SELB!IBA1 ;WRITE LOW BYTE OF ADDRESS
(1)		001063	MICPC=MICPC+1
(1)	014470	056224	<MOVE!WROUT!MEMX!INCMAR!<SELB!IBA1>>
1315	014472		OUTPUT <MEMX!INCMAR>,SELB!IBA2 ;WRITE HIGH BYTE OF ADDRESS
(1)		001064	MICPC=MICPC+1
(1)	014472	056225	<MOVE!WROUT!MEMX!INCMAR!<SELB!IBA2>>
1316	014474		SP MEMX,SELB,SP7 ;HIGH BYTE OF COUNT TO SP7
(1)		001065	MICPC=MICPC+1
(1)	014474	043227	<MOVE!SPX!MEMX!SELB!SP7>
1317			;WAIT TO MASK OFF MEM EXT. BITS
1318	014476		SP IMM,300,SPO ;MASK FOR MXT
(1)		001066	MICPC=MICPC+1
(1)	014476	003300	<MOVE!SPX!IMM!300!SPO>
1319	014500		BRWRT MEMX!INCMAR,AANDB!SPO ;TURN OFF CC2
(1)		001067	MICPC=MICPC+1
(1)	014500	054660	<MOVE!WRTEBR!MEMX!INCMAR!<AANDB!SPO>>
1320	014502		BRSHFT ;SHIFT BITS INTO CORRECT POSITION
(1)		001070	MICPC=MICPC+1
(1)	014502	001620	<MOVE!SHFTBR!WRTEBR!SELB>
1321	014504		BRSHFT
(1)		001071	MICPC=MICPC+1
(1)	014504	001620	<MOVE!SHFTBR!WRTEBR!SELB>
1322	014506		BRSHFT
(1)		001072	MICPC=MICPC+1
(1)	014506	001620	<MOVE!SHFTBR!WRTEBR!SELB>
1323	014510		BRSHFT
(1)		001073	MICPC=MICPC+1
(1)	014510	001620	<MOVE!SHFTBR!WRTEBR!SELB>
1324	014512		OUT BR,SELB,ONPR
(1)		001074	MICPC=MICPC+1

(1)	014512	061230	<MOVE!WROUTX!BR!<SELB!ONPR>>	
1325	014514		SPBR MEMX!INCMAR,SELB,SP6	;LOWBYTE OF COUNT TO SP6
(1)		001075	MICPC=MICPC+1	
(1)	014514	057626	<MOVE!SPBRX!MEMX!INCMAR!SELB!SP6>	
1326	014516		OUTPUT BR,SELB:TMTDAT	;WRITE IT TO TMTR SILO
(1)		001076	MICPC=MICPC+1	
(1)	014516	062230	<MOVE!WROUT!BR!<SELB!TMTDAT>>	
1327	014520		ALWAYS IDLE	
(1)		001077	MICPC=MICPC+1	
(1)	014520	100445	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
1328			:	

```

1330
1331
1332 014522 001100
(1) 014522 000477
1333 014524 001101
(1) 014524 063667
1334 014526 001102
(1) 014526 062230
1335 014530 001103
(1) 014530 010071
1336 014532 001104
(1) 014532 000406
1337 014534 001105
(1) 014534 062416
1338 014536 001106
(1) 014536 000545
1339 014540 001107
(1) 014540 060376
1340 014542 001110
(1) 014542 111513
1341 014544 001111
(1) 014544 000515
1342 014546 001112
(1) 014546 100451
1343 014550 001113
(1) 014550 002473
1344 014552 001114
(1) 014552 110511
1345

```

```

.SBTTL TMTC--OUTPUT SECOND CHAR OF COUNT
TMTC: BRWRT IMM,77 ;MASK TO CLEAR MXT BITS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<77>>
SPBR BR,AANDB,SP7 ;CLEAR THEM
MICPC=MICPC+1
<MOVE!SPBRX!BR!AANDB!SP7>
OUTPUT DP,<SELB!TMTDAT> ;WRITE TO TMT SILO
MICPC=MICPC+1
<MOVE!WROUT!DP!<SELB!TMTDAT>>
LDMA IMM,LTC ;POINT TO TMT BUFFER
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<LTC&377>>
BRWRT IMM,6 ;OFFSET TO NEXT LINK
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<6>>
MEM BR,ADD!SP16 ;UPDATE THE POINTER
MICPC=MICPC+1
<MOVE!WRMEM!BR!<ADD!SP16>>
BRWRT IMM,TML8 ;GET WRAPAROUND ADDRESS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TML8>>
CMP BR,SP16 ;WRAPAORUND
MICPC=MICPC+1
<SUBTC!BR!SP16>
Z 10$
MICPC=MICPC+1
<JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
5$: STATE TMTD
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTD-INIT&777/2>>
ALWAYS XEXIT
MICPC=MICPC+1
<JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
10$: MEM IMM,TML1 ;GO BACK TO FIRST LINK
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<TML1>
ALWAYS 5$
MICPC=MICPC+1
<JUMP!ALCOND!<5$-INIT&3000*4>!<5$-INIT&777/2>>
:

```

1347
 1348 014554
 (1) 001115
 (1) 014554 000523
 1349 014556
 (1) 001116
 (1) 014556 063166
 1350 014560
 (1) 001117
 (1) 014560 111121
 1351 014562
 (1) 001120
 (1) 014562 063167
 1352 014564
 (1) 001121
 (1) 014564 010171

```

TMTD: .SBTTL TMTD--RESPONSE FIELD-NUMBERED MESSAGE
STATE TMTD
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTD-INIT&777/2>>
SP BR,DECA,SP6 ;ADJUST COUNT FOR TWO'S COMPLEMENT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP6>
C TD2 ;NO OVERFLOW
MICPC=MICPC+1
<JUMP!CCOND!<TD2-INIT&3000*4>!<TD2-INIT&777/2>>
SP BR,DECA,SP7 ;DECREMENT HIGH BYTE OF COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP7>
TD2: LDMA IMM,ISP11 ;RESP FIELD ADDR TO MAR
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<ISP11&377>>

```

M09

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10:08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 11
TMTD--RESPONSE FIELD-NUMBERED MESSAGE

PAGE: 0116

1363 014566
(1) 001122
(1) 014566 110606

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

1367
 1368 014570
 1369 014570
 (1) 001123
 (1) 014570 123600
 1370 014572
 (1) 001124
 (1) 014572 102052
 1371 014574
 (1) 001125
 (1) 014574 060612
 1372 014576
 (1) 001126
 (1) 014576 062230
 1373 014600
 (1) 001127
 (1) 014600 000531
 1374 014602
 (1) 001130
 (1) 014602 110573

.SBTTL TMTE--NUMBER FIELD--NUMBERED MESSAGE
 TMTE: SPBR IBUS,NPR,SPO ;READ NPR CONTROL REGISTER
 MICPC=MICPC+1
 <MOVE!SPBRX!IBUS!NPR!SPO>
 BRO I1 ;BUSY - GET OUT
 MICPC=MICPC+1
 <JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
 BRWRTÉ BR SELA!SP12
 MICPC=MICPC+1
 <MOVE!WRTEBR!BR!<SELA!SP12>>
 OUTPUT BR <SELB!TMTDAT> ;WRITE IT TO THE SILO
 MICPC=MICPC+1
 <MOVE!WROUT!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>>

B10

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10:09

MACY11 30(1046) 11-JUL-77 12:18 PAGE 12-1
TMTF--NUMBERED MSG ADDRESS FIELD

PAGE: 0118

1376		
1377		
1378	014604	001131
(1)		000535
(1)	014604	
1379	014606	001132
(1)		063222
(1)	014606	
1380	014610	001133
(1)		000401
(1)	014610	
1386	014612	001134
(1)		110421
(1)	014612	

```

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

```

```

1389
1390 014614 001135
(1) 014614 000402
1391 014616 001136
(1) 014616 062231
1392 014620 001137
(1) 014620 062230
1393 014622 001140
(1) 014622 020500
1394 014624 001141
(1) 014624 112155
1395 014626 001142
(1) 014626 000544
1396 014630 001143
(1) 014630 100451

```

```

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

```

```

1398
1399
1400
1401 014632 001144
(1) 014632 123600
1405 014634 001145
(1) 014634 102052
1406 014636 001146
(1) 014636 022010
1407 014640 001147
(1) 014640 023100
1408 014642 001150
(1) 014642 062064
1409 014644 001151
(1) 014644 063166
1410 014646 001152
(1) 014646 111155
1411 014650 001153
(1) 014650 063167
1412 014652 001154
(1) 014652 115402
1413 014654
1418 014654 001155
(1) 014654 000557
1419 014656 001156
(1) 014656 100451
1420 014660
1422 014660 001157
(1) 014660 123600
1423 014662 001160
(1) 014662 102052
1425 014664 001161
(1) 014664 022030
1426 014666 001162
(1) 014666 023100
1427 014670 001163
(1) 014670 062064
1428 014672 001164
(1) 014672 111372

```

```

;*****TIME CRITICAL PATH--MODIFY WITH GREAT CARE
.SBTTL TMTH--ROUTINE TO OUTPUT DATA CHARACTERS
TMTH: SPBR IBUS,NPR,SPO ;READ NPR CONTROL
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!NPR!SPO>
BRO I1 ;IF NPR IN PROGRESS --BRANCH
MICPC=MICPC+1
<JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
SS: OUTPUT IBUS,<INDAT1!TMTDAT> ;WRITE THE EVEN CHAR TO TMT SILO
MICPC=MICPC+1
<MOVE!WROUT!IBUS!<INDAT1!TMTDAT>>
SP IBUS,IIBA1,SPO ;READ LOW BYTE OF BA TO SF
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>>
T46:
TMTF1: STATE TMTHO
MICPC=MICPC+1
<MOVE!WTEBR!IMM!<TMTHO-INIT&777/2>>
ALWAYS XEXIT
MICPC=MICPC+1
<JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
TMTHO: SPBR IBUS,NPR,SPO ;NPR BUSY
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!NPR!SPO>
BRO I1
MICPC=MICPC+1
<JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
TH9: OUTPUT IBUS,<INDAT2!TMTDAT> ;ODD CHAR TO SILO
MICPC=MICPC+1
<MOVE!WROUT!IBUS!<INDAT2!TMTDAT>>
SP IBUS,IIBA1,SPO ;READ LOW BYTE TO BA
MICPC=MICPC+1
<MOVE!SPX!IBUS!IIBA1!SPO>
OUTPUT BR,<INCA!IBA1> ;OUTPUT THE INCREMENTED BA
MICPC=MICPC+1
<MOVE!WROUT!BR!<INCA!IBA1>>
C HOINCH
MICPC=MICPC+1
<JUMP!CCOND!<HOINCH-INIT&3000*4>!<HOINCH-INIT&777/2>>

```

E10

```

1429 014674 001165
(1) 014674 063166
1430 014676 001166
(1) 014676 111171
1431 014700 001167
(1) 014700 063167
1432 014702 001170
(1) 014702 115402
1433 014704 001171
(1) 014704 123600
1437 014706 001172
(1) 014706 000544
1438 014710 001173
(1) 014710 063222
1439 014712 001174
(1) 014712 000557
1440 014714 001175
(1) 014714 063260
1441 014716 001176
(1) 014716 000401
1449 014720 001177
(1) 014720 104643
1451
1452

```

```

TH8: SP BR,DECA,SP6 ;DECREMENT CHARACTERCOUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP6>
C TH7 ;NO OVERFLOW
MICPC=MICPC+1
<JUMP!CCOND!<TH7-INIT&3000*4>!<TH7-INIT&777/2>>
SP BR,DECA,SP7 ;DECREMENT HIGH BYTE OF COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP7>
Z HEH1 ;BYTE COUNT ZERO
MICPC=MICPC+1
<JUMP!ZCOND!<HEH1-INIT&3000*4>!<HEH1-INIT&777/2>>
TH7: SPBR IBUS,NPR,SPO ;READ NPR REGISTER
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!NPR!SPO>
STATE TMTM
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<TMTM-INIT&777/2>>
TH3: SP BR,SELB,SP2 ;SAVE TSTATE
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
TH3X: BRWRT IMM,157 ;CLEAR CO AND CI
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<157>>
SP BR,AANDB,SPO ;CLEAR THE BITS
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SPO>
BRWRT IMM,1 ;WRITE NPR BITS TO BR
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<1>>
ALWAYS RK7
MICPC=MICPC+1
<JUMP!ALCOND!<RK7-INIT&3000*4>!<RK7-INIT&777/2>>
;*****END TIME CRITICAL PATH*****
:

```

F10

```

1454
1455 014722 001200
      (1) 014722 010153
1456 014724 001201
      (1) 014724 043226
1457 014726 001202
      (1) 014726 000604
1458 014730 001203
      (1) 014730 110606
1459
1460
1461 014732 001204
      (1) 014732 010154
1462 014734 001205
      (1) 014734 000610
1463 014736 001206
      (1) 014736 042230
1464 014740 001207
      (1) 014740 100451
1465

```

```

      .SBTTL TMTI--SEND UNNUMBERED TYPE FIELD
TMTI:  LDMA IMM T           ;ADDRESS OF TYPE FIELD TO MAR
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<T&377>>
      SP MEMX SELB,SP6      ;COPY IT TO SP6
      MICPC=MICPC+1
      <MOVE!SPX!MEMX!SELB!SP6>
      STATE TMTJ
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTJ-INIT&777/2>>
      ALWAYS TJI
      MICPC=MICPC+1
      <JUMP!ALCOND!<TJI-INIT&3000*4>!<TJI-INIT&777/2>>
      ;
      .SBTTL TMTJ--SEND SUB-TYPE FIELD
TMTJ:  LDMA IMM ST          ;ADDRESS OF SUB-TYPE FIELD TO MAR
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<ST&377>>
      STATE TMTK
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTK-INIT&777/2>>
TJI:   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>>
      ;

```

1467
1468
1469 014742 001210
(1) 014742 000403
1470 014744 001211
(1) 014744 060346
1471 014746 001212
(1) 014746 000616
(1) 001213
(1) 014750 063222
1472 014752 001214
(1) 014752 111223
1473 014754 001215
(1) 014754 110521
1474
1475
1476 014756 001216
(1) 014756 000627
(1) 001217
(1) 014760 063222
1477 014762 001220
(1) 014762 000403
1478 014764 001221
(1) 014764 060366
1479 014766 001222
(1) 014766 111625
1480 014770 001223
(1) 014770 000400
1481 014772 001224
(1) 014772 110421
1482
1483 014774 001225
(1) 014774 060572
1489 014776 001226
(1) 014776 110421
1490

```

.SBTTL TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG)
TMTK: BRWRT IMM,3 ;WRITE A 3 TO BR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<3>>
      NOP BR,SUB,SP6 ;IF TYPE LESS THAN 3
      MICPC=MICPC+1
      <BR!SUB!SP6>
      TSTATE TMTL
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTL-INIT&777/2>>
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      C TMTLO
      MICPC=MICPC+1
      <JUMP!CCOND!<TMTLO-INIT&3000*4>!<TMTLO-INIT&777/2>>
      ALWAYS TD2
      MICPC=MICPC+1
      <JUMP!ALCOND!<TD2-INIT&3000*4>!<TD2-INIT&777/2>>
      ;
.SBTTL TMTL--UNNUMB MSG NUMBER FIELD
TMTL: TSTATE TMTM
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTM-INIT&777/2>>
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      BRWRT IMM,3
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<3>>
      CMP BR,SP6 ;IS MESSAGE REP
      MICPC=MICPC+1
      <SUBTC!BR!SP6>
      Z TMTL1 ;YES
      MICPC=MICPC+1
      <JUMP!ZCOND!<TMTL1-INIT&3000*4>!<TMTL1-INIT&777/2>>
TMTLO: BRWRT IMM,0 ;ADDRESS CONTNAT OF ZERO
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<0>>
      ALWAYS TMTAS
      MICPC=MICPC+1
      <JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>
      ;
TMTL1: BRWRT BR,DECA!SP12 ;WRITE A RESPONSE
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<DECA!SP12>>
      ALWAYS TMTAS
      MICPC=MICPC+1
      <JUMP!ALCOND!<TMTAS-INIT&3000*4>!<TMTAS-INIT&777/2>>
      ;

```

```

1493
1494 015000 001227
      (1) 015000 000631
1495 015002 001230
      (1) 015002 110532
1496 015004 001231
      (1) 015004 000402
1497 015006 001232
      (1) 015006 062231
1498 015010 001233
      (1) 015010 062230
1499 015012 001234
      (1) 015012 000404
1500 015014 001235
      (1) 015014 063710
1501 015016 001236
      (1) 015016 060530
1502 015020 001237
      (1) 015020 113643
1503 015022 001240
      (1) 015022 000776
1504 015024 001241
      (1) 015024 063270
1505 015026 001242
      (1) 015026 110734
1506 015030 001243
      (1) 015030 000576
1507 015032 001244
      (1) 015032 110641

```

```

.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: BRWRTÉ IMM,2 ;END OF MESSAGE TO BR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<2>>
      OUTPUT BR <SELB!OTMTCO>
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!OTMTCO>> ;OUTPUT A GARBAGE CHARACTER
      OUTPUT BR <SELB!TMTDAT>
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!TMTDAT>> ;SET UP LINE HAS GONE IDLE MASK
      BRWRTÉ IMM,4
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<4>>
      SPBR BR AORB,SP10 ;UPDATE LINE STATUS WORD
      MICPC=MICPC+1
      <MOVE!SPBRX!BR!AORB!SP10>
      BRWRTÉ BR AA!SP10 ;SHIFT STATUS LEFT
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<AA!SP10>>
      BR7 10$ ;IF HDX SET---BRANCH TO CLEAR OK TO SEND
      MICPC=MICPC+1
      <JUMP!BR7CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
      BRWRTÉ 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>>
10$: BRWRTÉ 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>>

```



```

1509
1510
1511
1512 015034 001245
      (1) 015034 123220
1513 015036 001246
      (1) 015036 000577
1514
1515
1516
1517
1518 015040 001247
      (1) 015040 061271
1519 015042 001250
      (1) 015042 060601
1520 015044 001251
      (1) 015044 103445
1521 015046 001252
      (1) 015046 063175
1522 015050 001253
      (1) 015050 111662
1523 015052 001254
      (1) 015052 060610
1524 015054 001255
      (1) 015054 116737
1525 015056 001256
      (1) 015056 116337
1526 015060 001257
      (1) 015060 001620
1527 015062 001260
      (1) 015062 103045
1528 015064 001261
      (1) 015064 110727
1529 015066 001262
1530 015066 000402
      (1) 015066 000402
1531 015070 001263
      (1) 015070 063235
1532 015072 001264
      (1) 015072 060610

```

```

.SBTTL TIMSRV--TIMEOUT ROUTINE--SENDS REP
;
;ENABLE LSB
TIMSRV. SP IBUS,UBBR,SPO ;REAC UNIBUS BR REGISTER
MICPC=MICPC+1
<MOVE!SPX!IBUS!UBBR!SPO>
BRWRT IMM,177 ;MASK OFF BR REQ
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<177>>
;RESET TIMER---SLICK MOVE
;SINCE TIMER IS RESET BY WRITING
;A 1 AND THE EXPIRATION LOOKS
;LIKE 1--VOILA
;AND THE BIT ON
OUT BR,<AANDB!OBR>
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AANDB!OBR>>
BRWRT BR,SELA!SP1 ;READ STATUS BYTE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
BR7 IDLE ;IF IN MAINT. MODE DISABLE TIMER
MICPC=MICPC+1
<JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
SP BR,DECA,SP15 ;DECREMENT THE COUNTER
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP15>
Z 20$ ;IF ALL ONES HAS EXPIRED
MICPC=MICPC+1
<JUMP!ZCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
10$: BRWRT BR,SELA!SP10 ;READ LINE STATUS
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP10>>
BR1 TABUPD ;NUMBERED MESSAGE IN PROGRESS
MICPC=MICPC+1
<JUMP!BR1CON!<TABUPD-INIT&3000*4>!<TABUPD-INIT&777/2>>
BR0 TABUPD ;UNNUMBMSGIN PROGRESS
MICPC=MICPC+1
<JUMP!BR0CON!<TABUPD-INIT&3000*4>!<TABUPD-INIT&777/2>>
BRSHFT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR4 IDLE ;START MODE
MICPC=MICPC+1
<JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
ALWAYS SNDACK ;ELSE SEND ACK
MICPC=MICPC+1
<JUMP!ALCOND!<SNDACK-INIT&3000*4>!<SNDACK-INIT&777/2>>
TIME1:
20$: BRWRT IMM,2
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>>
SP BR,SELB,SP15 ;RESET THE TIMER TICK COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP15>
BRWRT DP,<SELA!SP10> ;READ LINE STATUS WORD
MICPC=MICPC+1
<MOVE!WRTEBR!DP!<SELA!SP10>>

```

1533	015074	001265	BRSFT		
(1)	(1)	001620	MICPC=MICPC+1		
1534	015074	001620	<MOVE!SHFTBR!WRTEBR!SELB>		
(1)	(1)	001266	BR4 BSI		; IF IN START MODE--BRANCH
(1)	015076	103076	MICPC=MICPC+1		
1535	015100	001267	<JUMP!BR4CON!<BSI-INIT&3000*4>!<BSI-INIT&777/2>>		
(1)	(1)	060572	BRWRTE BR,DECA!SP12		;GET LAST NUMBER SENT
1536	015102	001270	MICPC=MICPC+1		
(1)	(1)	060377	<MOVE!WRTEBR!BR!<DECA!SP12>>		
1537	015104	001270	CMP BR,SP17		;COMPARE TO LAST ACKED
(1)	(1)	060377	MICPC=MICPC+1		
1538	015106	001271	<SUBTC!BR!SP17>		
(1)	(1)	111654	Z 105		; IF EQ --SEND ACK
1539	015110	001272	MICPC=MICPC+1		
(1)	(1)	010153	<JUMP!ZCOND!<105-INIT&3000*4>!<105-INIT&777/2>>		
1540	015112	001273	LDMA IMM,T		;LOAD ADDRESS OF TYPE FIELD IN UNNUMB SK
(1)	(1)	016403	MICPC=MICPC+1		
1541	015114	001274	<MOVE!LDMA!IMM!<T&377>>		
(1)	(1)	016700	MEMINC IMM,3		;LOAD REP TYPE
1542	015116	001275	MICPC=MICPC+1		
(1)	(1)	010015	<MOVE!WRMEM!INCMAR!IMM!<3>>		
1543	015120	001276	MEMINC IMM,300		;ZERO THE SUB-TYPE
(1)	(1)	043220	MICPC=MICPC+1		
1544	015122	001277	<MOVE!WRMEM!INCMAR!IMM!<300>>		
(1)	(1)	062460	LDMA IMM,REPCS		;CUMULATIVE REPS RECD
1545	015124	001300	MICPC=MICPC+1		
(1)	(1)	010003	<MOVE!LDMA!IMM!<REPCS&377>>		
1546	015126	001301	SP MEMX,SELB,SPO		;COPY IT TO SPO
(1)	(1)	040620	MICPC=MICPC+1		
1547	015130	001302	<MOVE!SPX!MEMX!SELB!SPO>		
(1)	(1)	001303	MEM BR,INCA!SPO		;INCREMENT IT
1548	015132	001304	MICPC=MICPC+1		
(1)	(1)	102353	<MOVE!WRMEM!BR!<INCA!SPO>>		
1549	015134	001305	LDMA IMM,REPST		;ADDRESS DYNAMIC REP COUNTER
(1)	(1)	000601	MICPC=MICPC+1		
1550	015136	001306	<MOVE!LDMA!IMM!<REPST&377>>		
(1)	(1)	063310	BRWRTE MEMX,SELB		;COPY IT TO THE BR
1551	015140	001307	MICPC=MICPC+1		
(1)	(1)	001307	<MOVE!WRTEBR!MEMX!<SELB>>		
			BRSFT		
			MICPC=MICPC+1		
			<MOVE!SHFTBR!SELB!BR>		
			MEM BR,SELB		
			MICPC=MICPC+1		
			<MOVE!WRMEM!BR!<SELB>>		
			BRO RTHRES		
			MICPC=MICPC+1		
			<JUMP!BROCON!<RTHRES-INIT&3000*4>!<RTHRES-INIT&777/2>>		
			BRWRTE IMM,201		;MASK FOR OK TO SEND
			MICPC=MICPC+1		
			<MOVE!WRTEBR!IMM!<201>>		
			SP BR,AORB,SP10		;OR IT IN
			MICPC=MICPC+1		
			<MOVE!SPX!BR!AORB!SP10>		
			ALWAYS IDLE		
			MICPC=MICPC+1		

K10

DMC1: DDCMP PROTOCOL IMPLEMENTATION
DDCNEW.MAC 21-APR-77 10:08

MACY11 30(1046) 11-JUL-77 12:18 PAGE 12-10
TIMSRV--TIMEOUT ROUTINE--SENDS REP

PAGE: 0:27

.1) 015140 100445
:552
:553

<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
.DSABLE LSB
:

1555 015142 001310
 (1) 015142 120620
 1556 015144 001311
 (1) 015144 116063
 1557 015146 001312
 (1) 015146 000402
 1558 015150 001313
 (1) 015150 062231
 1559 015152 001314
 (1) 015152 062230
 1560 015154 001315
 (1) 015154 060601
 1561 015156 001316
 (1) 015156 113755
 1562 015160 001317
 (1) 015160 063072
 1563 015162 001320
 (1) 015162 010071
 1564 015164 001321
 (1) 015164 050220
 1565 015166 001322
 (1) 015166 040620
 1566 015170 001323
 (1) 015170 112334
 1567 015172 001324
 (1) 015172 000775
 1568 015174 001325
 (1) 015174 063670
 1569 015176 001326
 (1) 015176 112334
 1570
 1571 015200 001327
 (1) 015200 010153
 1572 015202 001330
 (1) 015202 016401
 1573 015204 001331
 (1) 015204 000405
 1574 015206

TEOM: BRWRT IBUS,UBBR
 MICPC=MICPC+1
 <MOVE!WRTEBR!IBUS!<UBBR>>
 BR0 NXMERR ;NON-EXISTANT MEMCRY
 MICPC=MICPC+1
 <JUMP!BROCON!<NXMERR-INIT&3000*4>!<NXMERR-INIT&777/2>>
 BRWRT IMM,2 ;EOM TO BR
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<2>>
 OUTPUT BR<SELB!OTMTCO> ;WRITE TMT CONTROL
 MICPC=MICPC+1
 <MOVE!WROUT!BR!<SELB!OTMTCO>>
 OUTPUT BR<SELB!TMTDAT> ;WRITE GARBAGE DATA
 MICPC=MICPC+1
 <MOVE!WROUT!BR!<SELB!TMTDAT>>
 BRWRT BR,SELA!SP1 ;CHECK FOR BOOT MODE
 MICPC=MICPC+1
 <MOVE!WRTEBR!BR!<SELA!SP1>>
 BR7 BTEOM ;---IF SET IS MAINT MSG
 MICPC=MICPC+1
 <JUMP!BR7CON!<BTEOM-INIT&3000*4>!<BTEOM-INIT&777/2>>
 SP BR,INCA,SP12 ;INCREMENT THE MESSAGE NUMBER
 MICPC=MICPC+1
 <MOVE!SPX!BR!INCA!SP12>
 TEOM1: LDMA IMM,LTC ;ADDRESS LAST TMT LINK
 MICPC=MICPC+1
 <MOVE!LDMA!IMM!<LTC&377>>
 LDMA MEMX,SELB
 MICPC=MICPC+1
 <MOVE!LDMA!MEMX!<SELB>>
 BRWRT MEMX,SELB
 MICPC=MICPC+1
 <MOVE!WRTEBR!MEMX!<SELB>>
 BR0 TEOM2
 MICPC=MICPC+1
 <JUMP!BROCON!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>
 TEOM3: BRWRT IMM,375 ;TURN OFF MESSAGE PENDING
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<375>>
 SPBR BR,AANDB,SP10 ;
 MICPC=MICPC+1
 <MOVE!SPBRX!BR!AANDB!SP10>
 BR0 TEOM2 ;IF UNNUMB PENDING--GO AWAY
 MICPC=MICPC+1
 <JUMP!BROCON!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>
 .SBTTL SNDACK--ROUTINE TO SEND AN ACK
 SNDACK: LDMA IMM,T
 MICPC=MICPC+1
 <MOVE!LDMA!IMM!<T&377>>
 MEMINC IMM,1
 MICPC=MICPC+1
 <MOVE!WRMEM!INCMAR!IMM!<1>>
 BRWRT IMM,5
 MICPC=MICPC+1
 <MOVE!WRTEBR!IMM!<5>>
 SA2: MEMINC IMM,300

(1)		001332	MICPC=MICPC+1
(1)	015206	016700	<MOVE!WRMEM!INCMAR!IMM!<300>>
1575	015210		SA3: SP BR,AORB,SP10
(1)		001333	MICPC=MICPC+1
(1)	015210	063310	<MOVE!SPX!BR!AORB!SP10>
1576			.
1577	015212		TEOM2: STATE TMTA
(1)		001334	MICPC=MICPC+1
(1)	015212	000403	<MOVE!WRTEBR!IMM!<TMTA-INIT&777/2>>
1578	015214		ALWAYS XEXIT
(1)		001335	MICPC=MICPC+1
(1)	015214	100451	<JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
1579	015216		FUDGE: BRWRT IBUS,NPR ;READ NPR CONTROL
(1)		001336	MICPC=MICPC+1
(1)	015216	120600	<MOVE!WRTEBR!IBUS!<NPR>>
1580	015220		BR0 IDLE ;IF NPR GOING---LEAVE
(1)		001337	MICPC=MICPC+1
(1)	015220	102045	<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1581	015222		BRWRT BR!LDMAR,SELA!SP4 ;LOAD THE MAR
(1)		001340	MICPC=MICPC+1
(1)	015222	070604	<MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>
1582	015224		BR7 BS2 ;IF SET - READ BACK ALL 200
(1)		001341	MICPC=MICPC+1
(1)	015224	103505	<JUMP!BR7CON!<BS2-INIT&3000*4>!<BS2-INIT&777/2>>
1583	015226		MEMINC IBUS,INDAT1 ;OTHERWISE RESTORE TWO BYTES
(1)		001342	MICPC=MICPC+1
(1)	015226	036400	<MOVE!WRMEM!INCMAR!IBUS!<INDAT1>>
1584	015230		MEMINC IBUS,INDAT2 ;..
(1)		001343	MICPC=MICPC+1
(1)	015230	036420	<MOVE!WRMEM!INCMAR!IBUS!<INDAT2>>
1585	015232		BRWRT IMM,2 ;UPDATE---UNIBUS ADDRESS
(1)		001344	MICPC=MICPC+1
(1)	015232	000402	<MOVE!WRTEBR!IMM!<2>>
1586	015234		SP BR,ADD,SP4 ;UPDATE NPR COUNTER
(1)		001345	MICPC=MICPC+1
(1)	015234	063004	<MOVE!SPX!BR!ADD!SP4>
1587	015236		SP IBUS,IIBA1,SPO ;UPDATE ADDRESS LOW
(1)		001346	MICPC=MICPC+1
(1)	015236	023100	<MOVE!SPX!IBUS!IIBA1!SPO>
1588	015240		OUTPUT BR,ADD!IBA1
(1)		001347	MICPC=MICPC+1
(1)	015240	062004	<MOVE!WROUT!BR!<ADD!IBA1>>
1589	015242		SP IBUS,IIBA2,SPO ;READ HIGH ADDRESS
(1)		001350	MICPC=MICPC+1
(1)	015242	023120	<MOVE!SPX!IBUS!IIBA2!SPO>
1590	015244		OUTPUT BR,AC!IBA2 ;UPDATE HIGH
(1)		001351	MICPC=MICPC+1
(1)	015244	062105	<MOVE!WROUT!BR!<AC!IBA2>>
1591	015246		C RESEXT ;IF CARRY---UPDATE MXT
(1)		001352	MICPC=MICPC+1
(1)	015246	105367	<JUMP!CCOND!<RESEXT-INIT&3000*4>!<RESEXT-INIT&777/2>>
1592	015250		RES1: SP IBUS,NPR,SPO ;READ NPR REGISTER
(1)		001353	MICPC=MICPC+1
(1)	015250	123200	<MOVE!SPX!IBUS!NPR!SPO>
1593	015252		ALWAYS TH3X ;GO DO ANOTHER NPR
(1)		001354	MICPC=MICPC+1

(1) 015252 110574
1594 015254
(1) 015254 001355
(1) 015254 000774
1595 015256
(1) 015256 001356
(1) 015256 063270
1596 015260
(1) 015260 001357
(1) 015260 063233
1597
1598 015262
(1) 015262 001360
(1) 015262 010070
1599 015264
(1) 015264 001361
(1) 015264 043220
1600 015266
(1) 015266 001362
(1) 015266 000431
(1) 015270 001363
(1) 015270 063222
1601 015272
(1) 015272 001364
(1) 015272 114534
1602 015274
(1) 015274 001365
(1) 015274 000715
(1) 015276 001366
(1) 015276 063223
1603 015300
(1) 015300 001367
(1) 015300 123200
1610 015302
(1) 015302 001370
(1) 015302 000621
1611 015304
(1) 015304 001371
(1) 015304 104643
1613 015306
(1) 015306 001372
(1) 015306 023120
1614 015310
(1) 015310 001373
(1) 015310 062065
1615 015312
(1) 015312 001374
(1) 015312 111376
1616 015314
(1) 015314 001375
(1) 015314 110565
1617
1618 015316
(1) 015316 001376
(1) 015316 123200
1619 015320

```

<JUMP!ALCOND!<TH3X-INIT&3000*4>!<TH3X-INIT&777/2>>
BTEOM: BRWRTE IMM,374 ;MASK FOR CLEAR MSG PENDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<374>>
SP BR,AANDB,SP10 ;TURN THEM OFF IN LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SP10>
SP BR,SELB,SP13 ;STORE UNRECOGNIZABLE VALUE INTO SP13
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP13>
;SO "RH3" WILL EXIT BACK TO IDLE LOOP
LDMA IMM,STC ;ADDRESS START OF TMT CHAIN
MICPC=MICPC+1
<MOVE!LDMA!IMM!<STC&377>>
SP MEMX,SELB,SPO ;COPY LINK ADDRESS
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SPO>
TSTATE NUMSYN ;CHANGE XMIT STATE TO LINE IS IDLE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<NUMSYN-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
ALWAYS TDON2 ;POST A DONE
MICPC=MICPC+1
<JUMP!ALCOND!<TDON2-INIT&3000*4>!<TDON2-INIT&777/2>>
RL4: RSTATE RCVL
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVL-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>
SP IBUS,NPR,SPO ;READ NPR CONTROL REGISTER
MICPC=MICPC+1
<MOVE!SPX!IBUS!NPR!SPO>
BRWRTE IMM,221
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<221>>
ALWAYS RK7
MICPC=MICPC+1
<JUMP!ALCOND!<RK7-INIT&3000*4>!<RK7-INIT&777/2>>
HOINCH: SP IBUS,IIBA2,SPO
MICPC=MICPC+1
<MOVE!SPX!IBUS!IIBA2!SPO>
OUTPUT BR,INCA!IBA2 ;OUTPUT INCREMENTED BA
MICPC=MICPC+1
<MOVE!WROUT!BR!<INCA!IBA2>>
;INCREMENT BYTEW COUNT
C 55
MICPC=MICPC+1
<JUMP!CCOND!<55-INIT&3000*4>!<55-INIT&777/2>>
ALWAYS TH8
MICPC=MICPC+1
<JUMP!ALCOND!<TH8-INIT&3000*4>!<TH8-INIT&777/2>>
;INCREMENT MXT BITS
SS: SP IBUS,NPR,SPO ;READ NPR REG IWTH CURRENT MXT BITS
MICPC=MICPC+1
<MOVE!SPX!IBUS!NPR!SPO>
BRWRTE IMM,4 ;WRITE BIT TO ADD

```

```

(1) 001377
(1) 015320 000404
1620 015322
(1) 001400
(1) 015322 061010
1621 015324
(1) 001401
(1) 015324 110565
1622
1623 015326
(1) 001402
(1) 015326 000710
1624 015330
(1) 001403
(1) 015330 100451
1625

```

```

MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<4>>
OUT BR,<ADD!ONPR> ;TURN ON PROPER MXT BITS
MICPC=MICPC+1
<MOVE!WROUTX!BR!<ADD!ONPR>>
ALWAYS TH8
MICPC=MICPC+1
<JUMP!ALCOND!<TH8-INIT&3000*4>!<TH8-INIT&777/2>>
:
HEH1: STATE TEOM
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TEOM-INIT&777/2>>
ALWAYS XEXIT
MICPC=MICPC+1
<JUMP!ALCOND!<XEXIT-INIT&3000*4>!<XEXIT-INIT&777/2>>
:

```

1627
 1628 015332 001404
 (1) 015332 010016
 1629 015334 001405
 (1) 015334 043220
 1630 015336 001406
 (1) 015336 062460
 1631 015340 001407
 (1) 015340 010153
 1632 015342 001410
 (1) 015342 016402
 1633 015344 001411
 (1) 015344 016703
 1634 015346 001412
 (1) 015346 114712
 1635
 1636
 1637 015350 001413
 (1) 015350 060610
 1638 015352 001414
 (1) 015352 001620
 1639 015354 001415
 (1) 015354 117021
 1640
 1641 015356 001416
 (1) 015356 010177
 1642 015360 001417
 (1) 015360 000600
 1643 015362 001420
 (1) 015362 114525
 1644 015364 001421
 (1) 015364 010153
 1645 015366 001422
 (1) 015366 016407
 1646 015370 001423
 (1) 015370 000411
 1647 015372 001424
 (1) 015372 110732
 1648

```

:SBTTL REP HANDLER
REP: LDMA IMM,REPCR ;LOAD MAR ADDRESS WITH POINTER TO REPS RECC
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<REPC?&377>>
      SP MEMX,SELB,SPO ;READ NUMBER OF REPS RECC
      MICPC=MICPC+1
      <MOVE!SPX!MEMX!SELB!SPO>
      MEM DP,<INCA!SPO> ;INCREMNT REPS RECC
      MICPC=MICPC+1
      <MOVE!WRMEM!DP!<INCA!SPO>>
      LDMA IMM,T ;LOAD ADDRESS OF TYPE FIELD
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<T&377>>
      MEMINC IMM,2 ;LOAD NAK TYPE
      MICPC=MICPC+1
      <MOVE!WRMEM!INCMAR!IMM!<2>>
      MEMINC IMM,303 ;LOAD REP RESPONSE SUB-TYPE
      MICPC=MICPC+1
      <MOVE!WRMEM!INCMAR!IMM!<303>>
      ALWAYS SNAK ;SEND AN UNNUMB MSG
      MICPC=MICPC+1
      <JUMP!ALCOND!<SNAK-INIT&3000*4>!<SNAK-INIT&777/2>>
:SBTTL START HANDLER
START: BRWTE DP,<SELA!SP10> ;READ LINE STATUS WORD
       MICPC=MICPC+1
       <MOVE!WTEBR!DP!<SELA!SP10>>
       BRSHFT ;GET START MODE BIT IN TESTABLE POSITION
       MICPC=MICPC+1
       <MOVE!SHFTBR!WTEBR!SELB>
       BR4 10$ ;IF IN START MODE SET STACK
       MICPC=MICPC+1
       <JUMP!BR4CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
       ;ELSE SET UP START ERROR
       LDMA IMM,<<RTHRS+3>>
       MICPC=MICPC+1
       <MOVE!LDMAR!IMM!<<RTHRS+3>&377>>
       BRWTE IMM,200
       MICPC=MICPC+1
       <MOVE!WTEBR!IMM!<200>>
       ALWAYS RCEXY
       MICPC=MICPC+1
       <JUMP!ALCOND!<RCEXY-INIT&3000*4>!<RCEXY-INIT&777/2>>
10$: LDMA IMM,T ;SET UP ADDRESS OF TYPE FIELD
     MICPC=MICPC+1
     <MOVE!LDMAR!IMM!<T&377>>
     MEMINC IMM,7 ;WRITE STACK TYPE
     MICPC=MICPC+1
     <MOVE!WRMEM!INCMAR!IMM!<7>>
     BRWTE IMM,11 ;SET START RECC AND UNNUMB PENDING
     MICPC=MICPC+1
     <MOVE!WTEBR!IMM!<11>>
     ALWAYS SA2 ;SEND THE UNNUMBERED MESSAGE
     MICPC=MICPC+1
     <JUMP!ALCOND!<SA2-INIT&3000*4>!<SA2-INIT&777/2>>

```


1649		
1650	015374	001425
(1)		000727
(1)	015374	000727
1651	015376	001426
(1)		063270
(1)	015376	063270
1652	015400	001427
(1)		110662
(1)	015400	110662

```

STACK: .SBTTL STACK HANDLER
        BRWRT IMM,327 ;MASK TO CLEAR START MODE
        MICPC=MICPC+1
        <MOVE!WRTBR!IMM!<327>>
        SP BR AANDB,SP10 ;CLEAR START MODE
        MICPC=MICPC+1
        <MOVE!SPX!BR!AANDB!SP10>
        ALWAYS TIME1 ;RESET TIMER AND IDLE
        MICPC=MICPC+1
        JUMP!ALCOND!<TIME1-INIT&3000*4! TIME1-INIT&777/2>

```

1654 015402 001430
(1) 015402 023160
1655 015404 001431
(1) 015404 062067
1656 015406 001432
(1) 015406 115034
1657 015410 001433
(1) 015410 104660
1658
1659 015412 001434
(1) 015412 123220
1660 015414 001435
(1) 015414 000404
1661 015416 001436
(1) 015416 061011
1662 015420 001437
(1) 015420 104660
1663 015422 001440
(1) 015422 003200
1664 015424 001441
(1) 015424 062212
1665 015426 001442
(1) 015426 104575
1666 015430 001443
(1) 015430 010070
1667 015432 001444
(1) 015432 057220
1668 015434 001445
(1) 015434 062600
1669 015436 001446
(1) 015436 060477
1670 015440 001447
(1) 015440 063232
1671 015442 001450
(1) 015442 000406
1672
1673 015444 001451
(1) 015444 063310

```
ICBA22: SP      IBUS,IOBA2,SPO      ;READTHEHIGH ORDERBITS OF BA TO SPC
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!IOBA2!SPO>
        OUTPUT DP,<INCA!OBA2>      ;OUTPUT THE INCREMENTED COUNT
        MICPC=MICPC+1
        <MOVE!WROUT!DP!<INCA!OBA2>>
        C      SS      ;IF CARRY SET INCREMENT THE MXTBITS
        MICPC=MICPC+1
        <JUMP!CCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
        ALWAYS RK9
        MICPC=MICPC+1
        <JUMP!ALCOND!<RK9-INIT&3000*4>!<RK9-INIT&777/2>>
        .
SS:     SP      IBUS,U8BR,SPO
        MICPC=MICPC+1
        <MOVE!SPX!IBUS!U8BR!SPO>
        BRWRT IMM,4
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<4>>
        OUT BR,<ADD!OBR>
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<ADD!OBR>>
        ALWAYS RK9
        MICPC=MICPC+1
        <JUMP!ALCOND!<RK9-INIT&3000*4>!<RK9-INIT&777/2>>
FLUSH1: SP      IMM,200,SPO      ;FLUSH THE RECVR
        MICPC=MICPC+1
        <MOVE!SPX!IMM!200!SPO>
        OUTPUT BR,<SELA!ORCVCO>
        MICPC=MICPC+1
        <MOVE!WROUT!BR!<SELA!ORCVCO>>
        ALWAYS CG1
        MICPC=MICPC+1
        <JUMP!ALCOND!<CG1-INIT&3000*4>!<CG1-INIT&777/2>>
NAK:   LDMA IMM,STC      ;ADDRESS START OF TMT CHAIN
        MICPC=MICPC+1
        <MOVE!LDMA!IMM!<STC&377>>
        SP      MEMX!INCMAR,SELB,SPO      ;COPY IT TO SPO
        MICPC=MICPC+1
        <MOVE!SPX!MEMX!INCMAR!SELB!SPO>
        MEM BR,SELA!SPO      ;COPY START OF CHAIN
        MICPC=MICPC+1
        <MOVE!WRMEM!BR!<SELA!SPO>>
        BRWRT BR,INCA!SP17      ;GETLASTMESSAGE ACKED
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<INCA!SP17>>
        SP BR,SELB,SP12      ;COPY TO CURRENT NUMBER
        MICPC=MICPC+1
        <MOVE!SPX!BR!SELB!SP12>
        BRWRT IMM,6      ;WRITE NUMBERED MSG PENDING
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<6>>
        .
        SP BR,AORB,SP10      ; AND LINE HAS GONE IDLE
        MICPC=MICPC+1
        <MOVE!SPX!BR!AORB!SP10>      ;SET IT IN LINE STATUS WORD
```

1674	015446		SP BR SELB,SP15	;RESET TIMER COUNT
(1)		001452	MICPC=MICPC+1	
(1)	015446	063235	<MOVE!SPX!BR!SELB!SP15>	
1675	015450		ALWAYS TEOM1	
(1)		001453	MICPC=MICPC+1	
(1)	015450	110720	<JUMP!ALCOND!<TEOM1-INIT&3000*4>!<TEOM1-INIT&777/2>>	
1676	015452		ININT: BRWRT IMM,15	;MASK FOR TURN OFF ALL BUT EXT MEM BITS + NXM
(1)		001454	MICPC=MICPC+1	
(1)	015452	000415	<MOVE!WRTEBR!IMM!<15>>	
1677	015454		SP IBUS,UBBR,SPO	;READ BR CONTROL REGISTER
(1)		001455	MICPC=MICPC+1	
(1)	015454	123220	<MOVE!SPX!IBUS!UBBR!SPO>	
1678	015456		SP BR,AANDB,SPO	;MASK OFF VECTOR TO X04
(1)		001456	MICPC=MICPC+1	
(1)	015456	063260	<MOVE!SPX!BR!AANDB!SPO>	
1679	015460		BRWRT IMM,200	;MASK FOR INTERRUPT
(1)		001457	MICPC=MICPC+1	
(1)	015460	000600	<MOVE!WRTEBR!IMM!<200>>	
1680	015462		OUT BR,AORB,OBR	;INTERRUPT
(1)		001460	MICPC=MICPC+1	
(1)	015462	061311	<MOVE!WROUTX!BR!<AORB!OBR>>	
1681	015464		SP IBUS,INCON,SPO	;RESTORE INPLT CONTROLCSF
(1)		001461	MICPC=MICPC+1	
(1)	015464	123000	<MOVE!SPX!IBUS!INCON!SPO>	
1682	015466		ALWAYS NIDLE4	
(1)		001462	MICPC=MICPC+1	
(1)	015466	100563	<JUMP!ALCOND!<NIDLE4-INIT&3000*4>!<NIDLE4-INIT&777/2>>	
1683			:	

```

1685
1686 015470 001463
(1) 015470 010177
1687 015472 001464
(1) 015472 016401
1688 015474 001465
(1) 015474 002400
1689 015476 001466
(1) 015476 043230
1690 015500 001467
(1) 015500 114527

```

```

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

```

H11

```

1692
1693
1694 015502 001470
      (1) 015502 023605
1695 015504 001471
      (1) 015504 117477
1696 015506 001472
      (1) 015506 060525
1697 015510 001473
      (1) 015510 117502
1698 015512 001474
      (1) 015512 000477
1699 015514 001475
      (1) 015514 063665
1700 015516 001476
      (1) 015516 164463
1701
1702 015520 001477
      (1) 015520 000600
1703 015522 001500
      (1) 015522 063310
1704 015524 001501
      (1) 015524 114472
1705 015526 001502
      (1) 015526 000420
1706 015530 001503
      (1) 015530 063310
1707 015532 001504
      (1) 015532 114474

```

```

.SBTTL SELQSY--ROUTINETOCHECK SELECT AND QSYNC AND DIDDLE LINE STATUS WORD
:USES SPS. ALWAYS CALLED BY FIRST INSTR IN A RSTATE
SELQSY: SPBR IBUS,RCVDAT,SP5 ;READCHARACTERINTO SPS AND THE BR
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!RCVDAT!SP5>
        BR7 15$ ;SELECT SET?--BRANCH
        MICPC=MICPC+1
        <JUMP!BR7CON!<15$-INIT&3000*4>!<15$-INIT&777/2>>
5$: BRWRT BR,AA!SP5 ;SHIFTBR LEFT
        MICPC=MICPC+1
        <MOVE!WRTEBR!BR!<AA!SP5>>
        BR7 20$ ;FINAL SET?
        MICPC=MICPC+1
        <JUMP!BR7CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>
10$: BRWRT IMM,77 ;MASK TO BR
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<77>>
        SPBR BR,AANDB,SP5 ;TURN OFF SELECTANDFINAL
        MICPC=MICPC+1
        <MOVE!SPBRX!BR!AANDB!SP5>
        ALWAYS BR,INCA,SP3!PAGE1
        MICPC=MICPC+1
        <JUMP!ALCOND!BR!INCA!SP3!PAGE1>
iss: BRWRT IMM,200 ;SET OK TO SEND
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<200>>
        SP BR,AORB,SP10 ;IN LINE STATUS WORD
        MICPC=MICPC+1
        <MOVE!SPX!BR!AORB!SP10>
        ALWAYS 5$
        MICPC=MICPC+1
        <JUMP!ALCOND!<5$-INIT&3000*4 !<5$-INIT&777/2>>
20$: BRWRT IMM,20 ;SETCLEARACTIVE
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<20>>
        SP BR,AORB,SP10 ;IN LINE STATUS WORD
        MICPC=MICPC+1
        <MOVE!SPX!BR!AORB!SP10>
        ALWAYS 10$
        MICPC=MICPC+1
        <JUMP!ALCOND!<10$-INIT&3000*4 !<10$-INIT&777/2>>

```

```

1709
1710 015534 001505
(1) 015534 020540
1711 015536 001506
(1) 015536 116111
1716 015540 001507
(1) 015540 000674
1717 015542 001510
(1) 015542 104422
1718
1719 015544 001511
(1) 015544 000626
1720 015546 001512
(1) 015546 104422
1721
1722 015550 001513
(1) 015550 040364
1723 015552 001514
(1) 015552 115116
1724 015554 001515
(1) 015554 104475
1725 015556 001516
(1) 015556 010153
1726 015560 001517
(1) 015560 016402
1727 015562 001520
(1) 015562 002711
1728 015564 001521
(1) 015564 010175
1729 015566 001522
(1) 015566 036540
1730 015570 001523
(1) 015570 036560
1731 015572 001524
(1) 015572 000420
1732 015574 001525
(1) 015574 016400
1733 015576 001526

```

```

: FUGITIVE RECEIVE ROUTINES---DON'T FIT IN PAGE
RH1: BRWRT IBUS IOBA1 ;READ LOW BYTE OF IN BA
MICPC=MICPC+1
<MOVE!WRTEBR!IBUS!<IOBA1>>
BRD RCVODD ;IF SET IS ODD TRANSFER
MICPC=MICPC+1
<JUMP!BROCON!<RCVODD-INIT&3000*4>!<RCVODD-INIT&777/2>>
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
<MOVE!LDMAR!IMM!<T&377>>
MEMINC IMM,2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<2>>
MEM IMM,311
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<311>>
LDMA IMM,<<RTHRS+1>> ;ADDRESS ERROR LINK
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<<RTHRS+1>&377>>
MEMINC IBUS IOBA1
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<IOBA1>>
MEMINC IBUS IOBA2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<IOBA2>>
BRWRT IMM,20
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<20>>

RCEXY: MEMINC IMM,0
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<0>>
MEM BR SELB
MICPC=MICPC+1

```

(1) 015576 062620
 1734 015600
 (1) 001527
 (1) 015600 002212
 1735 015602
 (1) 001530
 (1) 015602 003001
 1736 015604
 (1) 001531
 (1) 015604 114674
 1737
 1738 015606
 (1) 001532
 (1) 015606 040757
 1739 015610
 (1) 001533
 (1) 015610 107567
 1740 015612
 (1) 001534
 (1) 015612 070200
 1741 015614
 (1) 001535
 (1) 015614 002400
 1742 015616
 (1) 001536
 (1) 015616 010970
 1743 015620
 (1) 001537
 (1) 015620 002473
 1744 015622
 (1) 001540
 (1) 015622 000545
 1745 015624
 (1) 001541
 (1) 015624 060360
 1746 015626
 (1) 001542
 (1) 015626 115545
 1747 015630
 (1) 001543
 (1) 015630 000406
 1748 015632
 (1) 001544
 (1) 015632 062400
 1749 015634
 (1) 001545
 (1) 015634 010241
 1750 015636
 (1) 001546
 (1) 015636 053223
 1751
 1752 015640
 (1) 001547
 (1) 015640 016600
 1753 015642
 (1) 001550

```

RCEXX: <MOVE!WRMEM!BR!<SELB>>
        OUTPUT IMM,<200!ORCVCO> ;FLUSH THE INPUT SILO
        MICPC=MICPC+1
        <MOVE!WROUT!IMM!<200!ORCVCO>>
        SP IMM,1,SP1 ;SET INIT MODE IN PORT STATUS WORD
        MICPC=MICPC+1
        <MOVE!SPX!IMM!1!SP1>
        ALWAYS NTRS1
        MICPC=MICPC+1
        <JUMP!ALCOND!<NTRS1-INIT&3000*4>!<NTRS1-INIT&777/2>>

TDON3: BRWRTE MEMX,SUB!SP17 ;COMPARE RESPONSE TO MSG NC
        MICPC=MICPC+1
        <MOVE!WRTEBR!MEMX!<SUB!SP17>>
        BR7 RH3 ;IF NEGATIVE EXIT
        MICPC=MICPC+1
        <JUMP!BR7CON!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>

TDON2: LDMA BR,SELA!SPO ;ADDRESS THE TRANSMITLINK
        MICPC=MICPC+1
        <MOVE!LDMAR!BR!<SELA!SPO>>
        MEM IMM,0 ;TURN OF ASSIGNEDAND TMTED BITS IN FLAG
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<0>>
        LDMA IMM,STC
        MICPC=MICPC+1
        <MOVE!LDMAR!IMM!<STC&377>>
        MEM IMM,TML1 ;ASSUME WRAPAROUND
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<TML1>>
        BRWRTE IMM,TMLB ;WRAPAROUND?
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<TMLB>>
        CMP BR,SPO
        MICPC=MICPC+1
        <SUBTC!BR!SPO>
        Z TDON4 ;YES
        MICPC=MICPC+1
        <JUMP!ZCOND!<TDON4-INIT&3000*4>!<TDON4-INIT&777/2>>
        BRWRTE IMM,6 ;OFFSET FOR NEXT TMT LINK
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<6>>
        MEM BR,ADD!SPO ;UPDATE THE POINTER
        MICPC=MICPC+1
        <MOVE!WRMEM!BR!<ADD!SPO>>

TDON4: LDMA IMM,NXTSP ;ADDRESS DONE LINK
        MICPC=MICPC+1
        <MOVE!LDMAR!IMM!<NXTSP&377>>
        LDMA MEMX,SELB!SPX!SP3 ;ADDRESS THE LINK,COPYING
        MICPC=MICPC+1
        <MOVE!LDMAR!MEMX!<SELB!SPX!SP3>>
        ;ITS ADDRESS TO SPO
        MEMINC IMM,200 ;WRITE THE INTERRUPT TYPE
        MICPC=MICPC+1
        <MOVE!WRMEM!INCMAR!IMM!<200>>
        MEM BR,INCA!SPO ;COPY ACTUAL LINK ADDRESS
        MICPC=MICPC+1
  
```

(1)	015642	062460	<MOVE!WRMEM!BR!<INCA!SPO>>	
1754	015644		LDMA IMM,NXTSP	;ADDRESS PTR INT STACK
(1)		001551	MICPC=MICPC+1	
(1)	015644	010241	<MOVE!LDMAR!IMM!<NXTSP&377>>	
1755	015646		MEM IMM,INTSTK	;ASSUME WRAP AROUND
(1)		001552	MICPC=MICPC+1	
(1)	015646	002642	<MOVE!WRMEM!IMM!<INTSTK>>	
1756	015650		BRWRT IMM,<<MMEND-2>>	;ADDRESS ENDOFINT STACK
(1)		001553	MICPC=MICPC+1	
(1)	015650	000776	<MOVE!WRTEBR!IMM!<<MMEND-2>>>	
1757	015652		CMP BR,SP3	;WRAPAROUND?
(1)		001554	MICPC=MICPC+1	
(1)	015652	060363	<SUBTC!BR!SP3>	
1758	015654		Z TDON40	;YES---BRANCH
(1)		001555	MICPC=MICPC+1	
(1)	015654	115560	<JUMP!ZCOND!<TDON40-INIT&3000*4>!<TDON40-INIT&777/2>>	
1759	015656		BRWRT IMM,2	;OFFSET TO NEXT PAIR
(1)		001556	MICPC=MICPC+1	
(1)	015656	000402	<MOVE!WRTEBR!IMM!<2>>	
1760	015660		MEM BR,ADD!SP3	;UPDATE POINTER
(1)		001557	MICPC=MICPC+1	
(1)	015660	062403	<MOVE!WRMEM!BR!<ADD!SP3>>	
1761	015662		TDON40: BRWRT IMM,20	;WRITE INTERRUPT PENDING
(1)		001560	MICPC=MICPC+1	
(1)	015662	000420	<MOVE!WRTEBR!IMM!<20>>	
1762	015664		SP BR,AORB,SP1	;IN PORT STATUS WORD
(1)		001561	MICPC=MICPC+1	
(1)	015664	063301	<MOVE!SPX!BR!AORB!SP1>	
1763	015666		LDMA IMM,ETC	;ADDRESS NEXT EMPTY PTR
(1)		001562	MICPC=MICPC+1	
(1)	015666	010072	<MOVE!LDMAR!IMM!<ETC&377>>	
1764	015670		SP MEMX,SELB,SPO	;COPY IT TO SPO
(1)		001563	MICPC=MICPC+1	
(1)	015670	043220	<MOVE!SPX!MEMX!SELB!SPO>	
1765	015672		LDMA IMM,STC	;GET NEXT DONE PTR
(1)		001564	MICPC=MICPC+1	
(1)	015672	010070	<MOVE!LDMAR!IMM!<STC&377>>	
1766	015674		CMP MEMX,SPO	;IDENTICAL?
(1)		001565	MICPC=MICPC+1	
(1)	015674	040360	<SUBTC!MEMX!SPO>	
1767	015676		Z RH3	;FINISH PROCESSING HEADER
(1)		001566	MICPC=MICPC+1	
(1)	015676	105567	<JUMP!ZCOND!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>	
1768				
1769	015700		TDON1: LDMA IMM,ISP17	;GET LAST ACKED
(1)		001567	MICPC=MICPC+1	
(1)	015700	010155	<MOVE!LDMAR!IMM!<ISP17&377>>	
1770	015702		SP MEMX,SELB,SP17	;STORE IT IN SP17
(1)		001570	MICPC=MICPC+1	
(1)	015702	043237	<MOVE!SPX!MEMX!SELB!SP17>	
1771	015704		LDMA IMM,STC	;GET START OF TMT CHAIN
(1)		001571	MICPC=MICPC+1	
(1)	015704	010070	<MOVE!LDMAR!IMM!<STC&377>>	
1772	015706		LDMA MEMX,SELB!SPBRX!SPO	;ADDRESS THE LINK
(1)		001572	MICPC=MICPC+1	
(1)	015706	053620	<MOVE!LDMAR!MEMX!<SELB!SPBRX!SPO>>	

1773 015710
 (1) 001573
 (1) 015710 054620
 1774 015712
 (1) 001574
 (1) 015712 116532
 1775 015714
 (1) 001575
 (1) 015714 104567
 1776
 1777 015716
 (1) 001576
 (1) 015716 000404
 1778 015720
 (1) 001577
 (1) 015720 114671
 1779
 1780
 1781
 1782
 1783 015722
 (1) 001600
 (1) 015722 023336
 1784 015724
 (1) 001601
 (1) 015724 115605
 1785 015726
 (1) 001602
 (1) 015726 000406
 1786 015730
 (1) 001603
 (1) 015730 060360
 1787 015732
 (1) 001604
 (1) 015732 115613
 1788 015734
 (1) 001605
 (1) 015734 060601
 1789 015736
 (1) 001606
 (1) 015736 001620
 1790 015740
 (1) 001607
 (1) 015740 106751
 1791 015742
 (1) 001610
 (1) 015742 001620
 1792 015744
 (1) 001611
 (1) 015744 106743
 1793 015746
 (1) 001612
 (1) 015746 104661
 1794 015750
 (1) 001613
 (1) 015750 063164

```

BRWRT  MEMX! INCMAR, SELB          ;GET THE FLAGS
MICPC=MICPC+1
<MOVE!WRTEBR!MEMX! INCMAR!<SELB>>
BR1    TDON3                      ;IFBLFFER ASSIGNED PROCEED
MICPC=MICPC+1
<JUMP!BRICON!<TDON3-INIT&3000*4>!<TDON3-INIT&777/2>>
ALWAYS RH3                        ;ELSE---EXIT
MICPC=MICPC+1
<JUMP!ALCOND!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>

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

: INPUTS:
: SPO = RECEIVE CHARACTER

PASWRD: SP    IBUS, LNOSW, SP16    ;READ PASSWD SWITCH
MICPC=MICPC+1
<MOVE!SPX!IBUS!LNOSW!SP16>
Z      10$                        ;IF ALL ONES NO RLD ENABLED
MICPC=MICPC+1
<JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
BRWRT  IMM,6                      ;CHECK FOR ENTER MOP MODE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<6>>
CMP    BR, SPO
MICPC=MICPC+1
<SUBTC!BR!SPC>
Z      20$                        ;IF EQUAL ENTER MOP
MICPC=MICPC+1
<JUMP!ZCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
10$:  BRWRT  BR, SELA!SP1          ;READ STATUS BYTE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
BRSHFT                                ;SHIFT IT RIGHT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR1    RHX                        ;MESSAGE WITH NO BUFFER ASSIGNED
MICPC=MICPC+1
<JUMP!BRICON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
BRSHFT                                ;SHIFT RIGHT AGAIN
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR1    RCVMO                      ;DLE RECEIVED IN NORMAL MODE
MICPC=MICPC+1
<JUMP!BRICON!<RCVMO-INIT&3000*4>!<RCVMO-INIT&777/2>>
ALWAYS RK3                          ;HANDLE MAINT MODE MESSAGE
MICPC=MICPC+1
<JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>
20$:  SP    BR, DECA, SP4         ;COUNT FOR NUMB OF CCMPARES
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP4>

```

1795	015752	001614	STATE EM2	
(1)		000746	MICPC=MICPC+1	
(1)	015752		<MOVE!WRTEBR!IMM!<EM2-INIT&777/2>>	
1796	015754	001615	ALWAYS REXIT	
(1)		104422	MICPC=MICPC+1	
(1)	015754		<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>	
1797	015756	001616	RCVM1: LDMA IMM,ISP11	;ADDRESS SP11 IMAGE
(1)		010171	MICPC=MICPC+1	
(1)	015756		<MOVE!LDMAR!IMM!<ISP11&377>>	
1798	015760	001617	MEM BR,DECA!SP11	;COPY SP11
(1)		062571	MICPC=MICPC+1	
(1)	015760		<MOVE!WRMEM!BR!<DECA!SP11>>	
1799	015762	001620	LDMA IMM,NXTSP	
(1)		010241	MICPC=MICPC+1	
(1)	015762		<MOVE!LDMAR!IMM!<NXTSP&377>>	
1800	015764	001621	SP MEMX!LDMAR,SELB,SP3	;COPY TO SP3
(1)		053223	MICPC=MICPC+1	
(1)	015764		<MOVE!SPX!MEMX!LDMAR!SELB!SP3>	
1801	015766	001622	MEMINC IMM,204	;RECEIVE DONE IMAGE
(1)		016604	MICPC=MICPC+1	
(1)	015766		<MOVE!WRMEM!INCMAR!IMM!<204>>	
1802	015770	001623	MEM BR!LDMAR,SELA!SP14	;COPY LINK ADDRESS TO NEXT INTE
(1)		072614	MICPC=MICPC+1	
(1)	015770		<MOVE!WRMEM!BR!LDMAR!<SELA!SP14>>	
1803	015772	001624	MEMINC IMM,0	;ZERO THE FLAGS
(1)		016400	MICPC=MICPC+1	
(1)	015772		<MOVE!WRMEM!INCMAR!IMM!<0>>	
1804	015774	001625	SP IBUS,I0BA1,SP4	;GET BEGIN ADDRESS LOW
(1)		023144	MICPC=MICPC+1	
(1)	015774		<MOVE!SPX!IBUS!I0BA1!SP4>	
1805	015776	001626	SP IBUS,I0BA2,SP5	;AND HIGH BYTE
(1)		023165	MICPC=MICPC+1	
(1)	015776		<MOVE!SPX!IBUS!I0BA2!SP5>	
1806	016000	001627	SP MEMX!INCMAR,SUB,SP4	;SUBTRACT TO GET COUNT
(1)		057344	MICPC=MICPC+1	
(1)	016000		<MOVE!SPX!MEMX!INCMAR!SUB!SP4>	
1807	016002	001630	C 10\$;IF CARRY SET THEN NO CARRY!
(1)		115232	MICPC=MICPC+1	
(1)	016002		<JUMP!CCOND!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>	
1808	016004	001631	SP BR,DECA,SP5	;DECREMENT HIGH BYTE OF ADDRESS
(1)		063165	MICPC=MICPC+1	
(1)	016004		<MOVE!SPX!BR!DECA!SP5>	
1809	016006	001632	SP MEMX!INCMAR,SUB,SP5	;SUBTRACT FOR COUNT
(1)		057345	MICPC=MICPC+1	
(1)	016006		<MOVE!SPX!MEMX!INCMAR!SUB!SP5>	
1810	016010	001633	MEMINC BR,SELA!SP5	;COPY TO MEMORY LINK
(1)		076605	MICPC=MICPC+1	
(1)	016010		<MOVE!WRMEM!INCMAR!BR!<SELA!SP5>>	
1811	016012	001634	MEMINC BR,SELA!SP4	
(1)		076604	MICPC=MICPC+1	
(1)	016012		<MOVE!WRMEM!INCMAR!BR!<SELA!SP4>>	
1812	016014	001635	BRWTE IMM,2	
(1)		000402	MICPC=MICPC+1	
(1)	016014		<MOVE!WRTEBR!IMM!<2>>	
1813	016016	001636	LDMA IMM,NXTSP	;ADDRESS NEXT INT STACK
(1)			MICPC=MICPC+1	

(1)	016016	010241	<MOVE!LDMAR!IMM!<NXTSP&377>>	
1814	016020		MEM BR,ADD!SP3	
(1)		001637	MICPC=MICPC+1	
(1)	016020	062403	<MOVE!WRMEM!BR!<ADD!SP3>>	
1815	016022		BRWRT IMM,<<MMEND-2>>	;ADDRESEND OF INT STACK
(1)		001640	MICPC=MICPC+1	
(1)	016022	000776	<MOVE!WRTEBR!IMM!<<MMEND-2>>>	
1816	016024		CMP BR,SP3	;WRAP AROUND
(1)		001641	MICPC=MICPC+1	
(1)	016024	060363	<SUBTC!BR!SP3>	
1817	016026		Z RMIFLP	;IFYES-- BRANCH
(1)		001642	MICPC=MICPC+1	
(1)	016026	115644	<JUMP!ZCOND!<RMIFLP-INIT&3000*4>!<RMIFLP-INIT&777/2>>	
1818	016030		ALWAYS RMX1	
(1)		001643	MICPC=MICPC+1	
(1)	016030	114645	<JUMP!ALCOND!<RMX1-INIT&3000*4>!<RMX1-INIT&777/2>>	
1819	016032		RMIFLP: MEM IMM,INTSTK	
(1)		001644	MICPC=MICPC+1	
(1)	016032	002642	<MOVE!WRMEM!IMM!<INTSTK>>	
1820	016034		RMX1: LDMA IMM,LRC	
1821	016034		MICPC=MICPC+1	
(1)		001645	<MOVE!LDMAR!IMM!<LRC&377>>	
(1)	016034	010024	BRWRT IMM,5	;INDEX TO NEXT BUFFER
1822	016036		MICPC=MICPC+1	
(1)		001646	<MOVE!WRTEBR!IMM!<5>>	
(1)	016036	000405	SP BR,ADD,SP14	;UPDATE COPY OF POINTER
1823	016040		MICPC=MICPC+1	
(1)		001647	<MOVE!SPX!BR!ADD!SP14>	
(1)	016040	063014	BRWRT IMM,STC	;ADDRESS OF WRAP AROUND POINT
1824	016042		MICPC=MICPC+1	
(1)		001650	<MOVE!WRTEBR!IMM!<STC>>	
(1)	016042	000470	CMP BR,SP14	;WRAPAROUND?
1825	016044		MICPC=MICPC+1	
(1)		001651	<SUBTC!BR!SP14>	
(1)	016044	060374	Z RMFLIP	;IF YES---BRANCH
1826	016046		MICPC=MICPC+1	
(1)		001652	<JUMP!ZCOND!<RMFLIP-INIT&3000*4>!<RMFLIP-INIT&777/2>>	
(1)	016046	115655	MEM BR,SELA!SP14	;ELSE UPDATE THE POINTER
1827	016050		MICPC=MICPC+1	
(1)		001653	<MOVE!WRMEM!BR!<SELA!SP14>>	
(1)	016050	062614	ALWAYS RMX	
1828	016052		MICPC=MICPC+1	
(1)		001654	<JUMP!ALCOND!<RMX-INIT&3000*4>!<RMX-INIT&777/2>>	
(1)	016052	114656	RMFLIP: MEM IMM,RCL1	;POINT TO FIRST LINK
1829	016054		MICPC=MICPC+1	
(1)		001655	<MOVE!WRMEM!IMM!<RCL1>>	
(1)	016054	002425	BRWRT IMM,20	;MASK FOR INTERUPT PENDING
1830	016056		MICPC=MICPC+1	
(1)		001656	<MOVE!WRTEBR!IMM!<20>>	
(1)	016056	000420	SP DP,AORB,SP1	;UPDATE PORT STATUS WORD
1831	016060		MICPC=MICPC+1	
(1)		001657	<MOVE!SPX!DP!AORB!SP1>	
(1)	016060	063301	BRWRT DP,<SELA!SP10>	;READ LINE STATUS WORD
1832	016062		MICPC=MICPC+1	
(1)		001660	<MOVE!WRTEBR!DP!<SELA!SP10>>	
(1)	016062	060610		

1833 016064
 (1) 001661
 (1) 016064 107015
 1834 016066
 (1) 001662
 (1) 016066 000400
 1835 016070
 (1) 001663
 (1) 016070 104422
 1836 016072
 (1) 001664
 (1) 016072 010154
 1837 016074
 (1) 001665
 (1) 016074 043620
 1838 016076
 (1) 001666
 (1) 016076 060400
 1839 016100
 (1) 001667
 (1) 016100 117176
 1840 016102
 (1) 001670
 (1) 016102 000401
 1841 016104
 1842 016104
 (1) 001671
 (1) 016104 010177
 1843 016106
 (1) 001672
 (1) 016106 016400
 1844 016110
 (1) 001673
 (1) 016110 062620
 1845 016112
 (1) 001674
 (1) 016112 010241
 1846 016114
 (1) 001675
 (1) 016114 053220
 1847 016116
 (1) 001676
 (1) 016116 016601
 1848 016120
 (1) 001677
 (1) 016120 002574
 1849 016122
 (1) 001700
 (1) 016122 010241
 1850 016124
 (1) 001701
 (1) 016124 002642
 1851 016126
 (1) 001702
 (1) 016126 000776
 1852 016130

```

BR4 FLUSH ;IF CLEAR ACTIVE SET---FLUSH
MICPC=MICPC+1
<JUMP!BR4CON!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
RMI: STATE RCVA
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVA-INIT&777/2>>
ALWAYS REXIT
MICPC=MICPC+1
<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
NTHRS: LDMA IMM ST
MICPC=MICPC+1
<MOVE!LDMA!IMM!<ST&377>>
SPBR MEMX SELB SPO
MICPC=MICPC+1
<MOVE!SPBR!MEMX!SELB!SPO>
BRWRT BR ADD!SPO ;SHIFT LEFT
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<ADD!SPO>>
BR4 OVRUN
MICPC=MICPC+1
<JUMP!BR4CON!<OVRUN-INIT&3000*4>!<OVRUN-INIT&777/2>>
BRWRT IMM 1
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<1>>
ERRXX:
NTRSO: LDMA IMM,<<RTHRS+3>>
MICPC=MICPC+1
<MOVE!LDMA!IMM!<<RTHRS+3>&377>>
MEMINC IMM 0
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<0>>
MEM BR SELB
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELB>>
NTRSI: LDMA IMM,NXTSP
MICPC=MICPC+1
<MOVE!LDMA!IMM!<NXTSP&377>>
LDMA MEMX SELB!SPX!SPO
MICPC=MICPC+1
<MOVE!LDMA!MEMX!<SELB!SPX!SPO>>
MEMINC IMM 201
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<201>>
MEM IMM,<<RTHRS>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<RTHRS>>>
LDMA IMM,NXTSP
MICPC=MICPC+1
<MOVE!LDMA!IMM!<NXTSP&377>>
MEM IMM,INTSTK ;ASSUME QUEJE WRAP AROUND
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<INTSTK>>
BRWRT IMM,<<MMEND-2>>
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<<MMEND-2>>>
CMP BR,SPO

```

(1)		001703	MICPC=MICPC+1	
(1)	016130	060360	<SUBTC!BR!SPO>	
1853	016132		Z NTRS2	:IT DID WRAP AROUND
(1)		001704	MICPC=MICPC+1	
(1)	016132	115707	<JUMP!ZCOND!<NTRS2-INIT&3000*4>!<NTRS2-INIT&777/2>>	
1854	016134		BRWRT IMM,2	:OFFSET TO NEXT PAIR
(1)		001705	MICPC=MICPC+1	
(1)	016134	000402	<MOVE!WRTEBR!IMM!<2>>	
1855	016136		MEM BR,ADD!SPO	:UPDATE QUEUE POINTER
(1)		001706	MICPC=MICPC+1	
(1)	016136	062400	<MOVE!WRMEM!BR!<ADD!SPO>>	
1856	016140		NTRS2: BRWRT IMM,20	
(1)		001707	MICPC=MICPC+1	
(1)	016140	000420	<MOVE!WRTEBR!IMM!<20>>	
1857	016142		SPBR BR,AORB,SP1	
(1)		001710	MICPC=MICPC+1	
(1)	016142	063701	<MOVE!SPBRX!BR!AORB!SP1>	
1858	016144		BRC TAB1	:FLAGGED BY ERROR TYPE
(1)		001711	MICPC=MICPC+1	
(1)	016144	116342	<JUMP!BROCON!<TAB1-INIT&3000*4>!<TAB1-INIT&777/2>>	
1859	016146		SNAK: LDMA IMM,ISP11	
(1)		001712	MICPC=MICPC+1	
(1)	016146	010171	<MOVE!LDMA!IMM!<ISP11&377>>	
1860	016150		SP MEMX,SELB,SP11	
(1)		001713	MICPC=MICPC+1	
(1)	016150	043231	<MOVE!SPX!MEMX!SELB!SP11>	
1861	016152		SP BR,INCA,SP11	:INCREMENT MSG EXPECTED
(1)		001714	MICPC=MICPC+1	
(1)	016152	063071	<MOVE!SPX!BR!INCA!SP11>	
1862	016154		SNAK1: BRWRT IMM,1	:UNNUMB PENING BIT TO BR
(1)		001715	MICPC=MICPC+1	
(1)	016154	000401	<MOVE!WRTEBR!IMM!<1>>	
1863	016156		SNAK2: SP BR,AORB,SP10	:UPDATE LINE STATUS WORD
(1)		001716	MICPC=MICPC+1	
(1)	016156	063310	<MOVE!SPY!BR!AORB!SP10>	
1864	016160		ALWAYS FLUSH	
(1)		001717	MICPC=MICPC+1	
(1)	016160	104415	<JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>	
1865				
1866	016162		EMTRIG: BRWRT IMM,24	
(1)		001720	MICPC=MICPC+1	
(1)	016162	000424	<MOVE!WRTEBR!IMM!<24>>	
1867	016164		OUTPUT BR,<SELB!OBA1>	
(1)		001721	MICPC=MICPC+1	
(1)	016164	062226	<MOVE!WROUT!BR!<SELB!OBA1>>	
1868	016166		BRWRT IMM,0	
(1)		001722	MICPC=MICPC+1	
(1)	016166	000400	<MOVE!WRTEBR!IMM!<0>>	
1869	016170		OUTPUT BR,<SELB!OBA2>	
(1)		001723	MICPC=MICPC+1	
(1)	016170	062227	<MOVE!WROUT!BR!<SELB!OBA2>>	
1870	016172		SPBR IBUS,BM873,SPO	:READ BM873 ADDRESS---
(1)		001724	MICPC=MICPC+1	
(1)	016172	023740	<MOVE!SPBRX!IBUS!BM873!SPO>	
1871	016174		OUTPUT BR,SELB!OUTDA1	:SET UP LOW BYTE OF ADDRESS
(1)		001725	MICPC=MICPC+1	

```

(1) 016174 062222 <MOVE!WROUT!BR!<SELB!OUTDA1>>
1872 016176 001726 BRWRT IMM,366 ;HIGH BYTE BASE FOR ROM BOOT
(1) 016176 000766 MICPC=MICPC+1
(1) 016176 000766 <MOVE!WRTEBR!IMM!<366>>
1873 016200 001727 OUTPUT BR,SELB!OUTDA2 ;
(1) 016200 062223 MICPC=MICPC+1
1874 016202 001730 <MOVE!WROUT!BR!<SELB!OUTDA2>>
EM6: 016202 000421 BRWRT IMM,21 ;MASK FOR TIMER AND ALSO TO START NPR
(1) 016202 001730 MICPC=MICPC+1
(1) 016202 000421 <MOVE!WRTEBR!IMM!<21>>
1875 016204 001731 CUT BR,<SELB!OBR>
(1) 016204 061231 MICPC=MICPC+1
(1) 016204 061231 <MOVE!WROUTX!BR!<SELB!OBR>>
1876 016206 001732 OUT BR,<SELB!ONPR>
(1) 016206 061230 MICPC=MICPC+1
1877 016210 001733 <MOVE!WROUTX!BR!<SELB!ONPR>>
EM1: 016210 120600 BRWRT IBUS,NPR ;READ NPR CONTROL
(1) 016210 001733 MICPC=MICPC+1
1878 016212 102372 <MOVE!WRTEBR!IBUS!<NPR>>
(1) 016212 102372 BRO CKTIME
(1) 016212 102372 MICPC=MICPC+1
1879 016214 001735 <JUMP!BROCON!<CKTIME-INIT&3000*4>!<CKTIME-INIT&777/2>>
(1) 016214 002662 MEMADR RM1 ;IF NPR DONE
1880 016216 001736 <MOVE!WRMEM!<RM1-INIT&777/2>>
(1) 016216 100765 ALWAYS ACLOW
1881 016220 001737 MICPC=MICPC+1
TABUPD: 016220 023640 <JUMP!ALCOND!<ACLOW-INIT&3000*4>!<ACLOW-INIT&777/2>>
(1) 016220 023640 SPBR IBUS,RCVCON,SPO ;READ RECEIVER CONTROL REG
(1) 016220 001737 MICPC=MICPC+1
1882 016222 001740 <MOVE!SPBRX!IBUS!RCVCON!SPO>
(1) 016222 060400 BRWRT BR,ADD!SPO ;SHIFT LEFT
(1) 016222 060400 MICPC=MICPC+1
1883 016224 001741 <MOVE!WRTEBR!BR!<ADD!SPO>>
(1) 016224 103445 BR7 IDLE ;RECEIVE ACTIVE--IDLE
(1) 016224 103445 MICPC=MICPC+1
1884 016226 001742 <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
TAB1: 016226 010156 LDMA IMM,IMG10
(1) 016226 010156 MICPC=MICPC+1
1885 016230 001743 <MOVE!LDMA!IMM!<IMG10&377>>
(1) 016230 000402 BRWRT IMM,2
(1) 016230 000402 MICPC=MICPC+1
1886 016232 001744 <MOVE!WRTEBR!IMM!<2>>
(1) 016232 076670 MEMINC BR,AANDB!SP10 ;SAVE BIT 1 OF SP10
(1) 016232 076670 MICPC=MICPC+1
1887 016234 001745 <MOVE!WRMEM!INCMAR!BR!\AANDB!SP10>>
(1) 016234 076611 MEMINC BR,SELA!SP11 ;SAVE SP11
(1) 016234 076611 MICPC=MICPC+1
1888 016236 001746 <MOVE!WRMEM!INCMAR!BR!<SELA!SP11>>
(1) 016236 076612 MEMINC BR,SELA!SP12 ;SAVE SP12
(1) 016236 076612 MICPC=MICPC+1
1889 016240 001747 <MOVE!WRMEM!INCMAR!BR!<SELA!SP12>>
(1) 016240 076617 MEMINC BR,SELA!SP17 ;SAVE SP17
(1) 016240 076617 MICPC=MICPC+1
1890 016240 076617 <MOVE!WRMEM!INCMAR!BR!<SELA!SP17>>
;

```

```

1891 016242 STATE RB2 ;NOTE: THE BRG CANNOT BE CHANGED FROM THIS
(1) 001750 MICPC=MICPC+1
(1) 016242 000461 <MOVE!WRTEBR!IMM!<RB2-INIT&777/2>>
1892 ;POINT UNTIL THE BRANCH TO RBO
1893 016244 CMP BR,SP3
(1) 001751 MICPC=MICPC+1
(1) 016244 060363 <SUBTC!BR!SP3>
1894 016246 Z IDLE
(1) 001752 MICPC=MICPC+1
(1) 016246 101445 <JUMP!ZCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1895
1896 016250 SP MEMX,SELB,SP13
(1) 001753 MICPC=MICPC+1
(1) 016250 043233 <MOVE!SPX!MEMX!SELB!SP13>
1997 016252 PSTATE TBU1 ;NEW PORT STATE ADDRESS
(1) 016252 MEM IMM,<<TBU1-INIT&777/2>>
(2) 001754 MICPC=MICPC+1
(2) 016252 002775 <MOVE!WRMEM!IMM!<<TBU1-INIT&777/2>>>
1898 016254 SP IMM,4,SP4 ;INITIALIZE COUNT
(1) 001755 MICPC=MICPC+1
(1) 016254 003004 <MOVE!SPX!IMM!4!SP4>
1899 ;NOTE: FIRST 6 RAM LOCATIONS ARE NOT WRITTEN
1900 ;TO CORE TABLE.
1901 016256 LDMA IMM,BASE ;MAR NOW POINTS TO BASE
(1) 001756 MICPC=MICPC+1
(1) 016256 010017 <MOVE!LDMAR!IMM!<BASE&377>>
1902 016260 ALWAYS RBO ;THE BRG MUST BE PRESET TO STATE RB2
(1) 001757 MICPC=MICPC+1
(1) 016260 104443 <JUMP!ALCOND!<RBO-INIT&3000*4>!<RBO-INIT&777/2>>
1903 016262 EC2: BRWRT IMM,2 ;INCREMENT COUNT/TEST
(1) 001760 MICPC=MICPC+1
(1) 016262 000402 <MOVE!WRTEBR!IMM!<2>>
1904 016264 SP BR,ADD,SP4
(1) 001761 MICPC=MICPC+1
(1) 016264 063004 <MOVE!SPX!BR!ADD!SP4>
1905 016266 SP IBUS,I0BA1,SPC ;POINT TO NEXT ADDRESS
(1) 001762 MICPC=MICPC+1
(1) 016266 023140 <MOVE!SPX!IBUS!I0BA1!SPC>
1906 016270 OUTPUT BR,ADD!OBA1
(1) 001763 MICPC=MICPC+1
(1) 016270 052006 <MOVE!WRROUT!BR!<ADD!OBA1>>
1907 016272 SP IBUS,I0BA2,SPC
(1) 001764 MICPC=MICPC+1
(1) 016272 023160 <MOVE!SPX!IBUS!I0BA2!SPC>
1908 016274 OUTPUT BR,AC!OBA2
(1) 001765 MICPC=MICPC+1
(1) 016274 062107 <MOVE!WRROUT!BR!<AC!OBA2>>
1909 016276 C TABMXT
(1) 001766 MICPC=MICPC+1
(1) 016276 105373 <JUMP!CCOND!<TABMXT-INIT&3000*4>!<TABMXT-INIT&777/2>>
1910
1911 016300 ECX: BRWRT BR!LDMAR,SELA!SP4 ;READ COUNTER
(1) 001767 MICPC=MICPC+1
(1) 016300 070604 <MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>
1912 016302 BR7 20$ ;ALL DONE
(1) 001770 MICPC=MICPC+1

```

```

(1) 016302 117774
1913 016304 001771
(1) 016304 056222
1914 016306 001772
(1) 016306 056223
1915 016310 001773
(1) 016310 104641
1916
1917 016312 001774
(1) 016312 010162
1918 01 4 001775
(1) 016314 062613
1919 016316 001776
(1) 016316 114662
1920 016320 001777
(1) 016320 000000
1921 000001

```

```

<JUMP!BR7CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>
OUTPUT MEMX!INCMAR,SELB!OUTDA1 ;STORE COUNTS OF ERRORS
MICPC=MICPC+1
<MOVE!WROUT!MEMX!INCMAR!<SELB!OUTDA1>>
OUTPUT MEMX!INCMAR,SELB!OUTDA2
MICPC=MICPC+1
<MOVE!WROUT!MEMX!INCMAR!<SELB!OUTDA2>>
ALWAYS RKB
MICPC=MICPC+1
<JUMP!ALCOND!<RKB-INIT&3000*4>!<RKB-INIT&777/2>>

20$: LDMA IMM,PRST
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<PRST&377>>
MEM BR,SELA!SP13
MICPC=MICPC+1
<MOVE!WRMEM!BR!<SELA!SP13>>
ALWAYS RMI
MICPC=MICPC+1
<JUMP!ALCOND!<RMI-INIT&3000*4>!<RMI-INIT&777/2>>
$ZERO
MICPC=MICPC+1
000000
.END

```

. ABS. 016322 COC

ERRORS DETECTED: 0

```

DDCMP/CRF/DS:CRF+DMCNEW,LOW,DDCNEW
RUN-TIME: 5 8 3 SECONDS
RUN-TIME RATIO: 70/13=5.1
COPE USED: 6K (11 PAGES)

```

1668

01400

G12

02DMG MACY11 30.1046) 11-JUL-77 12:11 PAGE 35
02DMG.P11 22-APR-77 09:29

PAGE: 3149

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

:669
:670 0:6322

02800
02900 HIMAP:

:HIGH SPEED (LOCAL) MICRC-CODE

6	MACRO DEFINITIONS
8	REVISION 00
9	FEBRUARY 25, 1975
11	REVISION 01
12	MARCH 18, 1975
13	NEW CSR BOARD CHANGES
14	
15	HARVEY M. SCHLESINGER
17	COPYRIGHT 1975 DIGITAL EQUIPMENT CORPORATION
59	MICRO INSTRUCTION DEFINITIONS
70	BRANCH INSTRUCTIONS
111	INDEXED BRANCH INSTRUCTIONS
140	MOVE INSTRUCTIONS
257	INPUT/OUTPUT ASSIGNMENTS
309	PROTOCOL DEPENDANT MACROS
352	DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT
359	VERSION 00A FEBRUARY 26, 1975
360	
361	HARVEY M. SCHLESINGER
362	
363	COPYRIGHT 1975, DIGITAL EQUIPMENT CORPORATION
364	
365	VERSION 00B MARCH 17, 1975
366	CSR AND MICROPROCESSOR CHANGES
367	
368	VERSION 00C NOVEMBER 6, 1975
369	RETRANSMISSION CHANGES
370	
371	VERSION 00D DECEMBER 3, 1975
372	TRANSMIT DONE CHANGES
373	
374	THE LATEST MODIFICATIONS WERE ADDED ON:
375	NOVEMBER 16, 1976
376	MICROPROCESSOR MAIN MEMORY ASSIGNMENTS
442	SCRATCH PAD ASSIGNMENTS
477	INIT--INITIALIZATION ROUTINE
534	IDLE--PROGRAM IDLE LOOP
565	BASSRV---- BASE SERVICE ROUTINE
602	MIDLE2---NO CSR ACTIVITY STATE
643	INWAIT---WAIT FOR RQI TO CLEAR
693	OUTINT---SET UP OUTPUT INTERRUPT [RDY0]
741	OUTWAI---WAIT FOR RDY0 TO GO AWAY
753	CTLSRV--CNTL I SERVICE
773	TBASRV--TRANSMITTER BUFFER ADDRESS SERVICE
793	RBASRV--RECEIVE BUFFER ADDRESS SERVICE
859	RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER
896	RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD
923	RCVC--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD, SELECT AND FINAL
954	RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES
975	PCVE--ROUTINE TO HANDLE N FIELD OF NUMBERED MESSAGE
988	PCVF--ROUTINE TO IGNORE ADDRESS
996	PCVG--ROUTINE TO IGNORE CRC1
1001	PCVH--ROUTINE TO HANDLE CRC2 AND TO DISPATCH NUMBERED AND UNNUMBERED TYPES
1066	RCVK01--ROUTINE TO HANDLE FIRST BYTE ODD RECEIVE
1076	RCVK0--PROCESS ODD CHARACTER

1096	RCVKE--HANDLE EVEN BYTES
1146	RCVI--STORE UNNUMBERED MESSAGE TYPE
1152	RCVJ--ROUTINE TO HANDLE SUBTYPE FIELD, SELECT AND FINAL
1156	RCVR--UNNUMBERED MESSAGE RESPONSE FIELD
1176	RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD
1182	RCVL--PROCESS CRC3
1204	RCVM--PROCESS CRC4--END OF DATA MESSAGE
1226	EM2--PROCESS RLD MESSAGE
1246	NYMERR ---NON EXISTANT MEMORY HANDLER
1295	TMTDA--TRANSMITTER DISPATCH ROUTINE
1301	TMTA--FIRST CHARACTER OF HEADER
1303	TMTB--OUTPUT FIRST CHAR OF COUNT
1404	TMTC--OUTPUT SECOND CHAR OF COUNT
1428	TMTD--RESPONSE FIELD-NUMBERED MESSAGE
1438	TMTE--NUMBER FIELD--NUMBERED MESSAGE
1447	TMTF--NUMBERED MSG ADDRESS FIELD
1460	TF1-NUMBERED MSG HEADER EOM
1470	TMTH--ROUTINE TO OUTPUT DATA CHARACTERS
1527	TMTI--SEND UNNUMBERED TYPE FIELD
1533	TMTJ--SEND SUB-TYPE FIELD
1538	TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG)
1546	TMTL--UNNUMB MSG NUMBER FIELD
1564	TMTM--UNNUMB MSG--STATION ADDRESS
1580	TIMSRV--TIMEOUT ROUTINE--SENDS REP
1646	SNDACK--ROUTINE TO SEND AN ACK
1713	REP HANDLER
1722	START HANDLER
1735	STACK HANDLER

500-0990-005
11-01-12
12-01-13
13-01-14
14-01-15
15-01-16
16-01-17
17-01-18
18-01-19
19-01-20

.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 :975 DIGITAL EQUIPMENT CORPORATION
:

20 000000
21 000000
22 100000
23 020000
24 000000
25 040000
26 060000
27 060000
28 060000
29 010000
30 014000
31 000400
32 001000
33 001400
34 002000
35 002400
36 003000
37 003400
38 000200
39 000220
40 000240
41 000260
42 000300
43 000320
44 000340
45 000360
46 000000
47 000020
48 000040
49 000060
50 000100
51 000120
52 000140
53 000160
54 004000
55 010000
56 014000
57 001000
58 001400
59 000400
60 002000
61 002400
62 003000
63 003400

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

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

;FUNCTIONS
SELF=200 ;FUNCTION-SELECT A
SELE=220 ;FUNCTION-SELECT B
AORNB=240 ;FUNCTION-A OR NOT B
ANCB=260 ;FUNCTION A AND B
AOR=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
BR0CON=2000 ;CONDITION BR0
BR1CON=2400 ;CONDITION BR1
BR4CON=3000 ;CONDITION BR4
BR7CON=3400 ;CONDITION BR7
```

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
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250

100000

.SBTTL MICRO INSTRUCTION DEFINITIONS
.SBTTL BRANCH INSTRUCTIONS
JUMP=100000 ;JUMP OP CODE

.SBTTL INDEXED BRANCH INSTRUCTIONS

000000

.SBTTL MOVE INSTRUCTIONS
MOVE=0 ;MOVE OPCODE

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

N12

DMC-11 MICROPROCESSOR INSTRUCTIONS
DMCHGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 3-1
PROTOCOL DEPENDANT MACROS

PAGE: 0156

309
316
323
328
332
336
342
348
349
350

177777

.SBTTL PROTOCOL DEPENDANT MACROS

.....

MICPC=177777 ;INIT MICRO PC

B13

DMC-11 MICROPROCESSOR INSTRUCTIONS
HILOW.MAC 03-DEC-76 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 5
DMC11 DDCMP MICRO CODE ASSEMBLED FOR USE WITH THE M8201 LINE UNIT

PAGE: 0157

352
353

000000

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

358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375

.TITLE DMC11 DDCMP PROTOCOL IMPLEMENTATION
.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 NOVEMBER 16, 1976

```

377 .SBTTL MICROPROCESSOR MAIN MEMORY ASSIGNMENTS
378 ;ALLOCATION OF MICROPROCESSOR MAIN MEMORY
379 000000 NAKSR=0 ;NAKS RECD--DYNAMIC
380 000001 NAKS=NAKSR+1 ;NAKS TMTED--DYNAMIC
381 000002 REPSR=NAKST+1 ;REPS RECD--DYNAMIC
382 000003 REPST=REPSR+1 ;REPS TMTED--DYNAMIC
383 000006 NP=REPST+3 ;CONSTANT 0
384 000007 NTLR=NP+1 ;NAKS-MSG NO BUFFERS CUMUL.
385 000010 NHDR=NTLR+1 ;NAKS-MSG HEADER BAD
386 000011 NDATA=NHDR+1 ;NAKS-DATA BAD
387 000012 NTLS=NDATR+1 ;NAK SENT --NO BUFFERS
388 000013 NHDS=NTLS+1 ;NAK SENT BAD HEADER
389 000014 NDATS=NHDS+1 ;NAK SENT BAD DATA
390 000015 REPCS=NDATS+1 ;REPS SENT CUMUL
391 000016 REPCR=REPCS+1 ;REPS RECD CUMUL
392 000017 BASE=REPCR+1 ;CORE TABLE BASE ADDRESS
393 000022 SRC=BASE+3 ;START OF INPUT CHAIN--NEXT RECV DONE
394 000023 ERC=SRC+1 ;END OF INPUT CHAIN
395 000024 RCL1=ERC+1 ;RECEIVE LINK #1
396 000031 RCL2=RCL1+5 ; " " #2
397 000036 RCL3=RCL2+5 ; " " #3
398 000043 RCL4=RCL3+5
399 000050 RCL5=RCL4+5
400 000055 RCL6=RCL5+5
401 000062 RCL7=RCL6+5
402 000067 STC=RCL7+5 ;START OF OUTPUT CHAIN---NEXT TMT DONE
403 000070 ETC=STC+1 ;END OF TRANSMIT CHAIN
404 000071 TML1=ETC+1 ;TRANSMIT LINK #1
405 000077 TML2=TML1+6 ; " " #2
406 000105 TML3=TML2+6 ; " " #3
407 000113 TML4=TML3+6
408 000121 TML5=TML4+6
409 000127 TML6=TML5+6
410 000135 TML7=TML6+6
411 000143 TML8=TML7+6
412 000151 T=TML8+6 ;TYPE FIELD
413 000152 ST=T+1 ;SUBTYPE FIELD
414 000153 ISP17=ST+1 ;MSG ACKED IMAGE
415 000154 IMG10=ISP17+1 ;IMAGE OF BIT 1 OF SP10
416 000155 IMG11=IMG10+1 ;IMAGE OF SP11
417 000156 IMG12=IMG11+1 ;IMAGE OF SP12
418 000157 IMG14=IMG12+1 ;IMAGE OF SP14
419 000160 IMG16=IMG14+1 ;IMAGE OF SP16
420 000161 IMG17=IMG16+1 ;IMAGE OF SP17
421 000162 TYPTAB=IMG17+1 ;TYPE TABLE---
422 ;72 TYPE TABLE REP
423 ;73 " " NAK
424 000164 TYPSTT=TYPTAB+2 ;74 " " START
425 ;75 " " STACK
426
427
428 000167 BC=TYPSTT+3 ;RECEIVE BYTE COUNT
429 000171 ISP11=BC+2 ;SP11 IMAGE
430 000172 ISP12=ISP11+1 ;SP12 IMAGE
431 000173 INCONS=ISP12+1 ;IN CONTROL CSR IMAGE
432 000174 RTHRS=INCONS+1 ;RECV THRESHOLD LINK

```

E13

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCHGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 6-2
MICROPROCESSOR MAIN MEMORY ASSIGNMENTS

PAGE: 0160

433
434
435 000210
436 000211
437 000240
438 000241
439 000242
440 000400

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

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

442		.SBTTL	SCRATCH PAD ASSIGNMENTS
443	000000	SP0=0	:SP0---SCRATCH REGISTER
444	000001	SP1=1	:SP1---PORT STATUS WORD
445			:BIT ASSIGNMENTS
446			:BIT0---INIT MODE
447			:BIT1---SEC STATION SELECT(UNUSED)
448			:BIT2---NO BUFFER ASSIGNED IN BOOT MODE
449			:BIT3---DLE RECEIVED WHILE NOT IN MAINT MODE
450			:BIT4---INTERRUPT PENDING
451			:BIT6---DISCONNECT ERROR
452			:BIT7---BOOT MODE
453	000002	SP2=2	:SP2---TRANSMIT STATE POINTER
454	000003	SP3=3	:SP3---RECEIVE STATE POINTER
455	000004	SP4=4	:SP4---END RECV ADDRESS
456	000005	SP5=5	:SP5---END RECEIVE ADDRESS
457	000006	SP6=6	:SP6---END TRANSMIT ADDRESS
458	000007	SP7=7	:SP7---END TRANSMIT ADDRESS
459	000010	SP10=10	:SP10---LINE STATUS WORD
460			:BIT ASSIGNMENTS
461			:BIT0---UNNUMB PENDING
462			:BIT1---MESSAGE IN PROGRESS
463			:BIT2---LINE HAS GONE IDLE
464			:BIT3---START RECEIVED
465			:BIT4---CLEAR ACTIVE ON END
466			:BIT5---START MODE
467			:BIT6---HALF DUPLEX
468			:BIT7---OK TO SEND
469	000011	SP11=11	:SP11---R FIELD
470	000012	SP12=12	:SP12---N FIELD
471	000013	SP13=13	:SP13---TYPE
472	000014	SP14=14	:SP14---RECEIVE LINK IMAGE
473	000015	SP15=15	:SP15---TIMER ENTRY---NUMBER OF ONE SECOND TICKS
474	000016	SP16=16	:SP16---POINTER TO TMT LINK COPY IN MAIN MEM
475	000017	SP17=17	:SP17---LAST MESSAGE ACKNOWLEDGED

485 016322 016322
 486 016322 063220
 016324 063223
 487 016326 063237
 016330 063237
 488 016330 061200
 016332 061202
 489 016332 063370
 016334 063370
 491 016336 063130
 016336 063130
 493 016340 076423
 016340 076423
 495 016342 063060
 016342 063060
 497 016344 101413
 016344 101413
 498 016346 10040E
 016346 10040E
 499 016350 003401
 016350 003401
 500 016352 063231
 016352 063231
 501 016354 063232
 016354 063232
 502 016356 010162

```

.SBTTL INIT--INITIALIZATION ROUTINE
;ZEROS MAIN MEMORY
;LOOPS WAITING FOR RECEIVE DATA(BOOT?)
;OR FOR RQI TO BE SET
;WILL ACCEPT ONLY BASE FORMAT. ALL OTHERS WILL RETURN A PROCEDURE ERROR
;AT INITIALIZATION --- THE HARDWARE CLEARS THE BR AND MAR
.=16322
INIT: SP BR,SELB,SP0 ;CLEAR SP0
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP0>
SP BR,SELB,SP3 ;PAGE ONE TRANSFER ADDRESS
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP3>
SP BR,SELB,SP17 ;CLEAR SP17
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP17>
OUT BR,<SELA!OINCON> ;ZERO THE IN CONTROL CSR
MICPC=MICPC+1
<MOVE!WROUTX!BR!<SELA!OINCON>>
OUT BR,<SELA!OOCON> ;ZERO THE OUT CONTROL CSR
MICPC=MICPC+1
<MOVE!WROUTX!BR!<SELA!OOCON>>
SP IMM,370,SP10 ;WRITE 5 ONE BITS TO THE HIGH ORDER
MICPC=MICPC+1
<MOVE!SPX!IMM!370!SP10>
;BITS OF SP10
SS: SP BR,AA,SP10 ;SHIFT SP10 LEFT SETTING RBY THE
MICPC=MICPC+1
<MOVE!SPX!BR!AA!SP10>
;FIRST 5 TIMES THRU THE LOOP
MEMINC BR,ADDC!SP3 ;WRITE A ONE TO THE FIRST 5 MEMORY
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<ADDC!SP3>>
;LOCATIONS AND ZERO THE REST
SP BR,INCA,SP0 ;INCREMENT COUNTER
MICPC=MICPC+1
<MOVE!SPX!BR!INCA!SP0>
Z 10$ ;ALL DONE
MICPC=MICPC+1
<JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777*2>>
ALWAYS SS ;KEEP GOING
MICPC=MICPC+1
<JUMP!ALCOND!<SS-INIT&3000*4>!<SS-INIT&777*2>>
10$: SPBR IMM,1,SP1 ;WRITE A 1 TO THE BR AND SP1
MICPC=MICPC+1
<MOVE!SPBRX!IMM!1!SP1>
SP BR,SELB,SP11 ;WRITE A 1 TO SP11
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP11>
SP BR,SELB,SP12 ;WRITE A 1 TO SP12
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP12>
LDMA IMM,TYPTAB ;POINT MAP TO TYPE TABLE
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<TYPTAB&377>>

```

503	016360		BRWRT IMM,226	:WRITE SYNC TO MEMORY
(1)		000017	MICPC=MICPC+1	
(1)	016360	000626	<MOVE!WRTEBR!IMM!<226>>	
504	016362		OUTPUT BR,SELB!SYNC	:LOAD THE SYNC REGISTER
(1)		000020	MICPC=MICPC+1	
(1)	016362	062234	<MOVE!WROUT!BR!<SELB!SYNC>>	
505	016364		MEMINC IMM,3	:REP
(1)		000021	MICPC=MICPC+1	
(1)	016364	016403	<MOVE!WRMEM!INCMAR!IMM!<3>>	
506	016366		MEM IMM,2	:NAK
(1)		000022	MICPC=MICPC+1	
(1)	016366	002402	<MOVE!WRMEM!IMM!<2>>	
507	016370		SP MEMX!INCMAR,SELB,SP15	:SET STARTING COUNT
(1)		001023	MICPC=MICPC+1	
(1)	016370	057235	<MOVE!SPX!MEMX!INCMAR!SELB!SP15>	
508	016372		MEMINC IMM,6	:START
(1)		000024	MICPC=MICPC+1	
(1)	016372	016406	<MOVE!WRMEM!INCMAR!IMM!<6>>	
509	016374		MEMINC IMM,7	:STACK
(1)		000025	MICPC=MICPC+1	
(1)	016374	016407	<MOVE!WRMEM!INCMAR!IMM!<7>>	
510	016376		MEMINC IMM,1	:ACK
(1)		000026	MICPC=MICPC+1	
(1)	016376	016401	<MOVE!WRMEM!INCMAR!IMM!<1>>	
511	016400		LDMA IMM,TABST	:POINT TO TABLE UPDATE STATE
(1)		000027	MICPC=MICPC+1	
(1)	016400	010210	<MOVE!LDMA!IMM!<TABST&377>>	
512	016402		PSTATI I3	:INITIALIZE IT
(1)		016402	MEMINC IMM,<<I3-INIT&777/2>>	
(2)		000030	MICPC=MICPC+1	
(2)	016402	016460	<MOVE!WRMEM!INCMAR!IMM!<<I3-INIT&777/2>>>	
513	016404		PSTATI NIDLE2	:INITIALIZE PORT STATUS
(1)		016404	MEMINC IMM,<<NIDLE2-INIT&777/2>>	
(2)		000031	MICPC=MICPC+1	
(2)	016404	016533	<MOVE!WRMEM!INCMAR!IMM!<<NIDLE2-INIT&777/2>>>	
514	016406		LDMA IMM,STC	:LOAD ADDRESS OF LAST TMT CHAIN
(1)		000032	MICPC=MICPC+1	
(1)	016406	010067	<MOVE!LDMA!IMM!<STC&377>>	
515	016410		MEMINC IMM,TML1	:STORE ADDRESS OF FIRST TMT LINK
(1)		000033	MICPC=MICPC+1	
(1)	016410	016471	<MOVE!WRMEM!INCMAR!IMM!<TML1>>	
516	016412		MEM IMM,TML1	
(1)		000034	MICPC=MICPC+1	
(1)	016412	002471	<MOVE!WRMEM!IMM!<TML1>>	
517	016414		SP MEMX,SELB,SP16	:INITIALIZE LAST XMIT POINTER
(1)		000035	MICPC=MICPC+1	
(1)	016414	043236	<MOVE!SPX!MEMX!SELB!SP16>	
518	016416		LDMA IMM,SRC	:LOAD ADDRESS OF LAST RECV CHAIN
(1)		000036	MICPC=MICPC+1	
(1)	016416	010022	<MOVE!LDMA!IMM!<SRC&377>>	
519	016420		MEMINC IMM,RCL1	:SET UP ADDRESS OF FIRST RECV LINK
(1)		000037	MICPC=MICPC+1	
(1)	016420	016424	<MOVE!WRMEM!INCMAR!IMM!<RCL1>>	
520	016422		MEM IMM,RCL1	
(1)		000040	MICPC=MICPC+1	
(1)	016422	002424	<MOVE!WRMEM!IMM!<RCL1>>	

```

521 016424      SP      MEMX,SELB,SP14
      (1)      000041    MICPC=MICPC+1
      (1) 016424 043234  <MOVE!SPX!MEMX!SELB!SP14>
522 016426      LDMA   IMM,NXTINT      ;ADDRESS OF NEXT INTERRUPT POINTER TO MAR
      (1)      000042    MICPC=MICPC+1
      (1) 016426 010240  <MOVE!LDMA!IMM!<NXTINT&377>>
523 016430      MEMINC IMM,INTSTK      ;INITIALIZE NEXT INTERRUPT POINTER
      (1)      000043    MICPC=MICPC+1
      (1) 016430 016642  <MOVE!WRMEM!INCMAR!IMM!<INTSTK>>
524 016432      MEM     IMM,INTSTK      ;INITIALIZE INSERTION POINTER
      (1)      000044    MICPC=MICPC+1
      (1) 016432 002642  <MOVE!WRMEM!IMM!<INTSTK>>
525 016434      BRWRT  IMM,200          ;WRITE THE PLN BIT TO THE BR
      (1)      000045    MICPC=MICPC+1
      (1) 016434 000600  <MOVE!WRTEBR!IMM!<200>>
526 016436      OUT     BR,<SELB!OMAIN> ;WRITE THE RUN BIT TO MAINT CSR
      (1)      000046    MICPC=MICPC+1
      (1) 016436 061221  <MOVE!WROUTX!BR!<SELB!OMAIN>>
527                                     ;FALL INTO IDLE LOOP
529 016440      ALWAYS TEOM2
      (1)      000047    MICPC=MICPC+1
      (1) 016440 110740  <JUMP!ALCOND!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>
530
531 016442      REXIT: SP      BR,SELB,SP3
      (1)      000050    MICPC=MICPC+1
      (1) 016442 063223  <MOVE!SPX!BR!SELB!SP3>

```


534			.SATTI IDLE--PROGRAM IDLE LOOP
535			.PROGRAM IDLE LOOP
536			.DISPATCHES TO APPROPRIATE SERVICE ROUTINES
537			.USES STATE POINTERS FOR TMT,RCV,CSR ACTIVITY
538			.
539	016444	000051	IDLE: BRWRT BR, <SELA!SP10> ;READ TRANSMIT STATUS WORD FROM SP10 TO BR
(1)		060610	MICPC=MICPC+1
(1)	016444		<MOVE!WRTEBR!BR!<SELA!SP10>>
540	016446	000052	BR1 TMTDA ;IF DATA TO SEND-- BRANCH
(1)		112400	MICPC=MICPC+1
(1)	016446		<JUMP!BR1CON!<TMTDA-INIT&3000*4>!<TMTDA-INIT&777/2>>
541	016450	000053	BRO TMTDA ;IF DATA TO SEND-- BRANCH
(1)		112000	MICPC=MICPC+1
(1)	016450		<JUMP!BROCON!<TMTDA-INIT&3000*4>!<TMTDA-INIT&777/2>>
547	016452	000054	I1: BRWRT IBUS,RCVCON ;READ LINE UNIT RECEIVE CONTROL WORD
(1)		020640	MICPC=MICPC+1
(1)	016452		<MOVE!WRTEBR!IBUS!<RCVCON>>
548	016454	000055	.BR4 BR,SELA,SP3!PAGE1 ;BRANCH BASED UPON RECV STATE
(1)		167203	MICPC=MICPC+1
(1)	016454		<JUMP!BR4CON!BR!SELA!SP3!PAGE1>
549	016456	000056	I2: LDMA IMM,TABST ;POINT TO TABLE UPDATE STATE
(1)		010210	MICPC=MICPC+1
(1)	016456		<MOVE!LDMAR!IMM!<TABST&377>>
550	016460	000057	.ALWAY MEMX,SELB,0
(1)		140620	MICPC=MICPC+1
(1)	016460		<JUMP!ALCOND!MEMX!SELB!0>
551	016462		I3: STATE TMTA+2 ;GET IDLE TRANSMIT STATE + 1
553	016462	000060	MICPC=MICPC+1
(1)		000404	<MOVE!WRTEBR!IMM!<TMTA+2-INIT&777/2>>
554	016464	000061	NOP BR,SUB,SP2 ;SUBTRACT FROM CURRENT STATE
(1)		060342	MICPC=MICPC+1
(1)	016464		<BR!SUB!SP2>
555	016466	000062	C TMTDA ;NON-IDLE STATE
(1)		111000	MICPC=MICPC+1
(1)	016466		<JUMP!CCOND!<TMTDA-INIT&3000*4>!<TMTDA-INIT&777/2>>
557	016470	000063	IDLE0: SPBR IBUS,UBBR,SPO ;TIMER EXPIRES?
(1)		123620	MICPC=MICPC+1
(1)	016470		<MOVE!SPBRX!IBUS!UBBR!SPO>
558	016472	000064	BR4 TIMSRV
(1)		113255	MICPC=MICPC+1
(1)	016472		<JUMP!BR4CON!<TIMSRV-INIT&3000*4>!<TIMSRV-INIT&777/2>>
559	016474	000065	SP IBUS,RCVCON,SPO ;READ THE RECEIVE CONTROL REGISTER
(1)		023240	MICPC=MICPC+1
(1)	016474		<MOVE!SPX!IBUS!RCVCON!SPO>
560	016476	000066	BRWRT BR,AA!SPO ;SHIFT IT LEFT
(1)		060520	MICPC=MICPC+1
(1)	016476		<MOVE!WRTEBR!BR!<AA!SPO>>
561	016500	000067	BR7 I1 ;RECEIVE ACTIVE, DON'T DO PORT STATUS
(1)		103454	MICPC=MICPC+1
(1)	016500		<JUMP!BR7CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
562	016502	000070	LDMA IMM,PRST ;ADDRESS PORT STATE
(1)		010211	MICPC=MICPC+1
(1)	016502		<MOVE!LDMAR!IMM!<PRST&377>>
563	016504	000071	.ALWAY MEMX,SELB,0 ;INDEX
(1)			MICPC=MICPC+1

DMS11 DDCMP PROTOCOL IMPLEMENTATION
DDCHG.MAC 21-APR-77 10:16

K13

MACY11 30(1046) 11-JUL-77 12:25 PAGE 6-8
IDLE--PROGRAM IDLE LOOP

PAGE: 0166

.1 016504 143620

<JUMP!ALCOND!MEMX!SELB!0>

565
 566 016506
 (1) 016506
 (2) 000072
 (2) 016506 002533
 567 016510
 (1) 000073
 (1) 016510 010017
 568 016512
 (1) 000074
 (1) 016512 136500
 569 016514
 (1) 000075
 (1) 016514 136520
 570 016516
 (1) 000076
 (1) 016516 122560
 571 016520
 (1) 000077
 (1) 016520 123000
 572 016522
 (1) 000100
 (1) 016522 000500
 573 016524
 (1) 000101
 (1) 016524 061260
 574 016526
 (1) 000102
 (1) 016526 002133
 575 016530
 (1) 000103
 (1) 016530 040620
 576 016532
 (1) 000104
 (1) 016532 103113
 577 016534
 (1) 000105
 (1) 016534 010151
 578 016536
 (1) 000106
 (1) 016536 016406
 579 016540
 (1) 000107
 (1) 016540 002700
 580 016542
 (1) 000110
 (1) 016542 063161
 581 016544
 (1) 000111
 (1) 016544 000641
 582 016546
 (1) 000112
 (1) 016546 110737
 583 016550
 (1) 000113
 (1) 016550 003004

```

.SBTTL BASSRV---- BASE SERVICE ROUTINE
BASSRV: PSTATE NIDLE2
MEM IMM, <<NIDLE2-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>
LDMA IMM, BASE ;CLEAR TO MAR SO IT POINTS TO BASE POINT
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<BASE&377>>
MEMINC IBUS, PORT1 ;READ CSR4
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT1>>
MEMINC IBUS, PORT2 ;READ CSR5
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IBUS!<PORT2>>
MEM IBUS, PORT4
MICPC=MICPC+1
<MOVE!WRMEM!IBUS!<PORT4>>
SP IBUS, INCON, SPO ;READ INPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPX!IBUS!INCON!SPO>
BRWRT IMM, 100 ;CLEAR THE BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<100>>
OUT BR, <AANDB!OINCON> ;CLEAR THE INCONTROL CSR
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AANDB!OINCON>>
OUTPUT IMM, <120!OMODEM> ;MASK FOR HDX AND DTR
MICPC=MICPC+1
<MOVE!WROUT!IMM!<120!OMODEM>>
BRWRT MEMX, SELB ;READ SEL6
MICPC=MICPC+1
<MOVE!WRTEBR!MEMX!<SELB>>
BR4 RESUME ;IF SET RESUME
MICPC=MICPC+1
<JUMP!BR4CON!<RESUME-INIT&3000*4>!<RESUME-INIT&777/2>>
LDMA IMM, T ;LOAD ADDRESS OF TYPE FIELD
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<T&377>>
MEMINC IMM, 6 ;WRITE START TYPE TO MEMORY
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<6>>
MEM IMM, 300 ;WRITE SELECT AND FINAL TO MEMORY
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<300>>
SP BR, DECA, SP1 ;TURN OFF INIT MODE
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP1>
BS1: BRWRT IMM, 241 ;SET OK TO SEND, STARTMODE AND UNNUM PENDING
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<241>>
ALWAYS SA3
MICPC=MICPC+1
<JUMP!ALCOND!<SA3-INIT&3000*4>!<SA3-INIT&777/2>>
RESUME: SP IMM, SP4, 4 ;SET UP SP4 FOR COUNTING NPRS
MICPC=MICPC+1
<MOVE!SPX!IMM!SP4!4>

```

```

584 016552          SP      BR,INCA,SP10          ;SET UNNUMB MESSAGE PENDING TO
(1) 016552 000114  MICPC=MICPC+1
(1) 016552 063070  <MOVE!SPX!BR!INCA!SP10>
585 016554          LDMA   IMM,BASE          ;TRICK TRANSMITTER CODE
586 016554          MICPC=MICPC+1          ;ADDRESS BASE TABLE ADDRESS
(1) 016554 000115  <MOVE!LDMAR!IMM!<BASE&377>>
(1) 016554 010017  STATE  FUDGE          ;SET TMTR STATE TO ENTER TABLE UPDATE
587 016556          MICPC=MICPC+1
(1) 016556 000116  <MOVE!WRTEBR!IMM!<FUDGE-INIT&777/2>>
(1) 016556 000743  ALWAYS  TBO          ;GO SET UP MXT BITS AND ADRESS OF BASE FOR NPRS
588 016560          MICPC=MICPC+1
(1) 016560 000117  <JUMP!ALCOND!<TBO-INIT&3000*4>!<TBC-INIT&777/2>>
589 016562          BS2: LDMA   IMM,IMG10
(1) 016562 000120  MICPC=MICPC+1
(1) 016562 010154  <MOVE!LDMAR!IMM!<IMG10&377>>
590 016564          SP      MEMX!INCMAR,AORB,SP10      ;RESTORE BIT 1 OF SP10
(1) 016564 000121  MICPC=MICPC+1
(1) 016564 057310  <MOVE!SPX!MEMX!INCMAR!AORB!SP10>
591 016566          SP      MEMX!INCMAR,SE_B,SP11      ;RESTORE SP11
(1) 016566 000122  MICPC=MICPC+1
(1) 016566 057231  <MOVE!SPX!MEMX!INCMAR!SELB!SP11>
592 016570          SP      MEMX!INCMAR,SELB,SP12      ;RESTORE SP12
(1) 016570 000123  MICPC=MICPC+1
(1) 016570 057232  <MOVE!SPX!MEMX!INCMAR!SELB!SP12>
593 016572          SP      MEMX!INCMAR,SELB,SP14      ;RESTORE SP14
(1) 016572 000124  MICPC=MICPC+1
(1) 016572 057234  <MOVE!SPX!MEMX!INCMAR!SELB!SP14>
594 016574          SP      MEMX!INCMAR,SELB,SP16      ;RESTORE SP16
(1) 016574 000125  MICPC=MICPC+1
(1) 016574 057236  <MOVE!SPX!MEMX!INCMAR!SELB!SP16>
595 016576          SP      MEMX,SELB,SP17          ;RESTORE SP17
(1) 016576 000126  MICPC=MICPC+1
(1) 016576 043237  <MOVE!SPX!MEMX!SELB!SP17>
596 016600          SP      BR,DECA,SP10          ;TURN OFF UNNUM MESSAGE PENDING AND
(1) 016600 000127  MICPC=MICPC+1
(1) 016600 063170  <MOVE!SPX!BR!DECA!SP10>
597 016602          ;ZERO THE BRG
598 016602          ;CLEAR INIT MODE
(1) 016602 000130  SP      BR,DECA,SP1
(1) 016602 063161  MICPC=MICPC+1
599 016604          <MOVE!SPX!BR!DECA!SP1>
(1) 016604 000131  BRWRTE IMM,200          ;SET OK TO SEND
(1) 016604 000600  MICPC=MICPC+1
600 016606          <MOVE!WRTEBR!IMM!<200>>
(1) 016606 000132  ALWAYS  SA3
(1) 016606 110737  MICPC=MICPC+1
<JUMP!ALCOND!<SA3-INIT&3000*4>!<SA3-INIT&777/2>>

```

```

602
603 016610
(1) 016610 000133
(1) 016610 0606C.
605 016612
(1) 016612 000134
(1) 016612 103141
610 016614
(1) 016614 000135
(1) 016614 123400
611 016616
(1) 016616 000136
(1) 016616 001620
612 016620
(1) 016620 000137
(1) 016620 103146
613
614
616 016622
(1) 016622 000140
(1) 016622 100451
618 016624
639 016624
(1) 016624
(2) 016624 000141
(2) 016624 002614
640 016626
(1) 016626 000142
(1) 016626 100451

.SBTTL NIDLE2---NO CSR ACTIVITY STATE
NIDLE2: BRWRTE BR,SELA!SP1 ;READ PORT STATUS WORD
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
BR4 NIDLES ;INTERRUPT PENDING?---BRANCH
MICPC=MICPC+1
<JUMP!BR4CON!<NIDLES-INIT&3000*4>!<NIDLES-INIT&777/2>>
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

ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
NIDLE6:
NIDLES: PSTATE OUTINT ;SET STATE FOR INTERRUPT PROCESSING
MEM IMM,<<OUTINT-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<OUTINT-INIT&777/2>>>
ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

```

643
644 016630 000143
(1) 016630 123400
645 016632 000144
(1) 016632 060520
646 016634 000145
(1) 016634 103550
647
648 016636 000146
(1) 016636 123400
649 016640 000147
(1) 016640 103557
650 016642 000150
(1) 016642 002546
(2) 016642 002546
651 016644 000151
(1) 016644 060520
652 016646 000152
(1) 016646 117460
653 016650 000153
(1) 016650 002543
(2) 016650 002543
654 016652 000154
(1) 016652 000600
655 016654 000155
(1) 016654 061300
656 016656 000156
(1) 016656 100451
657
658 016660 000157
(1) 016660 001620
663 016662 000160
(1) 016662 103051
664 016664 000161
(1) 016664 002563
(2) 016664 002563
665 016666 000162
(1) 016666 100451
667 016670 000163
(1)

```

.SBTTL INWAIT---WAIT FOR RQI TO CLEAR
INWAIT: SPBR   IBUS,INCON,SPO           ;READ INPLT CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!INCON!SPO>
        BRWRT  BR, <AA!SPO>           ;SHIFT IT LEFT
        MICPC=MICPC+1
        <MOVE!WRTBR!BR!<AA!SPO>>
        BR7    NIDLE3                 ;INTERRUPT ENABLE HAS BEEN SET
        MICPC=MICPC+1
        <JUMP!BR7CON!<NIDLE3-INIT&3000*4>!<NIDLE3-INIT&777/2>>

INWAIT1: SPBR   IBUS,INCON,SPO         ;READ THE INPUT CONTROL CSR
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!INCON!SPO>
        BR7    INWAIT2                ;READY IN STILL SET
        MICPC=MICPC+1
        <JUMP!BR7CON!<INWAIT2-INIT&3000*4>!<INWAIT2-INIT&777/2>>
NIDLE3:  PSTATE INWAIT                ;UPDATE STATE TO INPUT
        MEM    IMM, <<INWAIT1-INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<<INWAIT1-INIT&777/2>>>
        BRWRT  BR, AA!SPO             ;SHIFT CSR LEFT
        MICPC=MICPC+1
        <MOVE!WRTBR!BR!<AA!SPO>>
        BR7    ININT
        MICPC=MICPC+1
        <JUMP!BR7CON!<ININT-INIT&3000*4>!<ININT-INIT&777/2>>
        PSTATE INWAIT                ;UPDATE STATE POINTER TO NO INTERRUPT GENERATED
        MEM    IMM, <<INWAIT-INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<<INWAIT-INIT&777/2>>>
NIDLE4:  BRWRT  IMM, 200
        MICPC=MICPC+1
        <MOVE!WRTBR!IMM!<200>>
        OUT    BR, AORB!OINCON        ;SET THE RDYI
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<AORB!OINCON>>
        ALWAYS IDLE
        MICPC=MICPC+1
        <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

INWAIT2: BRSHFT                        ;SHIFT THE BR RIGHT
        MICPC=MICPC+1
        <MOVE!SHFTBR!WRTBR!SELB>
        BR4    IDLE
        MICPC=MICPC+1
        <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        PSTATE INSRV                ;SET NEXT STATE TO INPUT SERVICE
        MEM    IMM, <<INSRV-INIT&777/2>>
        MICPC=MICPC+1
        <MOVE!WRMEM!IMM!<<INSRV-INIT&777/2>>>
        ALWAYS IDLE
        MICPC=MICPC+1
        <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
INSRV:  SPBR   IBUS,INCON,SPO         ;READ THE INPUT CONTROL CSR
        MICPC=MICPC+1
    
```

```

(1) 016670 123400 <MOVE!SPBRX!IBUS!INCON!SP0>
668 016672 000164 BR1 30$ ;--SENSE OR BASE
(1) 016672 102600 MICPC=MICPC+1
(1) 016672 102600 <JUMP!BR1CON!<30$-INIT&3000*4>!<30$-INIT&777/2>>
669 016674 000165 BRO 10$ ;CNTL I
(1) 016674 102172 MICPC=MICPC+1
(1) 016674 102172 <JUMP!BROCON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
670 016676 000166 BRSHFT ;MUST BE BA/CC-SHIFT FOR IN OR OUT
(1) 016676 000166 MICPC=MICPC+1
(1) 016676 001620 <MOVE!SHFTBR!WRTEBR!SELB>
671 016700 000167 BR1 15$
(1) 016700 102574 MICPC=MICPC+1
(1) 016700 102574 <JUMP!BR1CON!<15$-INIT&3000*4>!<15$-INIT&777/2>>
672 016702 000170 PSTATE TBASRV ;TRANSMITTER
(1) 016702 000170 MEM IMM,<<TBASRV-INIT&777/2>>
(2) 016702 002700 MICPC=MICPC+1
(2) 016702 002700 <MOVE!WRMEM!IMM!<<TBASRV-INIT&777/2>>>
673 016704 000171 ALWAYS 20$
(1) 016704 100575 MICPC=MICPC+1
(1) 016704 100575 <JUMP!ALCONO!<20$-INIT&3000*4>!<20$-INIT&777/2>>
674 016706 10$: PSTATE CTLSRV
(1) 016706 000172 MEM IMM,<<CTLSRV-INIT&777/2>>
(2) 016706 002657 MICPC=MICPC+1
(2) 016706 002657 <MOVE!WRMEM!IMM!<<CTLSRV-INIT&777/2>>>
675 016710 000173 ALWAYS 20$
(1) 016710 100575 MICPC=MICPC+1
(1) 016710 100575 <JUMP!ALCONO!<20$-INIT&3000*4>!<20$-INIT&777/2>>
676 016712 15$: PSTATE RBASRV
(1) 016712 000174 MEM IMM,<<RBASRV-INIT&777/2>>
(2) 016712 002721 MICPC=MICPC+1
(2) 016712 002721 <MOVE!WRMEM!IMM!<<RBASRV-INIT&777/2>>>
677 016714 20$: BRWRTE BR SELA!SP1 ;INIT MODE
(1) 016714 000175 MICPC=MICPC+1
(1) 016714 060601 <MOVE!WRTEBR!BR!<SELA!SP1>>
678 016716 000176 BRO PROCER ;IF INIT MODE--ERROR
(1) 016716 102201 MICPC=MICPC+1
(1) 016716 102201 <JUMP!BROCON!<PROCER-INIT&3000*4>!<PROCER-INIT&777/2>>
679 016720 ALWAYS IDLE
(1) 016720 000177 MICPC=MICPC+1
(1) 016720 100451 <JUMP!ALCONO!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
680 016722 30$: BRO INSRV1 ;IF BASE---PROCESS
(1) 016722 000200 MICPC=MICPC+1
(1) 016722 102211 <JUMP!BROCON!<INSRV1-INIT&3000*4>!<INSRV1-INIT&777/2>>
681 016724 PROCER: PSTATE NIDLE2 ;RESET PORT STATUS
(1) 016724 000201 MEM IMM,<<NIDLE2-INIT&777/2>>
(2) 016724 002533 MICPC=MICPC+1
(2) 016724 002533 <MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>
682 016726 BRWRTE IMM,100 ;CLEAR INPUT CONTROL CSR
(1) 016726 000202 MICPC=MICPC+1
(1) 016726 000500 <MOVE!WRTEBR!IMM!<100>>
683 016730 OUT BR AANDB!OINCON ;;
(1) 016730 000203 MICPC=MICPC+1
(1) 016730 061260 <MOVE!WROUTX!BR!<AANDB!OINCON>>
684 016732 LDMA IMM,<<RTHRS+3>> ;ADDRESS ERROR LINK
(1) 016732 000204 MICPC=MICPC+1
(1) 016732 010177 <MOVE!LDMAR!IMM!<<RTHRS+3 3377>>

```

685 016734
 (1) 016734 000205
 (1) 016734 016402
 686 016736
 (1) 016736 000206
 (1) 016736 002400
 687 016740
 (1) 016740 000207
 (1) 016740 042233
 688 016742
 (1) 016742 000210
 (1) 016742 114524
 689 016744
 (1) 016744 000211
 (1) 016744 060601
 690 016746
 (1) 016746 000212
 (1) 016746 102072
 691 016750
 (1) 016750 000213
 (1) 016750 100601

```

MEMINC IMM,2
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!<2>>
MEM IMM,0
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<0>>
OUTPUT MEMX,SELB!OMODEM ;CLEAR DATA TERMINAL READY
MICPC=MICPC+1
<MOVE!WROUT!MEMX!<SELB!OMODEM>>
ALWAYS RCEXX ;POST THE ERROR - FATAL
MICPC=MICPC+1
<JUMP!ALCOND!<RCEXX-INIT&3000*4>!<RCEXX-INIT&777/2>>
INSRV1: BRWRITE BR SELA!SP1 ;INIT MODE?
MICPC=MICPC+1
<MOVE!WRITEBR!BR!<SELA!SP1>>
BRO BASSRV
MICPC=MICPC+1
<JUMP!BROCON!<BASSRV-INIT&3000*4>!<BASSRV-INIT&777/2>>
ALWAYS PROCER ;NO - PROCEDURE ERROR
MICPC=MICPC+1
<JUMP!ALCOND!<PROCER-INIT&3000*4>!<PROCER-INIT&777/2>>

```


693			.SBTTL OUTINT---SET UP OUTPUT INTERRUPT [RDY0]
694	016752		OUTINT: PSTATE PINT2
696	016752		MEM IMM, <<PINT2-INIT&777/2>>
(1)	016752		MICPC=MICPC+1
(2)		000214	<MOVE!WRMEM!IMM!<<PINT2-INIT&777/2>>>
(2)	016752	002631	;COMPLETION
701			LDMA IMM, NXTINT ;ADDRESS OF NEXT INTERRUPT PCINTER
702	016754		MICPC=MICPC+1
(1)		000215	<MOVE!LDMA!IMM!<NXTINT&377>>
(1)	016754	010240	LDMA MEMX, SELB ;NEXT INTERRUPT
703	016756		MICPC=MICPC+1
(1)		000216	<MOVE!LDMA!MEMX!<SELB>>
(1)	016756	050220	SP IBUS, OCON, SPO ;READ THE OUTPUT CONTROL CSR
704	016760		MICPC=MICPC+1
(1)		000217	<MOVE!SPX!IBUS!OCON!SPO>
(1)	016760	123040	OUT <MEMX!INCMAR>, <AORB!OCON> ;WRITE THE OUT CONTROL CSR
705	016762		MICPC=MICPC+1
(1)		000220	<MOVE!WROUTX!MEMX!INCMAR!<AORB!OCON>>
(1)	016762	055302	LDMA MEMX, SELB ;ADDRESS LINK
706	016764		MICPC=MICPC+1
(1)		000221	<MOVE!LDMA!MEMX!<SELB>>
(1)	016764	050220	BRWRT <BR!INCMAR>, <AA!SPO> ;KICK PAST LINK STATUS BYTE
707	016766		MICPC=MICPC+1
(1)		000222	<MOVE!WRTEBR!BR!INCMAR!<AA!SPO>>
(1)	016766	074520	;SHIFT CSRO IMAGE LEFT
708			OUT <MEMX!INCMAR>, <SELB!OPORT1> ;***DO NOT CHANGE BR UNTIL BR7***
709			MICPC=MICPC+1 ;WRITE LOW BYTE OF BA TO CSR
(1)		000223	<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT1>>
(1)	016770	055224	OUT <MEMX!INCMAR>, <SELB!OPORT2> ;WRITE HIGH BYTE OF BA TO CSR
710	016770		MICPC=MICPC+1
(1)		000224	<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT2>>
(1)	016772	055225	OUT <MEMX!INCMAR>, <SELB!OPORT4> ;WRITE HIGH BYTE OF COUNT TO CSR
711	016772		MICPC=MICPC+1
(1)		000225	<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT4>>
(1)	016774	055227	OUT <MEMX!INCMAR>, <SELB!OPORT3> ;WRITE THE LOW BYTE OF COUNT
712	016774		MICPC=MICPC+1
(1)		000226	<MOVE!WROUTX!MEMX!INCMAR!<SELB!OPORT3>>
(1)	016776	055226	;***HERE IS BR7***
713	016776		BR7 PE1 ;INTERRUPT ENABLE IS SET
(1)		000227	MICPC=MICPC+1
(1)	017000	103757	<JUMP!BR7CON!<PE1-INIT&3000*4>!<PE1-INIT&777/2>>
714	017000		;GENERATE AN INTERRUPT
715	017002		ALWAYS IDLE
(1)		000230	MICPC=MICPC+1
(1)	017002	100451	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
716	017004		PINT2: PSTATE OUTWAIT
(1)		000231	MEM IMM, <<OUTWAIT-INIT&777/2>>
(2)		002652	MICPC=MICPC+1
(2)	017004		<MOVE!WRMEM!IMM!<<OUTWAIT-INIT&777/2>>>
717	017006		LDMA IMM, NXTINT ;ADDRESS NEXT INTERRUPT QUEUE
(1)		000232	MICPC=MICPC+1
(1)	017006	010240	<MOVE!LDMA!IMM!<NXTINT&377>>
718	017010		SP MEMX, SELB, SPO ;COPY ADDRESS FOR NEXT INT TO SPC
(1)		000233	MICPC=MICPC+1

(1)	017010	043220	<MOVE!SPX!MEMX!SELB!SPO>	
726	017012		MEM IMM,INTSTK	;ASSUME WRAP AROUND CASE
(1)		000234	MICPC=MICPC+1	
(1)	017012	002642	<MOVE!WRMEM!IMM!<INTSTK>>	
727	017014		BRWRT IMM,<<MMEND-2>>	;ADDRESS OF LAST INT IN STACK
(1)		000235	MICPC=MICPC+1	
(1)	017014	000776	<MOVE!WRTEBR!IMM!<<MMEND-2>>>	
728	017016		CMP BR,SPO	;SHOULD WE WRAP
(1)		000236	MICPC=MICPC+1	
(1)	017016	060360	<SUBTC!BR!SPO>	
729	017020		Z SS	;YES--BRANCH
(1)		000237	MICPC=MICPC+1	
(1)	017020	101642	<JUMP!ZCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>	
730	017022		BRWRT IMM,2	;OFFSET FOR NEXT POINTER
(1)		000240	MICPC=MICPC+1	
(1)	017022	000402	<MOVE!WRTEBR!IMM!<2>>	
731	017024		MEM BR,ADD!SPO	;UPDATE POINTER
(1)		000241	MICPC=MICPC+1	
(1)	017024	062400	<MOVE!WRMEM!BR!<ADD!SPO>>	
732	017026		SS: SP MEMX,SELB,SPO	;COPY POINTER TO SPO
(1)		000242	MICPC=MICPC+1	
(1)	017026	043220	<MOVE!SPX!MEMX!SELB!SPO>	
733	017030		LDMA IMM,NXTSP	;PICK UP START OF IN QUEUE
(1)		000243	MICPC=MICPC+1	
(1)	017030	010241	<MOVE!LDMA!IMM!<NXTSP&377>>	
734	017032		CMP MEMX,SPO	;COMPARE TO END
(1)		000244	MICPC=MICPC+1	
(1)	017032	040360	<SUBTC!MEMX!SPO>	
735	017034		Z 10S	;IF EQUAL--CLEAR INT PENDING
(1)		000245	MICPC=MICPC+1	
(1)	017034	101647	<JUMP!ZCOND!<10S-INIT&3000*4>!<10S-INIT&777/2>>	
736	017036		ALWAYS IDLE	
(1)		000246	MICPC=MICPC+1	
(1)	017036	100451	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
737	017040		10S: BRWRT IMM,357	;MASK TO CLEAR INT PENDING
(1)		000247	MICPC=MICPC+1	
(1)	017040	000757	<MOVE!WRTEBR!IMM!<357>>	
738	017042		CLRIDL: SP BR,AANDB,SP1	
(1)		000250	MICPC=MICPC+1	
(1)	017042	063261	<MOVE!SPX!BR!AANDB!SP1>	
739	017044		ALWAYS IDLE	
(1)		000251	MICPC=MICPC+1	
(1)	017044	100451	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	

741
742 017046 000252
(1) 123440
747 017050 000253
(1) 103451
749 017052 000254
(1) 000500
750 017054 000255
(1) 061262
751 017056 000256
(1) 100671

.SBTTL OUTWAI--WAIT FOR RDYO TO GO AWAY
OUTWAI: SPBR IBUS,OCON,SPO ;READ OUTPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!OCON!SPO>
BR7 IDLE
MICPC=MICPC+1
<JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
BRWRT IMM,100 ;CLEAR CONTROL BITS
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<100>>
OUT BR,OCON!AANDB
MICPC=MICPC+1
<MOVE!WROUTX!BR!<OCON!AANDB>>
ALWAYS INS13
MICPC=MICPC+1
<JUMP!ALCOND!<INS13-INIT&3000*4>!<INS13-INIT&777/2>>

753			SBTTL CTLSRV--CNTL I SERVICE	
754	017060		CTLSRV: SPBR IBUS,PORT4,SP0	:TO SPO
(1)		000257	MICPC=MICPC+1	
(1)	017060	123560	<MOVE!SPBRX!IBUS!PORT4!SPO>	
755	017062		BRSHFT	
(1)		000260	MICPC=MICPC+1	
(1)	017062	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
756	017064		BR1 HDSEL	:IF SET IS HALF DUPLEX
(1)		000261	MICPC=MICPC+1	
(1)	017064	102754	<JUMP!BR1CON!<HDSEL-INIT&3000*4>!<HDSEL-INIT&777/2>>	
757	017066		OUTPUT IMM,<100!OMODEM>	:MASK DTR, TURN OFF HDX
(1)		000262	MICPC=MICPC+1	
(1)	017066	002113	<MOVE!WROUT!IMM!<100!OMODEM>>	
758	017070		INS11: BRWRITE DP <SELA!SPO>	:RESTORE THE CNTL WORD
(1)		000263	MICPC=MICPC+1	
(1)	017070	060600	<MOVE!WRTEBR!DP!<SELA!SPO>>	
759	017072		BR0 CBOOT	:IF SET IS BOOT
(1)		000264	MICPC=MICPC+1	
(1)	017072	102273	<JUMP!BR0CON!<CBOOT-INIT&3000*4>!<CBOOT-INIT&777/2>>	
760	017074		INS12: SP IBUS,INCON,SP0	:READ THE INPUT CONTROL CSR
(1)		000265	MICPC=MICPC+1	
(1)	017074	123000	<MOVE!SPX!IBUS!INCON!SPO>	
761	017076		BRWRITE IMM,100	:ZERO THE BR REGISTER EXCEPT INT ENABLE
(1)		000266	MICPC=MICPC+1	
(1)	017076	000500	<MOVE!WRTEBR!IMM!<100>>	
762	017100		OUT BR,<AANDB!0INCON>	:CLEAR IN CONTROL CSR
(1)		000267	MICPC=MICPC+1	
(1)	017100	061260	<MOVE!WROUTX!BR!<AANDB!0INCON>>	
763	017102		LDMA IMM,PRST	:ADDRESS PORT STATE
(1)		000270	MICPC=MICPC+1	
(1)	017102	010211	<MOVE!LDMA!IMM!<PRST&377>>	
764	017104		INS13: PSTATE NIDLE2	
(1)	017104		MEM IMM,<<NIDLE2-INIT&777/2>>	
(2)		000271	MICPC=MICPC+1	
(2)	017104	002533	<MOVE!WRMEM!IMM!<<NIDLE2-INIT&777/2>>>	
765	017106		ALWAYS IDLE	
(1)		000272	MICPC=MICPC+1	
(1)	017106	100451	<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
766				
767	017110		CBOOT: BRWRITE IMM,200	:MASK FOR BOOT MODE
(1)		000273	MICPC=MICPC+1	
(1)	017110	000600	<MOVE!WRTEBR!IMM!<200>>	
768	017112		SP BR,AORB,SP1	:IN PORT STATUS WORD
(1)		000274	MICPC=MICPC+1	
(1)	017112	063301	<MOVE!SPX!BR!AORB!SP1>	
769	017114		BRWRITE IMM,204	:MASK FOR OK TO SEND AND LINE IDLE
(1)		000275	MICPC=MICPC+1	
(1)	017114	000604	<MOVE!WRTEBR!IMM!<204>>	
770	017116		SP BR,SELB,SP10	:IN LINE STATUS
(1)		000276	MICPC=MICPC+1	
(1)	017116	063230	<MOVE!SPX!BR!SELB!SP10>	
771	017120		ALWAYS INS12	
(1)		000277	MICPC=MICPC+1	
(1)	017120	100665	<JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>	

773
774 017122 000300
(1) 017122 010070
775 017124 000301
(1) 017124 053220
776 017126 000302
(1) 017126 016401
777 017130 000303
(1) 017130 014543
778
779 017132 000304
(1) 017132 136500
780 017134 000305
(1) 017134 136520
781 017136 000306
(1) 017136 136560
782 017140 000307
(1) 017140 136540
783 017142 000310
(1) 017142 010070
784 017144 000311
(1) 017144 002471
785 017146 000312
(1) 017146 060360
786 017150 000313
(1) 017150 101716
787 017152 000314
(1) 017152 000406
788 017154 000315
(1) 017154 062400
789 017156 000316
(1) 017156 000402
790 017160 000317
(1) 017160 063310
791 017162 000320
(1) 017162 100665

```

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

```

10\$:

793
794 017164 000321
(1) 017164 010023
795 017166 000322
(1) 017166 053220
796 017170 000323
(1) 017170 016401
797 017172 000324
(1) 017172 136500
798 017174 000325
(1) 017174 136520
799 017176 000326
(1) 017176 136560
800 017200 000327
(1) 017200 136540
801
802 017202 000330
(1) 017202 010023
803 017204 000331
(1) 017204 002424
804 017206 000332
(1) 017206 000462
805 017210 000333
(1) 017210 060360
806 017212 000334
(1) 017212 101665
807 017214 000335
(1) 017214 000405
808 017216 000336
(1) 017216 062400
809 017220 000337
(1) 017220 100665
810 017222 000340
(1) 017222 000717
811 017224 000341
(1) 017224 063670
812 017226 000342
(1) 017226 000400

```

SBTTL RBASRV--RECEIVE BUFFER ADDRESS SERVICE
RBASRV: LDMA IMM,ERC ;ADDRESS END OF RECEIVE CHAIN
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<ERC&377>>
LDMA MEMX,<SELB!SPX!SPO> ;GET THE POINTER TO LINK
MICPC=MICPC+1
<MOVE!LDMAR!MEMX!<SELB!SPX!SPO>>
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
<MOVE!LDMAR!IMM!<ERC&377>>
MEM IMM,RCL1 ;ASSUME WRAP AROUND CASE
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<RCL1>>
BRWRT IMM,RCL7 ;GET ADDRESS OF END OF CASH AREA
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCL7>>
CMP BR,SPO ;CHECK FOR END
MICPC=MICPC+1
<SUBTC!BR!SPO>
Z INS12 ;IF EQUAL BRANCH
MICPC=MICPC+1
<JUMP!ZCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
BRWRT IMM,5 ;CALCULATE ADDRESS OF NEXT LINK
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<5>>
MEM BR,ADD!SPO ;...
MICPC=MICPC+1
<MOVE!WRMEM!BR!<ADD!SPO>>
ALWAYS INS12 ;EXIT
MICPC=MICPC+1
<JUMP!ALCOND!<INS12-INIT&3000*4>!<INS12-INIT&777/2>>
RA1: BRWRT IMM,317 ;MASK TO CLEAR START MODE AND CLR ACTIVE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<317>>
SPBR BR,AANDB,SP10 ;CLEAR BIT IN LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPBRX!BR!AANDB!SP10>
RA3: BRWRT IMM,0 ;CLEAR BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<0>>

```

813 017230
 (1) 000343
 (1) 017230 063233
 814 017232
 (1) 000344
 (1) 017232 000424
 815 017234
 (1) 000345
 (1) 017234 100450
 816
 818 017236
 (1) 000346
 (1) 017236 060530
 819 017240
 (1) 000347
 (1) 017240 103351
 820 017242
 (1) 000350
 (1) 017242 100451
 821 017244
 (1) 000351
 (1) 017244 000727
 822 017246
 (1) 000352
 (1) 017246 063270
 823 017250
 (1) 000353
 (1) 017250 104507

```

SP      BR,SELB,SP13          ;SET NUMB MESSAGE TYPE IN SP13
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP13>
STATE  RCVB                    ;CHANGE RECEIVE STATE POINTER TO STATE B
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVB-INIT&777/2>>
ALWAYS REXIT
MICPC=MICPC+1
<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
;
ACK:   BRWRT  BR,AA!SP10        ;READ LINE STATUS SHIFTING LEFT
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<AA!SP10>>
BR4    SS                        ;IF START RECD--CLEAR START MODE
MICPC=MICPC+1
<JUMP!BR4CON!<SS-INIT&3000*4>!<SS-INIT&777/2>>
ALWAYS IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
SS:   BRWRT  IMM,327            ;CLEAR START MODE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<327>>
SP      BR,AANDB,SP10          ;IN LINE STATUS
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SP10>
ALWAYS RDS
MICPC=MICPC+1
<JUMP!ALCOND!<RDS-INIT&3000*4>!<RDS-INIT&777/2>>

```

826 017252 000354
 (1) 017252 000500
 827 017254 000355
 (1) 017254 063310
 828 017256 000356
 (1) 017256 100663
 829
 830 017260 000357
 (1) 017260 000700
 831 017262 000360
 (1) 017262 123220
 832 017264 000361
 (1) 017264 061311
 834 017266 000362
 (1) 017266 100451
 839
 840 017270 000363
 (1) 017270 002722
 841
 842 017272 000364
 (1) 017272 000402
 843 017274 000365
 (1) 017274 061231
 844 017276 000366
 (1) 017276 120620
 845 017300 000367
 (1) 017300 102766
 846 017302 000370
 (1) 017302 154620
 847 017304 000371
 (1) 017304 120620
 848 017306 000372
 (1) 017306 103363
 849 017310 000373
 (1) 017310 114725
 850
 851 017312 000374
 (1) 017312 120600
 852 017314

HDSEL: BRWRITE IMM,100 ;HD MASK TO BR
 MICPC=MICPC+1
 <MOVE!WRITEBR!IMM!<100>>
 SP BR,AORB,SP10 ;UPDATE PORT STATUS WORD
 MICPC=MICPC+1
 <MOVE!SPX!BR!AORB!SP10>
 ALWAYS INS11
 MICPC=MICPC+1
 <JUMP!ALCOND!<INS11-INIT&3000*4>!<INS11-INIT&777/2>>
 .
 PE1: BRWRITE IMM,300 ;MASK FOR INTERRUPT AND VECTOR THROUGH X04
 MICPC=MICPC+1
 <MOVE!WRITEBR!IMM!<300>>
 SP IBUS,UBBR,SP0 ;READ BR CONTROL REG
 MICPC=MICPC+1
 <MOVE!SPX!IBUS!UBBR!SP0>
 OUT BR,<AORB!OBR> ;INTERRUPT
 MICPC=MICPC+1
 <MOVE!WROUTX!BR!<AORB!OBR>>
 ALWAYS IDLE
 MICPC=MICPC+1
 <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
 .
 HALTED: MEMADR EM6
 MICPC=MICPC+1
 <MOVE!WRMEM!<EM6-INIT&777/2>>
 .
 ACLOW: BRWRITE IMM,2 ;FALL INTO ACLOW
 MICPC=MICPC+1 ;CAUSE AN AC LOW
 <MOVE!WRITEBR!IMM!<2>>
 OUT BR,<SELB!OBR>
 MICPC=MICPC+1
 <MOVE!WROUTX!BR!<SELB!OBR>>
 SS: BRWRITE IBUS,UBBR ;WAIT FOR IT TO COMPLETE
 MICPC=MICPC+1
 <MOVE!WRITEBR!IBUS!<UBBR>>
 BR1 SS
 MICPC=MICPC+1
 <JUMP!BR1CON!<SS-INIT&3000*4>!<SS-INIT&777/2>>
 .ALWAY MEMX,SELB,PAGE3
 MICPC=MICPC+1
 <JUMP!ALCOND!MEMX!SELB!PAGE3>
 CKTIME: BRWRITE IBUS,UBBR ;READ BR CONTROL REG
 MICPC=MICPC+1
 <MOVE!WRITEBR!IBUS!<UBBR>>
 BR4 HALTED
 MICPC=MICPC+1
 <JUMP!BR4CON!<HALTED-INIT&3000*4>!<HALTED-INIT&777/2>>
 ALWAYS EM1
 MICPC=MICPC+1
 <JUMP!ALCOND!<EM1-INIT&3000*4>!<EM1-INIT&777/2>>
 .
 †BUI: BRWRITE IBUS,NPR
 MICPC=MICPC+1
 <MOVE!WRITEBR!IBUS!<NPR>>
 BRO IDLE

M14

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCHGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 6-23
RBASRV--RECEIVE BUFFER ADDRESS SERVICE

PAGE: 0181

(1) 000375
(1) 017314 102051
853 017316
(1) 000376
(1) 017316 114752
854 017320
(1) 000377
(1) 017320 000000
855

MICPC=MICPC+1
<JUMP! BROCON! <IDLE-INIT&3000*4>! IDLE-INIT&777/2>>
ALWAYS EC2
MICPC=MICPC+1
<JUMP! AI COND! <EC2-INIT&3000*4>! <EC2-INIT&777/2>>
SZERC
MICPC=MICPC+1
000000

857	017322	. = INIT+1000
858	000377	MICPC=377
859		.SBTTL RCVA--ROUTINE TO HANDLE FIRST DDCMP CHARACTER
860		: ENTERED FROM IDLE LOOP
861		: DETERMINES IF MESSAGE TYPE IS NUMBERED, UNNUMBERED OR BOOT
862		: SETS UP APROPRIATE STATES FOR REST OF MESSAGE.
863	017322	RCVA: SP IBUS RCV DAT, SPO ; READ RECEIVE CHARACTER TO SPO
(1)	000400	MICPC=MICPC+1
(1)	017322 023200	<MOVE!SPX!IBUS!RCV DAT!SPO>
864	017324	BRWRT BR SELA!SP1 ; READ PORT STATUS WORD
(1)	000401	MICPC=MICPC+1
(1)	017324 060601	<MOVE!WRTEBR!BR!<SELA!SP1>>
865	017326	BR0 S\$; IF INIT MODE---ONLY BOOT OK
(1)	000402	MICPC=MICPC+1
(1)	017326 106012	<JUMP!BR0CON!<S\$-INIT&3000*4>!<S\$-INIT&777/2>>
866	017330	BR7 S\$; IF BOOT MODE---ONLY BOOT OK
(1)	000403	MICPC=MICPC+1
(1)	017330 107412	<JUMP!BR7CON!<S\$-INIT&3000*4>!<S\$-INIT&777/2>>
867	017332	BRWRT IMM,201 ; SOH TO BR
(1)	000404	MICPC=MICPC+1
(1)	017332 000601	<MOVE!WRTEBR!IMM!<201>>
868	017334	CMP BR SPO ; COMPARE BR TO SPO
(1)	000405	MICPC=MICPC+1
(1)	017334 060360	<SUBTC!BR!SPO>
869	017336	Z RA1 ; IF EQUAL-IS NUMBERED MESSAGE
(1)	000406	MICPC=MICPC+1
(1)	017336 101740	<JUMP!ZCOND!<RA1-INIT&3000*4>!<RA1-INIT&777/2>>
870	017340	BRWRT IMM,5 ; ENQ TO BR
(1)	000407	MICPC=MICPC+1
(1)	017340 000405	<MOVE!WRTEBR!IMM!<5>>
871	017342	CMP BR SPO ; COMPARE ENQ TO SPO
(1)	000410	MICPC=MICPC+1
(1)	017342 060360	<SUBTC!BR!SPO>
872	017344	Z RA2 ; IF EQUAL-IS UNNUMBERED MESSAGE
(1)	000411	MICPC=MICPC+1
(1)	017344 105422	<JUMP!ZCOND!<RA2-INIT&3000*4>!<RA2-INIT&777/2>>
873	017346	BRWRT IMM,220 ; DLE TO BR
(1)	000412	MICPC=MICPC+1
(1)	017346 000620	<MOVE!WRTEBR!IMM!<220>>
874	017350	CMP BR SPO ; COMPARE DLE TO SPO
(1)	000413	MICPC=MICPC+1
(1)	017350 060360	<SUBTC!BR!SPO>
875	017352	Z BOOT ; IF EQUAL IS BOOT
(1)	000414	MICPC=MICPC+1
(1)	017352 105756	<JUMP!ZCOND!<BOOT-INIT&3000*4>!<BOOT-INIT&777/2>>
876	017354	FLUSH: OUTF JT IMM,<200!ORCVCO> ; FLUSH INPUT SILC
(1)	000415	MICPC=MICPC+1
(1)	017354 002212	<MOVE!WROUT!IMM!<200!ORCVCO>>
877		; (LOW ORDER BITS READ ONLY)
878	017356	BRWRT IMM,357 ; MASK TO CLEAR--CLEAR ACTIVE
(1)	000416	MICPC=MICPC+1
(1)	017356 000757	<MOVE!WRTEBR!IMM!<357>>
879	017360	SP BR AANDB,SP10 ; IN LINE STATUS WORD
(1)	000417	MICPC=MICPC+1
(1)	017360 063270	<MOVE!SPX!BR!AANDB!SP10>
884	017362	RMI: STATE RCVA

(1)		000420
(1)	017362	000400
885	017364	
(1)		000421
(1)	017364	100450
887	017366	
(1)		000422
(1)	017366	000665
889	017370	
(1)		000423
(1)	017370	100450

```

MICPC=MICPC+1
<MOVE!WRTBR!IMM!<RCVA-INIT&777/2>>
ALWAYS REXIT
MICPC=MICPC+1
<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
RA2: STATE RCVI ;CHANGE RECEIVE STATE TO I
MICPC=MICPC+1
<MOVE!WRTBR!IMM!<RCVI-INIT&777/2>>
ALWAYS REXIT
MICPC=MICPC+1
<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>

```

896			.SBTTL RCVB--ROUTINE TO HANDLE FIRST CHARACTER OF COUNT FIELD
897			; ENTERED FROM IDLE LOOP
898			; STORES COUNT FIELD AND SETS UP RCVC AS NEXT STATE
899			;
900	017372		RCVB: SP IBUS,RCVDAT,SP4 ;READ CHARACTER TO SP4
901	017372		MICPC=MICPC+1
(1)		000424	<MOVE!SPX!IBUS!RCVDAT!SP4>
(1)	017372	023204	LDMA BR,<SELA!SP14> ;LOAD MAR WITH ADDRESS OF CURRENT BR
902	017374		MICPC=MICPC+1
(1)		000425	<MOVE!LDMAR!BR!<SELA!SP14>>
(1)	017374	070214	BRWRT MEMX,INCMAR,SELB ;READ FLAGS BYTE
903	017376		MICPC=MICPC+1
(1)		000426	<MOVE!WRTEBR!MEMX!<INCMAR!SELB>>
(1)	017376	054620	BRO RB1 ;RCV BUFFER ASSIGNED---CONTINUE
904	017400		MICPC=MICPC+1
(1)		000427	<JUMP!BROCON!<RB1-INIT&3000*4>!<RB1-INIT&777/2>>
(1)	017400	106041	BRWRT BR,SELA!SP1 ;READ STATUS BYTE
905	017402		MICPC=MICPC+1
(1)		000430	<MOVE!WRTEBR!BR!<SELA!SP1>>
(1)	017402	060601	BR7 RB3 ;MAINT MODE
906	017404		MICPC=MICPC+1
(1)		000431	<JUMP!BR7CON!<RB3-INIT&3000*4>!<RB3-INIT&777/2>>
(1)	017404	107437	LDMA IMM,T ;ERROR--LOAD TYPE FIELD ADDRESS IN MAR
907	017406		MICPC=MICPC+1
(1)		000432	<MOVE!LDMAR!IMM!<T&377>>
(1)	017406	010151	MEMINC IMM,2 ;LOAD NAK TYPE
908	017410		MICPC=MICPC+1
(1)		000433	<MOVE!WRMEM!INCMAR!IMM!<2>>
(1)	017410	016402	MEM IMM,310 ;LOAD SUB-TYPE NO BUFFERS
909	017412		MICPC=MICPC+1
(1)		000434	<MOVE!WRMEM!IMM!<310>>
(1)	017412	002710	LDMA IMM,NTLS
910	017414		MICPC=MICPC+1
(1)		000435	<MOVE!LDMAR!IMM!<NTLS&377>>
(1)	017414	010012	ALWAYS RNS ;BRANCH TO SEND NAK ROUTINE
911	017416		MICPC=MICPC+1
(1)		000436	<JUMP!ALCOND!<RHS-INIT&3000*4>!<RHS-INIT&777/2>>
(1)	017416	104552	BRWRT IMM,4 ;MASK FOR NO BUFFER AVAILABLE
912	017420		MICPC=MICPC+1
(1)		000437	<MOVE!WRTEBR!IMM!<4>>
(1)	017420	000404	SP BR,AORB,SP1 ;SET THE FLAG
913	017422		MICPC=MICPC+1
(1)		000440	<MOVE!SPX!BR!AORB!SP1>
(1)	017422	063301	RB1: STATE RCVC
914	017424		MICPC=MICPC+1
(1)		000441	<MOVE!WRTEBR!IMM!<RCVC-INIT&777/2>>
(1)	017424	000461	RB0: SP BR,SELB,SP3
915	017426		MICPC=MICPC+1
(1)		000442	<MOVE!SPX!BR!SELB!SP3>
(1)	017426	063223	OUTPUT <MEMX!INCMAR>,<SELB!OBA1> ;OUTPUT LOW ORDER BYTE OF ADDRESS
916	017430		MICPC=MICPC+1
(1)		000443	<MOVE!WROUT!MEMX!INCMAR!<SELB!OBA1>>
(1)	017430	056226	OUTPUT MEMX!INCMAR,<SELB!OBA2> ;OUTPUT HIGH BYTE OF ADDRESS
917	017432		MICPC=MICPC+1
(1)		000444	<MOVE!WROUT!MEMX!INCMAR!<SELB!OBA2>>
(1)	017432	056227	

```

919 017434 SP IBUS,UBBR,SPO ;READ THE BUS REQ REGISTER
(1) 000445 MICPC=MICPC+1
(1) 017434 123220 <MOVE!SPX!IBUS!UBBR!SPO>
919 017436 BRWRT IMM,101 ;MASK OFF ALL BUT NXM AND VECH BITS
(1) 000446 MICPC=MICPC+1
(1) 017436 000501 <MOVE!WRTEBR!IMM!<101>>
920 017440 SP BR,AANDB,SPO ;AND SAVE IN SPO
(1) 000447 MICPC=MICPC+1
(1) 017440 063260 <MOVE!SPX!BR!AANDB!SPO>
921 017442 SP IMM,300,SP5 ;MASK TO ISOLATE EX. MEM BITS
(1) 000450 MICPC=MICPC+1
(1) 017442 003305 <MOVE!SPX!IMM!300!SP5>
922 ;NOTE THIS REALLY WRITES A 305 BUT THE
923 ;5 GETS SHIFTED OUT
924 017444 BRWRT MEMX,AANDB!SP5 ;MASK ALL BUT EX. MEM BITS
(1) 000451 MICPC=MICPC+1
(1) 017444 040665 <MOVE!WRTEBR!MEMX!<AANDB!SP5>>
925 017446 BRSHFT ;SHIFT THEM INTO THE CORRECT POSITION
(1) 000452 MICPC=MICPC+1
(1) 017446 001620 <MOVE!SHFTBR!WRTEBR!SELB>
926 017450 BRSHFT
(1) 000453 MICPC=MICPC+1
(1) 017450 001620 <MOVE!SHFTBR!WRTEBR!SELB>
927 017452 BRSHFT
(1) 000454 MICPC=MICPC+1
(1) 017452 001620 <MOVE!SHFTBR!WRTEBR!SELB>
928 017454 BRSHFT
(1) 000455 MICPC=MICPC+1
(1) 017454 001620 <MOVE!SHFTBR!WRTEBR!SELB>
929 017456 OUT BR,AORB,OBR ;WRITE EX MEM BITS OUT
(1) 000456 MICPC=MICPC+1
(1) 017456 061311 <MOVE!WROUTX!BR!<AORB!OBR>>
930 017460 ALWAYS IDLE
(1) 000457 MICPC=MICPC+1
(1) 017460 100451 <JUMP!ALCOND!<IDLE-INIT$3000*4>!<IDLE-INIT$777.2>>
931 017462 ALWAYS I2
(1) 000460 MICPC=MICPC+1
(1) 017462 100456 <JUMP!ALCOND!<I2-INIT$3000*4 ! I2-INIT$777.2 >

```

P92:

933
934
935
936
937
938 017464
943 017464
(1) 000461
(1) 017464 023205
944 017466
(1) 000462
(1) 017466 000600
945 017470
(1) 000463
(1) 017470 060665
946 017472
(1) 000464
(1) 017472 063310
947 017474
(1) 000465
(1) 017474 010167
948 017476
(1) 000466
(1) 017476 076604
949 017500
(1) 000467
(1) 017500 076605
951 017502
(1) 000470
(1) 017502 000472
952 017504
(1) 000471
(1) 017504 100450

```
.SBTTL RCVC--ROUTINE TO HANDLE SECOND CHARACTER OF COUNT FIELD, SELECT AND FINAL
; ENTERED FROM IDLE LOOP
; INTERPRETS SELECT AND FINAL
; CHECKS FOR COUNT TOO LARGE
;
RCVC:
    SP      IBUS,RCVDAT,SP5      ;GET CHARACTER
    MICPC=MICPC+1
    <MOVE!SPX!IBUS!RCVDAT!SP5>
    BRWRT  IMM,200              ;SEPARATE SELECT BIT FROM COUNT
    MICPC=MICPC+1
    <MOVE!WRTBR!IMM!<200>>
    BRWRT  BR,AANDB!SP5
    MICPC=MICPC+1
    <MOVE!WRTBR!BR!<AANDB!SP5>>
    SP      BR,AORB,SP10
    MICPC=MICPC+1
    <MOVE!SPX!BR!AORB!SP10>
    LDMA   IMM,BC                ;LOAD MAR TO BYTE COUNT
    MICPC=MICPC+1
    <MOVE!LDMA!IMM!<BC&377>>
    MEMINC BR,SELA!SP4           ;SAVE LOW BYTE
    MICPC=MICPC+1
    <MOVE!WRMEM!INCMAR!BR!<SELA!SP4>>
    MEMINC BR,SELA!SP5           ;AND NOW HIGH BYTE
    MICPC=MICPC+1
    <MOVE!WRMEM!INCMAR!BR!<SELA!SP5>>
RC5:    STATE  RCVD              ;SET NEXT STATE TO D
    MICPC=MICPC+1
    <MOVE!WRTBR!IMM!<RCVD-INIT&777/2>>
    ALWAYS REXIT
    MICPC=MICPC+1
    <JUMP!ALCOND!<REXIT-INIT&3000+4>!<REXIT-INIT&777/2>>
```

954
955
956 017506 000472
(1) 017506 000513
957 017510 000473
(1) 017510 063223
958 017512 000474
(1) 017512 023600
959 017514 000475
(1) 017514 060757
960 017516 000476
(1) 017516 107500
961 017520 000477
(1) 017520 100451
962 017522 000500
(1) 017522 060601
963 017524 000501
(1) 017524 103451
964 017526 000502
(1) 017526 060610
965 017530 000503
(1) 017530 001620
966 017532 000504
(1) 017532 103051
967 017534 000505
(1) 017534 010153
968 017536 000506
(1) 017536 062600
969 017540 000507
(1) 017540 010403
970
971 017542 000510
(1) 017542 002401
972 017544 000511
(1) 017544 063235
973 017546 000512
(1) 017546 100451

```
.SBTTL RCVD--ROUTINE TO HANDLE RESPONSE FIELD FOR NUMBERED MESSAGES
RCVD: STATE RCVE
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVE-INIT&777/2>>
RD2:  SP BR SELB,SP3 ;SAVE THE STATE
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP3>
      SPBR IBUS,RCVDAT,SPO ;INPUT THE CHARACTER
      MICPC=MICPC+1
      <MOVE!SPBRX!IBUS!RCVDAT!SPO>
      BRWRTE BR,SUB!SP17 ;COMPARE NEW R TO LAST R
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SUB!SP17>>
      BR7 10$ ;IF NEW IS GREATER---PROCESS
      MICPC=MICPC+1
      <JUMP!BR7CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
      ALWAYS IDLE
      MICPC=MICPC+1
      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
10$: BRWRTE BR,SELA!SP1 ;READ STATUS BYTE
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
      BR7 IDLE ;MAINT. MODE - GET OUT
      MICPC=MICPC+1
      <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      BRWRTE BR,SELA!SP10
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP10>>
      BRSHFT
      MICPC=MICPC+1
      <MOVE!SHFTBR!WRTEBR!SELB>
      BR4 IDLE
      MICPC=MICPC+1
      <JUMP!BR4CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      LDMA IMM,ISP17 ;ADDRESS LAST ACKED IMAGE
      MICPC=MICPC+1
      <MOVE!LDMA!IMM!<ISP17&377>>
      MEM BR,SELA!SPO ;COPY THE CHAR
      MICPC=MICPC+1
      <MOVE!WRMEM!BR!<SELA!SPO>>
RDS: BRWRTE IMM!LDMA,REPST ;SET UP COUNT FOR TIMER
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!LDMA!<REPST>>
      ;***DEPENDENT ON REPST = 2
      ;RESET REP THRESHOLD
      MEM IMM,1
      MICPC=MICPC+1
      <MOVE!WRMEM!IMM!<1>>
      SP BR,SELB,SP15 ;RESET THE COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP15>
      ALWAYS IDLE
      MICPC=MICPC+1
      <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
```

```

975 .SBTTL RCVE--ROUTINE TO HANDLE N FIELD OF NUMBERED MESSAGE
976
977 017550 000513
(1) 017550 060601
978 017552 000514
(1) 017552 107703
979 017554 000515
(1) 017554 020600
980 017556 000516
(1) 017556 060371
981 017560 000517
(1) 017560 105522
982 017562 000520
(1) 017562 063173
983 017564 000521
(1) 017564 104523
984 017566 000522
(1) 017566 063071
985 017570 000523
(1) 017570 000525
986 017572 000524
(1) 017572 103450

```

```

RCVE: BRWRTE BR SELA!SP1 ;READ THE STATUS BYTE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
BR7 RCVQ
MICPC=MICPC+1
<JUMP!BR7CON!<RCVQ-INIT&3000*4>!<RCVQ-INIT&777/2>>
BRWRTE IBUS,RCV DAT ;INPUT THE CHARACTER
MICPC=MICPC+1
<MOVE!WRTEBR!IBUS!<RCVDAT>>
CMP BR,SP11
MICPC=MICPC+1
<SUBTC!BR!SP11>
Z SS
MICPC=MICPC+1
<JUMP!ZCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
SP BR,DECA,SP12 ;FORCE MSG TYPE TO -1
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP13>
ALWAYS RE2
MICPC=MICPC+1
<JUMP!ALCOND!<RE2-INIT&3000*4>!<RE2-INIT&777/2>>
SS: SP BR,INCA,SP11 ;UPDATE R FIELD
MICPC=MICPC+1
<MOVE!SPX!BR!INCA!SP11>
RE2: STATE RCVF ;NEXT RECEIVE STATE IS F
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RCVF-INIT&777/2>>
ALWAYS REXIT
MICPC=MICPC+1
<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>

```


988			.SBTTL RCVF--ROUTINE TO IGNORE ADDRESS
989	017574	000525	RCVF: SP BR,DECA,SP4 ;DECREMENTLOW BYTE OF COUNT
(1)			MICPC=MICPC+1
(1)	017574	063164	<MOVE!SPX!BR!DECA!SP4>
990	017576		C RCVFO ;NO OVERFLOW
(1)			MICPC=MICPC+1
(1)	017576	105130	<JUMP!CCOND!<RCVFO-INIT\$3000*4>!<RCVFO-INIT\$777/2>>
991	017600		SP BR,DECA,SP5 ;OVERFLOW - DECREMENT HIGH BYTE
(1)			MICPC=MICPC+1
(1)	017600	063165	<MOVE!SPX!BR!DECA!SP5>
992	017602		RCVFO: STATE RCVG
(1)			MICPC=MICPC+1
(1)	017602	000530	<MOVE!WRTEBR!IMM!<RCVG-INIT\$777/2>>
993	017604	000533	RCVF1: NOP IBUS,RCV DAT,0 ;INPUT CHARACTER - AND DISCARD
(1)			MICPC=MICPC+1
(1)	017604	020200	<IBUS!RCV DAT!0>
994	017606		ALWAYS REXIT
(1)			MICPC=MICPC+1
(1)	017606	000532	<JUMP!ALCOND!<REXIT-INIT\$3000*4>!<REXIT-INIT\$777/2>>
(1)			100450
995			.SBTTL RCVG--ROUTINE TO IGNORE CRC1
996			.
997			.
998	017610		RCVG: STATE RCVH ;NEXT STATE IS RCVH
(1)			MICPC=MICPC+1
(1)	017610	000533	<MOVE!WRTEBR!IMM!<RCVH-INIT\$777/2>>
999	017612		ALWAYS RCVF1
(1)			MICPC=MICPC+1
(1)	017612	000534	<JUMP!ALCOND!<RCVF1-INIT\$3000*4>!<RCVF1-INIT\$777/2>>
(1)			104531

1001
 1002
 1003 017614
 1004 017614
 (1) 000535
 (1) 017614 023200
 1005 017616
 (1) 000536
 (1) 017616 020640
 1006 017620
 (1) 000537
 (1) 017620 116165
 1007 017622
 (1) 000540
 (1) 017622 060601
 1008 017624
 (1) 000541
 (1) 017624 107740
 1009 017626
 (1) 000542
 (1) 017626 060610
 1010 017630
 (1) 000543
 (1) 017630 001620
 1011 017632
 (1) 000544
 (1) 017632 117307
 1012 017634
 (1) 000545
 (1) 017634 010151
 1013 017636
 (1) 000546
 (1) 017636 016402
 1014 017640
 (1) 000547
 (1) 017640 016701
 1015 017642
 (1) 000550
 (1) 017642 062617
 1016 017644
 (1) 000551
 (1) 017644 010013
 1017 017646
 (1) 000552
 (1) 017646 043220
 1018 017650
 (1) 000553
 (1) 017650 062460
 1019 017652
 (1) 000554
 (1) 017652 010001
 1020 017654
 (1) 000555
 (1) 017654 040620
 1021 017656
 (1) 000556

```

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

```

(1)	017656	061620	<MOVE!SHFTBR!SELB!BR>	
1022	017660		MEM BR SELB	:UPDATE IT
(1)		000557	MICPC=MICPC+1	
(1)	017660	062620	<MOVE!WARMEM!BR!<SELB>>	
1023	017662		BRO NTHRES	:BRANCH IF THRESHOLD EXCEEDED
(1)		000560	MICPC=MICPC+1	
(1)	017662	116256	<JUMP!BROCON!<NTHRES-INIT&3000*4>!<NTHRES-INIT&777/2>>	
1024	017664		ALWAYS SNAK	
(1)		000561	MICPC=MICPC+1	
(1)	017664	114704	<JUMP!ALCOND!<SNAK-INIT&3000*4>!<SNAK-INIT&777/2>>	
1025	017666		BRWRT DP,<DECA!SP13>	:LOAD TYPE RECEIVED--DECREMENTING
(1)		000562	MICPC=MICPC+1	
(1)	017666	060573	<MOVE!WRTEBR!DP!<DECA!SP13>>	
1026	017670		Z RH1	:IF ALUOUT IS ALL ONES IS NUMBERED MSG
(1)		000563	MICPC=MICPC+1	
(1)	017670	115467	<JUMP!ZCOND!<RH1-INIT&3000*4>!<RH1-INIT&777/2>>	
1027	017672		RSTATE RCVA	
(1)		000564	MICPC=MICPC+1	
(1)	017672	000400	<MOVE!WRTEBR!IMM!<RCVA-INIT&777/2>>	
(1)		000565	MICPC=MICPC+1	
(1)	017674	063223	<MOVE!SPX!BR!SELB!SP3>	
1028	017676		BRWRT DP,<SELA!SP10>	:LOAD LINE STATUS WORD IN BR
(1)		000566	MICPC=MICPC+1	
(1)	017676	060610	<MOVE!WRTEBR!DP!<SELA!SP10>>	
1034	017700		OUTPUT IMM,<200!ORCVCO>	
(1)		000567	MICPC=MICPC+1	
(1)	017700	002212	<MOVE!WROUT!IMM!<200!ORCVCO>>	
1036	017702		BRSHFT	:SHIFT RIGHT
(1)		000570	MICPC=MICPC+1	
(1)	017702	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
1037	017704		BR4 10\$	
(1)		000571	MICPC=MICPC+1	
(1)	017704	107177	<JUMP!BR4CON!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>	
1038	017706		LDMA IMM, TYPTAB	:ADDRESS TYPE TABLE
(1)		000572	MICPC=MICPC+1	
(1)	017706	010162	<MOVE!LDMAR!IMM!<TYPTAB&377>>	
1039	017710		CMP <MEMX!INCMAR>,SP13	
(1)		000573	MICPC=MICPC+1	
(1)	017710	054373	<SUBTC!MEMX!INCMAR!SP13>	
1040	017712		Z REP	
(1)		000574	MICPC=MICPC+1	
(1)	017712	115411	<JUMP!ZCOND!<REP-INIT&3000*4>!<REP-INIT&777/2>>	
1041	017714		CMP <MEMX!INCMAR>,SP13	
(1)		000575	MICPC=MICPC+1	
(1)	017714	054373	<SUBTC!MEMX!INCMAR!SP13>	
1042	017716		Z NAK	
(1)		000576	MICPC=MICPC+1	
(1)	017716	115445	<JUMP!ZCOND!<NAK-INIT&3000*4>!<NAK-INIT&777/2>>	
1043	017720		LDMA IMM, TYPSTT	:SET POINTER TO START TYPE
(1)		000577	MICPC=MICPC+1	
(1)	017720	010164	<MOVE!LDMAR!IMM!<TYPSTT&377>>	
1044	017722		CMP <MEMX!INCMAR>,SP13	
(1)		000600	MICPC=MICPC+1	
(1)	017722	054373	<SUBTC!MEMX!INCMAR!SP13>	
1045	017724		Z START	
(1)		000601	MICPC=MICPC+1	

1065
 1066
 1067 017740 000607
 (1) 017740 123600
 1068 017742 000610
 (1) 017742 102051
 1069 017744 000611
 (1) 017744 000600
 1071 017746 000612
 (1) 017746 063300
 1076 017750 000613
 (1) 017750 000653
 1077 017752 000614
 (1) 017752 104620
 1078
 1079 017754 000615
 (1) 017754 123600
 1081 017756 000616
 (1) 017756 106247
 1086 017760 000617
 (1) 017760 000625
 1087 017762 000620
 (1) 017762 063223
 1088 017764 000621
 (1) 017764 022203
 1089 017766 000622
 (1) 017766 000421
 1093 017770 000623
 (1) 017770 061310
 1094 017772 000624
 (1) 017772 100451

```

;*****TIME CRITICAL CODE-- CHANGE WITH GREAT CARE*****
.SBTTL RCVK01--ROUTINE TO HANDLE FIRST BYTE ODD RECEIVE
RCVK01: SPBR IBUS,NPR,SPO ;READ NPR REGISTER
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!NPR!SPO>
        BR0 IDLE
        MICPC=MICPC+1
        <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
        BRWRTE IMM,200 ;MASK FOR CO(BYTE TRANSFER)
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<200>>
        SP BR,AORB,SPO
        MICPC=MICPC+1
        <MOVE!SPX!BR!AORB!SPO>
        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
RCVK0: SPBR IBUS,NPR,SPO ;IS AN NPR GOING
        MICPC=MICPC+1
        <MOVE!SPBRX!IBUS!NPR!SPO>
        BR0 RK66 ;IF SO, REITERATE ODD AND EXIT
        MICPC=MICPC+1
        <JUMP!BROCON!<RK66-INIT&3000*4>!<RK66-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>>
RK8: BRWRTE IMM,21 ;SET OUT NPR (C1) AND NPR REQ
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<21>>
RK7: OUT BR,<AORB!ONPR> ;WRITE NPR REGISTER
        MICPC=MICPC+1
        <MOVE!WROUTX!BR!<AORB!ONPR>>
        ALWAYS IDLE
        MICPC=MICPC+1
        <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>

```

1096
1097 017774
(1) 000625
(1) 017774 120600
1099 017776
(1) 000626
(1) 017776 107251
1104 020000
(1) 000627
(1) 020000 023140
1105 020002
(1) 000630
(1) 020002 062066
1106 020004
(1) 000631
(1) 020004 063164
1107 020006
(1) 000632
(1) 020006 105235
1108 020010
(1) 000633
(1) 020010 063165
1109 020012
(1) 000634
(1) 020012 105711
1110 020014
(1) 000635
(1) 020014 022202
1111 020016
(1) 000636
(1) 020016 023140
1112 020020
(1) 000637
(1) 020020 062066
1113 020022
(1) 000640
(1) 020022 115035
1114 020024
(1) 000641
(1) 020024 063164
1115 020026
(1) 000642
(1) 020026 105245
1116 020030
(1) 000643
(1) 020030 063165
1117 020032
(1) 000644
(1) 020032 111772
1119 020034
(1) 000645
(1) 020034 020640
1120 020036
(1) 000646
(1) 020036 107215
1121 020040

```

RCVKE: SBRTL RCVKE--HANDLE EVEN BYTES
        BRWRT IBUS,NPR ;READ NPR CONTROL REGISTER
        MICPC=MICPC+1
        <MOVE!WRTBR!IBUS!<NPR>>
        BR4 RK4 ;IF RECV NPR--BRANCH
        MICPC=MICPC+1
        <JUMP!BR4CON!<RK4-INIT&3000*4>!<RK4-INIT&777/2>>
RK5: SP IBUS IOBA1,SP0 ;READ LOW BYTE OF BA TO SP
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IOBA1!SP0>
      OUTPUT DP,<INCA!OBA1> ;WRITE INCREMENTED BA
      MICPC=MICPC+1
      <MOVE!WROUT!DP!<INCA!OBA1>>
RK50: SP BR,DECA,SP4 ;DECREMENT CHARACTER COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP4>
      C 10$ ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
      SP BR,DECA,SP5 ;OVERFLOW - DECREMENT HIGH BYTE
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP5>
      Z RL3 ;BYTE COUNT ZERO
      MICPC=MICPC+1
      <JUMP!ZCOND!<RL3-INIT&3000*4>!<RL3-INIT&777/2>>
10$: OUTPUT IBUS,<RCVDAT!OUTDA1> ;READ CHARACTER AND WRITE IT
      MICPC=MICPC+1
      <MOVE!WROUT!IBUS!<RCVDAT!OUTDA1>>
      SP IBUS IOBA1,SP0 ;READ INCREMENTED BA
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IOBA1!SP0>
      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>>
RK3: SP BR,DECA,SP4 ;DECREMENT THE COUNT OF BYTES
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP4>
      C RK6 ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<RK6-INIT&3000*4>!<RK6-INIT&777/2>>
      SP BR,DECA,SP5 ;DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP5>
      Z RL4 ;BYTE COUNT ZERO
      MICPC=MICPC+1
      <JUMP!ZCOND!<RL4-INIT&3000*4>!<RL4-INIT&777/2>>
RK6: BRWRT IBUS,RCVCON ;READ RECEIVER CONTROL REGISTER
      MICPC=MICPC+1
      <MOVE!WRTBR!IBUS!<RCVCON>>
      BR4 RCVK0 ;IF ANOTHER CHARACTER--PROCESS
      MICPC=MICPC+1
      <JUMP!BR4CON!<RCVK0-INIT&3000*4>!<RCVK0-INIT&777/2>>
RK66: STATE RCVK0
    
```

(1) 020040 000615
 1122 020042 000650
 (1) 020042 100450
 1123 020044 000651
 (1) 020044 102051
 1124 020046 000652
 (1) 020046 104627
 1130
 1131 020050 000653
 (1) 020050 123200
 1133 020052 000654
 (1) 020052 102051
 1135 020054 000655
 (1) 020054 000577
 1136 020056 000656
 (1) 020056 061270
 1137 020060 000657
 (1) 020060 104631
 1138
 1139
 1140 020062 000660
 (1) 020062 023200
 1141 020064 000661
 (1) 020064 062202
 1142 020066 000662
 (1) 020066 060601
 1143 020070 000663
 (1) 020070 117575
 1144 020072 000664
 (1) 020072 104641

```

MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<RCVKE-INIT&777/2>>
ALWAYS REXIT
MICPC=MICPC+1
<JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
RK4: BR0 IDLE
MICPC=MICPC+1
<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
ALWAYS RKS ;IF NO NPR --PROCESS
MICPC=MICPC+1
<JUMP!ALCOND!<RKS-INIT&3000*4>!<RKS-INIT&777/2>>
RKE1: SP IBUS,NPR,SPO ;READ NPR REGISTER
MICPC=MICPC+1
<MOVE!SPX!IBUS!NPR!SPO>
BR0 IDLE ;NPR STILL IN PROGRESS
MICPC=MICPC+1
<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
BRWTE IMM,177 ;MASK FOR ALL BUT CO
MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<177>>
OUT BR,<AANDB!ONPR> ;TURN OFF ALL BUT CO
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AANDB!ONPR>>
ALWAYS RK50
MICPC=MICPC+1
<JUMP!ALCOND!<RK50-INIT&3000*4>!<RK50-INIT&777/2>>
;*****END OF TIME CRITICAL PATH*****
RCVKE0: SP IBUS,RCVDAT,SPO ;READ CHARACTER AND SAVE IN SPO
MICPC=MICPC+1
<MOVE!SPX!IBUS!RCVDAT!SPO>
OUTPUT BR,<SELA!OUTDA1> ;SEND NONSENSE CHARACTER
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELA!OUTDA1>>
BRWTE BR,SELA!SP1 ;READ STATUS BYTE
MICPC=MICPC+1
<MOVE!WRITEBR!BR!<SELA!SP1>>
BR7 PASWRD ;MAINT MODE - SEE IF RLD MESSAGE
MICPC=MICPC+1
<JUMP!BR7CON!<PASWRD-INIT&3000*4>!<PASWRD-INIT&777/2>>
ALWAYS RK3 ;OTHERWISE PROCESS NORMALLY
MICPC=MICPC+1
<JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>

```

1146		
1147	020074	000665
(1)		023213
(1)	020074	
1148	020076	000666
(1)		000670
(1)	020076	
1149	020100	000667
(1)		100450
(1)	020100	
1150		

```

RCVI: .SBTTL RCVI--STORE UNNUMBERED MESSAGE TYPE
      SP      IBUS,RCVDAT,SP13      ;STORE UNNUMBERED TYPE
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!RCVDAT!SP13>
      STATE  RCVJ                      ;NEXT STATE IS J
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVJ-INIT&777/2>>
      ALWAYS REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>! REXIT-INIT&777/2>
      ;

```


1152
1153 020102
1158 020102
 (1) 000670
 (1) 020102 023205
1159 020104
 (1) 000671
 (1) 020104 000600
1160 020106
 (1) 000672
 (1) 020106 060665
1161 020110
 (1) 000673
 (1) 020110 063310
1163 020112
 (1) 000674
 (1) 020112 000676
1164 020114
 (1) 000675
 (1) 020114 100450

```

.SBTTL RCVJ--ROUTINE TO HANDLE SUBTYPE FIELD, SELECT AND FINAL
RCVJ:
    SP      IBUS, RCVDAT, SPS          ;GET CHARACTER
    MICPC=MICPC+1
    <MOVE!SPX!IBUS!RCVDAT!SPS>
    BRWATE IMM, 200                    ;CONDITIONALLY SET BIT
    MICPC=MICPC+1
    <MOVE!WATEBR!IMM!<200>>
    BRWATE BR, AANDB!SPS
    MICPC=MICPC+1
    <MOVE!WATEBR!BR!<AANDB!SPS>>
    SP      BR, AORB, SP10
    MICPC=MICPC+1
    <MOVE!SPX!BR!AORB!SP10>
    STATE  RCVR                          ;NEXT STATE IS N
    MICPC=MICPC+1
    <MOVE!WATEBR!IMM!<RCVR-INIT&777/2>>
    ALWAYS REXIT
    MICPC=MICPC+1
    JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>
    
```

1166			.SBTTL RCVR--UNNUMBERED MESSAGE RESPONSE FIELD
1167			;ENTERED FROM IDLE LOOP
1168			:
1169	020116		RCVR: BRWRTE IMM,3 ;REP MESSAGE TYPE TO BR
(1)		000676	MICPC=MICPC+1
(1)	020116	000403	<MOVE!WRTEBR!IMM!<3>>
1170	020120		NOP BR,SUB,SP13 ;IS TYPE ACK OR NAK
(1)		000677	MICPC=MICPC+1
(1)	020120	060353	<BR!SUB!SP13>
1171	020122		STATE RCVQ ;NEXT STATE IS RCVQ
(1)		000700	MICPC=MICPC+1
(1)	020122	000703	<MOVE!WRTEBR!IMM!<RCVQ-INIT&777/2>>
1172			;***NOTE THIS INSTR DOES NOT CLOCK "C"
1173	020124		C RCVF1 ;IF NOT IGNORE
(1)		000701	MICPC=MICPC+1
(1)	020124	105131	<JUMP!CCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2>>
1174	020126		ALWAYS RD2 ;DO RANGE CHECKS
(1)		000702	MICPC=MICPC+1
(1)	020126	104473	JUMP!ALCOND!<RD2-INIT&3000*4>!<RD2-INIT&777/2>>

E16

DMC11 DDCMF PROTOCOL IMPLEMENTATION
DDCHGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 6-41
RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD

PAGE: 0199

1176		
1177		
1178		
1179	020130	000703
(1)		
(1)	020130	000525
1180	020132	000704
(1)		
(1)	020132	104531

```

.SBTTL RCVQ--UNNUMBERED MESSAGE--NUMBER FIELD
;ENTER FROM IDLE
RCVQ: STATE RCVF ;NEXT STATE IS ADDRESS
MICPC=MICPC+1
<MOVE!WRITEBR!IMM!<RCVF-INIT&777/2>>
ALWAYS RCVF1
MICPC=MICPC+1
<JUMP!ALCOND!<RCVF1-INIT&3000*4>!<RCVF1-INIT&777/2 /

```

1182
1183
1184 020134 000705
(1) 020134 123600
1186 020136 000706
(1) 020136 107314
1191 020140 000707
(1) 020140 000576
1192 020142 000710
(1) 020142 061270
1193
1194 020144 000711
(1) 020144 020200
1195 020146 000712
(1) 020146 000716
1196 020150 000713
(1) 020150 100450
1197
1199 020152 000714
(1) 020152 102051
1200 020154 000715
(1) 020154 104707
1202

```

:SBTTL RCVL--PROCESS CRC3
:ENTERED FROM IDLE LOOP
RCVL: SPBR IBUS,NPR,SPO ;READ NPR CONTROL
      MICPC=MICPC+1
      <MOVE!SPBRX!IBUS!NPR!SPO>
      BR4 RL1 ;RCV NPR BRANCH
      MICPC=MICPC+1
      <JUMP!BR4CON!<RL1-INIT&3000*4>!<RL1-INIT&777/2>>
RC2: BRWRT IMM,176 ;MASK TO TURN OFF CO
      MICPC=MICPC+1
      <MOVE!WRTBR!IMM!<176>>
      OUT BR,ANDB!ONPR
      MICPC=MICPC+1
      <MOVE!WROUTX!BR!<ANDB!ONPR>>
      ;
RC3: NOP IBUS,RCV DAT,0 ;INPUT CHARACTER AND DISCARD
      MICPC=MICPC+1
      <IBUS!RCV DAT!0>
      STATE RCVM
      MICPC=MICPC+1
      <MOVE!WRTBR!IMM!<RCVM-INIT&777/2>>
      ALWAYS REXIT
      MICPC=MICPC+1
      <JUMP!ALCOND!<REXIT-INIT&3000*4>!<REXIT-INIT&777/2>>
      ;
RL1: BRO IDLE ;NPR GOING --GET OUT
      MICPC=MICPC+1
      <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
      ALWAYS RL2
      MICPC=MICPC+1
      <JUMP!ALCOND!<RL2-INIT&3000*4>!<RL2-INIT&777/2>>
      ;

```

```

1204 .SBTTL RCVM--PROCESS CRC4--END OF DATA MESSAGE
1205 ;ENTERED FROM IDLE LOOP
1206 ;IF CRC CORRECT -- QUEUE INTERRUPT AND UPDATE RESPONSE
1207 ;
1208 ;IF CRC WRONG SEND NAK
RCVM: BRWRT IBUS,UBBR ;READ UNIBUS BR REGISTER
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<UBBR>>
1209 020156 000716
      BPD NXMERR ;NON-EXISTANT MEMORY
      MICPC=MICPC+1
      <JUMP!BROCON!<NXMERR-INIT&3000*4>!<NXMERR-INIT&777/2>>
1210 020160 120620
      SP IBUS,RCVDAT,SPO ;READ CRC CHARACTER
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!RCVDAT!SPO>
1211 020162 000720
      BRWRT IBUS,RCVCON ;READ RECEIVER CONTROL REGISTER
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<RCVCON>>
1212 020164 023200
      BRG RCVMI ;IF CRC GOOD -- PROCESS
      MICPC=MICPC+1
      <JUMP!BROCON!<RCVMI-INIT&3000*4>!<RCVMI-INIT&777/2>>
1213 020166 000721
      BRWRT BR,SELA!SP1 ;READ STATUS BYTE
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
1214 020170 060601
      BR7 RHX ;CRC ERROR IN BOOT MODE - FLUSH
      MICPC=MICPC+1
      <JUMP!BR7CON!<RHX-INIT&3000*4>!<RHX-INIT&777/2>>
1215 020172 000724
      LDMA IMM,T ;ELSE SEND NAK --DATA ERROR
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<T&377>>
1216 020174 010151
      MEMINC IMM,2 ;NAK TYPE
      MICPC=MICPC+1
      <MOVE!WRMEM!INCMAR!IMM!<2>>
1217 020176 000726
      MEMINC IMM,302 ;DATA ERROR SUBTYPE
      MICPC=MICPC+1
      <MOVE!WRMEM!INCMAR!IMM!<302>>
1218 020200 016402
      LDMA IMM,NDATS
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<NDATS&377>>
1219 020202 010014
      ALWAYS RHS ;SEND NAK
      MICPC=MICPC+1
      <JUMP!ALCOND!<RHS-INIT&3000*4>!<RHS-INIT&777/2>>
1220 020204 104552
RCVM0: LDMA IMM,<<RTHRS+3>> ;POINT TO ERROR WORD
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<<RTHRS+3>&377>>
1221 020206 000732
      BRWRT IMM,10 ;MAINT MESSAGE ERROR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<10>>
1222 020208 010177
      ALWAYS RCEXY ;GIVE FATAL ERROR
      MICPC=MICPC+1
      <JUMP!ALCOND!<RCEXY-INIT&3000*4>!<RCEXY-INIT&777/2>>
1223 020210 000733
1224 020212 000410
      000734
      114522

```

```

1226      .SBTTL EM2--PROCESS RLD MESSAGE
1227      :ENTERED FROM IDLE LOOP
1228      ;IF RLD PASSWORD CHECKS TRIGGER THE BOOT ROM
1229
1230 EM2:   BRWRT  IBUS,RCV DAT          ;READ THE CHAR
          MICPC=MICPC+1
          <MOVE!WRTEBR!IBUS!<RCV DAT>>
1231      CMP     BR,SP13              ;IS IT A MATCH
          MICPC=MICPC+1
          <SUBTC!BR!SP13>
1232      Z      EM3
          MICPC=MICPC+1
          <JUMP!ZCOND!<EM3-INIT&3000*4>!<EM3-INIT&777/2>>
          ;FALL INTO RHX
1233      RHX:  BRWRT  BR,AA!SP1        ;READ STATUS BYTE SHIFTED LEFT
          MICPC=MICPC+1
          <MOVE!WRTEBR!BR!<AA!SP1>>
1234      BR4    10$                  ;DLE RECEIVED IN NORMAL MODE
          MICPC=MICPC+1
          <JUMP!BR4CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
1235      ALWAYS FLUSH                ;ALREADY IN MAINT MODE
          MICPC=MICPC+1
          <JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
1236      10$:  BRWRT  IMM,163         ;MASK TO CLEAR ALL MAINT RELATED BITS
          MICPC=MICPC+1
          <MOVE!WRTEBR!IMM!<163>>
1237      SP     BR,AANDB,SP1         ;CLEAR THEM
          MICPC=MICPC+1
          <MOVE!SPX!BR!AANDB!SP1>
1238      ALWAYS FLUSH
          MICPC=MICPC+1
          <JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>
1239      EM3:  SP     BR,DECA,SP4     ;DECREMENT CHARACTER COUNT BY ONE
          MICPC=MICPC+1
          <MOVE!SPX!BR!DECA!SP4>
1240      Z      EMTRIG                ;TRIGGER AC LOW
          MICPC=MICPC+1
          <JUMP!ZCOND!<EMTRIG-INIT&3000*4>!<EMTRIG-INIT&777/2>>
1241      ALWAYS IDLE
          MICPC=MICPC+1
          <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1242
1243
1244

```

```

1246
1247 020244 000751
      (1) 010177
1248 020246 000752
      (1) 016401
1249 020250 000753
      (1) 002400
1250 020252 000754
      (1) 043230
1251 020254 000755
      (1) 114524

```

```

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

```

1273			
1274	020256	000756	BOOT: BRWRT BR SELA!SP1 ;SEE IF IN MAINT. MODE
(1)		060601	MICPC=MICPC+1
(1)	020256		<MOVE!WRTEBR!BR!<SELA!SP1>>
1275	020260	000757	BR7 RA3 ;BRANCH IF SO AND TREAT DLE LIKE NUM. MSG.
(1)		103742	MICPC=MICPC+1
(1)	020260		<JUMP!BR7CON!<RA3-INIT&3000*4>!<RA3-INIT&777/2>>
1276	020262	000760	BRWRT IMM 210 ;MASK TO SET MAINT MODE AND DLE RECV'D
(1)		000610	MICPC=MICPC+1
(1)	020262		<MOVE!WRTEBR!IMM!<210>>
1277	020264	000761	SP BR AORB,SP1 ;SET THE BITS
(1)		063301	MICPC=MICPC+1
(1)	020264		<MOVE!SPX!BR!AORB!SP1>
1278	020266	000762	ALWAYS RA3 ;TREAT LIKE NUMBERED MESSAGE
(1)		100742	MICPC=MICPC+1
(1)	020266		<JUMP!ALCOND!<RA3-INIT&3000*4>!<RA3-INIT&777/2>>
1279	020270	000763	RESEXT: BRWRT IMM 4 ;ADD TO MXT BITS
(1)		000404	MICPC=MICPC+1
(1)	020270		<MOVE!WRTEBR!IMM!<4>>
1280	020272	000764	SP BR ADD,SPO
(1)		063000	MICPC=MICPC+1
(1)	020272		<MOVE!SPX!BR!ADD!SPO>
1281	020274	000765	ALWAYS TH3X
(1)		110601	MICPC=MICPC+1
(1)	020274		<JUMP!ALCOND!<TH3X-INIT&3000*4>!<TH3X-INIT&777/2>>
1282	020276	000766	TABMXT: BRWRT IMM 4 ;INCREMENT MXT
(1)		000404	MICPC=MICPC+1
(1)	020276		<MOVE!WRTEBR!IMM!<4>>
1283	020300	000767	SP IBUS,UBBR,SPO ;READ BR CONTROL
(1)		123220	MICPC=MICPC+1
(1)	020300		<MOVE!SPX!IBUS!UBBR!SPO>
1284	020302	000770	OUT BR,ADD!OBR
(1)		061011	MICPC=MICPC+1
(1)	020302		<MOVE!WRQUTX!BR!<ADD!OBR>>
1285	020304	000771	ALWAYS ECX
(1)		114761	MICPC=MICPC+1
(1)	020304		<JUMP!ALCOND!<ECX-INIT&3000*4>!<ECX-INIT&777/2>>
1286			
1287	020306	000772	ATHRES: BRWRT IMM 2
(1)		000402	MICPC=MICPC+1
(1)	020306		<MOVE!WRTEBR!IMM!<2>>
1288	020310	000773	ALWAYS ERRXX
(1)		114663	MICPC=MICPC+1
(1)	020310		<JUMP!ALCOND!<ERRXX-INIT&3000*4>!<ERRXX-INIT&777/2>>
1291	020312	000774	\$ZERO
(2)		000000	MICPC=MICPC+1
(2)	020312		000000
(1)	020314		\$ZERO
(2)		000775	MICPC=MICPC+1
(2)	020314		000000
(1)	020316		\$ZERO
(2)		000776	MICPC=MICPC+1
(2)	020316		000000
(1)	020320		\$ZERO
(2)		000777	MICPC=MICPC+1
(2)	020320		000000

1293		020322
1294		000777
1295		
1296		
1297	020322	
(1)		001000
(1)	020322	020620
1298	020324	
(1)		001001
(1)	020324	173202
1299	020326	
(1)		001002
(1)	020326	100454

```

.=INIT+2000
MICPC=777
.SBTTL TMTDA--TRANSMITTER DISPATCH ROUTINE
TMTDA: BRWTE IBUS,TMTCON ;READ TRANSMITTER CONTROL REGISTER
MICPC=MICPC+1
<MOVE!WRTBR!IBUS!<TMTCON>>
.BR4 DP,SELA,<2!PAGE2> ;IF READY PROCEED
MICPC=MICPC+1
<JUMP! BR4CON!DP!SELA!2!PAGE2>
ALWAYS I1 ;ELSE IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<I1-INIT$3000*4>!<I1-INIT&777/2>>

```

1301
1302
1303 020330
1309 020330
 (1) 001003
 (1) 020330 060610
1310 020332
 (1) 001004
 (1) 020332 112007
1311 020334
 (1) 001005
 (1) 020334 001620
1312 020336
 (1) 001006
 (1) 020336 103063
1318 020340
 (1) 001007
 (1) 020340 060610
1319 020342
 (1) 001010
 (1) 020342 113412
1320 020344
 (1) 001011
 (1) 020344 100454
1321 020346
 (1) 001012
 (1) 020346 020660
1322 020350
 (1) 001013
 (1) 020350 001620
1323 020352
 (1) 001014
 (1) 020352 103054
1324 020354
 (1) 001015
 (1) 020354 000773
1325 020356
 (1) 001016
 (1) 020356 063270
1326 020360
 (1) 001017
 (1) 020360 000424
 (1) 001020
 (1) 020362 063222
1327 020364
 (1) 001021
 (1) 020364 000412
1328 020366
 (1) 001022
 (1) 020366 063226
1329 020370
 (1) 001023
 (1) 020370 100454
1330 020372
 (1) 001024
 (1) 020372 063166

```

.SBTTL TMTA--FIRST CHARACTER OF HEADER
;
TMTA:
BRWRT  BR, SELA!SP10                ;REREAD STATUS
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP10>>
BRO    NUMSYN                        ;IF UNNUMBPENDING -- SEND IT
MICPC=MICPC+1
<JUMP!BROCON!<NUMSYN-INIT&3000*4>!<NUMSYN-INIT&777/2>>
BRSHFT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR4    IDLED                          ;IF START MODE--EXIT
MICPC=MICPC+1
<JUMP!BR4CON!<IDLED-INIT&3000*4>!<IDLED-INIT&777/2>>
NUMSYN: BRWRT  BR, <SELA!SP10>        ;READ LINE STATUS WORD
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP10>>
BR7    SS                              ;IF OK TO SEND--PROCEED
MICPC=MICPC+1
<JUMP!BR7CON!<SS-INIT&3000*4>!<SS-INIT&777/2>>
ALWAYS I1                            ;ELSE--IDLE
MICPC=MICPC+1
<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
SS:   BRWRT  IBUS, MODEM              ;ARE WE STILL SENDING?
MICPC=MICPC+1
<MOVE!WRTEBR!IBUS!<MODEM>>
BRSHFT
MICPC=MICPC+1
<MOVE!SHFTBR!WRTEBR!SELB>
BR4    I1                              ;RTS SET? IF SO WE ARE--STALL
MICPC=MICPC+1
<JUMP!BR4CON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
BRWRT  IMM, 373                       ;MASK TO TURN OFFLINE IDLE
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<373>>
SP     BR, AANDB, SP10                ;IN LINE STATUS WORD
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SP10>
TSTATE TMTA1
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTA1-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
BRWRT  IMM, 12
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<12>>
SP     BR, SELB, SP6                  ;STORE IN SP6
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP6>
ALWAYS I1                            ;BACK TO IDLE LOOP
MICPC=MICPC+1
<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
TMTA1: SP     BR, DECA, SP6            ;DECREMENT SYN COUNT
MICPC=MICPC+1
<MOVE!SPX!BR!DECA!SP6>

```

1331 020374
(1) 001025
(1) 020374 111432
1332 020376
(1) 001026
(1) 020376 002011
1333 020400
(1) 001027
(1) 020400 000626
1334 020402
(1) 001030
(1) 020402 06223C
1335 020404
(1) 001031
(1) 020404 100454
1337 020406
(1) 001032
(1) 020406 060610
1338 020410
(1) 001033
(1) 020410 112043
1339 020412
(1) 001034
(1) 020412 000451
(1) 001035
(1) 020414 063222
1340 020416
(1) 001036
(1) 020416 060601
1341 020420
(1) 001037
(1) 020420 113447
1342 020422
(1) 001040
(1) 020422 000601
1343 020424
(1) 001041
(1) 020424 062230
1344 020426
(1) 001042
(1) 020426 100454
1345 020430
(1) 001043
(1) 020430 000610
(1) 001044
(1) 020432 063222
1346 020434
(1) 001045
(1) 020434 000405
1347 020436
(1) 001046
(1) 020436 110441
1348 020440
(1) 001047
(1) 020440 000620
1349 020442

```
Z      TMTXT
MICPC=MICPC+1
<JUMP!ZCOND!<TMTXT-INIT&3000*4>!<TMTXT-INIT&777/2>>
OUTPUT IMM,<1!OTMTCO>          ;WRITE SOM TO TMTR CONTRL
MICPC=MICPC+1
<MOVE!WROUT!IMM!<1!OTMTCO>>
BRWRTE IMM,226                ;SYNC CHAR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<226>>
OUTPUT BR,<SELB!TMTDAT>       ;SEND THE CHARACTER
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELB!TMTDAT>>
ALWAYS I!
MICPC=MICPC+1
<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
TMTXT: BRWRTE BR,<SELA!SP10>   ;UNNUMB MESSAGE?
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP10>>
BR0      TMTUN                ;IF SO --BRANCH
MICPC=MICPC+1
<JUMP!BROCON!<TMTUN-INIT&3000*4>!<TMTUN-INIT&777/2>>
TSTATE TMTB
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<TMTB-INIT&777/2>>
MICPC=MICPC+1
<MOVE!SPX!BR!SELB!SP2>
BRWRTE BR,SELA!SP1           ;ARE WE IN BOOT MODE
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<SELA!SP1>>
BR7      TMTBT                ;IF SO SEND DLE
MICPC=MICPC+1
<JUMP!BR7CON!<TMTBT-INIT&3000*4>!<TMTBT-INIT&777/2>>
BRWRTE IMM,201              ;ELSE STORE SCH
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<201>>
TMTAS: OUTPUT BR,<SELB!TMTDAT> ;IN TMT SILO
MICPC=MICPC+1
<MOVE!WROUT!BR!<SELB!TMTDAT>>
ALWAYS I!
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>
BRWRTE IMM,5                ;ENQ 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: BRWRTE IMM,220        ;WRITE A DLE TO BR
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<220>>
ALWAYS TMTAS                ;SEND IT
```

B01

00000: DDMP PROTOCOL IMPLEMENTATION
00000-00000 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 6-50
TMTA--FIRST CHARACTER OF HEADER

PAGE: 0208

: 00:050
: 02042 110441

MICPC=MICPC+1
(JUMP!ALCOND!(TMTAS-INIT&3000*4)!(TMTAS-INIT&777/2),

```

:373 .SBTTL TMTB--OUTPUT FIRST CHAR OF COUNT
:374 .
:375 020444 001051 TMTB: LDMA BR SELA!SP16 ;GETPOINTER TO NEXT TMT LINK
:376 020444 070216 MICPC=MICPC+1
:376 020446 001052 <MOVE!LDMAR!BR!<SELA!SP16>>
:377 020446 016403 MEMINC IMM,3 ;WRITE MSG TMTD TO FLAGS
:378 020450 001053 MICPC=MICPC+1
:378 020452 076612 <MOVE!WRMEM!INCMAR!IMM!<3>>
:379 020452 001054 MEMINC BR SELA!SP12 ;PICK UP MSGNO
:380 020454 000476 MICPC=MICPC+1
:380 020454 063222 <MOVE!WRMEM!INCMAR!BR!<SELA!SP12>> ;ADDRESS TMTR STATE
:381 020456 001055 STATE TMTC
:381 020456 056224 MICPC=MICPC+1
:382 020460 001057 <MOVE!WRTEBR!IMM!<TMTC-INIT&777/2>> ;UPDATE IT
:383 020462 043227 SP BR SELB,SP2
:384 020464 001061 MICPC=MICPC+1
:384 020464 123200 <MOVE!SPX!BR!SELB!SP2>
:385 020466 001062 OUTPUT <MEMX!INCMAR>,SELB!IBA1 ;WRITE LOW BYTE OF ADDRESS
:386 020470 001063 MICPC=MICPC+1
:386 020470 063260 <MOVE!WROUT!MEMX!INCMAR!<SELB!IBA1>>
:387 020472 001064 OUTPUT <MEMX!INCMAR>,SELB!IBA2 ;WRITE HIGH BYTE OF ADDRESS
:388 020472 003306 MICPC=MICPC+1
:388 020472 054666 <MOVE!WROUT!MEMX!INCMAR!<SELB!IBA2>>
:389 020474 001066 SP MEMX SELB,SP7 ;HIGH BYTE OF COUNT TO SP7
:390 020500 001067 MICPC=MICPC+1
:390 020500 001620 <MOVE!SPX!MEMX!SELB!SP7> ;WAIT TO MASK OFF MEM EXT. BITS
:391 020502 001070 SP IBUS,NPR,SP0
:391 020502 001620 MICPC=MICPC+1
:392 020504 001071 <MOVE!SPX!IBUS!NPR!SP0>
:392 020504 001620 BRWRT IMM,220
:393 020506 001072 MICPC=MICPC+1
:393 020506 001072 <MOVE!WRTEBR!IMM!<220>>
:394 020506 001072 SP BR AANDB,SP0
:394 020506 001072 MICPC=MICPC+1
:395 020506 001072 <MOVE!SPX!BR!AANDB!SP0>
:396 020506 001072 SP IMM,300,SP6 ;MASK FOR MXT
:397 020506 001072 MICPC=MICPC+1
:397 020506 001072 <MOVE!SPX!IMM!300!SP6>
:398 020506 001072 BRWRT MEMX!INCMAR,AANDB!SP6 ;TURN OFF CC2
:399 020506 001072 MICPC=MICPC+1
:399 020506 001072 <MOVE!WRTEBR!MEMX!INCMAR!<AANDB!SP6>>
:400 020506 001072 OUTPUT MEMX SELB!TMTDAT ;ALSO WRITE COUNT TO TMTR SILC
:401 020506 001072 MICPC=MICPC+1
:402 020506 001072 <MOVE!WROUT!MEMX!<SELB!TMTDAT>>
:403 020506 001072 BRSHFT ;SHIFT BITS INTO CORRECT POSITION
:404 020506 001072 MICPC=MICPC+1
:405 020506 001072 <MOVE!SHFTBR!WRTEBR!SELB>
:406 020506 001072 BRSHFT
:407 020506 001072 MICPC=MICPC+1
:408 020506 001072 <MOVE!SHFTBR!WRTEBR!SELB>
:409 020506 001072 BRSHFT
:410 020506 001072 MICPC=MICPC+1
:411 020506 001072 <MOVE!SHFTBR!WRTEBR!SELB>
:412 020506 001072 BRSHFT
:413 020506 001072 MICPC=MICPC+1

```

DO1

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCHGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 9-1
TMTB--OUTPUT FIRST CHAR OF COUNT

PAGE: 0210

(1)	020506	001620	<MOVE!SHFTBR!WRTEBR!SELB>	
1394	020510		OUT BR,AORB!ONPR	
(1)		001073	MICPC=MICPC+1	
(1)	020510	061310	<MOVE!WROUTX!BR!<AORB!ONPR>>	
1395	020512		SPBR MEMX,SELB,SP6	:LOWBYTE OF COUNT TO SP6
(1)		001074	MICPC=MICPC+1	
(1)	020512	043626	<MOVE!SPBRX!MEMX!SELB!SP6>	
1400	020514		ALWAYS I1	
(1)		001075	MICPC=MICPC+1	
(1)	020514	100454	<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>	
1402			:	

E01

1404
 1405
 1406 020516 001076
 (1) 020516 000477
 1407 020520 001077
 (1) 020520 063667
 1408 020522 001100
 (1) 020522 062230
 1409 020524 001101
 (1) 020524 000543
 1410 020526 001102
 (1) 020526 060376
 1411 020530 001103
 (1) 020530 111511
 1412 020532 001104
 (1) 020532 000406
 1413 020534 001105
 (1) 020534 063016
 1414 020536
 1420 020536 001106
 (1) 020536 000514
 (1) 001107
 (1) 020540 063222
 1421 020542 001110
 (1) 020542 100454
 1423 020544 001111
 (1) 020544 000471
 1424 020546 001112
 (1) 020546 063236
 1425 020550 001113
 (1) 020550 110506
 1426

```

.SBTTL TMTC--OUTPUT SECOND CHAR OF COUNT
TMTC: BRWRT IMM,77 ;MASK TO CLEAR MXT BITS
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<77>>
      SPBR BR,AANDB,SP7 ;CLEAR THEM
      MICPC=MICPC+1
      <MOVE!SPBRX!BR!AANDB!SP7>
      OUTPUT DP,<SELB!TMTDAT> ;WRITE TO TMT SILO
      MICPC=MICPC+1
      <MOVE!WROUT!DP!<SELB!TMTDAT>> ;GET WRAPAROUND ADDRESS
      BRWRT IMM,TML8
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TML8>>
      CMP BR,SP16 ;WRAPAORUND
      MICPC=MICPC+1
      <SUBTC!BR!SP16>
      Z 10$
      MICPC=MICPC+1
      <JUMP!ZCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
      BRWRT IMM,6 ;OFFSET TO NEXT LINK
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<6>>
      SP BR,ADD,SP16 ;UPDATE THE POINTER
      MICPC=MICPC+1
      <MOVE!SPX!BR!ADD!SP16>
SS: TSTATE TMTD
     MICPC=MICPC+1
     <MOVE!WRTEBR!IMM!<TMTD-INIT&777/2>>
     MICPC=MICPC+1
     <MOVE!SPX!BR!SELB!SP2>
     ALWAYS I1 ;***OCTOBER 29, 1976
     MICPC=MICPC+1
     <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
10$: BRWRT IMM,TML1 ;GO BACK TO FIRST LINK
     MICPC=MICPC+1
     <MOVE!WRTEBR!IMM!<TML1>>
     SP BR,SELB,SP16
     MICPC=MICPC+1
     <MOVE!SPX!BR!SELB!SP16>
     ALWAYS SS
     MICPC=MICPC+1
     <JUMP!ALCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>
     ;
    
```

F01

DMC11 DDCMP PROTOCCL IMPLEMENTATION
DDCHGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 9-3
TMTD--RESPONSE FIELD-NUMBERED MESSAGE

PAGE: 0212

1428
1429 020552 001114
(1) 020552 000524
1430 020554 001115
(1) 020554 063166
1431 020556 001116
(1) 020556 111120
1432 020560 001117
(1) 020560 063167
1433 020562 001120
(1) 020562 010171
1434 020564 001121
(1) 020564 042230
1435 020566 001122
(1) 020566 063222
1436 020570 001123
(1) 020570 100454

```

SBTTL TMTD--RESPONSE FIELD-NUMBERED MESSAGE
TMTD: STATE TMTD
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<TMTD-INIT&777/2>>
      SP BR,DECA,SP6 ;ADJSRT COUNT FOR TWO'S COMPLEMENT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP6> ;NO OVERFLOW
      C TD2 ;DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      <JUMP!CCOND!<TD2-INIT&3000*4>!<TD2-INIT&777/2>>
      SP BR,DECA,SP7 ;RESP FIELD ADDR TO MAR
      <MOVE!SPX!BR!DECA!SP7>
TD2: LDMA IMM,ISP11 ;WRITE IT TO SILC
      MICPC=MICPC+1
      <MOVE!LDMA!IMM!<ISP11&377>>
TD3: OUTPUT MEMX,SELB!TMTDAT
      MICPC=MICPC+1
      <MOVE!WROUT!MEMX!<SELB!TMTDAT>>
XEXIT2: SP BR,SELB,SP2
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      ALWAYS I1
      MICPC=MICPC+1
      <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
    
```


GO1

JMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCMGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 9-4
TMTF--NUMBER FIELD--NUMBERED MESSAGE

PAGE: 0213

1438
1439 020572
1440 020572
(1) 001124
(1) 020572 123600
1441 020574
(1) 001125
(1) 020574 102054
1442 020576
(1) 001126
(1) 020576 060612
1443 020600
(1) 001127
(1) 020600 062230
1444 020602
(1) 001130
(1) 020602 000532
1445 020604
(1) 001131
(1) 020604 110600

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

HO1

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCHGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 9-5
TMTF--NUMBERED MSG ADDRESS FIELD

PAGE: 0214

1447		
1448		
1449	020606	001132
(1)		000537
(1)	020606	
1450	020610	001133
(1)		063222
(1)	020610	
1451	020612	001134
(1)		000401
(1)	020612	
1453	020614	001135
(1)		062230
(1)	020614	
1454	020616	001136
(1)		100454
(1)	020616	

```

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

```

IO1

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCHGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 9-6
TF1-NUMBERED MSG HEADER EOM

PAGE: 0215

1460			.SBTTL TF1-NUMBERED MSG HEADER EOM	
1461	020620	001137	BRWRT IMM 2	:EOM MASK TO BR
(1)		000402	MICPC=MICPC+1	
(1)	020620		<MOVE!WRTEBR!IMM!<2>>	
1462	020622	001140	OUTPUT BR <SELB!OTMTCO>	:UPDATE TMTR CONTRCL REGISTER
(1)		062231	MICPC=MICPC+1	
(1)	020622		<MOVE!WROUT!BR!<SELB!OTMTCO>>	
1463	020624	001141	OUTPUT BR <SELB!TMTDAT>	:OUTPUT A GARBAGE CHAR
(1)		062230	MICPC=MICPC+1	
(1)	020624		<MOVE!WROUT!BR!<SELB!TMTDAT>>	
1464	020626	001142	BRWRT IBUS IIBA1	:READ LOW ORDER FROM INBA
(1)		020500	MICPC=MICPC+1	
(1)	020626		<MOVE!WRTEBR!IBUS!<IIBA1>>	
1465	020630	001143	BRO TMTF1	:IF ODD BYTE--BRANCH
(1)		112162	MICPC=MICPC+1	
(1)	020630		<JUMP!BROCON!<TMTF1-INIT&3000*4>!<TMTF1-INIT&777*2>>	
1466	020632	001144	STATE TMTH	
(1)		000546	MICPC=MICPC+1	
(1)	020632		<MOVE!WRTEBR!IMM!<TMTH-INIT&777*2>>	
1467	020634	001145	ALWAYS XEXIT	
(1)		110563	MICPC=MICPC+1	
(1)	020634		JUMP!ALCOND!<EXIT-INIT&3000*4>!<EXIT-INIT&777*2>	

1469
1470
1471
1472 020636 001146
(1) 020636 123600
1474 020640 001147
(1) 020640 113151
1476 020642 001150
(1) 020642 102054
1477 020644 001151
(1) 020644 022010
1479 020646 001152
(1) 020646 023100
1479 020650 001153
(1) 020650 062064
1480 020652 001154
(1) 020652 063166
1481 020654 001155
(1) 020654 111160
1482 020656 001156
(1) 020656 063167
1483 020660 001157
(1) 020660 115407
1484 020662
1486 020662 001160
(1) 020662 020620
1487 020664 001161
(1) 020664 113165
1489 020666 001162
(1) 020666 000565
1494 020670 001163
(1) 020670 063222
1495 020672 001164
(1) 020672 100454
1497 020674
1502 020674 001165
(1) 020674 022030
1503 020676 001166
(1) 020676 023100

```

:*****TIME CRITICAL PATH--MODIFY WITH GREAT CARE
.SBTTL TMTH--ROUTINE TO OUTPUT DATA CHARACTERS
TMTH: SPBR IBUS,NPR,SP0 ;READ NPR CONTROL
      MICPC=MICPC+1
      <MOVE!SPBRX!IBUS!NPR!SP0>
      BR4 55 ;IF RECV NPR --PROCESS
      MICPC=MICPC+1
      <JUMP!BR4CON!<55-INIT&3000*4>!<55-INIT&777/2>>
      BRO I1 ;IF NPR IN PROGRESS --BRANCH
      MICPC=MICPC+1
      <JUMP!BROCON!<I1-INIT&3000*4>!<I1-INIT&777/2>>
SS:  OUTPUT IBUS,<INDAT1!TMTDAT> ;WRITE THE EVEN CHAR TO TMT SILO
      MICPC=MICPC+1
      <MOVE!WROUT!IBUS!<INDAT1!TMTDAT>>
      SP IBUS,IIBA1,SP0 ;READ LOW BYTE OF BA TO SP
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IIBA1!SP0>
      OUTPUT BR,<INCA!IBA1> ;OUTPUT INCREMENTED BA
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<INCA!IBA1>>
      SP BR,DECA,SP6 ;DECREMENT CHARACTER COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP6>
      C TH6 ;NO OVERFLOW
      MICPC=MICPC+1
      <JUMP!CCOND!<TH6-INIT&3000*4>!<TH6-INIT&777/2>>
      SP BR,DECA,SP7 ;DECREMENT HIGH BYTE OF COUNT
      MICPC=MICPC+1
      <MOVE!SPX!BR!DECA!SP7>
      Z HEH1 ;BYTE COUNT ZERO
      MICPC=MICPC+1
      <JUMP!ZCOND!<HEH1-INIT&3000*4>!<HEH1-INIT&777/2>>
TH6: BRWRT IBUS,TMTCON ;READ TMTR CONTROL CSR
      MICPC=MICPC+1
      <MOVE!WRTBR!IBUS!<TMTCON>>
      BR4 TH9 ;IF MORE ROOM IN SILO--BRANCH
      MICPC=MICPC+1
      <JUMP!BR4CON!<TH9-INIT&3000*4>!<TH9-INIT&777/2>>
TMTF1: STATE TMTHO
      MICPC=MICPC+1
      <MOVE!WRTBR!IMM!<TMTHO-INIT&777/2>>
XEXIT: SP BR,SELB,SP2 ;STORE NEW TRANSMIT STATE
      MICPC=MICPC+1
      <MOVE!SPX!BR!SELB!SP2>
      ALWAYS I1
      MICPC=MICPC+1
      <JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
TMTHO:
TH9:  OUTPUT IBUS,<INDAT2!TMTDAT> ;ODD CHAR TO SILO
      MICPC=MICPC+1
      <MOVE!WROUT!IBUS!<INDAT2!TMTDAT>>
      SP IBUS,IIBA1,SP0 ;READ LOW BYTE TO BA
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IIBA1!SP0>

```

K01

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DOCHGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 9-8
TMTH--ROUTINE TO OUTPUT DATA CHARACTERS

PAGE: 0217

1504	020700	001167	OUTPUT BR <INCA!IBA1>	;OUTPUT THE INCREMENTED BA
(1)			MICPC=MICPC+1	
(1)	020700	062064	<MOVE!WROUT!BR!<INCA!IBA1>>	
1505	020702		C HOINCH	
(1)		001170	MICPC=MICPC+1	
(1)	020702	111377	<JUMP!CCOND!<HOINCH-INIT&3000*4>!<HOINCH-INIT&777/2>>	
1506	020704		TH8: SP BR,DECA,SP6	;DECREMENT CHARACTERCOUNT
(1)		001171	MICPC=MICPC+1	
(1)	020704	063166	<MOVE!SPX!BR!DECA!SP6>	
1507	020706		C TH7	;NO OVERFLOW
(1)		001172	MICPC=MICPC+1	
(1)	020706	111175	<JUMP!CCOND!<TH7-INIT&3000*4>!<TH7-INIT&777/2>>	
1508	020710		SP BR,DECA,SP7	;DECREMENT HIGH BYTE OF COUNT
(1)		001173	MICPC=MICPC+1	
(1)	020710	063167	<MOVE!SPX!BR!DECA!SP7>	
1509	020712		Z HEH1	;BYTE COUNT ZERO
(1)		001174	MICPC=MICPC+1	
(1)	020712	115407	<JUMP!ZCOND!<HEH1-INIT&3000*4>!<HEH1-INIT&777/2>>	
1510	020714		TH7: SPBR IBUS,NPR,SPO	;READ NPR REGISTER
(1)		001175	MICPC=MICPC+1	
(1)	020714	123600	<MOVE!SPBRX!IBUS!NPR!SPO>	
1512	020716		BRO TH2	;IF NPR BUSY WAIT TO GO
(1)		001176	MICPC=MICPC+1	
(1)	020716	112205	<JUMP!BROCON!<TH2-INIT&3000*4>!<TH2-INIT&777/2>>	
1514	020720		STATE TMTH	
(1)		001177	MICPC=MICPC+1	
(1)	020720	000546	<MOVE!WRITEBR!IMM!<TMTH-INIT&777/2>>	
1515	020722		TH3: SP BR,SELB,SP2	;SAVE TSTATE
(1)		001200	MICPC=MICPC+1	
(1)	020722	063222	<MOVE!SPX!BR!SELB!SP2>	
1516	020724		TH3X: BRWRITE IMM,156	;CLEAR C0 AND C1
(1)		001201	MICPC=MICPC+1	
(1)	020724	000556	<MOVE!WRITEBR!IMM!<156>>	
1517	020726		SP BR,AANDB,SPO	;CLEAR THE BITS
(1)		001202	MICPC=MICPC+1	
(1)	020726	063260	<MOVE!SPX!BR!AANDB!SPO>	
1518	020730		OUT BR,<INCA!ONPR>	
(1)		001203	MICPC=MICPC+1	
(1)	020730	061070	<MOVE!WROUTX!BR!<INCA!ONPR>>	
1519	020732		ALWAYS I1	
(1)		001204	MICPC=MICPC+1	
(1)	020732	100454	<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>	
1521	020734		TH2: TSTATE TH7	
(1)		001205	MICPC=MICPC+1	
(1)	020734	000575	<MOVE!WRITEBR!IMM!<TH7-INIT&777/2>>	
(1)		001206	MICPC=MICPC+1	
(1)	020736	063222	<MOVE!SPX!BR!SELB!SP2>	
1522	020740		ALWAYS I1	
(1)		001207	MICPC=MICPC+1	
(1)	020740	100454	<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>	
1524			;*****END TIME CRITICAL PATH*****	
1525			;	

1527
 1528 020742 001210
 (1) 020742 010151
 1529 020744 001211
 (1) 020744 043226
 1530 020746 001212
 (1) 020746 000614
 1531 020750 001213
 (1) 020750 110521
 1532
 1533
 1534 020752 001214
 (1) 020752 010152
 1535 020754 001215
 (1) 020754 000617
 1536 020756 001216
 (1) 020756 110521

```

.SBTTL TMTI--SEND UNNUMBERED TYPE FIELD
TMTI: LDMA IMM,T ;ADDRESS OF TYPE FIELD TO MAR
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<T&377>>
      SP MEMX,SELB,SP6 ;COPY IT TO SP6
      MICPC=MICPC+1
      <MOVE!SPX!MEMX!SELB!SP6>
      STATE TMTJ
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTJ-INIT&777/2>>
      ALWAYS TD3
      MICPC=MICPC+1
      <JUMP!ALCOND!<TD3-INIT&3000*4>!<TD3-INIT&777/2>>
      ;
.SBTTL TMTJ--SEND SUB-TYPE FIELD
TMTJ: LDMA IMM,ST ;ADDRESS OF SUB-TYPE FIELD TO MAR
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<ST&377>>
      STATE TMTK
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<TMTK-INIT&777/2>>
      ALWAYS TD3
      MICPC=MICPC+1
      <JUMP!ALCOND!<TD3-INIT&3000*4>!<TD3-INIT&777/2>>
  
```

MO1

DMC11 DDCMP PROTOCOL IMPLEMENTATION
DDCHGH.MAC 21-APR-77 10:16

MACY11 30(1046) 11-JUL-77 12:25 PAGE 9-10
TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG)

PAGE: 0219

1538			.SBTTL TMTK--OUTPUT RESPONSE FIELD (UNNUMB MSG)
1539			;
1540	020760		TMTK: BRWRT IMM 3 ;WRITE A 3 TO BR
(1)		001217	MICPC=MICPC+1
(1)	020760	000403	<MOVE!WRTEBR!IMM!<3>>
1541	020762		NOP BR, SUB, SP6 ;IF TYPE LESS THAN 3
(1)		001220	MICPC=MICPC+1
(1)	020762	060346	<BR!SUB!SP6>
1542	020764		TSTATE TMTL
(1)		001221	MICPC=MICPC+1
(1)	020764	000625	<MOVE!WRTEBR!IMM!<TMTL-INIT&777/2>>
(1)		001222	MICPC=MICPC+1
(1)	020766	063222	<MOVE!SPX!BR!SELB!SP2>
1543	020770		C TMTLO
(1)		001223	MICPC=MICPC+1
(1)	020770	111232	<JUMP!CCOND!<TMTLO-INIT&3000*4>!<TMTLO-INIT&777/2>>
1544	020772		ALWAYS TD2
(1)		001224	MICPC=MICPC+1
(1)	020772	110520	<JUMP!ALCOND!<TD2-INIT&3000*4>!<TD2-INIT&777/2>>
1545			;
1546			.SBTTL TMTL--UNNUMB MSG NUMBER FIELD
1547	020774		TMTL: TSTATE TMTM
(1)		001225	MICPC=MICPC+1
(1)	020774	000637	<MOVE!WRTEBR!IMM!<TMTM-INIT&777/2>>
(1)		001226	MICPC=MICPC+1
(1)	020776	J63222	<MOVE!SPX!BR!SELB!SP2>
1548	021000		BRWRT IMM 3
(1)		001227	MICPC=MICPC+1
(1)	021000	000403	<MOVE!WRTEBR!IMM!<3>>
1549	021002		CMP BR SP6 ;IS MESSAGE REP
(1)		001230	MICPC=MICPC+1
(1)	021002	060366	<SUBTC!BR!SP6>
1550	021004		Z TMTL1 ;YES
(1)		001231	MICPC=MICPC+1
(1)	021004	111635	<JUMP!ZCOND!<TMTL1-INIT&3000*4>!<TMTL1-INIT&777/2>>
1551	021006		TMTLO: BRWRT IMM 0 ;ADDRESS CONTRAT OF ZERO
(1)		001232	MICPC=MICPC+1
(1)	021006	000400	<MOVE!WRTEBR!IMM!<0>>
1556	021010		OUTPUT BR <SELB!TMTDAT> ;SEND IT OUT
(1)		001233	MICPC=MICPC+1
(1)	021010	062730	<MOVE!WROUT!BR!<SELB!TMTDAT>>
1557	021012		ALWAYS I1 ;BACK TO IDLE LOOP
(1)		001234	MICPC=MICPC+1
(1)	021012	100454	<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
1559			;
1560	021014		TMTL1: BRWRT BR,DECA!SP12 ;WRITE A RESPONSE
(1)		001235	MICPC=MICPC+1
(1)	021014	060572	<MOVE!WRTEBR!BR!<DECA!SP12>>
1561	021016		ALWAYS TMTA5
(1)		001236	MICPC=MICPC+1
(1)	021016	110441	<JUMP!ALCOND!<TMTA5-INIT&3000*4>!<TMTA5-INIT&777/2>>
1562			;

1564
 1565 021020 001237
 (1) 000641
 (1) 021020
 1566 021022 001240
 (1) 110533
 (1) 021022
 1567 021024 001241
 (1) 000402
 (1) 021024
 1568 021026 001242
 (1) 062231
 (1) 021026
 1569 021030 001243
 (1) 062230
 (1) 021030
 1570 021032 001244
 (1) 000404
 (1) 021032
 1571 021034 001245
 (1) 063710
 (1) 021034
 1572 021036 001246
 (1) 060530
 (1) 021036
 1573 021040 001247
 (1) 113653
 (1) 021040
 1574 021042 001250
 (1) 000776
 (1) 021042
 1575 021044 001251
 (1) 063270
 (1) 021044
 1576 021046 001252
 (1) 110740
 (1) 021046
 1577 021050 001253
 (1) 000576
 (1) 021050
 1578 021052 001254
 (1) 110651
 (1) 021052

```

.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: BRWRT IMM,2 ;END OF MESSAGE TO BR
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<2>>
      OUTPUT BR<SELB!OTMTCO>
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!OTMTCO>>
      OUTPUT BR<SELB!TMTDAT> ;OUTPUT A GARBAGE CHARACTER
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!TMTDAT>>
      BRWRT IMM,4 ;SET UP LINE HAS GONE IDLE MASK
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<4>>
      SPBR BR,AORB,SP10 ;UPDATE LINE STATUS WORD
      MICPC=MICPC+1
      <MOVE!SPBRX!BR!AORB!SP10>
      BRWRT BR,AA!SP10 ;SHIFT STATUS LEFT
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<AA!SP10>>
      BR7 10$ ;IF HDX SET---BRANCH TO CLEAR OK TO SEND
      MICPC=MICPC+1
      <JUMP!BR7CON!<10$-INIT&3000*4>!<10$-INIT&777/2>>
      BRWRT IMM,376 ;MASK TO TURN OFF UNNUMB PENDDING
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<376>>
      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>>
      BRWRT IMM,176 ;CLEAR OK TO SEND AND UNNUMB PENPENDING
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<176>>
      ALWAYS 5$
      MICPC=MICPC+1
      <JUMP!ALCOND!<5$-INIT&3000*4>!<5$-INIT&777/2>>

```



```

1580          .SBTTL  TIMSRV--TIMEOUT ROUTINE--SENDS  REP
1581          ;
1582          ;ENABLE LSB
1583 021054  TIMSRV: BRWRT  IMM,177          ;MASK OFF  BR REG
                  MICPC=MICPC+1
                  <MOVE!WRTEBR!IMM!<177>>
1584          ;
1585          ;RESET TIMER---SLICK MOVE
1586          ;SINCE TIMER IS RESET BY WRITING
1587          ;A 1 AND THE EXPIRATION  LOCKS
1588 021056          OUT    BR <AANDB!OBR>          ;AND THE BIT ON
                  MICPC=MICPC+1
                  <MOVE!WROUTX!BR!<AANDB!OBR>>
1589 021060          BRWRT  BR SELA!SP1          ;READ STATUS BYTE
                  MICPC=MICPC+1
                  <MOVE!WRTEBR!BR!<SELA!SP1>>
1590 021062          BRO    IDLE
                  MICPC=MICPC+1
                  <JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1591 021064          BR7    IDLE          ;IF IN MAINT. MODE DISABLE TIMER
                  MICPC=MICPC+1
                  <JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1592 021066          SP     BR,DECA,SP15          ;DECREMENT THE COUNTER
                  MICPC=MICPC+1
                  <MOVE!SPX!BR!DECA!SP15>
1593 021070          Z      20$          ;IF ALL ONES HAS EXPIRED
                  MICPC=MICPC+1
                  <JUMP!ZCOND!<20$-INIT&3000*4>!<20$-INIT&777/2>>
1594 021072          :OS:  BRWRT  BR SELA!SP10          ;READ LINE STATUS
                  MICPC=MICPC+1
                  <MOVE!WRTEBR!BR!<SELA!SP10>>
1595 021074          BR1    TABUPD          ;NUMBERED MESSAGE IN PROGRESS
                  MICPC=MICPC+1
                  <JUMP!BR1CON!<TABUPD-INIT&3000*4>!<TABUPD-INIT&777/2>>
1596 021076          BRO    TABUPD          ;UNNUMBMSGIN PROGRESS
                  MICPC=MICPC+1
                  <JUMP!BROCON!<TABUPD-INIT&3000*4>!<TABUPD-INIT&777/2>>
1597 021100          ALWAYS IDLE          ;ELSE BACK TO IDLE LOOP
                  MICPC=MICPC+1
                  <JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1598 021102          TIME1:
1599 021102          20$:  BRWRT  IMM,2          ;
                  MICPC=MICPC+1
                  <MOVE!WRTEBR!IMM!<2>>
1600 021104          SP     BR,SELB,SP15          ;RESET THE TIMER TICK COUNT
                  MICPC=MICPC+1
                  <MOVE!SPX!BR!SELB!SP15>
1602 021106          BRWRT  IMM,201          ;SET OK TO SEND AND
                  MICPC=MICPC+1
                  <MOVE!WRTEBR!IMM!<201>>
1603 021110          SPBR   BR,AORB,SP10          ;UNNUM MSG FENCING
                  MICPC=MICPC+1
                  <MOVE!SPBRX!BR!AORB!SP10>
1608 021112          BRSHFT
                  MICPC=MICPC+1
                  <MOVE!SHFTBR!WRTEBR!SELB

```

1609	021114	001275	BR4 BS1	; IF IN START MODE--BRANCH
(1)		103111	MICPC=MICPC+1	
(1)	021114		<JUMP!BR4CON!<BS1-INIT&3000*4>!<BS1-INIT&777/2>>	
1610	021116	001276	BRWRT BR DECA!SP12	; GET LAST NUMBER SENT
(1)		060572	MICPC=MICPC+1	
(1)	021116		<MOVE!WRTEBR!BR!<DECA!SP12>>	
1611	021120	001277	CMP BR SP17	; COMPARE TO LAST ACKED
(1)		060377	MICPC=MICPC+1	
(1)	021120		<SUBTC!BR!SP17>	
1612	021122	001300	Z SNOACK	; IF EQ --SEND ACK
(1)		111733	MICPC=MICPC+1	
(1)	021122		<JUMP!ZCONO!<SNOACK-INIT&3000*4>!<SNOACK-INIT&777/2>>	
1613	021124	001301	LDMA IMM,T	; LOAD ADDRESS OF TYPE FIELD IN UNNUMB SP
(1)		010151	MICPC=MICPC+1	
(1)	021124		<MOVE!LDMA!IMM!<T&377>>	
1614	021126	001302	MEMINC IMM,3	; LOAD REP TYPE
(1)		016403	MICPC=MICPC+1	
(1)	021126		<MOVE!WRMEM!INCMAR!IMM!<3>>	
1615	021130	001303	MEMINC IMM,300	; ZERO THE SUB-TYPE
(1)		016700	MICPC=MICPC+1	
(1)	021130		<MOVE!WRMEM!INCMAR!IMM!<300>>	
1616	021132	001304	LDMA IMM,REPCS	; CUMULATIVE REPS RECD
(1)		010015	MICPC=MICPC+1	
(1)	021132		<MOVE!LDMA!IMM!<REPCS&377>>	
1617	021134	001305	SP MEMX,SELB,SPD	; COPY IT TO SPD
(1)		043220	MICPC=MICPC+1	
(1)	021134		<MOVE!SPX!MEMX!SELB!SPD>	
1618	021136	001306	MEM BR,INCA!SPD	; INCREMENT IT
(1)		062460	MICPC=MICPC+1	
(1)	021136		<MOVE!WRMEM!BR!<INCA!SPD>>	
1619	021140	001307	LDMA IMM,REPST	; ADDRESS DYNAMIC REP COUNTER
(1)		010003	MICPC=MICPC+1	
(1)	021140		<MOVE!LDMA!IMM!<REPST&377>>	
1620	021142	001310	BRWRT MEMX,SELB	; COPY IT TO THE BR
(1)		040620	MICPC=MICPC+1	
(1)	021142		<MOVE!WRTEBR!MEMX!<SELB>>	
1621	021144	001311	BSHFTB	
(1)		061620	MICPC=MICPC+1	
(1)	021144		<MOVE!SHFTBR!SELB!BR>	
1622	021146	001312	MEM BR,SELB	
(1)		062620	MICPC=MICPC+1	
(1)	021146		<MOVE!WRMEM!BR!<SELB>>	
1623	021150	001313	BRD RTHRES	
(1)		106372	MICPC=MICPC+1	
(1)	021150		<JUMP!BROCON!<RTHRES-INIT&3000*4>!<RTHRES-INIT&777/2>>	
1628	021152	001314	ALWAYS IDLE	
(1)		100451	MICPC=MICPC+1	
(1)	021152		<JUMP!ALCOND!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>	
1629			.DSABLE LSB	
1630			:	

TIME2:

1632 021154 001315
 (1) 021154 120620
 1633 021156 001316
 (1) 021156 106351
 1634 021160 001317
 (1) 021160 000402
 1635 021162 001320
 (1) 021162 062231
 1636 021164 001321
 (1) 021164 062230
 1637 021166 001322
 (1) 021166 060601
 1638 021170 001323
 (1) 021170 113762
 1639 021172 001324
 (1) 021172 063072
 1640 021174 001325
 (1) 021174 070216
 1641 021176 001326
 (1) 021176 040620
 1642 021200 001327
 (1) 021200 112340
 1643 021202 001330
 (1) 021202 000775
 1644 021204 001331
 (1) 021204 063670
 1645 021206 001332
 (1) 021206 112340
 1646 021210 001333
 (1) 021210 010151
 1648 021212 001334
 (1) 021212 016401
 1649 021214 001335
 (1) 021214 000405
 1650 021216 001336
 (1) 021216 016700
 1651 021220

```

TEOM: BRWRITE IBUS,UBBR
      MICPC=MICPC+1
      <MOVE!WRITEBR!IBUS!<UBBR>>
      BR0 NXMERR ;NON-EXISTANT MEMCRY
      MICPC=MICPC+1
      <JUMP!BROCON!<NXMERR-INIT&3000*4>!<NXMERR-INIT&777/2>>
      BRWRITE IMM,2 ;EOM TO BR
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<2>>
      OUTPUT BR,<SELB!OTMTCO> ;WRITE TMTR CONTROL
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!OTMTCO>>
      OUTPUT BR,<SELB!TMTDAT> ;WRITE GARBAGE DATA
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<SELB!TMTDAT>>
      BRWRITE BR,SELA!SP1 ;CHECK FOR BOOT MODE
      MICPC=MICPC+1
      <MOVE!WRITEBR!BR!<SELA!SP1>>
      BR7 BTEOM ;---IF SET IS MAINT MSG
      MICPC=MICPC+1
      <JUMP!BR7CON!<BTEOM-INIT&3000*4>!<BTEOM-INIT&777/2>>
      SP BR,INCA,SP12 ;INCREMENT THE MESSAGE NUMBER
      MICPC=MICPC+1
      <MOVE!SPX!BR!INCA!SP12>
TEOM1: LDMA BR,SELA!SP16 ;ADDRESS LAST TMT LINK
      MICPC=MICPC+1
      <MOVE!LDMA!BR!<SELA!SP16>>
      BRWRITE MEMX,SELB
      MICPC=MICPC+1
      <MOVE!WRITEBR!MEMX!<SELB>>
      BR0 TEOM2
      MICPC=MICPC+1
      <JUMP!BROCON!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>
TEOM3: BRWRITE IMM,375 ;TURN OFF MESSAGE PENDING
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<375>>
      SPBR BR,AANDB,SP10 ;
      MICPC=MICPC+1
      <MOVE!SPBRX!BR!AANDB!SP10>
      BR0 TEOM2 ;IF UNNUMB PENDING--GO AWAY
      MICPC=MICPC+1
      <JUMP!BROCON!<TEOM2-INIT&3000*4>!<TEOM2-INIT&777/2>>
      .SBTTL SNDACK--ROUTINE TO SEND AN ACK
SNDACK: LDMA IMM,T
      MICPC=MICPC+1
      <MOVE!LDMA!IMM!<T&377>>
      MEMINC IMM,1
      MICPC=MICPC+1
      <MOVE!WRMEM!INCMAR!IMM!<1>>
      BRWRITE IMM,5
      MICPC=MICPC+1
      <MOVE!WRITEBR!IMM!<5>>
SA2: MEMINC IMM,300
      MICPC=MICPC+1
      <MOVE!WRMEM!INCMAR!IMM!<300>>
SA3: SP BR,AORB,SP10

```

(1)		001337	MICPC=MICPC+1
(1)	021220	063310	<MOVE!SPX!BR!AORB!SP10>
1652			
1658	021222		TEOM2: †STATE TMTA
(1)		001340	MICPC=MICPC+1
(1)	021222	000403	<MOVE!WRITEBR!IMM!<TMTA-INIT&777/2>>
(1)		001341	MICPC=MICPC+1
(1)	021224	063222	<MOVE!SPX!BR!SELB!SP2>
1659	021226		ALWAYS I1
(1)		001342	MICPC=MICPC+1
(1)	021226	100454	<JUMP!ALCOND!<I1-INIT&3000*4>!<I1-INIT&777/2>>
1661	021230		FUDGE: BRWRITE IBUS,NPR ;READ NPR CONTROL
(1)		001343	MICPC=MICPC+1
(1)	021230	120600	<MOVE!WRITEBR!IBUS!<NPR>>
1662	021232		BRO IDLE ;IF NPR GOING---LEAVE
(1)		001344	MICPC=MICPC+1
(1)	021232	102051	<JUMP!BROCON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
1663	021234		BRWRITE BR!LDMAR,SELA!SP4 ;LOAD THE MAR
(1)		001345	MICPC=MICPC+1
(1)	021234	070604	<MOVE!WRITEBR!BR!LDMAR!<SELA!SP4>>
1664	021236		BR7 BS2 ;IF SET - READ BACK ALL 200
(1)		001346	MICPC=MICPC+1
(1)	021236	103520	<JUMP!BR7CON!<BS2-INIT&3000*4>!<BS2-INIT&777/2>>
1665	021240		MEMINC IBUS,INDAT1 ;OTHERWISE RESTORE TWO BYTES
(1)		001347	MICPC=MICPC+1
(1)	021240	036400	<MOVE!WRMEM!INCMAR!IBUS!<INDAT1>>
1666	021242		MEMINC IBUS,INDAT2 ;...
(1)		001350	MICPC=MICPC+1
(1)	021242	036420	<MOVE!WRMEM!INCMAR!IBUS!<INDAT2>>
1667	021244		BRWRITE IMM,2 ;UPDATE---UNIBUS ADDRESS
(1)		001351	MICPC=MICPC+1
(1)	021244	000402	<MOVE!WRITEBR!IMM!<2>>
1668	021246		SP BR,ADD,SP4 ;UPDATE NPR COUNTER
(1)		001352	MICPC=MICPC+1
(1)	021246	063004	<MOVE!SPX!BR!ADD!SP4>
1669	021250		SP IBUS,IIBA1,SPO ;UPDATE ADDRESS LOW
(1)		001353	MICPC=MICPC+1
(1)	021250	023100	<MOVE!SPX!IBUS!IIBA1!SPO>
1670	021252		OUTPUT BR,ADD!IBA1
(1)		001354	MICPC=MICPC+1
(1)	021252	062004	<MOVE!WROUT!BR!<ADD!IBA1>>
1671	021254		SP IBUS,IIBA2,SPO ;READ HIGH ADDRESS
(1)		001355	MICPC=MICPC+1
(1)	021254	023120	<MOVE!SPX!IBUS!IIBA2!SPO>
1672	021256		OUTPUT BR,AC!IBA2 ;UPDATE HIGH
(1)		001356	MICPC=MICPC+1
(1)	021256	062105	<MOVE!WROUT!BR!<AC!IBA2>>
1673	021260		SP IBUS,NPR,SPO ;READ NPR REGISTER
(1)		001357	MICPC=MICPC+1
(1)	021260	123200	<MOVE!SPX!IBUS!NPR!SPO>
1674	021262		C RESEXT ;IF CARRY---UPDATE MAX
(1)		001360	MICPC=MICPC+1
(1)	021262	105363	<JUMP!CCOND!<RESEXT-INIT&3000*4>!<RESEXT-INIT&777/2>>
1675	021264		ALWAYS TH3X ;GO DO ANOTHER NPR
(1)		001361	MICPC=MICPC+1
(1)	021264	110601	<JUMP!ALCOND!<TH3X-INIT&3000*4>!<TH3X-INIT&777/2>>

1676	021266	001362	BTEOM: BRWRT IMM,374	;MASK FOR CLEAR MSG PENDING
(1)			MICPC=MICPC+1	
(1)	021266	000774	<MOVE!WRTEBR!IMM!<374>>	
1677	021270	001363	SP BR, AANDB, SP10	;TURN THEM OFF IN LINE STATUS WORD
(1)			MICPC=MICPC+1	
(1)	021270	063270	<MOVE!SPX!BR!AANDB!SP10>	
1678	021272	001364	SP BR, SELB, SP13	;STORE UNRECOGNIZABLE VALUE INTO SP13
(1)			MICPC=MICPC+1	
(1)	021272	063233	<MOVE!SPX!BR!SELB!SP13>	
1679				;SO "RH3" WILL EXIT BACK TO IDLE LOOP
1580	021274		LDMA IMM,STC	;ADDRESS START OF TMT CHAIN
(1)		001365	MICPC=MICPC+1	
(1)	021274	010067	<MOVE!LDMA!IMM!<STC&377>>	
1681	021276		SP MEMX, SELB, SPO	;COPY LINK ADDRESS
(1)		001366	MICPC=MICPC+1	
(1)	021276	043220	<MOVE!SPX!MEMX!SELB!SPO>	
1686	021300		TSTATE TMTA	;CHANGE XMIT STATE TO LINE IS IDLE
(1)		001367	MICPC=MICPC+1	
(1)	021300	000403	<MOVE!WRTEBR!IMM!<TMTA-INIT&777/2>>	
(1)		001370	MICPC=MICPC+1	
(1)	021302	063222	<MOVE!SPX!BR!SELB!SP2>	
1688	021304		ALWAYS TDON2	;POST A DONE
(1)		001371	MICPC=MICPC+1	
(1)	021304	114532	<JUMP!ALCOND!<TDON2-INIT&3000*4>!<TDON2-INIT&777/2>>	
1689	021306		RL4: RSTATE RCVL	
(1)		001372	MICPC=MICPC+1	
(1)	021306	000705	<MOVE!WRTEBR!IMM!<RCVL-INIT&777/2>>	
(1)		001373	MICPC=MICPC+1	
(1)	021310	063223	<MOVE!SPX!BR!SELB!SP3>	
1690	021312		SP IBUS, NPR, SPO	;READ NPR CONTROL REGISTER
(1)		001374	MICPC=MICPC+1	
(1)	021312	123200	<MOVE!SPX!IBUS!NPR!SPO>	
1691	021314		BRWRT IMM,221	
(1)		001375	MICPC=MICPC+1	
(1)	021314	000621	<MOVE!WRTEBR!IMM!<221>>	
1692	021316		ALWAYS RK7	
(1)		001376	MICPC=MICPC+1	
(1)	021316	104623	<JUMP!ALCOND!<RK7-INIT&3000*4>!<RK7-INIT&777/2>>	
1693	021320		HCINCH: SP IBUS, IIBA2, SPO	
(1)		001377	MICPC=MICPC+1	
(1)	021320	023120	<MOVE!SPX!IBUS!IIBA2!SPO>	
1694	021322		OUTPUT BR, INCA, IBA2	;OUTPUT INCREMENTED BA
(1)		001400	MICPC=MICPC+1	
(1)	021322	062065	<MOVE!WROUT!BR!<INCA!IBA2>>	
1695	021324		C SS	;INCREMENT BYTEW COUNT
(1)		001401	MICPC=MICPC+1	
(1)	021324	115003	<JUMP!CCOND!<SS-INIT&3000*4>!<SS-INIT&777/2>>	
1696	021326		ALWAYS TH8	
(1)		001402	MICPC=MICPC+1	
(1)	021326	110571	<JUMP!ALCOND!<TH8-INIT&3000*4>!<TH8-INIT&777/2>>	
1697				;INCREMENT NXT BITS
1698	021330		ES: SP IBUS, NPR, SPO	;READ NPR REG IWITH CURRENT NXT BITS
(1)		001403	MICPC=MICPC+1	
(1)	021330	123200	<MOVE!SPX!IBUS!NPR!SPO>	
1699	021332		BRWRT IMM,4	;WRITE BIT TO ADD
(1)		001404	MICPC=MICPC+1	

(1) 021332 000404
 1700 021334
 (1) 001405
 (1) 021334 061010
 1701 021336
 (1) 001406
 (1) 021336 110571
 1702
 1704 021340
 (1) 001407
 (1) 021340 000715
 1705 021342
 (1) 001410
 (1) 021342 110563
 1707

```

    <MOVE!WRTEBR!IMM!<4>>
    OUT BR,<ADD!ONPR> ;TURN ON PROPER MXT BITS
    MICPC=MICPC+1
    <MOVE!WROUTX!BR!<ADD!ONPR>>
    ALWAYS TH8
    MICPC=MICPC+1
    <JUMP!ALCOND!<TH8-INIT&3000*4>!<TH8-INIT&777/2>>
    ;
HEH1: STATE TEOM
    MICPC=MICPC+1
    <MOVE!WRTEBR!IMM!<TEOM-INIT&777/2>>
    ALWAYS XEXIT
    MICPC=MICPC+1
    <JUMP!ALCOND! XEXIT-INIT&3000*4;!<XEXIT-INIT&777/2>>
    ;

```

H02

1713			.SBTTL REP HANDLER	
1714	021344		LDMA IMM,REPCR	:LOAD MAR ADDRESS WITH PCINTER TO REPS RECC
(1)		001411	MICPC=MICPC+1	
(1)	021344	010016	<MOVE!LDMAR!IMM!<REPCR&377>>	:READ NUMBER OF REPS RECD
1715	021346		SP MEMX,SELB,SPO	
(1)		001412	MICPC=MICPC+1	
(1)	021346	043220	<MOVE!SPX!MEMX!SELB!SPO>	:INCREMNT REPS RECD
1716	021350		MEM DP<INCA!SPO>	
(1)		001413	MICPC=MICPC+1	
(1)	021350	062450	<MOVE!WRMEM!DP!<INCA!SPO>>	:LOAD ADDRESS OF TYPE FIELD
1717	021352		LDMA IMM,T	
(1)		001414	MICPC=MICPC+1	
(1)	021352	010151	<MOVE!LDMAR!IMM!<T&377>>	:LOAD NAK TYPE
1718	021354		MEMINC IMM,2	
(1)		001415	MICPC=MICPC+1	
(1)	021354	016402	<MOVE!WRMEM!INCMAR!IMM!<2>>	:LOAD REP RESPONSE SUB-TYPE
1719	021356		MEMINC IMM,303	
(1)		001416	MICPC=MICPC+1	
(1)	021356	016703	<MOVE!WRMEM!INCMAR!IMM!<303>>	:SEND AN UNNUMB MSG
1720	021360		ALWAYS SNAK	
(1)		001417	MICPC=MICPC+1	
(1)	021360	114704	<JUMP!ALCOND!<SNAK-INIT&3000*4>!<SNAK-INIT&777/2>>	
1721			:	
1722			.SBTTL START HANDLER	
1723	021362		BRWRT DP<SELA!SP10>	:READ LINE STATUS WORD
(1)		001420	MICPC=MICPC+1	
(1)	021362	060610	<MOVE!WRTEBR!DP!<SELA!SP10>>	:GET START MODE BIT IN TESTABLE POSITION
1724	021364		BRSHFT	
(1)		001421	MICPC=MICPC+1	
(1)	021364	001620	<MOVE!SHFTBR!WRTEBR!SELB>	:IF IN START MODE SET STACK
1725	021366		BR4 10\$	
(1)		001422	MICPC=MICPC+1	
(1)	021366	117026	<JUMP!BR4CON!<10\$-INIT&3000*4>!<10\$-INIT&777/2>>	:ELSE SET UP START ERROR
1726			:	
1727	021370		LDMA IMM,<<RTHRS+3>>	
(1)		001423	MICPC=MICPC+1	
(1)	021370	010177	<MOVE!LDMAR!IMM!<<RTHRS+3>&377>>	
1728	021372		BRWRT IMM,200	
(1)		001424	MICPC=MICPC+1	
(1)	021372	000600	<MOVE!WRTEBR!IMM!<200>>	
1729	021374		ALWAYS RCEXY	
(1)		001425	MICPC=MICPC+1	
(1)	021374	114522	<JUMP!ALCOND!<RCEXY-INIT&3000*4>!<RCEXY-INIT&777/2>>	:SET UP ADDRESS OF TYPE FIELD
1730	021376		LDMA IMM,T	
(1)		001426	MICPC=MICPC+1	
(1)	021376	010151	<MOVE!LDMAR!IMM!<T&377>>	:WRITE STACK TYPE
1731	021400		MEMINC IMM,7	
(1)		001427	MICPC=MICPC+1	
(1)	021400	016407	<MOVE!WRMEM!INCMAR!IMM!<7>>	:SET START RECD AND UNNUMB PENDING
1732	021402		BRWRT IMM,11	
(1)		001430	MICPC=MICPC+1	
(1)	021402	000411	<MOVE!WRTEBR!IMM!<11>>	:SEND THE UNNUMBERED MESSAGE
1733	021404		ALWAYS SA2	
(1)		001431	MICPC=MICPC+1	
(1)	021404	110736	<JUMP!ALCOND!<SA2-INIT&3000*4>!<SA2-INIT&777/2>>	
1734			:	

1735		
1736	021406	001432
(1)		
(1)	021406	000727
1737	021410	001433
(1)		
(1)	021410	063270
1738	021412	001434
(1)		
(1)	021412	110670

```

STACK:  SBTL  STACK HANDLER
        BRWRT IMM,327          ;MASK TO CLEAR START MODE
        MICPC=MICPC+1
        <MOVE!WRTEBR!IMM!<327>>
        SP BR,AANDB,SP10      ;CLEAR START MODE
        MICPC=MICPC+1
        <MOVE!SPX!BR!AANDB!SP10>
        ALWAYS TIME1         ;RESET TIMER AND IDLE
        MICPC=MICPC+1
        <JUMP!ALCOND!<TIME1-INIT$3000*4>!<TIME1-INIT$777/2>

```


1740	021414	001435	ICBA22: SP IBUS, IOBA2, SPO	; READ THE HIGH ORDER BITS OF BA TO SPC
(1)			MICPC=MICPC+1	
(1)	021414	023160	<MOVE!SPX!IBUS!IOBA2!SPO>	
1741	021416	001436	OUTPUT DP <INCA!OBA2>	; OUTPUT THE INCREMENTED COUNT
(1)			MICPC=MICPC+1	
(1)	021416	062067	<MOVE!WROUT!DP!<INCA!OBA2>>	
1742	021420	001437	C 55	; IF CARRY SET INCREMENT THE MYTBITS
(1)			MICPC=MICPC+1	
(1)	021420	115041	<JUMP!CCOND!<55-INIT&3000*4>!<55-INIT&777/2>>	
1743	021422	001440	ALWAYS RK3	
(1)			MICPC=MICPC+1	
(1)	021422	104641	<JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>	
1744			:	
1745	021424	001441	SS: SP IBUS, UBBR, SPO	
(1)			MICPC=MICPC+1	
(1)	021424	123220	<MOVE!SPX!IBUS!UBBR!SPO>	
1746	021426	001442	BRWRT IMM, 4	
(1)			MICPC=MICPC+1	
(1)	021426	000404	<MOVE!WRTEBR!IMM!<4>>	
1747	021430	001443	OUT BR <ADD!OBR>	
(1)			MICPC=MICPC+1	
(1)	021430	061011	<MOVE!WROUTX!BR!<ADD!OBR>>	
1748	021432	001444	ALWAYS RK3	
(1)			MICPC=MICPC+1	
(1)	021432	104641	<JUMP!ALCOND!<RK3-INIT&3000*4>!<RK3-INIT&777/2>>	
1753	021434	001445	NAK: LDMA IMM, NDATA	; CUMMULATIVE NAK COUNTER
(1)			MICPC=MICPC+1	
(1)	021434	010011	<MOVE!LDMA!IMM!<NDATA&377>>	
1754	021436	001446	SP MEMX, SELB, SPO	; READ IT
(1)			MICPC=MICPC+1	
(1)	021436	043220	<MOVE!SPX!MEMX!SELB!SPO>	
1755	021440	001447	MEM MEMX, INCA, SPO	; INCREMENT THE COUNTER
(1)			MICPC=MICPC+1	
(1)	021440	042460	<MOVE!WRMEM!MEMX!<INCA!SPO>>	
1756	021442	001450	LDMA IMM, STC	; ADDRESS START OF TMT CHAIN
(1)			MICPC=MICPC+1	
(1)	021442	010067	<MOVE!LDMA!IMM!<STC&377>>	
1757	021444	001451	SP MEMX, SELB, SP16	; COPY START OF CHAIN TO LAST XMIT POINTER
(1)			MICPC=MICPC+1	
(1)	021444	043236	<MOVE!SPX!MEMX!SELB!SP16>	
1758	021446	001452	BRWRT BR, INCA, SP17	; GETLASTMESSAGE ACKED
(1)			MICPC=MICPC+1	
(1)	021446	060477	<MOVE!WRTEBR!BR!<INCA!SP17>>	
1759	021450	001453	SP BR, SELB, SP12	; COPY TO CURRENT NUMBER
(1)			MICPC=MICPC+1	
(1)	021450	063232	<MOVE!SPX!BR!SELB!SP12>	
1760	021452	001454	BRWRT IMM, 6	; WRITE NUMBERED MSG PENDING
(1)			MICPC=MICPC+1	
(1)	021452	000406	<MOVE!WRTEBR!IMM!<6>>	
1761			:	
1762	021454	001455	SP BR, AORB, SP10	; AND LINE HAS GONE IDLE
(1)			MICPC=MICPC+1	
(1)	021454	063310	<MOVE!SPX!BR!AORB!SP10>	; SET IT IN LINE STATUS WORD
1763	021456	001456	SP BR, SELB, SP15	; RESET TIMER COUNT
(1)			MICPC=MICPC+1	
(1)	021456	063235	<MOVE!SPX!BR!SELB!SP15>	

```

1764 021460
(1)
(1) 021460 001457
1765 021462 110725
(1)
(1) 021462 001460
1766 021464 000415
(1)
(1) 021464 001461
1767 021466 123220
(1)
(1) 021466 001462
1768 021470 063260
(1)
(1) 021470 001463
1769 021472 000600
(1)
(1) 021472 001464
1770 021474 061311
(1)
(1) 021474 001465
1771 021476 123000
(1)
(1) 021476 001466
1772 021476 100554

```

```

ALWAYS TEOM1
MICPC=MICPC+1
<JUMP!ALCOND!<TEOM1-INIT&3000*4>!<TEOM1-INIT&777/2>>
ININT: BRWRT IMM,15 ;MASK FOR TURN OFF ALL BUT EXT MEM BITS + NXM
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<15>>
SP IBUS,UBBR,SPO ;READ BR CONTROL REGISTER
MICPC=MICPC+1
<MOVE!SPX!IBUS!UBBR!SPO>
SF BR,AANDB,SPO ;MASK OFF VECTOR TO X04
MICPC=MICPC+1
<MOVE!SPX!BR!AANDB!SPO>
BRWRT IMM,200 ;MASK FOR INTERRUPT
MICPC=MICPC+1
OUT BR,AORB,OBR ;INTERRUPT
MICPC=MICPC+1
<MOVE!WROUTX!BR!<AORB!OBR>>
SP IBUS,INCON,SPO ;RESTORE INPUT CONTROL CSR
MICPC=MICPC+1
<MOVE!SPX!IBUS!INCON!SPO>
ALWAYS NIDLE4
MICPC=MICPC+1
<JUMP!ALCOND!<NIDLE4-INIT&3000*4>!<NIDLE4-INIT&777/2>>
:

```

1783
 1784 021500
 (1) 001467
 (1) 021500 000477
 1785 021502
 (1) 001470
 (1) 021502 063265
 1786 021504
 (1) 001471
 (1) 021504 070074
 1787 021506
 (1) 001472
 (1) 021506 077220
 1788 021510
 (1) 001473
 (1) 021510 074601
 1789 021512
 (1) 001474
 (1) 021512 001620
 1790 021514
 (1) 001475
 (1) 021514 116502
 1791 021516
 (1) 001476
 (1) 021516 054660
 1792 021520
 (1) 001477
 (1) 021520 060365
 1793 021522
 (1) 001500
 (1) 021522 115113
 1794 021524
 (1) 001501
 (1) 021524 115510
 1795 021526
 (1) 001502
 (1) 021526 020540
 1796 021530
 (1) 001503
 (1) 021530 116106
 1801 021532
 (1) 001504
 (1) 021532 000660
 1802 021534
 (1) 001505
 (1) 021534 100450
 1803
 1804 021536
 (1) 001506
 (1) 021536 000607
 1805 021540
 (1) 001507
 (1) 021540 100450
 1806
 1807 021542
 (1) 001510

```

: FUGITIVE RECEIVE ROUTINES---DON'T FIT IN PAGE
RH1: BRWRT IMM,77
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<77>>
      SP BR,AANDB,SP5
      MICPC=MICPC+1
      <MOVE!SPX!BR!AANDB!SP5>
      LDMA BR,<INCA!SP14> ;LOAD ADDRESS OF CURRENT COUNT
      MICPC=MICPC+1
      <MOVE!LDMA!BR!<INCA!SP14>>
      SP BR!INCMAR,SELB,SPO ;SAVE MASK
      MICPC=MICPC+1
      <MOVE!SPX!BR!INCMAR!SELB!SPO>
      BRWRT BR!INCMAR,SELA!SP1 ;READ STATUS BYTE
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!INCMAR!SELA!SP1>>
      BRSHFT ;SHIFT IT RIGHT
      MICPC=MICPC+1
      <MOVE!SHFTBR!WRTEBR!SELB>
      BR1 RH2 ;NO BUFFER ASSIGNED IN MAINT MODE
      MICPC=MICPC+1
      <JUMP!BR1CON!<RH2-INIT&3000*4>!<RH2-INIT&777/2>>
      BRWRT MEMX!INCMAR,AANDB!SPO ;GET HIGH BYTE COUNT BITS
      MICPC=MICPC+1
      <MOVE!WRTEBR!MEMX!INCMAR!<AANDB!SPO>>
      CMP BR,SP5 ;COMPARE HIGH ORDER BITS OF COUNT
      MICPC=MICPC+1
      <SUBTC!BR!SP5>
      C RCFATL ;IF CARRY--TOO BIG ERROR
      MICPC=MICPC+1
      <JUMP!CCOND!<RCFATL-INIT&3000*4>!<RCFATL-INIT&777/2>>
      Z RCLOW ;IF EQUAL COMPARE LOW ORDER BITS OF COUNT
      MICPC=MICPC+1
      <JUMP!ZCOND!<RCLOW-INIT&3000*4>!<RCLOW-INIT&777/2>>
RH2: BRWRT IBUS,IOBA1 ;READ LOW BYTE OF IN BA
      MICPC=MICPC+1
      <MOVE!WRTEBR!IBUS!<IOBA1>>
      BRO RCVODD ;IF SET IS ODD TRANSFER
      MICPC=MICPC+1
      <JUMP!BROCON!<RCVODD-INIT&3000*4>!<RCVODD-INIT&777/2>>
      STATE RCVKEO
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<RCVKEO-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
  
```

1808	(1)	021542	040364	(SUBTC!MEMX!SP4)	
	(1)	021544	001511	C RCFATL	;CARRY--TOO BIG
	(1)	021544	115113	MICPC=MICPC+1	
1809	(1)	021546	001512	<JUMP!CCOND!<RCFATL-INIT&3000*4>!<RCFATL-INIT&777/2>>	
	(1)	021546	114502	ALWAYS RH2	;ELSE CONTINUE
1810	(1)	021550	001513	MICPC=MICPC+1	
	(1)	021550	010151	<JUMP!ALCOND!<RH2-INIT&3000*4>!<RH2-INIT&777/2>>	
1811	(1)	021552	001514	RCFATL: LDMA IMM,1	
	(1)	021552	016402	MICPC=MICPC+1	
1812	(1)	021554	001515	<MOVE!LDMAR!IMM!<T&377>>	
	(1)	021554	002711	MEMINC IMM,2	
1813	(1)	021556	001516	MICPC=MICPC+1	
	(1)	021556	010175	<MOVE!WRMEM!INCMAR!IMM!<2>>	
1814	(1)	021560	001517	MEM IMM,311	
	(1)	021560	036540	MICPC=MICPC+1	
1815	(1)	021562	001520	<MOVE!WRMEM!IMM!<311>>	
	(1)	021562	036560	LDMA IMM,<<RTHRS+1>>	;ADDRESS ERROR LINK
1816	(1)	021564	001521	MICPC=MICPC+1	
	(1)	021564	000420	<MOVE!LDMAR!IMM!<<RTHRS+1>&377>>	
1817	(1)	021566	001522	MEMINC IBUS,IOBA1	
	(1)	021566	016400	MICPC=MICPC+1	
1818	(1)	021570	001523	<MOVE!WRMEM!INCMAR!IBUS!<IOBA1>>	
	(1)	021570	062620	MEMINC IBUS,IOBA2	
1819	(1)	021572	001524	MICPC=MICPC+1	
	(1)	021572	002212	<MOVE!WRMEM!INCMAR!IBUS!<IOBA2>>	
1820	(1)	021574	001525	BRWRT IMM,20	
	(1)	021574	003002	MICPC=MICPC+1	
1821	(1)	021576	001526	<MOVE!WRTEBR!IMM!<20>>	
	(1)	021576	003001	RCEXY: MEMINC IMM,0	
1822	(1)	021600	001527	MICPC=MICPC+1	
	(1)	021600	114666	<MOVE!WRMEM!INCMAR!IMM!<0>>	
1823	(1)	021602	001530	MEM BR,SELB	
	(1)	021602	040757	MICPC=MICPC+1	
1824	(1)	021604	001531	<MOVE!WRMEM!BR!<SELB>>	
	(1)	021604	107562	RCEXX: OUTPUT IMM,<200!ORCVCO>	;FLUSH INPUT SILO
1825	(1)	021606	001532	MICPC=MICPC+1	
1826	(1)	021610	070200	<MOVE!WROUT!IMM!<200!ORCVCO>>	
	(1)	021610		SP IMM,SP2,2	;INHIBIT FURTHER TRANSMISSIONS
	(1)	021610		MICPC=MICPC+1	
	(1)	021610		<MOVE!SPX!IMM!SP2!2>	
	(1)	021610		SP IMM,1,SP1	;SET INIT MODE IN PORT STATUS WORD
	(1)	021610		MICPC=MICPC+1	
	(1)	021610		<MOVE!SPX!IMM!1!SP1>	
	(1)	021610		ALWAYS NTRS1	
	(1)	021610		MICPC=MICPC+1	
	(1)	021610		<JUMP!ALCOND!<NTRS1-INIT&3000*4>!<NTRS1-INIT&777/2>>	
	(1)	021610		TDON3: BRWRT MEMX,SUB!SP17	;COMPARE RESPONSE TO MSG NO
	(1)	021610		MICPC=MICPC+1	
	(1)	021610		<MOVE!WRTEBR!MEMX!<SUB!SP17>>	
	(1)	021610		BR7 RH3	;IF NEGATIVE EXIT
	(1)	021610		MICPC=MICPC+1	
	(1)	021610		<JUMP!BR7CON!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>	
	(1)	021610		TDON2: LDMA BR,SELA!SPO	;ADDRESS THE TRANSMITLINK
	(1)	021610		MICPC=MICPC+1	
	(1)	021610		<MOVE!LDMAR!BR!<SELA!SPO>>	
	(1)	021610		MEM IMM,0	;TURN OF ASSIGNEDAND TMTED BITS IN FLAG

(1)		001533	MICPC=MICPC+1	
(1)	021610	002400	<MOVE!WRMEM!IMM!<0>>	
1827	021612		LDMA IMM,STC	
(1)		001534	MICPC=MICPC+1	
(1)	021612	010067	<MOVE!LDMAR!IMM!<STC&377>>	
1828	021614		MEM IMM,TML1	;ASSUME WRAPAROUND
(1)		001535	MICPC=MICPC+1	
(1)	021614	002471	<MOVE!WRMEM!IMM!<TML1>>	
1829	021616		BRWRT IMM,TML8	;WRAPAROUND?
(1)		001536	MICPC=MICPC+1	
(1)	021616	000543	<MOVE!WRTEBR!IMM!<TML8>>	
1830	021620		CMP BR,SPO	
(1)		001537	MICPC=MICPC+1	
(1)	021620	060360	<SUBTC!BR!SPO>	
1831	021622		Z TDON4	;YES
(1)		001540	MICPC=MICPC+1	
(1)	021622	115543	<JUMP!ZCOND!<TDON4-INIT&3000*4>!<TDON4-INIT&777/2>>	
1832	021624		BRWRT IMM,6	;OFFSET FOR NEXT TMT LINK
(1)		001541	MICPC=MICPC+1	
(1)	021624	000406	<MOVE!WRTEBR!IMM!<6>>	
1833	021626		MEM BR,ADD!SPO	;UPDATE THE POINTER
(1)		001542	MICPC=MICPC+1	
(1)	021626	062400	<MOVE!WRMEM!BR!<ADD!SPO>>	
1834	021630		LDMA IMM,NXTSP	;ADDRESS DONE LINK
(1)		001543	MICPC=MICPC+1	
(1)	021630	010241	<MOVE!LDMAR!IMM!<NXTSP&377>>	
1835	021632		LDMA MEMX,SELB!SPX!SP3	;ADDRESS THE LINK,COPYING
(1)		001544	MICPC=MICPC+1	
(1)	021632	053223	<MOVE!LDMAR!MEMX!<SELB!SPX!SP3>>	
1836				;ITS ADDRESS TO SPO
1837	021634		MEMINC IMM,200	;WRITE THE INTERRUPT TYPE
(1)		001545	MICPC=MICPC+1	
(1)	021634	016600	<MOVE!WRMEM!INCMAR!IMM!<200>>	
1838	021636		MEM BR,INCA!SPO	;COPY ACTUAL LINK ADDRESS
(1)		001546	MICPC=MICPC+1	
(1)	021636	062460	<MOVE!WRMEM!BR!<INCA!SPO>>	
1839	021640		LDMA IMM,NXTSP	;ADDRESS PTR INT STACK
(1)		001547	MICPC=MICPC+1	
(1)	021640	010241	<MOVE!LDMAR!IMM!<NXTSP&377>>	
1840	021642		MEM IMM,INTSTK	;ASSUME WRAP AROUND
(1)		001550	MICPC=MICPC+1	
(1)	021642	002642	<MOVE!WRMEM!IMM!<INTSTK>>	
1841	021644		BRWRT IMM,<<MMEND-2>>	;ADDRESS ENDOFINT STACK
(1)		001551	MICPC=MICPC+1	
(1)	021644	000776	<MOVE!WRTEBR!IMM!<<MMEND-2>>>	
1842	021646		CMP BR,SP3	;WRAPAROUND?
(1)		001552	MICPC=MICPC+1	
(1)	021646	060363	<SUBTC!BR!SP3>	
1843	021650		Z TDON40	;YES---BRANCH
(1)		001553	MICPC=MICPC+1	
(1)	021650	115556	<JUMP!ZCOND!<TDON40-INIT&3000*4>!<TDON40-INIT&777/2>>	
1844	021652		BRWRT IMM,2	;OFFSET TO NEXT PAIR
(1)		001554	MICPC=MICPC+1	
(1)	021652	000402	<MOVE!WRTEBR!IMM!<2>>	
1845	021654		MEM BR,ADD!SP3	;UPDATE POINTER
(1)		001555	MICPC=MICPC+1	

TDON4:

(1) 021654 062403
 1846 021656 001556
 (1) 021656 000420
 1847 021660 001557
 (1) 021660 063301
 1848 021662 00156C
 (1) 021662 010070
 1849 021664 001561
 (1) 021664 043220
 1850 021666 001562
 (1) 021666 010067
 1851 021670 001563
 (1) 021670 04036C
 1852 021672 001564
 (1) 021672 105562
 1853
 1854 021674 001565
 (1) 021674 010153
 1855 021676 001566
 (1) 021676 043237
 1856 021700 001567
 (1) 021700 010067
 1857 021702 00157C
 (1) 021702 053620
 1858 021704 001571
 (1) 021704 054620
 1859 021706 001572
 (1) 021706 11653C
 1860 021710 001573
 (1) 021710 104562
 1861
 1862 021712 001574
 (1) 021712 000404
 1863 021714 001575
 (1) 021714 114663
 1864
 1865
 1866
 1867
 1868 021716

```

  TDON40: <MOVE!WRMEM!BR!<ADD!SP3>>
          BRWRT IMM,20 ;WRITE INTERUP* PENDING
          MICPC=MICPC+1
          <MOVE!WRTEBR!IMM!<20>>
          SP BR,AORB,SP1 ;IN PORT STATUS WORD
          MICPC=MICPC+1
          <MOVE!SPX!BR!AORB!SP1>
          LDMA IMM,ETC ;ADDRESS NEXT EMPTY PTR
          MICPC=MICPC+1
          <MOVE!LDMA!IMM!<ETC&377>>
          SP MEMX,SELB,SPO ;COPY IT TO SPO
          MICPC=MICPC+1
          <MOVE!SPX!MEMX!SELB!SPO>
          LDMA IMM,STC ;GET NEXT DONE PTR
          MICPC=MICPC+1
          <MOVE!LDMA!IMM!<STC&377>>
          CMP MEMX,SPO ;IDENTICAL?
          MICPC=MICPC+1
          <SUBTC!MEMX!SPO>
          Z RH3 ;FINISH PROCESSING HEADER
          MICPC=MICPC+1
          <JUMP!ZCOND!<RH3-INIT&3000*4>!<RH3-INIT&777/2>>

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

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

  : INPUTS:
  : SPO = RECEIVE CHARACTER

  PASWD: SP IBUS,LNOSW,SP13 ;READ PASSWD SWITCH

```

```

(1) 001576
(1) 021716 023333
1869 021720 001577
(1) 021720 115603
1870 021722 001600
(1) 021722 000406
1871 021724 001601
(1) 021724 060360
1872 021726 001602
(1) 021726 115611
1873 021730 001603
(1) 021730 060601
1874 021732 001604
(1) 021732 001620
1875 021734 001605
(1) 021734 106740
1876 021736 001606
(1) 021736 001620
1877 021740 001607
(1) 021740 106732
1878 021742 001610
(1) 021742 104641
1879 021744 001611
(1) 021744 063164
1880 021746 001612
(1) 021746 000735
1881 021750 001613
(1) 021750 100450
1882
1883
1884
1885 021752 001614
(1) 021752 010001
1886 021754 001615
(1) 021754 002401
1887 021756 001616
(1) 021756 010167
1888 021760 001617
(1) 021760 057224

```

```

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

```

1889	021762		SP MEMX! INCMAR, SELB, SP5 ;AND SP5
(1)		001620	MICPC=MICPC+1
(1)	021762	057225	<MOVE!SPX!MEMX!INCMAR!SELB!SP5>
1890	021764		MEM BR DECA!SP11 ;COPY SP11 FROM MEMORY
(1)		001621	MICPC=MICPC+1
(1)	021764	062571	<MOVE!WRMEM!BR!<DECA!SP11>>
1891	021766		LDMA IMM NXTSP
(1)		001622	MICPC=MICPC+1
(1)	021766	010241	<MOVE!LDMAR!IMM!<NXTSP&377>>
1892	021770		SP MEMX!LDMAR, SELB, SP3 ;COPY TO SP3
(1)		001623	MICPC=MICPC+1
(1)	021770	053223	<MOVE!SPX!MEMX!LDMAR!SELB!SP3>
1893	021772		MEMINC IMM, 204 ;RECEIVE DONE IMAGE
(1)		001624	MICPC=MICPC+1
(1)	021772	016604	<MOVE!WRMEM!INCMAR!IMM!<204>>
1894	021774		MEM BR!LDMAR, SELA!SP14 ;COPY LINK ADDRESS TO NEXT INTER
(1)		001625	MICPC=MICPC+1
(1)	021774	072614	<MOVE!WRMEM!BR!LDMAR!<SELA!SP14>>
1895	021776		MEMINC IMM, 0 ;ZERO THE FLAGS
(1)		001626	MICPC=MICPC+1
(1)	021776	016400	<MOVE!WRMEM!INCMAR!IMM!<0>>
1896	022000		SP IMM!INCMAR, SPO, 300 ;WRITE A 300 TO SPO
(1)		001627	MICPC=MICPC+1
(1)	022000	017300	<MOVE!SPX!IMM!INCMAR!SPO!300>
1897	022002		BRWRT IMM!INCMAR, 2 ;PREPARE TO ADDRESS NEXT
(1)		001630	MICPC=MICPC+1
(1)	022002	014402	<MOVE!WRTEBR!IMM!INCMAR!<2>>
1898			; INTERRUPT STACK AND INCREMENT
1899			; THE MAR
1900	022004		MEM MEMX, AANDB!SPO ;MASK OFF ORIGINAL HIGH BYTE
(1)		001631	MICPC=MICPC+1
(1)	022004	042650	<MOVE!WRMEM!MEMX!<AANDB!SPO>>
1901			; OF COUNT SAVING EXTENDED MEM BITS
1902	022006		MEMINC MEMX, AORB!SP5 ;COPY TO MEMORY LINK
(1)		001632	MICPC=MICPC+1
(1)	022006	056705	<MOVE!WRMEM!INCMAR!MEMX!<AORB!SP5>>
1903	022010		MEMINC BR SELA!SP4
(1)		001633	MICPC=MICPC+1
(1)	022010	076604	<MOVE!WRMEM!INCMAR!BR!<SELA!SP4>>
1904	022012		LDMA IMM NXTSP ;ADDRESS NEXT INT STACK
(1)		001634	MICPC=MICPC+1
(1)	022012	010241	<MOVE!LDMAR!IMM!<NXTSP&377>>
1905	022014		MEM BR ADD!SP3
(1)		001635	MICPC=MICPC+1
(1)	022014	062403	<MOVE!WRMEM!BR!<ADD!SP3>>
1906	022016		BRWRT IMM, <<MMEND-2>> ;ADDRESSEND OF INT STACK
(1)		001636	MICPC=MICPC+1
(1)	022016	000776	<MOVE!WRTEBR!IMM!<<MMEND-2>>>
1907	022020		CMP BR SP3 ;WRAP AROUND
(1)		001637	MICPC=MICPC+1
(1)	022020	060363	<SUBTC!BR!SP3>
1908	022022		Z 40\$;IF YES-- BRANCH
(1)		001640	MICPC=MICPC+1
(1)	022022	115651	<JUMP!ZCOND!<40\$-INIT&3000+4>!<40\$-INIT&777-2>>
1909	022024		BRWRT IMM, 5 ;INDEX TO NEXT BUFFER
1910	022024		

208:

(1)		001641	MICPC=MICPC+1	
(1)	022024	000405	<MOVE!WRITEBR!IMM!<5>>	
1911	022026		SP BR ADD,SP14	:UPDATE COPY OF POINTER
(1)		001642	MICPC=MICPC+1	
(1)	022026	063014	<MOVE!SPX!BR!ADD!SP14>	
1912	022030		BRWTE IMM,STC	:ADDRESS OF WRAP AROUND POINT
(1)		001643	MICPC=MICPC+1	
(1)	022030	000467	<MOVE!WRITEBR!IMM!<STC>>	
1913	022032		CMP BR,SP14	:WRAPAROUND?
(1)		001644	MICPC=MICPC+1	
(1)	022032	060374	<SUBTC!BR!SP14>	
1914	022034		Z 50\$:IF YES---BRANCH
(1)		001645	MICPC=MICPC+1	
(1)	022034	115653	<JUMP!ZCOND!<50\$-INIT&3000*4>!<50\$-INIT&777/2>>	
1915	022036		30\$: BRWTE IMM,20	:MASK FOR INTERRUPT PENDING
(1)		001646	MICPC=MICPC+1	
(1)	022036	000420	<MOVE!WRITEBR!IMM!<20>>	
1916	022040		SP DP,AORB,SP1	:UPDATE PORT STATUS WORD
(1)		001647	MICPC=MICPC+1	
(1)	022040	063301	<MOVE!SPX!DP!AORB!SP1>	
1924	022042		ALWAYS FLUSH	
(1)		001650	MICPC=MICPC+1	
(1)	022042	104415	<JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>	
1926	022044		40\$: MEM IMM,INTSTK	:POINT TO START OF INTERRUPT STACK
(1)		001651	MICPC=MICPC+1	
(1)	022044	002542	<MOVE!WRMEM!IMM!<INTSTK>>	
1927	022046		ALWAYS 20\$	
(1)		001652	MICPC=MICPC+1	
(1)	022046	114641	<JUMP!ALCOND!<20\$-INIT&3000*4>!<20\$-INIT&777/2>>	
1928	022050		50\$: BRWTE IMM,RCL1	:POINT TO START OF RECEIVE QUEUE
(1)		001653	MICPC=MICPC+1	
(1)	022050	000424	<MOVE!WRITEBR!IMM!<RCL1>>	
1929	022052		SP BR SELB,SP14	
(1)		001654	MICPC=MICPC+1	
(1)	022052	063234	<MOVE!SPX!BR!SELB!SP14>	
1930	022054		ALWAYS 30\$	
(1)		001655	MICPC=MICPC+1	
(1)	022054	114646	<JUMP!ALCOND!<30\$-INIT&3000*4>!<30\$-INIT&777/2>>	
1931			.DSABL LSB	
1932	022056		NTHR5: LDMA IMM,ST	
(1)		001656	MICPC=MICPC+1	
(1)	022056	010152	<MOVE!LDMAR!IMM!<ST&377>>	
1933	022060		SPBR MEMX,SELB,SP0	
(1)		001657	MICPC=MICPC+1	
(1)	022060	043620	<MOVE!SPBRX!MEMX!SELB!SP0>	
1934	022062		BRWTE BR,ADD!SP0	:SHIFT LEFT
(1)		001660	MICPC=MICPC+1	
(1)	C ?	060400	<MOVE!WRITEBR!BR!ADD!SP0>	
1935	C		BR4 OVRUN	
(1)		001661	MICPC=MICPC+1	
(1)	022064	117174	<JUMP!BR4CON!<OVRUN-INIT&3000*4>!<OVRUN-INIT&777/2>>	
1936	022066		BRWTE IMM,1	
(1)		001662	MICPC=MICPC+1	
(1)	022066	000401	<MOVE!WRITEBR!IMM!<1>>	
1937	022070		ERRXX:	
1938	022070		NTFR5: LDMA IMM,((RTHR5+3	

(1) 001663
(1) 022070 010177
1939 022072
(1) 001664
(1) 022072 016400
1940 022074
(1) 001665
(1) 022074 062620
1941 022076
(1) 001666
(1) 022076 010241
1942 022100
(1) 001667
(1) 022100 053220
1943 022102
(1) 001670
(1) 022102 016601
1944 022104
(1) 001671
(1) 022104 002574
1945 022106
(1) 001672
(1) 022106 010241
1946 022110
(1) 001673
(1) 022110 002642
1947 022112
(1) 001674
(1) 022112 000776
1948 022114
(1) 001675
(1) 022114 060360
1949 022116
(1) 001676
(1) 022116 115701
1950 022120
(1) 001677
(1) 022120 000402
1951 022122
(1) 001700
(1) 022122 062400
1952 022124
(1) 001701
(1) 022124 000420
1953 022126
(1) 001702
(1) 022126 063701
1954 022130
(1) 001703
(1) 022130 116334
1955 022132
(1) 001704
(1) 022132 010171
1956 022134
(1) 001705
(1) 022134 043231

```

MICPC=MICPC+1
<MOVE!LDMAR!IMM!(<<RTHRS+3>&377)>>
MEMINC IMM,0
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!(<0>>)
MEM BR,SELB
MICPC=MICPC+1
<MOVE!WRMEM!BR!(<SELB>>)
NTRS1: LDMA IMM,NXTSP
MICPC=MICPC+1
<MOVE!LDMAR!IMM!(<NXTSP&377)>>
LDMA MEMX,SELB!SPX!SPO
MICPC=MICPC+1
<MOVE!LDMAR!MEMX!(<SELB!SPX!SPO>>)
MEMINC IMM,201
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!IMM!(<201>>)
MEM IMM,<<RTHRS>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!(<<RTHRS>>>)
LDMA IMM,NXTSP
MICPC=MICPC+1
<MOVE!LDMAR!IMM!(<NXTSP&377)>>
MEM IMM,INTSTK ;ASSUME QUEUE WRAP AROUND
MICPC=MICPC+1
<MOVE!WRMEM!IMM!(<INTSTK>>)
BRWRT 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>>)
BRWRT 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>)
NTRS2: BRWRT IMM,20
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!(<20>>)
SPBR BR,AORB,SP1
MICPC=MICPC+1
<MOVE!SPBRX!BR!AORB!SP1>
BRD TAB1 ;FLAGGED BY ERROR TYPE
MICPC=MICPC+1
<JUMP!BRCOND!(<TAB1-INIT&3000*4>!<TAB1-INIT&777/2>>)
SNAK: LDMA IMM,ISP11
MICPC=MICPC+1
<MOVE!LDMAR!IMM!(<ISP11&377)>>
SP MEMX,SELB,SP11
MICPC=MICPC+1
<MOVE!SPX!MEMX!SELB!SP1:

```

1957	022136	001706	SP BR, INCA, SP11	: INCREMENT MSG EXPECTED
(1)		063071	MICPC=MICPC+1	
(1)	022136		<MOVE!SPX!BR!INCA!SP11>	
1958	022140	001707	SNAK1: BRWRT IMM,1	: UNNUMB PENING BIT TO BR
(1)		000401	MICPC=MICPC+1	
(1)	022140		<MOVE!WRTEBR!IMM!<1>>	
1959	022142	001710	SNAK2: SP BR, AORB, SP10	: UPDATE LINE STATUS WORD
(1)		063310	MICPC=MICPC+1	
(1)	022142		<MOVE!SPX!BR!AORB!SP10>	
1960	022144	001711	ALWAYS FLUSH	
(1)		104415	MICPC=MICPC+1	
(1)	022144		<JUMP!ALCOND!<FLUSH-INIT&3000*4>!<FLUSH-INIT&777/2>>	
1961				
1962	022146	001712	EMTRIG: BRWRT IMM,24	
(1)		000424	MICPC=MICPC+1	
(1)	022146		<MOVE!WRTEBR!IMM!<24>>	
1963	022150	001713	OUTPUT BR, <SELB!OBA1>	
(1)		062226	MICPC=MICPC+1	
(1)	022150		<MOVE!WROUT!BR!<SELB!OBA1>>	
1964	022152	001714	BRWRT IMM,0	
(1)		000400	MICPC=MICPC+1	
(1)	022152		<MOVE!WRTEBR!IMM!<0>>	
1965	022154	001715	OUTPUT BR, <SELB!OBA2>	
(1)		062227	MICPC=MICPC+1	
(1)	022154		<MOVE!WROUT!BR!<SELB!OBA2>>	
1966	022156	001716	SPBR IBUS, BM873, SPO	: READ BM873 ADDRESS---
(1)		023740	MICPC=MICPC+1	
(1)	022156		<MOVE!SPBRX!IBUS!BM873!SPO>	
1967	022160	001717	OUTPUT BR, SELB!OUTDA1	: SET UP LOW BYTE OF ADDRESS
(1)		062222	MICPC=MICPC+1	
(1)	022160		<MOVE!WROUT!BR!<SELB!OUTDA1>>	
1968	022162	001720	BRWRT IMM,366	: HIGH BYTE BASE FOR ROM BOOT
(1)		000766	MICPC=MICPC+1	
(1)	022162		<MOVE!WRTEBR!IMM!<366>>	
1969	022164	001721	OUTPUT BR, SELB!OUTDA2	:
(1)		062223	MICPC=MICPC+1	
(1)	022164		<MOVE!WROUT!BR!<SELB!OUTDA2>>	
1970	022166	001722	EM6: BRWRT IMM,21	: MASK FOR TIMER AND ALSO TO START NPR
(1)		000421	MICPC=MICPC+1	
(1)	022166		<MOVE!WRTEBR!IMM!<21>>	
1971	022170	001723	OUT BR, <SELB!OBR>	
(1)		061231	MICPC=MICPC+1	
(1)	022170		<MOVE!WROUTX!BR!<SELB!OBR>>	
1972	022172	001724	OUT BR, <SELB!ONPR>	
(1)		061230	MICPC=MICPC+1	
(1)	022172		<MOVE!WROUTX!BR!<SELB!ONPR>>	
1973	022174	001725	EM1: BRWRT IBUS, NPR	: READ NPR CONTROL
(1)		120600	MICPC=MICPC+1	
(1)	022174		<MOVE!WRTEBR!IBUS!<NPR>>	
1974	022176	001726	BRO CKTIME	
(1)		102371	MICPC=MICPC+1	
(1)	022176		<JUMP!BROCON!<CKTIME-INIT&3000*4>!<CKTIME-INIT&777/2>>	
1975	022200	001727	MEMADR RM1	: IF NPR DONE
(1)		002420	MICPC=MICPC+1	
(1)	022200		<MOVE!WRMEM!RM1-INIT&777/2>>	
1976	022202		ALWAYS ACLOW	

```

(1) 001730
(1) 022202 100764
1977 022204
(1) 001731
(1) 022204 023640
1979 022206
(1) 001732
(1) 022206 060400
1979 022210
(1) 001733
(1) 022210 103451
1980 022212
(1) 001734
(1) 022212 010154
1981 022214
(1) 001735
(1) 022214 000402
1982 022216
(1) 001736
(1) 022216 076670
1983 022220
(1) 001737
(1) 022220 076611
1984 022222
(1) 001740
(1) 022222 076612
1985 022224
(1) 001741
(1) 022224 076614
1986 022226
(1) 001742
(1) 022226 076616
1987 022230
(1) 001743
(1) 022230 076617
1988
1989 022232
(1) 001744
(1) 022232 000460
1990 022234
(1) 001745
(1) 022234 010210
1991 022236
(1) 022236
(2) 001746
(2) 022236 002774
1992 022240
(1) 001747
(1) 022240 003004
1993
1994
1995 022242
(1) 001750
(1) 022242 010017
1996 022244
(1) 001751

```

```

MICPC=MICPC+1
<JUMP!ALCOND!<ACLOW-INIT&3000*4>!<ACLOW-INIT&777/2>>
TABUPD: SPBR IBUS,RCVCON,SP0 ;READ RECEIVER CONTROL REG
MICPC=MICPC+1
<MOVE!SPBRX!IBUS!RCVCON!SP0>
BRWRT BR,ADD!SP0 ;SHIFT LEFT
MICPC=MICPC+1
<MOVE!WRTEBR!BR!<ADD!SP0>>
BR7 IDLE ;RECEIVE ACTIVE--IDLE
MICPC=MICPC+1
<JUMP!BR7CON!<IDLE-INIT&3000*4>!<IDLE-INIT&777/2>>
TAB1: LDMA IMM,IMG10
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<IMG10&377>>
BRWRT IMM,2
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<2>>
MEMINC BR,AANDB!SP10 ;SAVE BIT 1 OF SP10
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<AANDB!SP10>>
MEMINC BR,SELA!SP11 ;SAVE SP11
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<SELA!SP11>>
MEMINC BR,SELA!SP12 ;SAVE SP12
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<SELA!SP12>>
MEMINC BR,SELA!SP14 ;SAVE SP14
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<SELA!SP14>>
MEMINC BR,SELA!SP16 ;SAVE SP16
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<SELA!SP16>>
MEMINC BR,SELA!SP17 ;SAVE SP17
MICPC=MICPC+1
<MOVE!WRMEM!INCMAR!BR!<SELA!SP17>>
;MAR NOW POINTS TO BASE
STATE RB2 ;DO NOT CHANGE BR UNTIL RBC
MICPC=MICPC+1
<MOVE!WRTEBR!IMM!<RB2-INIT&777/2>>
LDMA IMM,TABST ;POINT TO TABLE UPDATE STATE
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<TABST&377>>
PSTATE TBU1 ;NEW PORT STATE ADDRESS
MEM IMM,<<TBU1-INIT&777/2>>
MICPC=MICPC+1
<MOVE!WRMEM!IMM!<<TBU1-INIT&777/2>>>
SP IMM,4,SP4 ;INITIALIZE COUNT
MICPC=MICPC+1
<MOVE!SPX!IMM!4!SP4>
;NOTE: FIRST 6 RAM LOCATIONS ARE NOT WRITTEN
;TO CORE TABLE.
LDMA IMM,BASE
MICPC=MICPC+1
<MOVE!LDMAR!IMM!<BASE&377>>
ALWAYS RBO
MICPC=MICPC+1

```

(1) 022244 104442
1997 022246 001752
(1) 022246 000402
1998 022250 001753
(1) 022250 063004
1999 022252 001754
(1) 022252 023140
2000 022254 001755
(1) 022254 062006
2001 022256 001756
(1) 022256 023160
2002 022260 001757
(1) 022260 062107
2003 022262 001760
(1) 022262 105366
2004 022264 001761
(1) 022264 060601
2005 022266 001762
(1) 022266 116374
2006
2007 022270 001763
(1) 022270 070604
2008 022272 001764
(1) 022272 117371
2009 022274 001765
(1) 022274 056222
2010 022276 001766
(1) 022276 056223
2012 022300 001767
(1) 022300 123200
2014 022302 001770
(1) 022302 104622
2015 022304 001771
(1) 022304 010210
2016 022306 001772
(1) 022306 002460
(2) 022306 001773
2017 022310

```

EC2:  <JUMP!ALCOND!<RBO-INIT&3000*4>!<RBO-INIT&777/2>>
      BRWRTE IMM,2 ;INCREMENT COUNT/TEST
      MICPC=MICPC+1
      <MOVE!WRTEBR!IMM!<2>>
      SP BR,ADD,SP4
      MICPC=MICPC+1
      <MOVE!SPX!BR!ADD!SP4>
      SP IBUS IOBA1,SPO ;POINT TO NEXT ADDRESS
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IOBA1!SPO>
      OUTPUT BR,ADD!OBA1
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<ADD!OBA1>>
      SP IBUS IOBA2,SPO
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!IOBA2!SPO>
      OUTPUT BR,AC!OBA2
      MICPC=MICPC+1
      <MOVE!WROUT!BR!<AC!OBA2>>
      C TABMXT
      MICPC=MICPC+1
      <JUMP!CCOND!<TABMXT-INIT&3000*4>!<TABMXT-INIT&777/2>>
EC4:  BRWRTE BR,SELA!SP1 ;READ PORT STATUS
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!<SELA!SP1>>
      BRO 30$ ;INIT MODE, WRITE OUT 200 BYTES
      MICPC=MICPC+1
      <JUMP!BROCON!<30$-INIT&3000*4>!<30$-INIT&777/2>>
      ;OTHERWISE ONLY WRITE OUT ERROR COUNTERS
      BRWRTE BR!LDMAR,SELA!SP4 ;READ COUNTER
      MICPC=MICPC+1
      <MOVE!WRTEBR!BR!LDMAR!<SELA!SP4>>
      BR4 20$ ;ALL DONE
      MICPC=MICPC+1
      <JUMP!BR4CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>
10$:  OUTPUT MEMX!INCMAR,SELB!OUTDA1 ;STORE COUNTS OF ERRORS
      MICPC=MICPC+1
      <MOVE!WROUT!MEMX!INCMAR!<SELB!OUTDA1>>
      OUTPUT MEMX!INCMAR,SELB!OUTDA2
      MICPC=MICPC+1
      <MOVE!WROUT!MEMX!INCMAR!<SELB!OUTDA2>>
      SP IBUS,NPR,SPO
      MICPC=MICPC+1
      <MOVE!SPX!IBUS!NPR!SPO>
      ALWAYS RKB
      MICPC=MICPC+1
      <JUMP!ALCOND!<RKB-INIT&3000*4>!<RKB-INIT&777/2>>
20$:  LDMA IMM,TABST
      MICPC=MICPC+1
      <MOVE!LDMAR!IMM!<TABST&377>>
      PSTATE I3
      MEM IMM, <<I3-INIT&777/2>>
      MICPC=MICPC+1
      <MOVE!WRMEM!IMM!<<I3-INIT&777/2>>>
      ALWAYS RMI
      MICPC=MICPC+1
    
```

```

(1) 022310 104420
2018 022312 001774
(1) 022312 070604
2019 022314 001775
(1) 022314 117771
2020 022316 001776
(1) 022316 114765
2021 022320 001777
(1) 022320 000000
2022
2023 000001

```

```

30$: <JUMP!ALCOND!<RM1-INIT&3000*4>!<RM1-INIT&777/2>>
BRWRITE BR!LDMAR,SELA!SP4 ;READ COUNTER
MICPC=MICPC+1
<MOVE!WRITEBR!BR!LDMAR!<SELA!SP4>>
BR7 20$ ;ALL DONE
MICPC=MICPC+1
<JUMP!BR7CON!<20$-INIT&3000*4>!<20$-INIT&777/2>>
ALWAYS 10$ ;KEEP GOING
MICPC=MICPC+1
<JUMP!ALCOND!<10$-INIT&3000*4>!<10$-INIT&777/2>>
$ZERO
MICPC=MICPC+1
000000
:
:END

```

. ABS. 022322 000

ERRORS DETECTED: 0

```

DDCMP/CRF/DS:CRF+DMCHGH,HILOW,DDCHGH
RUN-TIME: 5 8 0 SECONDS
RUN-TIME RATIO: 68/13=4.9
CORE USED: 6K (11 PAGES)

```

157:

03000

K03

02800

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

TEST 1

TST1: MOV #1,TSTNO
MOV #TST2,NEXT
TST ROMNUM ;R1 CONTAINS BASE DMC11 ADDRESS
BNE 1\$;FIRST TIME HERE?
TYPE, ROM1 ;SKIP IF NOT
1\$: MOV #-1,ROMNUM ;TYPE PART NUMBERS
SCOPE ;SET FLAG TO ONLY TYPE ONCE
ROMNUM: 0
ROM1: .ASCII <377><12>/DZDMG-C SUPPORTS THE FOLLOWING CROM PART NUMBERS:
.ASCII <377><12>/DMC11-AR (M8200-YA)/
.ASCII <377>/23-414A9/<15><12>/23-415A9/<15><12>/23-416A9/
.ASCII <15><12>/23-417A9/<15><12>/23-418A9/<15><12>/23-419A9/
.ASCII <15><12>/23-420A9/<15><12>/23-421A9/
.ASCII <377><12>/DMC11-AL (M8200-YB)/
.ASCII <377>/23-392A9/<15><12>/23-393A9/<15><12>/23-394A9/
.ASCII <15><12>/23-395A9/<15><12>/23-396A9/<15><12>/23-397A9/
.ASCII <15><12>/23-398A9/
.ASCII <15><12>/23-399A9/
.EVEN

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

TEST 2

TST2: MOV #2,TSTNO
MOV #TST3,NEXT
JSR PC,MAPCK ;R1 CONTAINS BASE DMC11 ADDRESS
BIT #BIT15,STAT1 ;CHECK FOR HI OR LO
BEQ 2\$;BE SURE DMC HAS CROM
CLR RO ;SKIP IF NO CROM
MOV ROMMAP,R2 ;RO=CROM ADDRESS
1\$: MOV #BIT10,(R1) ;R2 POINTS TO ROMMAP
;SET ROM0

1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685 022322 012737 000001 001226
1686 022330 012737 022760 001216
1687
1688 022336 005737 022360
1689 022342 001002
1690 022344 104402 022362
1691 022350 012737 177777 022360
1692 022356 104400
1693 022360 000000
1694 022362 005377 055104 046504
(1) 022445 377 042012 041515
(1) 022473 377 031462 032055
(1) 022530 005015 031462 032055
(1) 022566 005015 031462 032055
(1) 022612 005377 046504 030503
(1) 022640 031377 026463 034463
(1) 022675 015 031012 026463
(1) 022733 015 031012 026463
(1) 022745 015 031012 026463
(1)
(3)
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708 022760 012737 000002 001226
1709 022766 012737 023052 001216
1710
1711 022774 004737 027012
1712 023000 032737 100000 001366
1713 023006 001420
1714 023010 005000
1715 023012 013702 012320
1716 023016 012711 002000

L03

GENERAL UTILITIES (TYPEOUT, ERROR, SCOPE, ETC)

1717 023022 010061 000004
 1718 023026 012261 000006
 1719 023032 052711 020000
 1720 023036 005200
 1721 023040 022700 002(0)
 1722 023044 001364
 1723 023046 005011
 1724 023050 104400

```

MOV      R0,4(R1)      ;LOAD CRAM ADDRESS
MOV      (R2)+,6(R1)  ;LOAD WORD TO BE WRITTEN
BIS      #BIT13,(R1)  ;WRITE IT!
INC      R0            ;NEXT ADDRESS
CMP      #2000,R0     ;DONE YET?
BNE      1$          ;BR IF NO
CLR      (R1)         ;CLEAR SEL0
SCOPE    ;SCOPE THIS TEST
  
```

2\$:

```

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

; TEST 3

1735 023052 012737 000003 001226
 1736 023060 012737 023166 001216

TST3:

```

MOV      #3,TSTNO
MOV      #TST4,NEXT
; R1 CONTAINS BASE DMC11 ADDRESS
MSTCLR  ;MASTER CLEAR DMC11
MSTCLR  ;MASTER CLEAR DMC11
MOV      DMCSR,R1   ;R1 = DMC BASE ADDRESS
CLR      (R1)       ;CLEAR SEL0
MOV      #52525,R5  ;START WITH 125
MOV      R5,4(R1)  ;PORT4+125
ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
120500  ;BR ← PORT4
ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
061620  ;BR RSH+BR, SHFT BR RIGHT
ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
061225  ;PORT5+BR
ROR      R5         ;R5 = "EXPECTED"
MOV      5(R1),R4   ;R4 = "FOUND"
CMP      R5,R4      ;DID BR SHIFT RIGHT ONCE?
BEQ      1$         ;BR IF YES
HLT      12        ;BR RIGHT SHIFT ERROR
  
```

1\$:

1756 023140 104414
 1757 023142 061620
 1758 023144 104414
 1759 023146 061225
 1760 023150 006005
 1761 023152 116104 000005
 1762 023156 120504
 1763 023160 001401
 1764 023162 104012
 1765 023164 104400

```

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"
CMP      R5,R4      ;DID BR SHIFT RIGHT?
BEQ      2$         ;BR IF YES
HLT      12        ;BR RIGHT SHIFT ERROR
SCOPE    ;SCOPE THIS TEST
  
```

2\$:

```

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

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
1792
1793
1794
1795
1796
1797
1798
1799 023166 012737 000004 001226
1800 023174 012737 023362 001216
1801 023202 012737 023240 001220
1802
1803 023210 104412
1804 023212 032737 100000 001366
1805 023220 001057
1806 023222 004737 027012
1807 023226 005011
1808 023230 013700 012320
1809 023234 005002
1810 023236 005003
1811 023240 042737 014377 023260
1812 023246 050237 023260
1813 023252 050337 023260
1814 023256 104414
1815 023260 100400
1816 023262 012711 002000
1817 023266 011005
1818 023270 016104 000006
1819 023274 020504
1820 023276 001414
1821 023300 010337 001252
1822 023304 000241
1823 023306 006037 001252
1824 023312 006037 001252
1825 023316 006037 001252
1826 023322 050237 001252
1827 023326 104004
1828 023330 104401

```

```

: *IF THIS TEST FAILS CHECK YOUR CROM PART NUMBERS.
: *DZDMG-C SUPPORTS THE FOLLOWING PART NUMBERS:
: *
: *DMC11-AR (M8200-YA)
: * 23-414A9
: * 23-415A9
: * 23-416A9
: * 23-417A9
: * 23-418A9
: * 23-419A9
: * 23-420A9
: * 23-421A9
: *
: *DMC11-AL (M8200-YB)
: * 23-392A9
: * 23-393A9
: * 23-394A9
: * 23-395A9
: * 23-396A9
: * 23-397A9
: * 23-398A9
: * 23-399A9
: *****

```

```

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

MSTCLR
BIT #BIT15,STAT1
BNE 4$
JSR PC,MAPCK
CLR (R1)
MOV ROMMAP,R0
CLR R2
CLR R3
1$: BIC #14377.2$,
BIS R2,2$
BIS R3,2$
ROMCLK
2$: 100400
MOV #BIT10,(R1)
MOV (R0),R5
MOV 6(R1),R4
CMP R5,R4
BEQ 3$
MOV R3,TEMP3
CLC
ROR TEMP3
ROR TEMP3
ROR TEMP3
BIS R2,TEMP3
HLT 4
3$: SCOP1

;R1 CONTAINS BASE DMC11 ADDRESS
;MASTER CLEAR DMC11
;IS IT RAM OR ROM
;SKIP TEST IF CROM
;CHECK FOR HI OR LO
;CLEAR RUN
;R0 POINTS TO SOFTWARE ROM MAP
;R2 CONTAINS ROM ADDRESS BITS 0-7
;R3 CONTAINS ROM ADDRESS BITS 9&10 IN BITS 11&12
;CLEAR ADDRESS FIELDS OF INSTRUCTION
;ADD BITS 0-7 TO INSTRUCTION
;ADD BITS 11&12 TO INSTRUCTION
;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
;JUMP(I) TO ROM ADDRESS IN R2 & R3
;SET ROMO
;PUT "EXPECTED" IN R5
;PUT "FOUND" IN R4
;COMPARE ROM CONTENTS TO SOFT COP
;BR IF OK
;PUT ROM ADDRESS IN TEMP3
;FOR ERROR TYPEOUT

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

```

1829	023332	005720			TST	(R0)+		;BUMP SOFT POINTER
1830	023334	005202			INC	R2		;BUMP ROM ADDRESS
1831	023336	022702	C00400		CMP	#400,R2		;IS R2 TO MAX YET?
1832	023342	001336			BNE	1\$;BR IF NO
1833	023344	005002			CLR	R2		;YES, RESET R2 TO 0
1834	023346	062703	004000		ADD	#4000,R3		;INC TO NEXT PAGE OF ROM
1835	023352	022703	020000		CMP	#20000,R3		;DONE YET?
1836	023356	001330			BNE	1\$;BR IF NO
1837	023360	104400		4\$:	SCOPE			;SCOPE THIS TEST
1838								
1839								
1840								;***** TEST 5 *****
1841								;*CROM TEST OF JUMP(I) NEVER MICRO-PROCESSOR INSTRUCTION.
1842								;*PERFORM THE JUMP INSTRUCTION
1843								;*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
1844								;*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE..
1845								;*****
1846								
1847								; TEST 5
1848								
1849	023362	012737	000005	001226	TST5:	MOV	#5,TSTNO	
1850	023370	012737	023556	001216		MOV	#TST6,NEXT	
1851	023376	012737	023422	001220		MOV	#1\$,LOCK	
1852								;R1 CONTAINS BASE DMC11 ADDRESS
1853	023404	104412				MSTCLR		;MASTER CLEAR DMC11
1854	023406	032737	103000	001366		BIT	#BIT15,STAT1	;IS IT CROM?
1855	023414	001057				BNE	6\$+2	;SKIP TEST IF YES
1856	023416	004737	027012			JSR	PC,MAPCK	;CHECK FOR HI OR LO
1857	023422				1\$:			
1858	023422	004737	026656			JSR	PC,CLRALL	;CLEAR ALL CONDITIONS
1859	023426	104414				ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1860	023430	100403				100400		;START AT ROM PC=0
1861	023432	104414				ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1862	023434	114377				114377! <400*0>		;JUMP TO ROM PC OF 1777
1863	023436	004737	026750			JSR	PC,ROMDAT	;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1864	023442	000002				2		;INDEX
1865	023444	020504				CMP	R5,R4	;ARE NEW PC CONTENTS CORRECT?
1866	023446	001401				BEQ	2\$;BR IF YES
1867	023450	104006				HLT	6	;ERROR, CROM PC IS WRONG
1868	023452	104401			2\$:	SCOPI		;LOOP TO 1\$ IF SW09=1
1869	023454	012737	023462	001220		MOV	#3\$,LOCK	;NEW SCOPI
1870	023462				3\$:			
1871	023462	004737	026656			JSR	PC,CLRALL	;CLEAR ALL CONDITIONS
1872	023466	104414				ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1873	023470	100403				100403		;START AT ROM PC=3
1874	023472	104414				ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1875	023474	100000				100000! <400*0>		;JUMP TO ROM PC OF 0
1876	023476	004737	026750			JSR	PC,ROMDAT	;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1877	023502	000010				10		;INDEX
1878	023504	020504				CMP	R5,R4	;ARE NEW PC CONTENTS CORRECT?
1879	023506	001401				BEQ	4\$;BR IF YES
1880	023510	104006				HLT	6	;ERROR, CROM PC IS WRONG
1881	023512	104401			4\$:	SCOPI		;LOOP TO 3\$ IF SW09=1
1882	023514	012737	023522	001220		MOV	#5\$,LOCK	;NEW SCOPI
1883	023522				5\$:			
1884	023522	004737	026656			JSR	PC,CLRALL	;CLEAR ALL CONDITIONS

1885	023526	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1886	023530	100406			100406		:START AT ROM PC=6
1887	023532	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1888	023534	104125			104125! <400*0>		:JUMP TO ROM PC OF 525
1889	023536	004737	026750		JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1890	023542	000016			16		:INDEX
1891	023544	020504			CMP	R5,R4	:ARE NEW ROM PC CONTENTS CORRECT?
1892	023546	001401			BEG	6\$:BR IF YES
1893	023550	104006			HLT	6	:ERROR, CROM PC IS WRONG
1894	023552	104401		6\$:	SCOPI		:LOOP TO 5\$ IF SW59=1
1895	023554	104400			SCOPE		:SCOPE THIS TEST
1896							
1897							
1898							
1899							
1900							
1901							
1902							
1903							
1904							
1905							
1906	023556	012737	000006	001226	TST6:	MOV #6,TSTNO	
1907	023564	012737	023736	001216		MOV #TST7,NEXT	
1908	023572	012737	023616	001220		MOV #1\$,LOCK	
1909							
1910	023600	104412			MSTCLR		:R1 CONTAINS BASE DMC11 ADDRESS
1911	023602	032737	100000	001366	BIT	#BIT15,STAT1	:MASTER CLEAR DMC11
1912	023610	001051			BNE	6\$+2	:IS IT CROM?
1913	023612	004737	027012		JSR	PC,MAPCK	:SKIP TEST IF YES
1914	023616				1\$:		:CHECK FOR HI OR LO
1915	023616	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1916	023620	100400			100400		:START AT ROM PC=0
1917	023622	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1918	023624	114777			114377! <400*1>		:JUMP TO ROM PC OF 1777
1919	023626	004737	026750		JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1920	023632	003776			3776		:INDEX
1921	023634	020504			CMP	R5,R4	:ARE NEW PC CONTENTS CORRECT?
1922	023636	001401			BEG	2\$:BR IF YES
1923	023640	104006			HLT	6	:ERROR, CROM PC IS WRONG
1924	023642	104401			2\$:	SCOPI	:LOOP TO 1\$ IF SW09=1
1925	023644	012737	023652	001220	MOV	#3\$,LOCK	:NEW SCOPI
1926	023652				3\$:		
1927	023652	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1928	023654	100403			100403		:START AT ROM PC=3
1929	023656	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1930	023660	100400			100000! <400*1>		:JUMP TO ROM PC OF 0
1931	023662	004737	026750		JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1932	023666	000000			0		:INDEX
1933	023670	020504			CMP	R5,R4	:ARE NEW PC CONTENTS CORRECT?
1934	023672	001401			BEG	4\$:BR IF YES
1935	023674	104006			HLT	6	:ERROR, CROM PC IS WRONG
1936	023676	104401			4\$:	SCOPI	:LOOP TO 3\$ IF SW09=1
1937	023700	012737	023706	001220	MOV	#5\$,LOCK	:NEW SCOPI
1938	023706				5\$:		
1939	023706	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1940	023710	100406			100406		:START AT ROM PC=6

1941	023712	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1942	023714	104525			104125!(<400*1)		:JUMP TO ROM PC OF 525
1943	023716	004737	026750		JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1944	023722	001252			1252		:INDEX
1945	023724	020504			CMP	R5,R4	:ARE NEW ROM PC CONTENTS CORRECT?
1946	023726	001401			BEQ	6\$:BR IF YES
1947	023730	104006			HLT	6	:ERROR, CROM PC IS WRONG
1948	023732	104401		6\$:	SCOPI		:LOOP TO 5\$ IF SW59=1
1949	023734	10440C			SCOPE		:SCOPE THIS TEST
1950							
1951							
1952							
1953							
1954							
1955							
1956							
1957							
1958							
1959							
1960	023736	012737	000007	001226	TEST:	MOV	#7,TSTMC
1961	023744	012737	024132	001216		MOV	#TST10,NEXT
1962	023752	012737	023776	001220		MOV	#1\$,LOCK
1963							
1964	023760	104412			MSTCLR		:RI CONTAINS BASE DMC11 ADDRESS
1965	023762	032737	100000	001366	BIT	#BIT15,STAT1	:MASTER CLEAR DMC11
1966	023770	001057			BNE	6\$+2	:IS IT CROM?
1967	023772	004737	027012		JSR	PC,MAPCK	:SKIP TEST IF YES
1968	023776				1\$:		:CHECK FOR HI OR LO
1969	023776	004737	026724		JSR	PC,SETC	:SET THE C BIT
1970	024002	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1971	024004	100400			100400		:START AT ROM PC=C
1972	024006	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1973	024010	115377			114377!(<400*2)		:JUMP TO ROM PC OF 1777
1974	024012	004737	026750		JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1975	024016	003776			3776		:INDEX
1976	024020	020504			CMP	R5,R4	:ARE NEW PC CONTENTS CORRECT?
1977	024022	001401			BEQ	2\$:BR IF YES
1978	024024	104006			HLT	E	:ERROR, CROM PC IS WRONG
1979	024026	104401		2\$:	SCOPI		:LOOP TO 1\$ IF SW09=1
1980	024030	012737	024036	001220	MOV	#3\$,LOCK	:NEW SCOPI
1981	024036				3\$:		
1982	024036	004737	026724		JSR	PC,SETC	:SET THE C BIT
1983	024042	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1984	024044	100403			100403		:START AT ROM PC=3
1985	024046	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1986	024050	101000			100000!(<400*2)		:JUMP TO ROM PC OF 0
1987	024052	004737	026750		JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
1988	024056	000000			C		:INDEX
1989	024060	020504			CMP	R5,R4	:ARE NEW PC CONTENTS CORRECT?
1990	024062	001401			BEQ	4\$:BR IF YES
1991	024064	104006			HLT	6	:ERROR, CROM PC IS WRONG
1992	024066	104401		4\$:	SCOPI		:LOOP TO 3\$ IF SW09=1
1993	024070	012737	024076	001220	MOV	#5\$,LOCK	:NEW SCOPI
1994	024076				5\$:		
1995	024076	004737	026724		JSR	PC,SETC	:SET THE C BIT
1996	024102	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

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

```

: TEST 7

```

-----
MOV #7,TSTMC
MOV #TST10,NEXT
MOV #1$,LOCK

```

```

MSTCLR ;RI CONTAINS BASE DMC11 ADDRESS
BIT #BIT15,STAT1 ;MASTER CLEAR DMC11
BNE 6$+2 ;IS IT CROM?
JSR PC,MAPCK ;SKIP TEST IF YES
;CHECK FOR HI OR LO

```

```

JSR PC,SETC ;SET THE C BIT
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 ;START AT ROM PC=C
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 E ;ERROR, CROM PC IS WRONG
2$: SCOPI ;LOOP TO 1$ IF SW09=1
MOV #3$,LOCK ;NEW SCOPI

```

```

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
C ;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

```

```

JSR PC,SETC ;SET THE C BIT
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

```

1997 024104 100406 100406 ;START AT ROM PC=6
1998 024106 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
1999 024110 105125 104125!<400*2> ;JUMP TO ROM PC OF 525
2000 024112 004737 026750 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2001 024116 001252 1252 ;INDEX
2002 024120 020504 CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
2003 024122 001401 BEQ 6$ ;BR IF YES
2004 024124 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2005 024126 104401 6$: SCOPI ;LOOP TO 5$ IF SW59=1
2006 024130 104400 SCOPE ;SCOPE THIS TEST
2007
2008
2009 ;***** TEST 10 *****
2010 ;*CROM TEST OF JUMP(I) ON Z BIT SET MICRO-PROCESSOR INSTRUCTION.
2011 ;*SET THE Z BIT, PERFORM THE JUMP INSTRUCTION.
2012 ;*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
2013 ;*****
2014
2015 ; TEST 10
2016
2017 024132 012737 000010 001226 TSTIC: MOV #10,TSTNO
2018 024140 012737 024326 001216 MOV #TST11,NEXT
2019 024146 012737 024172 001220 MOV #1$,LOCK
2020
2021 024154 104412 MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
2022 024156 032737 100000 001366 BIT #BIT15,STAT1 ;MASTER CLEAR DMC11
2023 024164 001057 9NE 6$+2 ;IS IT CROM?
2024 024166 004737 027012 JSR PC,MAPCK ;SKIP TEST IF YES
2025 024172 15: JSR PC,SETZ ;SET THE Z BIT ;CHECK FOR HI OR LO
2026 024176 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2027 024200 100400 100400 ;START AT ROM PC=0
2028 024202 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2029 024204 115777 114377!<400*3> ;JUMP TO ROM PC OF 1777
2030 024206 004737 026750 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2031 024212 003776 3776 ;INDEX
2032 024214 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2033 024216 001401 BEQ 2$ ;BR IF YES
2034 024220 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2035 024222 104401 2$: SCOPI ;LOOP TO 1$ IF SW09=1
2036 024224 012737 024232 001220 MOV #3$,LOCK ;NEW SCOPI
2037
2038 3$:
2039 024232 004737 026742 JSR PC,SETZ ;SET THE Z BIT
2040 024236 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2041 024240 100403 100403 ;START AT ROM PC=3
2042 024242 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2043 024244 101400 100000!<400*3> ;JUMP TO ROM PC OF 0
2044 024246 004737 026750 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2045 024252 000000 0 ;INDEX
2046 024254 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2047 024256 001401 BEQ 4$ ;BR IF YES
2048 024260 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2049 024262 104401 4$: SCOPI ;LOOP TO 3$ IF SW09=1
2050 024264 012737 024272 001220 MOV #5$,LOCK ;NEW SCOPI
2051 024272 5$:
2052 024272 004737 026742 JSR PC,SETZ ;SET THE Z BIT

```

E04

2053	024276	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2054	024300	100406			100406		;START AT ROM PC=6
2055	024302	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2056	024304	105525			104125!(400*3)		;JUMP TO ROM PC OF 525
2057	024306	004737	026750		JSR	PC,ROMDAT	;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2058	024312	001252			1252		;INDEX
2059	024314	020504			CMP	R5,R4	;ARE NEW ROM PC CONTENTS CORRECT?
2060	024316	001401			BEQ	6\$;BR IF YES
2061	024320	104006			HLT	6	;ERROR, CROM PC IS WRONG
2062	024322	104401		6\$:	SCOPI		;LOOP TO 5\$ IF SW59=1
2063	024324	104400			SCOPE		;SCOPE THIS TEST
2064							
2065							
2066							;***** TEST 11 *****
2067							;*CROM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
2068							;*SET THE BRO BIT, PERFORM THE JUMP INSTRUCTION.
2069							;*VERIFY THE JUMP BY READING THE CONTENTS OF THE NEW ROM PC
2070							;*****
2071							
2072							; TEST 11
2073							
2074	024326	012737	000011	001226	TST11:	MOV	#11,TSTNO
2075	024334	012737	024522	001216		MOV	#TST12,NEXT
2076	024342	012737	024366	001220		MOV	#1\$,LOCK
2077							;R1 CONTAINS BASE DMC11 ADDRESS
2078	024350	104412			MSTCLR		;MASTER CLEAR DMC11
2079	024352	032737	100000	001366	BIT	#BIT15,STAT1	;IS IT CROM?
2080	024360	001057			BNE	6\$+2	;SKIP TEST IF YES
2081	024362	004737	027012		JSR	PC,MAPCK	;CHECK FOR HI OR LO
2082	024366				1\$:		
2083	024366	004737	026674		JSR	PC,SETBRO	;SET THE BRO BIT
2084	024372	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2085	024374	100400			100400		;START AT ROM PC=0
2086	024376	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2087	024400	116377			114377!(400*4)		;JUMP TO ROM PC OF 1777
2088	024402	004737	026750		JSR	PC,ROMDAT	;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2089	024406	003776			3776		;INDEX
2090	024410	020504			CMP	R5,R4	;ARE NEW PC CONTENTS CORRECT?
2091	024412	001401			BEQ	2\$;BR IF YES
2092	024414	104006			HLT	6	;ERROR, CROM PC IS WRONG
2093	024416	104401		2\$:	SCOPI		;LOOP TO 1\$ IF SW09=1
2094	024420	012737	024426	001220	MOV	#3\$,LOCK	;NEW SCOPI
2095	024426				3\$:		
2096	024426	004737	026674		JSR	PC,SETBRO	;SET THE BRO BIT
2097	024432	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2098	024434	100403			100403		;START AT ROM PC=3
2099	024436	104414			ROMCLK		;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2100	024440	102000			100000!(400*4)		;JUMP TO ROM PC OF 0
2101	024442	004737	026750		JSR	PC,ROMDAT	;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2102	024446	000000			0		;INDEX
2103	024450	020504			CMP	R5,R4	;ARE NEW PC CONTENTS CORRECT?
2104	024452	001401			BEQ	4\$;BR IF YES
2105	024454	104006			HLT	6	;ERROR, CROM PC IS WRONG
2106	024456	104401		4\$:	SCOPI		;LOOP TO 3\$ IF SW09=1
2107	024460	012737	024466	001220	MOV	#5\$,LOCK	;NEW SCOPI
2108	024466				5\$:		

F04

2109	024466	004737	026674	JSR	PC,SETBRO	:SET THE BRO BIT'
2110	024472	104414		ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2111	024474	100406		100406		:START AT ROM PC=6
2112	024476	104414		ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2113	024500	106125		104125!(400*4)		:JUMP TO ROM PC OF 525
2114	024502	004737	026750	JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2115	024506	001252		1252		:INDEX
2116	024510	020504		CMP	R5,R4	:ARE NEW ROM PC CONTENTS CORRECT'
2117	024512	001401		BEQ	6\$:BR IF YES
2118	024514	104006		HLT	6	:ERROR, CROM PC IS WRONG
2119	024516	104401		6\$:	SCOPE	:LOOP TO 5\$ IF SW59=1
2120	024520	104400		SCOPE		:SCOPE THIS TEST

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

: TEST 12

2131	024522	012737	000012	001226	TST12:	MOV	#12,TSTNO	
2132	024530	012737	024716	001216		MOV	#TST13,NEXT	
2133	024536	012737	024562	001220		MOV	#15,LOCK	
2134								:R1 CONTAINS BASE DMC!! ADDRESS
2135	024544	104412				MSTCLR		:MASTER CLEAR DMC!!
2136	024546	032737	100000	001366		BIT	#BIT15,STAT1	:IS IT CROM?
2137	024554	001057				BNE	6\$+2	:SKIP TEST IF YES
2138	024556	004737	027012			JSR	PC,MAPCK	:CHECK FOR HI OR LO
2139	024562				1\$:			
2140	024562	004737	026702			JSR	PC,SETBRI	:SET THE BRI BIT'
2141	024566	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2142	024570	100400				100400		:START AT ROM PC=0
2143	024572	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2144	024574	116777				114377!(400*5)		:JUMP TO ROM PC OF 1777
2145	024576	004737	026750			JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2146	024602	003776				3776		:INDEX
2147	024604	020504				CMP	R5,R4	:ARE NEW PC CONTENTS CORRECT?
2148	024606	001401				BEQ	2\$:BR IF YES
2149	024610	104006				HLT	6	:ERROR, CROM PC IS WRONG
2150	024612	104401			2\$:	SCOPE		:LOOP TO 1\$ IF SW09=1
2151	024614	012737	024622	001220		MOV	#3\$,LOCK	:NEW SCOPE
2152	024622				3\$:			
2153	024622	004737	026702			JSR	PC,SETBRI	:SET THE BRI BIT'
2154	024626	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2155	024630	100403				100403		:START AT ROM PC=3
2156	024632	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2157	024634	102400				100000!(400*5)	:JUMP TO	:JUMP TO ROM PC OF 0
2158	024636	004737	026750			JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2159	024642	000000				0		:INDEX
2160	024644	020504				CMP	R5,R4	:ARE NEW PC CONTENTS CORRECT?
2161	024646	001401				BEQ	4\$:BR IF YES
2162	024650	104006				HLT	6	:ERROR, CROM PC IS WRONG
2163	024652	104401			4\$:	SCOPE		:LOOP TO 3\$ IF SW09=1
2164	024654	012737	024662	001220		MOV	#5\$,LOCK	:NEW SCOPE

```

2165 024662          5$: JSR    PC,SETBR1      ;SET THE BR1 BIT'
2166 024662 004737 026702 ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2167 024666 104414          ROMCLK  ;START AT ROM PC=6
2168 024670 100406          ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2169 024672 104414          ROMCLK  ;JUMP TO ROM PC OF 525
2170 024674 106525          JSR    PC,ROMDAT     ;R5=EXPECTED ROM DATA,R4=ACTJAL ROM DATA
2171 024676 004737 026750 1252      ;INDEX
2172 024702 001252          CMP    R5,R4        ;ARE NEW ROM PC CONTENTS CORRECT?
2173 024704 020504          BEQ    6$           ;BR IF YES
2174 024706 001401          HLT    6            ;ERROR, CROM PC IS WRONG
2175 024710 104006          SCOPE  ;LOOP TO 5$ IF SW59=1
2176 024712 104401          SCOPE  ;SCOPE THIS TEST
2177 024714 104400
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188 024716 012737 000013 001226 TST13: MOV    #13,TSTNO
2189 024724 012737 025112 001216      MOV    #TST14,NEXT
2190 024732 012737 024756 001220      MOV    #1$,LOCK
2191
2192 024740 104412          MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
2193 024742 032737 100000 001366      BIT    #BIT15,STAT1 ;MASTER CLEAR DMC11
2194 024750 001057          BNE    6$+2        ;IS IT CROM?
2195 024752 004737 027012          JSR    PC,MAPCK    ;SKIP TEST IF YES
2196 024756          1$: JSR    PC,SETBR4    ;CHECK FOR HI OR LO
2197 024756 004737 026710          JSR    PC,SETBR4    ;SET THE BR4 BIT'
2198 024762 104414          ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2199 024764 100400          ROMCLK  ;START AT ROM PC=0
2200 024766 104414          ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2201 024770 117377          JSR    PC,ROMDAT     ;JUMP TO ROM PC OF 1777
2202 024772 004737 026750          3776    ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2203 024776 003776          ;INDEX
2204 025000 020504          CMP    R5,R4        ;ARE NEW PC CONTENTS CORRECT?
2205 025002 001401          BEQ    2$           ;BR IF YES
2206 025004 104006          HLT    6            ;ERROR, CROM PC IS WRONG
2207 025006 104401          SCOPE  ;LOOP TO 1$ IF SW09=1
2208 025010 012737 025016 001220      MOV    #3$,LOCK    ;NEW SCOPE
2209 025016          3$: JSR    PC,SETBR4    ;SET THE BR4 BIT'
2210 025016 004737 026710          JSR    PC,SETBR4    ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2211 025022 104414          ROMCLK  ;START AT ROM PC=3
2212 025024 100403          ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2213 025026 104414          ROMCLK  ;JUMP TO ROM PC OF 0
2214 025030 103000          JSR    PC,ROMDAT     ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2215 025032 004737 026750          100000! ;JUMP TO ROM PC OF 0
2216 025036 000000          ;INDEX
2217 025040 020504          CMP    R5,R4        ;ARE NEW PC CONTENTS CORRECT?
2218 025042 001401          BEQ    4$           ;BR IF YES
2219 025044 104006          HLT    6            ;ERROR, CROM PC IS WRONG
2220 025046 104401          SCOPE  ;LOOP TO 3$ IF SW09=1

```

```

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

```

: TEST 13

```

-----
TST13: MOV    #13,TSTNO
        MOV    #TST14,NEXT
        MOV    #1$,LOCK

```

```

MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
BIT    #BIT15,STAT1 ;MASTER CLEAR DMC11
BNE    6$+2        ;IS IT CROM?
JSR    PC,MAPCK    ;SKIP TEST IF YES
;CHECK FOR HI OR LO

```

```

1$: JSR    PC,SETBR4    ;SET THE BR4 BIT'
    ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
    100400  ;START AT ROM PC=0
    ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
    114377! ;JUMP TO ROM PC OF 1777
    JSR    PC,ROMDAT     ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
    3776    ;INDEX
    CMP    R5,R4        ;ARE NEW PC CONTENTS CORRECT?
    BEQ    2$           ;BR IF YES
    HLT    6            ;ERROR, CROM PC IS WRONG
2$: SCOPE  ;LOOP TO 1$ IF SW09=1
    MOV    #3$,LOCK    ;NEW SCOPE

```

```

3$: JSR    PC,SETBR4    ;SET THE BR4 BIT'
    ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
    100403  ;START AT ROM PC=3
    ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
    100000! ;JUMP TO ROM PC OF 0
    JSR    PC,ROMDAT     ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
    0       ;INDEX
    CMP    R5,R4        ;ARE NEW PC CONTENTS CORRECT?
    BEQ    4$           ;BR IF YES
    HLT    6            ;ERROR, CROM PC IS WRONG
4$: SCOPE  ;LOOP TO 3$ IF SW09=1

```



```

2221 025050 012737 025056 001220          5$: MOV    #5$,LOCK      ;NEW SCOPE1
2222 025056                                JSR    PC,SETBR4    ;SET THE BR4 BIT
2223 025056 004737 026710                ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2224 025062 104414                        100406 ;START AT ROM PC=6
2225 025064 100406                        ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2226 025066 104414                        104125! <400*6> ;JUMP TO ROM PC OF 525
2227 025070 107125                        JSR    PC,ROMDAT    ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2228 025072 004737 026750                1252 ;INDEX
2229 025076 001252                        CMP    R5,R4        ;ARE NEW ROM PC CONTENTS CORRECT?
2230 025100 020504                        BEQ    6$           ;BR IF YES
2231 025102 001401                        HLT    6            ;ERROR, CROM PC IS WRONG
2232 025104 104006                        SCOPE1 ;LOOP TO 5$ IF SW59=1
2233 025106 104401                        SCOPE ;SCOPE THIS TEST
2234 025110 104400
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245 025112 012737 000014 001226          TST14: MOV    #14,TSTNO
2246 025120 012737 025306 001216          MOV    #TST15,NEXT
2247 025126 012737 025152 001220          MOV    #1$,LOCK
2248
2249 025134 104412                        MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
2250 025136 032737 100000 001366          BIT    #BIT15,STAT1 ;MASTER CLEAR DMC11
2251 025144 001057                        BNE    6$+2         ;IS IT CROM?
2252 025146 004737 027012                        JSR    PC,MAPCK     ;SKIP TEST IF YES
2253 025152                                ;CHECK FOR HI OR LO
2254 025152 004737 026716          1$: JSR    PC,SETBR7    ;SET THE BR7 BIT
2255 025156 104414                        ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2256 025160 100400                        100400 ;START AT ROM PC=0
2257 025162 104414                        ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2258 025164 117777                        114377! <400*7> ;JUMP TO ROM PC OF 1777
2259 025166 004737 026750                JSR    PC,ROMDAT    ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2260 025172 003776                        3776 ;INDEX
2261 025174 020504                        CMP    R5,R4        ;ARE NEW PC CONTENTS CORRECT?
2262 025176 001401                        BEQ    2$           ;BR IF YES
2263 025200 104006                        HLT    6            ;ERROR, CROM PC IS WRONG
2264 025202 104401                        SCOPE1 ;LOOP TO 1$ IF SW09=1
2265 025204 012737 025212 001220          2$: MOV    #3$,LOCK  ;NEW SCOPE1
2266 025212                                3$:
2267 025212 004737 026716          JSR    PC,SETBR7    ;SET THE BR7 BIT
2268 025216 104414                        ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2269 025220 100403                        100403 ;START AT ROM PC=3
2270 025222 104414                        ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2271 025224 103400                        100000! <400*7> ;JUMP TO ROM PC OF 0
2272 025226 004737 026750                JSR    PC,ROMDAT    ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2273 025232 000000                        0 ;INDEX
2274 025234 020504                        CMP    R5,R4        ;ARE NEW PC CONTENTS CORRECT?
2275 025236 001401                        BEQ    4$           ;BR IF YES
2276 025240 104006                        HLT    6            ;ERROR, CROM PC IS WRONG

```

```

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

```

TEST 14

```

-----
TST14: MOV    #14,TSTNO
MOV    #TST15,NEXT
MOV    #1$,LOCK
MSTCLR ;R1 CONTAINS BASE DMC11 ADDRESS
BIT    #BIT15,STAT1 ;MASTER CLEAR DMC11
BNE    6$+2         ;IS IT CROM?
JSR    PC,MAPCK     ;SKIP TEST IF YES
;CHECK FOR HI OR LO
1$: JSR    PC,SETBR7    ;SET THE BR7 BIT
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100400 ;START AT ROM PC=0
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
114377! <400*7> ;JUMP TO ROM PC OF 1777
JSR    PC,ROMDAT    ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
3776 ;INDEX
CMP    R5,R4        ;ARE NEW PC CONTENTS CORRECT?
BEQ    2$           ;BR IF YES
HLT    6            ;ERROR, CROM PC IS WRONG
2$: MOV    #3$,LOCK  ;NEW SCOPE1
3$:
JSR    PC,SETBR7    ;SET THE BR7 BIT
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100403 ;START AT ROM PC=3
ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
100000! <400*7> ;JUMP TO ROM PC OF 0
JSR    PC,ROMDAT    ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
0 ;INDEX
CMP    R5,R4        ;ARE NEW PC CONTENTS CORRECT?
BEQ    4$           ;BR IF YES
HLT    6            ;ERROR, CROM PC IS WRONG

```

```

2277 025242 104401 025252 001220 4S: SCOP1 ; LOOP TO 3S IF SW09=1
2278 025249 012737 025252 001220 4S: MOV #5S,LOCK ; NEW SCOPI
2279 025252 004737 026716 5S: JSR PC,SETBR7 ; SET THE BR7 BIT
2280 025256 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2281 025260 100406 100406 ; START AT ROM PC=6
2282 025262 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2283 025264 107525 104125! <400*7> ; JUMP TO ROM PC OF 525
2284 025266 004737 026750 JSR PC,ROMDAT ; R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2285 025272 001252 1252 ; INDEX
2286 025274 020504 CMP R5,R4 ; ARE NEW ROM PC CONTENTS CORRECT?
2287 025276 001401 BEQ 6S ; BR IF YES
2288 025300 :04006 HLT 6 ; ERROR, CROM PC IS WRONG
2289 025302 104401 6S: SCOP1 ; LOOP TO 5S IF SW59=1
2290 025304 :04400 SCOPE ; SCOPE THIS TEST
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303 025306 012737 000015 001226 TEST15: MOV #15,TS*NO
2304 025314 012737 025502 001216 MOV #TST16,NEXT
2305 025322 012737 025346 001220 MOV #15,LOCK
2306 ; R1 CONTAINS BASE DMC11 ADDRESS
2307 025330 104412 MSTCLR ; MASTER CLEAR DMC11
2308 025332 032737 100000 001366 BIT #BIT15,STAT1 ; IS IT CRAM?
2309 025340 001057 BNE 6S+2 ; SKIP TEST IF YES
2310 025342 004737 027012 JSR PC,MAPCK ; CHECK FOR HI OR LO
2311 025346 004737 026656 1S: JSR PC,CLRALL ; CLEAR ALL CONDITIONS
2312 025348 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2313 025352 100400 100400 ; START AT ROM PC=0
2314 025354 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2315 025360 115377 114377! <400*2> ; JUMP TO ROM PC OF 177?
2316 025362 004737 026750 JSR PC,ROMDAT ; R5=EXPECTED ROM DATA R4=ACTUAL ROM DATA
2317 025366 000002 2 ; INDEX
2318 025370 020504 CMP R5,R4 ; ARE NEW PC CONTENTS CORRECT?
2319 025372 001401 BEQ 2S ; BR IF YES
2320 025374 104006 HLT 6 ; ERROR, CROM PC IS WRONG
2321 025376 104401 2S: SCOP1 ; LOOP TO 1S IF SW09=:
2322 025400 012737 025406 001220 3S: MOV #3S,LOCK ; NEW SCOPI
2323 025406 004737 026656 JSR PC,CLRALL ; CLEAR ALL CONDITIONS
2324 025412 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2325 025414 100403 100403 ; START AT ROM PC=3
2326 025416 104414 ROMCLK ; NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2327 025420 101000 100000! <400*2> ; JUMP TO ROM PC OF 0
2328 025422 004737 026750 JSR PC,ROMDAT ; R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2329 025426 000010 10 ; INDEX
2330 025430 020504 CMP R5,R4 ; ARE NEW PC CONTENTS CORRECT?

```

```

2333 025432 001401 BEQ 45 ;BR IF YES
2334 025434 104006 HLT 6 ;ERROR CROM PC IS WRONG
2335 025436 104401 45: SCOPI ;LOOP TO 35 IF SW09=1
2336 025440 012737 025446 001220 MOV #55,LOCK ;NEW SCOPI
2337 025446 004737 026656 55: JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2338 025452 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2339 025454 100406 ;START AT ROM PC=6
2340 025454 100406 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2341 025456 104414 ;JUMP TO ROM PC OF 525
2342 025460 105125 104125! <400*2> JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2343 025462 004737 026750 JSR PC,ROMDAT ;INDEX
2344 025466 000016 ;ARE NEW ROM PC CONTENTS CORRECT?
2345 025470 020504 CMP R5,R4 ;BR IF YES
2346 025472 001401 BEQ 65 ;ERROR CROM PC IS WRONG
2347 025474 104006 HLT 6 ;LOOP TO 55 IF SW59=1
2348 025476 104401 65: SCOPI ;SCOPE THIS TEST
2349 025500 104400 SCOPE

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

: TEST 16
:-----
2361 025502 012737 000016 001226 TST16: MOV #16,TSTNO
2362 025510 012737 025676 001216 MCV #TST17,NEXT
2363 025516 012737 025542 001220 MOV #15,LOCK
2364 ;R1 CONTAINS BASE DMC11 ADDRESS
2365 025524 104412 MSTCLR ;MASTER CLEAR DMC11
2366 025526 032737 100000 001366 BIT #BIT15,STAT1 ;IS IT CRAM?
2367 025534 001057 BNE 65+2 ;SKIP TEST IF YES
2368 025536 004737 027012 JSR PC,MAPCK ;CHECK FOR HI OR LO
2369 025542 004737 026656 15: JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2370 025546 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2371 025550 100400 100400 ;START AT ROM PC=0
2372 025552 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2373 025554 115777 114377! <400*3> JSR PC,ROMDAT ;JUMP TO ROM PC OF 1777
2374 025556 004737 026750 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2375 025562 000002 ;INDEX
2376 025564 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2377 025566 001401 BEQ 25 ;BR IF YES
2378 025570 104006 HLT 6 ;ERROR CROM PC IS WRONG
2379 025574 104401 25: SCOPI ;LOOP TO 15 IF SW09=1
2380 025576 012737 025602 001220 MOV #35,LOCK ;NEW SCOPI
2381 025602 004737 026656 35: JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2382 025606 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2383 025610 100403 ;START AT ROM PC=3
2384 025612 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2385 025614 101400 100000! <400*3> ;JUMP TO ROM PC OF 0
2386 025616 004737 026750 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA

```

2389	025622	000010				10		:INDEX
2390	025624	020504				CMP	R5,R4	:ARE NEW PC CONTENTS CORRECT?
2391	025626	001401				BEQ	4\$:BR IF YES
2392	025630	104006				HLT	6	:ERROR, CROM PC IS WRONG
2393	025632	104401			4\$:	SCOPI		:LOOP TO 3\$ IF SW09=1
2394	025634	012737	025642	001220		MOV	#5\$,LOCK	:NEW SCOPI
2395	025642				5\$:			
2396	025642	004737	026656			JSR	PC,CLRALL	:CLEAR ALL CONDITIONS
2397	025646	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2398	025650	100406				100406		:START AT ROM PC=6
2399	025652	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2400	025654	105525				104125! <400*3>		:JUMP TO ROM PC OF 525
2401	025656	004737	026750			JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2402	025662	000016				16		:INDEX
2403	025664	020504				CMP	R5,R4	:ARE NEW ROM PC CONTENTS CORRECT?
2404	025666	001401				BEQ	6\$:BR IF YES
2405	025670	104006				HLT	6	:ERROR, CROM PC IS WRONG
2406	025672	104401			5\$:	SCOPI		:LOOP TO 5\$ IF SW59=1
2407	025674	104400				SCOPE		:SCOPE THIS TEST
2408								
2409								
2410								
2411								:***** TEST 17 *****
2412								:*CROM TEST OF JUMP(I) ON BRO SET MICRO-PROCESSOR INSTRUCTION.
2413								:*CLEAR THE BRO BIT, PERFORM THE JUMP INSTRUCTION.
2414								:*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
2415								:*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).
2416								:*****
2417								
2418								: TEST 17
2419	025676	012737	000017	001226	TST17:	MOV	#17,TSTNO	
2420	025704	012737	026072	001216		MOV	#120,NEXT	
2421	025712	012737	025736	001220		MOV	#1\$,LOCK	
2422								:R1 CONTAINS BASE DMC11 ADDRESS
2423	025720	104412				MS*CLR		:MASTER CLEAR DMC11
2424	025722	032737	100000	001366		BIT	#BIT15,STAT1	:IS IT CROM?
2425	025730	001057				BNE	6\$+2	:SKIP TEST IF YES
2426	025732	004737	027012			JSR	PC,MAPCK	:CHECK FOR HI OR LO
2427	025736				1\$:			
2428	025736	004737	026656			JSR	PC,CLRALL	:CLEAR ALL CONDITIONS
2429	025742	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2430	025744	100400				100400		:START AT ROM PC=0
2431	025746	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2432	025750	116377				114377! <400*4>		:JUMP TO ROM PC OF 1777
2433	025752	004737	026750			JSR	PC,ROMDAT	:R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2434	025756	000002				2		:INDEX
2435	025760	020504				CMP	R5,R4	:ARE NEW PC CONTENTS CORRECT?
2436	025762	001401				BEQ	2\$:BR IF YES
2437	025764	104006				HLT	6	:ERROR, CROM PC IS WRONG
2438	025766	104401			2\$:	SCOPI		:LOOP TO 1\$ IF SW09=1
2439	025770	012737	025776	001220		MOV	#3\$,LOCK	:NEW SCOPI
2440	025776				3\$:			
2441	025776	004737	026656			JSR	PC,CLRALL	:CLEAR ALL CONDITIONS
2442	026032	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2443	026004	100403				100403		:START AT ROM PC=3
2444	026006	104414				ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```

2445 026010 102000 100000! <400*4> JUMP TO ROM PC OF 0
2446 026012 004737 026750 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2447 026016 000010 10 INDEX
2448 026020 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2449 026022 001401 BEQ 4$ ;BR IF YES
2450 026024 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2451 026026 104401 4$: SCOPI ;LOOP TO 3$ IF SW09=1
2452 026030 012737 026036 001220 MOV #5$,LOCK ;NEW SCOPI
2453 026036 5$:
2454 026036 004737 026656 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2455 026042 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2456 026044 100406 100406 ;START AT ROM PC=6
2457 026046 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2458 026050 106125 104125! <400*4> JUMP TO ROM PC OF 525
2459 026052 004737 026750 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2460 026056 000016 16 INDEX
2461 026060 020504 CMP R5,R4 ;ARE NEW ROM PC CONTENTS CORRECT?
2462 026062 001401 BEQ 6$ ;BR IF YES
2463 026064 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2464 026066 104401 6$: SCOPI ;LOOP TO 5$ IF SW59=1
2465 026070 104400 SCOPE ;SCOPE THIS TEST
2466
2467
2468 ;*****TEST 20*****
2469 ;*CROM TEST OF JUMP(I) ON BRI SET MICRO-PROCESSOR INSTRUCTION.
2470 ;*CLEAR THE BRI BIT, PERFORM THE JUMP INSTRUCTION,
2471 ;*VERIFY THAT THE JUMP DID NOT OCCUR BY READING
2472 ;*THE CONTENTS OF THE NEW ROM PC(IT SHOULD INCREMENT BY ONE).
2473 ;:*****
2474
2475 : TEST 20
2476 :-----
2477 026072 012737 000020 001226 TST20: MOV #20,TSTNO
2478 026100 012737 026266 001216 MOV #TST21,NEXT
2479 026106 012737 026132 001220 MOV #1$,LOCK
2480 ;R1 CONTAINS BASE DMC11 ADDRESS
2481 026114 104412 MSTCLR ;MASTER CLEAR DMC11
2482 026116 032737 100000 001366 BIT #BIT15,STAT1 ;IS IT CROM?
2483 026124 001057 BNE 6$+2 ;SKIP TEST IF YES
2484 026126 004737 027012 JSR PC,MAPCK ;CHECK FOR HI OR LO
2485 026132 1$:
2486 026132 004737 026656 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2487 026136 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2488 026140 100400 100400 ;START AT ROM PC=0
2489 026142 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2490 026144 116777 114377! <400*5> JUMP TO ROM PC OF 1777
2491 026146 004737 026750 JSR PC,ROMDAT ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2492 026152 000002 2 INDEX
2493 026154 020504 CMP R5,R4 ;ARE NEW PC CONTENTS CORRECT?
2494 026156 001401 BEQ 2$ ;BR IF YES
2495 026160 104006 HLT 6 ;ERROR, CROM PC IS WRONG
2496 026162 104401 2$: SCOPI ;LOOP TO 1$ IF SW09=1
2497 026164 012737 026172 001220 MOV #3$,LOCK ;NEW SCOPI
2498 026172 3$:
2499 026172 004737 026656 JSR PC,CLRALL ;CLEAR ALL CONDITIONS
2500 026176 104414 ROMCLK ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```



```

2613 026554 012737 026562 001220      3$: MOV    #3$,LOCK      ;NEW SCOPI
2614 026562                                ;CLEAR ALL CONDITIONS
2615 026562 004737 026656      JSR    PC,CLRALL      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2616 026566 104414      ROMCLK  ;START AT ROM PC=3
2617 026570 100403      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2618 026572 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2619 026574 103400      10000! <400*7> ; JUMP TO ROM PC OF 0
2620 026576 004737 026750      JSR    PC,ROMDAT      ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2621 026602 000010      10      ;INDEX
2622 026604 020504      CMP    R5,R4          ;ARE NEW PC CONTENTS CORRECT?
2623 026606 001401      BEQ    4$             ;BR IF YES
2624 026610 104006      HLT    6              ;ERROR, CROM PC IS WRONG
2625 026612 104401      4$: SCOPI            ;LOOP TO 3$ IF SW05=1
2626 026614 012737 026622 001220      MOV    #5$,LOCK      ;NEW SCOPI
2627 026622                                ;CLEAR ALL CONDITIONS
2628 026622 004737 026656      JSR    PC,CLRALL      ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2629 026626 104414      ROMCLK  ;START AT ROM PC=6
2630 026630 100406      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2631 026632 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2632 026634 107525      104125! <400*7> ; JUMP TO ROM PC OF 525
2633 026636 004737 026750      JSR    PC,ROMDAT      ;R5=EXPECTED ROM DATA,R4=ACTUAL ROM DATA
2634 026642 000016      16      ;INDEX
2635 026644 020504      CMP    R5,R4          ;ARE NEW ROM PC CONTENTS CORRECT?
2636 026646 001401      BEQ    6$             ;BR IF YES
2637 026650 104006      HLT    6              ;ERROR, CROM PC IS WRONG
2638 026652 104401      6$: SCOPI            ;LOOP TO 5$ IF SW59=1
2639 026654 104400      SCOPE            ;SCOPE THIS TEST
2640                                00300
2641                                00400
2642                                00500      ;SUBROUTINES
2643                                00600      ;-----
2644                                00700
2645 026656                                00800 CLRALL:
2646                                00900      ;THIS SUBROUTINE CLEARS THE C&Z BITS AND THE BP
2647                                01000
2648 026656 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2649 026660 000400      01200      000400      ;BR+0
2650 026662 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2651 026664 063220      01400      063220      ;SP(0)+BR
2652 026666 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2653 026670 060400      01600      060400      ;BR+SP(0)+BR
2654 026672 000207      01700      RTS    PC
2655                                01800
2656                                01900
2657 026674                                02000 SETBRO:
2658                                02100      ;THIS SUBROUTINE SETS BRO BIT
2659                                02200
2660 026674 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2661 026676 000401      02400      000401      ;BR+001
2662 026700 000207      02500      RTS    PC
2663                                02600
2664                                02700
2665 026702                                02800 SETBRI:
2666                                02900      ;THIS SUBROUTINE SETS BRI BIT
2667                                03000
2668 026702 104414      ROMCLK  ;NEXT WORD IS INSTRUCTION, ROMCLK PC=5304

```


2669	026704	000402		03200	000402		:BR+002
2670	026706	000207		03300	RTS	PC	
2671				03400			
2672				03500			
2673	026710			03600	SETBR4:		:THIS SUBROUTINE SETS BR4 BIT
2674				03700			
2675				03800			
2676	026710	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2677	026712	000420		04000	000420		:BR+020
2678	026714	000207		04100	RTS	PC	
2679				04200			
2680				04300			
2681	026716			04400	SETBR7:		:THIS SUBROUTINE SETS BR7 BIT
2682				04500			
2683				04600			
2684	026716	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2685	026720	000600		04800	000600		:BR+200
2686	026722	000207		04900	RTS	PC	
2687				05000			
2688				05100			
2689	026724			05200	SETC:		:THIS SUBROUTINE SETS THE C BIT
2690				05300			
2691				05400			
2692	026724	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2693	026726	000777		05600	000777		:BR+377
2694	026730	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2695	026732	063220		05800	063220		:SP(0)+BR
2696	026734	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2697	026736	060400		06000	060400		:BR+SP(0)+BR
2698	026740	000207		06100	RTS	PC	
2699				06200			
2700				06300			
2701	026742			06400	SETZ:		:THIS SUBROUTINE SETS THE Z BIT
2702				06500			
2703				06600			
2704	026742	104414			ROMCLK		:NEXT WORD IS INSTRUCTION, ROMCLK PC=5304
2705	026744	000777		06800	000777		:BR+377
2706	026746	000207		06900	RTS	PC	
2707				07000			
2708				07100			
2709	026750			07200	ROMDAT:		:THIS SUBROUTINE LOADS R5 WITH EXPECTED ROM CONTENTS
2710				07300			:AND LOADS R4 WITH ACTUAL ROM CONTENTS
2711				07400			
2712				07500			
2713	026750	017600	000000	07600	MOV	#(SP),R0	:INDEX FOR COMPARE
2714	026754	06271E	000002	07700	ADD	#2,(SP)	:ADJUST STACK
2715	026760	0127.1	002000	07800	MOV	#BIT10,(R1)	:SET ROMO
2716	026764	016005	012322	07900	MOV	LOMAP(R0),R5	:PUT EXPECTED IN R5 (LOSPEED)
2717	026770	032737	000002	08000	BIT	#BIT1,STAT3	:LOW OR HIGH SPEED
2718	026776	001402		08100	BEQ	IS	:BR IF LOW SPEED
2719	027000	016005	016322	08200	MOV	HIMAP(R0),R5	:PUT EXPECTED IN R5 (HISPEED)
2720	027004	016104	000006	08300	IS:	MOV	#(R1),R4
2721	027010	000207		08400	RTS	PC	:RETURN
2722				08500			
2723	027012			08600	MAPCR:		:THIS SUBROUTINE CHECKS THE STATUS TABLE AND LOADS
2724				08700			

027532	001270		05000	SAVR4	
027534	000003		05100	DT7:	3
027536	003	010	05200	.BYTE	3.10
027540	001272		05300	SAVR5	
027542	003	004	05400	.BYTE	3.4
027544	001270		05500	SAVR4	
027546	004	002	05600	.BYTE	4.2
027550	001264		05700	SAVR2	
027552	000003		05800	DT10:	3
027554	003	007	05900	.BYTE	3.7
027556	001272		06000	SAVR5	
027560	003	004	06100	.BYTE	3.4
027562	001270		06200	SAVR4	
027564	006	002	06300	.BYTE	6.2
027566	001252		06400	TEMP3	
027570	000002		06500	DT11:	2
027572	006	004	06600	.BYTE	6.4
027574	001252		06700	TEMP3	
027576	006	002	06800	.BYTE	6.2
027600	001254		06900	TEMP4	
			07000		
027602			07100	.ERRTAB:	
027602	000000		07200	0	
027604	000000		07300	0	
027606	000000		07400	0	
027610	027040		07500	EM1	
027612	027342		07600	DH1	:HLT 1
027614	027436		07700	DT1	
027616	027061		07800	EM2	
027620	027342		07900	DH1	:HLT 2
027622	027436		08000	DT1	
027624	027040		08100	EM1	
027626	027342		08200	DH1	:HLT 3
027630	027454		08300	DT2	
027632	027115		08400	EM3	
027634	027342		08500	DH1	:HLT 4
027636	027472		08600	DT3	
027640	027136		08700	EM4	
027642	027374		08800	DH2	:HLT 5
027644	027510		08900	DT4	
027646	027136		09000	EM4	
027650	027374		09100	DH2	:HLT 6
027652	027522		09200	DT5	
027654	027152		09300	EM5	
027656	027374		09400	DH2	:HLT 7
027660	027510		09500	DT4	
027662	000000		09600	0	
027664	000000		09700	0	
027666	000000		09800	0	
027670	027204		09900	EM7	
027672	027342		10000	DH1	:HLT 11
027674	027534		10100	DT7	
027676	027222		10200	EM10	
027700	027374		10300	DH2	:HLT 12
027702	027510		10400	DT4	
027704	027247		10500	EM1:	

DDMG MACY11 30(1046) 11-JUL-77 12:11 PAGE 57
DDMG.P11 22-APR-77 09:29 SUBROUTINES

027706	027342	10600	DH1	:HLT	13
027710	027552	10700	DT10		
027712	027273	10800	EM12		
027714	000000	10900	0	:HLT	14
027716	000000	11000	0		
027720	027273	11100	EM12		
027722	027374	11200	DH2	:HLT	15
027724	027522	11300	DT5		
027726	027317	11400	EM13		
027730	027415	11500	DH3	:HLT	16
027732	027570	11600	DT11		
		11700			
		11800			
027734		11900	CORMAX:		
	000001	12400	.END		

CROSS REFERENCE TABLE -- USER SYMBOLS

MODU	006704	1187#	1453																	
MPASSX	006106	741	1187#																	
MPFAIL	005675	1121	1187#																	
MQM	005666	861	1187#	1352	1427	1437	1447	1462	1476											
MR	005755	723	1187#	1346																
MRESET=	004000	96#																		
MS*CLR=	104412	235#	1126	1738	1739	1803	1853	1910	1964	2021	2078	2135	2192	2249						
		2307	2365	2423	2481	2539	2597													
MTITLE	001000	136#	511																	
MTSTN	006130	1059	1187#	1330																
MT*ST*PC	006031	1187#																		
MVECX	006100	739	1187#																	
NEXT	001216	157#	799	1092	1686*	1709*	1736*	1800*	1850*	1907*	1961*	2018*	2075*	2132*						
		2189*	2246*	2304*	2362*	2420*	2478*	2536*	2594*											
NOACT	007154	523	1187#	1282																
NODEV	002674	577	616#																	
NUM	006450	1187#	1382																	
OK	002646	603	610#	639																
ONE	001302	187#																		
PACT00	001702	365#																		
PACT01	001706	368#																		
PACT02	001712	371#																		
PACT03	001716	374#																		
PACT04	001722	377#																		
PACT05	001726	380#																		
PACT06	001732	383#																		
PACT07	001736	386#																		
PACT10	001742	389#																		
PACT11	001746	392#																		
PACT12	001752	395#																		
PACT13	001756	398#																		
PACT14	001762	401#																		
PACT15	001766	404#																		
PACT16	001772	407#																		
PACT17	001776	410#																		
PARAM =	104405	225#	1331	1393	1396	1405	1484	1493												
PARAM1	004234	881#	898																	
PARBIT=	040000	96#																		
PARERR	004310	884	886	888	897#	904	906	908												
PASCNT	001230	162#	734*	735	746	771	1304*													
PERFOR=	004537	96#																		
PFTAB	005430	1122	1128#																	
POPPO =	012600	72#	1086																	
POP!SP=	005726	70#																		
FOP2SP=	022626	74#	801																	
PRIO	006547	1187#	1413																	
PS =	177776	63#	476*	711*	1617*	1627*														
PUSHRD=	010046	71#	1083																	
PUSH1S=	005746	69#																		
PUSH2S=	024646	73#																		
QV.FLG	001327	204#	482*	750*	790															
RESREG	005221	1075	1078#																	
RESTAR	00535J	1108	1114#																	
RESTAT	003534	749	753	761#																
RESOS =	104407	229#	1078																	
RE*JRN	001214	156#	492*	720*	724	761*	799*	803	1092*	1095	1127	1345*	1355*	1357						

