RM03/2

DISKLESS 1
CZRMJC0

AH-B019C-MC

FICHE 1 OF 2

DEC 1978
digital
MADE IN USA

RM03/2

DISKLESS 1
CZRMJC0

AH-B019C-MC

FICHE 2 OF 2

DEC 1978
digital
MADE IN USA

```
                                                    B 1
                        .REM    \
```

IDENTIFICATION

PRODUCT CODE:	AC-B018C-MC
PRODUCT NAME:	CZRMJC0 RM03/RM02 DISKLESS DIAGNOSTIC
DATE CREATED:	FEB 78
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	DOUG RIIKONEN


```
.PAGE
```

CONTENTS

6. XXDP COMPATIBILITY

7. OPERATING SYSTEM COMPATIBILITY

6. TEST DESCRIPTION

1.0 INTRODUCTION

1.1 ABSTRACT

THE RM03 DISKLESS DIAGNOSTIC IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL AND DIAGNOSTIC MEANS TO VERIFY THE OPERABILITY OF THE RM03 DISK SUBSYSTEM EXCLUDING AND INDEPENDENTLY OF THE STORAGE MODULE DRIVE. IN PARTICULAR, THE PROGRAM SERVES THE FOLLOWING PURPOSES:

TO DETECT ERRORS AND FAULTS IN THE RH70 MASSBUS CONTROLLER;

TO DETECT ERRORS AND FAULTS IN THE RM03 MASSBUS ADAPTER;

TO RESOLVE HARDWARE FAILURES IN THE RH70 AND RM03 TO A FIELD REPLACEABLE MODULE OR MODULES.

1.2 UNIT UNDER TEST

THE UNIT UNDER TEST (UUT) IS THE RM03 DISK SUBSYSTEM, EXCLUDING THE STORAGE MODULE DISK DRIVE AND THE RH70 MASSBUS CONTROLLER.

2.0 OPERATING REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

THE FOLLOWING MINIMUM HARDWARE CONFIGURATION, ASSUMED TO BE OPERATIONAL, IS REQUIRED TO LOAD AND EXECUTE THE RM03 DISKLESS DIAGNOSTIC:

PDP-11 PROCESSOR
24K MEMORY
KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH CONTROLLER

UNIT UNDER TEST,
WHERE THE UNIT UNDER TEST CONSISTS OF ONE TO EIGHT RM03 ADAPTERS.

2.2 MEDIA REQUIREMENTS

NONE

2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

NONE

3.0 OPERATING PROCEDURE

3.1 LOADING

THE PROGRAM MAY BE LOADED BY EITHER PAPER TAPE, USING THE STANDARD PAPER TAPE LOADING PROCEDURE, OR XXDP MEDIA, USING THE APPROPRIATE LOADING DEVICE.

3.2 SWITCH OPTIONS

THE FOLLOWING SWITCH OPTIONS ARE INVOKED WHEN THE APPROPRIATE SWITCH IS ON.

SW15 HALT ON ERROR
SW14 LOOP ON TEST (CURRENTLY BEING EXECUTED)
SW13 INHIBIT ERROR TYPEOUTS
SW12 UNUSED
SW11 INHIBIT TEST ITERATIONS
SW10 BELL ON ERROR
SW09 LOOP ON ERROR
SW08 LOOP ON TEST IN SW07-00

THE LOW ORDER 8 SWITCHES ARE USED IN CONJUNCTION WITH SW08 TO SPECIFY A PARTICULAR TEST WHICH THE PROGRAM WILL LOOP ON.

3.3 STARTING

THE PROGRAM STARTS AT LOCATION 200 WHICH USES DEFAULT VALUES OF PARAMETERS AND PROVIDES MAXIMUM TESTING WITH THE SWITCH REGISTER EQUAL TO ZERO.

4.0 OPERATOR INTERFACE

4.1 PROGRAM ID

THE PROGRAM TYPES ITS NAME AND MAINDEC NUMBER THE FIRST TIME IT IS STARTED AFTER BEING LOADED.

4.2 PROGRESS REPORTS

AN END OF PASS REPORT OCCURRS EACH TIME THE PROGRAM IS EXECUTED FOR ALL ADAPTERS IN THE TEST QUE. THE END OF PASS REPORT INCLUDES A MESSAGE AND AN ERROR SUMMARY.

4.3 PERFORMANCE REPORT

NO PERFORMANCE REPORTS ARE GIVEN DURING THE EXECUTION OF THE PROGRAM.

4.4 PROGRAM HALTS

THERE ARE NO SCHEDULED HALTS DURING THE EXECUTION OF THE PROGRAM. PROCESSOR HALTS ARE DUE TO THE TRAP CATCHER.

4.5 ERROR REPORTS

THE RM03 DISKLESS DIAGNOSTIC PROVIDES COMPREHENSIVE ERROR REPORTS INTENDED TO (1) AID IN FAULT RESOLUTION AND (2) MINIMIZE REFERENCES TO PROGRAM LISTINGS.

THE FIRST LINE OF THE ERROR REPORT CONTAINS THE NUMBER OF THE UNIT BEING TESTED, THE TEST NUMBER, THE ERROR NUMBER AND THE VALUE OF THE PROGRAM COUNTER WHERE THE ERROR WAS CALLED. THIS LINE IS FOLLOWED BY THE ERROR MESSAGE: SEVERAL LINES OF TEXT WHICH GIVE A COMPREHENSIVE DESCRIPTION OF THE ERROR, AND A LIST OF FAILING MODULES IN ORDER OF DECREASING PROBABILITY. THE ERROR MESSAGE IS NORMALLY FOLLOWED BY ONE OR MORE PAIRS OF LINES CONTAINING DATA HEADERS AND DATA PERTININENT TO THE ERROR, INCLUDING EXPECTED AND ACTUAL TEST

```
                                                                    H  1
   266                          RESULTS.
```

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

THE RM03 DISKLESS DIAGNOSTIC IS EXECUTABLE ON ANY PDP-11 PROCESSOR, PROVIDING PREVIOUSLY MENTIONED HARDWARE REQUIREMENTS ARE MET.

5.2 DUAL PORT CONFIGURATIONS

THE RM03 DISKLESS DIAGNOSTIC IS NOT EXECUTABLE ON RM03 SUBSYSTEMS HAVING THE DUAL PORT OPTION UNLESS THE DUAL PORT SWITCH IS SET TO THE APPROPRIATE PORT (A OR B) AND NOT TO THE PROGRAMMABLE POSITION (A/B).

5.3 MEMORY PARITY HARDWARE

MEMORY PARITY HARDWARE WILL NOT BE USED DURING THE EXECUTION OF THE RM03 DISKLESS DIAGNGOSTIC.

5.4 MEMORY MANAGEMENT HARDWARE

MEMORY MANAGEMENT HARDWARE WILL NOT BE USED DURING THE RM03 DISKLESS DIAGNOSTIC.

5.5 ACT11, APT11 COMPATIBILITY

THE RM03 DISKLESS DIAGNOSTIC PROGRAM IS COMPATIBLE WITH ACT11 AND APT11 IN BOTH DUMP AND AUTOMATIC MODES. FURTHER, THE PROGRAM WILL EXECUTE A QUICK PASS DURING THE FIRST PASS IN SUPPORT OF QUICK VERIFY MODE.

5.6 XXDP COMPATIBILITY

THE RM03 DISKLESS DIAGNOSTIC PROGRAM IS COMPATIBLE WITH XXDP IN DUMP AND CHAIN MODES.

5.7 OPERATING SYSTEM COMPATIBILITY

THE PROGRAM IS NOT REQUIRED TO BE COMPATIBLE WITH ANY OPERATING SYSTEM.

6.0 TEST DESCRIPTION

THE PROGRAM IS DESIGNED IN A BOTTOM UP MANNER SUCH THAT EACH TEST GENERALLY USES A MORE COMPLEX SUBSET OF HARDWARE THAN THE PREVIOUS TEST.

MODULE CALLOUT IS PREDICATED ON THE ASSUMPTION THAT EARLIER TESTS HAVE BEEN COMPLETED WITHOUT ERROR AND THAT ERRORS ARE DUE TO SINGLE, NONTRANSIENT HARDWARE FAILURES.

THE ''RM03 DISKLES DIAGNOSTIC'' CAN BE EXECUTED USING AN RH70 OR AN RH11 MASSBUS CONTROLLER.

UNLESS SPECIFIED BY THE OPERATOR OR BY THE ENVIRONMENT TABLE THE TEST IS REPEATED FOR EACH POSSIBLE DEVICE STARTING WITH DEVICE 0.

THE MODULES WHICH MAY BE CALLED OUT DURING THE EXECUTION OF THE TEST ARE AS FOLLOWS:

IF
CS
DS
MASSBUS MODULE

THE RADIAL MODULE (RD) IS NOT TESTED BY THIS PROGRAM.

TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE RM03 CAN COMPLETE A REGISTER TRANSFER ON THE MASSBUS, AND, IN PARTICULAR, TO VERIFY THAT ''TRANSFER'' IS NOT STUCK IN AN INACTIVE STATE.

PROCEDURE:

THE PROGRAM WRITES AND READS REMOTE REGISTERS FOR THE SELECTED DEVICE. REGISTER CONTENTS AND PARITY ERRORS ARE IGNORED, AND THE TEST FAILS IF A ''NONEXISTENT DEVICE ERROR'' OR BUS TIMEOUT OCCURRS FOR EVERY REGISTER ACCESS. IF THE TEST FAILS THE PROGRAM JUMPS TO THE END OF PASS HANDLER WHICH SELECTS THE NEXT DEVICE TO BE TESTED.

PROBABLE FAULT:

THE TEST FAILS IF THE SELECTED DEVICE IS NONEXISTENT OR IS SWITCHED TO THE PROGRAMMABLE POSITION OR TO THE ALTERNATE PORT. THE FOLLOWING FAULTS ARE APPLICABLE ONLY WHEN THE DEVICE IS PRESENT AND IS SWITCHED TO THE APPROPRIATE PORT.

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE
3. CS MODULE

CTOD TEST

PURPOSE:

TO VERIFY THAT DATA CAN BE TRANSFERRED TO AND FROM THE RM03 USING THE CONTROL BUS AND, IN PARTICULAR, TO VERIFY THAT ''CONTROLLER TO DEVICE'' HAS NOT FAILED.

PROCEDURE:

THE TEST WRITES ONES IN REMOTE REGISTERS THEN READS EACH REGISTER WHICH WILL WRITE ZEROS IN THE REGISTER IF ''IF3 CTOD HOLD H'' IS STUCK AT ONE. THE TEST THEN READS AS MANY REMOTE REGISTERS AS ARE NECESSARY TO OBTAIN ONE OR MORE ONE BITS.

PROBABLE FAULT:

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE

MASSBUS INITIALIZE TEST

PURPOSE:

TO VERIFY THAT THE MASSBUS ADAPTER IS BEING INITIALIZED BY THE MASS BUS.

PROCEDURE:

USING CONTROLLER CLEAR TO INITIALIZE THE SELECTED UNIT, THIS TEST THEN READS MASSBUS ADAPTER REGISTERS TO VERIFY THAT AT LEAST ONE BIT IS CLEARED. MASSBUS ADAPTER REGISTERS ARE PRESET TO A NON ZERO VALUE PRIOR TO CONTROLLER CLEAR.

PROBABLE FAULT:

1. ASYNCHRONOUS MASSBUS MODULE
2. IF MODULE
3. CS MODULE

CLEAR STUCK ACTIVE TEST

PURPOSE:

TO VERIFY THAT 'MBA CLR L'' ON THE CS MODULE IS NOT STUCK IN AN ACTIVE STATE.

PROCEDURE:

CONTROLLER CLEAR IS USED TO INITIALIZE THE SELECTED UNIT, AFTER WHICH 1'S ARE WRITTEN IN ERROR REGISTERS 1 AND 2 AND MAINTENANCE REGISTER 1. IF ANY 1 BITS CAN BE READ BACK THE TEST IS OK, ELSE, 'MBA CLR L'' IS PROBABLY STUCK ACTIVE.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE
3. ASYNCHRONOUS MASSBUS MODULE

TRISTATE TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE PATH TO AND FROM THE MASSBUS ADAPTER TRI-STATE REGISTER BUS IS NOT STUCK AT ONE OR ZERO AND THAT EACH BIT POSITION IS INDEPENDENT.

PROCEDURE:

THIS TEST PRESETS MASSBUS ADAPTER REGISTERS TO A NONZERO VALUE, THEN, ASSUMING THE REGISTERS ARE PRESET, IT CLEARS THEM USING A MOVE INSTRUCTION. THE TEST THEN READS AS MANY REGISTERS AS IS NECESSARY TO OBTAIN ONE OR MORE ZEROS FROM EACH BIT POSITION.

M 1

489

THE TEST CLEARS MASSBUS ADAPTER REGISTERS, THEN, ASSUMING THE REGISTERS ARE CLEARED, IT LOADS THEM WITH ONES AND READS AS MANY REGISTERS AS IS NECESSARY TO OBTAIN ONE OR MORE ONE BITS IN EACH BIT POSITION.

FINALLY, THE TEST WRITES A SINGLE ONE BIT PATTERN IN BIT 0 OF SELECTED REMOTE REGISTERS AND VERIFIES THAT THE PATTERN CAN BE READ BACK. THE ONE BIT IS SHIFTED AND THE TEST REPEATED FOR ALL BIT POSITIONS.

PROBABLE FAULT:

1. ASYNCHRONOUS MASSBUS MODULE
2. IF MODULE
3. CS MODULE
4. DS MODULE

REGISTER SELECT TEST

PURPOSE:

TO VERIFY THAT THE REGISTER SELECT LINES ARE NOT IN A STUCK POSITION.

PROCEDURE:

EACH REGISTER SELECT LINE IS TESTED BY WRITING ZEROS IN THOSE DEVICE REGISTERS FOR WHICH THE LINE MUST BE ZERO, THEN WRITING ONES IN THOSE DEVICE REGISTERS FOR WHICH THE LINE MUST BE ONE. THE ZERO REGISTER IS READ BACK AND IF THE SELECT LINE IS STUCK AT ZERO, THE ZERO REGISTER WILL CONTAIN ONES. THE PROCESS IS REPEATED TO DETECT A STUCK AT ONE FAULT, EXCEPT IN THIS CASE, THE ONES REGISTER IS WRITTEN FIRST.

REGISTER SELECT LINES 1, 2, 4 AND 8 ARE TESTED IN THIS MANNER; SELECT LINE 16 IS EXPLICITLY TESTED IN THE ''ILR TEST''.

PROBABLE FAULT:

1. IF MODULE
2. ASYNCHRONOUS MASSBUS MODULE

DRIVE TYPE TEST

PURPOSE:

TO TEST THE 'DRIVE TYPE' REGISTER, RMDT.

PROCEDURE:

THE PROGRAM READS RMDT AND VERIFIES THAT THE RESULT CORRESPONDS TO A SINGLE PORT OR DUAL PORT RM03 DRIVE.

PROBABLE FAULT:

1. IF MODULE

DEVICE AVAILABLE TEST

PURPOSE:

TO VERIFY THAT DEVICE AVAILABLE STATUS IS SET.

PROCEDURE:

THE PROGRAM TESTS 'DVA', BIT 11 OF RMCS1.

PROBABLE FAULT:

1. IF MODULE

HOLDING REGISTER TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE HOLDING REGISTER IS NOT STUCK AT ONE, STUCK AT ZERO, AND THAT THERE IS NO BIT INTERFERENCE.

PROCEDURE:

THE PROGRAM TRANSFERS ONES, THEN ZEROS TO THE HOLDING REGISTER AND VERIFIES THAT NONE OF THE BITS ARE STUCK AT ONE.

THE PROGRAM TRANSFERS ZEROS, THEN ONES TO THE HOLDING REGISTER AND VERIFIES THAT NONE OF THE BITS ARE STUCK AT ZERO.

FINALLY, THE TEST TRANSFERS A SHIFTING ONE BIT PATTERN AND VERIFIES THAT EACH BIT IS INDEPENDENT.

PROBABLE FAULT:

1. IF MODULE

CONTROL STATUS #1 TRANSFER TEST

PURPOSE:

TO VERIFY THAT BITS 01 THROUGH 05 OF CONTROL STATUS REGISTER 1 ARE NOT STUCK AT ONE OR ZERO, AND THAT THERE IS NOT BIT INTERFERENCE.

PROCEDURE:

THIS TEST WRITES ONES IN CONTROL STATUS REGISTER 1, RMCS1, THEN WRITES ZEROS AND VERIFIES THAT THE BITS ARE NOT STUCK AT ONE. THE GO BIT IS NOT TESTED IN THIS TEST.

NEXT, THE TEST CLEARS THE CONTROL STATUS REGISTER, RMCS1, WRITES ONES IN BITS 01 THROUGH 05 AND VERIFIES THAT THE BITS ARE NOT STUCK AT ZERO. THE GO BIT IS NOT TESTED.

THE TEST TRANSFERS A SHIFTING ONE BIT DATA PATTERN TO AND FROM RMCS1 AND CHECKS FOR ADJACENT BIT INTERFERENCE.

PROBABLE FAULT:

1. IF MODULE

ERROR REGISTER #1 TRANSFER TEST

PURPOSE:

TO VERIFY THAT ERROR REGISTER 1 IS NOT STUCK AT ONE OR ZERO, AND THAT THERE IS NOT BIT INTERFERENCE.

PROCEDURE:

THIS TEST WRITES ONES IN ERROR REGISTER 1, RMER1, THEN WRITES ZEROS AND VERIFIES THAT THE REGISTER IS NOT STUCK AT ONE.

```
'UNSAFE'' IS NOT TESTED DURING THIS TEST.  IN ORDER TO  LIMIT  THE
PROBABLE  FAULTS TO ONE OR TWO MODULES, THE TEST IS EXECUTED IN 3
```

PARTS WITH EACH PART TESTING THOSE BITS WHOSE PRESET FUNCTIONS ARE DERIVED FROM THE SAME MODULE.

THE TEST WRITES ZEROS IN ERROR REGISTER 1, RMER1, THEN WRITES ONES AND VERIFIES THAT THE REGISTER IS NOT STUCK AT ZERO.

FINALLY, THE TEST WRITES A SHIFTING ONE BIT PATTERN IN RMER1 AND CHECKS FOR ADJACENT BIT INTERFERENCE.

PROBABLE FAULT:

1. IF MODULE

2. CS MODULE

3. DS MODULE

ERROR REGISTER #2 TRANSFER TEST

PURPOSE:

TO VERIFY THAT ERROR REGISTER 2, RMER2, IS NOT STUCK AT ONE, STUCK AT ZERO, AND THAT THERE IS NOT BIT INTERFERENCE.

PROCEDURE:

THE TEST WRITES ONES THEN WRITES ZEROS IN RMER2 AND VERIFIES THAT NONE OF THE BITS ARE STUCK AT ONE. ''SKI'' AND ''DVC'' ARE NOT TESTED. IN ORDER TO LIMIT THE NUMBER OF PROBABLE FAULTS TO ONE OR TWO MODULES, THE TEST IS EXECUTED IN 3 PARTS WITH EACH PART TESTING THOSE BITS WHOSE PRESET FUNCTIONS ARE DERIVED FROM THE SAME MODULE.

THEN THE TEST WRITES ZEROS IN ERROR REGISTER 2, AND WRITES ONES VERIFYING THAT THE REGISTER IS NOT STUCK AT ZERO.

FINALLY, THE TEST WRITES A SHIFTING ONE BIT PATTERN IN THE REGISTER AND VERIFIES THAT ALL BIT POSITIONS ARE INDEPENDENT.

PROBABLE FAULT:

1. IF MODULE

2. CS MODULE

3. DS MODULE

CLEAR OFFSET STUCK ACTIVE TEST

PURPOSE:

TO VERIFY THAT THE SIGNAL WHICH CLEARS OFFSET MODE IS NOT STUCK IN ACTIVE STATE.

PROCEDURE:

THE TEST WRITES A ONE IN THE OFFSET DIRECTION BIT WHICH IS CLEARED BY THE SIGNAL AND VERIFIES THAT A ONE CAN BE READ BACK.

PROBABLE FAULT:

1. IF MODULE
2. DS MODULE

OFFSET REGISTER TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE OFFSET REGISTER IS NOT STUCK AT ONE, STUCK AT ZERO, AND THAT THERE IS NO ADJACENT BIT INTERFERENCE.

PROCEDURE:

THE OFFSET REGISTER, RMOF, IS WRITTEN WITH ONES, THEN WRITTEN WITH ZEROS AND READ TO VERIFY THAT NONE OF THE BITS ARE STUCK AT ONE.

THEN THE OFFSET REGISTER IS WRITTEN WITH ZEROS AND WRITTEN WITH ONES TO VERIFY THAT THE REGISTER IS NOT STUCK AT ZERO.

FINALLY, THE OFFSET REGISTER IS TESTED WITH A SHIFTING ONE BIT PATTERN.

PROBABLE FAULT:

1. IF MODULE

SERIAL NUMBER TEST

PURPOSE:

TO VERIFY THAT THE SERIAL NUMBER CAN BE READ.

PROCEDURE:

THE TEST READS THE SERIAL NUMBER REGISTER SEVERAL TIMES AND VERIFIES THAT THE NUMBER IS THE SAME EACH TIME.

PROBABLE FAULT:

1. CS MODULE

CONTROL BUS PARITY DETECTION TEST

PURPOSE:

TO TEST THE RM03'S PARITY CHECKING LOGIC FOR THE MASSBUS ASYNCHRONOUS CONTROL BUS.

PROCEDURE:

THIS TEST WRITES A SHIFTING ONE BIT DATA PATTERN IN THE DISK ADDRESS REGISTER USING 'PAT'' TO CONTROL THE STATE OF THE PARITY BIT. 'PAR'' STATUS, BIT 03 OF RMER1, IS CHECKED AFTER EACH PATTERN IS TRANSFERRED..NOTE THE FOLLOWING TABLE SHOWS A SET OF TEST PATTERNS THAT COULD BE USED INSTEAD OF A SHIFTING ONE BIT PATTERN.

DATA PATTERN	PAT	PAR
000000	1	1
075266	0	0
163753	0	0
116535	1	1

PROBABLE FAULT:

1. IF MODULE

2. ASSYNCHRONOUS MASSBUS MODULE

CONTROL BUS PARITY GENERATION TEST

PURPOSE:

TO TEST THE RM03'S PARITY GENERATING LOGIC FOR THE MASSBUS ASYNCHRONOUS CONTROL BUS.

PROCEDURE:

THE TEST TRANSFERS A SHIFTING ONE BIT DATA PATTERN TO THE DISK ADDRESS REGISTER. AFTER EACH PATTERN IS READ BACK, 'MASSBUS CONTROL BUS PARITY ERROR'' IS TESTED AND SHOULD BE ZERO..NOTE THE FOLLOWING SET OF TEST PATTERNS COULD BE USED INSTEAD OF THE SHIFTING ONE BIT PATTERN.

DATA PATTERN	MCPE
000000	0
056747	0
135672	0
163135	0

PROBABLE FAULT:

1. IF MODULE

2. ASYNCHRONOUS MASSBUS MODULE

RMDA, RMDC FAULT TEST

PURPOSE:

TO VERIFY THAT THERE ARE NOT FAULTS WHICH INHIBIT THE PROGRAM FROM WRITING RMDC AND RMDA. SPECIFICALLY, THESE FAULTS INCLUDE:

.''GO H'' STUCK HIGH, WHICH WOULD INHIBIT THE REGISTER LOAD FUNCTION,

.''RIP L'' STUCK LOW, WHICH WOULD CONSTANTLY CLEAR THE REGISTER

.''EBL'' STUCK, WHICH WOULD INHIBIT THE CLOCK FUNCTION.

PROCEDURE:

THE TEST WRITES AND READS BOTH RMDC, AND RMDA. WITH ZEROS, THEN ONES. THE TEST PASSES IF EITHER REGISTER CAN BE WRITTEN

```
                                        WITH ONES.
```

PROBABLE FAULT:

1. DS MODULE

2. IF MODULE

3. CS MODULE

DISK ADDRESS TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE DISK ADDRESS REGISTER IS NOT STUCK AT ONE OR ZERO, AND THAT THERE IS NOT BIT INTERFERENCE.

PROCEDURE:

THIS TEST PRESETS THE DISK ADDRESS TO A NONZERO VALUE, THEN USES A MOVE TO CLEAR THE REGISTER.THE TEST THEN READS RMDA AND VERIFIES THAT NONE OF THE BITS ARE STUCK AT ONE.

THEN THE TEST PRECLEARS THE MASSBUS ADAPTER DISK ADDRESS REGISTER (RMDA), LOADS IT TO ALL ONES, AND VERIFIES THAT NONE OF THE BITS ARE STUCK AT ZERO.

A SHIFTING ONE BIT PATTERN IS TRANSFERRED TO AND FROM THE DISK ADDRESS REGISTER, RMDA, AND THE TEST VERIFIES THAT EACH BIT IS INDEPENDENT.

PROBABLE FAULT:

1. DS MODULE

2. IF MODULE

DESIRED CYLINDER TRANSFER TEST

PURPOSE:

TO VERIFY THAT THE DESIRED CYLINDER ADDRESS REGISTER, RMDC, IS NOT STUCK AT ONE OR ZERO, AND THAT THERE IS NOT BIT INTERFERENCE.

933 PROCEDURE:

THIS TEST WRITES ONES IN THE DESIRED CYLINDER REGISTER RMDC, THEN WRITES ZEROS AND VERIFIES THAT THE REGISTER IS NOT STUCK AT ONE.

THEN THE TEST WRITES ZEROS IN THE DESIRED CYLINDER REGISTER, RMDC, WRITES ONES AND VERIFIES THAT THE REGISTER IS NOT STUCK AT ZERO.

FINALLY, A SHIFTING 1 BIT PATTERN IS TRANSFERRED TO AND FROM RMDC AND THE PROGRAM CHECKS FOR BIT INTERFERENCE.

PROBABLE FAULT:

1. DS MODULE

2. IF MODULE

ILLEGAL REGISTER TEST

PURPOSE:

TO TEST ILLEGAL REGISTER ERROR DETECTION IN THE RM03.

PROCEDURE:

THIS TEST READS ALL LEGAL REGISTERS AND VERIFIES THAT ''ILR'', BIT 2 OF RMER1 DOES NOT SET. THEN, TO THE EXTENT ALLOWED BY THE MASSBUS CONTROLLER, IT READS ILLEGAL REGISTERS AND VERIFIES THAT ''ILR'' IS SET.

PROBABLE FAULT:

1. IF MODULE

2. ASSYNCHRONOUS MASSBUS MODULE

RESET GO BY INIT TEST

PURPOSE:

TO VERIFY THAT GO CAN BE RESET BY INITIALIZE.

PROCEDURE:

THE TEST SETS GO THEN CLEARS GO USING MASSBUS INITIALIZE, I.E., CONTROLLER CLEAR.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

DIAGNOSTIC MODE TEST

PURPOSE:

TO VERIFY THAT 'DIAGNOSTIC MODE', BIT 0 OF RMMR1, IS NOT STUCK AT ONE OR ZERO.

PROCEDURE:

THE RM03 IS INITIALIZED AND 'DMD' IS CHECKED FOR ZERO. 'DMD' IS WRITTEN WITH ONE AND READ TO VERIFY THAT IT IS NOT STUCK AT ZERO, THEN WRITTEN WITH ZERO AND READ TO VERIFY THAT IT IS NOT STUCK AT ONE.

PROBABLE FAULT:

1. CS MODULE
2. IF MODULE

MOL TEST

PURPOSE:

TO VERIFY THAT 'MEDIUM ON LINE' STATUS CAN BE SET AND RESET USING MAINTENANCE UNIT READY.

```
                                                                N 2
1039
1040                                    PROCEDURE:
```

AFTER INITIALIZING THE SUBSYSTEM, THE TEST SETS 'DIAGNOSTIC MODE: AND READS THE DRIVE STATUS REGISTER, RMDS, EXPECTING MOL, BIT 12 TO BE ZERO. 'MAINTENANCE UNIT READY", BIT 9 OF RMMR1, IS SET AND MOL SHOULD BE ONE. THE TEST THEN WRITES A ZERO IN MUR AND READS RMDS, VERIFYING THAT 'MEDIUM ON LINE" IS ZERO.

PROBABLE FAULT:

1. CS MODULE

2. IF MODULE

WRITE LOCK TEST

PURPOSE:

TO VERIFY THAT 'WRITE LOCK" STATUS, WRL, CAN BE SET AND RESET USING 'MAINTENANCE WRITE PROTECT", MWP.

PROCEDURE:

WITH DIAGNOSTIC MODE SET, THE PROGRAM SETS MWP, BIT 03 OF RMMR1, AND READS RMDS TO VERIFY THAT WRL, BIT 11 IS SET. THEN MWP IS RESET AND WRL SHOULD BE ZERO.

PROBABLE FAULT:

1. CS MODULE

2. IF MODULE

DRIVE FAULT TEST

PURPOSE:

TO VERIFY THAT 'DEVICE CHECK", DVC, AND 'UNSAFE", UNS, CAN BE SET AND RESET USING 'MAINTENANCE DRIVE FAULT", MDF.

PROCEDURE:

WITH DIAGNOSTIC MODE SET, THE PROGRAM SETS MDF, BIT 06 OF RMMR1, AND READS RMER3 TO VERIFY THAT DVC, BIT 07 IS SET RMER1 IS ALSO READ AND UNS, BIT 14 SHOULD ALSO BE SET. THEN MDF IS RESET

1097 AND DVC AND UNS SHOULD BE RESET.

PROBABLE FAULT:

1. CS MODULE

2. IF MODULE

SEEK ERROR TEST

PURPOSE:

TO VERIFY THAT ''SEEK ERROR'', SKI, CAN BE SET AND RESET USING 'MAINTENANCE SEEK ERROR'', MSER.

PROCEDURE:

WITH DIAGNOSTIC MODE SET, THE TEST SETS MSER, BIT 07 OF RMMR1 AND READS RMER3 TO VERIFY THAT SKI, BIT 14 IS SET. MSER IS RESET AND SKI SHOULD RESET.

PROBABLE FAULT:

1. CS MODULE

2. IF MODULE

PIP TEST

PURPOSE:

TO VERIFY THAT 'POSITIONING IN PROGRESS'', PIP, CAN BE SET AND RESET USING 'MAINTENANCE ON CYLINDER'', MOC.

PROCEDURE:

DIAGNOSTIC MODE IS SET THEN MOC, BIT 08 OF RMMR1 IS SET AND PIP, BIT 13 OF RMDS, SHOULD BE ZERO. MOC IS THEN RESET AND PIP SHOULD BE ONE.

PROBABLE FAULT:

1. CS MODULE

2. IF MODULE

1154

EBL TEST

PURPOSE:

TO VERIFY THAT END OF BLOCK STATUS ''EBL'' CAN BE SET AND RESET USING DIAGNOSTIC END OF BLOCK 'DEBL''.

PROCEDURE:

THE PROGRAM SETS DIAGNOSTIC MODE AND VERIFIES THAT EBL IS RESET. THEN IT SETS DEBL AND VERIFIES THAT EBL IS SET. FINALLY, THE TEST TRANSFERS A SHIFTING ONE BIT TO RMMR1, AND CHECKS FOR DEBL BEING SET BY AN ADJACENT BIT.

PROBABLE FAULT:

1. CS MODULE

LAST SECTOR, LAST TRACK TEST

PURPOSE:

TO VERIFY THE DESIRED TRACK/SECTOR PLA ON THE DS MODULE USING RMMR1, BITS 01 AND 02.

PROCEDURE:

THE TEST WRITES ALL POSSIBLE PATTERNS IN THE DISK ADDRESS REGISTER, RMDA, AND VERIFIES ''LS'' AND ''LS/T'' STATUS FOR EACH PATTERN. THE PROCEDURE IS DONE ONCE FOR 16 BIT FORMAT AND ONCE FOR 18 BIT FORMAT.

PROBABLE FAULT:

1. DS MODULE

2. CS MODULE

RMDA COUNT TEST

PURPOSE

TO VERIFY THAT THE DISK ADDRESS REGISTER (RMDA) INCREMENTS

G 3

1211 PROPERLY.

PROCEDURE:

THE TEST INCREMENTS RMDA USING DIAGNOSTIC END OF BLOCK ''DEBL'' AND VERIFIES THE RESULT IN BOTH 16 AND 18 BIT FORMAT.

PROBABLE FAULT:

1. DS MODULE

RMDC COUNT TEST

PURPOSE:

TO VERIFY THAT THE DESIRED CYLINDER REGISTER, RMDC, INCREMENTS PROPERLY.

PROCEDURE:

THE PROGRAM INCREMENTS RMDC USING DIAGNOSTIC END OF BLOCK, ''DEBL'', VERIFYING THAT RMDC INCREMENTS THROUGH A COMPLETE CYCLE.

PROBABLE FAULT:

1. DS MODULE

LBT TEST

PURPOSE:

TO INSURE THAT LAST BLOCK TAKEN STATUS, ''LBT'', CLEARS WHEN RMDA IS WRITTEN, AND SETS WHEN THE LAST SECTOR IS TRANSFERRED.

PROCEDURE:

THE TEST USES DIAGNOSTIC EBL TO SET LBT, AND TRANSFERS TO RMDA TO RESET LBT.

PROBABLE FAULT:

1. DS MODULE

2. IF MODULE

COMPOSITE ERROR TEST

PURPOSE:

TO TEST ''COMPOSITE ERROR'', BIT 14 OF RMDS.

PROCEDURE:

THE TEST USES INITIALIZE AND DIAGNOSTIC MODE TO FORCE ALL ERRORS TO ZERO THEN VERIFIES THAT 'ERR'' IS ZERO. EACH ERROR IS INDIVIDUALLY SET AND 'ERR'' SHOULD BE ONE FOR EVERY ERROR TESTED. ADDRESSES #2 AND #17 OF THE COMPOSITE ERROR PLA ARE NOT TESTED. ''ABORT'' AND 'EXCEPTION'' OUTPUTS OF THE PLA ARE NOT TESTED. THE TEST FAILS IF ERR IS NOT ZERO WITH ALL SET ARGUMENTS ZERO OR IF ERR IS NOT ONE WITH ANY SET ARGUMENT ONE.

PROBABLE FAULT:

1. IF MODULE

WRITE GO TEST

PURPOSE:

TO VERIFY THAT GO CAN BE SET.

PROCEDURE:

THE TEST ENABLES THE DEBUG CLOCK, THEN TRANSFERS A NOP FUNCTION CODE AND GO BIT TO RMCS1, VERIFYING THAT GO SETS. ALL FUNCTION CODES ARE TESTED.

PROBABLE FAULT:

1. IF MODULE

2. CS MODULE

BRANCH MULTIPLEXOR TEST

PURPOSE:

TO VERIFY THAT THE OUTPUT OF THE COMMAND SEQUENCER BRANCH

1324 MULTIPLEXOR DOES NOT HAVE A FAULT.

PROCEDURE:

WITH DEBUG CLOCK ENABLED, THE TEST USES VARIOUS FUNCTION CODES AND REGISTER CONDITIONS TO ADDRESS THE TEST BIT MULTIPLEXOR SUCH THAT THE TEST BIT, BIT12 OF RMMR2, CAN BE CHECKED FOR A STUCK FAULT.

PROBABLE FAULT:

1. CS MODULE

SET/RESET GO TEST

PURPOSE:

TO VERIFY THAT GO CAN BE SET AND RESET.

PROCEDURE:

THE SUBSYSTEM IS INITIALIZED AND PUT IN DIAGNOSTIC MODE WITH 'DEBUG CLOCK ENABLE', BIT 14 OF RMMR1 SET.CERTAIN FUNCTION CODES ARE WRITTEN IN RMCS1 AND THE PROGRAM READS RMCS1 TO VERIFY THAT GO IS SET. RMDS IS ALSO READ TO VERIFY THAT 'DRY' IS RESET. THEN THE PROGRAM STEPS THE DEBUG CLOCK USING BIT 15 OF RMMR1 AND VERIFIES THAT 'GO' RESETS AND 'DRY' SETS. USING A FUNCTION CODE THAT RESETS GO AT A DIFFERENT PROM ADDRESS. THE TEST FAILS IF GO DOES NOT SET OR CANNOT BE RESET BY THE COMMAND SEQUENCER. THE TEST ALSO FAILS IF 'DRIVE READY' IS NOT THE COMPLIMENT OF GO.

PROBABLE FAULT:

1. CS MODULE

2. IF MODULE

END 1 RESET GO TEST

PURPOSE:

TO VERIFY THAT THE COMMAND SEQUENCER CAN RESET GO AT THE END1 LOCATION.

PROCEDURE:

THE TEST EXECUTES RELEASE, SEARCH AND ILLEGAL FUNCTION CODE 32 IN DIAGNOSTIC MODE AND VERIFIES THAT GO RESETS ON THE SPECIFIED CLOCK CYCLE.

PROBABLE FAULT:

1. CS MODULE

SET PULSE TEST

PURPOSE:

TO VERIFY THAT THE COMMAND SEQUENCER CAN GENERATE SET PULSE.

PROCEDURE:

WHICH DEBUG CLOCK ENABLED, THE TEST STEPS THE COMMAND SEQUENCER THROUGH PARTS OF VARIOUS FUNCTION CODES AND CHECKS CONTINUE, BIT 06 OF RMMR1 TO DETERMINE IF SET PULSE IS BEING GENERATED.

PROBABLE FAULT:

1. CS MODULE

SET/RESET IVC TEST

PURPOSE:

TO TEST ''INVALID COMMAND'' STATUS FOR EACH FUNCTION CODE.

PROCEDURE:

THE PROGRAM RESETS VOLUME VALID USING 'MAINTENANCE UNIT READY'', BIT09 OF RMMR1, THEN LOADS THE FUNCTION CODE AND GO IN RMCS1. EACH FUNCTION CODE IS TESTED AND ''IVC'', BIT 12 OF RMER2 IS CHECKED.

PROBABLE FAULT:

1. CS MODULE

2. IF MODULE

1437

SET LSC TEST

PURPOSE:

TO VERIFY THAT 'LOSS OF SYSTEM CLOCK'' CAN SET AND RESET.

PROCEDURE:

THE TEST ENABLES THE DEBUG CLOCK AND SETS THE GO BIT. AFTER WAITING ENOUGH TIME FOR THE ONE SHOT TO SET, THE TEST DISABLES THE DEBUG CLOCK AND VERIFIES THAT LSC SETS.

PROBABLE FAULT:

1. CS MODULE

2. IF MODULE

DECODE TEST

PURPOSE:

TO VERIFY THAT THE 'DECODE'' FLOP ON THE IF MODULE SETS WITH THE LEADING EDGE OF ''SET PULSE'' EXCEPT WHEN ''COMPOSITE ERROR'' IS ACTIVE.

PROCEDURE:

THE TEST USES ''VOLUME VALID'' AND 'OCCUPPIED'' TO DETERMINE IF THE DECODE FLOP IS SET OR RESET. INITIALLY, VV AND OCCUPPIED ARE RESET AND THE TEST EXECUTES THOSE COMMANDS WHICH SET VV OR OCC AND VERIFIES THAT ONE OR BOTH BITS SET. THE SAME COMMANDS ARE EXECUTED AGAIN WITH COMPOSITE ERROR SET, AND THE TEST VERIFIES THAT NEITHER BIT SETS.

PROBABLE FAULT:

1. IF MODULE

SET/RESET VOLUME VALID TEST

PURPOSE:

TO VERIFY THAT ''VOLUME VALID'' RESETS WITH THE LEADING EDGE OF UNIT READY, AND SETS WITH PACK ACKNOWLEDGE AND READ IN PRESET COMMANDS.

PROCEDURE:

USING ''MAINTENANCE UNIT READY'', BIT 9 OF RMMR1, THIS TEST FORCES A ZERO TO ONE TRANSITION OF UNIT READY AND VERIFIES THAT VOLUME VALID,BIT 6 OF RMDS IS ZERO. THEN THE TEST EXECUTES A PACK ACKNOWLEDGE COMMAND, VERIFYING THAT VV SETS. THE PROCEDURE IS REPEATED WITH A READ IN PRESET COMMAND.

PROBABLE FAULT:

1. IF MODULE

ILLEGAL FUNCTION TEST

PURPOSE:

TO TEST ILLEGAL FUNCTION ERROR IN THE RM03.

PROCEDURE:

WITH DIAGNOSTIC CLOCK ENABLED TO INHIBIT THE COMMAND SEQUENCER, THIS TEST VERIFIES THAT ''ILF'', BIT 0 OF RMER1, IS OFF FOR LEGAL FUNCTION CODES AND ON FOR ILLEGAL FUNCTIONCODES. THE STATUS OF THE ''GO'' BIT IS IGNORED.

PROBABLE FAULT:

1. IF MODULE

OCCUPIED TEST

PURPOSE:

TO VERIFY THAT ''OCCUPIED'' IS SET DURING DATA TRANSFERS AND IS RESET FOR ALL OTHER COMMANDS.

PROCEDURE:

FOR EACH DATA TRANSFER COMMAND, ''OCC'', BIT 15 OF RMMR1

```
1549                    SHOULD  BE  ONE,  DEBUG  CLOCK  IS  ENABLED  TO  PREVENT  GO FROM
1550                    RESETTING BEFORE STATUS IS SAMPLED.
```

PROBABLE FAULT:

1. IF MODULE

2. CS MODULE

READ IN PRESET TEST

PURPOSE:

TO VERIFY THAT "READ IN PRESET" COMMAND IS DECODED, AND IN PARTICULAR, TO VERIFY THAT "IF5 READ IN CMD L" IS NOT STUCK AT ONE.

PROCEDURE:

EACH VISIBLE STATUS OR REGISTER BIT WHICH IS CLEARED BY "READ IN PRESET" IS SET. THEN THE RIP COMMAND IS EXECUTED AND THE TEST VERIFIES THAT ONE OR MORE BITS ARE CLEARED. THE FOLLOWING ARE USED DURING THE TEST.

. BITS 10-12 OF RMOF ARE SET BY A MOVE INSTRUCTION AND THE TEST PASSES IF THESE BITS ARE ZERO AFTER THE RIP COMMAND.

. THE DESIRED CYLINDER REGISTER, RMDC, IS SET WITH A MOVE INSTRUCTION AND THE TEST PASSES IF BITS 00-03, OR BITS 04-07, OR BITS 08-09 ARE ZERO AFTER THE RIP COMMAND.

. THE DISK ADDRESS REGISTER, RMDA, IS SET WITH A MOVE INSTRUCTION AND THE TEST PASSES IF BITS 00-03, OR BITS 04-07, OR BITS 08-11, OR BITS 12-15 ARE ZERO AFTER THE RIP COMMAND.

THE TEST FAILS IF NONE OF THE PRESET TERMS ARE ZERO AFTER THE RIP COMMAND.

PROBABLE FAULT:

1. IF MODULE

RIP/RMOF TEST

PURPOSE:

TO VERIFY THAT ''READ IN PRESET'' RESETS FMT16, ECI AND HCI, BITS 10,11 AND 12 OF RMOF.

PROCEDURE:

FMT16, ECI AND HCI ARE SET, THEN A RIP COMMAND IS EXECUTED AND EACH BIT SHOULD BE ZERO.

PROBABLE FAULT:

1. IF MODULE

RMDA/RMDC/RIP TEST

PURPOSE:

TO VERIFY THAT ''READ IN PRESET'' RESETS THE DESIRED CYLINDER ADDRESS, RMDC, AND THE DISK ADDRESS, RMDA.

PROCEDURE:

RMDA AND RMDC ARE PRESET THEN TESTED FOR ZERO AFTER THE RIP COMMAND.

PROBABLE FAULT:

1. DS MODULE

OFFSET COMMAND TEST

PURPOSE:

TO VERIFY THAT ''OFFSET MODE'' SETS WITH OFFSET COMMAND.

PROCEDURE:

THE TEST EXECUTES OFFSET COMMAND AND VERIFIES THAT ''OM'', BIT 00 OF RMDS IS ONE.

PROBABLE FAULT:

1. IF MODULE

RETURN TO CENTER TEST

PURPOSE:

TO VERIFY THAT 'RETURN TO CENTER'' RESETS OFFSET MODE.

PROCEDURE:

OFFSET MODE, BIT 00 OF RMDS, IS SET WITH OFFSET COMMAND, THEN THE TEST EXECUTES A RETURN TO CENTER COMMAND AND VERIFIES THAT OFFSET MODE RESETS. OFFSET DIRECTION IS ALSO SET AND CHECKED FOR ZERO AFTER THE COMMAND.

PROBABLE FAULT:

1. IF MODULE

RMDC CLEAR OFFSET TEST

PURPOSE:

TO VERIFY THAT CLEAR OFFSET IS ACTIVE WHEN THE DESIRED CYLINDER ADDRESS IS WRITTEN.

PROCEDURE:

THE TEST EXECUTES AN OFFSET COMMAND, WRITES RMDC, AND VERIFIES THAT OM, BIT 00 OF RMDS IS ZERO.

PROBABLE FAULT:

1. DS MODULE

2. IF MODULE

EBL CLEAR OFFSET TEST

PURPOSE:

TO VERIFY THAT OFFSET MODE CLEARS WHEN HEAD SWITCHING OCCURS.

PROCEDURE:

THE TEST EXECUTES AN OFFSET COMMAND TO SET OFFSET MODE. AFTER SETTING THE FORMAT BIT AND LOADING THE LAST SECTOR/TRACK ADDRESS IN RMDA, THE TEST FORCES AN EBL AND VERIFIES THAT OFFSET MODE RESETS.

PROBABLE FAULT:

1. DS MODULE

RUN AND GO TEST

PURPOSE:

TO VERIFY THAT 'RUN AND GO'' FLOP SETS DURING READ AND WRITE COMMANDS.

PROCEDURE:

THE RM03 IS INITIALIZED AND A DATA TRANSFER COMMAND WITH GO SET IS WRITTEN IN RMCS1. 'RUN AND GO'', BIT 14 OF RMMR1 SHOULD BE ONE FOR EACH DATA COMMAND. THE DEBUG CLOCK IS ENABLED SO THAT GO DOES NOT RESET BEFORE STATUS IS TESTED.

PROBABLE FAULT:

1. CS MODULE

2. SYNCHRONOUS MASSBUS MODULE

SET IAE TEST

PURPOSE:

TO VERIFY THAT INVALID ADDRESS ERROR CAN SET.

PROCEDURE:

THE TEST LOADS INVALID SECTOR, TRACK AND CYLINDER ADDRESSES AND EXECUTES A SEARCH COMMAND, VERIFYING THAT ''IAE'' SETS. THE PROCESS IS REPEATED WITH A DIFFERENT COMMAND IF THE IAE DOES NOT SET, AND THE TEST FAILS IF IAE CANNOT BE SET.

PROBABLE FAULT:

1. DS MODULE

2. IF MODULE

SEARCH, SEEK, READ, WRITE TEST

PURPOSE:

TO VERIFY THAT THE ''SCH SK R OR W'' DECODE ON THE IF MODULE IS CORRECT FOR ALL FUNCTION CODES.

PROCEDURE:

THE TEST LOADS INVALID SECTOR, TRACK AND CYLINDER ADDRESSES AND EXECUTES EACH COMMAND TO WHERE SET PULSE IS ACTIVE AND VERIFIES THE DECODE BY CHECKING ''IAE''.

PROBABLE FAULT:

1. IF MODULE

INVALID SECTOR/TRACK TEST

PUPOSE:

TO VERIFY THAT INVALID SECTOR AND TRACK ADDRESSES ARE DETECTED.

PROCEDURE:

THE TEST LOADS THE TEST PATTERN IN RMDA AND EXECUTES A SEARCH COMMAND, VERIFYING THAT ''IAE'' SETS.

PROBABLE FAULT:

1. DS MODULE

2. TRACK ADDRESS OPTION JUMPER

INVALID CYLINDER TEST

PURPOSE:

TO VERIFY THAT INVALID CYLINDER ADDRESSES ARE DETECTED.

PROCEDURE:

THE TEST LOADS THE TEST PATTERN IN RMDC AND EXECUTES A SEARCH COMMAND, VERIFYING THAT ''IAE'' SETS.

PROBABLE FAULTS:

1. DS MODULE

2. CYLINDER ADDRESS OPTION JUMPER

SET AOE TEST

PRUPOSE:

TO VERIFY THAT ADDRESS OVERFLOW ERROR IS DETECTED.

PROCEDURE:

THE TEST LOADS THE ADDRESS OF THE LAST SECTOR IN RMDA AND RMDC, THEN INITIATES A DATA COMMAND WITH DEBUG CLOCK ENABLED. END OF BLOCK IS FORCED TO INCREMENT THE SECTOR ADDRESS, AND THE TEST VERIFIES THAT ''AOE'' IS SET.

PROBABLE FAULT:

1. DS MODULE

SET RMR TEST

PURPOSE:

TO VERIFY THAT ''REGISTER MODIFICATION REFUSED'' SETS WHEN A REGISTER IS WRITTEN WHILE GO IS SET, EXCEPT WHEN THE ATTENTION OR MAINTENANCE REGISTER IS WRITTEN.

PROCEDURE:

'DEBUG CLOCK ENABLE'' IS SET TO INHIBIT THE COMMAND SEQUENCER, THEN A NOP COMMAND AND GO BIT IS WRITTEN IN RMCS1. WITHOUT STEPPING THE DEBUG CLOCK, THE TEST WRITES RMMR AND RMAS, WHICH SHOULD NOT SET RMR STATUS. THEN RMDA IS WRITTEN AND RMR STATUS, BIT 02 OF RMER1, SHOULD BE ONE.

PROBABLE FAULT:

1. IF MODULE

PGM STATUS CHECK

PURPOSE:

TO VERIFY THAT THE PROGRAMMABLE STATUS BIT AND THE DRIVE REQUEST STATUS BIT ARE COMPATABLE.

PROCEDURE:

THE TEST REPORTS AN ERROR IF PGM IS ON AND DRQ IS OFF. PGM IS NOT PREDICTABLE IN THE CASE WHERE DRQ IS ON BECAUSE OF THE PORT SELECT SWITCH.

PROBABLE FAULT:

1. IF MODULE

DVA/DPR STATUS CHECK PURPOSE:

TO VERIFY THAT DRIVE PRESENT STATUS AND DEVICE AVAILABLE STATUS ARE SET.

PROCEDURE:

DPR AND DVA ARE TESTED AND BOTH SHOULD BE ON.

PROBABLE FAULT:

1. IF MODULE

PORT REQUEST TEST, PART 1/3

PURPOSE:

TO VERIFY THAT THE PORT REQUEST FLOPS ON THE IF MODULE SET WHEN THE PROGRAM READS RMCS1.

PROCEDURE:

THE TEST EXECUTES A RELEASE COMMAND, THEN, ASSUMING THE PORT IS RELEASED, IT READS RMCS1, THEN READS RMMR2 AND VERIFIES THAT ONE OF THE PORT REQUEST FLOPS IS SET.

PROBABLE FAULT:

1. IF MODULE

2. CS MODULE

PORT REQUEST TEST, PART 2/3

PURPOSE:

TO VERIFY THAT THE PORT REQUEST FLOPS ON THE IF MODULE SET WHEN THE PROGRAM WRITES RMAS.

PROCEDURE:

THE TEST EXECUTES A RELEASE COMMAND THEN WRITES RMAS AND READS RMMR2, VERIFYING THAT ONE OF THE REQUEST FLOPS IS SET.

PROBABLE FAULT:

1. IF MODULE

2. CS MODULE

DATA COMMAND TESTS

PURPOSE:

TO VERIFY THE COMMAND SEQUENCER DURING DATA COMMANDS.

1991 PROCEDURE:

THIS TEST, LIKE RECALIBRATE, SEEK, AND SEARCH TESTS, USES THE MAINTENANCE REGISTER TO SIMULATE DRIVE CONDITIONS AND FORCE THE COMMAND SEQUENCER THROUGH EACH BRANCH PATH. ADDITIONAL ITEMS WHICH ARE TESTED INCLUDE OFFSET PLUS AND MINUS ON THE TAG BUS AND 'ENABLE SEARCH', BIT 11 OF RMMR1.

PROBABLE FAULT:

1. CS MODULE

DATA SEQUENCER READ TEST

PURPOSE:

TO TEST THE CS AND DS MODULES FOR EXECUTION OF A READ COMMAND.

PROCEDURE:

THE RM03 IS INITIALIZED AND, USING DIAGNOSTIC MODE, A READ COMMAND IS STEPPED TO THE POINT WHERE THE COMMAND SEQUENCER ENABLES SEARCH. THE PROGRAM THEN STARTS STEPPING THE DATA SEQUENCER ON THE CS MODULE AND MONITORS HARDWARE OPERATION USING THE FOLLOWING STATUS:

.PLFS, BIT 10 OF RMMR1, SHOWS THE PERIOD WHEN THE DATA PROM HAS ENABLED SYNC BYTE DETECTION;

.PDA, BIT 08 OF RMMR1, SHOWS WHEN THE DATA PROM IS IN THE DATA AREA OF THE SECTOR;

.PHA, BIT 07 OF RMMR1, SHOWS WHEN THE DATA PROM IS IN THE HEADER AREA OF THE SECTOR;

.BB00, BIT00 OF RMMR2, SHOWS THE STATE OF WRITE GATE PROVIDING TAG IS HIGH;

.BB01, BIT 01 OF RMMR2, SHOWS THE STATE OF READ GATE PROVING TAG IS HIGH;

.WC, BIT 05 OF RMMR1, SHOWS THE STATE OF WORD CLOCK;

.EECC, BIT 04 OF RMMR1, SHOWS WHEN ECC CHARACTER IS ENABLED DURING WRITE;

.ECRC, BIT 09 OF RMMR1, SHOWS WHEN CRC CHARACTER IS ENABLED DURING WRITE;

.WD, BIT 03 OF RMMR1, SHOWS THE STATE OF WRITE DATA.

ADDITIONALLY, STATUS BITS IN RMER1 ARE USED TO TEST COMMAND EXECUTION. THE WRITE ONLY BITS OF RMMR1 ARE USED TO SIMULATE DRIVE STATUS AND CONTROL SIGNALS, AND TO SPECIFY READ DATA.

PROBABLE FAULT:

1. CS MODULE

2. DS MODULE

DATA SEQUENCER WRITE TEST

PURPOSE:

TO TEST THE CS AND DS MODULES FOR EXECUTION OF A WRITE COMMAND.

PROCEDURE:

THE PROCEDURE IS THE SAME AS THE DATA SEQUENCER READ TEST.

PROBABLE FAULT:

1. CS MODULE

2. DS MODULE

DATA SEQUENCER FORMAT TEST

PURPOSE:

TO TEST CS AND DS MODULES FOR EXECUTION OF A FORMAT COMMAND.

PROCEDURE:

THE PROCEDURE IS THE SAME AS THE DATA SEQUENCER READ TEST.

PROBABLE FAULT:

1. CS MODULE

2. DS MODULE

18 BIT WORD CLOCK TEST

PURPOSE:

TO TEST THE GENERATION OF WORD CLOCK IN 18 BIT FORMAT.

PROCEDURE:

DIAGNOSTIC MODE IS USED TO EXECUTE A READ COMMAND. THE PROGRAM VERIFIES THAT WORD CLOCK SWITCHES TO 18 BIT FORMAT AT THE DATA AREA AND SWITCHES BACK TO 16 BIT FORMAT DURING ECC. THE TEST IS REPEATED FOR A WRITE COMMAND.

PROBABLE FAULT:

1. CS MODULE

2. DS MODULE

HCRC SET TEST

PURPOSE:

TO VERIFY THAT 'HCRC' STATUS SETS WHEN THE HEADER CRC WORD IS IN ERROR.

PROCEDURE:

USING DIAGNOSTIC MODE TO EXECUTE A READ COMMAND, THE PROGRAM STEPS THE DATA SEQUENCER THROUGH THE HEADER AND PROVIDES READ DATA WHICH SHOULD CAUSE A CRC ERROR. THE TEST FAILS IF 'HCRC', BIT 08 OF RMER1 DOES NOT SET.

PROBABLE FAULT:

1. DS MODULE

2. IF MODULE

HCE SET TEST

PURPOSE:

TO VERIFY THAT ''HCE'' STATUS SETS IF THE HEADER DOES NOT COMPARE WITH RMDC AND RMDA.

PROCEDURE:

THE PROGRAM EXECUTES A READ COMMAND USING DIAGNOSTIC MODE. READ DATA IS SUPPLIED WHICH CAUSES A HEADER COMPARE ERROR IN BITS 00-03 OF HEADER WORD ONE AND THE PROGRAM VERIFIES THAT ''HCE'', BIT 07 OF RMER1 IS ONE. THIS PROCESS IS REPEATED FOR BITS 04-07 AND 08-11.

PROBABLE FAULT:

1. DS MODULE

2. CS MODULE

3. IF MODULE

FER SET TEST

PURPOSE:

TO VERIFY THAT ''FER'' STATUS SETS WHEN THE FORMAT BIT DOES NOT COMPARE WITH THE HEADER.

PROCEDURE:

THIS PROGRAM EXECUTES A READ COMMAND USING DIAGNOSTIC MODE AND FORCES A FORMAT ERROR IN THE FIRST HEADER WORD. THE TEST FAILS IF ''FER'', BIT 04 OF RMER1 DOES NOT SET.

PROBABLE FAULT:

1. DS MODULE

2. CS MODULE

3. IF MODULE

DCK SET TEST

PURPOSE:

TO VERIFY THAT "DCK" STATUS SETS WHEN A CORRECTABLE ECC ERROR OCCURRS.

PROCEDURE:

THE PROGRAM SIMULATES A READ OPERATION AND FORCES AN ECC ERROR WHICH IS CORRECTABLE. "DCK", BIT 15 OF RMER1 SHOULD BE SET AND "ECH", BIT 06 OF RMER1 SHOULD BE RESET. RMEC1 AND RMEC2 SHOULD PERMIT DATA CORRECTION.

PROBABLE FAULT:

1. DS MODULE
2. CS MODULE
3. IF MODULE

ECH SET TEST

PURPOSE:

TO VERIFY THAT "ECH" STATUS SETS WHEN A NONCORRECTABLE ECC ERROR OCCURRS DURING READ.

PROCEDURE:

THIS TEST IS LIKE THE DCK SET TEST EXCEPT THE PROGRAM USES AN UNCORRECTABLE ERROR BURST.

PROBABLE FAULT:

1. DS MODULE
2. CS MODULE
3. IF MODULE

DVA/DPR

STATUS CHECK PURPOSE:

TO VERIFY THAT DRIVE PRESENT STATUS AND DEVICE AVAILABLE STATUS ARE SET.

PROCEDURE:

DPR AND DVA ARE TESTED AND BOTH SHOULD BE ON.

PROBABLE FAULT:

1. IF MODULE

PGM STATUS CHECK

PURPOSE:

TO VERIFY THAT THE PROGRAMMABLE STATUS BIT AND THE DRIVE REQUEST STATUS BIT ARE COMPATABLE.

PROCEDURE:

THE TEST REPORTS AN ERROR IF PGM IS ON AND DRQ IS OFF. PGM IS NOT PREDICTABLE IN THE CASE WHERE DRQ IS ON BECAUSE OF THE PORT SELECT SWITCH.

PROBABLE FAULT:

1. IF MODULE

SET AOE TEST

PRUPOSE:

TO VERIFY THAT ADDRESS OVERFLOW ERROR IS DETECTED.

PROCEDURE:

THE TEST LOADS THE ADDRESS OF THE LAST SECTOR IN RMDA AND RMDC, THEN INITIATES A DATA COMMAND WITH DEBUG CLOCK ENABLED.

```
2314                    END OF BLOCK IS FORCED TO INCREMENT THE SECTOR ADDRESS,  AND  THE
2315                    TEST VERIFIES THAT ''AOE'' IS SET.
```

PROBABLE FAULT:

1. DS MODULE

CLEAR OFFSET STUCK ACTIVE TEST

PURPOSE:

TO VERIFY THAT THE SIGNAL WHICH CLEARS OFFSET MODE IS NOT STUCK IN ACTIVE STATE.

PROCEDURE:

THE TEST WRITES A ONE IN THE OFFSET DIRECTION BIT WHICH IS CLEARED BY THE SIGNAL AND VERIFIES THAT A ONE CAN BE READ BACK.

PROBABLE FAULT:

1. IF MODULE

2. DS MODULE

INVALID SECTOR/TRACK TEST

PUPOSE:

TO VERIFY THAT INVALID SECTOR AND TRACK ADDRESSES ARE DETECTED.

PROCEDURE:

THE TEST LOADS THE TEST PATTERN IN RMDA AND EXECUTES A SEARCH COMMAND, VERIFYING THAT "IAE" SETS.

PROBABLE FAULT:

1. DS MODULE

2. TRACK ADDRESS OPTION JUMPER

INVALID CYLINDER TEST

PURPOSE:

TO VERIFY THAT INVALID CYLINDER ADDRESSES ARE DETECTED.

PROCEDURE:

THE TEST LOADS THE TEST PATTERN IN RMDC AND EXECUTES A SEARCH COMMAND, VERIFYING THAT ''IAE'' SETS.

PROBABLE FAULTS:

1. DS MODULE

2. CYLINDER ADDRESS OPTION JUMPER

SEARCH, SEEK, READ, WRITE TEST

PURPOSE:

TO VERIFY THAT THE ''SCH SK R OR W'' DECODE ON THE IF MODULE IS CORRECT FOR ALL FUNCTION CODES.

PROCEDURE:

THE TEST LOADS INVALID SECTOR, TRACK AND CYLINDER ADDRESSES AND EXECUTES EACH COMMAND TO WHERE SET PULSE IS ACTIVE AND VERIFIES THE DECODE BY CHECKING ''IAE''.

PROBABLE FAULT:

1. IF MODULE

EBL CLEAR OFFSET TEST

PURPOSE:

TO VERIFY THAT OFFSET MODE CLEARS WHEN HEAD SWITCHING OCCURS.

PROCEDURE:

THE TEST EXECUTES AN OFFSET COMMAND TO SET OFFSET MODE. AFTER SETTING THE FORMAT BIT AND LOADING THE LAST SECTOR/TRACK ADDRESS IN RMDA, THE TEST FORCES AN EBL AND VERIFIES THAT OFFSET MODE RESETS.

PROBABLE FAULT:

1. DS MODULE

SET LSC TEST

PURPOSE:

TO VERIFY THAT ''LOSS OF SYSTEM CLOCK'' CAN SET AND RESET.

PROCEDURE:

THE TEST ENABLES THE DEBUG CLOCK AND SETS THE GO BIT. AFTER WAITING ENOUGH TIME FOR THE ONE SHOT TO SET, THE TEST DISABLES THE DEBUG CLOCK AND VERIFIES THAT LSC SETS.

PROBABLE FAULT:

1. CS MODULE

2. IF MODULE

DECODE TEST

PURPOSE:

TO VERIFY THAT THE ''DECODE'' FLOP ON THE IF MODULE SETS WITH THE LEADING EDGE OF ''SET PULSE'' EXCEPT WHEN ''COMPOSITE ERROR'' IS ACTIVE.

PROCEDURE:

THE TEST USES ''VOLUME VALID'' AND ''OCCUPPIED'' TO DETERMINE IF THE DECODE FLOP IS SET OR RESET. INITIALLY, VV AND OCCUPPIED ARE RESET AND THE TEST EXECUTES THOSE COMMANDS WHICH SET VV OR OCC AND VERIFIES THAT ONE OR BOTH BITS SET. THE SAME COMMANDS ARE EXECUTED AGAIN WITH COMPOSITE ERROR SET, AND THE TEST VERIFIES THAT NEITHER BIT SETS.

```
2480
2481                                    PROBABLE FAULT:
```

1. IF MODULE

SET IAE TEST

PURPOSE:

TO VERIFY THAT INVALID ADDRESS ERROR CAN SET.

PROCEDURE:

THE TEST LOADS INVALID SECTOR, TRACK AND CYLINDER ADDRESSES AND EXECUTES A SEARCH COMMAND, VERIFYING THAT "IAE" SETS. THE PROCESS IS REPEATED WITH A DIFFERENT COMMAND IF THE IAE DOES NOT SET, AND THE TEST FAILS IF IAE CANNOT BE SET.

PROBABLE FAULT:

1. DS MODULE

2. IF MODULE

PORT REQUEST TEST, PART 3/3

PURPOSE:

TO VERIFY THAT PORT REQUEST SETS WHEN ANY REGISTER EXCEPT RMAS IS WRITTEN.

PROCEDURE:

THE TEST WRITES THE DISK ADDRESS REGISTER AND VERIFIES THAT THE PORT REQUEST FLOP IS ON.

PROBABLE FAULT:

1. IF MODULE

2. CS MODULE

WRITE ATA TEST

PURPOSE:

TO VERIFY THAT ATTENTION CAN BE CLEARED BY WRITING THE ATTENTION SUMMARY REGISTER.

PROCEDURE:

THE PROGRAM RESETS AND SETS UNIT READY WHICH SHOULD CAUSE AN ATTENTION, THEN WRITES THE ATTENTION SUMMARY REGISTER AND VERIFIES THAT ATTENTION IS RESET.

PROBABLE FAULT:

1. IF MODULE

2. CS MODULE

RESET ATA BY GO TEST

PURPOSE:

TO VERIFY THAT ATA RESETS WHEN GO IS ON AND COMPOSITE ERROR IS OFF.

PROCEDURE:

THE PROGRAM SETS MAINTENANCE UNIT READY WHICH SHOULD CAUSE AN ATTENTION. THEN, WITH DEBUG CLOCK ENABLED, GO IS SET, AND ATA SHOULD BE ZERO.

PROBABLE FAULT:

1. IF MODULE

UNIT READY ATA TEST

PURPOSE:

TO VERIFY THAT ONE-ZERO AND ZERO-ONE TRANSITIONS OF UNIT READY SET ATTENTION.

2590 PROCEDURE:

THE TEST USES DIAGNOSTIC MODE TO FORCE BOTH TRANSITIONS OF UNIT READY AND VERIFIES THAT ATA SETS WITH EACH TRANSITION.

PROBABLE FAULT:

1. IF MODULE

ERROR ATA TEST

PURPOSE:

TO VERIFY THAT ATTENTION SETS WHEN COMPOSITE ERROR OCCURS WHILE GO IS OFF.

PROCEDURE:

THE PROGRAM CLEARS THE DEVICE AND SETS AN ERROR, THEN VERIFIES ATA IS ON.

PROBABLE FAULT:

1. IF MODULE

REGISTER TRANSFER ATA TEST

TO VERIFY THAT ATTENTION SETS WHEN ANY REGISTER, EXCEPT FOR RMAS AND RMCS, IS WRITTEN WHILE COMP ERROR IS SET.

PROCEDURE:

THE PROGRAM FORCES AN ERROR THEN RESETS ATTENTION FROM THE ERROR. THE PROGRAM THEN WRITES RMAS AND RMCS AND VERIFIES THAT NO ATTEN N OCCURS, AND WRITES RMDC AND VERIFIES THAT ATTENTION DOES OCCU...

PROBABLE FAULT:

1. IF MODULE

```
                        ;PROGRAM REVISION #001

                        .TITLE  CZRMJC0 RM03/2 DSKLS PRT 1
                        ;*COPYRIGHT (C) 1977
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY DOUG RIIKONEN
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                        ;*
000001                  $TN=1
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        11            INHIBIT ITERATIONS
                        ;*        10            BELL ON ERROR
                        ;*         9            LOOP ON ERROR
                        ;*         8            LOOP ON TEST IN SWR<7:0>
                        ;*         7            TN128
                        ;*         6            TN64
                        ;*         5            TN32
                        ;*         4            TN16
                        ;*         3            TN8
                        ;*         2            TN4
                        ;*         1            TN2
                        ;*         0            TN1
                        .SBTTL  BASIC DEFINITIONS

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

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

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
000003                  R3=     %3              ;;GENERAL REGISTER
```

```
2701    000004                  R4=     %4              ;;GENERAL REGISTER
2702    000005                  R5=     %5              ;;GENERAL REGISTER
2703    000006                  R6=     %6              ;;GENERAL REGISTER
2704    000007                  R7=     %7              ;;GENERAL REGISTER
2705    000006                  SP=     %6              ;;STACK POINTER
2706    000007                  PC=     %7              ;;PROGRAM COUNTER
2707
2708                            ;*PRIORITY LEVEL DEFINITIONS
2709    000000                  PR0=    0               ;;PRIORITY LEVEL 0
2710    000040                  PR1=    40              ;;PRIORITY LEVEL 1
2711    000100                  PR2=    100             ;;PRIORITY LEVEL 2
2712    000140                  PR3=    140             ;;PRIORITY LEVEL 3
2713    000200                  PR4=    200             ;;PRIORITY LEVEL 4
2714    000240                  PR5=    240             ;;PRIORITY LEVEL 5
2715    000300                  PR6=    300             ;;PRIORITY LEVEL 6
2716    000340                  PR7=    340             ;;PRIORITY LEVEL 7
2717
2718                            ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
2719    100000                  SW15=   100000
2720    040000                  SW14=   40000
2721    020000                  SW13=   20000
2722    010000                  SW12=   10000
2723    004000                  SW11=   4000
2724    002000                  SW10=   2000
2725    001000                  SW09=   1000
2726    000400                  SW08=   400
2727    000200                  SW07=   200
2728    000100                  SW06=   100
2729    000040                  SW05=   40
2730    000020                  SW04=   20
2731    000010                  SW03=   10
2732    000004                  SW02=   4
2733    000002                  SW01=   2
2734    000001                  SW00=   1
2735                            .EQUIV  SW09,SW9
2736                            .EQUIV  SW08,SW8
2737                            .EQUIV  SW07,SW7
2738                            .EQUIV  SW06,SW6
2739                            .EQUIV  SW05,SW5
2740                            .EQUIV  SW04,SW4
2741                            .EQUIV  SW03,SW3
2742                            .EQUIV  SW02,SW2
2743                            .EQUIV  SW01,SW1
2744                            .EQUIV  SW00,SW0
2745
2746                            ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
2747    100000                  BIT15=  100000
2748    040000                  BIT14=  40000
2749    020000                  BIT13=  20000
2750    010000                  BIT12=  10000
2751    004000                  BIT11=  4000
2752    002000                  BIT10=  2000
2753    001000                  BIT09=  1000
2754    000400                  BIT08=  400
2755    000200                  BIT07=  200
2756    000100                  BIT06=  100
```

```
2757    000040          BIT05=  40
2758    000020          BIT04=  20
2759    000010          BIT03=  10
2760    000004          BIT02=  4
2761    000002          BIT01=  2
2762    000001          BIT00=  1
2763                    .EQUIV  BIT09,BIT9
2764                    .EQUIV  BIT08,BIT8
2765                    .EQUIV  BIT07,BIT7
2766                    .EQUIV  BIT06,BIT6
2767                    .EQUIV  BIT05,BIT5
2768                    .EQUIV  BIT04,BIT4
2769                    .EQUIV  BIT03,BIT3
2770                    .EQUIV  BIT02,BIT2
2771                    .EQUIV  BIT01,BIT1
2772                    .EQUIV  BIT00,BIT0
2773
2774                    ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
2775    000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
2776    000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
2777    000014          TBITVEC=14              ;;''T'' BIT
2778    000014          TRTVEC= 14              ;;TRACE TRAP
2779    000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
2780    000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
2781    000024          PWRVEC= 24              ;;POWER FAIL
2782    000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
2783    000034          TRAPVEC=34              ;;''TRAP'' TRAP
2784    000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
2785    000064          TPVEC=  64              ;;TTY PRINTER VECTOR
2786    000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
2787
2788                    .SBTTL  RM03 REGISTER BIT DEFINITIONS
2789
2790                    ;RMCS1  CONTROL STATUS REGISTER
2791
2792    004000          DVA     =       BIT11           ;DEVICE AVAILABLE-READ ONLY
2793    000040          F4      =       BIT05           ;FUNCTION CODE
2794    000020          F3      =       BIT04           ;FUNCTION CODE
2795    000010          F2      =       BIT03           ;FUNCTION CODE
2796    000004          F1      =       BIT02           ;FUNCTION CODE
2797    000002          F0      =       BIT01           ;FUNCTION CODE
2798    000001          GO      =       BIT00           ;GO BIT
2799    000077          FNCMSK  =       000077          ;FUNCTION CODE MASK
2800
2801                    ;FUNCTION CODES (BITS 01-05 OF RMCS1)
2802
2803    000000          NOP     =       000000          ;NOP COMMAND
2804    000002          ILF02   =       000002          ;ILLEGAL COMMAND
2805    000004          SEEK    =       000004          ;SEEK COMMAND
2806    000006          RECAL   =       000006          ;RECALIBRATE COMMAND
2807    000010          DRVCLR  =       000010          ;DRIVE CLEAR COMMAND
2808    000012          RLEASE  =       000012          ;RELEASE COMMAND
2809    000014          OFFSET  =       000014          ;OFFSET COMMAND
2810    000016          RTC     =       000016          ;RETURN TO CENTERLINE COMMAND
2811    000020          RIP     =       000020          ;READ IN PRESET COMMAND
2812    000022          PAKACK  =       000022          ;PACK ACKNOWLEDGE COMMAND
```

```
2813    000022                  PACACK  =       PAKACK
2814    000024                  ILF24   =       000024          ;ILLEGAL COMMAND
2815    000026                  ILF26   =       000026          ;ILLEGAL COMMAND
2816    000030                  SEARCH  =       000030          ;SEARCH COMMAND
2817    000030                  ILF30   =       000030          ;ILLEGAL COMMAND
2818    000032                  ILF32   =       000032          ;ILLEGAL COMMAND
2819    000034                  ILF34   =       000034          ;ILLEGAL COMMAND
2820    000036                  ILF36   =       000036          ;ILLEGAL COMMAND
2821    000040                  ILF40   =       000040          ;ILLEGAL COMMAND
2822    000042                  ILF42   =       000042          ;ILLEGAL COMMAND
2823    000044                  ILF44   =       000044          ;ILLEGAL COMMAND
2824    000046                  ILF46   =       000046          ;ILLEGAL COMMAND
2825    000050                  WCD     =       000050          ;WRITE CHECK DATA COMMAND
2826    000052                  WCH     =       000052          ;WRITE CHECK HEADER AND DATA
2827    000054                  ILF54   =       000054          ;ILLEGAL COMMAND
2828    000056                  ILF56   =       000056          ;ILLEGAL COMMAND
2829    000060                  WD      =       000060          ;WRITE DATA COMMAND
2830    000062                  WH      =       000062          ;WRITE HEADER AND DATA COMMAND
2831    000064                  ILF64   =       000064          ;ILLEGAL COMMAND
2832    000066                  ILF66   =       000066          ;ILLEGAL COMMAND
2833    000070                  RD      =       000070          ;READ DATA COMMAND
2834    000072                  RH      =       000072          ;READ HEADER AND DATA COMMAND
2835    000074                  ILF74   =       000074          ;ILLEGAL COMMAND
2836    000076                  ILF76   =       000076          ;ILLEGAL COMMAND
2837
2838                            ;RMDA   DISK ADDRESS REGISTER
2839
2840    002000                  TA4     =       BIT10           ;TRACK ADDRESS 4
2841    001000                  TA2     =       BIT09           ;TRACK ADDRESS 2
2842    000400                  TA1     =       BIT08           ;TRACK ADDRESS 1
2843    000020                  SA16    =       BIT04           ;SECTOR ADDRESS 16
2844    000010                  SA8     =       BIT03           ;SECTOR ADDRESS 8
2845    000004                  SA4     =       BIT02           ;SECTOR ADDRESS 4
2846    000002                  SA2     =       BIT01           ;SECTOR ADDRESS 2
2847    000001                  SA1     =       BIT00           ;SECTOR ADDRESS 1
2848
2849                            ;TRACK,SECTOR MASKS
2850
2851    003400                  TADMSK  =       003400          ;TRACK ADDRESS MASK
2852    000037                  SADMSK  =       000037          ;SECTOR ADDRESS MASK
2853
2854                            ;RMDS   DRIVE STATUS REGISTER
2855
2856    100000                  ATA     =       BIT15           ;ATTENTION ACTIVE
2857    040000                  ERR     =       BIT14           ;COMPOSITE ERROR
2858    020000                  PIP     =       BIT13           ;POSITIONING IN PROGRESS
2859    010000                  MOL     =       BIT12           ;MEDIUM ON LINE
2860    004000                  WRL     =       BIT11           ;WRITE LOCK
2861    002000                  LBT     =       BIT10           ;LAST BLOCK TRANSFERRED
2862    001000                  PGM     =       BIT09           ;PROGRAMMABLE
2863    000400                  DPR     =       BIT08           ;DRIVE PRESENT
2864    000200                  DRY     =       BIT07           ;DRIVE READY
2865    000100                  VV      =       BIT06           ;VOLUME VALID
2866    000001                  OM      =       BIT00           ;OFFSET MODE ACTIVE
2867
2868                            ;RMER1  ERROR REGISTER #1
```

```
2869
2870          100000                  DCK     =       BIT15           ;DATA CHECK ERROR
2871          040000                  UNS     =       BIT14           ;DRIVE UNSAFE
2872          020000                  OPI     =       BIT13           ;OPERATION INCOMPLETE
2873          010000                  DTE     =       BIT12           ;DRIVE TIMING ERROR
2874          004000                  WLE     =       BIT11           ;WRITE LOCK ERROR
2875          002000                  IAE     =       BIT10           ;INVALID ADDRESS ERROR
2876          001000                  AOE     =       BIT09           ;ADDRESS OVERFLOW ERROR
2877          000400                  HCRC    =       BIT08           ;HEADER CRC ERROR
2878          000200                  HCE     =       BIT07           ;HEADER COMPARE ERROR
2879          000100                  ECH     =       BIT06           ;ECC 'HARD' ERROR
2880          000040                  WCF     =       BIT05           ;WRITE CLOCK FAILURE
2881          000020                  FER     =       BIT04           ;FORMAT ERROR
2882          000010                  PAR     =       BIT03           ;PARITY ERROR
2883          000004                  RMR     =       BIT02           ;REGISTER MODIFICATION REFUSED
2884          000002                  ILR     =       BIT01           ;ILLEGAL REGISTER
2885          000001                  ILF     =       BIT00           ;ILLEGAL FUNCTION
2886
2887          115760                  NDTMSK  =       DCK!DTE!WLE!AOE!HCRC!HCE!ECH!WCF!FER
2888                                  ;'NDTMSK' IS USED TO MASK ERROR REGISTER 1 DURING NON - DATA
2889                                  ;COMMANDS, I.E., HOUSEKEEPING AND POSITIONING COMMANDS
2890
2891                                  ;RMAS   ATTENTION SUMMARY REGISTER
2892
2893          000377                  ATNMSK  =       377             ;MASK FOR ATTENTION BITS
2894
2895                                  ;RMLA   LOOK AHEAD REGISTER
2896
2897          002000                  SC4     =       BIT10           ;SECTOR COUNT = 16
2898          001000                  SC3     =       BIT09           ;SECTOR COUNT = 8
2899          000400                  SC2     =       BIT08           ;SECTOR COUNT = 4
2900          000200                  SC1     =       BIT07           ;SECTOR COUNT = 2
2901          000100                  SC0     =       BIT06           ;SECTOR COUNT = 1
2902
2903          003700                  SCTMSK  =       003700          ;SECTOR COUNT MASK
2904
2905                                  ;RMMR   MAINTENANCE REGISTER
2906
2907                                  ;       WRITE ONLY BITS
2908
2909          100000                  DBCK    =       BIT15           ;DEBUG CLOCK
2910          040000                  DBEN    =       BIT14           ;DEBUG CLOCK ENABLE
```

```
2911    020000          DEBL    =       BIT13           ;DIAGNOSTIC END OF BLOCK
2912    010000          DTO     =       BIT12           ;DIAGNOSTIC TIMEOUT
2913    004000          MCLK    =       BIT11           ;MAINTENANCE CLOCK
2914    002000          MRD     =       BIT10           ;READ DATA
2915    001000          MUR     =       BIT09           ;UNIT READY
2916    000400          MOC     =       BIT08           ;ON CYLINDER
2917    000200          MSER    =       BIT07           ;SEEK ERROR
2918    000100          MDF     =       BIT06           ;DRIVE FAULT
2919    000040          MS      =       BIT05           ;SECTOR PULSE
2920    000010          MWP     =       BIT03           ;WRITE PROTECT
2921    000004          MI      =       BIT02           ;INDEX PULSE
2922    000002          MSC     =       BIT01           ;SECTOR COMPARE
2923    000001          DMD     =       BIT00           ;DIAGNOSTIC MODE
2924
2925                    ;       READ ONLY BITS
2926
2927    100000          OCC     =       BIT15           ;OCCUPIED
2928    040000          RG      =       BIT14           ;RUN AND GO
2929    020000          EBL     =       BIT13           ;END OF BLOCK
2930    010000          REX     =       BIT12           ;EXCEPTION
2931    004000          ESRC    =       BIT11           ;ENABLE SEARCH
2932    002000          PLFS    =       BIT10           ;LOOKING FOR SYNC
2933    001000          ECRC    =       BIT09           ;ENABLE CRC OUT
2934    000400          PDA     =       BIT08           ;DATA AREA
2935    000200          PHA     =       BIT07           ;HEADER AREA
2936    000100          CONT    =       BIT06           ;CONTINUE
2937    000040          WC      =       BIT05           ;WORD CLOCK
2938    000020          EECC    =       BIT04           ;ENABLE ECC OUT
2939    000010          MWD     =       BIT03           ;WRITE DATA BIT
2940    000004          LS      =       BIT02           ;LAST SECTOR
2941    000002          LST     =       BIT01           ;LAST SECTOR AND TRACK
2942    000001          DMD     =       BIT00           ;DIAGNOSTIC MODE
2943
2944                    ;RMDT   DRIVE TYPE REGISTER
2945
2946    100000          NSA     =       BIT15           ;NOT SECTOR ADDRESSED=0
2947    040000          TAP     =       BIT14           ;TAPE DRIVE = 0
2948    020000          MOH     =       BIT13           ;MOVING HEAD = 1
2949    004000          DRQ     =       BIT11           ;DRIVE REQUEST REQUIRED
2950
2951    020024          SNGPRT  =       020024          ;SINGLE PORT DRIVE TYPE
2952    024024          DULPRT  =       024024          ;DUAL PORT DRIVE TYPE
2953
2954                    ;RMOF   OFFSET REGISTER
2955
2956    010000          FMT16   =       BIT12           ;16 BIT WORD FORMAT
2957    004000          ECI     =       BIT11           ;ECC INHIBIT
2958    002000          HCI     =       BIT10           ;HEADER COMPARE INHIBIT
2959    000200          OFD     =       BIT07           ;OFFSET FORWARD
2960    161577          XNUOF   =       161577          ;UNUSED BITS OF RMOF
2961
2962
2963                    ;RMDC   DESIRED CYLINDER ADDRESS REGISTER
2964    001777          CYLMSK  =       1777            ;MASK FOR CYLINDER ADDRESS
2965    176000          XNUDC   =       176000          ;UNUSED BITS OF RMDC
2966
```

```
                        ;RMMR2  MAINTENANCE REGISTER #2
                        ;       READ ONLY BITS
100000                  RQA     =       BIT15           ;PORT A REQUEST
040000                  RQB     =       BIT14           ;PORT B REQUEST
020000                  TAG     =       BIT13           ;TAG CONTROL
010000                  TST     =       BIT12           ;COMMAND SEQUENCE TEST BIT
004000                  CC      =       BIT11           ;CONTROL OR CYLINDER TAG
002000                  CH      =       BIT10           ;CONTROL OR HEAD TAG
001000                  BB09    =       BIT09           ;TAG BUS
000400                  BB08    =       BIT08           ;TAG BUS
000200                  BB07    =       BIT07           ;TAG BUS
000100                  BB06    =       BIT06           ;TAG BUS
000040                  BB05    =       BIT05           ;TAG BUS
000020                  BB04    =       BIT04           ;TAG BUS
000010                  BB03    =       BIT03           ;TAG BUS
000004                  BB02    =       BIT02           ;TAG BUS
000002                  BB01    =       BIT01           ;TAG BUS
000001                  BB00    =       BIT00           ;TAG BUS

                        ;RMER2  ERROR REGISTER 2

100000                  BSE     =       BIT15           ;BAD SECTOR ERROR
040000                  SKI     =       BIT14           ;SEEK INCOMPLETE
020000                  OPE     =       BIT13           ;OPERATOR PLUG ERROR
010000                  IVC     =       BIT12           ;INVALID COMMAND ERROR
004000                  LSC     =       BIT11           ;LOSS OF SYSTEM CLOCK
002000                  LBC     =       BIT10           ;LOSS OF BIT CLOCK
000200                  DVC     =       BIT07           ;DEVICE CHECK
000010                  DPE     =       BIT03           ;DATA PARITY ERROR
001567                  XNUER2  =       001567          ;UNUSED BITS OF RMER2

                        .SBTTL  PROGRAM MNEMONICS

100000                  MSE     =       BIT15           ;MANUFACTURING DETECTED SECTOR ERROR
040000                  USE     =       BIT14           ;USER DETECTED SECTOR ERROR

                        .SBTTL  RM03 REGISTER INDEX VALUES

000000                  RMCS1   =       00              ;CONTROL STATUS REGISTER
000006                  RMDA    =       06              ;DISK ADDRESS REGISTER
000012                  RMDS    =       12              ;DRIVE STATUS REGISTER
000014                  RMER1   =       14              ;ERROR REGISTER 1
000016                  RMAS    =       16              ;ATTENTION SUMMARY REGISTER
000020                  RMLA    =       20              ;LOOK AHEAD REGISTER
000024                  RMMR1   =       24              ;MAINTENANCE REGISTER
000026                  RMDT    =       26              ;DRIVE TYPE REGISTER
000030                  RMSN    =       30              ;SERIAL NUMBER REGISTER
000032                  RMOF    =       32              ;OFFSET REGISTER
000034                  RMDC    =       34              ;DESIRED CYLINDER REGISTER
000036                  RMHR    =       36              ;HOLDING REGISTER
000040                  RMMR2   =       40              ;MAINTENANCE REGISTER 2
000042                  RMER2   =       42              ;ERROR REGISTER 2
000044                  RMEC1   =       44              ;ECC POSITION REGISTER
000046                  RMEC2   =       46              ;ECC PATTERN REGISTER
```

```
3023    000050                  ILRG50  =       50              ;ILLEGAL REGISTER 50
3024    000052                  ILRG52  =       52              ;ILLEGAL REGISTER 52
3025    000054                  ILRG54  =       54              ;ILLEGAL REGISTER 54
3026    000056                  ILRG56  =       56              ;ILLEGAL REGISTER 56
3027    000060                  ILRG60  =       60              ;ILLEGAL REGISTER 60
3028    000062                  ILRG62  =       62              ;ILLEGAL REGISTER 62
3029    000064                  ILRG64  =       64              ;ILLEGAL REGISTER 64
3030    000066                  ILRG66  =       66              ;ILLEGAL REGISTER 66
3031    000070                  ILRG70  =       70              ;ILLEGAL REGISTER 70
3032    000072                  ILRG72  =       72              ;ILLEGAL REGISTER 72
3033    000074                  ILRG74  =       74              ;ILLEGAL REGISTER 74
3034    000076                  ILRG76  =       76              ;ILLEGAL REGISTER 76
3035
3036
3037    000077                  IDXMSK  =       77              ;MASK FOR REGISTER INDEX NUMBER
3038
3039                            .SBTTL  RH CONTROLLER REGISTER BIT DEFINITIONS
3040
3041                            ;RMCS1  CONTROL STATUS REGISTER #1
3042
3043    100000                  SC      =       BIT15           ;SPECIAL CONDITION-READ ONLY
3044    040000                  TRE     =       BIT14           ;TRANSFER ERROR
3045    020000                  MCPE    =       BIT13           ;MASSBUS CONTROL BUS PARITY
3046                                                            ;ERROR-READ ONLY
3047    002000                  PSEL    =       BIT10           ;PORT B SELECT
3048    001000                  A17     =       BIT09           ;ADDRESS EXTENSION
3049    000400                  A16     =       BIT08           ;ADDRESS EXTENSION
3050    000200                  RDY     =       BIT07           ;READY-READ ONLY
3051    000100                  IE      =       BIT06           ;INTERRUPT ENABLE
3052
3053                            ;RMCS2  RH CONTROL STATUS REGISTER #2
3054
3055    100000                  DLT     =       BIT15           ;DATA LATE-READ ONLY
3056    040000                  WCE     =       BIT14           ;WRITE CHECK ERROR-READ ONLY
3057    020000                  UPE     =       BIT13           ;UNIBUS PARITY ERROR
3058    010000                  NED     =       BIT12           ;NONEXISTANT DRIVE-READ ONLY
3059    004000                  NEM     =       BIT11           ;NONEXISTANT MEMORY-READ ONLY
3060    002000                  PGE     =       BIT10           ;PROGRAM ERROR-READ ONLY
3061    001000                  MXF     =       BIT09           ;MISSED TRANSFER
3062    000400                  MDPE    =       BIT08           ;MASSBUS DATA BUS PARITY
3063                                                            ;ERROR-READ ONLY
3064    000200                  OR      =       BIT07           ;OUTPUT READY-READ ONLY
3065    000100                  IR      =       BIT06           ;INPUT READY-READ ONLY
3066    000040                  CLR     =       BIT05           ;CONTROLLER CLEAR
3067    000020                  PAT     =       BIT04           ;PARITY TEST
3068    000010                  BAI     =       BIT03           ;UNIBUS ADDRESS INCREMENT
3069                                                            ;INHIBIT
3070    000004                  U2      =       BIT02           ;UNIT SELECT
3071    000002                  U1      =       BIT01           ;UNIT SELECT
3072    000001                  U0      =       BIT00           ;UNIT SELECT
3073
3074                            ;UNIT SELECT MASK
3075    000007                  UNTMSK  =       7               ;UNIT SELECT MASK
3076
3077                            ;RMCS3  RH70 CONTROL STATUS REGISTER #3
3078    100000                  APE     =       BIT15           ;ADDRESS PARITY ERROR
```

```
3079            040000                  DPEHI   =       BIT14           ;DATA PARITY ERROR HIGH WORD
3080            020000                  DPELO   =       BIT13           ;DATA PARITY ERROR LOW WORD
3081            010000                  WCEHI   =       BIT12           ;WRITE CHECK ERROR HIGH WORD
3082            004000                  WCELO   =       BIT11           ;WRITE CHECK ERROR LOW WORD
3083            002000                  DBL     =       BIT10           ;DOUBLE WORD TRANSFER
3084            000100                  IE      =       BIT06           ;INTERRUPT ENABLE
3085            000010                  IPCK3   =       BIT03           ;INVERT PARITY CHECK
3086            000004                  IPCK2   =       BIT02           ;INVERT PARITY CHECK
3087            000002                  IPCK1   =       BIT01           ;INVERT PARITY CHECK
3088            000001                  IPCK0   =       BIT00           ;INVERT PARITY CHECK
3089                                    .SBTTL  RH CONTROLLER REGISTER INDEX VALUES
3090
3091            000000                  RMCS1   =       00              ;CONTROL, STATUS REGISTER
3092            000002                  RMWC    =       02              ;WORD COUNT REGISTER
3093            000004                  RMBA    =       04              ;BUS ADDRESS REGISTER
3094            000010                  RMCS2   =       10              ;CONTROLLER STATUS REGISTER
3095            000022                  RMDB    =       22              ;DATA BUFFER
3096            000050                  RMBAE   =       50              ;BUS ADDRESS EXTENSION
3097            000052                  RMCS3   =       52              ;CONTROL STATUS REGISTER #3
3098
3099            176700                  ABASE   =       176700          ;UNIBUS ADDRESS
3100            120254                  AVECT1  =       120254          ;UNIBUS VECTOR ADDRESS AND PRIORITY
3101
3102
3103                                    .SBTTL  TRAP CATCHER
3104
3105            000000                          .=0
3106                                    ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
3107                                    ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
3108                                    ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
3109            000174                          .=174
3110    000174  000000                  DISPREG: .WORD  0                ;;SOFTWARE DISPLAY REGISTER
3111    000176  000000                  SWREG:   .WORD  0                ;;SOFTWARE SWITCH REGISTER
3112
3113
3114                                    .SBTTL  ACT11 HOOKS
3115
3116                                    ;;*****************************************************************
3117                                    ;HOOKS REQUIRED BY ACT11
3118            000200                          $SVPC=.                  ;SAVE PC
3119            000046                          .=46
3120    000046  057324                  $ENDAD                           ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
3121            000052                          .=52
3122    000052  000000                  .WORD   0                        ;;2)SET LOC.52 TO ZERO
3123            000200                  .=$SVPC                          ;; RESTORE PC
3124
3125                                    .SBTTL  STARTING ADDRESS
3126
3127                                    ;THE PROGRAM STARTS AT LOCATION 200
3128            000200                  .       =       200
3129    000200  000137  004542          JMP     START                   ;JUMP TO START OF PROGRAM
3130
3131            001100                  .=1100
3132                                    .SBTTL  APT PARAMETER BLOCK
3133
3134                                    ;;*****************************************************************
```

```
3135                                    ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
3136                                    ;;**************************************************
3137            001100                          .$X=.   ;;SAVE CURRENT LOCATION
3138            000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
3139  000024    000200                          200     ;;FOR APT START UP
3140            000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
3141  000044    001100                          $APTHDR ;;POINT TO APT HEADER BLOCK
3142            001100                          .=.$X   ;;RESET LOCATION COUNTER
3143                                    ;;**************************************************************
3144                                    ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
3145                                    ;INTERFACE SPEC.
3146
3147  001100                            $APTHD:
3148  001100    000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
3149  001102    001222                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
3150  001104    000001                  $TSTM:  .WORD   1       ;;RUN TIM OF LONGEST TEST
3151  001106    000002                  $PASTM: .WORD   2       ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
3152  001110    000002                  $UNITM: .WORD   2       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
3153  001112    000042                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
3154            001114                  TAGADR  =       .
```

```
                                .SBTTL  COMMON TAGS

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

001114                                  .=TAGADR
001114                          $CMTAG:                         ;;START OF COMMON TAGS
001114  000000                          .WORD   0
001116  000                     $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001117  000                     $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001120  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001122  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001124  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001126  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001130  000                     $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001131  001                     $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001132  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001134  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001136  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001140  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001142  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001144  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
001146  000000                          .WORD   0
001150  000                     $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001151  000                     $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001152  000000                          .WORD   0
001154  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001156  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001160  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001162  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001164  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001166  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001170  000                     $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001171  002                     $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001172  012                     $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A ''LINE FEED''
001173  000                     $TPFLG: .BYTE   0               ;;''TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
001174  000000                  $TMP0:  .WORD   0               ;;USER DEFINED
001176  000000                  $TMP1:  .WORD   0               ;;USER DEFINED
001200  000000                  $TMP2:  .WORD   0               ;;USER DEFINED
001202  000000                  $TMP3:  .WORD   0               ;;USER DEFINED
001204  000000                  $TMP4:  .WORD   0               ;;USER DEFINED
001206  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001210  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001212  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001216  077                     $QUES:  .ASCII  /?/             ;;QUESTION MARK
001217  015                     $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001220  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;*****************************************************************
                                .NLIST ME
                                ;
                                ;
```

```
3211                                    .EVEN
3212  001222                            $MAIL:                          ;;APT MAILBOX
3213  001222  000000                    $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
3214  001224  000000                    $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
3215  001226  000000                    $TESTN: .WORD   ATESTN          ;;TEST NUMBER
3216  001230  000000                    $PASS:  .WORD   APASS           ;;PASS COUNT
3217  001232  000000                    $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
3218  001234  000000                    $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
3219  001236  000000                    $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
3220  001240  000000                    $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
3221  001242                            $ETABLE:                        ;;APT ENVIRONMENT TABLE
3222  001242     000                    $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
3223  001243     000                    $ENVM:  .BYTE   AENVM           ;;ENVIRONMENT MODE BITS
3224  001244  000000                    $SWREG: .WORD   ASWREG          ;;APT SWITCH REGISTER
3225  001246  000000                    $USWR:  .WORD   AUSWR           ;;USER SWITCHES
3226  001250  000000                    $CPUOP: .WORD   ACPUOP          ;;CPU TYPE,OPTIONS
3227                                    ;*                              BITS 15-11=CPU TYPE
3228                                    ;*                                   11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
3229                                    ;*                                   11/70=06,PDQ=07,Q=10
3230                                    ;*                              BIT 10=REAL TIME CLOCK
3231                                    ;*                              BIT  9=FLOATING POINT PROCESSOR
3232                                    ;*                              BIT  8=MEMORY MANAGEMENT
3233  001252     000                    $MAMS1: .BYTE   AMAMS1          ;;HIGH ADDRESS,M.S. BYTE
3234  001253     000                    $MTYP1: .BYTE   AMTYP1          ;;MEM. TYPE,BLK#1
3235                                    ;*                              MEM.TYPE BYTE   --  (HIGH BYTE)
3236                                    ;*                                   900 NSEC CORE=001
3237                                    ;*                                   300 NSEC BIPOLAR=002
3238                                    ;*                                   500 NSEC MOS=003
3239  001254  000000                    $MADR1: .WORD   AMADR1          ;;HIGH ADDRESS,BLK#1
3240                                    ;*                                MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF ''TYPE'' ABOVE
3241  001256     000                    $MAMS2: .BYTE   AMAMS2          ;;HIGH ADDRESS,M.S. BYTE
3242  001257     000                    $MTYP2: .BYTE   AMTYP2          ;;MEM.TYPE,BLK#2
3243  001260  000000                    $MADR2: .WORD   AMADR2          ;;MEM.LAST ADDRESS,BLK#2
3244  001262     000                    $MAMS3: .BYTE   AMAMS3          ;;HIGH ADDRESS,M.S.BYTE
3245  001263     000                    $MTYP3: .BYTE   AMTYP3          ;;MEM.TYPE,BLK#3
3246  001264  000000                    $MADR3: .WORD   AMADR3          ;;MEM.LAST ADDRESS,BLK#3
3247  001266     000                    $MAMS4: .BYTE   AMAMS4          ;;HIGH ADDRESS,M.S.BYTE
3248  001267     000                    $MTYP4: .BYTE   AMTYP4          ;;MEM.TYPE,BLK#4
3249  001270  000000                    $MADR4: .WORD   AMADR4          ;;MEM.LAST ADDRESS,BLK#4
3250  001272  120254                    $VECT1: .WORD   AVECT1          ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
3251  001274  000000                    $VECT2: .WORD   AVECT2          ;;INTERRUPT VECTOR#2BUS PRIORITY#2
3252  001276  176700                    $BASE:  .WORD   ABASE           ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
3253  001300  000000                    $DEVM:  .WORD   ADEVM           ;;DEVICE MAP
3254  001302  000000                    $CDW1:  .WORD   ACDW1           ;;CONTROLLER DESCRIPTION WORD#1
3255  001304  000000                    $CDW2:  .WORD   ACDW2           ;;CONTROLLER DESCRIPTION WORD#2
3256  001306  000000                    $DDW0:  .WORD   ADDW0           ;;DEVICE DESCRIPTOR WORD#0
3257  001310  000000                    $DDW1:  .WORD   ADDW1           ;;DEVICE DESCRIPTOR WORD#1
3258  001312  000000                    $DDW2:  .WORD   ADDW2           ;;DEVICE DESCRIPTOR WORD#2
3259  001314  000000                    $DDW3:  .WORD   ADDW3           ;;DEVICE DESCRIPTOR WORD#3
3260  001316  000000                    $DDW4:  .WORD   ADDW4           ;;DEVICE DESCRIPTOR WORD#4
3261  001320  000000                    $DDW5:  .WORD   ADDW5           ;;DEVICE DESCRIPTOR WORD#5
3262  001322  000000                    $DDW6:  .WORD   ADDW6           ;;DEVICE DESCRIPTOR WORD#6
3263  001324  000000                    $DDW7:  .WORD   ADDW7           ;;DEVICE DESCRIPTOR WORD#7
3264  001326                            $ETEND:
3265                                    .MEXIT
3266
```

```
3267
3268
3269                            .SBTTL  REGISTER INPUT BUFFER
3270
3271                            ;       THE REGISTER INPUT BUFFER IS USED FOR STORING REGISTER
3272                            ;CONTENTS AS THEY ARE READ FROM THE DEVICE.
3273
3274  001326  000000            RMCS1I: .WORD                   ;CONTROL,STATUS REGISTER #1
3275  001330  000000            RMWCI: .WORD                    ;WORD COUNT REGISTER
3276  001332  000000            RMBAI: .WORD                    ;BUS ADDRESS REGISTER
3277  001334  000000            RMDAI: .WORD                    ;DISK ADDRESS REGISTER
3278  001336  000000            RMCS2I: .WORD                   ;CONTROL, STATUS REGISTER #2
3279  001340  000000            RMDSI: .WORD                    ;DRIVE STATUS REGISTER
3280  001342  000000            RMER1I: .WORD                   ;ERROR REGISTER 1
3281  001344  000000            RMASI: .WORD                    ;ATTENTION SUMMARY REGISTER
3282  001346  000000            RMLAI: .WORD                    ;LOOK AHEAD REGISTER
3283  001350  000000            RMDBI: .WORD                    ;DATA BUFFER
3284  001352  000000            RMMR1I: .WORD                   ;MAINTENANCE REGISTER #1
3285  001354  000000            RMDTI: .WORD                    ;DRIVE TYPE REGISTER
3286  001356  000000            RMSNI: .WORD                    ;SERIAL NUMBER REGISTER
3287  001360  000000            RMOFI: .WORD                    ;OFFSET REGISTER
3288  001362  000000            RMDCI: .WORD                    ;DESIRED CYLINDER REGISTER
3289  001364  000000            RMHRI: .WORD                    ;HOLDING REGISTER
3290  001366  000000            RMMR2I: .WORD                   ;MAINTENANCE REGISTER #2
3291  001370  000000            RMER2I: .WORD                   ;ERROR REGISTER 2
3292  001372  000000            RMEC1I: .WORD                   ;ECC POSITION REGISTER
3293  001374  000000            RMEC2I: .WORD                   ;ECC PATTERN REGISTER
3294  001376  000000            RMBAEI: .WORD           ;BUS ADDRESS EXTENSION REGISTER
3295  001400  000000            RMCS3I: .WORD           ;CONTROL, STATUS REGISTER #3
3296
3297
3298
3299                            .SBTTL  REGISTER OUTPUT BUFFER
3300
3301                            ;       THE REGISTER OUTPUT BUFFER IS USED FOR ASSEMBLING DATA TO
3302                            ;BE WRITTEN TO THE DEVICE.
3303
3304  001402  000000            RMCS1O: .WORD                   ;CONTROL,STATUS REGISTER #1
3305  001404  000000            RMWCO: .WORD                    ;WORD COUNT REGISTER
3306  001406  000000            RMBAO: .WORD                    ;BUS ADDRESS REGISTER
3307  001410  000000            RMDAO: .WORD                    ;DISK ADDRESS REGISTER
3308  001412  000000            RMCS2O: .WORD                   ;CONTROL, STATUS REGISTER #2
3309  001414  000000            RMDSO: .WORD                    ;DRIVE STATUS REGISTER
3310  001416  000000            RMER1O: .WORD                   ;ERROR REGISTER 1
3311  001420  000000            RMASO: .WORD                    ;ATTENTION SUMMARY REGISTER
3312  001422  000000            RMLAO: .WORD                    ;LOOK AHEAD REGISTER
3313  001424  000000            RMDBO: .WORD                    ;DATA BUFFER
3314  001426  000000            RMMR1O: .WORD                   ;MAINTENANCE REGISTER #1
3315  001430  000000            RMDTO: .WORD                    ;DRIVE TYPE REGISTER
3316  001432  000000            RMSNO: .WORD                    ;SERIAL NUMBER REGISTER
3317  001434  000000            RMOFO: .WORD                    ;OFFSET REGISTER
3318  001436  000000            RMDCO: .WORD                    ;DESIRED CYLINDER REGISTER
3319  001440  000000            RMHRO: .WORD                    ;HOLDING REGISTER
3320  001442  000000            RMMR2O: .WORD                   ;MAINTENANCE REGISTER #2
3321  001444  000000            RMER2O: .WORD                   ;ERROR REGISTER 2
3322  001446  000000            RMEC1O: .WORD                   ;ECC POSITION REGISTER
```

```
3323  001450  000000                  RMEC20: .WORD                   ;ECC PATTERN REGISTER
3324  001452  000000                  RMBAE0: .WORD             ;BUS ADDRESS EXTENSION REGISTER
3325  001454  000000                  RMCS30: .WORD             ;CONTROL, STATUS REGISTER #3
3326
3327                                  .SBTTL  TEST QUE
3328
3329                                  ;       EACH WORD OF THE TEST QUE CONTAINS THE DEVICE NUMBER IN
3330                                  ;THE LOW BYTE AND THE ATTENTION BIT IN THE HIGH BYTE. A ZERO
3331                                  ;WORD IS A BLANK AND REPRESENTS THE END OF THE QUE.
3332
3333  001456  000012                  TSTQUE: .BLKW   10.     ;TEST QUE
3334
3335  001502  172540                  $LPCSR: .WORD   172540          ;KW11-P CONTROL + STATUS REGISTER
3336  001504  172542                  $LPCSB: .WORD   172542          ;KW11-P COUNT SET BUFFER
3337  001506  000104                  $LPVEC: .WORD   104             ;KW11-P INTERRUPT VECTOR
3338  001510  000106                          .WORD   106
3339  001512  177546                  $LLCSR: .WORD   177546          ;KW11-L CONTROL + STATUS REGISTER
3340  001514  000100                  $LLVEC: .WORD   100             ;KW11-L INTERRUPT VECTOR
3341  001516  000102                          .WORD   102
3342  001520  000000                  $PSW:   .WORD                   ;STORAGE FOR PRIORITY
3343  001522  000000                  TIME:   .WORD                   ;STORAGE FOR ELAPSED TIME
3344  001524  000000                  WATCH:  .WORD                   ;STORAGE FOR REMAINING TIME
3345  001526  000000                  CLOCK:  .WORD                   ;ADDRESS OF START CLOCK SUB
3346  001530  000000                  STOP:   .WORD                   ;ADDRESS OF STOP CLOCK SUB
3347
3348                                  ;PUT TAGS HERE
3349
```

```
                                .SBTTL  ERROR POINTER TABLE

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

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


001532                          $ERRTB:
```

```
3367
3368                                    ;ERROR  1       CANNOT CLEAR NED STATUS
3369
3370  001532  066600                            EMT1
3371  001534  074456                            EHT1
3372  001536  074552                            EDT1
3373  001540  074600                            EFT1
3374
3375
3376                                    ;ERROR  2       CANNOT READ OR WRITE ANY DEVICE REG WITHOUT NED
3377
3378  001542  066606                            EMT2
3379  001544  074462                            EHT2
3380  001546  074554                            EDT2
3381  001550  074602                            EFT2
3382
3383
3384                                    ;ERROR  3       CANNOT WRITE/READ ONES TO ANY DEVICE REGISTER
3385
3386  001552  066634                            EMT3
3387  001554  000000                            0
3388  001556  000000                            0
3389  001560  000000                            0
3390
3391
3392                                    ;ERROR  4       CANNOT CLEAR ANY DEVICE REGISTER BITS W/MASSBUS INIT
3393
3394  001562  066654                            EMT4
3395  001564  000000                            0
3396  001566  000000                            0
3397  001570  000000                            0
3398
3399
3400                                    ;ERROR  5       CANNOT WRITE/READ ZEROS TO ALL BIT POSITIONS
3401
3402  001572  066676                            EMT5
3403  001574  074466                            EHT5
3404  001576  074556                            EDT5
3405  001600  074604                            EFT5
3406
3407
3408                                    ;ERROR  6       CANNOT WRITE/READ ONES TO ALL BIT POSITIONS
3409
3410  001602  066722                            EMT6
3411  001604  074466                            EHT5
3412  001606  074556                            EDT5
3413  001610  074604                            EFT5
3414
3415
3416                                    ;ERROR  7       CANNOT WRITE/READ SHIFTING ONE BIT TO ALL BIT POSITIONS
3417                                    ;               OF DEVICE REGISTERS
3418
3419  001612  066744                            EMT7
3420  001614  074472                            EHT7
3421  001616  074556                            EDT5
3422  001620  074604                            EFT5
```

```
3423
3424
3425                                    ;ERROR  10      REGISTER SELECT 1 APPEARS S-A-0
3426
3427  001622  066766                            EMT10
3428  001624  000000                            0
3429  001626  000000                            0
3430  001630  000000                            0
3431
3432
3433                                    ;ERROR  11      REGISTER SELECT 1 APPEARS S-A-1
3434
3435  001632  067004                            EMT11
3436  001634  000000                            0
3437  001636  000000                            0
3438  001640  000000                            0
3439
3440
3441                                    ;ERROR  12      REGISTER SELECT 2 APPEARS S-A-0
3442
3443  001642  067022                            EMT12
3444  001644  000000                            0
3445  001646  000000                            0
3446  001650  000000                            0
3447
3448
3449                                    ;ERROR  13      REGISTER SELECT 2 APPEARS S-A-1
3450
3451  001652  067040                            EMT13
3452  001654  000000                            0
3453  001656  000000                            0
3454  001660  000000                            0
3455
3456
3457                                    ;ERROR  14      REGISTER SELECT 4 APPEARS S-A-0
3458
3459  001662  067056                            EMT14
3460  001664  000000                            0
3461  001666  000000                            0
3462  001670  000000                            0
3463
3464
3465                                    ;ERROR  15      REGISTER SELECT 4 APPEARS S-A-1
3466
3467  001672  067074                            EMT15
3468  001674  000000                            0
3469  001676  000000                            0
3470  001700  000000                            0
3471
3472
3473                                    ;ERROR  16      REGISTER SELECT 8 APPEARS S-A-0
3474
3475  001702  067112                            EMT16
3476  001704  000000                            0
3477  001706  000000                            0
3478  001710  000000                            0
```

```
3479
3480
3481                                    ;ERROR  17      REGISTER SELECT 8 APPEARS S-A-1
3482
3483  001712  067130                            EMT17
3484  001714  000000                            0
3485  001716  000000                            0
3486  001720  000000                            0
3487
3488
3489                                    ;ERROR  20      CANT WRITE ZEROS RMDA
3490
3491  001722  067146                            EMT20
3492  001724  074456                            EHT1
3493  001726  074552                            EDT1
3494  001730  074600                            EFT1
3495
3496
3497                                    ;ERROR  21      CANT WRITE ONES RMDA
3498
3499  001732  067166                            EMT21
3500  001734  074456                            EHT1
3501  001736  074552                            EDT1
3502  001740  074600                            EFT1
3503
3504
3505                                    ;ERROR  22      BIT INTERFERENCE IN WRITING/READING RMDA
3506
3507  001742  067206                            EMT22
3508  001744  074456                            EHT1
3509  001746  074552                            EDT1
3510  001750  074600                            EFT1
3511
3512
3513                                    ;ERROR  23      CANT WRITE ZEROS RMCS1
3514
3515  001752  067222                            EMT23
3516  001754  074456                            EHT1
3517  001756  074552                            EDT1
3518  001760  074600                            EFT1
3519
3520
3521                                    ;ERROR  24      CANT WRITE ONES RMCS1
3522
3523  001762  067242                            EMT24
3524  001764  074456                            EHT1
3525  001766  074552                            EDT1
3526  001770  074600                            EFT1
3527
3528
3529                                    ;ERROR  25      BIT INTERFERENCE IN WRITING/READING RMCS1
3530
3531  001772  067262                            EMT25
3532  001774  074456                            EHT1
3533  001776  074552                            EDT1
3534  002000  074600                            EFT1
```

```
3535
3536
3537                                    ;ERROR  26      MBA CLR L IS STUCK ACTIVE
3538
3539  002002  067276                            EMT26
3540  002004  000000                            0
3541  002006  000000                            0
3542  002010  000000                            0
3543
3544
3545                                    ;ERROR  27      CANNOT CLEAR RMER1-PAR,RMR,ILF,ILR
3546
3547  002012  067330                            EMT27
3548  002014  074456                            EHT1
3549  002016  074552                            EDT1
3550  002020  074600                            EFT1
3551
3552
3553                                    ;ERROR  30      CANNOT CLEAR RMER1-DCK,IAE,AOE,HCRC,HCE,ECH,WCF,FER
3554
3555  002022  067342                            EMT30
3556  002024  074456                            EHT1
3557  002026  074552                            EDT1
3558  002030  074600                            EFT1
3559
3560
3561                                    ;ERROR  31      CANNOT CLEAR RMER1-OPI,DTE
3562
3563  002032  067356                            EMT31
3564  002034  074456                            EHT1
3565  002036  074552                            EDT1
3566  002040  074600                            EFT1
3567
3568
3569                                    ;ERROR  32      CANNOT WRITE 0 IN RMER1-PAR,RMR,ILF,ILR
3570
3571  002042  067372                            EMT32
3572  002044  074456                            EHT1
3573  002046  074552                            EDT1
3574  002050  074600                            EFT1
3575
3576
3577                                    ;ERROR  33      CANNOT WRITE 0 IN RMER1-DCK,IAE,AOE,HCRC,HCE,ECH,WCF,FER
3578
3579  002052  067406                            EMT33
3580  002054  074456                            EHT1
3581  002056  074552                            EDT1
3582  002060  074600                            EFT1
3583
3584
3585                                    ;ERROR  34      CANNOT WRITE 0 IN RMER1-OPI,DTE
3586
3587  002062  067424                            EMT34
3588  002064  074456                            EHT1
3589  002066  074552                            EDT1
3590  002070  074600                            EFT1
```

```
3591
3592
3593                                    ;ERROR  35      CANNOT WRITE 1 IN RMER1
3594
3595  002072  067442                            EMT35
3596  002074  074456                            EHT1
3597  002076  074552                            EDT1
3598  002100  074600                            EFT1
3599
3600
3601                                    ;ERROR  36      CANNOT WRITE SHIFTING 1 IN RMER1
3602
3603  002102  067456                            EMT36
3604  002104  074456                            EHT1
3605  002106  074552                            EDT1
3606  002110  074600                            EFT1
3607
3608
3609                                    ;ERROR  37      CANNOT WRITE ZEROS IN RMDC
3610
3611  002112  067472                            EMT37
3612  002114  074456                            EHT1
3613  002116  074552                            EDT1
3614  002120  074600                            EFT1
3615
3616
3617                                    ;ERROR  40      CANNOT WRITE ONES IN RMDC
3618
3619  002122  067506                            EMT40
3620  002124  074456                            EHT1
3621  002126  074552                            EDT1
3622  002130  074600                            EFT1
3623
3624
3625                                    ;ERROR  41      BIT INTERFERENCE IN WRITING/READING RMDC
3626
3627  002132  067526                            EMT41
3628  002134  074456                            EHT1
3629  002136  074552                            EDT1
3630  002140  074600                            EFT1
3631
3632
3633                                    ;ERROR  42      CANNOT WRITE 1'S IN RMDC OR RMDA
3634
3635  002142  067542                            EMT42
3636  002144  000000                            0
3637  002146  000000                            0
3638  002150  000000                            0
3639
3640
3641                                    ;ERROR  43      CANNOT CLEAR RMCS1-FUNCTION CODE
3642
3643  002152  067566                            EMT43
3644  002154  074456                            EHT1
3645  002156  074552                            EDT1
3646  002160  074600                            EFT1
```

```
3647
3648
3649                                    ;ERROR  44      UNUSED BITS OF RMER2 NOT ZERO
3650
3651  002162  067600                              EMT44
3652  002164  074456                              EHT1
3653  002166  074552                              EDT1
3654  002170  074600                              EFT1
3655
3656
3657                                    ;ERROR  45      CANNOT CLEAR RMER2-OPE,IVC,LSC
3658
3659  002172  067614                              EMT45
3660  002174  074456                              EHT1
3661  002176  074552                              EDT1
3662  002200  074600                              EFT1
3663
3664
3665                                    ;ERROR  46      CANNOT CLEAR RMER2-LBC,DPE
3666
3667  002202  067630                              EMT46
3668  002204  074456                              EHT1
3669  002206  074552                              EDT1
3670  002210  074600                              EFT1
3671
3672
3673                                    ;ERROR  47      CANNOT WRITE ZEROS RMER2-OPE,IVC,LSC
3674
3675  002212  067644                              EMT47
3676  002214  074456                              EHT1
3677  002216  074552                              EDT1
3678  002220  074600                              EFT1
3679
3680
3681                                    ;ERROR  50      CANNOT WRITE ZEROS RMER2-LBC,DPE
3682
3683  002222  067666                              EMT50
3684  002224  074456                              EHT1
3685  002226  074552                              EDT1
3686  002230  074600                              EFT1
3687
3688
3689                                    ;ERROR  51      CANNOT WRITE ONES RMER2
3690
3691  002232  067710                              EMT51
3692  002234  074456                              EHT1
3693  002236  074552                              EDT1
3694  002240  074600                              EFT1
3695
3696
3697                                    ;ERROR  52      CANNOT WRITE SHIFTING ONES RMER2
3698
3699  002242  067730                              EMT52
3700  002244  074456                              EHT1
3701  002246  074552                              EDT1
3702  002250  074600                              EFT1
```

```
3703
3704
3705                                    ;ERROR  53      UNUSED BITS OF RMOF ARE NOT ZERO
3706
3707  002252  067744                            EMT53
3708  002254  074456                            EHT1
3709  002256  074552                            EDT1
3710  002260  074600                            EFT1
3711
3712
3713                                    ;ERROR  54      CANNOT WRITE ZEROS RMOF-FMT,ECI,HCI,OFD
3714
3715  002262  067760                            EMT54
3716  002264  074456                            EHT1
3717  002266  074552                            EDT1
3718  002270  074600                            EFT1
3719
3720
3721                                    ;ERROR  55      CANNOT WRITE ONES RMOF-FMT,ECI,HCI,OFD
3722
3723  002272  070000                            EMT55
3724  002274  074456                            EHT1
3725  002276  074552                            EDT1
3726  002300  074600                            EFT1
3727
3728
3729                                    ;ERROR  56      CANNOT WRITE SHIFTING ONES RMOF
3730
3731  002302  070020                            EMT56
3732  002304  074456                            EHT1
3733  002306  074552                            EDT1
3734  002310  074600                            EFT1
3735
3736
3737                                    ;ERROR  57      DEVICE IS NOT AN RM03
3738
3739  002312  070034                            EMT57
3740  002314  074476                            EHT57
3741  002316  074560                            EDT57
3742  002320  074606                            EFT57
3743
3744
3745                                    ;ERROR  60      DEVICE AVAILABLE IS NOT SET
3746
3747  002322  070050                            EMT60
3748  002324  074456                            EHT1
3749  002326  074552                            EDT1
3750  002330  074600                            EFT1
3751
3752
3753                                    ;ERROR  61      CANNOT WRITE ZEROS RMHR
3754
3755  002332  070064                            EMT61
3756  002334  074456                            EHT1
3757  002336  074552                            EDT1
3758  002340  074600                            EFT1
```

```
3759
3760
3761                                    ;ERROR  62      CANNOT WRITE ONES RMHR
3762
3763  002342  070104                            EMT62
3764  002344  074456                            EHT1
3765  002346  074552                            EDT1
3766  002350  074600                            EFT1
3767
3768
3769                                    ;ERROR  63      CANNOT WRITE SHIFTING ONES RMHR
3770
3771  002352  070124                            EMT63
3772  002354  074456                            EHT1
3773  002356  074552                            EDT1
3774  002360  074600                            EFT1
3775
3776
3777                                    ;ERROR  64      CANNOT CLEAR ILR STATUS
3778
3779  002362  070140                            EMT64
3780  002364  074456                            EHT1
3781  002366  074552                            EDT1
3782  002370  074600                            EFT1
3783
3784
3785                                    ;ERROR  65      ILR ERROR SHOULD NOT BE SET
3786
3787  002372  070152                            EMT65
3788  002374  074502                            EHT65
3789  002376  074562                            EDT65
3790  002400  074610                            EFT65
3791
3792
3793                                    ;ERROR  66      ILR ERROR SHOULD BE SET
3794
3795  002402  070166                            EMT66
3796  002404  074502                            EHT65
3797  002406  074562                            EDT65
3798  002410  074610                            EFT65
3799
3800
3801                                    ;ERROR  67      CANNOT CLEAR PAR STATUS-DPE IS RESET
3802
3803  002412  070202                            EMT67
3804  002414  074456                            EHT1
3805  002416  074552                            EDT1
3806  002420  074600                            EFT1
3807
3808
3809                                    ;ERROR  70      CANNOT CLEAR PAR AND DPE STATUS
3810
3811  002422  070220                            EMT70
3812  002424  074456                            EHT1
3813  002426  074552                            EDT1
3814  002430  074600                            EFT1
```

```
3815
3816
3817                                    ;ERROR  71      'PAR'' ERROR SHOULD NOT BE SET-'PAT'' IS OFF
3818
3819  002432  070240                            EMT71
3820  002434  074506                            EHT71
3821  002436  074564                            EDT71
3822  002440  074612                            EFT71
3823
3824
3825                                    ;ERROR  72      'PAR'' ERROR SHOULD BE SET-'PAT'' IS ON
3826
3827  002442  070264                            EMT72
3828  002444  074506                            EHT71
3829  002446  074564                            EDT71
3830  002450  074612                            EFT71
3831
3832
3833                                    ;ERROR  73      'MCPE'' ERROR SHOULD NOT BE SET
3834
3835  002452  070310                            EMT73
3836  002454  074506                            EHT71
3837  002456  074564                            EDT71
3838  002460  074612                            EFT71
3839
3840
3841                                    ;ERROR  74      UNEXPECTED BUS TIMEOUT
3842
3843  002462  070330                            EMT74
3844  002464  074512                            EHT74
3845  002466  074566                            EDT74
3846  002470  074614                            EFT74
3847
3848
3849                                    ;ERROR  75      CANT CLEAR 'DMD''
3850
3851  002472  070340                            EMT75
3852  002474  074456                            EHT1
3853  002476  074552                            EDT1
3854  002500  074600                            EFT1
3855
3856
3857                                    ;ERROR  76      CANT WRITE ZERO 'DMD''
3858
3859  002502  070352                            EMT76
3860  002504  074456                            EHT1
3861  002506  074552                            EDT1
3862  002510  074600                            EFT1
3863
3864
3865                                    ;ERROR  77      CANT WRITE ONE 'DMD''
3866
3867  002512  070366                            EMT77
3868  002514  074456                            EHT1
3869  002516  074552                            EDT1
3870  002520  074600                            EFT1
```

```
3871
3872
3873                                    ;ERROR  100     DMD SET BY WRONG BIT
3874
3875  002522  070402                            EMT100
3876  002524  074506                            EHT71
3877  002526  074562                            EDT65
3878  002530  074610                            EFT65
3879
3880
3881                                    ;ERROR  101     CANT CLEAR 'MOL'' IN DIAGNOSTIC MODE
3882
3883  002532  070422                            EMT101
3884  002534  074456                            EHT1
3885  002536  074552                            EDT1
3886  002540  074600                            EFT1
3887
3888
3889                                    ;ERROR  102     CANT SET 'MOL'' IN DIAGNOSTIC MODE
3890
3891  002542  070442                            EMT102
3892  002544  074456                            EHT1
3893  002546  074552                            EDT1
3894  002550  074600                            EFT1
3895
3896
3897                                    ;ERROR  103     'MUR'' SET BY WRONG BIT
3898
3899  002552  070462                            EMT103
3900  002554  074506                            EHT71
3901  002556  074562                            EDT65
3902  002560  074610                            EFT65
3903
3904
3905                                    ;ERROR  104     CANT RESET 'WRL'' IN DIAGNOSTIC MODE
3906
3907  002562  070506                            EMT104
3908  002564  074456                            EHT1
3909  002566  074552                            EDT1
3910  002570  074600                            EFT1
3911
3912
3913                                    ;ERROR  105     CANT SET 'WRL'' IN DIAGNOSTIC MODE
3914
3915  002572  070526                            EMT105
3916  002574  074456                            EHT1
3917  002576  074552                            EDT1
3918  002600  074600                            EFT1
3919
3920
3921                                    ;ERROR  106     'MWP'' SET BY WRONG BIT
3922
3923  002602  070546                            EMT106
3924  002604  074506                            EHT71
3925  002606  074562                            EDT65
3926  002610  074610                            EFT65
```

```
3927
3928
3929                                    ;ERROR  107     CANT RESET 'DVC'' USING 'MDVC''
3930
3931  002612  070572                              EMT107
3932  002614  074456                              EHT1
3933  002616  074552                              EDT1
3934  002620  074600                              EFT1
3935
3936
3937                                    ;ERROR  110     'DVC'' IS RESET BUT 'UNS'' IS SET
3938
3939  002622  070612                              EMT110
3940  002624  074456                              EHT1
3941  002626  074552                              EDT1
3942  002630  074600                              EFT1
3943
3944
3945                                    ;ERROR  111     'DVC'' IS SET BUT 'UNS'' IS NOT SET
3946
3947  002632  070634                              EMT111
3948  002634  074456                              EHT1
3949  002636  074552                              EDT1
3950  002640  074600                              EFT1
3951
3952
3953                                    ;ERROR  112     CANT SET 'DVC'' USING MDVC''
3954
3955  002642  070654                              EMT112
3956  002644  074456                              EHT1
3957  002646  074552                              EDT1
3958  002650  074600                              EFT1
3959
3960
3961                                    ;ERROR  113     'DVC'' IS RESET BUT 'UNS'' IS SET
3962
3963  002652  070674                              EMT113
3964  002654  074456                              EHT1
3965  002656  074552                              EDT1
3966  002660  074600                              EFT1
3967
3968
3969                                    ;ERROR  114     'DVC'' IS SET BUT 'UNS'' IS NOT SET
3970
3971  002662  070720                              EMT114
3972  002664  074456                              EHT1
3973  002666  074552                              EDT1
3974  002670  074600                              EFT1
3975
3976
3977                                    ;ERROR  115     'MDF'' IS SET BY WRONG BIT
3978
3979  002672  070742                              EMT115
3980  002674  074516                              EHT115
3981  002676  074570                              EDT115
3982  002700  074616                              EFT115
```

```
3983
3984
3985                                    ;ERROR  116     CANT RESET ''SKI'' USING 'MSER''
3986
3987  002702  070766                            EMT116
3988  002704  074456                            EHT1
3989  002706  074552                            EDT1
3990  002710  074600                            EFT1
3991
3992
3993                                    ;ERROR  117     CANT SET ''SKI'' USING 'MSER''
3994
3995  002712  071006                            EMT117
3996  002714  074456                            EHT1
3997  002716  074552                            EDT1
3998  002720  074600                            EFT1
3999
4000
4001                                    ;ERROR  120     ''SKI'' SET BY WRONG BIT
4002
4003  002722  071026                            EMT120
4004  002724  074516                            EHT115
4005  002726  074570                            EDT115
4006  002730  074616                            EFT115
4007
4008
4009                                    ;ERROR  121     CANT RESET ''PIP'' USING 'MOC''
4010
4011  002732  071052                            EMT121
4012  002734  074456                            EHT1
4013  002736  074552                            EDT1
4014  002740  074600                            EFT1
4015
4016
4017                                    ;ERROR  122     CANT SET ''PIP'' USING 'MOC''
4018
4019  002742  071072                            EMT122
4020  002744  074456                            EHT1
4021  002746  074552                            EDT1
4022  002750  074600                            EFT1
4023
4024
4025                                    ;ERROR  123     ''MOC'' SET BY WRONG BIT
4026
4027  002752  071112                            EMT123
4028  002754  074516                            EHT115
4029  002756  074570                            EDT115
4030  002760  074616                            EFT115
4031
4032
4033                                    ;ERROR  124     CANT CLEAR ''EBL''
4034
4035  002762  071136                            EMT124
4036  002764  074456                            EHT1
4037  002766  074552                            EDT1
4038  002770  074600                            EFT1
```

```
4039
4040
4041                                    ;ERROR  125     'EBL'' NOT ZERO IN DIAGNOSTIC MODE
4042
4043  002772  071154                            EMT125
4044  002774  074456                            EHT1
4045  002776  074552                            EDT1
4046  003000  074600                            EFT1
4047
4048
4049                                    ;ERROR  126     CANT SET 'EBL'' USING 'DEBL''
4050
4051  003002  071174                            EMT126
4052  003004  074456                            EHT1
4053  003006  074552                            EDT1
4054  003010  074600                            EFT1
4055
4056
4057                                    ;ERROR  127     'DEBL'' SET BY WRONG BIT
4058
4059  003012  071212                            EMT127
4060  003014  074516                            EHT115
4061  003016  074570                            EDT115
4062  003020  074616                            EFT115
4063
4064
4065                                    ;ERROR  130     'LS'' NOT CORRECT ACCORDING TO RMDA
4066
4067  003022  071236                            EMT130
4068  003024  074522                            EHT130
4069  003026  074572                            EDT130
4070  003030  074620                            EFT130
4071
4072
4073                                    ;ERROR  131     'LST'' NOT CORRECT ACCORDING TO RMDA
4074
4075  003032  071254                            EMT131
4076  003034  074522                            EHT130
4077  003036  074572                            EDT130
4078  003040  074620                            EFT130
4079
4080
4081                                    ;ERROR  132     CANNOT INCREMENT SECTOR ADDRESS USING 'DEBL''
4082
4083  003042  071272                            EMT132
4084  003044  074526                            EHT132
4085  003046  074574                            EDT132
4086  003050  074622                            EFT132
4087
4088
4089                                    ;ERROR  133     CANNOT INCREMENT TRACK ADDRESS USING 'DEBL''
4090
4091  003052  071312                            EMT133
4092  003054  074526                            EHT132
4093  003056  074574                            EDT132
4094  003060  074622                            EFT132
```

```
 4095
 4096
 4097                                   ;ERROR  134     UNUSED BITS OF RMDC NOT ZERO
 4098
 4099  003062  071332                           EMT134
 4100  003064  074456                           EHT1
 4101  003066  074552                           EDT1
 4102  003070  074600                           EFT1
 4103
 4104
 4105                                   ;ERROR  135     'VV' NOT RESET BY UNIT READY
 4106
 4107  003072  071346                           EMT135
 4108  003074  074456                           EHT1
 4109  003076  074552                           EDT1
 4110  003100  074600                           EFT1
 4111
 4112
 4113                                   ;ERROR  136     SERIAL NUMBER IS INCONSISTENT
 4114
 4115  003102  071364                           EMT136
 4116  003104  074456                           EHT1
 4117  003106  074552                           EDT1
 4118  003110  074600                           EFT1
 4119
 4120
 4121                                   ;ERROR  137     CANT CLEAR 'GO' BIT
 4122
 4123  003112  071376                           EMT137
 4124  003114  074456                           EHT1
 4125  003116  074552                           EDT1
 4126  003120  074600                           EFT1
 4127
 4128
 4129                                   ;ERROR  140     CANT INCREMENT CYLINDER USING 'DEBL'
 4130
 4131  003122  071414                           EMT140
 4132  003124  074526                           EHT132
 4133  003126  074574                           EDT132
 4134  003130  074622                           EFT132
 4135
 4136
 4137                                   ;ERROR  141     CANT RESET 'LBT' BY WRITING RMDA
 4138
 4139  003132  071434                           EMT141
 4140  003134  074456                           EHT1
 4141  003136  074552                           EDT1
 4142  003140  074600                           EFT1
 4143
 4144
 4145                                   ;ERROR  142     CANT SET 'LBT' USING 'DEBL'
 4146
 4147  003142  071450                           EMT142
 4148  003144  074456                           EHT1
 4149  003146  074552                           EDT1
 4150  003150  074600                           EFT1
```

```
4151
4152
4153                                    ;ERROR  143     CANT READ ZERO FROM COMP ERROR
4154
4155  003152  071466                            EMT143
4156  003154  074456                            EHT1
4157  003156  074552                            EDT1
4158  003160  074600                            EFT1
4159
4160
4161                                    ;ERROR  144     CANT SET COMP ERROR WITH RMER1 OR RMER2
4162
4163  003162  071502                            EMT144
4164  003164  074456                            EHT1
4165  003166  074552                            EDT1
4166  003170  074600                            EFT1
4167
4168
4169                                    ;ERROR  145     COMP ERROR DID NOT SET
4170
4171  003172  071524                            EMT145
4172  003174  074532                            EHT145
4173  003176  074572                            EDT130
4174  003200  074620                            EFT130
4175
4176
4177                                    ;ERROR  146     CANT SET ''GO'' BIT
4178
4179  003202  071546                            EMT146
4180  003204  074456                            EHT1
4181  003206  074552                            EDT1
4182  003210  074600                            EFT1
4183
4184
4185                                    ;ERROR  147     CANT READ A ONE FROM ''TST''
4186
4187  003212  071564                            EMT147
4188  003214  074456                            EHT1
4189  003216  074552                            EDT1
4190  003220  074600                            EFT1
4191
4192
4193                                    ;ERROR  150     ''TST'' IS INCORRECT FOR THE FUNCTION CODE
4194
4195  003222  071576                            EMT150
4196  003224  074536                            EHT150
4197  003226  074570                            EDT115
4198  003230  074616                            EFT115
4199
4200
4201                                    ;ERROR  151     CANT SET THE ''GO'' BIT
4202
4203  003232  071620                            EMT151
4204  003234  074456                            EHT1
4205  003236  074552                            EDT1
4206  003240  074600                            EFT1
```

```
4207
4208
4209                                    ;ERROR  152     'DRY'' NOT THE COMPLEMENT OF ''GO''
4210
4211  003242  071632                            EMT152
4212  003244  074456                            EHT1
4213  003246  074552                            EDT1
4214  003250  074600                            EFT1
4215
4216
4217                                    ;ERROR  153     ''GO'' RESET EARLY
4218
4219  003252  071646                            EMT153
4220  003254  074456                            EHT1
4221  003256  074552                            EDT1
4222  003260  074600                            EFT1
4223
4224
4225                                    ;ERROR  154     ''GO'' DIDNT RESET ON TIME
4226
4227  003262  071666                            EMT154
4228  003264  074456                            EHT1
4229  003266  074552                            EDT1
4230  003270  074600                            EFT1
4231
4232
4233                                    ;ERROR  155     CANT CLEAR CONTINUE
4234
4235  003272  071706                            EMT155
4236  003274  074456                            EHT1
4237  003276  074552                            EDT1
4238  003300  074600                            EFT1
4239
4240
4241                                    ;ERROR  156     CONTINUE IS INCORRECT FOR THE FUNCTION CODE
4242
4243  003302  071724                            EMT156
4244  003304  074536                            EHT150
4245  003306  074570                            EDT115
4246  003310  074616                            EFT115
4247
4248
4249                                    ;ERROR  157     CANT CLEAR IVC
4250
4251  003312  071746                            EMT157
4252  003314  074456                            EHT1
4253  003316  074552                            EDT1
4254  003320  074600                            EFT1
4255
4256
4257                                    ;ERROR  160     IVC IS INCORRECT FOR THE FUNCTION CODE
4258
4259  003322  071764                            EMT160
4260  003324  074536                            EHT150
4261  003326  074570                            EDT115
4262  003330  074616                            EFT115
```

```
4263
4264
4265                                    ;ERROR  161     CANT CLEAR LSC
4266
4267  003332  072014                            EMT161
4268  003334  074456                            EHT1
4269  003336  074552                            EDT1
4270  003340  074600                            EFT1
4271
4272
4273                                    ;ERROR  162     CANT SET LSC
4274
4275  003342  072032                            EMT162
4276  003344  074456                            EHT1
4277  003346  074552                            EDT1
4278  003350  074600                            EFT1
4279
4280
4281                                    ;ERROR  163     COMMAND DECODE WAS ENABLED WITH COMP ERROR SET
4282
4283  003352  072050                            EMT163
4284  003354  074456                            EHT1
4285  003356  074552                            EDT1
4286  003360  074600                            EFT1
4287
4288
4289                                    ;ERROR  164     COMMAND DECODE WAS ENABLED WITH COMP ERROR SET
4290
4291  003362  072072                            EMT164
4292  003364  074456                            EHT1
4293  003366  074552                            EDT1
4294  003370  074600                            EFT1
4295
4296
4297                                    ;ERROR  165     DECODE DOES NOT SET
4298
4299  003372  072114                            EMT165
4300  003374  000000                            0
4301  003376  000000                            0
4302  003400  000000                            0
4303
4304
4305                                    ;ERROR  166     CANT CLEAR OCCUPIED
4306
4307  003402  072140                            EMT166
4308  003404  074456                            EHT1
4309  003406  074552                            EDT1
4310  003410  074600                            EFT1
4311
4312
4313                                    ;ERROR  167     ILF SET WITHOUT GO BIT
4314
4315  003412  072160                            EMT167
4316  003414  074456                            EHT1
4317  003416  074552                            EDT1
4318  003420  074600                            EFT1
```

```
4319
4320
4321                                    ;ERROR  170     CANT SET VOLUME VALID
4322
4323  003422  072202                            EMT170
4324  003424  074536                            EHT150
4325  003426  074570                            EDT115
4326  003430  074616                            EFT115
4327
4328
4329                                    ;ERROR  171     ILF IS INCORRECT
4330
4331  003432  072214                            EMT171
4332  003434  074536                            EHT150
4333  003436  074570                            EDT115
4334  003440  074616                            EFT115
4335
4336
4337                                    ;ERROR  172     CANT SET OFFSET DIRECTION BIT
4338
4339  003442  072230                            EMT172
4340  003444  074456                            EHT1
4341  003446  074552                            EDT1
4342  003450  074600                            EFT1
4343
4344
4345                                    ;ERROR  173     OCCUPIED IS INCORRECT FOR FUNCTION CODE
4346
4347  003452  072246                            EMT173
4348  003454  074536                            EHT150
4349  003456  074570                            EDT115
4350  003460  074616                            EFT115
4351
4352
4353                                    ;ERROR  174     READ IN PRESET DIDNT CLEAR RMDA, RMDC OR RMOF
4354
4355  003462  072270                            EMT174
4356  003464  000000                            0
4357  003466  000000                            0
4358  003470  000000                            0
4359
4360
4361                                    ;ERROR  175     READ IN PRESET DIDNT CLEAR RMOF
4362
4363  003472  072316                            EMT175
4364  003474  074456                            EHT1
4365  003476  074552                            EDT1
4366  003500  074600                            EFT1
4367
4368
4369                                    ;ERROR  176     READ IN PRESET DIDNT CLEAR RMDA
4370
4371  003502  072334                            EMT176
4372  003504  074456                            EHT1
4373  003506  074552                            EDT1
4374  003510  074600                            EFT1
```

```
4375
4376
4377                                    ;ERROR  177     READ IN PRESET DIDNT CLEAR RMDC
4378
4379  003512  072352                            EMT177
4380  003514  074456                            EHT1
4381  003516  074552                            EDT1
4382  003520  074600                            EFT1
4383
4384
4385                                    ;ERROR  200     CANT SET OFFSET MODE BY OFFSET COMMAND
4386
4387  003522  072370                            EMT200
4388  003524  074456                            EHT1
4389  003526  074552                            EDT1
4390  003530  074600                            EFT1
4391
4392
4393                                    ;ERROR  201     CANT RESET OFFSET MODE BY RTC COMMAND
4394
4395  003532  072406                            EMT201
4396  003534  074456                            EHT1
4397  003536  074552                            EDT1
4398  003540  074600                            EFT1
4399
4400
4401                                    ;ERROR  202     CANT RESET OFD BY RTC COMMAND
4402
4403  003542  072424                            EMT202
4404  003544  074456                            EHT1
4405  003546  074552                            EDT1
4406  003550  074600                            EFT1
4407
4408
4409                                    ;ERROR  203     CANT RESET OM BY RMDC
4410
4411  003552  072442                            EMT203
4412  003554  074456                            EHT1
4413  003556  074552                            EDT1
4414  003560  074600                            EFT1
4415
4416
4417                                    ;ERROR  204     CANT RESET OM BY EBL
4418
4419  003562  072464                            EMT204
4420  003564  074456                            EHT1
4421  003566  074552                            EDT1
4422  003570  074600                            EFT1
4423
4424
4425                                    ;ERROR  205     RUN AND GO NOT CORRECT FOR FUNCTION CODE
4426
4427  003572  072504                            EMT205
4428  003574  074536                            EHT150
4429  003576  074570                            EDT115
4430  003600  074616                            EFT115
```

```
4431
4432
4433                            ;ERROR  206     CANT SET IAE ERROR
4434
4435  003602  072524                      EMT206
4436  003604  000000                      0
4437  003606  000000                      0
4438  003610  000000                      0
4439
4440
4441                            ;ERROR  207     IAE IS INCORRECT FOR FUNCTION CODE
4442
4443  003612  072554                      EMT207
4444  003614  074536                      EHT150
4445  003616  074570                      EDT115
4446  003620  074616                      EFT115
4447
4448
4449                            ;ERROR  210     IAE IS INCORRECT FOR RMDA
4450
4451  003622  072570                      EMT210
4452  003624  074516                      EHT115
4453  003626  074570                      EDT115
4454  003630  074616                      EFT115
4455
4456
4457                            ;ERROR  211     IAE IS INCORRECT FOR RMDC
4458
4459  003632  072606                      EMT211
4460  003634  074516                      EHT115
4461  003636  074570                      EDT115
4462  003640  074616                      EFT115
4463
4464
4465                            ;ERROR  212     CANT SET AOE
4466
4467  003642  072624                      EMT212
4468  003644  074456                      EHT1
4469  003646  074552                      EDT1
4470  003650  074600                      EFT1
4471
4472
4473                            ;ERROR  213     RMR SET WHEN WRITING RMAS OR RMCS
4474
4475  003652  072636                      EMT213
4476  003654  074542                      EHT213
4477  003656  074570                      EDT115
4478  003660  074616                      EFT115
4479
4480
4481                            ;ERROR  214     CANT SET RMR
4482
4483  003662  072666                      EMT214
4484  003664  074542                      EHT213
4485  003666  074570                      EDT115
4486  003670  074616                      EFT115
```

```
4487
4488
4489                                    ;ERROR  215     DRQ IS 0 AND PGM IS 1
4490
4491  003672  072700                            EMT215
4492  003674  074456                            EHT1
4493  003676  074552                            EDT1
4494  003700  074600                            EFT1
4495
4496
4497                                    ;ERROR  216     DVA IS NOT SET
4498
4499  003702  072720                            EMT216
4500  003704  074456                            EHT1
4501  003706  074552                            EDT1
4502  003710  074600                            EFT1
4503
4504
4505                                    ;ERROR  217     DPR IS NOT SET
4506
4507  003712  072734                            EMT217
4508  003714  074456                            EHT1
4509  003716  074552                            EDT1
4510  003720  074600                            EFT1
4511
4512
4513                                    ;ERROR  220     CANT SET PORT REQUEST BY READING RMCS1
4514
4515  003722  072750                            EMT220
4516  003724  074546                            EHT220
4517  003726  074576                            EDT220
4518  003730  074624                            EFT220
4519
4520
4521                                    ;ERROR  221     CANT SET PORT REQUEST BY WRITING RMAS
4522
4523  003732  072766                            EMT221
4524  003734  074546                            EHT220
4525  003736  074576                            EDT220
4526  003740  074624                            EFT220
4527
4528
4529                                    ;ERROR  222     CANT SET PORT REQUEST BY WRITING RMDA
4530
4531  003742  073004                            EMT222
4532  003744  074546                            EHT220
4533  003746  074576                            EDT220
4534  003750  074624                            EFT220
4535
4536
4537                                    ;ERROR  223     CANT RESET PORT REQUEST BY RELEASE COMMAND
4538
4539  003752  073022                            EMT223
4540  003754  074546                            EHT220
4541  003756  074576                            EDT220
4542  003760  074624                            EFT220
```

```
4543
4544
4545                                    ;ERROR  224     CANT CLEAR ATA BY RMAS
4546
4547  003762  073040                              EMT224
4548  003764  074456                              EHT1
4549  003766  074552                              EDT1
4550  003770  074600                              EFT1
4551
4552
4553                                    ;ERROR  225     ATA IS RESET BUT RMAS NOT ZERO
4554
4555  003772  073060                              EMT225
4556  003774  074456                              EHT1
4557  003776  074552                              EDT1
4558  004000  074600                              EFT1
4559
4560
4561                                    ;ERROR  226     CANT RESET ATA BY GO
4562
4563  004002  073102                              EMT226
4564  004004  074456                              EHT1
4565  004006  074552                              EDT1
4566  004010  074600                              EFT1
4567
4568
4569                                    ;ERROR  227     ATA NOT SET BY UNIT READY
4570
4571  004012  073120                              EMT227
4572  004014  074456                              EHT1
4573  004016  074552                              EDT1
4574  004020  074600                              EFT1
4575
4576
4577                                    ;ERROR  230     ATA NOT SET BY UNIT READY
4578
4579  004022  073136                              EMT230
4580  004024  074456                              EHT1
4581  004026  074552                              EDT1
4582  004030  074600                              EFT1
4583
4584
4585                                    ;ERROR  231     ATA NOT SET BY COMP ERROR
4586
4587  004032  073154                              EMT231
4588  004034  074456                              EHT1
4589  004036  074552                              EDT1
4590  004040  074600                              EFT1
4591
4592
4593                                    ;ERROR  232     ATA SET/DID NOT SET WHEN REGISTER WRITTEN
4594                                    ;               WHILE COMP ERROR WAS SET
4595
4596  004042  073170                              EMT232
4597  004044  074542                              EHT213
4598  004046  074570                              EDT115
```

```
4599  004050  074616                        EFT115
4600
4601
4602                                ;ERROR  233     ATA NOT SET BY COMMAND SEQUENCER
4603
4604  004052  073212                        EMT233
4605  004054  074536                        EHT150
4606  004056  074570                        EDT115
4607  004060  074616                        EFT115
4608
4609
4610                                ;ERROR  234     WLE INCORRECT ACCORDING TO FUNCTION CODE
4611
4612  004062  073234                        EMT234
4613  004064  074536                        EHT150
4614  004066  074570                        EDT115
4615  004070  074616                        EFT115
4616
4617
4618                                ;ERROR  235     CANT CLEAR EXCEPTION
4619
4620  004072  073262                        EMT235
4621  004074  074456                        EHT1
4622  004076  074552                        EDT1
4623  004100  074600                        EFT1
4624
4625
4626                                ;ERROR  236     CANT SET EXCEPTION
4627
4628  004102  073302                        EMT236
4629  004104  074516                        EHT115
4630  004106  074570                        EDT115
4631  004110  074616                        EFT115
4632
4633
4634                                ;ERROR  237     CANT CLEAR IVC
4635
4636  004112  073334                        EMT237
4637  004114  074456                        EHT1
4638  004116  074552                        EDT1
4639  004120  074600                        EFT1
4640
4641
4642                                ;ERROR  240     CANT SET IVC
4643
4644  004122  073334                        EMT237
4645  004124  074536                        EHT150
4646  004126  074570                        EDT115
4647  004130  074616                        EFT115
4648
4649
4650                                ;ERROR  241     OPI NOT SET DURING RECALIBRATE
4651
4652  004132  073400                        EMT241
4653  004134  074456                        EHT1
4654  004136  074552                        EDT1
```

```
 4655  004140  074600                    EFT1
 4656
 4657
 4658                            ;ERROR  242     RECALIBRATE DID NOT ABORT WHEN DRIVE FAULT SET
 4659
 4660  004142  073424                    EMT242
 4661  004144  074456                    EHT1
 4662  004146  074552                    EDT1
 4663  004150  074600                    EFT1
 4664
 4665
 4666                            ;ERROR  243     OPI SHOULD HAVE SET BECAUSE ON CYLINDER NEVER
 4667                            ;               DROPPED DURING RECALIBRATE
 4668
 4669  004152  073450                    EMT243
 4670  004154  074456                    EHT1
 4671  004156  074552                    EDT1
 4672  004160  074600                    EFT1
 4673
 4674
 4675                            ;ERROR  244     ATA NOT SET DURING RECALIBRATE
 4676
 4677  004162  073474                    EMT244
 4678  004164  074456                    EHT1
 4679  004166  074552                    EDT1
 4680  004170  074600                    EFT1
 4681
 4682
 4683                            ;ERROR  245     GO RESET EARLY DURING RECALIBRATE
 4684
 4685  004172  073512                    EMT245
 4686  004174  074456                    EHT1
 4687  004176  074552                    EDT1
 4688  004200  074600                    EFT1
 4689
 4690
 4691                            ;ERROR  246     GO NOT RESET DURING RECALIBRATE
 4692
 4693  004202  073530                    EMT246
 4694  004204  074456                    EHT1
 4695  004206  074552                    EDT1
 4696  004210  074600                    EFT1
 4697
 4698
 4699                            ;ERROR  247     INCORRECT TAG BUS DURING RECALIBRATE
 4700
 4701  004212  073554                    EMT247
 4702  004214  074456                    EHT1
 4703  004216  074552                    EDT1
 4704  004220  074600                    EFT1
 4705
 4706
 4707                            ;ERROR  250     OPI SHOULD HAVE SET DURING SEEK BECAUSE UNIT
 4708                            ;               READY DROPPED
 4709
 4710  004222  073572                    EMT250
```

```
4711  004224  074456                  EHT1
4712  004226  074552                  EDT1
4713  004230  074600                  EFT1
4714
4715
4716                          ;ERROR  251     SEEK DID NOT ABORT WHEN DRIVE FAULT SET
4717
4718  004232  073616                  EMT251
4719  004234  074456                  EHT1
4720  004236  074552                  EDT1
4721  004240  074600                  EFT1
4722
4723
4724                          ;ERROR  252     OPI SHOULD HAVE SET BECAUSE ON CYLINDER NEVER
4725                          ;               DROPPED DURING SEEK
4726
4727  004242  073642                  EMT252
4728  004244  074456                  EHT1
4729  004246  074552                  EDT1
4730  004250  074600                  EFT1
4731
4732
4733                          ;ERROR  253     ATA NOT SET DURING SEEK
4734
4735  004252  073666                  EMT253
4736  004254  074456                  EHT1
4737  004256  074552                  EDT1
4738  004260  074600                  EFT1
4739
4740
4741                          ;ERROR  254     GO RESET EARLY DURING SEEK
4742
4743  004262  073704                  EMT254
4744  004264  074456                  EHT1
4745  004266  074552                  EDT1
4746  004270  074600                  EFT1
4747
4748
4749                          ;ERROR  255     GO DID NOT RESET DURING SEEK
4750
4751  004272  073722                  EMT255
4752  004274  074456                  EHT1
4753  004276  074552                  EDT1
4754  004300  074600                  EFT1
4755
4756
4757                          ;ERROR  256     INCORRECT TAG BUS DURING SEEK
4758
4759  004302  073746                  EMT256
4760  004304  074456                  EHT1
4761  004306  074552                  EDT1
4762  004310  074600                  EFT1
4763
4764
4765                          ;ERROR  257     OPI NOT SET DURING SEARCH
4766
```

```
 4767  004312  073764                    EMT257
 4768  004314  074456                    EHT1
 4769  004316  074552                    EDT1
 4770  004320  074600                    EFT1
 4771
 4772
 4773                          ;ERROR  260     SEARCH DID NOT ABORT WHEN DRIVE FAULT SET
 4774
 4775  004322  074010                    EMT260
 4776  004324  074456                    EHT1
 4777  004326  074552                    EDT1
 4778  004330  074600                    EFT1
 4779
 4780
 4781                          ;ERROR  261     OPI SHOULD HAVE SET BECAUSE ON CYLINDER NEVER
 4782                          ;               DROPPED DURING SEARCH
 4783
 4784  004332  074034                    EMT261
 4785  004334  074456                    EHT1
 4786  004336  074552                    EDT1
 4787  004340  074600                    EFT1
 4788
 4789
 4790                          ;ERROR  262     ATA NOT SET DURING SEARCH
 4791
 4792  004342  074060                    EMT262
 4793  004344  074456                    EHT1
 4794  004346  074552                    EDT1
 4795  004350  074600                    EFT1
 4796
 4797
 4798                          ;ERROR  263     GO RESET EARLY DURING SEARCH
 4799
 4800  004352  074076                    EMT263
 4801  004354  074456                    EHT1
 4802  004356  074552                    EDT1
 4803  004360  074600                    EFT1
 4804
 4805
 4806                          ;ERROR  264     GO DID NOT RESET DURING SEARCH
 4807
 4808  004362  074114                    EMT264
 4809  004364  074456                    EHT1
 4810  004366  074552                    EDT1
 4811  004370  074600                    EFT1
 4812
 4813
 4814                          ;ERROR  265     SEARCH ENABLE DIDNT SET DURING SEARCH
 4815
 4816  004372  074140                    EMT265
 4817  004374  074456                    EHT1
 4818  004376  074552                    EDT1
 4819  004400  074600                    EFT1
 4820
 4821
 4822                          ;ERROR  266     INCORRECT TAG BUS DURING SEARCH
```

```
4823
4824  004402  074156                    EMT266
4825  004404  074456                    EHT1
4826  004406  074552                    EDT1
4827  004410  074600                    EFT1
4828
4829
4830                          ;ERROR  267     OPI NOT SET BY SEARCH TIMEOUT
4831
4832  004412  074174                    EMT267
4833  004414  074456                    EHT1
4834  004416  074552                    EDT1
4835  004420  074600                    EFT1
4836
4837
4838                          ;ERROR  270     OPI NOT SET DURING DATA COMMAND
4839
4840  004422  074210                    EMT270
4841  004424  074456                    EHT1
4842  004426  074552                    EDT1
4843  004430  074600                    EFT1
4844
4845
4846                          ;ERROR  271     DATA COMMAND DID NOT ABORT WHEN DRIVE FAULT SET
4847
4848  004432  074234                    EMT271
4849  004434  074456                    EHT1
4850  004436  074552                    EDT1
4851  004440  074600                    EFT1
4852
4853
4854                          ;ERROR  272     EBL RESET EARLY DURING DATA COMMAND
4855
4856  004442  074260                    EMT272
4857  004444  074456                    EHT1
4858  004446  074552                    EDT1
4859  004450  074600                    EFT1
4860
4861
4862                          ;ERROR  273     EBL DIDNT RESET ON TIME DURING DATA COMMAND
4863
4864  004452  074276                    EMT273
4865  004454  074456                    EHT1
4866  004456  074552                    EDT1
4867  004460  074600                    EFT1
4868
4869
4870                          ;ERROR  274     GO NOT RESET DURING DATA COMMAND
4871
4872  004462  074314                    EMT274
4873  004464  074456                    EHT1
4874  004466  074552                    EDT1
4875  004470  074600                    EFT1
4876
4877
4878                          ;ERROR  275     RUN AND GO NOT SET DURING DATA COMMAND
```

```
4879
4880  004472  074332                    EMT275
4881  004474  074456                    EHT1
4882  004476  074552                    EDT1
4883  004500  074600                    EFT1
4884
4885
4886                          ;ERROR  276     INCORRECT TAG BUS DURING DATA COMMAND
4887
4888  004502  074352                    EMT276
4889  004504  074456                    EHT1
4890  004506  074552                    EDT1
4891  004510  074600                    EFT1
4892
4893
4894                          ;ERROR  277     OPI NOT SET DURING DATA COMMAND WHEN ON
4895                          ;               CYLINDER DIDNT DROP
4896
4897  004512  074370                    EMT277
4898  004514  074456                    EHT1
4899  004516  074552                    EDT1
4900  004520  074600                    EFT1
4901
4902
4903                          ;ERROR  300     DATA COMMAND DID NOT ABORT WHEN SEEK ERROR SET
4904
4905  004522  074414                    EMT300
4906  004524  074456                    EHT1
4907  004526  074552                    EDT1
4908  004530  074600                    EFT1
4909
4910
4911                          ;ERROR  301     SEARCH NOT ENABLED DURING DATA COMMAND
4912
4913  004532  074440                    EMT301
4914  004534  074456                    EHT1
4915  004536  074552                    EDT1
4916  004540  074600                    EFT1
4917
4918
4919                          ;PUT ERROR TABLE HERE
4920
```

```
                                        .SBTTL  START OF PROGRAM

                                        START:

                                        ;CLEAR AND SETUP FOR TEST EXECUTION
004542  000240                                NOP
004544  000005                                RESET                   ;INITIALIZE THE SYSTEM

                                        .SBTTL  INITIALIZE THE COMMON TAGS
                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
004546  012706  001114                        MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
004552  005026                                CLR     (R6)+           ;;CLEAR MEMORY LOCATION
004554  022706  001154                        CMP     #SWR,R6 ;;DONE?
004560  001374                                BNE     .-6             ;;LOOP BACK IF NO
004562  012706  001100                        MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                        ;;INITIALIZE A FEW VECTORS
004566  012737  062114  000020                MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
004574  012737  000340  000022                MOV     #340,@#IOTVEC+2 ;;LEVEL 7
004602  012737  062666  000030                MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
004610  012737  000340  000032                MOV     #340,@#EMTVEC+2 ;;LEVEL 7
004616  012737  064426  000034                MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
004624  012737  000340  000036                MOV     #340,@#TRAPVEC+2;LEVEL 7
004632  012737  064534  000024                MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
004640  012737  000340  000026                MOV     #340,@#PWRVEC+2 ;;LEVEL 7
004646  013737  057170  057162                MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
004654  005037  001206                        CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
004660  005037  001210                        CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
004664  112737  000001  001131                MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
004672  012737  004672  001122                MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
004700  012737  004700  001124                MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                        ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                        ;;EQUAL TO A ''-1'', SETUP FOR A SOFTWARE SWITCH REGISTER.
004706  013746  000004                        MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
004712  012737  004746  000004                MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
004720  012737  177570  001154                MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
004726  012737  177570  001156                MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
004734  022777  177777  174212                CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
004742  001012                                BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
004744  000403                                BR      65$             ;;BRANCH IF NO TIMEOUT
004746  012716  004754          64$:          MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
004752  000002                                RTI
004754  012737  000176  001154  65$:          MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
004762  012737  000174  001156                MOV     #DISPREG,DISPLAY
004770  012637  000004          66$:          MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

004774  005037  001230                        CLR     $PASS           ;;CLEAR PASS COUNT
005000  132737  000200  001243                BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
005006  001403                                BEQ     67$             ;;YES,USE NON-APT SWITCH
005010  012737  001244  001154                MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
005016                          67$:
005016  012746  000140                        MOV     #PR3,-(SP)      ;;PUT NEW PS ON STACK
005022  012746  005030                        MOV     #68$,-(SP)      ;;PUT NEW PC ON  STACK
005026  000002                                RTI                     ;;POP NEW PC AND PS
005030                          68$:
```

```
 4977                                           .SBTTL  TYPE PROGRAM NAME
 4978                                           ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
 4979  005030  005227  177777                          INC     #-1             ;;FIRST TIME?
 4980  005034  001062                                  BNE     69$             ;;BRANCH IF NO
 4981  005036  022737  057324  000042                  CMP     #$ENDAD,@#42    ;;ACT-11?
 4982  005044  001456                                  BEQ     69$             ;;BRANCH IF YES
 4983  005046  104401  005114                          TYPE    ,70$            ;;TYPE ASCIZ STRING
 4984                                           .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
 4985  005052  005737  000042                          TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
 4986  005056  001012                                  BNE     71$             ;;BRANCH IF YES
 4987  005060  123727  001242  000001                  CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
 4988  005066  001406                                  BEQ     71$             ;;BRANCH IF YES
 4989  005070  023727  001154  000176                  CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
 4990  005076  001005                                  BNE     72$             ;;BRANCH IF NO
 4991  005100  104407                                  GTSWR                   ;;GET SOFT-SWR SETTINGS
 4992  005102  000403                                  BR      72$
 4993  005104  112737  000001  001150  71$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
 4994  005112                         72$:
 4995  005112  000433                          BR      69$             ;;GET OVER THE ASCIZ
 4996                                   ;;70$:  .ASCIZ  <CRLF>@CZRMJC0, RM03/RM02 DISKLESS DIAGNOSTIC PROGRAM PT 1@<CRLF>
 4997  005202                         69$:
 4998
 4999
 5000                                           ;FIND OUT IF PROGRAM IS RUNNING IN STANDALONE MODE
 5001  005202  005737  000042                          TST     42              ;IS LOC 42 ZERO ??
 5002  005206  001003                                  BNE     10$             ;NO - NOT IN STANDALONE
 5003  005210  105737  001242                          TSTB    $ENV            ;IS APT ENVIRONMENT ZERO ??
 5004  005214  001451                                  BEQ     STANDALONE      ;YES - PROGRAM IN STANDALONE
 5005  005216                         10$:
 5006
 5007                                           ;PROGRAM NOT RUNNING IN STANDALONE - SEE IF SIZING IS ALLOWED
 5008  005216  132737  000200  001243  XSIZ:   BITB    #BIT7,$ENVM     ;SIZING ALLOWED ??
 5009  005224  001043                          BNE     20$             ;NO
 5010                                   ;       MOV     #377,$DEVM      ;YES - SET DEVICE MAP FOR ALL DEVICES
 5011  005226  005037  001300                  CLR     $DEVM           ;CLEAR THE BIT MAP
 5012  005232  005001                          CLR     R1              ;START FROM THE DRIVE 0
 5013  005234  012704  000001                  MOV     #BIT0,R4        ;BIT MAP DOR DRIVE 0
 5014  005240  013700  001276                  MOV     $BASE,R0                ;BASE ADDRESS
 5015  005244  012760  000040  000010  15$:    MOV     #CLR,RMCS2(R0)  ;LOAD THE DRIVE ADDRESS
 5016  005252  010160  000010                  MOV     R1,RMCS2(R0)    ;LOAD THE DRIVE ADDRESS
 5017  005256  016003  000010                  MOV     RMCS2(R0),R3    ;NED BIT SET ?
 5018  005262  032703  010000                  BIT     #NED,R3         ;BRANCH IF SO
 5019  005266  001010                          BNE     16$             ;TO NEXT DRIVE
 5020  005270  016003  000000                  MOV     RMCS1(R0),R3    ;CHECK THE DVA BIT
 5021  005274  032703  004000                  BIT     #DVA,R3         ;DRIVE AVAILABLE ?
 5022  005300  001403                          BEQ     16$             ;BRANCH IF DRIVE NOT AVAILABLE
 5023  005302  050437  001300                  BIS     R4,$DEVM        ;SET THE BIT MAP FOR ALL AVAILABLE DRIVE
 5024  005306  000405                          BR      17$             ;DONT TYPE THE NONE EXIST MESG
 5025  005310                         16$:
 5026  005310  104401  066015                  TYPE    ,NOTEX          ;TYPE NOT EXIT MESSAGE
 5027  005314  010146                          MOV     R1,-(SP)        ;DRIVE NUMBER
 5028  005316  104403                          TYPOS
 5029  005320     006                          .BYTE   6
 5030  005321     000                          .BYTE   0
 5031  005322  005201                 17$:    INC     R1              ;INCREMENT THE DRIVE ADDRESS
 5032  005324  006304                          ASL     R4              ;SET UP THE BIT MAP FOR NEXT DRIVE
```

```
 5033  005326  022701  000007                  CMP     #7,R1           ;ALL DRIVE ARE CHECKED ?
 5034  005332  103344                          BHIS    15$             ;BRANCH IF NOT
 5035  005334                          20$:
 5036
 5037                                  ;GO TO COMMON START CODE
 5038  005334  000137  006166                  JMP     CMNSTART
```

```
5039  005340                                STANDALONE:
5040
5041  005340  004737  063076                        JSR     PC,$TKINT
5042                                        ;SEE IF THIS IS THE FIRST START AFTER PROGRAM WAS LOADED
5043  005344  005327  000001                        DEC     #1              ;FIRST START ??
5044  005350  100024                          BPL     10$             ;YES !!
5045
5046
5047                                        ;SEE IF THE USER WANTS TO KEEP SAME DEVICES FOR TESTING
5048  005352  104401  065324                5$:     TYPE    ,CNSL00         ;MAINTAIN PREVIOUS PARAMETERS??
5049  005356  104411                          RDCHR                   ;GET RESPONSE
5050  005360  012637  001176                  MOV     (SP)+,$TMP1     ;ECHO RESPONSE
5051  005364  104401  001176                  TYPE    ,$TMP1
5052  005370  123727  001176  000131          CMPB    $TMP1,#'Y       ;YES RESPONSE??
5053  005376  001002                          BNE     6$              ;NO!!
5054  005400  000137  006326                  JMP     READY           ;KEEP PREVIOUS PARAMETERS
5055  005404  123727  001176  000116  6$:     CMPB    $TMP1,#'N       ;NO RESPONSE??
5056  005412  001420                          BEQ     20$             ;GET NEW PARAMETERS
5057  005414  104401  065167                  TYPE    ,QSTMRK         ;NOT YES OR NO, TYPE ''?''
5058  005420  000754                          BR      5$              ;RETRY
5059  005422                                10$:
5060
5061                                        ;SEE IF OPERATOR WANTS HELP FILE
5062  005422  104401  065171                  TYPE    ,HELPQST        ;WANT HELP ??
5063  005426  104411                          RDCHR                   ;GET RESPONSE
5064  005430  012637  001176                  MOV     (SP)+,$TMP1     ;SAVE AND ECHO RESPONSE
5065  005434  104401  001176                  TYPE    ,$TMP1
5066  005440  123727  001176  000131          CMPB    $TMP1,#'Y       ;WAS IT A YES RESPONSE ??
5067  005446  001002                          BNE     20$             ;NO - DONT TYPE HELP
5068  005450  104401  106724                  TYPE    ,HELP           ;YES - TYPE HELP TEXT
5069  005454                                20$:
5070
5071                                        ;SEE IF USER WANTS TO CHANGE RM03 UNIBUS ADDRESS
5072  005454  104401  065225                  TYPE    ,UBUSQST        ;WANT TO CHANGE ADDRESS ??
5073  005460  104411                          RDCHR                   ;GET RESPONSE
5074  005462  012637  001176                  MOV     (SP)+,$TMP1     ;SAVE AND ECHO RESPONSE
5075  005466  104401  001176                  TYPE    ,$TMP1
5076  005472  104411                          RDCHR                   ;READ S CR
5077  005474  005726                          TST     (SP)+           ;CLEAR THE STACK
5078  005476  123727  001176  000131          CMPB    $TMP1,#'Y       ;WAS IT A YES RESPONSE ??
5079  005504  001137                          BNE     90$             ;NO !!
5080  005506                                30$:
5081
5082                                        ;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, AND INTERRUPT VECTOR
5083  005506  104401  065363                  TYPE    ,CNSL01         ;TYPE CURRENT BUS ADDRESS
5084  005512  013746  001276                  MOV     $BASE,-(SP)     ;;SAVE $BASE FOR TYPEOUT
5085  005516  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
5086  005520  104401  065160                  TYPE    ,CLSPRN
5087  005524  104413                          RDOCT                   ;GET NEW BUS ADDRESS
5088  005526  012637  001176                  MOV     (SP)+,$TMP1     ;CARRIAGE RETURN??
5089  005532  001412                          BEQ     50$             ;YES-SKIP TO NEXT ENTRY
5090  005534  022737  160000  001176          CMP     #160000,$TMP1   ;BASE ADDRESS IN I/O PAGE??
5091  005542  101403                          BLOS    40$             ;YES
5092  005544  104401  065410                  TYPE    ,CNSL02         ;TYPE WARNING MESSAGE
5093  005550  000756                          BR      30$             ;RETRY
5094  005552  013737  001176  001276  40$:     MOV     $TMP1,$BASE     ;STORE NEW BUS ADDRESS
```

```
 5095  005560  113737  001272  001176  50$:    MOVB    $VECT1,$TMP1    ;TYPE CURRENT VECTOR ADDRESS
 5096  005566  105037  001177                  CLRB    $TMP1+1
 5097  005572  104401  065471                  TYPE    ,CNSL03
 5098  005576  013746  001176                  MOV     $TMP1,-(SP)     ;;SAVE $TMP1 FOR TYPEOUT
 5099  005602  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
 5100  005604     003                          .BYTE   3               ;;TYPE 3 DIGIT(S)
 5101  005605     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
 5102  005606  104401  065160                  TYPE    ,CLSPRN
 5103  005612  104413                          RDOCT                   ;GET NEW VECTOR ADDRESS
 5104  005614  012637  001176                  MOV     (SP)+,$TMP1     ;CARRIAGE RETURN?
 5105  005620  001412                          BEQ     70$             ;YES-SKIP TO NEXT ENTRY
 5106  005622  022737  001000  001176          CMP     #1000,$TMP1     ;VECTOR ADDRESS < 1000??
 5107  005630  101003                          BHI     60$             ;YES!!
 5108  005632  104401  065521                  TYPE    ,CNSL04         ;TYPE WARNING MESSAGE
 5109  005636  000750                          BR      50$             ;RETRY
 5110  005640  113737  001176  001272  60$:    MOVB    $TMP1,$VECT1    ;STORE NEW VECTOR ADDRESS
 5111  005646  113737  001273  001176  70$:    MOVB    $VECT1+1,$TMP1  ;TYPE CURRENT PRIORITY
 5112  005654  006237  001176                  ASR     $TMP1
 5113  005660  006237  001176                  ASR     $TMP1
 5114  005664  006237  001176                  ASR     $TMP1
 5115  005670  006237  001176                  ASR     $TMP1
 5116  005674  006237  001176                  ASR     $TMP1
 5117  005700  105037  001177                  CLRB    $TMP1+1
 5118  005704  104401  065575                  TYPE    ,CNSL05
 5119  005710  013746  001176                  MOV     $TMP1,-(SP)     ;;SAVE $TMP1 FOR TYPEOUT
 5120  005714  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
 5121  005716     001                          .BYTE   1               ;;TYPE 1 DIGIT(S)
 5122  005717     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
 5123  005720  104401  065160                  TYPE    ,CLSPRN
 5124  005724  104413                          RDOCT                   ;GET NEW PRIORITY
 5125  005726  012637  001176                  MOV     (SP)+,$TMP1     ;CARRIAGE RETURN??
 5126  005732  001424                          BEQ     90$             ;YES-SKIP TO NEXT ENTRY
 5127  005734  023727  001176  000007          CMP     $TMP1,#7        ;LEGAL PRIORITY??
 5128  005742  002403                          BLT     80$             ;YES!!
 5129  005744  104401  065631                  TYPE    ,CNSL06         ;TYPE WARNING MESSAGE
 5130  005750  000736                          BR      70$             ;RETRY
 5131  005752                          80$:                            ;STORE NEW PRIORITY
 5132  005752  006337  001176                  ASL     $TMP1
 5133  005756  006337  001176                  ASL     $TMP1
 5134  005762  006337  001176                  ASL     $TMP1
 5135  005766  006337  001176                  ASL     $TMP1
 5136  005772  006337  001176                  ASL     $TMP1
 5137  005776  113737  001176  001273          MOVB    $TMP1,$VECT1+1
 5138  006004                          90$:
 5139
 5140                                  ;DIALOGUE TO INPUT DEVICE NUMBERS
 5141  006004  005037  001300                  CLR     $DEVM           ;CLEAR DEVICE MAP
 5142  006010  104401  065656                  TYPE    ,CNSL07         ;TYPE INPUT INSTRUCTIONS
 5143  006014  104401  065163                  TYPE    ,PROMPT         ;TYPE PROMPTING CHARACTER
 5144  006020  104411                          RDCHR                   ;GET RESPONSE
 5145  006022  012637  001176                  MOV     (SP)+,$TMP1     ;ECHO RESPONSE
 5146  006026  104401  001176                  TYPE    ,$TMP1
 5147  006032  023727  001176  000101          CMP     $TMP1,#'A       ;TEST ALL DRIVES??
 5148  006040  001021                          BNE     110$            ;NO
 5149  006042  000137  005216                  JMP     XSIZ            ;TO SIZE ALL POWER UP DRIVES
 5150  006046  012737  000377  001300          MOV     #377,$DEVM      ;TEST ALL DEVICES
```

```
5151  006054  000444                          BR      140$            ;SKIP TO NEXT ENTRY
5152  006056  104401  065163          100$:   TYPE    ,PROMPT         ;TYPE PROMPTING CHARACTER
5153  006062  104411                          RDCHR                   ;GET RESPONSE
5154  006064  012637  001176                  MOV     (SP)+,$TMP1     ;ECHO RESPONSE
5155  006070  104401  001176                  TYPE    ,$TMP1
5156  006074  023727  001176  000015          CMP     $TMP1,#CR       ;CARRIAGE RETURN??
5157  006102  001431                          BEQ     140$
5158  006104  023727  001176  000060  110$:   CMP     $TMP1,#'0       ;NUMBER < 0??
5159  006112  002404                          BLT     120$            ;YES!!
5160  006114  023727  001176  000067          CMP     $TMP1,#'7       ;NUMBER > 7??
5161  006122  003403                          BLE     130$            ;NO!!
5162  006124  104401  065167          120$:   TYPE    ,QSTMRK         ;TYPE "?"
5163  006130  000752                          BR      100$            ;RETRY
5164  006132  013701  001176          130$:   MOV     $TMP1,R1        ;R1=DRIVE NUMBER
5165  006136  042701  177770                  BIC     #^C7,R1
5166  006142  116102  066142                  MOVB    ATNTBL(R1),R2           ;DECODE DEVICE NUMBER
5167  006146  042702  177400                  BIC     #^C377,R2               ;CLEAR UNUSED BITS
5168  006152  050237  001300                  BIS     R2,$DEVM                ;SET DEVICE # IN MAP
5169  006156  122737  000377  001300          CMPB    #377,$DEVM      ;DONE ??
5170  006164  101334                          BHI     100$            ;NO
5171  006166                          140$:
5172
```

```
5173  006166                                CMNSTART:
5174
5175                                        ;ASSEMBLE TEST QUE FROM DEVICE MAP
5176  006166  013700  001300                      MOV     $DEVM,R0          ;R0 = DEVICE MAP
5177  006172  012701  001460                      MOV     #TSTQUE+2,R1      ;R1 = ADDRESS OF FIRST ENTRY IN QUE
5178  006176  010137  001456                      MOV     R1,TSTQUE         ;INITIALIZE ENTRY POINTER
5179  006202  012702  000001                      MOV     #1,R2             ;R2 = DEVICE POINTER
5180  006206  005003                              CLR     R3                ;R3 = DEVICE NUMBER
5181  006210  030200                      10$:    BIT     R2,R0             ;IS THIS DEVICE IN MAP ??
5182  006212  001406                              BEQ     20$               ;NO !!
5183  006214  010311                              MOV     R3,(R1)           ;YES - ENTER DEVICE NUMBER IN QUE
5184  006216  116361  066142  000001              MOVB    ATNTBL(R3),1(R1);ENTER ATTENTION BIT IN QUE
5185  006224  062701  000002                      ADD     #2,R1             ;ADVANCE ENTRY POINTER
5186  006230  006302                      20$:    ASL     R2                ;ADVANCE DEVICE POINTER
5187  006232  105702                              TSTB    R2                ;DONE ALL DEVICES ??
5188  006234  001402                              BEQ     25$               ;YES
5189  006236  005203                              INC     R3                ;ADVANCE DEVICE NUMBER
5190  006240  000763                              BR      10$               ;ENTER NEXT DEVICE
5191  006242  005011                      25$:    CLR     (R1)              ;TERMINATE TEST QUE
5192
5193                                        ;SIZE FOR CLOCK
5194  006244  004737  060146                      JSR     PC,SIZCLK         ;SEE IF CLOCK PRESENT
5195  006250  000403                              BR      40$               ;YES - CLOCK IS PRESENT
5196  006252  104000                      30$:    ERROR                     ;NO CLOCK
5197  006254  000000                              HALT
5198  006256  000775                              BR      30$
5199  006260                              40$:
5200                                      .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
5201  006260  005737  000042                      TST     @#42              ;;ARE WE RUNNING UNDER XXDP/ACT?
5202  006264  001012                              BNE     64$               ;;BRANCH IF YES
5203  006266  123727  001242  000001              CMPB    $ENV,#1           ;;ARE WE RUNNING UNDER APT?
5204  006274  001406                              BEQ     64$               ;;BRANCH IF YES
5205  006276  023727  001154  000176              CMP     SWR,#SWREG        ;;SOFTWARE SWITCH REG SELECTED?
5206  006304  001005                              BNE     65$               ;;BRANCH IF NO
5207  006306  104407                              GTSWR                     ;;GET SOFT-SWR SETTINGS
5208  006310  000403                              BR      65$
5209  006312  112737  000001  001150      64$:    MOVB    #1,$AUTOB         ;;SET AUTO-MODE INDICATOR
5210  006320                              65$:
5211  006320  012737  000140  000032              MOV     #PR3,@#EMTVEC+2 ;DROP PRIORITY DURING ERROR TYPEOUT
5212  006326  000240                      READY:  NOP                       ;READY TO START TEST
5213  006330  004737  063076                      JSR     PC,$TKINT         ;INITIALIZE TTY
5214  006334  117737  173116  001234              MOVB    @TSTQUE,$UNIT     ;LOAD UNIT NUMBER
```

```
                        .SBTTL  REGISTER AND STORAGE USAGE

                        ;REGISTER ASSIGNMENTS
                        ;
                        ;R0     =       UNIBUS ADDRESS OF RH CONTROLLER
                        ;R1     =       ADDRESS OF ENTRY IN TEST QUE CORRESPONDING TO THE
                        ;               UNIT UNDER TEST
                        ;R2,R3  =       WORKING REGISTERS FOR TEST IN PROGRESS, MUST BE
                        ;               SAVED BY SUBROUTINES
                        ;R4,R5  =       GENERAL WORKING REGISTERS, ARE NOT SAVED BY
                        ;               SUBROUTINES
                        ;R6     =       STACK POINTER
                        ;R7     =       LINKAGE REGISTER TO SUBROUTINES

                        ;STORAGE ASSIGNMENTS
                        ;
                        ;$TMP0-$TMP4    TEMPORARY STORAGE, NOT SAVED BY SUBROUTINES
                        ;$GDDAT,$BDDAT  EXPECTED AND RRECEIVED STATUS FOR ERROR TYPEOUT
                        ;$GDADR,$BDADR  ADDRESS OF EXPECTED AND RECEIVED STATUS IF APPLICABLE,
                        ;               ALSO THE ADDRESS OF A REGISTER ERROR
                        ;
                        ;$TSTN  =       TEST NUMBER
                        ;$UNIT  =       NUMBER OF DEVICE BEING TESTED
                        ;RGINBF =       THE REGISTER INPUT BUFFER HAS A STORAGE LOCATION FOR
                        ;               EACH REGISTER, AND IS USED WHEN READING STATUS AND
                        ;               CONTROL DATA
                        ;RGOTBF =       THE REGISTER OUTPUT BUFFER HAS A STORAGE LOCATION FOR
                        ;               EACH REGISTER, AND IS USED FOR ASSEMBLING DATA TO BE
                        ;               WRITTEN IN REGISTERS
```

```
5244                                    ;;*****************************************************************
5245                                    ;*TEST 1         TRANSFER TEST
5246
5247                                    ;;*****************************************************************
5248  006342  000004                    TST1:   SCOPE
5249  006344  012737  000001  001226            MOV     #1,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
5250
5251  006352  000240                            NOP
5252  006354  012737  000024  001120            MOV     #20.,$ICNT      ;20 ITERATIONS
5253  006362  112737  000001  001131            MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
5254  006370  012737  006404  001122            MOV     #T1,$LPADR  ;LOAD LOOP ON TEST ADDRESS
5255  006376  012737  006404  001124            MOV     #T1,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
5256  006404                            T1:
5257  006404  012706  001100                    MOV     #STACK,SP       ;LOAD THE STACK POINTER
5258  006410  013700  001276                    MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
5259  006414  013701  001456                    MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
5260  006420  012702  000000                    MOV     #0,R2           ;R2 = REGISTER INDEX
5261  006424                            10$:
5262                                    ;CLEAR THE MASSBUS AND VERIFY THAT NONEXISTANT DEVICE ERROR IS RESET
5263  006424  012760  000040  000010            MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5264  006432  111160  000010                    MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5265
5266  006436  016037  000010  001142            MOV     RMCS2(R0),$BDDAT        ;STORE RMCS2 AT $BDDAT
5267  006444  032737  010000  001142            BIT     #NED,$BDDAT
5268  006452  001417                            BEQ     20$
5269  006454  111137  001140                    MOVB    (R1),$GDDAT
5270  006460  042737  177770  001140            BIC     #^CUNTMSK,$GDDAT
5271  006466  052737  000100  001140            BIS     #IR,$GDDAT
5272  006474  010037  001136                    MOV     R0,$BDADR
5273  006500  062737  000010  001136            ADD     #RMCS2,$BDADR
5274  006506  104001                            ERROR   1
5275  006510  000500                            BR      60$
5276  006512                            20$:
5277                                    ;READ THE REGISTER WHOSE INDEX IS IN R2 AND EXIT TEST IF THE READ
5278                                    ;DOES NOT SET 'NED' ERROR
5279  006512  010003                            MOV     R0,R3           ;R3 = REGISTER ADDRESS
5280  006514  060203                            ADD     R2,R3
5281  006516  011304                            MOV     (R3), R4                ;READ REGISTER
5282  006520  032760  010000  000010            BIT     #NED, RMCS2(R0)         ;IS 'NED' SET??
5283  006526  001473                            BEQ     70$             ;NO!!
5284  006530  012760  000040  000010            MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5285  006536  111160  000010                    MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5286
5287  006542  016037  000010  001142            MOV     RMCS2(R0),$BDDAT        ;STORE RMCS2 AT $BDDAT
5288  006550  032737  010000  001142            BIT     #NED,$BDDAT
5289  006556  001417                            BEQ     30$
5290  006560  111137  001140                    MOVB    (R1),$GDDAT
5291  006564  042737  177770  001140            BIC     #^CUNTMSK,$GDDAT
5292  006572  052737  000100  001140            BIS     #IR,$GDDAT
5293  006600  010037  001136                    MOV     R0,$BDADR
5294  006604  062737  000010  001136            ADD     #RMCS2,$BDADR
5295  006612  104001                            ERROR   1
5296  006614  000436                            BR      60$
5297  006616                            30$:
5298                                    ;WRITE THE REGISTER WHOSE INDEX IS IN R2 AND EXIT TEST IF THE WRITE
5299                                    ;DOES NOT SET 'NED' ERROR
```

```
5300  006616  012713  000000          MOV     #0,(R3)          ;WRITE REGISTER
5301  006622  032760  010000  000010  BIT     #NED, RMCS2      (R0)    ;IS 'NED' SET??
5302  006630  001432                  BEQ     70$              ;NO!!
5303
5304  006632                  40$:
5305                          ;COULD NOT READ OR WRITE THE REGISTER WITHOUT SETING 'NED' ERROR -
5306                          ;ADVANCE THE REGISTER INDEX AND REPEAT THE TEST FOR THE NEXT
5307                          ;AVAILABLE DEVICE REGISTER
5308  006632  062702  000002          ADD     #2,R2            ;ADVANCE TO NEXT REGISTER
5309  006636  022702  000002          CMP     #RMWC,R2                 ;IS THIS RMWC??
5310  006642  001773                  BEQ     40$              ;YES - TRY NEXT REGISTER
5311  006644  022702  000004          CMP     #RMBA,R2                 ;IS THIS RMBA??
5312  006650  001770                  BEQ     40$              ;YES - TRY NEXT REGISTER
5313  006652  022702  000010          CMP     #RMCS2,R2                ;IS THIS RMCS5??
5314  006656  001765                  BEQ     40$              ;YES - TRY ANOTHER REGISTER
5315  006660  022702  000016          CMP     #RMAS,R2                 ;IS THIS RMAS ??
5316  006664  001762                  BEQ     40$              ;YES - TRY ANOTHER REGISTER
5317  006666  022702  000022          CMP     #RMDB,R2                 ;IS THIS RMDB??
5318  006672  001757                  BEQ     40$              ;YES - TRY ANOTHER REGISTER
5319  006674  022702  000046          CMP     #RMEC2,R2                ;IS THIS A LEGAL REGISTER
5320  006700  103251                  BHIS    10$              ;YES - TRY THIS REGISTER
5321
5322  006702                  50$:
5323
5324                          ;GOT 'NONEXISTENT DEVICE' ERROR FOR EVERY REMOTE REGISTER ADDRESS
5325  006702  013737  001276  001136          MOV     $BASE, $BDADR            ;STORE BASE ADDRESS
5326  006710  104002                  ERROR   2                ;DEMAND OR TRANSFER FAILED
5327  006712  000137  057100          60$:    JMP     $EOSP            ;GO SELECT NEXT DEVICE
5328
5329  006716                  70$:
5330
5331                          ;;*****************************************************************
5332                          ;*TEST 2        CTOD TEST
5333
5334                          ;;*****************************************************************
5335  006716  000004          TST2:   SCOPE
5336  006720  012737  000002  001226          MOV     #2,$TESTN        ;;SET TEST NUMBER IN APT MAIL BOX
5337  006726  000240                  NOP
5338  006730  012737  000024  001120          MOV     #20.,$ICNT       ;20 ITERATIONS
5339  006736  112737  000001  001131          MOVB    #1,$ERMAX        ;ONE ERROR ALLOWED
5340  006744  012737  006760  001122          MOV     #T2,$LPADR  ;LOAD LOOP ON TEST ADDRESS
5341  006752  012737  006760  001124          MOV     #T2,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
5342  006760                  T2:
5343  006760  012706  001100          MOV     #STACK,SP        ;LOAD THE STACK POINTER
5344  006764  013700  001276          MOV     $BASE,R0         ;R0 = UNIBUS ADDRESS OF UUT
5345  006770  013701  001456          MOV     TSTQUE,R1        ;R1 = POINTER TO DEVICE
5346  006774  012760  000040  000010          MOV     #CLR,RMCS2(R0)   ;CLEAR THE MASSBUS
5347  007002  111160  000010          MOVB    (R1),RMCS2(R0)   ;SELECT UNIT
5348                          ;WRITE ONES IN REMOTE REGISTERS
5349
5350  007006  012760  000076  000000          MOV     #ILF76,RMCS1(R0)         ;LOAD RMCS1
5351
5352  007014  012760  177777  000006          MOV     #-1,RMDA(R0)     ;LOAD RMDA
5353
5354  007022  012760  001777  000034          MOV     #CYLMSK,RMDC(R0)         ;LOAD RMDC
5355
```

```
5356  007030  012760  016200  000032          MOV     #FMT16!ECI!HCI!OFD,RMOF(R0)     ;LOAD RMOF
5357                          ;READ REMOTE REGISTERS TWICE
5358  007036  012702  000001                  MOV     #1,R2
5359  007042                  10$:
5360
5361  007042  016037  000000  001326          MOV     RMCS1(R0),RMCS1I        ;STORE RMCS1 IN INPUT BUFFER
5362
5363  007050  016037  000006  001334          MOV     RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
5364
5365  007056  016037  000034  001362          MOV     RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
5366
5367  007064  016037  000032  001360          MOV     RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
5368  007072  005302                          DEC     R2
5369  007074  100362                          BPL     10$
5370                          ;SEE IF ANY ONE BITS CAME BACK
5371  007076  042737  177701  001326          BIC     #^CILF76,RMCS1I ;IS RMCS1 0??
5372  007104  001014                          BNE     20$             ;NO!!
5373  007106  005737  001334                  TST     RMDAI           ;IS RMDA 0??
5374  007112  001011                          BNE     20$             ;NO!!
5375  007114  042737  176000  001362          BIC     #XNUDC,RMDCI    ;IS RMDC 0??
5376  007122  001005                          BNE     20$             ;NO!!
5377  007124  042737  161577  001360          BIC     #XNUOF,RMOFI    ;IS RMOF 0 ??
5378  007132  001001                          BNE     20$             ;NO!!
5379                          ;CANNOT READ ANY ONE BITS FROM REMOTE REGISTERS
5380  007134  104003                          ERROR   3               ;CTOD MUST BE STUCK
5381  007136                  20$:
5382
5383                          ;;*****************************************************************
5384                          ;*TEST 3        MASSBUS INITIALIZE TEST
5385
5386                          ;;*****************************************************************
5387  007136  000004          TST3:   SCOPE
5388  007140  012737  000003  001226          MOV     #3,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
5389
5390  007146  000240                          NOP
5391  007150  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
5392  007156  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
5393  007164  012737  007200  001122          MOV     #T3,$LPADR  ;LOAD LOOP ON TEST ADDRESS
5394  007172  012737  007200  001124          MOV     #T3,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
5395  007200                  T3:
5396  007200  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
5397  007204  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
5398  007210  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
5399  007214  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5400  007222  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5401                          ;WRITE ONES IN SELECTED REGISTERS
5402
5403  007226  012760  000076  000000          MOV     #ILF76,RMCS1(R0)        ;LOAD RMCS1
5404
5405  007234  012760  177777  000014          MOV     #-1,RMER1(R0)   ;LOAD RMER1
5406
5407  007242  012760  177777  000042          MOV     #-1,RMER2(R0)   ;LOAD RMER2
5408                          ;USING CONTROLLER CLEAR, IE., BIT 5 OF RMCS2, INITIALIZE THE MASSBUS
5409  007250  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5410  007256  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5411                          ;READ THE REGISTERS THAT WERE WRITTEN
```

```
5412
5413  007262  016037  000000  001326          MOV     RMCS1(R0),RMCS1I        ;STORE RMCS1 IN INPUT BUFFER
5414
5415  007270  016037  000014  001342          MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
5416
5417  007276  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
5418                        ;SEE IF ANY REGISTER BITS WERE CLEARED
5419  007304  052737  177701  001326          BIS     #^CILF76,RMCS1I         ;SET ANY BIT NOT WRITTEN
5420  007312  052737  001567  001370          BIS     #XNUER2,RMER2I
5421  007320  022737  177777  001326          CMP     #-1,RMCS1I              ;ANY ZEROS IN RMCS1??
5422  007326  001011                          BNE     10$             ;YES!!
5423  007330  022737  177777  001342          CMP     #-1,RMER1I              ;ANY ZEROS IN RMERI??
5424  007336  001005                          BNE     10$             ;YES!!
5425  007340  022737  177777  001370          CMP     #-1,RMER2I              ;ANY ZEROS IN RMER2??
5426  007346  001001                          BNE     10$
5427                        ;NONE OF THE BITS WERE CLEARED
5428  007350  104004                          ERROR   4               ;MASSBUS INIT FAILED
5429  007352                  10$:
5430
5431                        ;;*****************************************************************
5432                        ;*TEST 4         CLEAR STUCK ACTIVE TEST
5433
5434                        ;;*****************************************************************
5435  007352  000004          TST4:   SCOPE
5436  007354  012737  000004  001226          MOV     #4,$TESTN               ;;SET TEST NUMBER IN APT MAIL BOX
5437
5438  007362  000240                          NOP
5439  007364  012737  000024  001120          MOV     #20.,$ICNT              ;20 ITERATIONS
5440  007372  112737  000001  001131          MOVB    #1,$ERMAX               ;ONE ERROR ALLOWED
5441  007400  012737  007414  001122          MOV     #T4,$LPADR      ;LOAD LOOP ON TEST ADDRESS
5442  007406  012737  007414  001124          MOV     #T4,$LPERR      ;LOAD LOOP ON ERROR ADDRESS
5443  007414                  T4:
5444  007414  012706  001100                  MOV     #STACK,SP               ;LOAD THE STACK POINTER
5445  007420  013700  001276                  MOV     $BASE,R0                ;R0 = UNIBUS ADDRESS OF UUT
5446  007424  013701  001456                  MOV     TSTQUE,R1               ;R1 = POINTER TO DEVICE
5447  007430  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5448  007436  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5449                        ;WRITE ONES IN TEST REGISTERS
5450
5451  007442  012760  177777  000014          MOV     #-1,RMER1(R0)   ;LOAD RMER1
5452
5453  007450  012760  177777  000042          MOV     #-1,RMER2(R0)   ;LOAD RMER2
5454
5455  007456  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
5456                        ;READ TEST REGISTERS AND SEE IF ANY BITS ARE ON
5457
5458  007464  016037  000014  001342          MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
5459
5460  007472  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
5461
5462  007500  016037  000024  001352          MOV     RMMR1(R0),RMMR1I        ;STORE RMMR1 IN INPUT BUFFER
5463  007506  042737  040000  001342          BIC     #UNS,RMER1I     ;DONT ACCEPT UNSAFE
5464  007514  001011                          BNE     10$             ;BRANCH IF ANY OTHER BITS ON
5465  007516  042737  040200  001370          BIC     #SKI!DVC,RMER2I ;DONT ACCEPT SKI OR DVC
5466  007524  001005                          BNE     10$             ;BRANCH IF ANY OTHER BITS ON
5467  007526  032737  000001  001352          BIT     #DMD,RMMR1I     ;BRANCH IF DMD IS ON
```

```
5468  007534  001001                          BNE     10$
5469  007536  104026                          ERROR   26              ;MBA CLR IS STUCK ACTIVE
5470  007540                          10$:
5471
5472
5473                                  ;;*********************************************************************
5474                                  ;*TEST 5         TRISTATE TRANSFER TEST
5475
5476                                  ;;*********************************************************************
5477  007540  000004                  TST5:   SCOPE
5478  007542  012737  000005  001226          MOV     #5,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
5479  007550  000240                          NOP
5480  007552  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
5481  007560  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
5482  007566  012737  007602  001122          MOV     #T5,$LPADR  ;LOAD LOOP ON TEST ADDRESS
5483  007574  012737  007602  001124          MOV     #T5,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
5484  007602                          T5:
5485  007602  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
5486  007606  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
5487  007612  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
5488  007616  005002                          CLR     R2              ;CLEAR ERROR FLAGS
5489  007620  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5490  007626  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5491                                  ;WRITE ONES IN SELECTED REGISTERS
5492
5493  007632  012760  000076  000000          MOV     #ILF76,RMCS1(R0)        ;LOAD RMCS1
5494
5495  007640  012760  177777  000006          MOV     #-1,RMDA(R0)    ;LOAD RMDA
5496
5497  007646  012760  177777  000014          MOV     #-1,RMER1(R0)   ;LOAD RMER1
5498
5499  007654  012760  177777  000032          MOV     #-1,RMOF(R0)    ;LOAD RMOF
5500
5501  007662  012760  177777  000042          MOV     #-1,RMER2(R0)   ;LOAD RMER2
5502                                  ;WRITE ZEROS IN SELECTED REGISTERS
5503
5504  007670  012760  000000  000000          MOV     #0,RMCS1(R0)    ;LOAD RMCS1
5505
5506  007676  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
5507
5508  007704  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
5509
5510  007712  012760  000000  000032          MOV     #0,RMOF(R0)     ;LOAD RMOF
5511
5512  007720  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
5513
5514  007726  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
5515                                  ;READ BACK ALL REGISTERS
5516
5517  007734  016037  000000  001326          MOV     RMCS1(R0),RMCS1I        ;STORE RMCS1 IN INPUT BUFFER
5518
5519  007742  016037  000006  001334          MOV     RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
5520
5521  007750  016037  000014  001342          MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
5522
5523  007756  016037  000032  001360          MOV     RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
```

```
5524
5525  007764  016037  000034  001362          MOV     RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
5526
5527  007772  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
5528                                ;CHECK EACH REGISTER CONTENT FOR ZERO BITS WRITTEN & READ
5529  010000  012702  177777                   MOV     #-1,R2          ;ACCUMULATE ZEROS IN R2
5530  010004  052737  177701  001326          BIS     #^CILF76,RMCS1I ;SET ALL BITS NOT WRITTEN
5531  010012  052737  161577  001360          BIS     #XNUOF,RMOFI
5532  010020  052737  176000  001362          BIS     #XNUDC,RMDCI
5533  010026  052737  001567  001370          BIS     #XNUER2,RMER2I
5534  010034  005137  001326                   COM     RMCS1I          ;COMPLEMENT REGISTER CONTENTS
5535  010040  005137  001334                   COM     RMDAI
5536  010044  005137  001342                   COM     RMER1I
5537  010050  005137  001360                   COM     RMOFI
5538  010054  005137  001362                   COM     RMDCI
5539  010060  005137  001370                   COM     RMER2I
5540  010064  043702  001326                   BIC     RMCS1I,R2               ;ACCUMULATE ALL ZERO BITS
5541  010070  043702  001334                   BIC     RMDAI,R2
5542  010074  043702  001342                   BIC     RMER1I,R2
5543  010100  043702  001360                   BIC     RMOFI,R2
5544  010104  043702  001362                   BIC     RMDCI,R2
5545  010110  043702  001370                   BIC     RMER2I,R2
5546  010114  001407                           BEQ     10$             ;BRANCH IF EACH BIT IS ZERO
5547                                ;ONE OR MORE BIT POSITIONS ARE NOT ZERO
5548  010116  010237  001142                   MOV     R2,$BDDAT               ;SAVE RESULT FOR TYPE
5549  010122  005037  001140                   CLR     $GDDAT          ;LOAD EXPECTED RESULT
5550  010126  104005                           ERROR   5               ;TRISTATE BUS IS STUCK AT ONE
5551  010130  052702  000001                   BIS     #BIT0,R2        ;SET ERROR FLAG
5552  010134                          10$:
5553
5554  010134  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5555  010142  111160  000010                   MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5556                                ;PRESET SELECTED REGISTERS TO ZEROS
5557                                ;(ASSUME RMCS1, RMER1, RMER2 WERE CLEARED BY INIT)
5558
5559  010146  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
5560
5561  010154  012760  000000  000032          MOV     #0,RMOF(R0)     ;LOAD RMOF
5562
5563  010162  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
5564                                ;WRITE ONES IN SELECTED REGISTERS
5565
5566  010170  012760  000076  000000          MOV     #ILF76,RMCS1(R0)        ;LOAD RMCS1
5567
5568  010176  012760  177777  000006          MOV     #-1,RMDA(R0)    ;LOAD RMDA
5569
5570  010204  012760  016200  000032          MOV     #^CXNUOF,RMOF(R0)       ;LOAD RMOF
5571
5572  010212  012760  001777  000034          MOV     #^CXNUDC,RMDC(R0)       ;LOAD RMDC
5573
5574  010220  012760  177777  000014          MOV     #-1,RMER1(R0)   ;LOAD RMER1
5575
5576  010226  012760  176210  000042          MOV     #^CXNUER2,RMER2(R0)     ;LOAD RMER2
5577                                ;READ ALL REGISTERS
5578
5579  010234  016037  000000  001326          MOV     RMCS1(R0),RMCS1I        ;STORE RMCS1 IN INPUT BUFFER
```

```
5580
5581  010242  016037  000006  001334          MOV     RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
5582
5583  010250  016037  000032  001360          MOV     RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
5584
5585  010256  016037  000034  001362          MOV     RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
5586
5587  010264  016037  000014  001342          MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
5588
5589  010272  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
5590                          ;CHECK EACH REGISTER CONTENT FOR ONE BITS WRITTEN & READ
5591  010300  042737  177701  001326          BIC     #^CILF76,RMCS1I         ;CLEAR ALL BITS NOT WRITTEN
5592  010306  042737  161577  001360          BIC     #XNUOF,RMOFI
5593  010314  042737  176000  001362          BIC     #XNUDC,RMDCI
5594  010322  042737  001567  001370          BIC     #XNUER2,RMER2I
5595  010330  005002                          CLR     R2              ;ACCUMULATE ONES IN R2
5596  010332  053702  001326                  BIS     RMCS1I,R2               ;ACCUMULATE ALL ONE BITS
5597  010336  053702  001334                  BIS     RMDAI,R2
5598  010342  053702  001360                  BIS     RMOFI,R2
5599  010346  053702  001362                  BIS     RMDCI,R2
5600  010352  053702  001342                  BIS     RMER1I,R2
5601  010356  053702  001370                  BIS     RMER2I,R2
5602  010362  022702  177777                  CMP     #-1,R2          ;SEE IF EACH BIT POSITION WAS ONE
5603  010366  001410                          BEQ     20$             ;BRANCH IF NONE STUCK
5604                          ;ONE OR MORE BIT POSITIONS ARE NOT ONE
5605  010370  010237  001142                  MOV     R2,$BDDAT               ;SAVE RESULT FOR TYPE
5606  010374  012737  177777  001140          MOV     #-1,$GDDAT              ;EXPECTED RESULT
5607  010402  104006                          ERROR   6
5608  010404  052702  000002                  BIS     #BIT1,R2        ;SET ERROR FLAG
5609  010410                          20$:
5610
5611  010410  005702                          TST     R2              ;ANY ERRORS DETECTED ??
5612  010412  001131                          BNE     30$             ;YES - DONT DO BIT TEST
5613  010414  012702  000001                  MOV     #1,R2           ;R2=BIT POSITION
5614  010420                          25$:
5615  010420  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5616  010426  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5617                          ;WRITE THE BIT PATTERN IN SELECTED DEVICE REGISTERS
5618
5619  010432  010260  000006                  MOV     R2,RMDA(R0)     ;LOAD RMDA
5620
5621  010436  010260  000032                  MOV     R2,RMOF(R0)     ;LOAD RMOF
5622
5623  010442  010260  000034                  MOV     R2,RMDC(R0)     ;LOAD RMDC
5624
5625  010446  010260  000014                  MOV     R2,RMER1(R0)    ;LOAD RMER1
5626
5627  010452  010260  000042                  MOV     R2,RMER2(R0)    ;LOAD RMER2
5628                          ;READ BACK THE REGISTERS
5629
5630  010456  016037  000006  001334          MOV     RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
5631
5632  010464  016037  000032  001360          MOV     RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
5633
5634  010472  016037  000034  001362          MOV     RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
5635
```

```
5636  010500  016037  000014  001342          MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
5637
5638  010506  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
5639                          ;CHECK REGISTER CONTENTS FOR CORRECT PATTERN
5640  010514  005003                  CLR     R3              ;R3=ACCUMULATED ONE BIT
5641  010516  012704  177777          MOV     #-1,R4          ;R4=ACCUMULATED ZERO BITS
5642  010522  013705  001334          MOV     RMDAI,R5                ;GET ANY GOOD BITS FROM RMDA
5643  010526  050503                  BIS     R5,R3
5644  010530  005105                  COM     R5
5645  010532  040504                  BIC     R5,R4
5646  010534  013705  001360          MOV     RMOFI,R5                ;GET GOOD BITS FROM RMOF
5647  010540  042705  161577          BIC     #XNUOF,R5
5648  010544  050503                  BIS     R5,R3
5649  010546  005105                  COM     R5
5650  010550  042705  161577          BIC     #XNUOF,R5
5651  010554  040504                  BIC     R5,R4
5652  010556  013705  001362          MOV     RMDCI,R5                ;GET GOOD BITS FROM RMDC
5653  010562  042705  176000          BIC     #XNUDC,R5
5654  010566  050503                  BIS     R5,R3
5655  010570  005105                  COM     R5
5656  010572  042705  176000          BIC     #XNUDC,R5
5657  010576  040504                  BIC     R5,R4
5658  010600  013705  001342          MOV     RMER1I,R5               ;GET GOOD BITS FROM RMER1
5659  010604  050503                  BIS     R5,R3
5660  010606  005105                  COM     R5
5661  010610  040504                  BIC     R5,R4
5662  010612  013705  001370          MOV     RMER2I,R5               ;GET GOOD BITS FROM RMER2
5663  010616  042705  001567          BIC     #XNUER2,R5
5664  010622  050503                  BIS     R5,R3
5665  010624  005105                  COM     R5
5666  010626  042705  001567          BIC     #XNUER2,R5
5667  010632  040504                  BIC     R5,R4
5668  010634  010205                  MOV     R2,R5           ;RESET ALL ONES IN R3 EXCEPT
5669  010636  005105                  COM     R5              ;FOR THE TEST BIT
5670  010640  040503                  BIC     R5,R3
5671  010642  040204                  BIC     R2,R4           ;RESET TEST BIT IN R4
5672  010644  050403                  BIS     R4,R3           ;COMBINE ACCUMULATED 1'S + 0'S
5673  010646  020302                  CMP     R3,R2           ;IS PATTERN OK??
5674  010650  001406                  BEQ     26$             ;YES!!
5675  010652  010237  001140          MOV     R2,$GDDAT               ;SAVE TEST PATTERN
5676  010656  010337  001142          MOV     R3,$BDDAT               ;SAVE RESULT
5677  010662  104007                  ERROR   7               ;BIT INTERFERENCE IN TRISTATE BUS
5678  010664  000404                  BR      30$             ;SKIP TO NEXT
5679  010666                  26$:
5680                          ;ADVANCE R2 TO THE NEXT PATTERN AND REPEAT TEST
5681  010666  006302                  ASL     R2              ;SHIFT THE BIT
5682  010670  001402                  BEQ     30$             ;EXIT IF DONE
5683  010672  000137  010420          JMP     25$
5684  010676                  30$:
5685
5686                          ;;*******************************************************************
5687                          ;*TEST 6         REGISTER SELECT TEST
5688
5689                          ;;*******************************************************************
5690  010676  000004          TST6:   SCOPE
5691  010700  012737  000006  001226          MOV     #6,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
```

```
5692
5693  010706  000240                          NOP
5694  010710  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
5695  010716  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
5696  010724  012737  010740  001122          MOV     #T6,$LPADR  ;LOAD LOOP ON TEST ADDRESS
5697  010732  012737  010740  001124          MOV     #T6,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
5698  010740                          T6:
5699  010740  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
5700  010744  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
5701  010750  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
5702                                  ;THE FOLLOWING TABLE GIVES MASSBUS REGISTER SELECT VALUES FOR
5703                                  ;EACH DEVICE REGISTER
5704
5705                                  ;       REGISTER        REG SEL
5706                                  ;       NAME            (16,8,4,2,1)
5707                                  ;
5708                                  ;       RMCS1           00000
5709                                  ;       RMDS            00001
5710                                  ;       RMER1           00010
5711                                  ;       RMMR1           00011
5712                                  ;       RMAS            00100
5713                                  ;       RMDA            00101
5714                                  ;       RMDT            00110
5715                                  ;       RMLA            00111
5716                                  ;       RMSN            01000
5717                                  ;       RMOF            01001
5718                                  ;       RMDC            01010
5719                                  ;       RMHR            01011
5720                                  ;       RMMR2           01100
5721                                  ;       RMER2           01101
5722                                  ;       RMEC1           01110
5723                                  ;       RMEC2           01111
5724
5725                                  ;EACH REGISTER SELECT LINE IS TESTED FOR A STUCK AT ONE,
5726                                  ;STUCK AT ZERO FAULT. AS AN EXAMPLE, TO TEST REG SEL 1.
5727                                  ;FOR S-A-0, RMER1 IS WRITTEN WITH ZEROS. THEN THE REGISTER
5728                                  ;THAT HAS THE SAME SELECT VALUE, EXCEPT FOR THE SELECT LINE
5729                                  ;BEING TESTED, IS WRITTEN WITH ONES. IN THIS EXAMPLE,
5730                                  ;RMMR1 IS WRITTEN WITH ONES. IF SELECT LINE 1 IS S-A-0,
5731                                  ;THE ALL ONES WORD WILL BE WRITTEN IN RMER1, AND RMER1
5732                                  ;WILL NOT BE 0 WHEN READ BACK.
5733
5734  010754  005002                          CLR     R2              ;R2= ZEROS SOURCE
5735  010756  012703  177777                  MOV     #-1,R3          ;R3= ONES  SOURCE
5736                                  ;TEST REG SEL 1 FOR S-A-0
5737  010762  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5738  010770  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5739
5740  010774  010260  000034                  MOV     R2,RMER1(R0)    ;LOAD RMER1
5741
5742  011000  010260  000034                  MOV     R2,RMDC(R0)     ;LOAD RMDC
5743
5744  011004  010360  000024                  MOV     R3,RMMR1(R0)    ;LOAD RMMR1
5745
5746  011010  010360  000036                  MOV     R3,RMHR(R0)     ;LOAD RMHR
5747
```

```
5748  011014  016037  000014  001342          MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
5749
5750  011022  016037  000034  001362          MOV     RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
5751  011030  020337  001342                  CMP     R3,RMER1I
5752  011034  001007                          BNE     10$
5753  011036  052737  176000  001362          BIS     #XNUDC,RMDCI
5754  011044  020337  001362                  CMP     R3,RMDCI
5755  011050  001001                          BNE     10$
5756  011052  104010                          ERROR   10              ;REG SEL 1 IS S-A-0
5757  011054                          10$:
5758
5759                                  ;TEST REG SEL 1 FOR S-A-1
5760  011054  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5761  011062  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5762
5763  011066  010260  000006                  MOV     R2,RMDA(R0)     ;LOAD RMDA
5764
5765  011072  010260  000032                  MOV     R2,RMOF(R0)     ;LOAD RMOF
5766
5767  011076  010260  000042                  MOV     R2,RMER2(R0)    ;LOAD RMER2
5768
5769  011102  010360  000016                  MOV     R3,RMAS(R0)     ;LOAD RMAS
5770
5771  011106  010360  000030                  MOV     R3,RMSN(R0)     ;LOAD RMSN
5772
5773  011112  010360  000040                  MOV     R3,RMMR2(R0)    ;LOAD RMMR2
5774
5775  011116  016037  000006  001334          MOV     RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
5776
5777  011124  016037  000032  001360          MOV     RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
5778
5779  011132  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
5780  011140  020337  001334                  CMP     R3,RMDAI
5781  011144  001015                          BNE     20$
5782  011146  052737  161577  001360          BIS     #XNUOF,RMOFI
5783  011154  020337  001360                  CMP     R3,RMOFI
5784  011160  001007                          BNE     20$
5785  011162  052737  001567  001370          BIS     #XNUER2,RMER2I
5786  011170  020337  001370                  CMP     R3,RMER2I
5787  011174  001001                          BNE     20$
5788  011176  104011                          ERROR   11              ;REG SEL 1 IS S-A-1
5789  011200                          20$:
5790
5791                                  ;TEST REG SEL 2 FOR S-A-0
5792  011200  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5793  011206  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5794
5795  011212  010260  000006                  MOV     R2,RMDA(R0)     ;LOAD RMDA
5796
5797  011216  010260  000032                  MOV     R2,RMOF(R0)     ;LOAD RMOF
5798
5799  011222  010260  000042                  MOV     R2,RMER2(R0)    ;LOAD RMER2
5800
5801  011226  010360  000020                  MOV     R3,RMLA(R0)     ;LOAD RMLA
5802
5803  011232  010360  000036                  MOV     R3,RMHR(R0)     ;LOAD RMHR
```

```
5804
5805  011236  010360  000046                MOV     R3,RMEC2(R0)    ;LOAD RMEC2
5806
5807  011242  016037  000006  001334        MOV     RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
5808
5809  011250  016037  000032  001360        MOV     RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
5810
5811  011256  016037  000042  001370        MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
5812  011264  020337  001334                CMP     R3,RMDAI
5813  011270  001015                        BNE     30$
5814  011272  052737  161577  001360        BIS     #XNUOF,RMOFI
5815  011300  020337  001360                CMP     R3,RMOFI
5816  011304  001007                        BNE     30$
5817  011306  052737  001567  001370        BIS     #XNUER2,RMER2I
5818  011314  020337  001370                CMP     R3,RMER2I
5819  011320  001001                        BNE     30$
5820  011322  104012                        ERROR   12              ;REG SEL 2 IS S-A-0
5821  011324                        30$:
5822
5823                                ;TEST REG SEL 2 FOR S-A-1
5824  011324  012760  000040  000010        MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5825  011332  111160  000010                MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5826
5827  011336  010260  000014                MOV     R2,RMER1(R0)    ;LOAD RMER1
5828
5829  011342  010260  000034                MOV     R2,RMDC(R0)     ;LOAD RMDC
5830
5831  011346  012760  000076  000000        MOV     #ILF76,RMCS1(R0)        ;LOAD RMCS1
5832
5833  011354  010360  000030                MOV     R3,RMSN(R0)     ;LOAD RMSN
5834
5835  011360  016037  000014  001342        MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
5836
5837  011366  016037  000034  001362        MOV     RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
5838  011374  052737  177701  001342        BIS     #^CILF76,RMER1I
5839  011402  020337  001342                CMP     R3,RMER1I
5840  011406  001007                        BNE     40$
5841  011410  052737  176000  001362        BIS     #XNUDC,RMDCI
5842  011416  020337  001362                CMP     R3,RMDCI
5843  011422  001001                        BNE     40$
5844  011424  104013                        ERROR   13              ;REG SEL 2 IS S-A-1
5845  011426                        40$:
5846
5847                                ;TEST REG SEL 4 FOR S-A-0
5848  011426  012760  000040  000010        MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5849  011434  111160  000010                MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5850
5851  011440  010260  000014                MOV     R2,RMER1(R0)    ;LOAD RMER1
5852
5853  011444  010260  000032                MOV     R2,RMOF(R0)     ;LOAD RMOF
5854
5855  011450  010260  000034                MOV     R2,RMDC(R0)     ;LOAD RMDC
5856
5857  011454  010360  000026                MOV     R3,RMDT(R0)     ;LOAD RMDT
5858
5859  011460  010360  000042                MOV     R3,RMER2(R0)    ;LOAD RMER2
```

```
5860
5861  011464  010360  000044                  MOV     R3,RMEC1(R0)    ;LOAD RMEC1
5862
5863  011470  016037  000014  001342          MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
5864
5865  011476  016037  000032  001360          MOV     RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
5866
5867  011504  016037  000034  001362          MOV     RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
5868  011512  020337  001342                  CMP     R3,RMER1I
5869  011516  001015                          BNE     50$
5870  011520  052737  161577  001360          BIS     #XNUOF,RMOFI
5871  011526  020337  001360                  CMP     R3,RMOFI
5872  011532  001007                          BNE     50$
5873  011534  052737  176000  001362          BIS     #XNUDC,RMDCI
5874  011542  020337  001362                  CMP     R3,RMDCI
5875  011546  001001                          BNE     50$
5876  011550  104014                          ERROR   14              ;REG SEL 4 IS S-A-0
5877  011552                          50$:
5878
5879                                  ;TEST REG SEL 4 FOR S-A-1
5880  011552  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5881  011560  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5882
5883  011564  010260  000006                  MOV     R2,RMDA(R0)     ;LOAD RMDA
5884
5885  011570  010260  000042                  MOV     R2,RMER2(R0)    ;LOAD RMER2
5886
5887  011574  010360  000012                  MOV     R3,RMDS(R0)     ;LOAD RMDS
5888
5889  011600  010360  000032                  MOV     R3,RMOF(R0)     ;LOAD RMOF
5890
5891  011604  016037  000006  001334          MOV     RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
5892
5893  011612  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
5894  011620  020337  001334                  CMP     R3,RMDAI
5895  011624  001007                          BNE     60$
5896  011626  052737  001567  001370          BIS     #XNUER2,RMER2I
5897  011634  020337  001370                  CMP     R3,RMER2I
5898  011640  001001                          BNE     60$
5899  011642  104015                          ERROR   15              ;REG SEL 4 IS S-A-1
5900  011644                          60$:
5901
5902                                  ;TEST REG SEL 8 FOR S-A-0
5903  011644  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5904  011652  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5905
5906  011656  010260  000014                  MOV     R2,RMER1(R0)    ;LOAD RMER1
5907
5908  011662  010260  000006                  MOV     R2,RMDA(R0)     ;LOAD RMDA
5909
5910  011666  010360  000034                  MOV     R3,RMDC(R0)     ;LOAD RMDC
5911
5912  011672  010360  000042                  MOV     R3,RMER2(R0)    ;LOAD RMER2
5913
5914  011676  016037  000014  001342          MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
5915
```

```
5916  011704  016037  000006  001334          MOV     RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
5917  011712  020337  001342                  CMP     R3,RMER1I
5918  011716  001004                          BNE     70$
5919  011720  020337  001334                  CMP     R3,RMDAI
5920  011724  001001                          BNE     70$
5921  011726  104016                          ERROR   16              ;REG SEL 8 IS S-A-0
5922  011730                          70$:
5923
5924                                  ;TEST REG SEL 8 FOR S-A-1
5925  011730  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
5926  011736  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
5927
5928  011742  010260  000032                  MOV     R2,RMOF(R0)     ;LOAD RMOF
5929
5930  011746  010260  000034                  MOV     R2,RMDC(R0)     ;LOAD RMDC
5931
5932  011752  010260  000042                  MOV     R2,RMER2(R0)    ;LOAD RMER2
5933
5934  011756  010360  000012                  MOV     R3,RMDS(R0)     ;LOAD RMDS
5935
5936  011762  010360  000014                  MOV     R3,RMER1(R0)    ;LOAD RMER1
5937
5938  011766  010360  000006                  MOV     R3,RMDA(R0)     ;LOAD RMDA
5939
5940  011772  016037  000032  001360          MOV     RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
5941
5942  012000  016037  000034  001362          MOV     RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
5943
5944  012006  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
5945  012014  052737  161577  001360          BIS     #XNUOF,RMOFI
5946  012022  001015                          BNE     80$
5947  012024  022737  176000  001362          CMP     #XNUDC,RMDCI
5948  012032  020337  001362                  CMP     R3,RMDCI
5949  012036  001007                          BNE     80$
5950  012040  052737  001567  001370          BIS     #XNUER2,RMER2I
5951  012046  020337  001370                  CMP     R3,RMER2I
5952  012052  001001                          BNE     80$
5953  012054  104017                          ERROR   17              ;REG SEL 8 IS S-A-1
5954  012056                          80$:
5955
5956                                  ;REGISTER SELECT 16 IS TESTED BY THE ILR TEST
5957
5958                                  ;;*****************************************************************
5959                                  ;*TEST 7         DRIVE TYPE TEST
5960
5961                                  ;;*****************************************************************
5962  012056  000004                  TST7:   SCOPE
5963  012060  012737  000007  001226          MOV     #7,$TESTN       ;;SET TEST NUMBER IN APT MAIL BOX
5964
5965  012066  000240                          NOP
5966  012070  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
5967  012076  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
5968  012104  012737  012120  001122          MOV     #T7,$LPADR  ;LOAD LOOP ON TEST ADDRESS
5969  012112  012737  012120  001124          MOV     #T7,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
5970  012120                          T7:
5971  012120  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
```

```
5972  012124  013700  001276            MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
5973  012130  013701  001456            MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
5974
5975  012134  016037  000026  001142    MOV     RMDT(R0),$BDDAT ;STORE RMDT AT $BDDAT
5976  012142  022737  020024  001142    CMP     #SNGPRT,$BDDAT  ;SINGLE PORT RM03??
5977  012150  001432                    BEQ     10$             ;YES!!
5978  012152  022737  024024  001142    CMP     #DULPRT,$BDDAT  ;DUAL PORT RM03??
5979  012160  001426                    BEQ     10$             ;YES!!
5980  012162  022737  020025  001142    CMP     #SNGPRT!BIT0,$BDDAT     ;SINGLE PROT RM02 ?
5981  012170  001422                    BEQ     10$                     ;BRANCH IF SO
5982  012172  022737  024025  001142    CMP     #DULPRT!BIT0,$BDDAT     ;DUAL PORT RM02 ?
5983  012200  001416                    BEQ     10$                     ;BRANCH IF SO
5984  012202  012737  020024  001174    MOV     #SNGPRT,$TMP0   ;LOAD ACCEPTABLE VALUES
5985  012210  012737  024024  001176    MOV     #DULPRT,$TMP1
5986  012216  010037  001136            MOV     R0,$BDADR       ;LOAD BAD ADDRESS
5987  012222  062737  000026  001136    ADD     #RMDT,$BDADR
5988  012230  104057                    ERROR   57              ;DEVICE NOT AN RM03
5989  012232  000137  057100            JMP     $EOSP           ;GO TO NEXT DEVICE
5990  012236                    10$:
5991
5992                            ;;*******************************************************************
5993                            ;*TEST 10        DEVICE AVAILABLE TEST
5994
5995                            ;;*******************************************************************
5996  012236  000004            TST10:  SCOPE
5997  012240  012737  000010  001226    MOV     #10,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
5998
5999  012246  000240                    NOP
6000  012250  012737  000024  001120    MOV     #20.,$ICNT      ;20 ITERATIONS
6001  012256  112737  000001  001131    MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
6002  012264  012737  012300  001122    MOV     #T10,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6003  012272  012737  012300  001124    MOV     #T10,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6004  012300                    T10:
6005  012300  012706  001100            MOV     #STACK,SP       ;LOAD THE STACK POINTER
6006  012304  013700  001276            MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6007  012310  013701  001456            MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6008  012314  012760  000040  000010    MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6009  012322  111160  000010            MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6010
6011  012326  016037  000000  001142    MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
6012  012334  042737  173777  001142    BIC     #^CDVA,$BDDAT   ;CLEAR ALL BUT DVA
6013  012342  001006                    BNE     10$             ;BRANCH IF DVA SET
6014  012344  012737  004000  001140    MOV     #DVA,$GDDAT     ;SETUP EXPECTED
6015  012352  010037  001136            MOV     R0,$BDADR       ;SETUP REG ADDRESS
6016  012356  104060                    ERROR   60              ;DEVICE NOT AVAILABLE
6017  012360                    10$:
6018
6019                            ;;*******************************************************************
6020                            ;*TEST 11        HOLDING REGISTER TRANSFER TEST
6021
6022                            ;;*******************************************************************
6023  012360  000004            TST11:  SCOPE
6024  012362  012737  000011  001226    MOV     #11,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
6025
6026  012370  000240                    NOP
6027  012372  012737  000024  001120    MOV     #20.,$ICNT      ;20 ITERATIONS
```

```
6028  012400  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
6029  012406  012737  012422  001122          MOV     #T11,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6030  012414  012737  012422  001124          MOV     #T11,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6031  012422                              T11:
6032  012422  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
6033  012426  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6034  012432  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6035  012436  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6036  012444  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6037  012450  005003                          CLR     R3              ;CLEAR ERROR FLAGS
6038  012452  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
6039  012456  062737  000036  001136          ADD     #RMHR,$BDADR
6040                                  ;WRITE ONES THEN ZEROS IN RMHR AND CHECK FOR S-A-1 BITS.
6041                                  ;NOTE THAT IT IS NECESSARY TO WRITE SOME OTHER REGISTER IN
6042                                  ;ORDER TO WRITE THE HOLDING REGISTER, AND RMDA IS USED FOR THIS
6043                                  ;PURPOSE.
6044
6045  012464  012760  177777  000006          MOV     #-1,RMDA(R0)    ;LOAD RMDA
6046
6047  012472  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
6048
6049  012500  016037  000036  001142          MOV     RMHR(R0),$BDDAT ;STORE RMHR AT $BDDAT
6050  012506  005137  001142                  COM     $BDDAT          ;ANY ERROR??
6051  012512  001405                          BEQ     10$             ;NO!!
6052  012514  005037  001140                  CLR     $GDDAT          ;LOAD EXPECTED
6053  012520  104061                          ERROR   61
6054  012522  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAGS
6055  012526                              10$:
6056                                  ;;**********************************************************************
6057                                  ;WRITE ZEROS THEN ONES IN RMHR AND CHECK FOR S-A-0 BITS.
6058
6059  012526  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
6060
6061  012534  012760  177777  000006          MOV     #-1,RMDA(R0)    ;LOAD RMDA
6062
6063  012542  016037  000036  001142          MOV     RMHR(R0),$BDDAT ;STORE RMHR AT $BDDAT
6064  012550  005137  001142                  COM     $BDDAT          ;RMHR IS COMPLEMENTED WHEN READ
6065  012554  012737  177777  001140          MOV     #-1,$GDDAT      ;SETUP EXPECTED
6066  012562  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;ANY ERROR??
6067  012570  001403                          BEQ     20$             ;NO!!
6068  012572  104062                          ERROR   62
6069  012574  052703  000002                  BIS     #BIT1,R3        ;SET ERROR FLAG
6070  012600                              20$:
6071                                  ;;**********************************************************************
6072                                  ;IF NO PREVIOUS ERRORS, WRITE AND READ SHIFTING ONE BIT PATTERN.
6073  012600  005703                          TST     R3              ;ANY FLAGS SET??
6074  012602  001025                          BNE     50$             ;YES!!
6075  012604  012702  000001                  MOV     #1,R2           ;R2=DATA PATTERN
6076
6077  012610                              30$:
6078
6079  012610  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
6080
6081  012616  010260  000006                  MOV     R2,RMDA(R0)     ;LOAD RMDA
6082
6083  012622  016037  000036  001142          MOV     RMHR(R0),$BDDAT ;STORE RMHR AT $BDDAT
```

```
6084  012630  005137  001142                  COM     $BDDAT          ;RMHR IS COMPLEMENTED
6085  012634  023702  001142                  CMP     $BDDAT,R2       ;ANY ERROR??
6086  012640  001404                          BEQ     40$             ;NO!!
6087  012642  010237  001140                  MOV     R2,$GDDAT       ;SETUP EXPECTED
6088  012646  104063                          ERROR   63
6089  012650  000402                          BR      50$             ;DO NOT COLLECT ALL ERRORS
6090
6091  012652  006302                  40$:    ASL     R2              ;SHIFT TO NEXT PATTERN
6092  012654  001355                          BNE     30$             ;CONTINUE IF NOT DONE
6093
6094  012656                          50$:
6095
6096                                  ;;*****************************************************************
6097                                  ;*TEST 12        CONTROL STATUS #1 TRANSFER TEST
6098
6099                                  ;;*****************************************************************
6100  012656  000004                  TST12:  SCOPE
6101  012660  012737  000012  001226          MOV     #12,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
6102
6103  012666  000240                          NOP
6104  012670  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
6105  012676  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
6106  012704  012737  012720  001122          MOV     #T12,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6107  012712  012737  012720  001124          MOV     #T12,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6108  012720                          T12:
6109  012720  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
6110  012724  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6111  012730  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6112  012734  005003                          CLR     R3              ;R3 = ERROR INDICATOR
6113  012736  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6114  012744  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6115                                  ;WRITE ONES IN RMCS1, BITS 01-05, THEN CLEAR. READ AND
6116                                  ;CHECK FOR S-A-1 BITS.
6117
6118  012750  012760  000076  000000          MOV     #ILF76,RMCS1(R0)        ;LOAD RMCS1
6119  012756  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6120  012764  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6121
6122  012770  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
6123  012776  042737  177701  001142          BIC     #^CILF76,$BDDAT
6124  013004  001410                          BEQ     5$
6125  013006  005037  001140                  CLR     $GDDAT
6126  013012  010037  001136                  MOV     R0,$BDADR
6127  013016  062737  000000  001136          ADD     #RMCS1,$BDADR
6128  013024  104043                          ERROR   43              ;CANT CLEAR RMCS1
6129  013026                          5$:
6130                                  ;WRITE ONES IN RMCS1, BITS 01-05, THEN WRITE ZEROS. READ AND CHECK FOR
6131                                  ;S-A-1 BITS.
6132
6133  013026  012760  000076  000000          MOV     #ILF76,RMCS1(R0)        ;LOAD RMCS1
6134
6135  013034  012760  000000  000000          MOV     #0,RMCS1(R0)    ;LOAD RMCS1
6136
6137  013042  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
6138  013050  042737  177701  001142          BIC     #^CILF76,$BDDAT
6139  013056  001412                          BEQ     10$
```

```
6140  013060  005037  001140                  CLR     $GDDAT
6141  013064  010037  001136                  MOV     R0,$BDADR
6142  013070  062737  000000  001136          ADD     #RMCS1,$BDADR
6143  013076  104023                          ERROR   23              ;CANT WRITE 0'S
6144  013100  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
6145  013104                          10$:
6146                                  ;WRITE ZEROS IN RMCS1, THEN ONES, READ AND CHECK S-A-0 BITS.
6147
6148  013104  012760  000000  000000          MOV     #0,RMCS1(R0)    ;LOAD RMCS1
6149
6150  013112  012760  000076  000000          MOV     #ILF76,RMCS1(R0)        ;LOAD RMCS1
6151
6152  013120  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
6153  013126  042737  177701  001142          BIC     #^CILF76,$BDDAT
6154  013134  012737  000076  001140          MOV     #ILF76,$GDDAT
6155  013142  023737  001140  001142          CMP     $GDDAT,$BDDAT
6156  013150  001410                          BEQ     20$
6157  013152  010037  001136                  MOV     R0,$BDADR
6158  013156  062737  000000  001136          ADD     #RMCS1,$BDADR
6159  013164  104024                          ERROR   24              ;CANT WRITE ONES
6160  013166  052703  000002                  BIS     #BIT1,R3        ;SET ERROR FLAG
6161  013172                          20$:
6162                                  ;WRITE A SHIFTING ONE BIT PATTERN IN RMCS1, READ AND CHECK FOR STUCK BITS.
6163  013172  005703                          TST     R3              ;OMIT IF ANY ERRORS
6164  013174  001035                          BNE     50$
6165  013176  012702  000002                  MOV     #2,R2           ;R2 = TEST PATTERN
6166  013202                          30$:
6167  013202  010203                          MOV     R2,R3           ;R3 = EXPECTED RESULT, BITS 1-5
6168  013204  042703  177701                  BIC     #^CILF76,R3
6169
6170  013210  012760  000000  000000          MOV     #0,RMCS1(R0)    ;LOAD RMCS1
6171
6172  013216  010260  000000                  MOV     R2,RMCS1(R0)    ;LOAD RMCS1
6173
6174  013222  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
6175  013230  042737  177701  001142          BIC     #^CILF76,$BDDAT
6176  013236  020337  001142                  CMP     R3,$BDDAT
6177  013242  001410                          BEQ     40$
6178  013244  010337  001140                  MOV     R3,$GDDAT
6179  013250  010037  001136                  MOV     R0,$BDADR
6180  013254  062737  000000  001136          ADD     #RMCS1,$BDADR
6181  013262  104025                          ERROR   25              ;CANT WRITE SHIFTING ONES
6182  013264  006302                  40$:    ASL     R2              ;SHIFT TO NEXT BIT
6183  013266  001345                          BNE     30$             ;CONTINUE IF R2 NOT ZERO
6184  013270                          50$:
6185
6186                                  ;;*******************************************************************
6187                                  ;*TEST 13        ERROR REGISTER 1 TRANSFER TEST
6188
6189                                  ;;*******************************************************************
6190  013270  000004                  TST13:  SCOPE
6191  013272  012737  000013  001226          MOV     #13,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
6192
6193  013300  000240                          NOP
6194  013302  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
6195  013310  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
```

```
6196  013316  012737  013332  001122          MOV     #T13,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6197  013324  012737  013332  001124          MOV     #T13,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6198  013332                                T13:
6199  013332  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
6200  013336  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6201  013342  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6202  013346  005003                           CLR     R3              ;CLEAR ERROR FLAG
6203                                        ;WRITE ONES IN RMER1, CLEAR AND CHECK FOR S-A-1 BITS
6204
6205  013350  012760  177777  000014          MOV     #-1,RMER1(R0)   ;LOAD RMER1
6206  013356  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6207  013364  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6208
6209  013370  016037  000014  001342          MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
6210  013376  013737  001342  001142          MOV     RMER1I,$BDDAT
6211  013404  042737  177760  001142          BIC     #^C<PAR!RMR!ILF!ILR>,$BDDAT
6212  013412  001410                          BEQ     10$
6213  013414  005037  001140                  CLR     $GDDAT
6214  013420  010037  001136                  MOV     R0,$BDADR
6215  013424  062737  000014  001136          ADD     #RMER1,$BDADR
6216  013432  104027                          ERROR   27              ;CANT CLEAR RMER1
6217  013434                                10$:
6218  013434  013737  001342  001142          MOV     RMER1I,$BDDAT
6219  013442  042737  074017  001142          BIC     #^C<DCK!IAE!AOE!HCRC!HCE!ECH!WCF!FER>,$BDDAT
6220  013450  001410                          BEQ     20$
6221  013452  005037  001140                  CLR     $GDDAT
6222  013456  010037  001136                  MOV     R0,$BDADR
6223  013462  062737  000014  001136          ADD     #RMER1,$BDADR
6224  013470  104030                          ERROR   30
6225  013472                                20$:
6226  013472  013737  001342  001142          MOV     RMER1I,$BDDAT
6227  013500  042737  147777  001142          BIC     #^C<OPI!DTE>,$BDDAT
6228  013506  001410                          BEQ     30$
6229  013510  005037  001140                  CLR     $GDDAT
6230  013514  010037  001136                  MOV     R0,$BDADR
6231  013520  062737  000014  001136          ADD     #RMER1,$BDADR
6232  013526  104031                          ERROR   31
6233  013530                                30$:
6234                                        ;WRITE ONES THEN ZEROS IN RMER1, READ AND CHECK FOR S-A-1 BITS
6235
6236  013530  012760  177777  000014          MOV     #-1,RMER1(R0)   ;LOAD RMER1
6237
6238  013536  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
6239
6240  013544  016037  000014  001342          MOV     RMER1(R0),RMER1I        ;STORE RMER1 IN INPUT BUFFER
6241  013552  013737  001342  001142          MOV     RMER1I,$BDDAT
6242  013560  042737  177770  001142          BIC     #^C<RMR!ILF!ILR>,$BDDAT
6243  013566  001412                          BEQ     40$
6244  013570  005037  001140                  CLR     $GDDAT
6245  013574  010037  001136                  MOV     R0,$BDADR
6246  013600  062737  000014  001136          ADD     #RMER1,$BDADR
6247  013606  104032                          ERROR   32
6248  013610  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
6249  013614  013737  001342  001142  40$:     MOV     RMER1I,$BDDAT
6250  013622  042737  074017  001142          BIC     #^C<DCK!IAE!AOE!HCRC!HCE!ECH!WCF!FER>,$BDDAT
6251  013630  001412                          BEQ     50$
```

```
6252  013632  005037  001140                  CLR     $GDDAT
6253  013636  010037  001136                  MOV     R0,$BDADR
6254  013642  062737  000014  001136          ADD     #RMER1,$BDADR
6255  013650  104033                          ERROR   33
6256  013652  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
6257  013656                          50$:
6258  013656  013737  001342  001142          MOV     RMER1I,$BDDAT
6259  013664  042737  147777  001142          BIC     #^C<OPI!DTE>,$BDDAT
6260  013672  001412                          BEQ     60$
6261  013674  005037  001140                  CLR     $GDDAT
6262  013700  010037  001136                  MOV     R0,$BDADR
6263  013704  062737  000014  001136          ADD     #RMER1,$BDADR
6264  013712  104034                          ERROR   34
6265  013714  052703  030000                  BIS     #BIT,R3
6266  013720                          60$:
6267                                  ;WRITE ZEROS THEN ONES IN RMER1,READ AND CHECK FOR S-A-0 BITS
6268
6269  013720  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
6270
6271  013726  012760  177777  000014          MOV     #-1,RMER1(R0)   ;LOAD RMER1
6272
6273  013734  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
6274  013742  012737  177777  001140          MOV     #-1,$GDDAT
6275  013750  023737  001140  001142          CMP     $GDDAT,$BDDAT
6276  013756  001410                          BEQ     70$
6277  013760  010037  001136                  MOV     R0,$BDADR
6278  013764  062737  000014  001136          ADD     #RMER1,$BDADR
6279  013772  104035                          ERROR   35
6280  013774  052703  000002                  BIS     #BIT1,R3
6281  014000                          70$:
6282                                  ;WRITE A SHIFTING 1 BIT IN RMER1 AND CHECK FOR STUCK BITS
6283                                  ;NOTE:  DONT TEST UNSAFE OR PARITY
6284  014000  005703                          TST     R3              ;SKIP THIS PART IF ANY ERRORS
6285  014002  001045                          BNE     120$
6286  014004  012702  000001                  MOV     #1,R2           ;R2 = TEST PATTERN
6287  014010                          80$:
6288  014010  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6289  014016  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6290
6291  014022  010260  000014                  MOV     R2,RMER1(R0)    ;LOAD RMER1
6292
6293  014026  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
6294  014034  032702  000010                  BIT     #PAR,R2         ;DONT TEST PAR = 0
6295  014040  001003                          BNE     90$
6296  014042  042737  000010  001142          BIC     #PAR,$BDDAT
6297  014050  032702  040000          90$:    BIT     #UNS,R2         ;DONT TEST UNS = 0
6298  014054  001003                          BNE     100$
6299  014056  042737  040000  001142          BIC     #UNS,$BDDAT
6300  014064  020237  001142          100$:   CMP     R2,$BDDAT
6301  014070  001410                          BEQ     110$
6302  014072  010237  001140                  MOV     R2,$GDDAT
6303  014076  010037  001136                  MOV     R0,$BDADR
6304  014102  062737  000014  001136          ADD     #RMER1,$BDADR
6305  014110  104036                          ERROR   36
6306
6307  014112  006302                  110$:   ASL     R2              ;SHIFT TO NEXT BIT
```

```
6308  014114  001335                          BNE     80$             ;CONTINUE IF R2 NOT ZERO
6309  014116                          120$:
6310
6311
6312                                  ;;*****************************************************************
6313                                  ;*TEST 14        ERROR REGISTER 2 TRANSFER TEST
6314
6315                                  ;;*****************************************************************
6316  014116  000004                  TST14:  SCOPE
6317  014120  012737  000014  001226          MOV     #14,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
6318
6319  014126  000240                          NOP
6320  014130  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
6321  014136  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
6322  014144  012737  014160  001122          MOV     #T14,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6323  014152  012737  014160  001124          MOV     #T14,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6324  014160                          T14:
6325  014160  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
6326  014164  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6327  014170  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6328  014174  005003                          CLR     R3              ;RESET ERROR FLAGS
6329  014176  010037  001136                  MOV     R0,$BDADR       ;SETUP BAD ADDRESS
6330  014202  062737  000042  001136          ADD     #RMER2,$BDADR
6331  014210  005037  001140                  CLR     $GDDAT          ;SETUP EXPECTED DATA
6332                                  ;;*****************************************************************
6333                                  ;WRITE ONES IN RMER2, CLEAR AND CHECK FOR S-A-1 BITS
6334
6335  014214  012760  177777  000042          MOV     #-1,RMER2(R0)   ;LOAD RMER2
6336  014222  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6337  014230  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6338
6339  014234  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
6340
6341                                  ;       TEST UNUSED BITS FOR ZERO-FAILURE ON IF
6342  014242  013737  001370  001142          MOV     RMER2I,$BDDAT
6343  014250  042737  176210  001142          BIC     #^CXNUER2,$BDDAT
6344  014256  001403                          BEQ     10$             ;BRANCH IF NO ERROR
6345  014260  104044                          ERROR   44              ;UNUSED BITS NOT ZERO
6346  014262  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
6347  014266                          10$:
6348                                  ;       TEST 'OPE', 'IVC', 'LSC' FOR ZERO-FAILURE ON CS, IF
6349  014266  013737  001370  001142          MOV     RMER2I,$BDDAT
6350  014274  042737  143777  001142          BIC     #^C<OPE!IVC!LSC>,$BDDAT
6351  014302  001403                          BEQ     20$             ;BRANCH IF NO ERROR
6352  014304  104045                          ERROR   45              ;CANT CLEAR OPE, IVC, LSC
6353  014306  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
6354  014312                          20$:
6355                                  ;       TEST 'LBC', 'DPE' FOR ZERO-FAILURE ON DS, IF
6356  014312  013737  001370  001142          MOV     RMER2I,$BDDAT
6357  014320  042737  175767  001142          BIC     #^C<LBC!DPE>,$BDDAT
6358  014326  001403                          BEQ     30$             ;BRANCH IF NO ERROR
6359  014330  104046                          ERROR   46              ;CANT CLEAR LBC, DPE
6360  014332  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
6361  014336                          30$:
6362
6363                                  ;;*****************************************************************
```

```
6364                                    ;WRITE ONES IN RMER2 THEN WRITE ZEROS AND CHECK FOR S-A-1 BITS.
6365
6366  014336  012760  177777  000042            MOV     #-1,RMER2(R0)   ;LOAD RMER2
6367
6368  014344  012760  000000  000042            MOV     #0,RMER2(R0)    ;LOAD RMER2
6369
6370  014352  016037  000042  001370            MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
6371
6372                                    ;       TEST ''OPE'', ''IVC'', ''LSC'' FOR ZERO-FAILURE ON CS, IF
6373  014360  013737  001370  001142            MOV     RMER2I,$BDDAT
6374  014366  042737  143777  001142            BIC     #^C<OPE!IVC!LSC>,$BDDAT
```

```
 6375  014374  001403                    BEQ     40$             ;BRANCH IF NO ERROR
```

```
 6376  014376  104047                          ERROR   47              ;CANT WRITE ZEROS
 6377  014400  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
 6378  014404                            40$:
 6379                                    ;       TEST 'LBC', 'DPE' FOR ZERO-FAILURE ON DS, IF
 6380  014404  013737  001370  001142          MOV     RMER2I,$BDDAT
```

```
6381  014412  042737  175767  001142          BIC     #^C<LBC!DPE>,$BDDAT
6382  014420  001403                          BEQ     50$             ;BRANCH IF NO ERROR
6383  014422  104050                          ERROR   50              ;CANT WRITE ZEROS
6384  014424  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
6385  014430                          50$:
6386
6387                                  ;;*******************************************************************
6388                                  ;WRITE ZEROS IN RMER2 THEN WRITE ONES AND CHECK FOR S-A-0 BITS.
6389
6390  014430  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
6391
6392  014436  012760  177777  000042          MOV     #-1,RMER2(R0)   ;LOAD RMER2
6393
6394  014444  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
6395                          ;               TEST UNUSED BITS FOR ZERO-FAILURE ON IF
6396  014452  013737  001370  001142          MOV     RMER2I,$BDDAT
6397  014460  042737  176210  001142          BIC     #^CXNUER2,$BDDAT
6398  014466  001403                          BEQ     60$             ;BRANCH IF NO ERROR
6399  014470  104044                          ERROR   44              ;UNUSED BITS NOT ZERO
6400  014472  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
6401  014476                          60$:
6402                          ;               TEST USED BITS FOR ONE-FAILURE ON IF
6403  014476  013737  001370  001142          MOV     RMER2I,$BDDAT
6404  014504  042737  001567  001142          BIC     #XNUER2,$BDDAT
6405  014512  012737  176210  001140          MOV     #^CXNUER2,$GDDAT
6406  014520  023737  001140  001142          CMP     $GDDAT,$BDDAT
6407  014526  001403                          BEQ     70$             ;BRANCH IF NO ERROR
6408  014530  104051                          ERROR   51              ;CANT WRITE ONES
6409  014532  052703  000002                  BIS     #BIT1,R3        ;SET ERROR FLAG
6410  014536                          70$:
6411
6412                                  ;;*******************************************************************
6413                                  ;IF NO PREVIOUS ERROR, TEST BIT INTERFERENCE WITH SHIFTING ONE BIT.
6414  014536  005703                          TST     R3              ;ANY ERRORS?
6415  014540  001044                          BNE     120$            ;YES!!
6416  014542  012702  000001                  MOV     #1,R2           ;R2=DATA PATTERN
6417  014546                          80$:
6418  014546  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6419  014554  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6420
6421  014560  010260  000042                  MOV     R2,RMER2(R0)    ;LOAD RMER2
6422
6423  014564  016037  000042  001142          MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
6424  014572  032702  040000                  BIT     #SKI,R2         ;IS SKI BEING SET?
6425  014576  001003                          BNE     90$             ;YES!!
6426  014600  042737  040000  001142          BIC     #SKI,$BDDAT     ;DONT TEST SKI FOR ZERO
6427  014606  032702  000200          90$:    BIT     #DVC,R2         ;IS DVC BEING SET??
6428  014612  001003                          BNE     100$            ;YES!!
6429  014614  042737  000200  001142          BIC     #DVC,$BDDAT     ;DONT TEST DVC FOR ZERO
6430  014622  010237  001140          100$:   MOV     R2,$GDDAT
6431  014626  042737  001567  001140          BIC     #XNUER2,$GDDAT  ;UNUSED BITS SHOULD BE ZERO
6432  014634  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;ANY ERRORS??
6433  014642  001401                          BEQ     110$            ;NO!!
6434  014644  104052                          ERROR   52              ;CANT WRITE SHIFTING ONES
6435
6436  014646  006302                  110$:   ASL     R2              ;SHIFT TO NEXT DATA BIT
```

```
6437  014650  001336                          BNE     80$             ;CONTINUE IF NOT DONE
6438
6439  014652                          120$:
6440
6441                                  ;;**********************************************************************
6442                                  ;*TEST 15        CLEAR OFFSET STUCK ACTIVE TEST
6443
6444                                  ;;**********************************************************************
6445  014652  000004                  TST15:  SCOPE
6446  014654  012737  000015  001226          MOV     #15,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
6447
6448  014662  000240                          NOP
6449  014664  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
6450  014672  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
6451  014700  012737  014714  001122          MOV     #T15,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6452  014706  012737  014714  001124          MOV     #T15,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6453  014714                          T15:
6454  014714  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
6455  014720  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6456  014724  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6457  014730  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6458  014736  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6459
6460  014742  012760  177777  000032          MOV     #-1,RMOF(R0)    ;LOAD RMOF
6461
6462  014750  016037  000032  001142          MOV     RMOF(R0),$BDDAT ;STORE RMOF AT $BDDAT
6463  014756  042737  177577  001142          BIC     #^COFD,$BDDAT
6464  014764  001011                          BNE     10$             ;BRANCH IF OFD IS A ONE
6465  014766  012737  000200  001140          MOV     #OFD,$GDDAT     ;SETUP ERROR MESSAGE
6466  014774  010037  001136                  MOV     R0,$BDADR
6467  015000  062737  000032  001136          ADD     #RMOF,$BDADR
6468  015006  104172                          ERROR   172             ;CANT SET OFD BIT
6469  015010                          10$:                            ;END OF TEST
6470
6471
6472                                  ;;**********************************************************************
6473                                  ;*TEST 16        OFFSET REGISTER TRANSFER TEST
6474
6475                                  ;;**********************************************************************
6476  015010  000004                  TST16:  SCOPE
6477  015012  012737  000016  001226          MOV     #16,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
6478
6479  015020  000240                          NOP
6480  015022  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
6481  015030  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
6482  015036  012737  015052  001122          MOV     #T16,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6483  015044  012737  015052  001124          MOV     #T16,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6484  015052                          T16:
6485  015052  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
6486  015056  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6487  015062  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6488  015066  005003                          CLR     R3              ;RESET ERROR FLAGS
6489  015070  010037  001136                  MOV     R0,$BDADR       ;SETUP BAD ADDRESS
6490  015074  062737  000032  001136          ADD     #RMOF,$BDADR
6491  015102  005037  001140                  CLR     $GDDAT          ;SETUP EXPECTED DATA
6492                                  ;;**********************************************************************
```

```
6493                                    ;WRITE ONES THEN ZEROS IN RMOF AND CHECK FOR S-A-1 BITS.
6494
6495  015106  012760  177777  000032            MOV     #-1,RMOF(R0)    ;LOAD RMOF
6496
6497  015114  012760  000000  000032            MOV     #0,RMOF(R0)     ;LOAD RMOF
6498
6499  015122  016037  000032  001360            MOV     RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
6500                                    ;       CHECK UNUSED BITS OF RMOF FOR ZERO
6501  015130  013737  001360  001142            MOV     RMOFI,$BDDAT    ;GET UNUSED BITS
6502  015136  042737  016200  001142            BIC     #^CXNUOF,$BDDAT
6503  015144  001403                            BEQ     10$             ;BRANCH IF NO ERROR
6504  015146  104053                            ERROR   53              ;UNUSED BITS NOT ZERO RMOF
6505  015150  052703  000001                    BIS     #BIT0,R3        ;SET ERROR FLAG
6506  015154                                10$:
6507                                    ;       CHECK USED BITS OF RMOF FOR ZERO
6508  015154  013737  001360  001142            MOV     RMOFI,$BDDAT    ;GET USED BITS
6509  015162  042737  161577  001142            BIC     #XNUOF,$BDDAT
6510  015170  001403                            BEQ     20$             ;BRANCH IF NO ERROR
6511  015172  104054                            ERROR   54              ;USED BITS RMOF S-A-1
6512  015174  052703  000001                    BIS     #BIT0,R3        ;SET ERROR FLAG
6513  015200                                20$:
6514
6515                                    ;;*****************************************************************
6516                                    ;WRITE ZEROS THEN ONES ON RMOF AND CHECK FOR S-A-0 BITS.
6517
6518  015200  012760  000000  000032            MOV     #0,RMOF(R0)     ;LOAD RMOF
6519
6520  015206  012760  177777  000032            MOV     #-1,RMOF(R0)    ;LOAD RMOF
6521
6522  015214  016037  000032  001360            MOV     RMOF(R0),RMOFI  ;STORE RMOF IN INPUT BUFFER
6523                                    ;       CHECK UNUSED BITS OF RMOF FOR ZERO.
6524  015222  013737  001360  001142            MOV     RMOFI,$BDDAT    ;GET UNUSED BITS
6525  015230  042737  016200  001142            BIC     #^CXNUOF,$BDDAT
6526  015236  001403                            BEQ     30$             ;BRANCH IF NO ERROR
6527  015240  104053                            ERROR   53              ;UNUSED BITS OF RMOF NOT ZERO
6528  015242  052703  000001                    BIS     #BIT0,R3
6529  015246                                30$:
6530                                    ;       CHECK USED BITS OF RMOF FOR ONE.
6531  015246  013737  001360  001142            MOV     RMOFI,$BDDAT    ;GET USED BITS
6532  015254  042737  161577  001142            BIC     #XNUOF,$BDDAT
6533  015262  012737  016200  001140            MOV     #FMT16!ECI!HCI!OFD,$GDDAT
6534  015270  023737  001140  001142            CMP     $GDDAT,$BDDAT
6535  015276  001403                            BEQ     40$             ;BRANCH IF NO ERROR
6536  015300  104055                            ERROR   55              ;USED BITS RMOF S-A-0
6537  015302  052703  000002                    BIS     #BIT1,R3
6538  015306                                40$:
6539
6540                                    ;;*****************************************************************
6541                                    ;IF NO PREVIOUS ERRORS, TEST RMOF WITH SHIFTING ONE BIT PATTERN.
6542  015306  005703                            TST     R3              ;ANY ERROR??
6543  015310  001025                            BNE     70$             ;YES!!
6544  015312  012702  000001                    MOV     #1,R2           ;STARTING DATA PATTERN
6545  015316                                50$:
6546
6547  015316  012760  000000  000032            MOV     #0,RMOF(R0)     ;LOAD RMOF
6548
```

```
6549  015324  010260  000032                  MOV     R2,RMOF(R0)       ;LOAD RMOF
6550
6551  015330  016037  000032  001142          MOV     RMOF(R0),$BDDAT ;STORE RMOF AT $BDDAT
6552  015336  010203                          MOV     R2,R3             ;SETUP EXPECTED RESULT
6553  015340  042703  161577                  BIC     #XNUOF,R3         ;CLEAR UNUSED BITS
6554  015344  020337  001142                  CMP     R3,$BDDAT         ;COMPARE EXPECTED & RECEIVED
6555  015350  001403                          BEQ     60$               ;BRANCH IF NO ERROR
6556  015352  010337  001140                  MOV     R3,$GDDAT         ;LOAD EXPECTED
6557  015356  104056                          ERROR   56                ;CANT WRITE SHIFTING 1 BIT
6558  015360  006302                  60$:    ASL     R2                ;SHIFT TO NEXT BIT
6559  015362  001355                          BNE     50$               ;CONTINUE IF NOT DONE
6560
6561  015364                          70$:
6562
6563                                  ;;*****************************************************************
6564                                  ;*TEST 17        SERIAL NUMBER TEST
6565
6566                                  ;;*****************************************************************
6567  015364  000004                  TST17:  SCOPE
6568  015366  012737  000017  001226          MOV     #17,$TESTN        ;;SET TEST NUMBER IN APT MAIL BOX
6569
6570  015374  000240                          NOP
6571  015376  012737  000024  001120          MOV     #20.,$ICNT        ;20 ITERATIONS
6572  015404  112737  000001  001131          MOVB    #1,$ERMAX         ;ONE ERROR ALLOWED
6573  015412  012737  015426  001122          MOV     #T17,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6574  015420  012737  015426  001124          MOV     #T17,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6575  015426                          T17:
6576  015426  012706  001100                  MOV     #STACK,SP         ;LOAD THE STACK POINTER
6577  015432  013700  001276                  MOV     $BASE,R0          ;R0 = UNIBUS ADDRESS OF UUT
6578  015436  013701  001456                  MOV     TSTQUE,R1         ;R1 = POINTER TO DEVICE
6579  015442  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6580  015450  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6581  015454  010037  001136                  MOV     R0,$BDADR         ;SETUP REGISTER ADDRESS FOR TYPEOUT
6582  015460  062737  000030  001136          ADD     #RMSN,$BDADR
6583  015466  012702  000031                  MOV     #25.,R2           ;READ RMSN 25 TIMES
6584                                  ;READ RMSN AND USE THE RESULT AS EXPECTED VALUE
6585
6586  015472  016037  000030  001140          MOV     RMSN(R0),$GDDAT ;STORE RMSN AT $GDDAT
6587  015500                          10$:
6588                                  ;READ RMSN AND COMPARE WITH INITIAL VALUE
6589
6590  015500  016037  000030  001142          MOV     RMSN(R0),$BDDAT ;STORE RMSN AT $BDDAT
6591  015506  023737  001140  001142          CMP     $GDDAT,$BDDAT
6592  015514  001401                          BEQ     20$               ;BRANCH IF SERIAL NUMBER CONSISTENT
6593  015516  104136                          ERROR   136               ;SERIAL NUMBER INCONSISTENT
6594  015520                          20$:
6595                                  ;DECREMENT COUNT AND CONTINUE IF NOT DONE
6596  015520  005302                          DEC     R2
6597  015522  100366                          BPL     10$
6598  015524                          30$:                              ;END OF TEST
6599
6600                                  ;;*****************************************************************
6601                                  ;*TEST 20        CONTROL BUS PARITY DETECTION TEST
6602
6603                                  ;;*****************************************************************
6604  015524  000004                  TST20:  SCOPE
```

```
6605  015526  012737  000020  001226          MOV     #20,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
6606
6607  015534  000240                          NOP
6608  015536  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
6609  015544  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
6610  015552  012737  015566  001122          MOV     #T20,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6611  015560  012737  015566  001124          MOV     #T20,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6612  015566                          T20:
6613  015566  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
6614  015572  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6615  015576  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6616                          ;SETUP FOR FIRST TEST LOOP (NO ERROR)
6617  015602  005037  001140                  CLR     $GDDAT          ;'PAR' SHOULD BE ZERO
6618  015606  111137  001412                  MOVB    (R1),RMCS20     ;SETUP RMCS2 VALUE
6619  015612  042737  177770  001412          BIC     #^CUNTMSK,RMCS20
6620  015620  012737  000001  001440          MOV     #1,RMHR0        ;INITIALIZE DATA PATTERN
6621  015626  000402                          BR      6$              ;SKIP INCREMENT FIRST TIME
6622  015630  006337  001440          5$:     ASL     RMHR0           ;SHIFT TO NEXT PATTERN
6623  015634                          6$:
6624                          ;CLEAR AND VERIFY THAT 'PAR' IS RESET
6625  015634  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6626  015642  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6627
6628  015646  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
6629
6630  015654  016037  000042  001370          MOV     RMER2(R0),RMER2I        ;STORE RMER2 IN INPUT BUFFER
6631  015662  042737  177767  001142          BIC     #^CPAR,$BDDAT   ;DID 'PAR' RESET?
6632  015670  001415                          BEQ     20$             ;YES!!
6633  015672  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
6634  015676  062737  000014  001136          ADD     #RMER1,$BDADR
6635  015704  032737  000010  001370          BIT     #DPE,RMER2I     ;IS 'DPE' SET??
6636  015712  001002                          BNE     10$             ;YES!!
6637  015714  104067                          ERROR   67              ;CANT CLEAR 'PAR'
6638  015716  000453                          BR      50$
6639  015720  104070          10$:    ERROR   70              ;CANT CLEAR 'PAR', 'DPE'
6640  015722  000451                          BR      50$
6641  015724                          20$:
6642
6643                          ;WRITE TEST PATTERN AND VERIFY 'PAR' STATUS
6644
6645  015724  013760  001412  000010          MOV     RMCS20,RMCS2(R0)        ;LOAD RMCS2
6646
6647  015732  013760  001440  000036          MOV     RMHR0,RMHR(R0)  ;LOAD RMHR
6648
6649  015740  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
6650  015746  042737  177767  001142          BIC     #^CPAR,$BDDAT
6651  015754  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IS 'PAR' CORRECT??
6652  015762  001410                          BEQ     40$             ;YES!!
6653  015764  032737  000020  001412          BIT     #PAT,RMCS20     ;SHOULD 'PAR' BE SET?
6654  015772  001002                          BNE     30$             ;YES!!
6655  015774  104071                          ERROR   71              ;'PAR' SHOULD NOT BE SET
6656  015776  000423                          BR      50$
6657  016000  104072          30$:    ERROR   72              ;'PAR' SHOULD BE SET
6658  016002  000421                          BR      50$             ;SKIP TO NEXT
6659                          ;GO TO NEXT PATTERN
6660  016004  005737  001440          40$:    TST     RMHR0           ;IS DATA PATTERN COMPLETE??
```

```
6661  016010  001307                          BNE     5$                  ;NO!!
6662  016012  032737  000020  001412          BIT     #PAT,RMCS20         ;IS TEST COMPLETE??
6663  016020  001012                          BNE     50$                 ;YES!!
6664                                ;SETUP FOR SECOND TEST LOOP (ERROR)
6665  016022  052737  000020  001412          BIS     #PAT,RMCS20         ;TURN ON BAD PARITY
6666  016030  012737  000010  001140          MOV     #PAR,$GDDAT         ;EXPECT ERROR
6667  016036  012737  000001  001440          MOV     #1,RMHRO            ;INITIALIZE DATA PATTERN
6668  016044  000673                          BR      6$                  ;START LOOP
6669
6670  016046                              50$:                            ;END OF TEST
6671
6672                                ;;*****************************************************************
6673                                ;*TEST 21        CONTROL BUS PARITY GENERATION TEST
6674
6675                                ;;*****************************************************************
6676  016046  000004                    TST21:  SCOPE
6677  016050  012737  000021  001226          MOV     #21,$TESTN          ;;SET TEST NUMBER IN APT MAIL BOX
6678
6679  016056  000240                          NOP
6680  016060  012737  000024  001120          MOV     #20.,$ICNT          ;20 ITERATIONS
6681  016066  112737  000001  001131          MOVB    #1,$ERMAX           ;ONE ERROR ALLOWED
6682  016074  012737  016110  001122          MOV     #T21,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6683  016102  012737  016110  001124          MOV     #T21,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6684  016110                              T21:
6685  016110  012706  001100                  MOV     #STACK,SP           ;LOAD THE STACK POINTER
6686  016114  013700  001276                  MOV     $BASE,R0            ;R0 = UNIBUS ADDRESS OF UUT
6687  016120  013701  001456                  MOV     TSTQUE,R1           ;R1 = POINTER TO DEVICE
6688
6689                                ;SETUP FOR TEST (NO ERROR)
6690  016124  012737  000001  001440          MOV     #1,RMHRO            ;INITIALIZE DATA PATTERN
6691  016132  005037  001140                  CLR     $GDDAT              ;MCPE SHOULD BE ZERO
6692  016136  000402                          BR      20$                 ;DONT SHIFT FIRST TIME
6693  016140                              10$:
6694                                ;SHIFT DATA PATTERN
6695  016140  006337  001440                  ASL     RMHRO
6696  016144                              20$:
6697                                ;CLEAR, THEN TRANSFER DATA TO RM03
6698  016144  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6699  016152  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6700
6701  016156  013760  001440  000036          MOV     RMHRO,RMHR(R0)  ;LOAD RMHR
6702                                ;TRANSFER DATA TO RH, VERIFY NO 'MCPE'' ERROR
6703
6704  016164  016037  000036  001364          MOV     RMHR(R0),RMHRI  ;STORE RMHR IN INPUT BUFFER
6705
6706  016172  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
6707  016200  042737  157777  001142          BIC     #^CMCPE,$BDDAT  ;WAS BAD PARITY DETECTED??
6708  016206  001402                          BEQ     30$                 ;NO!!
6709  016210  104073                          ERROR   73                  ;BAD PARITY DETECTED BY RH
6710  016212  000403                          BR      40$
6711
6712  016214                              30$:
6713                                ;GO TO NEXT PATTERN
6714  016214  005737  001440                  TST     RMHRO               ;DONE ALL PATTERNS??
6715  016220  001347                          BNE     10$                 ;NO!!
6716
```

```
6717  016222                                40$:                                ;END OF TEST
6718
6719                                        ;;******************************************************************
6720                                        ;*TEST 22       RMDA,RMDC FAULT TEST
6721
6722                                        ;;******************************************************************
6723  016222  000004                        TST22:  SCOPE
6724  016224  012737  000022  001226                MOV     #22,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
6725
6726  016232  000240                                NOP
6727  016234  012737  000024  001120                MOV     #20.,$ICNT      ;20 ITERATIONS
6728  016242  112737  000001  001131                MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
6729  016250  012737  016264  001122                MOV     #T22,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6730  016256  012737  016264  001124                MOV     #T22,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6731  016264                                T22:
6732  016264  012706  001100                        MOV     #STACK,SP       ;LOAD THE STACK POINTER
6733  016270  013700  001276                        MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6734  016274  013701  001456                        MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6735  016300  012760  000040  000010                MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6736  016306  111160  000010                        MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6737
6738                                        ;WRITE ZEROS, THEN ONES IN RMDA,RMDC-READ AND TEST FOR S-A-0
6739
6740  016312  012760  000000  000006                MOV     #0,RMDA(R0)     ;LOAD RMDA
6741
6742  016320  012760  000000  000034                MOV     #0,RMDC(R0)     ;LOAD RMDC
6743
6744  016326  012760  177777  000006                MOV     #-1,RMDA(R0)    ;LOAD RMDA
6745
6746  016334  012760  177777  000034                MOV     #-1,RMDC(R0)    ;LOAD RMDC
6747
6748  016342  016037  000006  001334                MOV     RMDA(R0),RMDAI  ;STORE RMDA IN INPUT BUFFER
6749
6750  016350  016037  000034  001362                MOV     RMDC(R0),RMDCI  ;STORE RMDC IN INPUT BUFFER
6751  016356  022737  177777  001334                CMP     #-1,RMDAI       ;IS ANY REGISTER ALL ONES??
6752  016364  001410                                BEQ     10$             ;YES!!
6753  016366  052737  176000  001362                BIS     #XNUDC,RMDCI    ;SET UNUSED BITS
6754  016374  022737  177777  001362                CMP     #-1,RMDCI
6755  016402  001401                                BEQ     10$             ;YES!!
6756  016404  104042                                ERROR   42              ;RMDC AND RMDA S-A-0
6757
6758  016406                                10$:
6759
6760
6761
6762                                        ;;******************************************************************
6763                                        ;*TEST 23       DISK ADDRESS TRANSFER TEST
6764
6765                                        ;;******************************************************************
6766  016406  000004                        TST23:  SCOPE
6767  016410  012737  000023  001226                MOV     #23,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
6768  016416  000240                                NOP
6769  016420  012737  000024  001120                MOV     #20.,$ICNT      ;20 ITERATIONS
6770  016426  112737  000001  001131                MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
6771  016434  012737  016450  001122                MOV     #T23,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6772  016442  012737  016450  001124                MOV     #T23,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
```

```
6773  016450                                T23:
6774  016450  012706  001100                      MOV     #STACK,SP       ;LOAD THE STACK POINTER
6775  016454  013700  001276                      MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6776  016460  013701  001456                      MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6777  016464  005003                            CLR     R3              ;R3 = ERROR INDICATOR
6778  016466  012760  000040  000010            MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6779  016474  111160  000010                    MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6780                                    ;WRITE ONES IN RMDA, THEN WRITE ZEROS, READ BACK AND CHECK FOR S-A-1 BITS
6781
6782  016500  012760  177777  000006            MOV     #-1,RMDA(R0)    ;LOAD RMDA
6783
6784  016506  012760  000000  000006            MOV     #0,RMDA(R0)     ;LOAD RMDA
6785
6786  016514  016037  000006  001142            MOV     RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
6787  016522  005737  001142                    TST     $BDDAT
6788  016526  001412                            BEQ     10$
6789  016530  005037  001140                    CLR     $GDDAT
6790  016534  010037  001136                    MOV     R0,$BDADR
6791  016540  062737  000006  001136            ADD     #RMDA,$BDADR
6792  016546  104020                            ERROR   20
6793  016550  052703  000001                    BIS     #BIT0,R3        ;SET ERROR FLAG
6794                                    ;WRITE ZEROS IN RMDA, THEN WRITE ONES, READ BACK AND CHECK FOR S-A-0 BITS
6795  016554                                10$:
6796
6797  016554  012760  000000  000006            MOV     #0,RMDA(R0)     ;LOAD RMDA
6798
6799  016562  012760  177777  000006            MOV     #-1,RMDA(R0)    ;LOAD RMDA
6800
6801  016570  016037  000006  001142            MOV     RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
6802  016576  023727  001142  177777            CMP     $BDDAT,#-1
6803  016604  001413                            BEQ     20$
6804  016606  012737  177777  001140            MOV     #-1,$GDDAT
6805  016614  010037  001136                    MOV     R0,$BDADR
6806  016620  062737  000006  001136            ADD     #RMDA,$BDADR
6807  016626  104021                            ERROR   21
6808  016630  052703  000002                    BIS     #BIT1,R3        ;SET ERROR FLAG
6809                                    ;WRITE A SHIFTING 1 BIT PATTERN IN RMDA, READ, AND CHECK FOR STUCK BITS
6810  016634  005703                    20$:    TST     R3              ;OMIT BIT TEST IF ANY ERROR
6811  016636  001027                            BNE     50$
6812  016640  012702  000001                    MOV     #1,R2           ;R2 = TEST PATTERN
6813  016644                                30$:
6814
6815  016644  012760  000000  000006            MOV     #0,RMDA(R0)     ;LOAD RMDA
6816
6817  016652  010260  000006                    MOV     R2,RMDA(R0)     ;LOAD RMDA
6818
6819  016656  016037  000006  001142            MOV     RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
6820  016664  023702  001142                    CMP     $BDDAT,R2
6821  016670  001410                            BEQ     40$
6822  016672  010237  001140                    MOV     R2,$GDDAT
6823  016676  010037  001136                    MOV     R0,$BDADR
6824  016702  062737  000006  001136            ADD     #RMDA,$BDADR
6825  016710  104022                            ERROR   22
6826  016712  006302                    40$:    ASL     R2              ;SHIFT TO NEXT BIT
6827  016714  001353                            BNE     30$             ;CONTINUE IF R2 NOT ZERO
6828  016716                                50$:
```

```
                                        ;;**********  ********************************************
                                        ;*TEST 24        DESIRED CYLINDER TRANSFER TEST
                                        ;;**************************************************************
016716  000004                  TST24:  SCOPE
016720  012737  000024  001226          MOV     #24,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
016726  000240                          NOP
016730  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
016736  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
016744  012737  016760  001122          MOV     #T24,$LPADR  ;LOAD LOOP ON TEST ADDRESS
016752  012737  016760  001124          MOV     #T24,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
016760                          T24:
016760  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
016764  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
016770  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
016774  005003                          CLR     R3              ;RESET ERROR FLAGS
016776  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
017004  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
017010  005037  001140                  CLR     $GDDAT          ;LOAD EXPECTED
017014  010037  001136                  MOV     R0,$BDADR       ;LOAD REG ADDRESS
017020  062737  000034  001136          ADD     #RMDC,$BDADR

                                        ;WRITE ONES IN RMDC AND VERIFY THAT UNUSED BITS ARE ZERO

017026  012760  177777  000034          MOV     #-1,RMDC(R0)    ;LOAD RMDC

017034  016037  000034  001142          MOV     RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
017042  042737  001777  001142          BIC     #^CXNUDC,$BDDAT         ;CLEAR ALL USED BITS
017050  001403                          BEQ     5$                      ;BRANCH IF NO ERROR
017052  104134                          ERROR   134             ;UNUSED BITS OF RMDC NOT ZERO
017054  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
017060                          5$:
                                        ;WRITE ONES IN RMDC, THEN WRITE ZEROS, READ AND CHECK FOR S-A-1 BITS

017060  012760  177777  000034          MOV     #-1,RMDC(R0)    ;LOAD RMDC

017066  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC

017074  016037  000034  001142          MOV     RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
017102  042737  176000  001142          BIC     #XNUDC,$BDDAT   ;CLEAR UNUSED BITS
017110  001403                          BEQ     10$             ;BRANCH IF NO ERROR
017112  104037                          ERROR   37              ;CANT WRITE ONES RMDC
017114  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
017120                          10$:
                                        ;WRITE ZEROS, THEN ONES IN RMDC, READ AND CHECK FOR S-A-0 BITS

017120  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC

017126  012760  177777  000034          MOV     #-1,RMDC(R0)    ;LOAD RMDC

017134  016037  000034  001142          MOV     RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
017142  052737  176000  001142          BIS     #XNUDC,$BDDAT   ;SET UNUSED BITS
017150  012737  177777  001140          MOV     #-1,$GDDAT      ;LOAD EXPECTED RESULT
017156  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IS RMDC ALL ONES ??
```

```
6885  017164  001403                          BEQ     20$             ;YES !!
6886  017166  104040                          ERROR   40
6887  017170  052703  000002                  BIS     #BIT1,R3        ;SET ERROR FLAG
6888  017174                          20$:
6889                                  ;OMIT BIT TEST IF ANY ERRORS
6890  017174  005703                          TST     R3
6891  017176  001026                          BNE     60$
6892  017200  012702  000001                  MOV     #1,R2           ;R2 = TEST PATTERN
6893  017204                          30$:
6894                                  ;TEST RMDC WITH SHIFTING ONE BIT
6895
6896  017204  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
6897
6898  017212  010260  000034                  MOV     R2,RMDC(R0)     ;LOAD RMDC
6899  017216  010203                          MOV     R2,R3           ;R3 = EXPECTED RESULT
6900  017220  042703  176000                  BIC     #XNUDC,R3       ;CLEAR ANY UNUSED BITS
6901
6902  017224  016037  000034  001142          MOV     RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
6903  017232  023703  001142                  CMP     $BDDAT,R3
6904  017236  001404                          BEQ     50$
6905  017240  010337  001140                  MOV     R3,$GDDAT
6906  017244  104041                          ERROR   41
6907  017246  000402                          BR      60$             ;SKIP TO NEXT
6908
6909  017250  006302                  50$:    ASL     R2              ;SHIFT TO NEXT BIT
6910  017252  001354                          BNE     30$             ;CONTINUE IF R2 NOT ZERO
6911
6912  017254                          60$:
6913
6914                                  ;;*****************************************************************
6915                                  ;*TEST 25        ILLEGAL REGISTER TEST
6916
6917                                  ;;*****************************************************************
6918  017254  000004                  TST25:  SCOPE
6919  017256  012737  000025  001226          MOV     #25,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
6920  017264  000240                          NOP
6921  017266  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
6922  017274  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
6923  017302  012737  017316  001122          MOV     #T25,$LPADR  ;LOAD LOOP ON TEST ADDRESS
6924  017310  012737  017316  001124          MOV     #T25,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
6925  017316                          T25:
6926  017316  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
6927  017322  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
6928  017326  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
6929  017332  005037  001140                  CLR     $GDDAT          ;''ILR'' SHOULD BE ZERO
6930  017336  012702  000000                  MOV     #0,R2           ;R2=REGISTER INDEX
6931  017342                          10$:
6932                                  ;;*****************************************************************
6933                                  ;CLEAR AND VERIFY THAT ''ILR'' STATUS IS ZERO
6934  017342  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
6935  017350  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
6936
6937  017354  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
6938  017362  042737  177775  001142          BIC     #^CILR,$BDDAT
6939  017370  001411                          BEQ     20$             ;BRANCH IF ILR IS RESET
6940  017372  005037  001140                  CLR     $GDDAT          ;SETUP GOOD DATA, REG ADR
```

```
6941  017376  010037  001136                  MOV     R0,$BDADR
6942  017402  062737  000014  001136          ADD     #RMER1,$BDADR
6943  017410  104064                          ERROR   64              ;CANT CLEAR ILR
6944  017412  000550                          BR      140$
6945  017414                          20$:
6946                                  ;;**********************************************************************
6947                                  ;WRITE THE REGISTER (INDEX=R2) AND TEST ILR STATUS
6948  017414  010003                          MOV     R0,R3           ;R3=REG ADDRESS
6949  017416  060203                          ADD     R2,R3
6950  017420  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
6951  017424  013746  000006                  MOV     ERRVEC+2,-(SP)          ;;PUSH ERRVEC+2 ON STACK
6952  017430  012737  017472  000004          MOV     #40$,ERRVEC     ;SETUP FOR BUS TIMEOUT
6953  017436  012737  000300  000006          MOV     #PR6,ERRVEC+2
6954  017444  005004                          CLR     R4              ;R4=REGISTER VALUE
6955  017446  022702  000010                  CMP     #RMCS2,R2
6956  017452  001001                          BNE     30$
6957  017454  111104                          MOVB    (R1),R4         ;SELECT DRIVE IF RMCS2
6958  017456  010413                  30$:    MOV     R4,(R3)         ;WRITE TEST REGISTER
6959  017460  012637  000006                  MOV     (SP)+,ERRVEC+2          ;;POP STACK INTO ERRVEC+2
6960  017464  012637  000004                  MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
6961  017470  000416                          BR      60$
6962  017472  012716  017500          40$:    MOV     #45$,(SP)       ;DUMMY RTI ADDRESS
6963  017476  000002                          RTI                     ;RESTORE PRIORITY
6964  017500                          45$:
6965  017500  012637  000006                  MOV     (SP)+,ERRVEC+2          ;;POP STACK INTO ERRVEC+2
6966  017504  012637  000004                  MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
6967  017510  020227  000046                  CMP     R2,#RMEC2       ;WERE ALL RM03 REGISTERS READ??
6968  017514  101003                          BHI     50$             ;YES!!
6969  017516  010337  001136                  MOV     R3,$BDADR
6970  017522  104074                          ERROR   74              ;UNEXPECTED BUS TIMEOUT
6971  017524  000503                  50$:    BR      140$
6972
6973  017526                          60$:
6974
6975  017526  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
6976  017534  042737  177775  001142          BIC     #^CILR,$BDDAT
6977  017542  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IS ''ILR'' OK??
6978  017550  001411                          BEQ     80$             ;YES!!
6979  017552  010337  001174                  MOV     R3,$TMP0        ;SAVE ADDRESS
6980  017556  032737  000002  001140          BIT     #ILR,$GDDAT     ;SHOULD ''ILR'' BE SET??
6981  017564  001002                          BNE     70$             ;YES!!
6982  017566  104065                          ERROR   65              ;''ILR'' SHOULD NOT BE SET
6983  017570  000401                          BR      80$
6984  017572  104066                  70$:    ERROR   66              ;''ILR'' SHOULD BE SET
6985
6986  017574                          80$:
6987                                  ;ADVANCE TO THE NEXT REGISTER ADDRESS
6988  017574  062702  000002                  ADD     #2,R2           ;INCREMENT INDEX
6989  017600  022702  000050                  CMP     #50,R2          ;TIME TO TRY RH70 ?
6990  017604  101256                          BHI     10$             ;BRANCH IF NOT
6991  017606  103437                          BLO     110$            ;BRANCH IF ALREADY CHECKED
6992                                  ;       CMP     #RMBAE,R2       ;IS THIS RMBAE??
6993                                  ;       BNE     100$            ;NO!!
6994  017610  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
6995  017614  013746  000006                  MOV     ERRVEC+2,-(SP)          ;;PUSH ERRVEC+2 ON STACK
6996  017620  012737  017716  000004          MOV     #130$,ERRVEC    ;SETUP FOR TIMEOUT
```

```
6997  017626  012737  000300  000006          MOV     #PR6,ERRVEC+2
6998                                  ;       PUTCS1  #A17!A16
6999                                  ;       GETBAE
7000                                  ;       BIC     #^C<BIT1!BIT0>,RMBAEI
7001                                  ;       BNE     90$             ;BRANCH IF RH70 CONTROLLER
7002                                  ;       BIS     #ILR,$GDDAT     ;"ILR" SHOULD BE SET
7003                                  ;90$:   POP     ERRVEC+2
7004                                  ;       POP     ERRVEC
7005                                  ;100$:  CMP     #RMCS3+2,R2     ;DONE ALL LEGAL REGISTERS
7006                                  ;       BNE     110$
7007                                  ;       BIS     #ILR,$GDDAT     ;"ILR" SHOULD BE SET
7008  017634  052737  000002  001140          BIS     #ILR,$GDDAT     ;SET ILR
7009  017642  012702  000054                  MOV     #54,R2          ;START AT INDEX 54 IF RH70/22REG
7010  017646  012760  001400  000000          MOV     #A17!A16,RMCS1(R0)      ;SET EXTEND BITS
7011  017654  016037  000050  001376          MOV     50(R0),RMBAEI   ;READ THE EXTENDED BITS
7012  017662  042737  177774  001376          BIC     #177774,RMBAEI  ;CHOP OFF
7013  017670  001002                          BNE     90$             ;BRANCH IF RH70/22-REG
7014  017672  012702  000050                  MOV     #50,R2          ;OTHERWISE NOT A RH70 OR RH70 WITH32REG
7015  017676  012637  000006          90$:    MOV     (SP)+,ERRVEC+2  ;
7016  017702  012637  000004                  MOV     (SP)+,ERRVEC    ;
7017  017706  022702  000074          110$:   CMP     #74,R2          ;DONE ALL TESTS
7018  017712  101410                          BLOS    140$            ;YES!!
7019  017714  000612                  120$:   BR      10$
7020
7021  017716  012716  017724          130$:   MOV     #135$,(SP)      ;DUMMY RTI ADDRESS
7022  017722  000002                          RTI                     ;RESTORE PRIORITY
7023  017724                          135$:
7024  017724  012637  000004                  MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
7025  017730  012637  000006                  MOV     (SP)+,ERRVEC+2          ;;POP STACK INTO ERRVEC+2
7026  017734                          140$:                           ;END OF TEST
7027
7028                                  ;;*****************************************************************
7029                                  ;*TEST 26       RESET GO BY INIT TEST
7030
7031                                  ;;*****************************************************************
7032  017734  000004                  TST26:  SCOPE
7033  017736  012737  000026  001226          MOV     #26,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7034
7035  017744  000240                          NOP
7036  017746  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
7037  017754  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
7038  017762  012737  017776  001122          MOV     #T26,$LPADR  ;LOAD LOOP ON TEST ADDRESS
7039  017770  012737  017776  001124          MOV     #T26,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
7040  017776                          T26:
7041  017776  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
7042  020002  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
7043  020006  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
7044  020012  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7045  020020  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7046  020024  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS FOR MSG
7047                                  ;SET GO, INITIALIZE AND VERIFY THAT GO IS RESET
7048
7049  020030  012760  000001  000000          MOV     #GO,RMCS1(R0)   ;LOAD RMCS1
7050  020036  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7051  020044  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7052
```

```
7053  020050  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
7054  020056  042737  177776  001142          BIC     #^CGO,$BDDAT
7055  020064  001403                          BEQ     10$             ;BRANCH IF GO IS RESET
7056  020066  005037  001140                  CLR     $GDDAT
7057  020072  104137                          ERROR   137             ;GO NOT CLEARED BY INIT
7058
7059  020074                          10$:                            ;END OF TEST
7060
7061                                  ;;*****************************************************************
7062                                  ;*TEST 27        DIAGNOSTIC MODE TEST
7063
7064                                  ;;*****************************************************************
7065  020074  000004                  TST27:  SCOPE
7066  020076  012737  000027  001226          MOV     #27,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7067
7068  020104  000240                          NOP
7069  020106  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
7070  020114  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
7071  020122  012737  020136  001122          MOV     #T27,$LPADR  ;LOAD LOOP ON TEST ADDRESS
7072  020130  012737  020136  001124          MOV     #T27,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
7073  020136                          T27:
7074  020136  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
7075  020142  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
7076  020146  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
7077  020152  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7078  020156  062737  000024  001136          ADD     #RMMR1,$BDADR
7079  020164  005003                          CLR     R3              ;INITIALIZE ERROR FLAGS
7080                                  ;INITIALIZE AND VERIFY THAT 'DMD' IS RESET
7081  020166  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7082  020174  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7083
7084  020200  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
7085  020206  042737  177776  001142          BIC     #^CDMD,$BDDAT
7086  020214  001403                          BEQ     10$             ;BRANCH IF 'DMD' IS ZERO
7087  020216  005037  001140                  CLR     $GDDAT
7088  020222  104075                          ERROR   75              ;CANT CLEAR 'DMD'
7089  020224                          10$:
7090                                  ;SET AND RESET 'DMD' USING REGISTER TRANSFER-VERIFY 'DMD' NOT S-A-1
7091
7092  020224  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7093
7094  020232  012760  000000  000024          MOV     #0,RMMR1(R0)    ;LOAD RMMR1
7095
7096  020240  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
7097  020246  042737  177776  001142          BIC     #^CDMD,$BDDAT
7098  020254  001405                          BEQ     20$             ;BRANCH IF DMD NOT S-A-1
7099  020256  005037  001140                  CLR     $GDDAT
7100  020262  104076                          ERROR   76              ;CANT WRITE DMD=0
7101  020264  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
7102  020270                          20$:
7103                                  ;RESET AND SET 'DMD' USING REGISTER TRANSFER-VERIFY 'DMD' NOT S-A-0
7104
7105  020270  012760  000000  000024          MOV     #0,RMMR1(R0)    ;LOAD RMMR1
7106
7107  020276  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7108
```

```
7109  020304  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
7110  020312  042737  177776  001142          BIC     #^CDMD,$BDDAT
7111  020320  001006                          BNE     30$                     ;BRANCH IF DMD NOT S-A-0
7112  020322  012737  000001  001140          MOV     #DMD,$GDDAT
7113  020330  104077                          ERROR   77                      ;CAN'T WRITE DMD=1
7114  020332  052703  000002                  BIS     #BIT1,R3                ;SET ERROR FLAG
7115  020336                          30$:
7116                                  ;IF NO PREVIOUS ERROR, TEST FOR BIT INTERFERENCE WITH SHIFTING
7117                                  ;ONE BIT PATTERN
7118  020336  005703                          TST     R3                      ;ANY ERRORS DETECTED??
7119  020340  001027                          BNE     60$                     ;YES!!
7120  020342  012702  000002                  MOV     #BIT1,R2                ;INITAILIZE DATA PATTERN
7121  020346                          40$:
7122
7123  020346  012760  000000  000024          MOV     #0,RMMR1(R0)    ;LOAD RMMR1
7124
7125  020354  010260  000024                  MOV     R2,RMMR1(R0)    ;LOAD RMMR1
7126
7127  020360  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
7128  020366  042737  177776  001142          BIC     #^CDMD,$BDDAT
7129  020374  001407                          BEQ     50$                     ;BRANCH IF DMD NOT SET
7130  020376  010237  001174                  MOV     R2,$TMP0                ;SAVE DATA
7131  020402  010237  001174                  MOV     R2,$TMP0                ;SAVE DATA
7132  020406  005037  001140                  CLR     $GDDAT                  ;DMD SHOULD BE ZERO
7133  020412  104100                          ERROR   100                     ;DMD SET BY WRONG BIT
7134  020414                          50$:
7135                                  ;       SHIFT TO NEXT DATA BIT AND CONTINUE TEST IF NOT DONE
7136  020414  006302                          ASL     R2
7137  020416  001353                          BNE     40$
7138  020420                          60$:                                    ;END OF TEST
7139                                  ;;*****************************************************************
7140                                  ;*TEST 30       MOL TEST
7141
7142                                  ;;*****************************************************************
7143  020420  000004                  TST30:  SCOPE
7144  020422  012737  000030  001226          MOV     #30,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7145
7146  020430  000240                          NOP
7147  020432  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
7148  020440  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
7149  020446  012737  020462  001122          MOV     #T30,$LPADR  ;LOAD LOOP ON TEST ADDRESS
7150  020454  012737  020462  001124          MOV     #T30,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
7151  020462                          T30:
7152  020462  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
7153  020466  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
7154  020472  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
7155  020476  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7156  020504  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7157  020510  010037  001136                  MOV     R0,$BDADR       ;R0=REGISTER ADDRESS
7158  020514  062737  000012  001136          ADD     #RMDS,$BDADR
7159  020522  005003                          CLR     R3              ;R3=ERROR FLAG
7160                                  ;SET DIAGNOSTIC MODE AND VERIFY THAT 'MOL'' IS ZERO
7161
7162  020524  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7163
7164  020532  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
```

```
7165  020540  042737  167777  001142          BIC     #^CMOL,$BDDAT
7166  020546  001405                          BEQ     10$
7167  020550  005037  001140                  CLR     $GDDAT
7168  020554  104101                          ERROR   101             ;'MOL'' NOT ZERO
7169  020556  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
7170  020562                          10$:
7171                                  ;SET MAINTENANCE UNIT READY AND VERIFY THAT 'MOL'' IS ONE
7172
7173  020562  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7174
7175  020570  012760  001001  000024          MOV     #DMD!MUR,RMMR1(R0)      ;LOAD RMMR1
7176
7177  020576  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
7178  020604  042737  167777  001142          BIC     #^CMOL,$BDDAT
7179  020612  001006                          BNE     20$
7180  020614  012737  010000  001140          MOV     #MOL,$GDDAT
7181  020622  104102                          ERROR   102             ;'MOL'' NOT ONE
7182  020624  052703  000002                  BIS     #BIT1,R3
7183  020630                          20$:
7184                                  ;IF NO PREVIOUS ERROR, VERIFY VOLUME VALID IS RESET AND
7185                                  ;TEST FOR BIT INTERFERENCE
7186  020630  005703                          TST     R3              ;ANY ERROR DETECTED??
7187  020632  001057                          BNE     70$             ;YES!!
7188
7189  020634  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
7190  020642  042737  177677  001142          BIC     #^CVV,$BDDAT
7191  020650  001403                          BEQ     25$             ;BRANCH IF VV RESET
7192  020652  005037  001140                  CLR     $GDDAT
7193  020656  104135                          ERROR   135             ;VV NOT RESET
7194  020660                          25$:
7195  020660  012702  000001                  MOV     #1,R2           ;INITIALIZE DATA PATTERN
7196  020664                          30$:
7197
7198  020664  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7199
7200  020672  010260  000024                  MOV     R2,RMMR1(R0)    ;LOAD RMMR1
7201
7202  020676  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
7203  020704  042737  167777  001142          BIC     #^CMOL,$BDDAT
7204  020712  005003                          CLR     R3              ;SETUP EXPECTED 'MOL''
7205  020714  032702  001000                  BIT     #MUR,R2
7206  020720  001402                          BEQ     40$
7207  020722  052703  010000                  BIS     #MOL,R3         ;'MOL'' SHOULD BE ONE
7208  020726  020337  001142          40$:    CMP     R3,$BDDAT       ;IS MOL OK??
7209  020732  001405                          BEQ     50$             ;YES!!
7210  020734  010237  001174                  MOV     R2,$TMP0        ;SAVE TEST PATTERN
7211  020740  010337  001140                  MOV     R3,$GDDAT
7212  020744  104103                          ERROR   103             ;MUR SET BY WRONG BIT
7213  020746                          50$:
7214                                  ;SHIFT TO NEXT PATTERN
7215  020746  042702  000001                  BIC     #DMD,R2         ;DONT SHIFT DMD
7216  020752  001002                          BNE     60$
7217  020754  012702  000001                  MOV     #DMD,R2         ;DONT TRUNCATE TEST
7218  020760  006302                  60$:    ASL     R2              ;SHIFT TO NEXT BIT
7219  020762  001403                          BEQ     70$             ;BRANCH IF DONE
7220  020764  052702  000001                  BIS     #DMD,R2         ;KEEP DMD ON
```

```
7221  020770  000735                        BR      30$             ;CONTINUE
7222
7223  020772                        70$:                            ;END OF TEST
7224
7225                                ;;*****************************************************************
7226                                ;*TEST 31        WRITE LOCK TEST
7227
7228                                ;;*****************************************************************
7229  020772  000004                TST31:  SCOPE
7230  020774  012737  000031  001226        MOV     #31,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7231
7232  021002  000240                        NOP
7233  021004  012737  000024  001120        MOV     #20.,$ICNT      ;20 ITERATIONS
7234  021012  112737  000001  001131        MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
7235  021020  012737  021034  001122        MOV     #T31,$LPADR  ;LOAD LOOP ON TEST ADDRESS
7236  021026  012737  021034  001124        MOV     #T31,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
7237  021034                        T31:
7238  021034  012706  001100                MOV     #STACK,SP       ;LOAD THE STACK POINTER
7239  021040  013700  001276                MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
7240  021044  013701  001456                MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
7241  021050  005003                        CLR     R3              ;R3=ERROR FLAG
7242  021052  010037  001136                MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7243  021056  062737  000012  001136        ADD     #RMDS,$BDADR
7244  021064  005037  001140                CLR     $GDDAT          ;WRL SHOULD BE ZERO
7245  021070  012760  000040  000010        MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7246  021076  111160  000010                MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7247                              ;SET DIAGNOSTIC MODE AND VERIFY "WRL" IS ZERO
7248
7249  021102  012760  000001  000024        MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7250
7251  021110  016037  000012  001142        MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
7252  021116  042737  173777  001142        BIC     #^CWRL,$BDDAT
7253  021124  001403                        BEQ     10$             ;BRANCH IF WRL IS ZERO
7254  021126  104104                        ERROR   104             ;WRL NOT ZERO
7255  021130  052703  000001                BIS     #BIT0,R3        ;SET ERROR FLAG
7256  021134                        10$:
7257                              ;SET MAINTENANCE WRITE PROTECT AND VERIFY "WRL" IS ONE
7258
7259  021134  012760  000001  000024        MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7260
7261  021142  012760  000011  000024        MOV     #DMD!MWP,RMMR1(R0)      ;LOAD RMMR1
7262
7263  021150  016037  000012  001142        MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
7264  021156  042737  173777  001142        BIC     #^CWRL,$BDDAT
7265  021164  001006                        BNE     20$             ;BRANCH IF WRL IS NOE
7266  021166  012737  004000  001140        MOV     #WRL,$GDDAT     ;WRL SHOULD BE SET
7267  021174  104105                        ERROR   105             ;CANT SET WRL
7268  021176  052703  000002                BIS     #BIT1,R3        ;SET ERROR FLAG
7269  021202                        20$:
7270                              ;IF NO PREVIOUS ERROR, TEST FOR BIT INTERFERENCE ON "MWP"
7271  021202  005703                        TST     R3              ;ANY OTHER ERROR??
7272  021204  001045                        BNE     70$             ;YES!!
7273  021206  012702  000001                MOV     #1,R2           ;INITIALIZE DATA PATTERN
7274
7275  021212                        30$:
7276                              ;       TRANSFER DATA TO RMMR1, READ RMDS AND VERIFY WRL
```

```
7277
7278  021212  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7279
7280  021220  010260  000024                  MOV     R2,RMMR1(R0)    ;LOAD RMMR1
7281
7282  021224  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
7283  021232  042737  173777  001142          BIC     #^CWRL,$BDDAT   ;CLEARUP RECEIVED 'WRL'
7284  021240  005003                          CLR     R3              ;GENERATE EXPECTED 'WRL'
7285  021242  032702  000010                  BIT     #MWP,R2
7286  021246  001402                          BEQ     40$
7287  021250  052703  004000                  BIS     #WRL,R3         ;WRL SHOULD BE SET
7288  021254  020337  001142          40$:    CMP     R3,$BDDAT       ;IS WRL OK??
7289  021260  001405                          BEQ     50$             ;YES!!
7290  021262  010337  001140                  MOV     R3,$GDDAT       ;SAVE EXPECTED
7291  021266  010237  001174                  MOV     R2,$TMP0        ;SAVE DATA PATTERN
7292  021272  104106                          ERROR   106             ;BIT INTERFERENCE WITH MWP
7293  021274                          50$:
7294                                  ;       ADVANCE TO NEXT DATA BIT
7295  021274  042702  000001                  BIC     #DMD,R2         ;DONT SHIFT DMD
7296  021300  001002                          BNE     60$
7297  021302  012702  000001                  MOV     #DMD,R2         ;DONT TRUNCATE TEST
7298  021306  006302                  60$:    ASL     R2              ;SHIFT DATA BIT
7299  021310  001403                          BEQ     70$             ;EXIT IF DONE
7300  021312  052702  000001                  BIS     #DMD,R2         ;KEEP DIAGNOSTIC MODE ON
7301  021316  000735                          BR      30$             ;CONTINUE TEST
7302
7303  021320                          70$:                            ;END OF TEST
7304
7305                                  ;;*****************************************************************
7306                                  ;*TEST 32       DRIVE FAULT TEST
7307
7308                                  ;;*****************************************************************
7309  021320  000004                  TST32:  SCOPE
7310  021322  012737  000032  001226          MOV     #32,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7311
7312  021330  000240                          NOP
7313  021332  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
7314  021340  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
7315  021346  012737  021362  001122          MOV     #T32,$LPADR  ;LOAD LOOP ON TEST ADDRESS
7316  021354  012737  021362  001124          MOV     #T32,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
7317  021362                          T32:
7318  021362  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
7319  021366  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
7320  021372  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
7321  021376  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7322  021404  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7323  021410  005003                          CLR     R3              ;INITIALIZE ERROR FLAGS
7324  021412  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7325  021416  062737  000042  001136          ADD     #RMER2,$BDADR
7326  021424  005037  001140                  CLR     $GDDAT          ;'DVC' AND 'UNS' SHOULD BE ZERO
7327                                  ;SET AND RESET MAINTENANCE DRIVE FAULT, VERIFY THAT 'DVC' IS NOT
7328                                  ;STUCK-AT-ONE.
7329
7330  021430  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7331
7332  021436  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
```

```
7333
7334  021444  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
7335
7336  021452  016037  000042  001142          MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
7337  021460  042737  177577  001142          BIC     #^CDVC,$BDDAT   ;IS 'DVC' RESET??
7338  021466  001406                          BEQ     10$             ;YES!!
7339  021470  104107                          ERROR   107             ;CANT RESET DVC
7340  021472  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
7341  021476  012737  040000  001140          MOV     #UNS,$GDDAT
7342  021504                          10$:
7343
7344                                  ;VERIFY THAT 'UNS' IS SAME AS 'DVC'
7345
7346  021504  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
7347  021512  042737  137777  001142          BIC     #^CUNS,$BDDAT
7348  021520  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;IS 'UNS' OK??
7349  021526  001414                          BEQ     30$             ;YES!!
7350  021530  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7351  021534  062737  000014  001136          ADD     #RMER1,$BDADR
7352  021542  032737  040000  001140          BIT     #UNS,$GDDAT     ;SHOULD 'UNS' BE ON??
7353  021550  001002                          BNE     20$             ;YES!!
7354  021552  104110                          ERROR   110             ;UNS IS SET, DVS IS RESET
7355  021554  000401                          BR      30$
7356  021556                          20$:
7357  021556  104111                          ERROR   111             ;UNS IS RESET, DVC IS SET
7358  021560                          30$:
7359
7360                                  ;RESET AND SET 'MDF', VERIFY THAT 'DVC' IS NOT S-A-0.
7361  021560  012737  000200  001140          MOV     #DVC,$GDDAT     ;DVC SHOULD BE ON
7362
7363  021566  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7364
7365  021574  012760  000101  000024          MOV     #DMD!MDF,RMMR1(R0)      ;LOAD RMMR1
7366
7367  021602  016037  000042  001142          MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
7368  021610  042737  177577  001142          BIC     #^CDVC,$BDDAT
7369  021616  001012                          BNE     40$             ;BRANCH IF DVC IS SET
7370  021620  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7371  021624  062737  000042  001136          ADD     #RMER2,$BDADR
7372  021632  104112                          ERROR   112             ;CANT SET DVC
7373  021634  052703  000002                  BIS     #BIT1,R3        ;SET ERROR FLAG
7374  021640  005037  001140                  CLR     $GDDAT          ;UNS SHOULD BE OFF
7375  021644                          40$:
7376
7377                                  ;VERIFY THAT 'UNS' IS SAME AS 'DVC'
7378  021644  005737  001140                  TST     $GDDAT          ;CHANGE DVC TO UNS
7379  021650  001403                          BEQ     50$
7380  021652  012737  040000  001140          MOV     #UNS,$GDDAT
7381  021660                          50$:
7382
7383  021660  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
7384  021666  042737  137777  001142          BIC     #^CUNS,$BDDAT
7385  021674  023737  001140  001142          CMP     $GDDAT,$BDDAT
7386  021702  001414                          BEQ     70$             ;BRANCH IF UNS IS OK
7387  021704  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7388  021710  062737  000014  001136          ADD     #RMER1,$BDADR
```

```
7389  021716  032737  040000  001140          BIT     #UNS,$GDDAT     ;SHOULD UNS BE ON??
7390  021724  001002                          BNE     60$             ;YES!!
7391  021726  104113                          ERROR   113             ;UNS IS SET, DVC IS RESET
7392  021730  000401                          BR      70$
7393  021732                          60$:
7394  021732  104114                          ERROR   114             ;UNS IS RESET, DVC IS SET
7395  021734                          70$:
7396
7397                                  ;IF THERE WERE NO PREVIOUS ERRORS, TEST FOR BIT INTERFERENCE ON
7398                                  ;'MDF''
7399  021734  005703                          TST     R3
7400  021736  001056                          BNE     120$            ;BRANCH IF ANY OTHER ERRORS
7401  021740  012702  000001                  MOV     #1,R2           ;INITIALIZE TEST PATTERN
7402  021744                          80$:
7403
7404  021744  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7405
7406  021752  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
7407
7408  021760  010260  000024                  MOV     R2,RMMR1(R0)    ;LOAD RMMR1
7409
7410  021764  016037  000042  001142          MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
7411  021772  042737  177577  001142          BIC     #^CDVC,$BDDAT   ;GET RESULTS
7412  022000  005037  001140                  CLR     $GDDAT          ;SETUP EXPECTED
7413  022004  032702  000100                  BIT     #MDF,R2         ;WAS MDF SET??
7414  022010  001403                          BEQ     90$             ;NO!!
7415  022012  052737  000200  001140          BIS     #DVC,$GDDAT     ;YES-DVC SHOULD BE ON
7416  022020  023737  001140  001142  90$:    CMP     $GDDAT,$BDDAT
7417  022026  001410                          BEQ     100$            ;BRANCH IF DVC IS OK
7418  022030  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7419  022034  062737  000042  001136          ADD     #RMER2,$BDADR
7420  022042  010237  001174                  MOV     R2,$TMP0        ;SAVE TEST PATTERN
7421  022046  104115                          ERROR   115             ;MDF SET BY WRONG BIT
7422  022050                          100$:
7423                                  ;SHIFT TO NEXT BIT POSITION
7424  022050  042702  000001                  BIC     #DMD,R2         ;DONT SHIFT DMD
7425  022054  001002                          BNE     110$
7426  022056  012702  000001                  MOV     #DMD,R2         ;DONT TRUNCATE TEST
7427  022062  006302                  110$:   ASL     R2              ;SHIFT
7428  022064  001403                          BEQ     120$            ;EXIT IF DONE
7429  022066  052702  000001                  BIS     #DMD,R2         ;KEEP DMD ON
7430  022072  000724                          BR      80$             ;CONTINUE
7431
7432  022074                          120$:                           ;END OF TEST
7433
7434                                  ;;*****************************************************************
7435                                  ;*TEST 33        SEEK ERROR TEST
7436
7437                                  ;;*****************************************************************
7438  022074  000004                  TST33:  SCOPE
7439  022076  012737  000033  001226          MOV     #33,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7440
7441  022104  000240                          NOP
7442  022106  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
7443  022114  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
7444  022122  012737  022136  001122          MOV     #T33,$LPADR  ;LOAD LOOP ON TEST ADDRESS
```

```
7445  022130  012737  022136  001124          MOV     #T33,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
7446  022136                          T33:
7447  022136  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
7448  022142  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
7449  022146  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
7450  022152  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7451  022160  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7452  022164  005003                          CLR     R3              ;CLEAR ERROR FLAGS
7453  022166  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7454  022172  062737  000042  001136          ADD     #RMER2,$BDADR
7455
7456                          ;SET DIAGNOSTIC MODE AND VERIFY THAT "SKI" CAN BE RESET
7457
7458  022200  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7459
7460  022206  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
7461
7462  022214  016037  000042  001142          MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
7463  022222  042737  137777  001142          BIC     #^CSKI,$BDDAT
7464  022230  001405                          BEQ     10$             ;BRANCH IF SKI IS RESET
7465  022232  005037  001140                  CLR     $GDDAT          ;SKI SHOULD BE ZERO
7466  022236  104116                          ERROR   116             ;CANT RESET "SKI"
7467  022240  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
7468  022244                          10$:
7469
7470                          ;SET MAINTENANCE SEEK ERROR AND VERIFY THAT "SKI" CAN BE SET
7471
7472  022244  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7473
7474  022252  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
7475
7476  022260  012760  000201  000024          MOV     #DMD!MSER,RMMR1(R0)     ;LOAD RMMR1
7477
7478  022266  016037  000042  001142          MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
7479  022274  042737  137777  001142          BIC     #^CSKI,$BDDAT
7480  022302  001005                          BNE     20$             ;BRANCH IF SKI IS SET
7481  022304  012737  040000  001140          MOV     #SKI,$GDDAT     ;CANT SET SKI
7482  022312  052703  000002                  BIS     #BIT1,R3        ;SET ERROR FLAG
7483  022316                          20$:
7484
7485                          ;IF NO PREVIOUS ERROR, CHECK FOR BIT INTERFERENCE SETTING MAINTENANCE
7486                          ;SEEK ERROR.
7487  022316  005703                          TST     R3
7488  022320  001051                          BNE     70$             ;BRANCH IF ANY OTHER ERRORS
7489  022322  012702  000001                  MOV     #1,R2           ;INITIALIZE TEST PATTERN
7490  022326                          30$:
7491
7492  022326  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7493
7494  022334  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
7495
7496  022342  010260  000024                  MOV     R2,RMMR1(R0)    ;LOAD RMMR1
7497
7498  022346  016037  000042  001142          MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
7499  022354  042737  137777  001142          BIC     #^CSKI,$BDDAT   ;GET SKI STATUS
7500  022362  005037  001140                  CLR     $GDDAT          ;SETUP EXPECTED RESULT
```

```
7501  022366  032702  000200              BIT     #MSER,R2
7502  022372  001403                      BEQ     40$
7503  022374  052737  040000  001140      BIS     #SKI,$GDDAT     ;SKI SHOULD BE ON
7504  022402  023737  001140  001142 40$: CMP     $GDDAT,$BDDAT
7505  022410  001403                      BEQ     50$             ;BRANCH IF SKI IS OK
7506  022412  010237  001174              MOV     R2,$TMP0        ;SAVE TEST PATTERN
7507  022416  104120                      ERROR   120             ;SKI SET BY WRONG BIT
7508  022420                          50$:
7509                                  ;       ADVANCE TEST PATTERN IN R2
7510  022420  042702  000001              BIC     #DMD,R2         ;DONT SHIFT DMD BIT
7511  022424  001002                      BNE     60$
7512  022426  012702  000001              MOV     #DMD,R2         ;DONT TRUNCATE TEST
7513  022432  006302                  60$: ASL     R2              ;SHIFT TO NEXT BIT
7514  022434  001403                      BEQ     70$             ;EXIT IF DONE
7515  022436  052702  000001              BIS     #DMD,R2         ;KEEP DMD ON
7516  022442  000731                      BR      30$             ;CONTINUE TEST
7517
7518  022444                          70$:                        ;END OF TEST
7519
7520                                  ;;*****************************************************************
7521                                  ;*TEST 34        PIP TEST
7522
7523                                  ;;*****************************************************************
7524  022444  000004                  TST34:  SCOPE
7525  022446  012737  000034  001226          MOV     #34,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7526
7527  022454  000240                          NOP
7528  022456  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
7529  022464  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
7530  022472  012737  022506  001122          MOV     #T34,$LPADR  ;LOAD LOOP ON TEST ADDRESS
7531  022500  012737  022506  001124          MOV     #T34,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
7532  022506                          T34:
7533  022506  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
7534  022512  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
7535  022516  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
7536  022522  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7537  022530  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7538  022534  005003                          CLR     R3              ;RESET ERROR FLAGS
7539  022536  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7540  022542  062737  000012  001136          ADD     #RMDS,$BDADR
7541
7542                                  ;SET MAINTENANCE ON CYLINDER 'MOC'' AND VERIFY THAT 'PIP'' CAN BE RESET.
7543
7544  022550  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7545
7546  022556  012760  000401  000024          MOV     #DMD!MOC,RMMR1(R0)      ;LOAD RMMR1
7547
7548  022564  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
7549  022572  042737  157777  001142          BIC     #^CPIP,$BDDAT
7550  022600  001405                          BEQ     10$             ;BRANCH IF PIP IS RESET
7551  022602  005037  001140                  CLR     $GDDAT
7552  022606  104121                          ERROR   121             ;CANT RESET PIP
7553  022610  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
7554  022614                          10$:
7555                                  ;RESET MAINTENANCE ON CYLINDER AND VERIFY THAT 'PIP'' CAN BE SET
7556
```

```
7557  022614  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7558
7559  022622  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
7560  022630  042737  157777  001142          BIC     #^CPIP,$BDDAT
7561  022636  001006                          BNE     20$             ;BRANCH IF PIP IS SET
7562  022640  012737  020000  001140          MOV     #PIP,$GDDAT
7563  022646  104122                          ERROR   122             ;CANT SET PIP
7564  022650  052703  000002                  BIS     #BIT1,R3        ;SET ERROR FLAG
7565  022654                          20$:
7566                                  ;IF NO PREVIOUS ERROR, TEST FOR ADJACENT BIT SETTING 'MOC'
7567  022654  005703                          TST     R3
7568  022656  001046                          BNE     70$             ;BRANCH IF ANY PREVIOUS ERROR
7569  022660  012702  000001                  MOV     #1,R2           ;INITIALIZE TEST PATTERN
7570  022664                          30$:
7571                                  ;       WRITE THE TEST PATTERN, CHECK MOC USING PIP
7572
7573  022664  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7574
7575  022672  010260  000024                  MOV     R2,RMMR1(R0)    ;LOAD RMMR1
7576
7577  022676  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
7578  022704  042737  157777  001142          BIC     #^CPIP,$BDDAT   ;GET PIP STATUS
7579  022712  005037  001140                  CLR     $GDDAT          ;SETUP EXPECTED RESULT
7580  022716  032702  000400                  BIT     #MOC,R2
7581  022722  001003                          BNE     40$
7582  022724  052737  020000  001140          BIS     #PIP,$GDDAT     ;PIP SHOULD BE SET
7583  022732  023737  001140  001142  40$:    CMP     $GDDAT,$BDDAT
7584  022740  001403                          BEQ     50$             ;BRANCH IF PIP OK
7585  022742  010237  001174                  MOV     R2,$TMP0        ;SAVE TEST PATTERN
7586  022746  104123                          ERROR   123             ;MOC SET BY WRONG BIT
7587  022750                          50$:
7588                                  ;       ADVANCE THE TEST PATTERN
7589  022750  042702  000001                  BIC     #DMD,R2         ;DONT SHIFT DMD
7590  022754  001002                          BNE     60$
7591  022756  012702  000001                  MOV     #DMD,R2         ;DONT TRUNCATE TEST
7592  022762  006302                  60$:    ASL     R2              ;SHIFT BIT
7593  022764  001403                          BEQ     70$             ;EXIT IF DONE
7594  022766  052702  000001                  BIS     #DMD,R2         ;KEEP DMD ON
7595  022772  000734                          BR      30$             ;CONTINUE TEST
7596
7597  022774                          70$:                            ;END OF TEST
7598
7599                                  ;;*****************************************************************
7600                                  ;*TEST 35        EBL TEST
7601
7602                                  ;;*****************************************************************
7603  022774  000004                  TST35:  SCOPE
7604  022776  012737  000035  001226          MOV     #35,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7605
7606  023004  000240                          NOP
7607  023006  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
7608  023014  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
7609  023022  012737  023036  001122          MOV     #T35,$LPADR  ;LOAD LOOP ON TEST ADDRESS
7610  023030  012737  023036  001124          MOV     #T35,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
7611  023036                          T35:
7612  023036  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
```

```
7613  023042  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
7614  023046  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
7615  023052  005003                          CLR     R3              ;RESET ERROR FLAGS
7616  023054  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7617  023060  062737  000024  001136          ADD     #RMMR1,$BDADR
7618  023066  005037  001140                  CLR     $GDDAT          ;SETUP EXPECTED RESULT
7619                                  ;CLEAR AND VERIFY THAT END OF BLOCK IS RESET
7620  023072  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7621  023100  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7622
7623  023104  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
7624  023112  042737  157777  001142          BIC     #^CEBL,$BDDAT
7625  023120  001403                          BEQ     10$             ;BRANCH IF EBL IS RESET
7626  023122  104124                          ERROR   124             ;CANT CLEAR EBL
7627  023124  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
7628  023130                          10$:
7629                                  ;SET AND RESET DIAGNOSTIC END OF BLOCK, CHECK FOR EBL S-A-1.
7630
7631  023130  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7632
7633  023136  012760  020001  000024          MOV     #DMD!DEBL,RMMR1(R0)     ;LOAD RMMR1
7634
7635  023144  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7636
7637  023152  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
7638  023160  042737  157777  001142          BIC     #^CEBL,$BDDAT
7639  023166  001403                          BEQ     20$             ;BRANCH IF EBL IS RESET
7640  023170  104125                          ERROR   125             ;DEBL APPEARS S-A-1
7641  023172  052703  000001                  BIS     #BIT0,R3        ;SET ERROR FLAG
7642  023176                          20$:
7643                                  ;RESET AND SET DIAGNOSTIC END OF BLOCK, CHECK FOR EBL S-A-0
7644
7645  023176  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7646
7647  023204  012760  020001  000024          MOV     #DMD!DEBL,RMMR1(R0)     ;LOAD RMMR1
7648
7649  023212  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
7650  023220  042737  157777  001142          BIC     #^CEBL,$BDDAT
7651  023226  001006                          BNE     30$             ;BRANCH IF EBL IS SET
7652  023230  012737  020000  001140          MOV     #EBL,$GDDAT
7653  023236  104126                          ERROR   126             ;CANT SET EBL
7654  023240  052703  000002                  BIS     #BIT1,R3        ;SET ERROR FLAG
7655  023244                          30$:
7656                                  ;IF NO PREVIOUS ERRORS, TEST FOR ADJACENT BIT INTERFERENCE ON 'DEBL'.
7657  023244  005703                          TST     R3
7658  023246  001042                          BNE     70$             ;BRANCH IF ANY ERROR
7659  023250  012702  000001                  MOV     #1,R2           ;INITIALIZE TEST PATTERN
7660  023254                          40$:
7661                                  ;       WRITE, READ AND VERIFY THE TEST PATTERN IN R2
7662
7663  023254  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7664
7665  023262  010260  000024                  MOV     R2,RMMR1(R0)    ;LOAD RMMR1
7666
7667  023266  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
7668  023274  042737  157777  001142          BIC     #^CEBL,$BDDAT
```

```
7669  023302  010203                        MOV     R2,R3           ;GENERATE EXPECTED RESULT
7670  023304  042703  157777                BIC     #^CEBL,R3
7671  023310  020337  001142                CMP     R3,$BDDAT
7672  023314  001405                        BEQ     50$             ;BRANCH IF EBL IS OK
7673  023316  010337  001140                MOV     R3,$GDDAT       ;SAVE EXPECTED RESULT
7674  023322  010237  001174                MOV     R2,$TMP0        ;SAVE TEST PATTERN
7675  023326  104127                        ERROR   127             ;DEBL SET(RESET) BY WRONG BIT
7676  023330                        50$:
7677                                ;       SHIFT TO NEXT BIT POSITION
7678  023330  042702  000001                BIC     #DMD,R2         ;DONT SHIFT DMD
7679  023334  001002                        BNE     60$
7680  023336  012702  000001                MOV     #DMD,R2         ;DONT TRUNCATE DMD
7681  023342  006302                60$:    ASL     R2              ;SHIFT TO NEXT BIT
7682  023344  001403                        BEQ     70$             ;EXIT IF DONE
7683  023346  052702  000001                BIS     #DMD,R2         ;KEEP DMD ON
7684  023352  000740                        BR      40$             ;CONTINUE TEST
7685
7686  023354                        70$:                            ;END OF TEST
7687
7688                                ;;**************************************************************
7689                                ;*TEST 36        LAST SECTOR, LAST TRACK TEST
7690
7691                                ;;**************************************************************
7692  023354  000004                TST36:  SCOPE
7693  023356  012737  000036  001226        MOV     #36,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7694  023364  000240                        NOP
7695  023366  012737  000024  001120        MOV     #20.,$ICNT      ;20 ITERATIONS
7696  023374  112737  000001  001131        MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
7697  023402  012737  023416  001122        MOV     #T36,$LPADR  ;LOAD LOOP ON TEST ADDRESS
7698  023410  012737  023416  001124        MOV     #T36,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
7699  023416                        T36:
7700  023416  012706  001100                MOV     #STACK,SP       ;LOAD THE STACK POINTER
7701  023422  013700  001276                MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
7702  023426  013701  001456                MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
7703  023432  012760  000040  000010        MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7704  023440  111160  000010                MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7705  023444  005002                        CLR     R2              ;INITIALIZE TEST PATTERN
7706  023446  010037  001136                MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7707  023452  062737  000024  001136        ADD     #RMMR1,$BDADR
7708  023460  005037  001434                CLR     RMOF0           ;START IN 18 BIT MODE
7709  023464                        10$:
7710                                ;TRANSFER TEST PATTERN TO RMDA THEN VERIFY LAST SECTOR 'LS' AND LAST
7711                                ;SECTOR/TRACK 'LST' FOR EACH TRANSFER. THE TABLE BELOW LISTS THE VALUE
7712                                ;OF RMDA FOR WHICH LS AND LST ARE SET.
7713                                ;
7714                                ;       18 BIT MODE     16 BIT MODE
7715                                ;LS=    XXX035          XXX037
7716                                ;
7717                                ;LST=   002035          002037
7718
7719
7720  023464  013760  001434  000032        MOV     RMOF0,RMOF(R0)  ;LOAD RMOF
7721
7722  023472  010260  000006                MOV     R2,RMDA(R0)     ;LOAD RMDA
7723
7724  023476  016037  000024  001352        MOV     RMMR1(R0),RMMR1I        ;STORE RMMR1 IN INPUT BUFFER
```

```
7725  023504  013737  001352  001142          MOV     RMMR1I,$BDDAT   ;VERIFY 'LS'
7726  023512  042737  177773  001142          BIC     #^CLS,$BDDAT
7727  023520  005037  001140                  CLR     $GDDAT          ;GENERATE EXPECTED 'LS'
7728  023524  032737  010000  001434          BIT     #FMT16,RMOFO    ;16 BIT MODE??
7729  023532  001007                          BNE     20$             ;YES!!
7730  023534  122702  000035                  CMPB    #035,R2
7731  023540  001012                          BNE     30$
7732  023542  052737  000004  001140          BIS     #LS,$GDDAT      ;LS SHOULD BE ON-18 BIT MODE
7733  023550  000406                          BR      30$
7734  023552  122702  000037          20$:    CMPB    #037,R2
7735  023556  001003                          BNE     30$
7736  023560  052737  000004  001140          BIS     #LS,$GDDAT      ;LS SHOULD BE ON-16 BIT MODE
7737  023566  023737  001140  001142  30$:    CMP     $GDDAT,$BDDAT
7738  023574  001407                          BEQ     40$             ;BRANCH IF LS IS CORRECT
7739  023576  010237  001174                  MOV     R2,$TMP0        ;SAVE TEST PATTERN
7740  023602  013737  001434  001176          MOV     RMOFO,$TMP1     ;SAVE OFFSET REGISTER FOR ERROR
7741  023610  104130                          ERROR   130             ;LS NOT CORRECT FOR RMDA
7742  023612  000453                          BR      80$             ;SKIP TO NEXT
7743  023614  013737  001352  001142  40$:    MOV     RMMR1I,$BDDAT   ;VERIFY 'LST'
7744  023622  042737  177775  001142          BIC     #^CLST,$BDDAT
7745  023630  005037  001140                  CLR     $GDDAT          ;GENERATE EXPECTED 'LST'
7746  023634  032737  010000  001434          BIT     #FMT16,RMOFO    ;16 BIT MODE??
7747  023642  001007                          BNE     50$             ;YES!!
7748  023644  022702  002035                  CMP     #002035,R2
7749  023650  001012                          BNE     60$
7750  023652  052737  000002  001140          BIS     #LST,$GDDAT     ;LST SHOULD BE ON-18 BIT MODE
7751  023660  000406                          BR      60$
7752  023662  022702  002037          50$:    CMP     #002037,R2
7753  023666  001003                          BNE     60$
7754  023670  052737  000002  001140          BIS     #LST,$GDDAT     ;LST SHOULD BE ON-16 BIT MODE
7755  023676  023737  001140  001142  60$:    CMP     $GDDAT,$BDDAT
7756  023704  001404                          BEQ     70$
7757  023706  010237  001174                  MOV     R2,$TMP0        ;SAVE TEST PATTERN
7758  023712  104131                          ERROR   131             ;LST NOT CORRECT FOR RMDA
7759  023714  000412                          BR      80$             ;SKIP TO NEXT
7760
7761  023716                          70$:
7762                                  ;       ADVANCE TO NEXT TEST PATTERN, CHANGE TO 16 BIT MODE IF ALL
7763                                  ;18 BIT TESTS DONE.
7764  023716  005202                          INC     R2              ;INCREMENT PATTERN
7765  023720  001261                          BNE     10$             ;CONTINUE IF NOT DONE
7766  023722  032737  010000  001434          BIT     #FMT16,RMOFO    ;DONE 16 BIT TOO??
7767  023730  001004                          BNE     80$             ;YES!!
7768  023732  012737  010000  001434          MOV     #FMT16,RMOFO    ;DO 16 BIT FORMAT TEST
7769  023740  000651                          BR      10$
7770
7771  023742                          80$:                            ;END OF TEST
7772
7773                                  ;;**********************************************************************
7774                                  ;*TEST 37        RMDA COUNT TEST
7775
7776                                  ;;**********************************************************************
7777  023742  000004                  TST37:  SCOPE
7778  023744  012737  000037  001226          MOV     #37,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7779
7780  023752  000240                          NOP
```

```
 7781  023754  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
 7782  023762  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
 7783  023770  012737  024004  001122          MOV     #T37,$LPADR  ;LOAD LOOP ON TEST ADDRESS
 7784  023776  012737  024004  001124          MOV     #T37,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
 7785  024004                              T37:
 7786  024004  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
 7787  024010  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
 7788  024014  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
 7789  024020  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
 7790  024024  062737  000006  001136          ADD     #RMDA,$BDADR
 7791  024032  005037  001434                  CLR     RMOF0           ;START WITH 18 BIT FORMAT
 7792  024036  012737  000035  024460          MOV     #035,110$       ;SETUP LAST SECTOR FOR 18 BIT
 7793  024044  012737  000001  001140          MOV     #1,$GDDAT       ;SETUP FIRST COUNT
 7794                              ;;*****************************************************************
```

```
7795                    ;INCREMENT SECTOR COUNT USING DIAGNOSTIC END OF BLOCK STARTING AT
7796                    ;SECTOR 0 AND CONTINUING UNTIL TRACK ADDRESS INCREMENTS
```

```
7797  024052                          10$:
7798                                  ;       .CLEAR THE MASSBUS
7799                                  ;       .SET FORMAT
7800                                  ;       .LOAD SECTOR AND TRACK ADDRESS
7801                                  ;       .ENABLE DEBUG CLOCK
7802                                  ;       .SET GO BIT
7803  024052  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7804  024060  111160  000010                 MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7805
7806  024064  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7807
7808  024072  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
7809
7810  024100  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
7811
7812  024106  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
7813
7814  024114  013760  001434  000032          MOV     RMOF0,RMOF(R0)  ;LOAD RMOF
7815  024122  013737  001434  001174          MOV     RMOF0,$TMP0     ;SAVE OFFSET FOR ERROR TYPE
7816
7817  024130  012760  040001  000024          MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
7818
7819  024136  012760  000001  000000          MOV     #GO,RMCS1(R0)   ;LOAD RMCS1
```

```
7820  024144                                  25$:
7821
7822                                          ;       SET AND RESET EBL TO INCREMENT RMDA THEN VERIFY RMDA.
7823
7824  024144  012760  060001  000024                  MOV     #DMD!DBEN!DEBL,RMMR1(R0)        ;LOAD RMMR1
7825
7826  024152  012760  040001  000024                  MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
7827
7828  024160  016037  000006  001142                  MOV     RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
7829  024166  023737  001142  001140                  CMP     $BDDAT,$GDDAT
7830  024174  001402                                  BEQ     30$             ;BRANCH IF RMDA OK
7831  024176  104132                                  ERROR   132             ;SECTOR COUNT NOT INCREMENTING
7832  024200  000416                                  BR      50$             ;OUT OF SYNC-SKIP TO NEXT
7833  024202                                  30$:
7834                                          ;       ADVANCE EXPECTED SECTOR COUNT AND CONTINUE IF ONE CYCLE NOT
7835                                          ;COMPLETE
7836  024202  005237  001140                          INC     $GDDAT          ;INCREMENT EXPECTED SECTOR
7837  024206  123737  001142  024460                  CMPB    $BDDAT,110$     ;WAS THE LAST SECTOR JUST COUNTED??
7838  024214  001004                                  BNE     40$             ;NO!!
7839  024216  105037  001140                          CLRB    $GDDAT          ;YES-NEXT SECTOR SHOULD BE ZERO
7840  024222  105237  001141                          INCB    $GDDAT+1        ;INCREMENT TRACK ADDRESS
7841  024226  105737  001142          40$:    TSTB    $BDDAT          ;HAS A FULL CYCLE BEEN COUNTED??
7842  024232  001401                                  BEQ     50$             ;YES-DO NEXT
7843  024234  000743                                  BR      25$             ;CONTINUE SECTOR TEST
7844
7845  024236                                  50$:
7846                                          ;;*****************************************************************
7847                                          ;INCREMENT TRACK COUNT USING DIAGNOSTIC END OF BLOCK.  START AT TRACK 0,
7848                                          ;LAST SECTOR AND COUNT ONE COMPLETE TRACK CYCLE.
7849  024236  013737  024460  001410                  MOV     110$,RMDA0      ;STARTING TRACK ADDRESS
7850  024244  012737  000400  001140                  MOV     #000400,$GDDAT  ;FIRST VALUE AFTER INCREMENT
7851
7852  024252                                  60$:
7853
7854                                          ;       .CLEAR THE MASSBUS
7855                                          ;       .SET FORMAT
7856                                          ;       .LOAD LAST SECTOR ADDRESS AND TEST TRACK ADDRESS
7857                                          ;       .ENABLE DEBUG CLOCK
7858                                          ;       .SET GO BIT
7859
7860  024252  012760  000040  000010                  MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7861  024260  111160  000010                          MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7862
7863  024264  013760  001434  000032                  MOV     RMOF0,RMOF(R0)  ;LOAD RMOF
7864
7865  024272  013760  001410  000006                  MOV     RMDA0,RMDA(R0)  ;LOAD RMDA
7866
7867  024300  012760  000001  000024                  MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7868
7869  024306  012760  040001  000024                  MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
7870
7871  024314  012760  000001  000000                  MOV     #GO,RMCS1(R0)   ;LOAD RMCS1
7872                                          ;CLOCK RMDA USING DIAGNOSTIC END OF BLOCK
7873
7874  024322  012760  060001  000024                  MOV     #DMD!DBEN!DEBL,RMMR1(R0)        ;LOAD RMMR1
7875
```

```
7876  024330  012760  040001  000024          MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
7877                                  ;       VERIFY RMDA ACCORDING TO $GDDAT
7878
7879  024336  016037  000006  001142          MOV     RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
7880  024344  023737  001140  001142          CMP     $GDDAT,$BDDAT
7881  024352  001402                          BEQ     70$
7882  024354  104133                          ERROR   133             ;TRACK COUNT NOT INCREMENTING
7883  024356  000420                          BR      90$             ;OUT OF SYNC-SKIP TO NEXT
7884  024360                          70$:
7885                                  ;       SETUP FOR NEXT INCREMENT OF RMDA TRACK ADDRESS
7886  024360  105237  001141                  INCB    $GDDAT+1        ;ADVANCE EXPECTED TRACK
7887  024364  123727  001143  000004          CMPB    $BDDAT+1,#4     ;WAS THE LAST TRACK JUST COUNTED??
7888  024372  001002                          BNE     80$             ;NO!!
7889  024374  005037  001140                  CLR     $GDDAT          ;YES-NEXT TRACK, SECTOR SHOULD BE ZERO
7890  024400  013737  001142  001410  80$:    MOV     $BDDAT,RMDAO    ;HAS A FULL CYCLE BEEN COUNTED??
7891  024406  001404                          BEQ     90$             ;YES!!
7892  024410  113737  024460  001410          MOVB    110$,RMDAO      ;INCREMENT FROM LAST SECTOR
7893  024416  000715                          BR      60$
7894  024420                          90$:
7895                                  ;;**************************************************************************
7896                                  ;REPEAT THE TEST IN 16 BIT FORAT
7897  024420  032737  010000  001434          BIT     #FMT16,RMOFO    ;DONE BOTH FORMATS??
7898  024426  001013                          BNE     100$            ;YES!!
7899  024430  012737  010000  001434          MOV     #FMT16,RMOFO    ;SET FORMAT BIT FOR 16
7900  024436  012737  000037  024460          MOV     #037,110$       ;SET LAST SECTOR FOR 16
7901  024444  012737  000001  001140          MOV     #1,$GDDAT       ;SET FIRST COUNT VALUE
7902  024452  000137  024052                  JMP     10$             ;REPEAT TEST
7903
7904  024456  000401                  100$:   BR      120$
7905
7906  024460  000000                  110$:   .WORD                   ;STORAGE FOR LAST SECTOR VALUE
7907
7908  024462                          120$:                           ;END OF TEST
```

```
7909
7910
7911                                    ;;*****************************************************************
7912                                    ;*TEST 40        RMDC COUNT TEST
7913
7914                                    ;;*****************************************************************
7915  024462  000004                    TST40:  SCOPE
7916  024464  012737  000040  001226            MOV     #40,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7917
7918  024472  000240                            NOP
7919  024474  012737  000024  001120            MOV     #20.,$ICNT      ;20 ITERATIONS
7920  024502  112737  000001  001131            MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
7921  024510  012737  024524  001122            MOV     #T40,$LPADR  ;LOAD LOOP ON TEST ADDRESS
7922  024516  012737  024524  001124            MOV     #T40,$LPERR  ;LOAD LOOP CN ERROR ADDRESS
7923  024524                            T40:
7924  024524  012706  001100                    MOV     #STACK,SP       ;LOAD THE STACK POINTER
7925  024530  013700  001276                    MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
7926  024534  013701  001456                    MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
7927  024540  012760  000040  000010            MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7928  024546  111160  000010                    MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7929                                    ;CLEAR RMDC, SET FORMAT AND SETUP PROGRAM PARAMETERS
7930  024552  010037  001136                    MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
7931  024556  062737  000034  001136            ADD     #RMDC,$BDADR
7932  024564  005037  001434                    CLR     RMOF0           ;START WITH 18 BIT FORMAT
7933  024570  012737  002035  001410            MOV     #002035,RMDA0   ;LOAD LAST SECTOR/TRACK VALUE
7934  024576  012737  000001  001140  10$:       MOV     #1,$GDDAT       ;LOAD FIRST INCREMENTAL VALUE
7935
7936  024604  012760  000000  000034            MOV     #0,RMDC(R0)     ;LOAD RMDC
7937
7938  024612  013760  001434  000032            MOV     RMOF0,RMOF(R0)  ;LOAD RMOF
7939  024620  013737  001434  001174            MOV     RMOF0,$TMP0     ;SAVE FOR ERROR MSG
7940  024626                            20$:
7941
7942                                    ;       .CLEAR THE MASSBUS
7943                                    ;       .SET OFFSET
7944                                    ;       .LOAD LAST SECTOR AND TRACK ADDRESS
7945                                    ;       .ENABLE DEBUG CLOCK
7946                                    ;       .SET GO BIT
7947
7948  024626  012760  000040  000010            MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
7949  024634  111160  000010                    MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
7950
7951  024640  013760  001434  000032            MOV     RMOF0,RMOF(R0)  ;LOAD RMOF
7952
7953  024646  013760  001410  000006            MOV     RMDA0,RMDA(R0)  ;LOAD RMDA
7954
7955  024654  012760  000001  000024            MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
7956
7957  024662  012760  040001  000024            MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
7958
7959  024670  012760  000001  000000            MOV     #GO,RMCS1(R0)   ;LOAD RMCS1
7960                                    ;CLOCK THE CYLINDER ADDRESS USING DEBL
7961
7962  024676  012760  060001  000024            MOV     #DMD!DBEN!DEBL,RMMR1(R0)        ;LOAD RMMR1
7963
7964  024704  012760  040001  000024            MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
```

```
7965
7966  024712  016037  000034  001142          MOV     RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
7967  024720  023737  001140  001142          CMP     $GDDAT,$BDDAT
7968  024726  001402                          BEQ     30$             ;BRANCH IF RMDC=RMDC+1
7969  024730  104140                          ERROR   140             ;CANT INCREMENT RMDC
7970  024732  000427                          BR      60$             ;OUT OF SYNC-SKIP TO END
7971  024734                          30$:
7972                                  ;ADVANCE EXPECTED RESULT FOR NEXT INCREMENT
7973  024734  005237  001140                  INC     $GDDAT          ;ADVANCE NEXT RESULT
7974  024740  022737  002000  001140          CMP     #1777+1,$GDDAT  ;SHOULD NEXT VALUE BE ZERO??
7975  024746  001002                          BNE     40$             ;NO!!
7976  024750  005037  001140                  CLR     $GDDAT          ;YES-RMDC SHOULD OVERFLOW
7977  024754  005737  001142          40$:    TST     $BDDAT          ;IS ONE CYCLE COMPLETE??
7978  024760  001401                          BEQ     50$             ;YES!!
7979  024762  000721                          BR      20$             ;CONTINUE
7980  024764                          50$:
7981                                  ;REPEAT TEST IN 16 BIT MODE-ELSE DONE
7982  024764  032737  010000  001434          BIT     #FMT16,RMOF0    ;DONE BOTH MODES??
7983  024772  001007                          BNE     60$             ;YES!!
7984  024774  012737  010000  001434          MOV     #FMT16,RMOF0    ;SET 16 BIT FORMAT
7985  025002  012737  002037  001410          MOV     #002037,RMDA0   ;LOAD LAST SECTOR/TRACK VALUE
7986  025010  000672                          BR      10$             ;REPEAT TEST
7987
7988  025012                          60$:                            ;END OF TEST
7989
7990                                  ;;*******************************************************************
7991                                  ;*TEST 41        LBT TEST
7992
7993                                  ;;*******************************************************************
7994  025012  000004                  TST41:  SCOPE
7995  025014  012737  000041  001226          MOV     #41,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
7996
7997  025022  000240                          NOP
7998  025024  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
7999  025032  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
8000  025040  012737  025054  001122          MOV     #T41,$LPADR  ;LOAD LOOP ON TEST ADDRESS
8001  025046  012737  025054  001124          MOV     #T41,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
8002  025054                          T41:
8003  025054  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
8004  025060  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
8005  025064  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
8006  025070  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
8007  025074  062737  000012  001136          ADD     #RMDS,$BDADR
8008                                  ;       CLEAR THE MASSBUS
8009                                  ;       SET FORMAT
8010                                  ;       LOAD LAST TRACK AND SECTOR
8011                                  ;       LOAD LAST CYLINDER
8012
8013                                  ;       VERIFY THAT "LBT" IS RESET
8014  025102  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
8015  025110  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
8016
8017  025114  012760  000000  000032          MOV     #0,RMOF(R0)     ;LOAD RMOF
8018
8019  025122  012760  002035  000006          MOV     #002035,RMDA(R0)        ;LOAD RMDA
8020
```

```
8021  025130  012760  001466  000034          MOV     #822.,RMDC(R0)  ;LOAD RMDC
8022
8023  025136  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
8024  025144  042737  175777  001142          BIC     #^CLBT,$BDDAT
8025  025152  001403                          BEQ     10$             ;BRANCH IF LBT IS RESET
8026  025154  005037  001140                  CLR     $GDDAT          ;LBT SHOULD BE ZERO
8027  025160  104141                          ERROR   141             ;CANNOT RESET 'LBT'
8028
8029  025162                          10$:
8030                                  ;       ENABLE DEBUG CLOCK
8031                                  ;       SET GO
8032                                  ;       FORCE EBL
8033                                  ;
8034                                  ;       VERIFY THAT LBT IS SET
8035
8036  025162  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
8037
8038  025170  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
8039
8040  025176  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
8041
8042  025204  012760  000001  000000          MOV     #GO,RMCS1(R0)   ;LOAD RMCS1
8043
8044  025212  012760  060001  000024          MOV     #DMD!DBEN!DEBL,RMMR1(R0)        ;LOAD RMMR1
8045
8046  025220  012760  040001  000024          MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
8047
8048  025226  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
8049  025234  042737  175777  001142          BIC     #^CLBT,$BDDAT
8050  025242  001004                          BNE     20$             ;BRANCH IF LBT IS SET
8051  025244  012737  002000  001140          MOV     #LBT,$GDDAT
8052  025252  104142                          ERROR   142             ;CANT SET LBT
8053
8054  025254                          20$:                            ;END OF TEST
8055
8056                                  ;;**********************************************************************
8057                                  ;*TEST 42        COMPOSITE ERROR TEST
8058
8059                                  ;;**********************************************************************
8060  025254  000004                  TST42:  SCOPE
8061  025256  012737  000042  001226          MOV     #42,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
8062
8063  025264  000240                          NOP
8064  025266  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
8065  025274  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
8066  025302  012737  025316  001122          MOV     #T42,$LPADR  ;LOAD LOOP ON TEST ADDRESS
8067  025310  012737  025316  001124          MOV     #T42,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
8068  025316                          T42:
8069  025316  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
8070  025322  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
8071  025326  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
8072  025332  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
8073  025340  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
8074  025344  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
8075  025350  062737  000012  001136          ADD     #RMDS,$BDADR
8076                                  ;USING DIAGNOSTIC MODE, CLEAR ALL ERRORS AND VERIFY THAT COMPOSITE
```

```
                                        ;ERROR IS RESET.
                                        
025356  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
                                        
025364  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
                                        
025372  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
                                        
025400  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
025406  042737  137777  001142          BIC     #^CERR,$BDDAT
025414  001403                          BEQ     10$             ;BRANCH IF ERR IS RESET
025416  005037  001140                  CLR     $GDDAT
025422  104143                          ERROR   143             ;CANT READ ZERO FROM ERR
025424  012737  040000  001140  10$:    MOV     #ERR,$GDDAT
                                        ;SET BOTH ERROR REGISTERS AND VERIFY THAT COMPOSITE ERROR IS SET
                                        
025432  012760  177777  000014          MOV     #-1,RMER1(R0)   ;LOAD RMER1
                                        
025440  012760  177777  000042          MOV     #-1,RMER2(R0)   ;LOAD RMER2
                                        
025446  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
025454  042737  137777  001142          BIC     #^CERR,$BDDAT
025462  001001                          BNE     20$             ;BRANCH IF ERR IS SET
025464  104144                          ERROR   144             ;CANT SET ERR
025466                          20$:
                                        ;VERIFY THAT COMPOSITE ERROR SETS FOR EACH BIT OF RMER1
025466  012702  000001                  MOV     #1,R2           ;INITIALIZE TEST PATTERN
025472                          30$:
                                        ;       WRITE THE TEST PATTERN AND VERIFY THAT ERR IS SET
025472  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
025500  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
                                        
025504  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
                                        
025512  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
                                        
025520  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
                                        
025526  010260  000014                  MOV     R2,RMER1(R0)    ;LOAD RMER1
                                        
025532  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
025540  042737  137777  001142          BIC     #^CERR,$BDDAT
025546  001005                          BNE     40$             ;BRANCH IF COMPOSITE ERROR SET
025550  010237  001174                  MOV     R2,$TMP0        ;SAVE RMER1 TEST PATTERN
025554  005037  001176                  CLR     $TMP1           ;SAVE RMER2 TEST PATTERN
025560  104145                          ERROR   145             ;ERR NOT SET BY RMER1 ERROR
025562                          40$:
                                        ;       ADVANCE THE TEST PATTERN FOR RMER1
025562  006302                          ASL     R2
025564  001342                          BNE     30$             ;CONTINUE IF TEST NOT DONE
025566                          50$:
                                        ;VERIFY THAT COMPOSITE ERROR SETS FOR EACH BIT OF RMER2
025566  012702  000001                  MOV     #1,R2           ;INITIALIZE TEST PATTERN
025572                          60$:
                                        ;       WRITE THE TEST PATTERN AND VERIFY THAT ERR IS SET
025572  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
```

```
8133  025600  111160  000010                    MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
8134
8135  025604  012760  000001  000024            MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
8136
8137  025612  012760  000000  000014            MOV     #0,RMER1(R0)    ;LOAD RMER1
8138
8139  025620  012760  000000  000042            MOV     #0,RMER2(R0)    ;LOAD RMER2
8140
8141  025626  010260  000042                    MOV     R2,RMER2(R0)    ;LOAD RMER2
8142
8143  025632  016037  000012  001142            MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
8144  025640  042737  137777  001142            BIC     #^CERR,$BDDAT
8145  025646  012737  040000  001140            MOV     #ERR,$GDDAT     ;SETUP EXPECTED VALUE FOR COMP ERROR
8146  025654  032702  001567                    BIT     #XNUER2,R2
8147  025660  001402                            BEQ     65$             ;BRANCH IF TEST BIT IS A USED BIT
8148  025662  005037  001140                    CLR     $GDDAT          ;TEST BIT IS NOT USED - ERR SHOULD BE 0
8149  025666  023737  001140  001142  65$:     CMP     $GDDAT,$BDDAT
8150  025674  001405                            BEQ     70$             ;BRANCH IF COMP ERROR IS OK
8151  025676  005037  001174                    CLR     $TMP0           ;SAVE RMER1 TEST PATTERN
8152  025702  010237  001176                    MOV     R2,$TMP1        ;SAVE RMER2 TEST PATTERN
8153  025706  104145                            ERROR   145             ;ERR NOT SET BY RMER2 ERROR
8154  025710                          70$:
8155                                  ;       ADVANCE THE TEST PATTERN FOR RMER2
8156  025710  006302                            ASL     R2
8157  025712  001327                            BNE     60$             ;CONTINUE IF TEST NOT DONE
8158
8159  025714                          80$:                            ;END OF TEST
8160
8161                                  ;;**********************************************************************
8162                                  ;*TEST 43        WRITE GO TEST
8163
8164                                  ;;**********************************************************************
8165  025714  000004                  TST43:  SCOPE
8166  025716  012737  000043  001226          MOV     #43,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
8167
8168  025724  000240                          NOP
8169  025726  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
8170  025734  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
8171  025742  012737  025756  001122          MOV     #T43,$LPADR  ;LOAD LOOP ON TEST ADDRESS
8172  025750  012737  025756  001124          MOV     #T43,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
8173  025756                          T43:
8174  025756  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
8175  025762  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
8176  025766  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
8177  025772  010037  001136                  MOV     R0,$BDADR       ;COPY RMCS1 ADDRESS
8178  025776  005002                          CLR     R2              ;INITIALIZE FUNCTION CODE
8179  026000                          10$:
8180
8181                                  ;CLEAR THE MASSBUS, SET DIAGNOSTIC MODE AND ENABLE DEBUG CLOCK
8182  026000  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
8183  026006  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
8184
8185  026012  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
8186
8187  026020  012760  041001  000024          MOV     #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
8188
```

```
 8189  026026  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
 8190
 8191  026034  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
 8192                          ;TRANSFER THE FUNCTION CODE AND GO BIT TO RMCS1, VERIFY GO IS SET
 8193  026042  010203                          MOV     R2,R3           ;SETUP FUNCTION CODE
 8194  026044  052703  000001                  BIS     #GO,R3
 8195
 8196  026050  010360  000000                  MOV     R3,RMCS1(R0)    ;LOAD RMCS1
 8197
 8198  026054  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
 8199  026062  032737  000001  001142          BIT     #GO,$BDDAT
 8200  026070  001007                          BNE     20$             ;BRANCH IF GO IS SET
 8201                          ;REPORT THE ERROR-CANT SET GO WITH THIS FUNCTION CODE
 8202  026072  042737  177700  001142          BIC     #^CFNCMSK,$BDDAT
 8203  026100  010337  001140                  MOV     R3,$GDDAT       ;SAVE FUNCTION CODE
 8204  026104  104146                          ERROR   146             ;CANT SET GO
 8205  026106  000405                          BR      30$
 8206  026110                        20$:
 8207                          ;ADVANCE R2 TO THE NEXT FUNCTION CODE
 8208  026110  062702  000002                  ADD     #2,R2
 8209  026114  022702  000076                  CMP     #ILF76,R2
 8210  026120  103327                          BHIS    10$
 8211
 8212  026122                        30$:                            ;END OF TEST
 8213
 8214                          ;;*****************************************************************
 8215                          ;*TEST 44        BRANCH MULTIPLEXOR TEST
 8216
 8217                          ;;*****************************************************************
 8218  026122  000004          TST44:  SCOPE
 8219  026124  012737  000044  001226          MOV     #44,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 8220
 8221  026132  000240                          NOP
 8222  026134  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
 8223  026142  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
 8224  026150  012737  026164  001122          MOV     #T44,$LPADR  ;LOAD LOOP ON TEST ADDRESS
 8225  026156  012737  026164  001124          MOV     #T44,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
 8226  026164                        T44:
 8227  026164  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
 8228  026170  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
 8229  026174  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
 8230  026200  010037  001136                  MOV     R0,$BDADR       ;COPY REGISTER ADDRESS
 8231  026204  062737  000040  001136          ADD     #RMMR2,$BDADR
 8232  026212  012702  026444                  MOV     #100$,R2        ;INITIALIZE TABLE POINTER
 8233                          ;CLEAR THE MASSBUS AND SET DEBUG CLOCK ENABLE
 8234  026216                        10$:
 8235  026216  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
 8236  026224  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
 8237
 8238  026230  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
 8239
 8240  026236  012760  041001  000024          MOV     #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
 8241
 8242  026244  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
 8243
 8244  026252  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
```

```
                                        ;THE TEST BIT SHOULD BE ONE BECAUSE THE ADDRESS IS ALL ONES WHEN
                                        ;THE COMMAND SEQUENCER IS INITIALIZED.

026260  016037  000040  001142            MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
026266  032737  010000  001142            BIT     #TST,$BDDAT
026274  001010                            BNE     15$             ;BRANCH IF TEST BIT IS ON
026276  042737  167777  001142            BIC     #^CTST,$BDDAT   ;SETUP FOR ERROR TYPE
026304  012737  010000  001140            MOV     #TST,$GDDAT
026312  104147                            ERROR   147             ;TEST BIT NOT SET
026314  000452                            BR      40$             ;SKIP REST OF TEST
026316                                  15$:
                                        ;GET THE FUNCTION CODE FROM THE TABLE AND TRANSFER IT TO THE DEVICE,
                                        ;THEN STEP THE COMMAND SEQUENCER ACCORDING TO THE TABLE.

026316  111203                            MOVB    (R2),R3
026320  052703  000001                    BIS     #GO,R3
026324  042703  177700                    BIC     #^CFNCMSK,R3    ;R3=FUNCTION CODE, GO BIT

026330  010360  000000                    MOV     R3,RMCS1(R0)    ;LOAD RMCS1
026334  010337  001174                    MOV     R3,$TMP0        ;SAVE R3 FOR ERROR MSG

026340  116203  000001                    MOVB    1(R2),R3        ;GET CLOCK COUNT IN R3
026344  042703  177400                    BIC     #^C377,R3
026350                                  20$:

026350  012760  141001  000024            MOV     #DMD!DBEN!MUR!DBCK,RMMR1(R0)    ;LOAD RMMR1

026356  012760  041001  000024            MOV     #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
026364  005303                            DEC     R3              ;DECREMENT CLOCK COUNT
026366  001370                            BNE     20$             ;ISSUE CLOCKS TILL ZERO

                                        ;GET THE TEST BIT AND COMPARE IT WITH THE TABLE ENTRY

026370  016037  000040  001142            MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
026376  042737  167777  001142            BIC     #^CTST,$BDDAT
026404  016237  000002  001140            MOV     2(R2),$GDDAT
026412  023737  001140  001142            CMP     $GDDAT,$BDDAT
026420  001402                            BEQ     30$             ;BRANCH IF TEST BIT OK
026422  104150                            ERROR   150             ;TEST BIT IS INCORRECT
026424  000406                            BR      40$             ;SKIP REST OF TEST
026426                                  30$:
                                        ;MOVE THE TABLE POINTER AND CONTINUE IF NEXT ENTRY POSITIVE
026426  062702  000004                    ADD     #4,R2
026432  105762  000001                    TSTB    1(R2)
026436  100401                            BMI     40$             ;BRANCH IF DONE TEST
026440  000666                            BR      10$             ;REPEAT TEST
026442  000436                        40$:BR      200$            ;JUMP OVER TABLE

026444                                  100$:

                                        ;TABLE OF FUNCTION CODES, CLOCK COUNTS, AND TEST BITS

026444      000                         .BYTE   NOP             ;MUX ADDRESS=DATA COMMAND
026445      001                         .BYTE   1
026446  000000                          .WORD   0               ;TEST BIT=0
```

```
8301  026450  000                    .BYTE   NOP             ;MUX ADDRESS=UNIT READY
8302  026451  002                    .BYTE   2
8303  026452  000000                 .WORD   0               ;TEST BIT=0
8304
8305  026454  010                    .BYTE   DRVCLR          ;MUX ADDRESS=F4
8306  026455  001                    .BYTE   1
8307  026456  010000                 .WORD   TST             ;TEST BIT=1
8308
8309  026460  050                    .BYTE   WCD             ;MUX ADDRESS=F4
8310  026461  001                    .BYTE   1
8311  026462  000000                 .WORD   0               ;TEST BIT=0
8312
8313  026464  012                    .BYTE   RLEASE          ;MUX ADDRESS=F4
8314  026465  001                    .BYTE   1
8315  026466  010000                 .WORD   TST             ;TEST BIT=1
8316
8317  026470  052                    .BYTE   WCH             ;MUX ADDRESS=F4
8318  026471  001                    .BYTE   1
8319  026472  000000                 .WORD   0               ;TEST BIT=0
8320
8321  026474  020                    .BYTE   RIP             ;MUX ADDRESS=F4
8322  026475  001                    .BYTE   1
8323  026476  010000                 .WORD   TST             ;TEST BIT=1
8324
8325  026500  060                    .BYTE   WD              ;MUX ADDRESS=F4
8326  026501  001                    .BYTE   1
8327  026502  000000                 .WORD   0               ;TEST BIT=0
8328
8329  026504  022                    .BYTE   PAKACK          ;MUX ADDRESS=F4
8330  026505  001                    .BYTE   1
8331  026506  010000                 .WORD   TST             ;TEST BIT=1
8332
8333  026510  062                    .BYTE   WH              ;MUX ADDRESS=F4
8334  026511  001                    .BYTE   1
8335  026512  000000                 .WORD   0               ;TEST BIT=0
8336
8337  026514  030                    .BYTE   SEARCH          ;MUX ADDRESS=F4
8338  026515  001                    .BYTE   1
8339  026516  010000                 .WORD   TST             ;TEST BIT=1
8340
8341  026520  070                    .BYTE   RD              ;MUX ADDRESS=F4
8342  026521  001                    .BYTE   1
8343  026522  000000                 .WORD   0               ;TEST BIT=0
8344
8345  026524  032                    .BYTE   ILF32           ;MUX ADDRESS=F4
8346  026525  001                    .BYTE   1
8347  026526  010000                 .WORD   TST             ;TEST BIT=1
8348
8349  026530  072                    .BYTE   RH              ;MUX ADDRESS=F4
8350  026531  001                    .BYTE   1
8351  026532  000000                 .WORD   0               ;TEST BIT=0
8352
8353  026534  000                    .BYTE                   ;END OF TABLE
8354  026535  377                    .BYTE   #-1
8355  026536  000000                 .WORD
8356  026540              200$:                              ;END OF TEST
```

```
8357
8358                                    ;;********************* ***************************************************
8359                                    ;*TEST 45        SET/RESET GO TEST
8360
8361                                    ;;*****************************************************************
8362  026540  000004                    TST45:  SCOPE
8363  026542  012737  000045  001226            MOV     #45,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
8364
8365  026550  000240                            NOP
8366  026552  012737  000024  001120            MOV     #20.,$ICNT      ;20 ITERATIONS
8367  026560  112737  000001  001131            MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
8368  026566  012737  026602  001122            MOV     #T45,$LPADR  ;LOAD LOOP ON TEST ADDRESS
8369  026574  012737  026602  001124            MOV     #T45,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
8370  026602                            T45:
8371  026602  012706  001100                    MOV     #STACK,SP       ;LOAD THE STACK POINTER
8372  026606  013700  001276                    MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
8373  026612  013701  001456                    MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
8374  026616  012702  027240                    MOV     #200$,R2        ;INITIALIZE FUNCTION CODE POINTER
8375                                    ;CLEAR, THEN SET DIAGNOSTIC MODE, CLEAR COMPOSITE ERROR, SET MEDIUM
8376                                    ;ON LINE AND ENABLE DEBUG CLOCK
8377  026622                            10$:
8378  026622  012760  000040  000010            MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
8379  026630  111160  000010                    MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
8380
8381  026634  012760  000001  000024            MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
8382
8383  026642  012760  041001  000024            MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
8384
8385  026650  012760  000000  000014            MOV     #0,RMER1(R0)    ;LOAD RMER1
8386
8387  026656  012760  000000  000042            MOV     #0,RMER2(R0)    ;LOAD RMER2
8388                                    ;TRANSFER THE FUNCTION CODE AND GO BIT TO RMCS1 AND VERIFY GO IS SET
8389  026664  111203                            MOVB    (R2),R3         ;GET FUNCTION CODE
8390  026666  042703  177701                    BIC     #^C1LF76,R3     ;CLEAR UNUSED BITS
8391  026672  052703  000001                    BIS     #GO,R3          ;SET GO
8392
8393  026676  010360  000000                    MOV     R3,RMCS1(R0)    ;LOAD RMCS1
8394
8395  026702  016037  000000  001142            MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
8396  026710  032737  000001  001142            BIT     #GO,$BDDAT
8397  026716  001011                            BNE     20$             ;BRANCH IF GO IS SET
8398  026720  042737  177700  001142            BIC     #^CFNCMSK,$BDDAT
8399  026726  010337  001140                    MOV     R3,$GDDAT       ;SAVE EXPECTED RESULT
8400  026732  010037  001136                    MOV     R0,$BDADR       ;COPY REGISTER ADDRESS
8401  026736  104151                            ERROR   151             ;CANT SET GO
8402  026740  000536                            BR      100$
8403
8404  026742                            20$:
8405                                    ;GET READY STATUS AND VERIFY THAT IT IS THE COMPLEMENT OF GO
8406  026742  005037  001140                    CLR     $GDDAT          ;EXPECT DRY TO BE OFF
8407  026746  032737  000001  001142            BIT     #GO,$BDDAT      ;WAS GO SET??
8408  026754  001003                            BNE     30$             ;YES!!
8409  026756  012737  000200  001140            MOV     #DRY,$GDDAT     ;GO WAS NOT SET, DRY SHOULD BE
8410  026764                            30$:
8411
8412  026764  016037  000012  001142            MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
```

```
8413  026772  042737  177577  001142          BIC     #^CDRY,$BDDAT
8414  027000  023737  001140  001142          CMP     $GDDAT,$BDDAT
8415  027006  001406                          BEQ     40$                 ;BRANCH IF DRY IS OK
8416  027010  010037  001136                  MOV     R0,$BDADR           ;COPY REGISTER ADDRESS
8417  027014  062737  000012  001136          ADD     #RMDS,$BDADR
8418  027022  104152                          ERROR   152                 ;DRY NOT COMPLEMENT OF GO
8419  027024                          40$:
8420                                  ;STEP THE DEBUG CLOCK AND VERIFY THAT GO REMAINS SET
8421  027024  116204  000001                  MOVB    1(R2),R4            ;GET NUMBER OF CLOCK CYCLES
8422  027030  042704  177400                  BIC     #^C377,R4
8423  027034                          50$:
8424
8425  027034  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
8426
8427  027042  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
8428
8429  027050  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
8430  027056  042737  177700  001142          BIC     #^CFNCMSK,$BDDAT  ;CLEAR UNUSED BITS
8431  027064  010037  001136                  MOV     R0,$BDADR           ;SETUP REGISTER ADDRESS
8432  027070  010337  001140                  MOV     R3,$GDDAT           ;SAVE EXPECTED RESULT
8433                                  ;DECREMENT CLOCK COUNT AND EXIT LOOP IF ZERO
8434  027074  005304                          DEC     R4
8435  027076  001406                          BEQ     60$
8436  027100  032737  000001  001142          BIT     #GO,$BDDAT          ;IS GO STILL SET??
8437  027106  001352                          BNE     50$                 ;YES!!
8438  027110  104153                          ERROR   153                 ;GO RESET EARLY
8439  027112  000451                          BR      100$                ;OUT OF SYNC=SKIP TO NEXT
8440  027114                          60$:
8441                                  ;GO SHOULD NOW BE RESET AND DRY SHOULD BE SET
8442  027114  032737  000001  001142          BIT     #GO,$BDDAT          ;IS GO RESET??
8443  027122  001405                          BEQ     70$                 ;YES!!
8444  027124  042737  000001  001140          BIC     #GO,$GDDAT          ;SETUP EXPECTED RESULT
8445  027132  104154                          ERROR   154                 ;GO DID NOT RESET
8446  027134  000440                          BR      100$
8447  027136                          70$:
8448  027136  012737  000200  001140          MOV     #DRY,$GDDAT         ;EXPECT DRIVE READY TO BE SET
8449  027144  032737  000001  001142          BIT     #GO,$BDDAT          ;DID GO RESET??
8450  027152  001402                          BEQ     80$                 ;YES!!
8451  027154  005037  001140                  CLR     $GDDAT              ;GO IS SET-
8452  027160                          80$:
8453
8454  027160  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
8455  027166  042737  177577  001142          BIC     #^CDRY,$BDDAT
8456  027174  010037  001136                  MOV     R0,$BDADR           ;COPY REGISTER ADDRESS
8457  027200  062737  000012  001136          ADD     #RMDS,$BDADR
8458  027206  023737  001140  001142          CMP     $GDDAT,$BDDAT       ;IS DRIVE READY OK??
8459  027214  001401                          BEQ     90$                 ;YES!!
8460  027216  104152                          ERROR   152                 ;DRY NOT COMPLEMENT OF GO
8461  027220                          90$:
8462                                  ;ADVANCE TO THE NEXT FUNCTION CODE TO BE TESTED-EXIT IF DONE
8463  027220  062702  000002                  ADD     #2,R2               ;MOVE TABLE POINTER
8464  027224  105762  000001                  TSTB    1(R2)               ;END OF TABLE??
8465  027230  100402                          BMI     100$                ;YES!!
8466  027232  000137  026622                  JMP     10$                 ;TEST THIS FUNCTION CODE
8467  027236  000423                  100$:   BR      300$                ;GO TO NEXT TEST
8468
```

```
8469  027240                  200$:
8470                          ;;**********************************************************
8471                          ;TABLE OF FUNCTION CODES AND CLOCK COUNTS USED DURING TEST
8472  027240  002                   .BYTE   ILF02           ;ILLEGAL FUNCTION CODE #2
8473  027241  001                   .BYTE   1
8474
8475  027242  004                   .BYTE   SEEK            ;SEEK COMMAND
8476  027243  001                   .BYTE   1
8477
8478  027244  006                   .BYTE   RECAL           ;RECALIBRATE COMMAND
8479  027245  001                   .BYTE   1
8480
8481  027246  014                   .BYTE   OFFSET          ;OFFSET COMMAND
8482  027247  001                   .BYTE   1
8483
8484  027250  016                   .BYTE   RTC             ;RETURN TO CENTER LINE COMMAND
8485  027251  001                   .BYTE   1
8486
8487  027252  024                   .BYTE   ILF24           ;ILLEGAL FUNCTION CODE #24
8488  027253  001                   .BYTE   1
8489
8490  027254  026                   .BYTE   ILF26           ;ILLEGAL FUNCTION CODE #26
8491  027255  001                   .BYTE   1
8492
8493  027256  034                   .BYTE   ILF34           ;ILLEGAL FUNCTION CODE #34
8494  027257  001                   .BYTE   1
8495
8496  027260  036                   .BYTE   ILF36           ;ILLEGAL FUNCTION CODE #36
8497  027261  001                   .BYTE   1
8498
8499  027262  042                   .BYTE   ILF42           ;ILLEGAL FUNCTION CODE #42
8500  027263  001                   .BYTE   1
8501
8502  027264  044                   .BYTE   ILF44           ;ILLEGAL FUNCTION CODE #44
8503  027265  001                   .BYTE   1
8504
8505  027266  046                   .BYTE   ILF46           ;ILLEGAL FUNCTION CODE #46
8506  027267  001                   .BYTE   1
8507
8508  027270  054                   .BYTE   ILF54           ;ILLEGAL FUNCTION CODE #54
8509  027271  001                   .BYTE   1
8510
8511  027272  056                   .BYTE   ILF56           ;ILLEGAL FUNCTION CODE #56
8512  027273  001                   .BYTE   1
8513
8514  027274  064                   .BYTE   ILF64           ;ILLEGAL FUNCTION CODE #64
8515  027275  001                   .BYTE   1
8516
8517  027276  066                   .BYTE   ILF66           ;ILLEGAL FUNCTION CODE #66
8518  027277  001                   .BYTE   1
8519
8520  027300  074                   .BYTE   ILF74           ;ILLEGAL FUNCTION CODE #74
8521  027301  001                   .BYTE   1
8522
8523  027302  076                   .BYTE   ILF76           ;ILLEGAL FUNCTION CODE #76
8524  027303  001                   .BYTE   1
```

```
8525
8526  027304  000                       .BYTE                   ;END OF TABLE
8527  027305  377                       .BYTE   #-1
8528
8529  027306                    300$:                           ;END OF TEST
8530
8531                            ;;*****************************************************************
8532                            ;*TEST 46        END 1 RESET GO TEST
8533
8534                            ;;*****************************************************************
8535  027306  000004            TST46:  SCOPE
8536  027310  012737  000046  001226    MOV     #46,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
8537
8538  027316  000240                    NOP
8539  027320  012737  000024  001120    MOV     #20.,$ICNT      ;20 ITERATIONS
8540  027326  112737  000001  001131    MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
8541  027334  012737  027350  001122    MOV     #T46,$LPADR  ;LOAD LOOP ON TEST ADDRESS
8542  027342  012737  027350  001124    MOV     #T46,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
8543  027350                    T46:
8544  027350  012706  001100            MOV     #STACK,SP       ;LOAD THE STACK POINTER
8545  027354  013700  001276            MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
8546  027360  013701  001456            MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
8547  027364  012702  027630            MOV     #100$,R2        ;INITIALIZE TABLE POINTER
8548  027370  010037  001136            MOV     R0,$BDADR       ;COPY RMCS1 ADDRESS
8549
8550  027374                    10$:
8551                            ;CLEAR MASSBUS, THEN SET MEDIUM ON LINE AND ENABLE DEBUG CLOCK
8552  027374  012760  000040  000010    MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
8553  027402  111160  000010            MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
8554
8555  027406  012760  000001  000024    MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
8556
8557  027414  012760  041001  000024    MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
8558
8559  027422  012760  000000  000014    MOV     #0,RMER1(R0)    ;LOAD RMER1
8560
8561  027430  012760  000000  000042    MOV     #0,RMER2(R0)    ;LOAD RMER2
8562                            ;TRANSFER THE FUNCTION CODE AND GO BIT TO RMCS1, VERIFY GO IS SET
8563  027436  111203                    MOVB    (R2),R3         ;GET FUNCTION CODE FROM
8564  027440  042703  177701            BIC     #^CILF76,R3     ;TABLE AND SET GO
8565  027444  052703  000001            BIS     #GO,R3
8566  027450  010337  001140            MOV     R3,$GDDAT       ;SAVE FUNCTION CODE FOR MSG
8567
8568  027454  010360  000000            MOV     R3,RMCS1(R0)    ;LOAD RMCS1
8569
8570  027460  016037  000000  001142    MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
8571  027466  032737  000001  001142    BIT     #GO,$BDDAT
8572  027474  001005                    BNE     20$             ;BRANCH IF GO IS SET
8573  027476  042737  177700  001142    BIC     #^CFNCMSK,$BDDAT
8574  027504  104151                    ERROR   151             ;GO DID NOT SET
8575  027506  000447                    BR      60$             ;OUT OF SYNC-SKIP
8576  027510                    20$:
8577                            ;GET THE NUMBER OF CLOCK CYCLES FROM THE TABLE, SAVE EXPECTED STATUS
8578  027510  116204  000001            MOVB    1(R2),R4        ;R4=CLOCK COUNT
8579  027514  042704  177400            BIC     #^C377,R4
8580  027520                    30$:
```

```
8581                                    ;STEP THE DEBUG CLOCK AND VERIFY GO STATUS ON UNTIL CLOCK COUNT EXPIRES.
8582
8583  027520  012760  141001  000024            MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
8584
8585  027526  012760  041001  000024            MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
8586
8587  027534  016037  000000  001142            MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
8588  027542  042737  177700  001142            BIC     #^CFNCMSK,$BDDAT
8589  027550  005304                            DEC     R4
8590  027552  001406                            BEQ     40$                     ;BRANCH IF GO SHOULD BE OFF
8591  027554  032737  000001  001142            BIT     #GO,$BDDAT
8592  027562  001356                            BNE     30$                     ;CONTINUE IF GO IS ON
8593  027564  104153                            ERROR   153                     ;GO RESET EARLY
8594  027566  000417                            BR      60$                     ;OUT OF SYNC-SKIP
8595  027570                                40$:
8596                                        ;VERIFY THAT GO RESET AT END1
8597  027570  032737  000001  001142            BIT     #GO,$BDDAT              ;DID GO RESET??
8598  027576  001405                            BEQ     50$                     ;YES!!
8599  027600  042737  000001  001140            BIC     #GO,$GDDAT
8600  027606  104154                            ERROR   154                     ;GO NOT RESET AT END1
8601  027610  000406                            BR      60$
8602  027612                                50$:
8603                                        ;GET THE NEXT FUNCTION CODE FROM THE TABLE
8604  027612  062702  000002                    ADD     #2,R2
8605  027616  105762  000001                    TSTB    1(R2)
8606  027622  100401                            BMI     60$                     ;BRANCH IF END OF TABLE
8607  027624  000663                            BR      10$                     ;TEST THIS FUNCTION CODE
8608  027626  000404                        60$:BR      200$                    ;JUMP OVER TABLE
8609
8610  027630                                100$:
8611                                        ;TABLE OF FUNCTION CODES AND CLOCK COUNTS USED DURING TEST
8612
8613  027630       012                            .BYTE   RLEASE          ;RELEASE COMMAND
8614  027631       002                            .BYTE   2
8615
8616  027632       030                            .BYTE   SEARCH          ;SEARCH COMMAND
8617  027633       002                            .BYTE   2
8618
8619  027634       032                            .BYTE   ILF32           ;ILLEGAL FUNCTON #32
8620  027635       002                            .BYTE   2
8621
8622  027636       000                            .BYTE                   ;END OF TABLE
8623  027637       377                            .BYTE   #-1
8624  027640                                200$:                           ;END OF TEST
8625
8626                                        ;;**************************************************
8627                                        ;*TEST 47        SET PULSE TEST
8628
8629                                        ;;**************************************************
8630  027640  000004                        TST47:  SCOPE
8631  027642  012737  000047  001226                MOV     #47,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
8632
8633  027650  000240                                NOP
8634  027652  012737  000024  001120                MOV     #20.,$ICNT      ;20 ITERATIONS
8635  027660  112737  000001  001131                MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
8636  027666  012737  027702  001122                MOV     #T47,$LPADR  ;LOAD LOOP ON TEST ADDRESS
```

```
8637  027674  012737  027702  001124          MOV     #T47,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
8638  027702                              T47:
8639  027702  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
8640  027706  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
8641  027712  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
8642  027716  010037  001136                  MOV     R0,$BDADR       ;COPY REG ADDRESS FOR MSG
8643  027722  062737  000024  001136          ADD     #RMMR1,$BDADR
8644  027730  012702  030174                  MOV     #100$,R2        ;INITIALIZE TABLE POINTER
8645  027734                              10$:
8646                                      ;CLEAR THE MASS BUS, ENABLE DEBUG CLOCK, AND RESET ERROR REGISTERS
8647  027734  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
8648  027742  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
8649
8650  027746  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
8651
8652  027754  012760  041001  000024          MOV     #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
8653
8654  027762  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
8655
8656  027770  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
8657                                      ;VERIFY THAT CONTINUE, ''CONT'' IS RESET AFTER CLEAR
8658
8659  027776  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
8660  030004  042737  177677  001142          BIC     #^CCONT,$BDDAT
8661  030012  001404                          BEQ     20$             ;BRANCH IF CONT WAS CLEARED
8662  030014  005037  001140                  CLR     $GDDAT          ;FOR ERROR MSG
8663  030020  104155                          ERROR   155             ;CANT CLEAR CONTINUE
8664  030022  000463                          BR      70$
8665  030024                              20$:
8666                                      ;GET THE FUNCTION CODE FROM THE TABLE AND TRANSFER IT TO RMCS1
8667  030024  111203                          MOVB    (R2),R3
8668  030026  052703  000001                  BIS     #GO,R3
8669  030032  042703  177700                  BIC     #^CFNCMSK,R3    ;R3=FUNCTION CODE AND GO
8670
8671  030036  010360  000000                  MOV     R3,RMCS1(R0)    ;LOAD RMCS1
8672  030042  010337  001174                  MOV     R3,$TMP0        ;SAVE FUNCTION CODE FOR MSG
8673                                      ;GET THE CLOCK COUNT FROM THE TABLE
8674  030046  116203  000001                  MOVB    1(R2),R3
8675  030052  042703  177400                  BIC     #^C377,R3
8676                                      ;GET THE BIT STREAM FOR CONTINUE FROM THE TABLE
8677  030056  016204  000002                  MOV     2(R2),R4
8678  030062                              30$:
8679                                      ;STEP THE COMMAND SEQUENCER AND VERIFY CONTINUE STATUS
8680
8681  030062  012760  141001  000024          MOV     #DMD!DBEN!MUR!DBCK,RMMR1(R0)    ;LOAD RMMR1
8682
8683  030070  012760  041001  000024          MOV     #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
8684
8685  030076  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
8686  030104  042737  177677  001142          BIC     #^CCONT,$BDDAT
8687  030112  005037  001140                  CLR     $GDDAT          ;GENERATE EXPECTED CONTINUE
8688  030116  032704  000001                  BIT     #BIT0,R4
8689  030122  001403                          BEQ     40$
8690  030124  012737  000100  001140          MOV     #CONT,$GDDAT
8691  030132  023737  001140  001142  40$:    CMP     $GDDAT,$BDDAT
8692  030140  001402                          BEQ     50$             ;BRANCH IF CONTINUE IS OK
```

```
                        T47     SET PULSE TEST

 8693  030142  104156                 ERROR   156             ;CONTINUE IS INCORRECT
 8694  030144  000412                 BR      70$             ;SKIP
 8695  030146                 50$:
 8696                           ;DECREMENT CLOCK COUNT AND SHIFT BIT STREAM
 8697  030146  005303                 DEC     R3
 8698  030150  001402                 BEQ     60$             ;BRANCH IF CLOCK COUNT EXPIRED
 8699  030152  006204                 ASR     R4              ;SHIFT TO NEXT CONTINUE BIT
 8700  030154  000742                 BR      30$             ;TEST NEXT CLOCK CYCLE
 8701  030156                 60$:
 8702                           ;ADVANCE TABLE POINTER-EXIT IF DONE
 8703  030156  062702  000004         ADD     #4,R2
 8704  030162  105762  000001         TSTB    1(R2)
 8705  030166  100401                 BMI     70$             ;EXIT IF CLOCK COUNT NEGATIVE
 8706  030170  000661                 BR      10$             ;CONTINUE TEST
 8707  030172  000442         70$:    BR      200$            ;JUMP OVER TABLE
 8708
 8709  030174                 100$:
 8710                           ;TABLE OF FUNCTION CODES, CLOCK COUNTS AND CONTINUE BITS FOR TEST
 8711
 8712  030174      000                .BYTE   NOP             ;NOP COMMAND
 8713  030175      004                .BYTE   4               ;4 CLOCKS
 8714  030176  000000                 .WORD   ^B0000          ;CONTINUE=0000
 8715
 8716  030200      002                .BYTE   ILF02           ;ILLEGAL FUNCTION 2
 8717  030201      002                .BYTE   2
 8718  030202  000000                 .WORD   ^B00
 8719
 8720  030204      004                .BYTE   SEEK            ;SEEK COMMAND
 8721  030205      002                .BYTE   2
 8722  030206  000000                 .WORD   ^B00
 8723
 8724  030210      006                .BYTE   RECAL           ;RECALIBRATE COMMAND
 8725  030211      002                .BYTE   2
 8726  030212  000000                 .WORD   ^B00
 8727
 8728  030214      010                .BYTE   DRVCLR          ;DRIVE CLEAR COMMAND
 8729  030215      002                .BYTE   2
 8730  030216  000001                 .WORD   ^B01
 8731
 8732  030220      012                .BYTE   RLEASE          ;RELEASE COMMAND
 8733  030221      003                .BYTE   3
 8734  030222  000000                 .WORD   ^B000
 8735
 8736  030224      014                .BYTE   OFFSET          ;OFFSET COMMAND
 8737  030225      002                .BYTE   2
 8738  030226  000000                 .WORD   ^B00
 8739
 8740  030230      016                .BYTE   RTC             ;RETURN TO CENTER COMMAND
 8741  030231      002                .BYTE   2
 8742  030232  000000                 .WORD   ^B00
 8743
 8744  030234      020                .BYTE   RIP             ;READ IN PRESET COMMAND
 8745  030235      004                .BYTE   4
 8746  030236  000016                 .WORD   ^B1110
 8747
 8748  030240      022                .BYTE   PAKACK          ;PACK ACKNOWLEDGE
```

```
8749  030241  004                             .BYTE   4
8750  030242  000016                          .WORD   ^B1110
8751
8752  030244  024                             .BYTE   ILF24           ;ILLEGAL FUNCTION 24
8753  030245  002                             .BYTE   2
8754  030246  000000                          .WORD   ^B00
8755
8756  030250  026                             .BYTE   ILF26           ;ILLEGAL FUNCTION 26
8757  030251  002                             .BYTE   2
8758  030252  000000                          .WORD   ^B00
8759
8760  030254  030                             .BYTE   SEARCH          ;SEARCH COMMAND
8761  030255  003                             .BYTE   3
8762  030256  000000                          .WORD   ^B000
8763
8764  030260  032                             .BYTE   ILF32           ;ILLEGAL FUNCTION 32
8765  030261  003                             .BYTE   3
8766  030262  000000                          .WORD   ^B000
8767
8768  030264  034                             .BYTE   ILF34           ;ILLEGAL FUNCTION 34
8769  030265  002                             .BYTE   2
8770  030266  000000                          .WORD   ^B00
8771
8772  030270  036                             .BYTE   ILF36           ;ILLEGAL FUNCTION 36
8773  030271  002                             .BYTE   2
8774  030272  000000                          .WORD   ^B00
8775
8776  030274  000                             .BYTE                   ;END OF TABLE
8777  030275  377                             .BYTE   -1
8778  030276  000000                          .WORD
8779
8780  030300                          200$:                           ;END OF TEST
8781
8782                                  ;;*****************************************************************
8783                                  ;*TEST 50        SET/RESET IVC TEST
8784
8785                                  ;;*****************************************************************
8786  030300  000004                  TST50:  SCOPE
8787  030302  012737  000050  001226          MOV     #50,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
8788
8789  030310  000240                          NOP
8790  030312  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
8791  030320  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
8792  030326  012737  030342  001122          MOV     #T50,$LPADR  ;LOAD LOOP ON TEST ADDRESS
8793  030334  012737  030342  001124          MOV     #T50,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
8794  030342                          T50:
8795  030342  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
8796  030346  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
8797  030352  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
8798  030356  010037  001136                  MOV     R0,$BDADR       ;SETUP REG ADDRESS
8799  030362  062737  000042  001136          ADD     #RMER2,$BDADR
8800  030370  005002                          CLR     R2              ;R2=FUNCTION CODE
8801  030372                          10$:
8802                                  ;INITIALIZE AND VERIFY THAT IVC STATUS IS ZERO.
8803  030372  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
8804  030400  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
```

```
8805
8806  030404  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
8807
8808  030412  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
8809
8810  030420  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
8811
8812  030426  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
8813
8814  030434  016037  000042  001142          MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
8815  030442  042737  167777  001142          BIC     #^CIVC,$BDDAT
8816  030450  001404                          BEQ     20$             ;BRANCH IF IVC IS ZERO
8817  030452  005037  001140                  CLR     $GDDAT
8818  030456  104157                          ERROR   157             ;CANT CLEAR IVC
8819  030460  000444                          BR      40$             ;SKIP REST OF TEST
8820  030462                          20$:
8821                                  ;LOAD THE FUNCTION CODE WITH GO BIT, STEP THE COMMAND SEQUENCER OFF
8822                                  ;ADDRESS 0 AND VERIFY IVC STATUS.
8823  030462  010203                          MOV     R2,R3           ;SETUP FUNCTION CODE
8824  030464  052703  000001                  BIS     #GO,R3
8825
8826  030470  010360  000000                  MOV     R3,RMCS1(R0)    ;LOAD RMCS1
8827
8828  030474  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
8829
8830  030502  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
8831
8832  030510  016037  000042  001142          MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
8833  030516  042737  167777  001142          BIC     #^CIVC,$BDDAT   ;SET ACTUAL STATUS
8834  030524  016237  066042  001140          MOV     FNCDTB(R2),$GDDAT  ;SETUP EXPECTED STATUS FROM
8835  030532  042737  167777  001140          BIC     #^CIVC,$GDDAT   ;FUNCTION CODE TABLE
8836  030540  023737  001140  001142          CMP     $GDDAT,$BDDAT
8837  030546  001403                          BEQ     30$             ;BRANCH IF IVC IS OK
8838  030550  010237  001174                  MOV     R2,$TMP0        ;SAVE FUNCTION CODE FOR MSG
8839  030554  104160                          ERROR   160             ;IVC IS INCORRECT
8840  030556                          30$:
8841                                  ;ADVANCE FUNCTION CODE AND REPEAT TEST IF NOT DONE
8842  030556  062702  000002                  ADD     #2,R2
8843  030562  022702  000076                  CMP     #iLF76,R2
8844  030566  103401                          BLO     40$             ;BRANCH IF DONE TEST
8845  030570  000700                          BR      10$
8846
8847  030572                          40$:                            ;END OF TEST
8848
8849                                  ;;*******************************************************************
8850                                  ;*TEST 51        SET LSC TEST
8851
8852                                  ;;*******************************************************************
8853  030572  000004                  TST51:  SCOPE
8854  030574  012737  000051  001226          MOV     #51,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
8855
8856  030602  000240                          NOP
8857  030604  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
8858  030612  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
8859  030620  012737  030634  001122          MOV     #T51,$LPADR  ;LOAD LOOP ON TEST ADDRESS
8860  030626  012737  030634  001124          MOV     #T51,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
```

```
8861  030634                                  T51:
8862  030634  012706  001100                          MOV     #STACK,SP       ;LOAD THE STACK POINTER
8863  030640  013700  001276                          MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
8864  030644  013701  001456                          MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
8865  030650  010037  001136                          MOV     R0,$BDADR
8866  030654  062737  000042  001136                  ADD     #RMER2,$BDADR
8867
8868                                  ;INITIALIZE AND VERIFY THAT LOSS OF SYSTEM CLOCK, 'LSC', IS RESET
8869  030662  012760  000040  000010                  MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
8870  030670  111160  000010                          MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
8871
8872  030674  012760  000001  000024                  MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
8873
8874  030702  012760  040001  000024                  MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
8875
8876  030710  012760  000000  000014                  MOV     #0,RMER1(R0)    ;LOAD RMER1
8877
8878  030716  012760  000000  000042                  MOV     #0,RMER2(R0)    ;LOAD RMER2
8879
8880  030724  016037  000042  001142                  MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
8881  030732  042737  173777  001142                  BIC     #^CLSC,$BDDAT
8882  030740  001403                                  BEQ     10$             ;BRANCH IF LSC IS ZERO
8883  030742  005037  001140                          CLR     $GDDAT
8884  030746  104161                                  ERROR   161             ;CANT CLEAR LSC
8885                                  ;WITH DEBUG CLOCK ENABLED, SET GO AND WAIT FOR ONE SHOT TO SET
8886  030750                          10$:
8887
8888  030750  012760  000001  000000                  MOV     #GO,RMCS1(R0)   ;LOAD RMCS1
8889  030756  012737  000001  001524                  MOV     #1,WATCH        ;SET WATCHDOG TIMER VALUE
8890  030764  004777  150536                          JSR     PC,@CLOCK       ;START THE CLOCK
8891  030770  005737  001524                  20$:    TST     WATCH
8892  030774  001375                                  BNE     20$             ;WAIT FOR WATCH ZERO
8893  030776  004777  150526                          JSR     PC,@STOP        ;STOP THE CLOCK
8894                                  ;ONE SHOT SHOULD BE SET-DISABLE DIAGNOSTIC CLOCK AND LSC SHOULD SET.
8895
8896  031002  012760  000001  000024                  MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
8897
8898  031010  016037  000042  001142                  MOV     RMER2(R0),$BDDAT        ;STORE RMER2 AT $BDDAT
8899  031016  042737  173777  001142                  BIC     #^CLSC,$BDDAT
8900  031024  001004                                  BNE     30$             ;BRANCH IF LSC SET
8901  031026  012737  004000  001140                  MOV     #LSC,$GDDAT
8902  031034  104162                                  ERROR   162             ;CANT SET LSC
8903
8904  031036                          30$:                            ;END OF TEST
8905
8906                                  ;;*****************************************************************
8907                                  ;*TEST 52        DECODE TEST
8908
8909                                  ;;*****************************************************************
8910  031036  000004                  TST52:  SCOPE
8911  031040  012737  000052  001226                  MOV     #52,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
8912
8913  031046  000240                          NOP
8914  031050  012737  000024  001120                  MOV     #20.,$ICNT      ;20 ITERATIONS
8915  031056  112737  000001  001131                  MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
8916  031064  012737  031100  001122                  MOV     #T52,$LPADR  ;LOAD LOOP ON TEST ADDRESS
```

```
8917  031072  012737  031100  001124          MOV     #T52,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
8918  031100                          T52:
8919  031100  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
8920  031104  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
8921  031110  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
8922  031114  005037  001416                  CLR     RMER10          ;NO ERROR FIRST TEST
8923  031120                          5$:
8924  031120  004737  031500                  JSR     PC,100$         ;INITIALIZE
8925
8926                                  ;EXECUTE A PACK ACKNOWLEDGE AND CHECK VOLUME VALID
8927
8928  031124  013760  001416  000014          MOV     RMER10,RMER1(R0)        ;LOAD RMER1
8929
8930  031132  012760  000023  000000          MOV     #PACACK!GO,RMCS1(R0)    ;LOAD RMCS1
8931  031140  012703  000003                  MOV     #3,R3
8932  031144                          10$:
8933
8934  031144  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
8935
8936  031152  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
8937  031160  005303                          DEC     R3
8938  031162  001370                          BNE     10$             ;ISSUE NEXT CLOCK IF COUNT NOT 0
8939
8940  031164  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
8941  031172  042737  177677  001142          BIC     #^CVV,$BDDAT
8942  031200  001414                          BEQ     20$             ;BRANCH IF VV IS ZERO
8943  031202  005737  001416                  TST     RMER10
8944  031206  001527                          BEQ     70$             ;BRANCH IF VV SHOULD BE SET
8945  031210  005037  001140                  CLR     $GDDAT          ;SETUP ERROR MESSAGE
8946  031214  010037  001136                  MOV     R0,$BDADR
8947  031220  062737  000012  001136          ADD     #RMDS,$BDADR
8948  031226  104163                          ERROR   163             ;DECODE SET WITH COMP ERROR ACTIVE
8949  031230  000522                          BR      80$             ;SKIP
8950  031232                          20$:
8951  031232  004737  031500                  JSR     PC,100$         ;INITIALIZE AND SET DIAGNOSTIC MODE
8952
8953                                  ;EXECUTE A READ IN PRESET AND CHECK VOLUME VALID
8954
8955  031236  013760  001416  000014          MOV     RMER10,RMER1(R0)        ;LOAD RMER1
8956
8957  031244  012760  000021  000000          MOV     #RIP!GO,RMCS1(R0)       ;LOAD RMCS1
8958  031252  012703  000003                  MOV     #3,R3           ;R3=CLOCK COUNT
8959  031256                          30$:
8960
8961  031256  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
8962
8963  031264  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
8964  031272  005303                          DEC     R3
8965  031274  001370                          BNE     30$             ;ISSUE NEXT CLOCK IF COUNT NOT ZERO
8966
8967  031276  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
8968  031304  042737  177677  001142          BIC     #^CVV,$BDDAT
8969  031312  001414                          BEQ     40$             ;BRANCH IF VOLUME VALID NOT SET
8970  031314  005737  001416                  TST     RMER10
8971  031320  001462                          BEQ     70$             ;BRANCH IF VOLUME VALID SHOULD BE SET
8972  031322  005037  001140                  CLR     $GDDAT          ;SETUP ERROR MESSAGE
```

```
 8973  031326  010037  001136                MOV     R0,$BDADR
 8974  031332  062737  000012  001136        ADD     #RMDS,$BDADR
 8975  031340  104163                        ERROR   163             ;DECODE SET WITH COMP ERROR ACTIVE
 8976  031342  000455                        BR      80$             ;SKIP
 8977  031344                        40$:
 8978  031344  004737  031500                JSR     PC,100$         ;INITIALIZE AND SET DIAGNOSTIC MODE
 8979
 8980                                ;EXECUTE A WRITE CHECK DATA AND CHECK OCCUPIED
 8981
 8982  031350  013760  001416  000014        MOV     RMER10,RMER1(R0)        ;LOAD RMER1
 8983
 8984  031356  012760  000051  000000        MOV     #WCD!GO,RMCS1(R0)       ;LOAD RMCS1
 8985  031364  012703  000002                MOV     #2,R3           ;R3=CLOCK COUNT
 8986  031370                        50$:
 8987
 8988  031370  012760  141001  000024        MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
 8989
 8990  031376  012760  041001  000024        MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 8991  031404  005303                        DEC     R3
 8992  031406  001370                        BNE     50$             ;ISSUE NEXT CLOCK IF COUNT NOT ZERO
 8993
 8994  031410  016037  000024  001142        MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
 8995  031416  042737  077777  001142        BIC     #^COCC,$BDDAT
 8996  031424  001414                        BEQ     60$             ;BRANCH IF OCCUPIED IS RESET
 8997  031426  005737  001416                TST     RMER10
 8998  031432  001415                        BEQ     70$             ;BRANCH IF OCCUPIED SHOULD BE SET
 8999  031434  005037  001140                CLR     $GDDAT          ;SETUP ERROR MESSAGE
 9000  031440  010037  001136                MOV     R0,$BDADR
 9001  031444  062737  000024  001136        ADD     #RMMR1,$BDADR
 9002  031452  104164                        ERROR   164             ;DECODE SET WITH COMP ERROR ACTIVE
 9003  031454  000410                        BR      80$
 9004  031456                        60$:
 9005                                ;VOLUME VALID AND OCCUPIED DID NOT SET-SEE IF COMP ERROR WAS ACTIVE
 9006  031456  005737  001416                TST     RMER10
 9007  031462  001005                        BNE     80$             ;BRANCH IF COMP ERROR WAS SET
 9008                                ;COULD NOT SET VV OR OCCUPIED-SUSPECT DECODE FLOP NOT SETTING
 9009  031464  104165                        ERROR   165             ;DECODE DOES NOT SET
 9010
 9011  031466                        70$:
 9012
 9013                                ;REPEAT TEST WITH COMPOSITE ERROR ACTIVE-VERIFY THAT DECODE FLOP
 9014                                ;DOES NOT SET, AS INDICATED BY VOLUME VALID AND OCCUPIED.
 9015  031466  012737  040000  001416        MOV     #UNS,RMER10     ;USE UNSAFE TO SET COMP ERROR
 9016  031474  000611                        BR      5$
 9017
 9018  031476  000513                80$:    BR      200$            ;END OF TEST
 9019
 9020  031500                        100$:
 9021
 9022                                ;SUBROUTINE USED DURING TEST
 9023
 9024                                ;USING DIAGNOSTIC MODE, RESET VOLUME VALID AND COMPOSITE ERROR.
 9025                                ;VERIFY THAT VV, ERR, AND OCC ARE ZERO.
 9026  031500  012760  000040  000010        MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
 9027  031506  111160  000010                MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
 9028
```

```
9029  031512  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
9030
9031  031520  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9032
9033  031526  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
9034
9035  031534  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
9036  031542  005037  001140                  CLR     $GDDAT          ;SETUP FOR ERROR MSG
9037  031546  010037  001136                  MOV     R0,$BDADR
9038  031552  062737  000012  001136          ADD     #RMDS,$BDADR
9039
9040  031560  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
9041  031566  042737  137777  001142          BIC     #^CERR,$BDDAT
9042  031574  001402                          BEQ     110$            ;BRANCH IF COMP ERROR ZERO
9043  031576  104143                          ERROR   143             ;CANT CLEAR COMP ERROR
9044  031600  000447                          BR      140$            ;SKIP TEST
9045  031602                          110$:
9046
9047  031602  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
9048  031610  042737  177677  001142          BIC     #^CVV,$BDDAT
9049  031616  001402                          BEQ     120$            ;BRANCH IF VOLUME VALID ZERO
9050  031620  104135                          ERROR   135             ;CANT RESET VOLUME VALID
9051  031622  000436                          BR      140$            ;SKIP TEST
9052  031624                          120$:
9053
9054  031624  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
9055  031632  042737  077777  001142          BIC     #^COCC,$BDDAT
9056  031640  001407                          BEQ     130$            ;BRANCH IF OCCUPIED ZERO
9057  031642  010037  001136                  MOV     R0,$BDADR       ;SETUP ERROR MESSAGE
9058  031646  062737  000024  001136          ADD     #RMMR1,$BDADR
9059  031654  104166                          ERROR   166             ;CANT CLEAR OCCUPIED
9060  031656  000420                          BR      140$            ;SKIP TEST
9061  031660                          130$:
9062                                  ;TO VERIFY THAT THE DECODE FLOP IS RESET, LOAD AN ILLEGAL FUNCTION
9063                                  ;IN RMCS1 AND VERIFY THAT ILF DOES NOT SET.
9064
9065  031660  012760  000024  000000          MOV     #ILF24,RMCS1(R0)        ;LOAD RMCS1
9066
9067  031666  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
9068  031674  042737  177776  001142          BIC     #^CILF,$BDDAT
9069  031702  001410                          BEQ     150$            ;BRANCH IF ILF IS ZERO
9070  031704  010037  001136                  MOV     R0,$BDADR       ;SETUP ERROR MESSAGE
9071  031710  062737  000014  001136          ADD     #RMER1,$BDADR
9072  031716  104167                          ERROR   167             ;DECODE FLOP APPEARS ON
9073  031720  012716  031726          140$:   MOV     #200$,(SP)      ;DONT GO BACK TO TEST
9074
9075  031724  000207                  150$:   RTS     PC              ;RETURN TO TEST OR EXIT TEST
9076
9077  031726                          200$:
9078
9079                                  ;;**************************************************************
9080                                  ;*TEST 53        SET/RESET VOLUME VALID TEST
9081
9082                                  ;;**************************************************************
9083  031726  000004                  TST53:  SCOPE
9084  031730  012737  000053  001226          MOV     #53,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
```

```
9085
9086  031736  000240                          NOP
9087  031740  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
9088  031746  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
9089  031754  012737  031770  001122          MOV     #T53,$LPADR  ;LOAD LOOP ON TEST ADDRESS
9090  031762  012737  031770  001124          MOV     #T53,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
9091  031770                          T53:
9092  031770  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
9093  031774  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
9094  032000  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
9095  032004  010037  001136                  MOV     R0,$BDADR       ;SETUP REGISTER ADDRESS
9096  032010  062737  000012  001136          ADD     #RMDS,$BDADR
9097  032016  012702  032230                  MOV     #100$,R2        ;R2=TABLE POINTER
9098  032022                          10$:
9099                                  ;INITIALIZE AND USE DIAGNOSTIC MODE TO RESET VOLUME VALID
9100  032022  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
9101  032030  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
9102
9103  032034  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
9104
9105  032042  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9106
9107  032050  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
9108
9109  032056  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
9110
9111  032064  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
9112  032072  042737  177677  001142          BIC     #^CVV,$BDDAT
9113  032100  001403                          BEQ     20$             ;BRANCH IF VOLUME VALID ZERO
9114  032102  005037  001140                  CLR     $GDDAT
9115  032106  104135                          ERROR   135             ;CANT RESET VV
9116  032110                          20$:
9117                                  ;EXECUTE THE FUNCTION CODE IN THE TABLE
9118  032110  111203                          MOVB    (R2),R3         ;GET FUNCTION CODE
9119  032112  042703  177701                  BIC     #^CILF76,R3
9120  032116  052703  000001                  BIS     #GO,R3
9121
9122  032122  010360  000000                  MOV     R3,RMCS1(R0)    ;LOAD RMCS1
9123  032126  116204  000001                  MOVB    1(R2),R4        ;GET CLOCK COUNT
9124  032132  042704  177400                  BIC     #^C377,R4
9125  032136                          30$:
9126
9127  032136  012760  141001  000024          MOV     #DMD!DBEN!MUR!DBCK,RMMR1(R0)    ;LOAD RMMR1
9128
9129  032144  012760  041001  000024          MOV     #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
9130  032152  005304                          DEC     R4
9131  032154  001370                          BNE     30$             ;ISSUE COCKS TIL R4 ZERO
9132
9133  032156  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
9134  032164  042737  177677  001142          BIC     #^CVV,$BDDAT
9135  032172  001007                          BNE     40$             ;BRANCH IF VOLUME VALID SET
9136  032174  010337  001174                  MOV     R3,$TMP0        ;SAVE FUNCTION CODE FOR MSG
9137  032200  012737  000100  001140          MOV     #VV,$GDDAT
9138  032206  104170                          ERROR   170             ;CANT SET VOLUME VALID
9139  032210  000406                          BR      50$
9140  032212                          40$:
```

```
9141
9142  032212  062702  000002              ;ADVANCE THE TABLE POINTER, EXIT IF DONE
                              ADD     #2,R2
9143  032216  105762  000001              TSTB    1(R2)
9144  032222  100401                      BMI     50$             ;EXIT IF COUNT IS NEGATIVE
9145  032224  000676                      BR      10$
9146  032226  000403              50$:    BR      200$            ;JUMP OVER TABLE
9147
9148  032230                      100$:
9149                              ;TABLE OF FUNCTION CODES AND CLOCK COUNTS
9150  032230      020                     .BYTE   RIP             ;READ IN PRESET COMMAND
9151  032231      003                     .BYTE   3
9152
9153  032232      022                     .BYTE   PAKACK          ;PACK ACKNOWLEDGE COMMAND
9154  032233      003                     .BYTE   3
9155
9156  032234      000                     .BYTE                   ;END OF TABLE
9157  032235      377                     .BYTE   -1
9158
9159  032236                      200$:                           ;END OF TEST
9160
9161                              ;;*******************************************************************
9162                              ;*TEST 54        ILLEGAL FUNCTION TEST
9163
9164                              ;;*******************************************************************
9165  032236  000004              TST54:  SCOPE
9166  032240  012737  000054  001226      MOV     #54,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
9167
9168  032246  000240                      NOP
9169  032250  012737  000024  001120      MOV     #20.,$ICNT      ;20 ITERATIONS
9170  032256  112737  000001  001131      MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
9171  032264  012737  032300  001122      MOV     #T54,$LPADR  ;LOAD LOOP ON TEST ADDRESS
9172  032272  012737  032300  001124      MOV     #T54,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
9173  032300                      T54:
9174  032300  012706  001100              MOV     #STACK,SP       ;LOAD THE STACK POINTER
9175  032304  013700  001276              MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
9176  032310  013701  001456              MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
9177  032314  005002                      CLR     R2              ;INITIALIZE FUNCTION CODE VALUE
9178  032316                      10$:
9179                              ;SET VOLUME VALID USING SUBROUTINE
9180  032316  004737  060556              JSR     PC,SETVV        ;GO SET VOLUME VALID
9181  032322  000402                      BR      20$             ;BRANCH TO 20$ IF NO ERROR
9182  032324  104000                      ERROR                   ;RETURN HERE IF ERROR
9183  032326  000460                      BR      50$             ;SKIP TEST IF ERROR
9184  032330  012704  000002      20$:    MOV     #2,R4           ;R4=CLOCK COUNT
9185                              ;EXECUTE THE TEST FUNCTION CODE AND VERIFY ILF
9186
9187  032334  012760  041001  000024      MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9188  032342  010203                      MOV     R2,R3           ;SETUP FUNCTION CODE IN R3
9189  032344  052703  000001              BIS     #GO,R3
9190
9191  032350  010360  000000              MOV     R3,RMCS1(R0)    ;LOAD RMCS1
9192  032354                      30$:
9193
9194  032354  012760  141001  000024      MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
9195
9196  032362  012760  041001  000024      MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
```

```
 9197  032370  005304                          DEC     R4
 9198  032372  001370                          BNE     30$
 9199
 9200  032374  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
 9201  032402  042737  177776  001142          BIC     #^CILF,$BDDAT   ;SETUP ACTUAL ILF STATUS
 9202  032410  016237  066042  001140          MOV     FNCDTB(R2),$GDDAT  ;GET EXPECTED ILF STATUS
 9203  032416  042737  177776  001140          BIC     #^CILF,$GDDAT
 9204  032424  023737  001140  001142          CMP     $GDDAT,$BDDAT
 9205  032432  001410                          BEQ     40$             ;BRANCH IF ILF IS OK
 9206  032434  010037  001136                  MOV     R0,$BDADR       ;SETUP FOR ERROR MSG
 9207  032440  062737  000014  001136          ADD     #RMER1,$BDADR
 9208  032446  010237  001174                  MOV     R2,$TMP0
 9209  032452  104171                          ERROR   171             ;ILF IS NOT CORRECT
 9210  032454                          40$:
 9211                                  ;ADVANCE TO THE NEXT FUNCTION CODE AND REPEAT TEST
 9212  032454  062702  000002                  ADD     #2,R2
 9213  032460  022702  000076                  CMP     #ILF76,R2
 9214  032464  103401                          BLO     50$
 9215  032466  000713                          BR      10$
 9216  032470                          50$:                            ;END OF TEST
 9217
 9218                                  ;;*****************************************************************
 9219                                  ;*TEST 55        OCCUPIED TEST
 9220
 9221                                  ;;*****************************************************************
 9222  032470  000004                  TST55:  SCOPE
 9223  032472  012737  000055  001226          MOV     #55,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 9224
 9225  032500  000240                          NOP
 9226  032502  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
 9227  032510  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
 9228  032516  012737  032532  001122          MOV     #T55,$LPADR  ;LOAD LOOP ON TEST ADDRESS
 9229  032524  012737  032532  001124          MOV     #T55,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
 9230  032532                          T55:
 9231  032532  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
 9232  032536  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
 9233  032542  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
 9234  032546  005002                          CLR     R2                      ;INITIALIZE FUNCTION CODE
 9235
 9236  032550                          10$:
 9237                                  ;GET THE DEVICE READY
 9238                                  ;SET VOLUME VALID USING SUBROUTINE
 9239  032550  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
 9240  032554  000402                          BR      20$             ;BRANCH TO 20$ IF NO ERROR
 9241  032556  104000                          ERROR                   ;RETURN HERE IF ERROR
 9242  032560  000464                          BR      50$
 9243  032562                          20$:
 9244                                  ;ENABLE DEBUG CLOCK AND LOAD THE FUNCTION CODE
 9245
 9246  032562  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 9247  032570  010203                          MOV     R2,R3                   ;ASSEMBLE FUNCTION CODE AND
 9248  032572  052703  000001                  BIS     #GO,R3                  ;GO BIT IN R3
 9249
 9250  032576  010360  000000                  MOV     R3,RMCS1(R0)    ;LOAD RMCS1
 9251  032602  012704  000002                  MOV     #2,R4                   ;R4=CLOCK COUNT
 9252                                  ;STEP THE DEBUG CLOCK UNTIL SET PULSE IS ACTIVE
```

```
 9253  032606                          30$:
 9254
 9255  032606  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
 9256
 9257  032614  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 9258  032622  005304                          DEC     R4
 9259  032624  001370                          BNE     30$                     ;ISSUE NEXT CLOCK TIL R4 ZERO
 9260                                  ;VERIFY OCCUPIED STATUS
 9261
 9262  032626  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
 9263  032634  042737  077777  001142          BIC     #^COCC,$BDDAT
 9264  032642  005037  001140                  CLR     $GDDAT          ;GENERATE OCC FROM AOE
 9265  032646  032762  001000  066042          BIT     #AOE,FNCDTB(R2)
 9266  032654  001403                          BEQ     35$
```

```
 9267  032656  012737  100000  001140          MOV     #OCC,$GDDAT
 9268  032664  023737  001140  001142  35$:    CMP     $GDDAT,$BDDAT
 9269  032672  001411                          BEQ     40$                     ;BRANCH IF OCC IS OK
 9270  032674  010237  001174                  MOV     R2,$TMP0                ;SAVE FUNCTION CODE
 9271  032700  010037  001136                  MOV     R0,$BDADR               ;SETUP REGISTER ADDRESS
 9272  032704  062737  000024  001136          ADD     #RMMR1,$BDADR
 9273  032712  104173                          ERROR   173                     ;OCCUPIED IS INCORRECT
 9274  032714  000406                          BR      50$
 9275  032716                          40$:
 9276                                  ;ADVANCE TO NEXT FUNCTIONCODE, EXIT IF DONE
 9277  032716  062702  000002                  ADD     #2,R2
 9278  032722  022702  000076                  CMP     #ILF76,R2
 9279  032726  103401                          BLO     50$                     ;EXIT IF DONE
 9280  032730  000707                          BR      10$
 9281
 9282  032732                          50$:                                    ;END OF TEST
 9283
 9284                                  ;;*******************************************************************
 9285                                  ;*TEST 56        READ IN PRESET TEST
 9286
 9287                                  ;;*******************************************************************
 9288  032732  000004                  TST56:  SCOPE
 9289  032734  012737  000056  001226          MOV     #56,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 9290
 9291  032742  000240                          NOP
 9292  032744  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
 9293  032752  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
 9294  032760  012737  032774  001122          MOV     #T56,$LPADR  ;LOAD LOOP ON TEST ADDRESS
 9295  032766  012737  032774  001124          MOV     #T56,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
 9296  032774                          T56:
 9297  032774  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
 9298  033000  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
 9299  033004  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
 9300                                  ;CLEAR AND ENABLE DEBUG CLOCK - LEAVE VOLUME VALID RESET
 9301  033010  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
 9302  033016  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
 9303
 9304  033022  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
 9305
 9306  033030  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 9307
 9308  033036  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
 9309
 9310  033044  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
 9311                                  ;LOAD ALL ONES IN RMDA, RMDC AND RMOF
 9312
 9313  033052  012760  177777  000006          MOV     #-1,RMDA(R0)    ;LOAD RMDA
 9314
 9315  033060  012760  177777  000034          MOV     #-1,RMDC(R0)    ;LOAD RMDC
 9316
 9317  033066  012760  177777  000032          MOV     #-1,RMOF(R0)    ;LOAD RMOF
 9318                                  ;LOAD READ IN PRESET COMMAND AND STEP THE CLOCK TILL SET PULSE
 9319
 9320  033074  012760  000021  000000          MOV     #RIP!GO,RMCS1(R0)       ;LOAD RMCS1
 9321  033102  012702  000003                  MOV     #3,R2                   ;R2=CLOCK COUNT
 9322  033106                          10$:
```

```
9323
9324  033106  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
9325
9326  033114  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9327  033122  005302                          DEC     R2
9328  033124  001370                          BNE     10$                     ;ISSUE 3 CLOCKS
9329                                  ;SEE IF RMDA OR RMDC OR RMOF IS ZERO
9330
9331  033126  016002  000006                  MOV     RMDA(R0),R2     ;STORE RMDA AT R2
9332  033132  005702                          TST     R2
9333  033134  001413                          BEQ     20$                     ;BRANCH IF RMDA IS ZERO
9334
9335  033136  016002  000034                  MOV     RMDC(R0),R2     ;STORE RMDC AT R2
9336  033142  042702  176000                  BIC     #XNUDC,R2               ;CLEAR UNUSED BITS
9337  033146  001406                          BEQ     20$                     ;BRANCH IF RMDC IS ZERO
9338
9339  033150  016002  000032                  MOV     RMOF(R0),R2     ;STORE RMOF AT R2
9340  033154  042702  161577                  BIC     #XNUOF,R2               ;CLEAR UNUSED BITS
9341  033160  001401                          BEQ     20$                     ;BRANCH IF RMOF IS ZERO
9342                                  ;READ IN PRESET COMMAND DIDNT CLEAR ANY OF THE 3 REGISTERS
9343  033162  104174                          ERROR   174                     ;READ IN PRESET FAILED
9344
9345  033164                          20$:                                    ;END OF TEST
9346
9347                                  ;;*******************************************************************
9348                                  ;*TEST 57        RIP/RMOF TEST
9349
9350                                  ;;*******************************************************************
9351  033164  000004                  T57:    SCOPE
9352  033166  012737  000057  001226          MOV     #57,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
9353
9354  033174  000240                          NOP
9355  033176  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
9356  033204  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
9357  033212  012737  033226  001122          MOV     #T57,$LPADR  ;LOAD LOOP ON TEST ADDRESS
9358  033220  012737  033226  001124          MOV     #T57,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
9359  033226                          T57:
9360  033226  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
9361  033232  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
9362  033236  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
9363  033242  010037  001136                  MOV     R0,$BDADR               ;SETUP REGISTER ADDRESS AND
9364  033246  062737  000032  001136          ADD     #RMOF,$BDADR
9365  033254  005037  001140                  CLR     $GDDAT                  ;EXPECTED RMOF
9366
9367                                  ;INITIALIZE AND SET BITS IN RMOF
9368  033260  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
9369  033266  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
9370
9371  033272  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
9372
9373  033300  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9374
9375  033306  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
9376
9377  033314  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
9378
```

```
 9379  033322  012760  177777  000032          MOV     #-1,RMOF(R0)    ;LOAD RMOF
 9380                          ;EXECUTE A READ IN PRESET IN DIAGNOSTIC MODE TILL SET PULSE
 9381
 9382  033330  012760  000021  000000          MOV     #RIP!GO,RMCS1(R0)       ;LOAD RMCS1
 9383  033336  012702  000003                  MOV     #3,R2                   ;R2=CLOCK COUNT
 9384  033342                  10$:
 9385
 9386  033342  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
 9387
 9388  033350  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 9389  033356  005302                  DEC     R2
 9390  033360  001370                  BNE     10$                     ;ISSUE 3 CLOCKS
 9391                          ;VERIFY THAT RMOF IS ZERO
 9392
 9393  033362  016037  000032  001142          MOV     RMOF(R0),$BDDAT ;STORE RMOF AT $BDDAT
 9394  033370  042737  161577  001142          BIC     #XNUOF,$BDDAT
 9395  033376  001401                  BEQ     20$                     ;BRANCH IF RMOF IS ZERO
 9396  033400  104175                  ERROR   175                     ;CANT CLEAR RMOF WITH RIP
 9397
 9398  033402                  20$:                                    ;END OF TEST
 9399
 9400                          ;;**************************************************************
 9401                          ;*TEST 60        RMDA/RMDC/RIP TEST
 9402
 9403                          ;;**************************************************************
 9404  033402  000004          TST60:  SCOPE
 9405  033404  012737  000060  001226          MOV     #60,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 9406
 9407  033412  000240                  NOP
 9408  033414  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
 9409  033422  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
 9410  033430  012737  033444  001122          MOV     #T60,$LPADR  ;LOAD LOOP ON TEST ADDRESS
 9411  033436  012737  033444  001124          MOV     #T60,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
 9412  033444                  T60:
 9413  033444  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
 9414  033450  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
 9415  033454  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
 9416  033460  005037  001140                  CLR     $GDDAT
 9417
 9418                          ;CLEAR, ENABLE DEBUG CLOCK, THEN PRESET RMDA AND RMDC
 9419  033464  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
 9420  033472  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
 9421
 9422  033476  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
 9423
 9424  033504  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 9425
 9426  033512  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
 9427
 9428  033520  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
 9429
 9430  033526  012760  177777  000006          MOV     #-1,RMDA(R0)    ;LOAD RMDA
 9431
 9432  033534  012760  177777  000034          MOV     #-1,RMDC(R0)    ;LOAD RMDC
 9433                          ;EXECUTE READ IN PRESET TILL SET PULSE
 9434
```

```
9435  033542  012760  000021  000000          MOV     #RIP!GO,RMCS1(R0)       ;LOAD RMCS1
9436  033550  012702  000003                  MOV     #3,R2
9437  033554                          10$:
9438
9439  033554  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
9440
9441  033562  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9442  033570  005302                          DEC     R2
9443  033572  001370                          BNE     10$                     ;ISSUE 3 CLOCKS
9444                          ;VERIFY RMDA IS ZERO
9445
9446  033574  016037  000006  001142          MOV     RMDA(R0),$BDDAT ;STORE RMDA AT $BDDAT
9447  033602  005737  001142                  TST     $BDDAT
9448  033606  001406                          BEQ     20$                     ;BRANCH IF RMDA RESET
9449  033610  010037  001136                  MOV     R0,$BDADR
9450  033614  062737  000006  001136          ADD     #RMDA,$BDADR
9451  033622  104176                          ERROR   176                     ;RMDA NOT RESET BY RIP
9452  033624                          20$:
9453                          ;VERIFY RMDC IS ZERO
9454
9455  033624  016037  000034  001142          MOV     RMDC(R0),$BDDAT ;STORE RMDC AT $BDDAT
9456  033632  042737  176000  001142          BIC     #XNUDC,$BDDAT
9457  033640  001406                          BEQ     30$                     ;BRANCH IF RMDC RESET
9458  033642  010037  001136                  MOV     R0,$BDADR
9459  033646  062737  000034  001136          ADD     #RMDC,$BDADR
9460  033654  104177                          ERROR   177                     ;RMDC NOT RESET BY RIP
9461
9462  033656                          30$:                                    ;END OF TEST
9463
9464                          ;;*****************************************************************
9465                          ;*TEST 61        OFFSET COMMAND TEST
9466
9467                          ;;*****************************************************************
9468  033656  000004          TST61:  SCOPE
9469  033660  012737  000061  001226          MOV     #61,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
9470
9471  033666  000240                          NOP
9472  033670  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
9473  033676  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
9474  033704  012737  033720  001122          MOV     #T61,$LPADR  ;LOAD LOOP ON TEST ADDRESS
9475  033712  012737  033720  001124          MOV     #T61,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
9476  033720                          T61:
9477  033720  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
9478  033724  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
9479  033730  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
9480  033734  010037  001136                  MOV     R0,$BDADR
9481  033740  062737  000012  001136          ADD     #RMDS,$BDADR
9482  033746  012737  000001  001140          MOV     #OM,$GDDAT
9483
9484                          ;SET VOLUME VALID USING SUBROUTINE
9485  033754  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
9486  033760  000402                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
9487  033762  104000                          ERROR                   ;RETURN HERE IF ERROR
9488  033764  000433                          BR      40$
9489  033766                          10$:
9490
```

```
9491  033766  012760  000000  000034          MOV     #0,RMDC(R0)      ;LOAD RMDC
9492                                  ;ENABLE DEBUG CLOCK AND EXECUTE OFFSET COMMAND
9493
9494  033774  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9495
9496  034002  012760  000015  000000          MOV     #OFFSET!GO,RMCS1(R0)    ;LOAD RMCS1
9497  034010  012702  000002                  MOV     #2,R2                   ;R2=CLOCK COUNT
9498  034014                          20$:
9499
9500  034014  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
9501
9502  034022  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9503  034030  005302                          DEC     R2
9504  034032  001370                          BNE     20$                     ;ISSUE 2 CLOCKS
9505                                  ;VERIFY THAT OFFSET MODE IS SET
9506
9507  034034  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
9508  034042  042737  177776  001142          BIC     #^COM,$BDDAT
9509  034050  001001                          BNE     40$                     ;BRANCH IF OM IS SET
9510  034052  104200                          ERROR   200                     ;CANT SET ON BY COMMAND
9511  034054                          40$:                                    ;END OF TEST
9512
9513                                  ;;**********************************************************************
9514                                  ;*TEST 62        RETURN TO CENTER TEST
9515
9516                                  ;;**********************************************************************
9517  034054  000004                  TST62:  SCOPE
9518  034056  012737  000062  001226          MOV     #62,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
9519
9520  034064  000240                          NOP
9521  034066  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
9522  034074  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
9523  034102  012737  034116  001122          MOV     #T62,$LPADR  ;LOAD LOOP ON TEST ADDRESS
9524  034110  012737  034116  001124          MOV     #T62,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
9525  034116                          T62:
9526  034116  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
9527  034122  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
9528  034126  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
9529  034132  010037  001136                  MOV     R0,$BDADR
9530  034136  062737  000012  001136          ADD     #RMDS,$BDADR
9531                                  ;SET OFFSET DIRECTION AND OFFSET MODE
9532                                  ;SET VOLUME VALID USING SUBROUTINE
9533  034144  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
9534  034150  000402                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
9535  034152  104000                          ERROR                   ;RETURN HERE IF ERROR
9536  034154  000465                          BR      60$
9537  034156                          10$:
9538
9539  034156  012760  000200  000032          MOV     #OFD,RMOF(R0)   ;LOAD RMOF
9540                                  ;SET OFFSET MODE USING SUBROUTINE
9541  034164  004737  060706                  JSR     PC,SETOM        ;GO SET OFFSET MODE
9542  034170  000401                          BR      20$             ;BRANCH TO 20$ IF NO ERROR
9543  034172  104000                          ERROR                   ;RETURN HERE IF ERROR
9544  034174                          20$:
9545                                  ;ENABLE DEBUG CLOCK AND EXECUTE RETURN TOCENTER COMMAND
9546                                  ;SET VOLUME VALID USING SUBROUTINE
```

```
9547  034174  004737  060556          JSR     PC,SETVV        ;GO SET VOLUME VALID
9548  034200  000402                  BR      30$             ;BRANCH TO 30$ IF NO ERROR
9549  034202  104000                  ERROR                   ;RETURN HERE IF ERROR
9550  034204  000451                  BR      60$
9551  034206                  30$:
9552
9553  034206  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9554
9555  034214  012760  000017  000000          MOV     #RTC!GO,RMCS1(R0)       ;LOAD RMCS1
9556  034222  012702  000002                  MOV     #2,R2
9557  034226                  40$:
9558
9559  034226  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
9560
9561  034234  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9562  034242  005302                  DEC     R2
9563  034244  001370                  BNE     40$                     ;ISSUE 2 CLOCKS
9564                          ;VERIFY THAT OFFSET MODE IS RESET
9565
9566  034246  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
9567  034254  042737  177776  001142          BIC     #^COM,$BDDAT
9568  034262  001403                  BEQ     50$                     ;BRANCH IF OFFSET MODE RESET
9569  034264  005037  001140                  CLR     $GDDAT
9570  034270  104201                  ERROR   201                     ;CANT RESET OFFSETMODE BY RTC
9571  034272                  50$:
9572                          ;VERIFY THAT OFFSET DIRECTION IS RESET
9573
9574  034272  016037  000032  001142          MOV     RMOF(R0),$BDDAT ;STORE RMOF AT $BDDAT
9575  034300  042737  177577  001142          BIC     #^COFD,$BDDAT
9576  034306  001410                  BEQ     60$                     ;BRANCH IF OFD IS RESET
9577  034310  005037  001140                  CLR     $GDDAT
9578  034314  010037  001136                  MOV     R0,$BDADR
9579  034320  062737  000032  001136          ADD     #RMOF,$BDADR
9580  034326  104202                  ERROR   202                     ;CANT RESET OFD BY RTC
9581  034330                  60$:                                    ;END OF TEST
9582
9583                          ;;*****************************************************************
9584                          ;*TEST 63        RMDC CLEAR OFFSET TEST
9585
9586                          ;;*****************************************************************
9587  034330  000004          TST63:  SCOPE
9588  034332  012737  000063  001226          MOV     #63,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
9589
9590  034340  000240                  NOP
9591  034342  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
9592  034350  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
9593  034356  012737  034372  001122          MOV     #T63,$LPADR  ;LOAD LOOP ON TEST ADDRESS
9594  034364  012737  034372  001124          MOV     #T63,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
9595  034372
9596  034372  012706  001100          T63:    MOV     #STACK,SP       ;LOAD THE STACK POINTER
9597  034376  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
9598  034402  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
9599  034406  010037  001136                  MOV     R0,$BDADR
9600  034412  062737  000012  001136          ADD     #RMDS,$BDADR
9601                          ;SET VOLUME VALID USING SUBROUTINE
9602  034420  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
```

```
9603  034424  000402                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
9604  034426  104000                          ERROR                   ;RETURN HERE IF ERROR
9605  034430  000421                          BR      40$                     ;SKIP REST OF TEST
9606  034432                          10$:
9607                                  ;SET OFFSET MODE USING SUBROUTINE
9608  034432  004737  060706                  JSR     PC,SETOM        ;GO SET OFFSET MODE
9609  034436  000401                          BR      20$             ;BRANCH TO 20$ IF NO ERROR
9610  034440  104000                          ERROR                   ;RETURN HERE IF ERROR
9611  034442                          20$:
9612                                  ;WRITE THE DESIRED CYLINDER REGISTER AND VERIFY THAT OFFSET IS ZERO
9613
9614  034442  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
9615
9616  034450  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
9617  034456  042737  177776  001142          BIC     #^COM,$BDDAT
9618  034464  001403                          BEQ     40$
9619  034466  005037  001140                  CLR     $GDDAT
9620  034472  104203                          ERROR   203                     ;CANT RESET OFFSET BY RMDC
9621
9622  034474                          40$:                            ;END OF TEST
9623
9624                                  ;;*******************************************************************
9625                                  ;*TEST 64        EBL CLEAR OFFSET TEST
9626
9627                                  ;;*******************************************************************
9628  034474  000004                  TST64:  SCOPE
9629  034476  012737  000064  001226          MOV     #64,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
9630
9631  034504  000240                          NOP
9632  034506  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
9633  034514  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
9634  034522  012737  034536  001122          MOV     #T64,$LPADR  ;LOAD LOOP ON TEST ADDRESS
9635  034530  012737  034536  001124          MOV     #T64,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
9636  034536                          T64:
9637  034536  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
9638  034542  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
9639  034546  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
9640  034552  010037  001136                  MOV     R0,$BDADR               ;SETUP REGISTER FOR ERROR MSG
9641  034556  062737  000012  001136          ADD     #RMDS,$BDADR
9642                                  ;SET VOLUME VALID USING SUBROUTINE
9643  034564  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
9644  034570  000402                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
9645  034572  104000                          ERROR                   ;RETURN HERE IF ERROR
9646  034574  000440                          BR      30$                     ;SKIP REST OF TEST IF ERROR
9647  034576                          10$:
9648                                  ;SET OFFSET MODE USING SUBROUTINE
9649  034576  004737  060706                  JSR     PC,SETOM        ;GO SET OFFSET MODE
9650  034602  000401                          BR      20$             ;BRANCH TO 20$ IF NO ERROR
9651  034604  104000                          ERROR                   ;RETURN HERE IF ERROR
9652  034606                          20$:
9653                                  ;SET 16 BIT FORMAT AND LOAD LAST SECTOR/TRACK ADDRESS
9654
9655  034606  012760  010000  000032          MOV     #FMT16,RMOF(R0) ;LOAD RMOF
9656
9657  034614  012760  002037  000006          MOV     #002037,RMDA(R0)        ;LOAD RMDA
9658                                  ;FORCE END OF BLOCK AND VERIFY THAT OFFSET MODE IS CLEARED
```

```
9659
9660  034622  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9661
9662  034630  012760  000001  000000          MOV     #GO,RMCS1(R0)   ;LOAD RMCS1
9663
9664  034636  012760  061001  000024          MOV     #DMD!MUR!DBEN!DEBL,RMMR1(R0)    ;LOAD RMMR1
9665
9666  034644  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9667
9668  034652  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
9669  034660  042737  177776  001142          BIC     #^COM,$BDDAT
9670  034666  001403                          BEQ     30$                     ;BRANCH IF OFFSET IS ZERO
9671  034670  005037  001140                  CLR     $GDDAT
9672  034674  104204                          ERROR   204                     ;OFFSET NO CLEARED BY EBL
9673  034676                          30$:                                    ;END OF TEST
9674
9675                                  ;;**********************************************************************
9676                                  ;*TEST 65        RUN AND GO TEST
9677
9678                                  ;;**********************************************************************
9679  034676  000004                  TST65:  SCOPE
9680  034700  012737  000065  001226          MOV     #65,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
9681  034706  000240                          NOP
9682  034710  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
9683  034716  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
9684  034724  012737  034740  001122          MOV     #T65,$LPADR  ;LOAD LOOP ON TEST ADDRESS
9685  034732  012737  034740  001124          MOV     #T65,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
9686  034740                          T65:
9687  034740  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
9688  034744  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
9689  034750  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
9690  034754  005037  001140                  CLR     $GDDAT                  ;INITIALIZE EXPECTED RESULT
9691  034760  005002                          CLR     R2                      ;INTIALIZE FUNCTION CODE
9692  034762  010037  001136                  MOV     R0,$BDADR
9693  034766  062737  000024  001136          ADD     #RMMR1,$BDADR
9694  034774                          10$:
9695                                  ;CLEAR THE MASSBUS AND ENABLE DEBUG CLOCK
9696  034774  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
9697  035002  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
9698
9699  035006  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
9700
9701  035014  012760  040001  000024          MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
9702
9703  035022  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
9704
9705  035030  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
9706                                  ;LOAD THE FUNCTION CODE AND VERIFY RUN AND GO FLOP
9707
9708  035036  010203                          MOV     R2,R3                   ;ASSEMBLE FUNCTION CODE AND GO
9709  035040  052703  000001                  BIS     #GO,R3
9710  035044  012737  000200  001524          MOV     #200,WATCH      ;SET WATCHDOG TIMER VALUE
9711  035052  004777  144450                  JSR     PC,@CLOCK       ;START THE CLOCK
9712
9713  035056  010360  000000                  MOV     R3,RMCS1(R0)    ;LOAD RMCS1
9714  035062                          15$:
```

```
9715
9716  035062  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
9717  035070  042737  137777  001142          BIC     #^CRG,$BDDAT
9718  035076  023737  001140  001142          CMP     $GDDAT,$BDDAT
9719  035104  001411                          BEQ     20$                     ;BRANCH IF RUN AND GO FLOP OK
9720  035106  005737  001524                  TST     WATCH           ;TAKE ANOTHER SAMPLE IF CLOCK NOT ZERO
9721  035112  001363                          BNE     15$
9722  035114  004777  144410                  JSR     PC,@STOP        ;STOP THE CLOCK
9723  035120  010237  001174                  MOV     R2,$TMP0                ;SAVE FUNCTION CODE FOR MSG
9724  035124  104205                          ERROR   205                     ;RUN AND GO INCORRECT
9725  035126  000416                          BR      40$                     ;SKIP REST OF
9726  035130                          20$:
9727  035130  004777  144374                  JSR     PC,@STOP        ;STOP THE CLOCK
9728                                  ;ADVNACE TO NEXT FUNCTION CODE - EXIT IF DONE
9729  035134  062702  000002                  ADD     #2,R2
9730  035140  022702  000076                  CMP     #ILF76,R2
9731  035144  103407                          BLO     40$                     ;EXIT IF DONE
9732  035146  020227  000050                  CMP     R2,#WCD                 ;CHANGE EXPECTED RESULTI IF
9733  035152  103403                          BLO     30$                     ;DATA COMMAND
9734  035154  012737  040000  001140          MOV     #RG,$GDDAT
9735  035162  000704                  30$:    BR      10$                     ;REPEAT TEST
9736
9737  035164                          40$:                                    ;END OF TEST
9738                                  ;;*********************************************************************
9739                                  ;*TEST 66        SET IAE TEST
9740
9741                                  ;;*********************************************************************
9742  035164  000004                  TST66:  SCOPE
9743  035166  012737  000066  001226          MOV     #66,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
9744
9745  035174  000240                          NOP
9746  035176  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
9747  035204  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
9748  035212  012737  035226  001122          MOV     #T66,$LPADR  ;LOAD LOOP ON TEST ADDRESS
9749  035220  012737  035226  001124          MOV     #T66,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
9750  035226                          T66:
9751  035226  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
9752  035232  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
9753  035236  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
9754  035242  012702  035410                  MOV     #100$,R2                ;R2=TABLE POINTER
9755  035246                          10$:
9756                                  ;SET VOLUME VALID USING SUBROUTINE
9757  035246  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
9758  035252  000402                          BR      20$             ;BRANCH TO 20$ IF NO ERROR
9759  035254  104000                          ERROR                   ;RETURN HERE IF ERROR
9760  035256  000453                          BR      50$                     ;SKIP REST OF TEST
9761  035260                          20$:
9762                                  ;LOAD INVALID TRACK, SECTOR AND CYLINDER ADDRESS AND SET FORMAT TO 18
9763
9764  035260  012760  177777  000006          MOV     #-1,RMDA(R0)    ;LOAD RMDA
9765
9766  035266  012760  177777  000034          MOV     #-1,RMDC(R0)    ;LOAD RMDC
9767
9768  035274  012760  000000  000032          MOV     #0,RMOF(R0)     ;LOAD RMOF
9769                                  ;ENABLE DEBUG CLOCK AND LOAD FUNCTION CODE IN RMCSI WITH GO ON
9770
```

```
 9771  035302  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 9772  035310  111203                          MOVB    (R2),R3
 9773  035312  042703  177701                  BIC     #^CILF76,R3
 9774  035316  052703  000001                  BIS     #GO,R3
 9775
 9776  035322  010360  000000                  MOV     R3,RMCS1(R0)    ;LOAD RMCS1
 9777  035326  116204  000001                  MOVB    1(R2),R4                ;GET CLOCK COUNT
 9778  035332  042704  177400                  BIC     #^C377,R4
 9779  035336                          30$:
 9780                                  ;CLOCK THE COMMAND SEQUENCER
 9781
 9782  035336  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
 9783
 9784  035344  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 9785  035352  005304                          DEC     R4
 9786  035354  001370                          BNE     30$
 9787                                  ;SEE IF IAE HAS SET
 9788
 9789  035356  016004  000014                  MOV     RMER1(R0),R4    ;STORE RMER1 AT R4
 9790  035362  042704  175777                  BIC     #^CIAE,R4
 9791  035366  001007                          BNE     50$                     ;BRANCH IF IAE SET
 9792                                  ;IAE DID NOT SET - TRY ANOTHER FUNCTION CODE
 9793  035370  062702  000002                  ADD     #2,R2
 9794  035374  105762  000001                  TSTB    1(R2)
 9795  035400  100401                          BMI     40$                     ;BRANCH IF ALL CODES TRIED
 9796  035402  000721                          BR      10$
 9797  035404                          40$:
 9798                                  ;CANNOT SET IAE WITH ANY COMBINATION OF ADDRESS AND FUNCTION CODE
 9799  035404  104206                          ERROR   206
 9800  035406                          50$:
 9801  035406  000411                          BR      200$                    ;JUMP OVER TABLE
 9802
 9803  035410                          100$:
 9804                                  ;TABLE OF FUNCTION CODES AND CLOCK COUNTS FOR TEST
 9805
 9806  035410  030                             .BYTE   SEARCH                  ;SEARCH COMMAND
 9807  035411  002                             .BYTE   2
 9808
 9809  035412  004                             .BYTE   SEEK                    ;SEEK COMMAND
 9810  035413  002                             .BYTE   2
 9811
 9812  035414  062                             .BYTE   WH                      ;WRITE HEADER COMMAND
 9813  035415  002                             .BYTE   2
 9814
 9815  035416  052                             .BYTE   WCH                     ;WRITE CHECK HEADER COMMAND
 9816  035417  002                             .BYTE   2
 9817
 9818  035420  072                             .BYTE   RH                      ;READ HEADER COMMAND
 9819  035421  002                             .BYTE   2
 9820
 9821  035422  060                             .BYTE   WD                      ;WRITE DATA COMMAND
 9822  035423  002                             .BYTE   2
 9823
 9824  035424  050                             .BYTE   WCD                     ;WRITE CHECK DATA COMMAND
 9825  035425  002                             .BYTE   2
 9826
```

```
 9827  035426  070                          .BYTE   RD                      ;READ DATA COMMAND
 9828  035427  002                          .BYTE   2
 9829
 9830  035430  000                          .BYTE                           ;END OF TABLE
 9831  035431  377                          .BYTE   -1
 9832
 9833  035432                       200$:                                  ;END OF TEST
 9834
 9835                                ;;*****************************************************************
 9836                                ;*TEST 67       SEARCH, SEEK, READ WRITE TEST
 9837
 9838                                ;;*****************************************************************
 9839  035432  000004               TST67:  SCOPE
 9840  035434  012737  000067 001226        MOV     #67,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
 9841
 9842  035442  000240                        NOP
 9843  035444  012737  000024 001120        MOV     #20.,$ICNT      ;20 ITERATIONS
 9844  035452  112737  000001 001131        MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
 9845  035460  012737  035474 001122        MOV     #T67,$LPADR  ;LOAD LOOP ON TEST ADDRESS
 9846  035466  012737  035474 001124        MOV     #T67,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
 9847  035474                       T67:
 9848  035474  012706  001100               MOV     #STACK,SP       ;LOAD THE STACK POINTER
 9849  035500  013700  001276               MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
 9850  035504  013701  001456               MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
 9851  035510  005002                        CLR     R2                      ;INITIALIZE FUNCTION CODE
 9852  035512                       10$:
 9853                                ;SET VOLUME VALID USING SUBROUTINE
 9854  035512  004737  060556               JSR     PC,SETVV        ;GO SET VOLUME VALID
 9855  035516  000402                        BR      20$             ;BRANCH TO 20$ IF NO ERROR
 9856  035520  104000                        ERROR                   ;RETURN HERE IF ERROR
 9857  035522  000472                        BR      50$
 9858  035524                       20$:
 9859                                ;LOAD INVALID TRACK, SECTOR AND CYLINDER ADDRESS AND SET FORMAT
 9860                                ;TO 18 BIT MODE
 9861
 9862  035524  012760  177777 000006        MOV     #-1,RMDA(R0)    ;LOAD RMDA
 9863
 9864  035532  012760  177777 000034        MOV     #-1,RMDC(R0)    ;LOAD RMDC
 9865
 9866  035540  012760  000000 000032        MOV     #0,RMOF(R0)     ;LOAD RMOF
 9867                                ;ENABLE DEBUG CLOCK AND LOAD FUNCTION CODE IN RMCS1 WITH GO ON
 9868
 9869  035546  012760  041001 000024        MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 9870  035554  010203                        MOV     R2,R3                   ;ASSEMBLE CODE AND GO
 9871  035556  052703  000001               BIS     #GO,R3
 9872
 9873  035562  010360  000000               MOV     R3,RMCS1(R0)    ;LOAD RMCS1
 9874                                ;CLOCK THE COMMAND SEQUENCER TO SET PULSE
 9875  035566  012704  000002               MOV     #2,R4
 9876  035572                       30$:
 9877
 9878  035572  012760  141001 000024        MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
 9879
 9880  035600  012760  041001 000024        MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
 9881  035606  005304                        DEC     R4
 9882  035610  001370                        BNE     30$
```

```
                                        ;VERIFY IAE ACCORDING TO FUNCTION CODE TABLE
                                        
035612  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
035620  042737  175777  001142          BIC     #^CIAE,$BDDAT
035626  016237  066042  001140          MOV     FNCDTB(R2),$GDDAT       ;ASSEMBLE EXPECTED IAE
035634  042737  175777  001140          BIC     #^CIAE,$GDDAT
035642  023737  001140  001142          CMP     $GDDAT,$BDDAT
035650  001411                          BEQ     40$                     ;BRANCH IF IAE OK
035652  010037  001136                  MOV     R0,$BDADR               ;SET UP ERROR MSG
035656  062737  000014  001136          ADD     #RMER1,$BDADR
035664  010237  001174                  MOV     R2,$TMP0
035670  104207                          ERROR   207                     ;IAE IS INCORRECT
035672  000406                          BR      50$                     ;SKIP REST OF TEST
035674                          40$:
                                        ;ADVANCE TO NEXT FUNCTIONCODE - EXIT IF DONE
035674  062702  000002                  ADD     #2,R2
035700  023702  000076                  CMP     ILF76,R2
035704  103401                          BLO     50$
035706  000701                          BR      10$
035710                          50$:                                    ;END OF TEST
                                        
                                        ;;*****************************************************************
                                        ;*TEST 70        INVALID SECTOR/TRACK TEST
                                        
                                        ;;*****************************************************************
035710  000004                  TST70:  SCOPE
035712  012737  000070  001226          MOV     #70,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
                                        
035720  000240                          NOP
035722  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
035730  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
035736  012737  035752  001122          MOV     #T70,$LPADR  ;LOAD LOOP ON TEST ADDRESS
035744  012737  035752  001124          MOV     #T70,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
035752                          T70:
035752  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
035756  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
035762  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
035766  012702  036146                  MOV     #100$,R2                ;INITIALIZE TABLE POINTER
                                        
035772                          10$:
                                        ;SET VOLUME VALID USING SUBROUTINE
035772  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
035776  000402                          BR      20$             ;BRANCH TO 20$ IF NO ERROR
036000  104000                          ERROR                   ;RETURN HERE IF ERROR
036002  000460                          BR      50$                     ;SKIP REST OF TEST
036004                          20$:
                                        ;CLEAR DESIRED CYLINDER, LOAD SECTOR/TRACK ADDRESS FROM TABLE, AND SET
                                        ;18 BIT FORMAT
                                        
036004  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
                                        
036012  012760  000000  000032          MOV     #0,RMOF(R0)     ;LOAD RMOF
                                        
036020  011260  000006                  MOV     (R2),RMDA(R0)   ;LOAD RMDA
                                        ;EXECUTE A SEARCH COMMAND TO WHERE "SET PULSE" IS ACTIVE
```

```
9939  036024  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9940
9941  036032  012760  000031  000000          MOV     #SEARCH!GO,RMCS1(R0)    ;LOAD RMCS1
9942  036040  012703  000002                  MOV     #2,R3
9943  036044                          30$:
9944
9945  036044  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
9946
9947  036052  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
9948  036060  005303                          DEC     R3
9949  036062  001370                          BNE     30$                     ;ISSUE 2 CLOCKS
9950                          ;VERIFY IAE IS SET
9951
9952  036064  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
9953  036072  042737  175777  001142          BIC     #^CIAE,$BDDAT
9954  036100  001014                          BNE     40$                     ;BRANCH IF IAE IS ON
9955  036102  012737  002000  001140          MOV     #IAE,$GDDAT             ;SETUP ERROR MESSAGE
9956  036110  010037  001136                  MOV     R0,$BDADR
9957  036114  062737  000014  001136          ADD     #RMER1,$BDADR
9958  036122  011237  001174                  MOV     (R2),$TMP0
9959  036126  104210                          ERROR   210                     ;IAE NOT SET BY RMDA ADDRESS
9960  036130  000405                          BR      50$
9961  036132                          40$:
9962                          ;ADVANCE TO NEXT ENTRY IN TABLE - EXIT IF DONE
9963  036132  062702  000002                  ADD     #2,R2
9964  036136  005712                          TST     (R2)
9965  036140  001401                          BEQ     50$                     ;EXIT IF END OF TABLE
9966  036142  000713                          BR      10$                     ;REPEAT TEST
9967  036144  000414                  50$:    BR      200$                    ;JUMP OVER TABLE
9968
9969  036146                          100$:
9970
9971                          ;TABLE OF SECTOR AND TRACK ADDRESSES FOR TEST
9972  036146      000                         .BYTE   0                       ;SECTOR ADDRESS = 0
9973  036147      005                         .BYTE   ^B00000101              ;TRACK ADDRESS = 5
9974
9975  036150      000                         .BYTE   0                       ;SECTOR = 0
9976  036151      006                         .BYTE   ^B00000110              ;TRACK  = 6
9977
9978  036152      000                         .BYTE   0                       ;SECTOR = 0
9979  036153      010                         .BYTE   ^B00001000              ;TRACK = 8
9980
9981  036154      000                         .BYTE   0                       ;SECTOR = 0
9982  036155      020                         .BYTE   ^B00010000              ;TRACK = 16
9983
9984  036156      000                         .BYTE   0                       ;SECTOR = 0
9985  036157      040                         .BYTE   ^B00100000              ;TRACK = 32
9986
9987  036160      000                         .BYTE   0                       ;SECTOR = 0
9988  036161      100                         .BYTE   ^B01000000              ;TRACK = 64
9989
9990  036162      000                         .BYTE   0                       ;SECTOR = 0
9991  036163      200                         .BYTE   ^B10000000              ;TRACK = 128
9992
9993  036164      036                         .BYTE   ^B00011110              ;SECTOR = 31
9994  036165      000                         .BYTE   0                       ;TRACK = 0
```

```
 9995
 9996  036166  040                         .BYTE   ^B00100000             ;SECTOR = 32
 9997  036167  000                         .BYTE   0                       ;TRACK = 0
 9998
 9999  036170  100                         .BYTE   ^B01000000             ;SECTOR = 64
10000  036171  000                         .BYTE   0                       ;TRACK = 0
10001
10002  036172  200                         .BYTE   ^B10000000             ;SECTOR = 128
10003  036173  000                         .BYTE   0                       ;TRACK = 0
10004
10005  036174  000                         .BYTE   0                       ;END OF TABLE
10006  036175  000                         .BYTE   0
10007
10008  036176                      200$:                                   ;END OF TEST
10009
10010                              ;;*****************************************************************
10011                              ;*TEST 71        INVALID CYLINDER TEST
10012
10013                              ;;*****************************************************************
10014  036176  000004              TST71:  SCOPE
10015  036200  012737  000071  001226     MOV     #71,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
10016
10017  036206  000240                     NOP
10018  036210  012737  000024  001120     MOV     #20.,$ICNT      ;20 ITERATIONS
10019  036216  112737  000001  001131     MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10020  036224  012737  036240  001122     MOV     #T71,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10021  036232  012737  036240  001124     MOV     #T71,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10022  036240                      T71:
10023  036240  012706  001100             MOV     #STACK,SP       ;LOAD THE STACK POINTER
10024  036244  013700  001276             MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10025  036250  013701  001456             MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10026  036254  012702  036426             MOV     #100$,R2                ;INITIALIZE TABLE POINTER
10027  036260                      10$:
10028                              ;SET VOLUME VALID USING SUBROUTINE
10029  036260  004737  060556             JSR     PC,SETVV        ;GO SET VOLUME VALID
10030  036264  000402                     BR      20$             ;BRANCH TO 20$ IF NO ERROR
10031  036266  104000                     ERROR                   ;RETURN HERE IF ERROR
10032  036270  000455                     BR      50$                     ;SKIP IF ERROR
10033  036272                      20$:
10034                              ;CLEAR RMDA, LOAD RMDC FROM TABLE
10035
10036  036272  012760  000000  000006     MOV     #0,RMDA(R0)     ;LOAD RMDA
10037
10038  036300  011260  000034             MOV     (R2),RMDC(R0)   ;LOAD RMDC
10039                              ;ENABLE DEBUG CLOCK AND EXECUTE SEARCH COMMAND
10040
10041  036304  012760  041001  000024     MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10042
10043  036312  012760  000031  000000     MOV     #SEARCH!GO,RMCS1(R0)    ;LOAD RMCS1
10044  036320  012703  000002             MOV     #2,R3
10045  036324                      30$:
10046
10047  036324  012760  141001  000024     MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
10048
10049  036332  012760  041001  000024     MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10050  036340  005303                     DEC     R3
```

```
10051  036342  001370                          BNE     30$                     ;ISSUE 2 CLOCKS
10052                                  ;VERIFY IAE IS SET
10053
10054  036344  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
10055  036352  042737  175777  001142          BIC     #^CIAE,$BDDAT
10056  036360  001014                          BNE     40$                     ;BRANCH IF IAE IS SET
10057  036362  012737  002000  001140          MOV     #IAE,$GDDAT             ;SETUP ERROR MESSAGE
10058  036370  010037  001136                  MOV     R0,$BDADR
10059  036374  062737  000014  001136          ADD     #RMER1,$BDADR
10060  036402  011237  001174                  MOV     (R2),$TMP0
10061  036406  104211                          ERROR   211                     ;IAE NOT SET BY RMDC
10062  036410  000405                          BR      50$
10063  036412                          40$:
10064                                  ;ADVANCE TABBLE POINTER - EXIT IF DONE
10065  036412  062702  000002                  ADD     #2,R2
10066  036416  005712                          TST     (R2)
10067  036420  001401                          BEQ     50$
10068  036422  000716                          BR      10$
10069  036424  000405                  50$:    BR      200$                    ;JUMP OVER TABLE
10070
10071  036426                          100$:
10072
10073                                  ;TABLE OF CYLINER ADDRESSES USED DURING TEST
10074
10075  036426  001467                          .WORD   ^B1100110111            ;CYLINDER 823
10076
10077  036430  001470                          .WORD   ^B1100111000            ;CYLINDER 824
10078
10079  036432  001500                          .WORD   ^B1101000000            ;CYLINDER 832
10080
10081  036434  001600                          .WORD   ^B1110000000            ;CYLINDER 896
10082
10083  036436  000000                          .WORD   0                       ;END OF TABLE
10084
10085  036440                          200$:                                   ;END OF TEST
10086
10087                                  ;;*****************************************************************
10088                                  ;*TEST 72        SET AOE TEST
10089
10090                                  ;;*****************************************************************
10091  036440  000004                  TST72:  SCOPE
10092  036442  012737  000072  001226          MOV     #72,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
10093
10094  036450  000240                          NOP
10095  036452  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
10096  036460  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10097  036466  012737  036502  001122          MOV     #T72,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10098  036474  012737  036502  001124          MOV     #T72,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10099  036502                          T72:
10100  036502  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
10101  036506  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10102  036512  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10103                                  ;SET VOLUME VALID USING SUBROUTINE
10104  036516  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
10105  036522  000402                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
10106  036524  104000                          ERROR                   ;RETURN HERE IF ERROR
```

```
10107  036526  000465                          BR      30$                     ;SKIP TEST IF ERROR
10108  036530                                  10$:
10109                                          ;ENABLE DEBUG CLOCK AND LOAD LAST SECTOR ADDRESS, MEMORY ADDRESS AND
10110                                          ;WORD COUNT, THEN LOAD WRITE DATA COMMAND WITH GO SET
10111
10112  036530  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10113
10114  036536  012760  000000  000032          MOV     #0,RMOF(R0)     ;LOAD RMOF
10115
10116  036544  012760  002035  000006          MOV     #002035,RMDA(R0)        ;LOAD RMDA
10117
10118  036552  012760  001466  000034          MOV     #001466,RMDC(R0)        ;LOAD RMDC
10119
10120  036560  012760  106724  000004          MOV     #BUFFER,RMBA(R0)        ;LOAD RMBA
10121
10122  036566  012760  177244  000002          MOV     #^C<2*256>+1,RMWC(R0)   ;LOAD RMWC
10123
10124  036574  012760  000061  000000          MOV     #WD!GO,RMCS1(R0)        ;LOAD RMCS1
10125  036602  012702  000014                  MOV     #14,R2                  ;R2 = CLOCK COUNT
10126                                                                          ;CLOCK COUNT 2-05 3-04 14-60...
10127                                          ;CLOCK THE COMMAND SEQUENCER TO GENERATE SET PULSE
10128  036606                                  20$:
10129
10130  036606  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
10131
10132  036614  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10133  036622  005302                          DEC     R2
10134  036624  001370                          BNE     20$                     ;ISSUE 2 CLOCKS
10135                                          ;FORCE EBL TO GET TO OVERFLOW ADDRESS
10136
10137  036626  012760  061001  000024          MOV     #DMD!MUR!DBEN!DEBL,RMMR1(R0)    ;LOAD RMMR1
10138
10139  036634  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10140                                          ;VERIFY THAT ADDRESS OVERFLOW ERROR IS SET
10141
10142  036642  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
10143  036650  042737  176777  001142          BIC     #^CAOE,$BDDAT
10144  036656  001011                          BNE     30$                     ;BRANCH IF AOE IS SET
10145  036660  010037  001136                  MOV     R0,$BDADR               ;SETUP ERROR MESSAGE
10146  036664  062737  000014  001136          ADD     #RMER1,$BDADR
10147  036672  012737  001000  001140          MOV     #AOE,$GDDAT
10148  036700  104212                          ERROR   212                     ;CANT SET AOE
10149  036702                                  30$:                            ;END OF TEST
10150
10151                                          ;;*****************************************************************
10152                                          ;*TEST 73        SET RMR TEST
10153
10154                                          ;;*****************************************************************
10155  036702  000004                          TST73:  SCOPE
10156  036704  012737  000073  001226          MOV     #73,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
10157
10158  036712  000240                          NOP
10159  036714  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
10160  036722  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10161  036730  012737  036744  001122          MOV     #T73,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10162  036736  012737  036744  001124          MOV     #T73,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
```

```
10163  036744                                  T73:
10164  036744  012706  001100                          MOV     #STACK,SP       ;LOAD THE STACK POINTER
10165  036750  013700  001276                          MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10166  036754  013701  001456                          MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10167  036760  010037  001136                          MOV     R0,$BDADR               ;SETUP REGISTER ADDRESS FOR MSG
10168  036764  062737  000014  001136                  ADD     #RMER1,$BDADR
10169  036772  012702  037146                          MOV     #100$,R2                ;INITIALIZE TABLE POINTER
10170  036776                                  10$:
10171                                          ;CLEAR THE DEVICE AND ENABLE DEBUG CLOCK
10172  036776  012760  000040  000010                  MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
10173  037004  111160  000010                          MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
10174
10175  037010  012760  000001  000024                  MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
10176
10177  037016  012760  041001  000024                  MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10178
10179  037024  012760  000000  000014                  MOV     #0,RMER1(R0)    ;LOAD RMER1
10180
10181  037032  012760  000000  000042                  MOV     #0,RMER2(R0)    ;LOAD RMER2
10182                                          ;SET GO THEN WRITE THE REGISTER SPECIFIED BY THE TABLE
10183
10184  037040  012760  000001  000000                  MOV     #GO,RMCS1(R0)   ;LOAD RMCS1
10185  037046  011203                                  MOV     (R2),R3                 ;GENERATE REGISTER ADDRESS
10186  037050  060003                                  ADD     R0,R3
10187  037052  012713  041001                          MOV     #DMD!MUR!DBEN,(R3)      ;WRITE THE REGISTER
10188                                          ;VERIFY RMR ACCORDING TO TABLE
10189
10190  037056  016037  000014  001142                  MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
10191  037064  042737  177773  001142                  BIC     #^CRMR,$BDDAT
10192  037072  016237  000002  001140                  MOV     2(R2),$GDDAT            ;GET EXPECTED RESULT FROM TABLE
10193  037100  023737  001140  001142                  CMP     $GDDAT,$BDDAT
10194  037106  001411                                  BEQ     30$             ;BRANCH IF RMR IS OK
10195  037110  011237  001174                          MOV     (R2),$TMP0      ;SAVE TEST REGISTER
10196  037114  032762  000004  000002                  BIT     #RMR,2(R2)
10197  037122  001002                                  BNE     20$             ;BRANCH IF ERROR SHOULD BE ONE
10198  037124  104213                                  ERROR   213             ;RMR SET WHEN WRITING REGISTER
10199  037126  000401                                  BR      30$
10200  037130  104214                          20$:    ERROR   214             ;RMR DID NOT SET
10201
10202  037132                                  30$:
10203                                          ;ADVANCE TABLE POINTER, EXIT IF DONE
10204  037132  062702  000004                          ADD     #4,R2
10205  037136  005712                                  TST     (R2)
10206  037140  100401                                  BMI     40$             ;EXIT IF ENTRY NEGATIVE
10207  037142  000715                                  BR      10$
10208  037144  000410                          40$:    BR      200$            ;JUMP OVER TABLE
10209
10210  037146                                  100$:
10211
10212                                          ;TABLE OF REGISTER ADDRESSES AND RMR VALUES
10213
10214  037146  000016                                  .WORD   #RMAS           ;ATTENTION SUMMARY REG
10215  037150  000000                                  .WORD                   ;RMR = 0
10216
10217  037152  000024                                  .WORD   #RMMR1          ;MAINTENANCE REG
10218  037154  000000                                  .WORD                   ;RMR = 0
```

```
10219
10220  037156  000006                          .WORD   #RMDA                   ;DISK ADDRESS REG
10221  037160  000004                          .WORD   RMR                     ;RMR = 1
10222
10223  037162  177777                          .WORD   #-1                     ;END OF TABLE
10224  037164  000000                          .WORD
10225
10226  037166                          200$:                                   ;END OF TEST
10227
10228                                  ;;*******************************************************************
10229                                  ;*TEST 74        PGM STATUS CHECK
10230
10231                                  ;;*******************************************************************
10232  037166  000004                  TST74:  SCOPE
10233  037170  012737  000074  001226          MOV     #74,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
10234
10235  037176  000240                          NOP
10236  037200  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
10237  037206  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10238  037214  012737  037230  001122          MOV     #T74,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10239  037222  012737  037230  001124          MOV     #T74,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10240  037230                          T74:
10241  037230  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
10242  037234  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10243  037240  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10244                                  ;CLEAR AND READ DRIVE TYPE AND DRIVE STATUS
10245  037244  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
10246  037252  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
10247
10248  037256  016037  000026  001174          MOV     RMDT(R0),$TMP0  ;STORE RMDT AT $TMP0
10249
10250  037264  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
10251                                  ;OMIT TEST IF DRQ IS ON - ELSE VERIFY THAT PGM IS OFF
10252  037272  032737  004000  001174          BIT     #DRQ,$TMP0
10253  037300  001014                          BNE     10$                     ;BRANCH IF DRQ IS ON
10254  037302  042737  176777  001142          BIC     #^CPGM,$BDDAT
10255  037310  001410                          BEQ     10$                     ;BRANCH IF PGM IS OFF
10256  037312  005037  001140                  CLR     $GDDAT
10257  037316  010037  001136                  MOV     R0,$BDADR
10258  037322  062737  000012  001134          ADD     #RMDS,$GDADR
10259  037330  104215                          ERROR   215                     ;DRQ IS OFF BUT PGM IS ON
10260  037332                          10$:                                    ;END OF TEST
10261
10262                                  ;;*******************************************************************
10263                                  ;*TEST 75        DPR STATUS CHECK
10264
10265                                  ;;*******************************************************************
10266  037332  000004                  TST75:  SCOPE
10267  037334  012737  000075  001226          MOV     #75,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
10268
10269  037342  000240                          NOP
10270  037344  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
10271  037352  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10272  037360  012737  037374  001122          MOV     #T75,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10273  037366  012737  037374  001124          MOV     #T75,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10274  037374                          T75:
```

```
10275  037374  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
10276  037400  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10277  037404  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10278                                  ;CLEAR AND VERIFY THAT DVA IS SET
10279  037410  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
10280  037416  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
10281
10282  037422  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
10283  037430  042737  173777  001142          BIC     #^CDVA,$BDDAT
10284  037436  001006                          BNE     10$                     ;BRANCH IF DVA IS ON
10285  037440  012737  004000  001140          MOV     #DVA,$GDDAT
10286  037446  010037  001136                  MOV     R0,$BDADR
10287  037452  104216                          ERROR   216                     ;DVA NOT SET
10288  037454                          10$:
10289                                  ;VERIFY THAT DPR IS SET
10290
10291  037454  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
10292  037462  042737  177377  001142          BIC     #^CDPR,$BDDAT
10293  037470  001011                          BNE     20$
10294  037472  012737  000400  001140          MOV     #DPR,$GDDAT
10295  037500  010037  001136                  MOV     R0,$BDADR
10296  037504  062737  000012  001136          ADD     #RMDS,$BDADR
10297  037512  104217                          ERROR   217                     ;DPR NOT SET
10298
10299  037514                          20$:                                    ;END OF TEST
10300
10301                                  ;;*****************************************************************
10302                                  ;*TEST 76        PORT REQUEST TEST, PART 1
10303
10304                                  ;;*****************************************************************
10305  037514  000004                  TST76:  SCOPE
10306  037516  012737  000076  001226          MOV     #76,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
10307
10308  037524  000240                          NOP
10309  037526  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
10310  037534  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10311  037542  012737  037556  001122          MOV     #T76,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10312  037550  012737  037556  001124          MOV     #T76,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10313  037556                          T76:
10314  037556  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
10315  037562  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10316  037566  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10317                                  ;SET VOLUME VALID USING SUBROUTINE
10318  037572  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
10319  037576  000402                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
10320  037600  104000                          ERROR                   ;RETURN HERE IF ERROR
10321  037602  000434                          BR      20$
10322  037604                          10$:
10323                                  ;EXECUTE A RELEASE TO RESET REQUEST FLOP
10324
10325  037604  012760  000013  000000          MOV     #RLEASE!GO,RMCS1(R0)    ;LOAD RMCS1
10326                                  ;READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET
10327
10328  037612  016002  000040                  MOV     RMMR2(R0),R2    ;STORE RMMR2 AT R2
10329  037616  042702  037777                  BIC     #^C<RQA!RQB>,R2
10330  037622  001024                          BNE     20$
```

```
10331                                   ;READ RMCS1 TO SET REQUEST FLOP
10332
10333  037624  016002  000000                   MOV     RMCS1(R0),R2    ;STORE RMCS1 AT R2
10334                                   ;VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET
10335
10336  037630  016005  000012                   MOV     RMDS(R0),R5     ;STORE RMDS AT R5
10337  037634  032705  001000                   BIT     #PGM,R5         ;SEE IF PGM IS SET
10338  037640  001415                           BEQ     20$             ;DONT TEST REQUEST IF PGM IS ZERO
10339
10340  037642  016037  000040  001142           MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
10341  037650  042737  037777  001142           BIC     #^C<RQA!RQB>,$BDDAT
10342  037656  001006                           BNE     20$                     ;BRANCH IF REQUEST IS SET
10343  037660  010037  001136                   MOV     R0,$BDADR
10344  037664  062737  000040  001136           ADD     #RMMR2,$BDADR
10345  037672  104220                           ERROR   220                     ;CANT SET REQUEST BY RMCS1
10346  037674                           20$:
10347
10348                                   ;;*****************************************************************
10349                                   ;*TEST 77       PORT REQUEST TEST, PART 2
10350
10351                                   ;;*****************************************************************
10352  037674  000004                   TST77:  SCOPE
10353  037676  012737  000077  001226           MOV     #77,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
10354
10355  037704  000240                           NOP
10356  037706  012737  000024  001120           MOV     #20.,$ICNT      ;20 ITERATIONS
10357  037714  112737  000001  001131           MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10358  037722  012737  037736  001122           MOV     #T77,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10359  037730  012737  037736  001124           MOV     #T77,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10360  037736                           T77:
10361  037736  012706  001100                   MOV     #STACK,SP       ;LOAD THE STACK POINTER
10362  037742  013700  001276                   MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10363  037746  013701  001456                   MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10364                                   ;SET VOLUME VALID USING SUBROUTINE
10365  037752  004737  060556                   JSR     PC,SETVV        ;GO SET VOLUME VALID
10366  037756  000402                           BR      10$             ;BRANCH TO 10$ IF NO ERROR
10367  037760  104000                           ERROR                   ;RETURN HERE IF ERROR
10368  037762  000435                           BR      20$
10369  037764                           10$:
10370                                   ;EXECUTE A RELEASE TO RESET REQUEST FLOP
10371
10372  037764  012760  000013  000000           MOV     #RLEASE!GO,RMCS1(R0)    ;LOAD RMCS1
10373                                   ;READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET
10374
10375  037772  016002  000040                   MOV     RMMR2(R0),R2    ;STORE RMMR2 AT R2
10376  037776  042702  037777                   BIC     #^C<RQA!RQB>,R2
10377  040002  001025                           BNE     20$
10378                                   ;WRITE THE ATTENTION SUMMARY REGISTER TO SET REQUEST FLOP
10379
10380  040004  012760  177777  000016           MOV     #-1,RMAS(R0)    ;LOAD RMAS
10381                                   ;VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET
10382
10383  040012  016005  000012                   MOV     RMDS(R0),R5     ;STORE RMDS AT R5
10384  040016  032705  001000                   BIT     #PGM,R5         ;SEE IF PGM IS SET
10385  040022  001415                           BEQ     20$             ;DONT TEST REQUEST IF PGM IS ZERO
10386
```

```
10387  040024  016037  000040  001142          MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
10388  040032  042737  037777  001142          BIC     #^C<RQA!RQB>,$BDDAT
10389  040040  001006                          BNE     20$
10390  040042  010037  001136                  MOV     R0,$BDADR
10391  040046  062737  000040  001136          ADD     #RMMR2,$BDADR
10392  040054  104221                          ERROR   221                     ;CANT SEE REQUEST BY RMAS
10393  040056                          20$:
10394
10395                                  ;;*******************************************************************
10396                                  ;*TEST 100       PORT REQUEST TEST, PART 3
10397
10398                                  ;;*******************************************************************
10399  040056  000004                  TST100: SCOPE
10400  040060  012737  000100  001226          MOV     #100,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
10401
10402  040066  000240                          NOP
10403  040070  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
10404  040076  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10405  040104  012737  040120  001122          MOV     #T100,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10406  040112  012737  040120  001124          MOV     #T100,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10407  040120                          T100:
*10408  040120  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
10409  040124  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10410  040130  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10411                                  ;SET VOLUME VALID USING SUBROUTINE
10412  040134  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
10413  040140  000402                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
10414  040142  104000                          ERROR                   ;RETURN HERE IF ERROR
10415  040144  000435                          BR      20$
10416  040146                          10$:
10417                                  ;EXECUTE A RELEASE COMMAND TO RESET REQUEST FLOP
10418
10419  040146  012760  000013  000000          MOV     #RLEASE!GO,RMCS1(R0)    ;LOAD RMCS1
10420                                  ;READ RMMR2 AND SKIP TEST IF REQUEST FLOPS ARENT RESET
10421
10422  040154  016002  000040                  MOV     RMMR2(R0),R2    ;STORE RMMR2 AT R2
10423  040160  042702  037777                  BIC     #^C<RQA!RQB>,R2
10424  040164  001025                          BNE     20$
10425                                  ;WRITE RMDA TO SET REQUEST FLOP
10426
10427  040166  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
10428                                  ;VERIFY THAT REQUEST FLOP IS SET IF PGM IS SET
10429
10430  040174  016005  000012                  MOV     RMDS(R0),R5     ;STORE RMDS AT R5
10431  040200  032705  001000                  BIT     #PGM,R5         ;SEE IF PGM IS SET
10432  040204  001415                          BEQ     20$             ;DONT TEST REQUEST IF PGM IS ZERO
10433
10434  040206  016037  000040  001142          MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
10435  040214  042737  037777  001142          BIC     #^C<RQA!RQB>,$BDDAT
10436  040222  001006                          BNE     20$
10437  040224  010037  001136                  MOV     R0,$BDADR
10438  040230  062737  000040  001136          ADD     #RMMR2,$BDADR
10439  040236  104222                          ERROR   222                     ;CANT SET REQUEST BY RMDA
10440
10441  040240                          20$:
10442
```

```
10443                                   ;;*****************************************************************
10444                                   ;*TEST 101       RELEASE TEST
10445
10446                                   ;;*****************************************************************
10447  040240  000004                   TST101: SCOPE
10448  040242  012737  000101  001226           MOV     #101,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
10449
10450  040250  000240                           NOP
10451  040252  012737  000024  001120           MOV     #20.,$ICNT      ;20 ITERATIONS
10452  040260  112737  000001  001131           MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10453  040266  012737  040302  001122           MOV     #T101,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10454  040274  012737  040302  001124           MOV     #T101,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10455  040302                           T101:
10456  040302  012706  001100                   MOV     #STACK,SP       ;LOAD THE STACK POINTER
10457  040306  013700  001276                   MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10458  040312  013701  001456                   MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10459                                   ;REQUEST FLOP SHOULD SET WHEN WRITING RMCS1
10460                                   ;SET VOLUME VALID USING SUBROUTINE
10461  040316  004737  060556                   JSR     PC,SETVV        ;GO SET VOLUME VALID
10462  040322  000402                           BR      10$             ;BRANCH TO 10$ IF NO ERROR
10463  040324  104000                           ERROR                   ;RETURN HERE IF ERROR
10464  040326  000454                           BR      40$
10465  040330                           10$:
10466                                   ;EXECUTE A RELEASE COMMAND
10467
10468  040330  012760  041001  000024           MOV     #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
10469
10470  040336  012760  000013  000000           MOV     #RLEASE!GO,RMCS1(R0)    ;LOAD RMCS1
10471  040344  012702  000002                   MOV     #2,R2
10472  040350                           20$:
10473
10474  040350  012760  141001  000024           MOV     #DMD!DBEN!MUR!DBCK,RMMR1(R0)    ;LOAD RMMR1
10475
10476  040356  012760  041001  000024           MOV     #DMD!DBEN!MUR,RMMR1(R0) ;LOAD RMMR1
10477  040364  005302                           DEC     R2
10478  040366  001370                           BNE     20$             ;ISSUE 2 CLOCKS
10479                                   ;VERIFY REQUEST FLOPS ARE RESET
10480
10481  040370  016037  000026  001174           MOV     RMDT(R0),$TMP0  ;STORE RMDT AT $TMP0
10482
10483  040376  016037  000040  001142           MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
10484  040404  042737  037777  001142           BIC     #^C<RQA!RQB>,$BDDAT
10485  040412  001422                           BEQ     40$             ;BRANCH IF REQUESTS ARE RESET
10486  040414  032737  004000  001174           BIT     #DRQ,$TMP0
10487  040422  001410                           BEQ     30$             ;BRANCH IF SINGLE PORT DEVICE
10488  040424  032737  100000  001142           BIT     #RQA,$BDDAT
10489  040432  001412                           BEQ     40$             ;BRANCH IF RQA IS RESET
10490  040434  032737  040000  001142           BIT     #RQB,$BDDAT
10491  040442  001406                           BEQ     40$             ;BRANCH IF RQB IS RESET
10492  040444                           30$:
10493                                   ;DRQ IS ZERO AND A REQUEST FLOP IS ON, OR, DRQ IS ONE AND
10494                                   ;BOTH REQUEST FLOPS ARE ON
10495  040444  010037  001136                   MOV     R0,$BDADR
10496  040450  062737  000040  001136           ADD     #RMMR2,$BDADR
10497  040456  104223                           ERROR   223             ;REQUEST FLOP NOT RESET
10498  040460                           40$:                            ;END OF TEST
```

```
10499
10500                                   ;;**********.*************************************************
10501                                   ;*TEST 102       WRITE ATA TEST
10502
10503                                   ;;*************************************************************
10504  040460  000004                   TST102: SCOPE
10505  040462  012737  000102  001226           MOV     #102,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
10506
10507  040470  000240                           NOP
10508  040472  012737  000024  001120           MOV     #20.,$ICNT      ;20 ITERATIONS
10509  040500  112737  000001  001131           MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10510  040506  012737  040522  001122           MOV     #T102,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10511  040514  012737  040522  001124           MOV     #T102,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10512  040522                           T102:
10513  040522  012706  001100                   MOV     #STACK,SP       ;LOAD THE STACK POINTER
10514  040526  013700  001276                   MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10515  040532  013701  001456                   MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10516                                   ;CLEAR THE DEVICE, SET DIAGNOSTIC MODE THEN SET UNIT READY
10517  040536  012760  000040  000010           MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
10518  040544  111160  000010                   MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
10519
10520  040550  012760  000001  000024           MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
10521
10522  040556  012760  000000  000014           MOV     #0,RMER1(R0)    ;LOAD RMER1
10523
10524  040564  012760  000000  000042           MOV     #0,RMER2(R0)    ;LOAD RMER2
10525
10526  040572  012760  001001  000024           MOV     #DMD!MUR,RMMR1(R0)      ;LOAD RMMR1
10527                                   ;WRITE THE ATTENTION SUMMARY REGISTER
10528  040600  111102                           MOVB    (R1),R2                 ;ASSEMBLE THE ATA BIT IN R3
10529  040602  042702  177770                   BIC     #^CUNTMSK,R2
10530  040606  116203  066142                   MOVB    ATNTBL(R2),R3
10531  040612  042703  177400                   BIC     #^CATNMSK,R3
10532
10533  040616  010360  000016                   MOV     R3,RMAS(R0)     ;LOAD RMAS
10534                                   ;READ RMDS AND VERIFY THAT ATA IS RESET
10535
10536  040622  016037  000012  001142           MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
10537  040630  042737  077777  001142           BIC     #^CATA,$BDDAT
10538  040636  001411                           BEQ     10$                     ;BRANCH IF ATA IS RESET
10539  040640  010037  001136                   MOV     R0,$BDADR
10540  040644  062737  000012  001136           ADD     #RMDS,$BDADR
10541  040652  005037  001140                   CLR     $GDDAT
10542  040656  104224                           ERROR   224                     ;CANT CLEAR ATA
10543  040660  000417                           BR      20$
10544  040662                           10$:
10545                                   ;READ RMAS AND VERIFY ATA IS RESET
10546
10547  040662  016037  000016  001142           MOV     RMAS(R0),$BDDAT ;STORE RMAS AT $BDDAT
10548  040670  005103                           COM     R3
10549  040672  040337  001142                   BIC     R3,$BDDAT
10550  040676  001410                           BEQ     20$                     ;BRANCH IF ATA IS RESET
10551  040700  005037  001140                   CLR     $GDDAT
10552  040704  010037  001136                   MOV     R0,$BDADR
10553  040710  062737  000016  001136           ADD     #RMAS,$BDADR
10554  040716  104225                           ERROR   225                     ;RMAS ATA NOT ZERO
```

```
10555  040720                          20$:                                    ;END OF TEST
10556
10557                                  ;;*****************************************************************
10558                                  ;*TEST 103      RESET ATA BY GO TEST
10559
10560                                  ;;*****************************************************************
10561  040720  000004                  TST103: SCOPE
10562  040722  012737  000103  001226          MOV     #103,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
10563
10564  040730  000240                          NOP
10565  040732  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
10566  040740  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10567  040746  012737  040762  001122          MOV     #T103,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10568  040754  012737  040762  001124          MOV     #T103,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10569  040762                          T103:
10570  040762  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
10571  040766  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10572  040772  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10573  040776  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
10574  041004  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
10575
10576  041010  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
10577
10578  041016  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
10579
10580  041024  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
10581
10582  041032  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10583                                  ;WITH DEBUG CLOCK ENABLED, SET GO AND VERIFY THAT ATA IS RESET
10584
10585  041040  012760  000001  000000          MOV     #GO,RMCS1(R0)   ;LOAD RMCS1
10586
10587  041046  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
10588  041054  042737  077777  001142          BIC     #^CATA,$BDDAT
10589  041062  001410                          BEQ     10$                     ;BRANCH IF ATA IS RESET
10590  041064  005037  001140                  CLR     $GDDAT
10591  041070  010037  001136                  MOV     R0,$BDADR
10592  041074  062737  000012  001136          ADD     #RMDS,$BDADR
10593  041102  104226                          ERROR   226                     ;CANT RESET ATA WITH GO
10594
10595  041104                          10$:                                    ;END OF TEST
10596
10597                                  ;;*****************************************************************
10598                                  ;*TEST 104      UNIT READY ATA TEST
10599
10600                                  ;;*****************************************************************
10601  041104  000004                  TST104: SCOPE
10602  041106  012737  000104  001226          MOV     #104,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
10603
10604  041114  000240                          NOP
10605  041116  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
10606  041124  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10607  041132  012737  041146  001122          MOV     #T104,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10608  041140  012737  041146  001124          MOV     #T104,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10609  041146                          T104:
10610  041146  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
```

```
10611  041152  013700  001276            MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10612  041156  013701  001456            MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10613                              ;SET DIAGNOSTIC MODE AND CLEAR ATA
10614  041162  012760  000040  000010    MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
10615  041170  111160  000010            MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
10616
10617  041174  012760  000001  000024    MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
10618
10619  041202  012760  000000  000014    MOV     #0,RMER1(R0)    ;LOAD RMER1
10620
10621  041210  012760  000000  000042    MOV     #0,RMER2(R0)    ;LOAD RMER2
10622
10623  041216  012760  177777  000016    MOV     #-1,RMAS(R0)    ;LOAD RMAS
10624                              ;SET UNIT READY AND VERIFY ATA IS SET
10625
10626  041224  012760  001001  000024    MOV     #DMD!MUR,RMMR1(R0)      ;LOAD RMMR1
10627
10628  041232  016037  000012  001142    MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
10629  041240  042737  077777  001142    BIC     #^CATA,$BDDAT
10630  041246  001011                    BNE     10$                     ;BRANCH IF ATA IS SET
10631  041250  010037  001136            MOV     R0,$BDADR
10632  041254  062737  000012  001136    ADD     #RMDS,$BDADR
10633  041262  012737  100000  001140    MOV     #ATA,$GDDAT
10634  041270  104227                    ERROR   227                     ;ATA NOT SET BY 01 READY
10635  041272                      10$:
10636                              ;CLEAR ATA, RESET UNIT READY, AND VERIFY ATA IS SET
10637
10638  041272  012760  177777  000016    MOV     #-1,RMAS(R0)    ;LOAD RMAS
10639
10640  041300  012760  000001  000024    MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
10641
10642  041306  016037  000012  001142    MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
10643  041314  042737  077777  001142    BIC     #^CATA,$BDDAT
10644  041322  001011                    BNE     20$
10645  041324  010037  001136            MOV     R0,$BDADR
10646  041330  062737  000012  001136    ADD     #RMDS,$BDADR
10647  041336  012737  100000  001140    MOV     #ATA,$GDDAT
10648  041344  104230                    ERROR   230                     ;ATA NOT SET BY 10 READY
10649
10650  041346                      20$:                                  ;END OF TEST
10651
10652                              ;;*****************************************************************
10653                              ;*TEST 105       ERROR ATA TEST
10654
10655                              ;;*****************************************************************
10656  041346  000004              TST105: SCOPE
10657  041350  012737  000105  001226    MOV     #105,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
10658
10659  041356  000240                    NOP
10660  041360  012737  000024  001120    MOV     #20.,$ICNT      ;20 ITERATIONS
10661  041366  112737  000001  001131    MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10662  041374  012737  041410  001122    MOV     #T105,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10663  041402  012737  041410  001124    MOV     #T105,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10664  041410                      T105:
10665  041410  012706  001100            MOV     #STACK,SP       ;LOAD THE STACK POINTER
10666  041414  013700  001276            MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
```

```
10667  041420  013701  001456                MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10668                              ;CLEAR THE DEVICE AND RESET ATTENTION
10669  041424  012760  000040  000010        MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
10670  041432  111160  000010                MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
10671
10672  041436  012760  000001  000024        MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
10673
10674  041444  012760  000000  000014        MOV     #0,RMER1(R0)    ;LOAD RMER1
10675
10676  041452  012760  000000  000042        MOV     #0,RMER2(R0)    ;LOAD RMER2
10677
10678  041460  012760  177777  000016        MOV     #-1,RMAS(R0)    ;LOAD RMAS
10679                              ;WRITE ONES IN ERROR REGISTER 1 AND VERIFY ATA IS SET
10680
10681  041466  012760  177777  000014        MOV     #-1,RMER1(R0)   ;LOAD RMER1
10682
10683  041474  016037  000012  001142        MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
10684  041502  042737  077777  001142        BIC     #^CATA,$BDDAT
10685  041510  001011                        BNE     10$
10686  041512  012737  100000  001140        MOV     #ATA,$GDDAT
10687  041520  010037  001136                MOV     R0,$BDADR
10688  041524  062737  000012  001136        ADD     #RMDS,$BDADR
10689  041532  104231                        ERROR   231                     ;ATA NOT SET BY COMP ERROR
10690
10691  041534                              10$:                              ;END OF TEST
10692
10693                              ;;*****************************************************************
10694                              ;*TEST 106      REGISTER TRANSFER ATA TEST
10695
10696                              ;;*****************************************************************
10697  041534  000004              TST106: SCOPE
10698  041536  012737  000106  001226        MOV     #106,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
10699  041544  000240                        NOP
10700  041546  012737  000024  001120        MOV     #20.,$ICNT      ;20 ITERATIONS
10701  041554  112737  000001  001131        MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10702  041562  012737  041576  001122        MOV     #T106,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10703  041570  012737  041576  001124        MOV     #T106,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10704  041576                              T106:
10705  041576  012706  001100                MOV     #STACK,SP       ;LOAD THE STACK POINTER
10706  041602  013700  001276                MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10707  041606  013701  001456                MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10708  041612  012702  041746                MOV     #100$,R2                ;INITIALIZE TABLE ADDRESS
10709                              ;FORCE COMPOSITE ERROR AND RESET ATA
10710  041616                              5$:
10711  041616  012760  000040  000010        MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
10712  041624  111160  000010                MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
10713
10714  041630  012760  000001  000024        MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
10715
10716  041636  012760  177777  000014        MOV     #-1,RMER1(R0)   ;LOAD RMER1
10717
10718  041644  012760  177777  000016        MOV     #-1,RMAS(R0)    ;LOAD RMAS
10719                              ;WRITE THE TEST REGISTER AND VERIFY ATA
10720  041652  011203                        MOV     (R2),R3                 ;GENERATE REGISTER ADDRESS
10721  041654  060003                        ADD     R0,R3
10722  041656  005013                        CLR     (R3)
```

```
10723
10724  041660  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
10725  041666  042737  077777  001142          BIC     #^CATA,$BDDAT
10726  041674  016237  000002  001140          MOV     2(R2),$GDDAT
10727  041702  023737  001140  001142          CMP     $GDDAT,$BDDAT
10728  041710  001410                          BEQ     10$                     ;BRANCH IF ATA IS OK
10729  041712  010337  001174                  MOV     R3,$TMP0
10730  041716  010037  001136                  MOV     R0,$BDADR
10731  041722  062737  000012  001136          ADD     #RMDS,$BDADR
10732  041730  104232                          ERROR   232                     ;ATA INCORRECT FOR REG WRITTEN
10733  041732                          10$:
10734                                  ;MOVE TABLE POINTER - EXIT IF DONE
10735  041732  062702  000004                  ADD     #4,R2
10736  041736  005712                          TST     (R2)
10737  041740  100401                          BMI     20$                     ;EXIT IF ENTRY MINUS
10738  041742  000725                          BR      5$
10739  041744  000410                  20$:    ER      200$                    ;JUMP OVER TABLE
10740
10741  041746                          100$:
10742                                  ;TABLE OF REGISTER ADDRESSES AND ATA BITS
10743  041746  000016                          .WORD   RMAS
10744  041750  000000                          .WORD
10745
10746  041752  000006                          .WORD   RMDA
10747  041754  100000                          .WORD   ATA
10748
10749  041756  000000                          .WORD   RMCS1
10750  041760  000000                          .WORD
10751
10752  041762  177777                          .WORD   -1                      ;END OF TABLE
10753  041764  000000                          .WORD
10754
10755  041766                          200$:                                   ;END OF TEST
10756
10757
10758                                  ;;*******************************************************************
10759                                  ;*TEST 107      P SET ATA TEST
10760
10761                                  ;;*******************************************************************
10762  041766  000004                  TST107: SCOPE
10763  041770  012737  000107  001226          MOV     #107,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
10764
10765  041776  000240                          NOP
10766  042000  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
10767  042006  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10768  042014  012737  042030  001122          MOV     #T107,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10769  042022  012737  042030  001124          MOV     #T107,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10770  042030                          T107:
10771  042030  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
10772  042034  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10773  042040  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10774  042044  012702  042206                  MOV     #100$,R2        ;INITIALIZE TABLE POINTER
10775
10776  042050                          10$:
10777                                  ;SET VOLUME VALID USING SUBROUTINE
10778  042050  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
```

```
10779  042054  000402                          BR      20$             ;BRANCH TO 20$ IF NO ERROR
10780  042056  104000                          ERROR                   ;RETURN HERE IF ERROR
10781  042060  000451                          BR      50$
10782  042062                          20$:
10783                                  ;EXECUTE THE COMMAND FROM THE TABLE
10784
10785  042062  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10786  042070  011203                          MOV     (R2),R3         ;GET FUNCTION CODE
10787  042072  052703  000001                  BIS     #GO,R3
10788
10789  042076  010360  000000                  MOV     R3,RMCS1(R0)    ;LOAD RMCS1
10790  042102  012703  000003                  MOV     #3,R3
10791  042106                          30$:
10792
10793  042106  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
10794
10795  042114  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10796  042122  005303                          DEC     R3
10797  042124  001370                          BNE     30$
10798                                  ;VERIFY THAT ATA IS SET
10799
10800  042126  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
10801  042134  042737  077777  001142          BIC     #^CATA,$BDDAT
10802  042142  001013                          BNE     40$
10803  042144  010037  001136                  MOV     R0,$BDADR
10804  042150  062737  000012  001136          ADD     #RMDS,$BDADR
10805  042156  012737  100000  001140          MOV     #ATA,$GDDAT
10806  042164  011237  001174                  MOV     (R2),$TMP0
10807  042170  104233                          ERROR   233             ;ATA NOT SET BY SEQUENCER
10808  042172                          40$:
10809                                  ;ADVANCE TABLE POINTER-EXIT IF DONE
10810  042172  062702  000002                  ADD     #2,R2
10811  042176  005712                          TST     (R2)
10812  042200  100401                          BMI     50$
10813  042202  000722                          BR      10$
10814  042204  000403                  50$:    BR      200$            ;JUMP OVER TABLE
10815
10816  042206                          100$:
10817                                  ;TABLE OF FUNCTION CODES
10818
10819  042206  000014                          .WORD   OFFSET
10820
10821  042210  000016                          .WORD   RTC
10822
10823  042212  177777                          .WORD   -1              ;END OF TABLE
10824  042214                          200$:
10825
10826                                  ;;*****************************************************************
10827                                  ;*TEST 110       SET WLE TEST
10828
10829                                  ;;*****************************************************************
10830  042214  000004                  TST110: SCOPE
10831  042216  012737  000110  001226          MOV     #110,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
10832
10833  042224  000240                          NOP
10834  042226  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
```

```
10835  042234  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10836  042242  012737  042256  001122          MOV     #T110,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10837  042250  012737  042256  001124          MOV     #T110,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10838  042256                              T110:
10839  042256  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
10840  042262  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10841  042266  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10842  042272  012702  042466                  MOV     #100$,R2        ;INITIALIZE TABLE POINTER
10843  042276                              10$:
10844                                      ;SET VOLUME VALID USING SUBROUTINE
10845  042276  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
10846  042302  000402                          BR      20$             ;BRANCH TO 20$ IF NO ERROR
10847  042304  104000                          ERROR                   ;RETURN HERE IF ERROR
10848  042306  000466                          BR      50$
10849  042310                              20$:
10850                                      ;ENABLE DEBUG CLOCK AND SET WRITE PROTECT ACCORDING TO TABLE
10851  042310  016203  000002                  MOV     2(R2),R3        ;GET WRITE PROTECT FROM TABLE
10852  042314  052703  041001                  BIS     #DMD!MUR!DBEN,R3        ;SET OTHER MAINT BITS
10853
10854  042320  010360  000024                  MOV     R3,RMMR1(R0)    ;LOAD RMMR1
10855                                      ;LOAD AND EXECUTE THE COMMAND FROM THE TABLE
10856  042324  011204                          MOV     (R2),R4         ;GET FUNCTION CODE FROM TABLE
10857  042326  052704  000001                  BIS     #GO,R4          ;SET GO
10858
10859  042332  010460  000000                  MOV     R4,RMCS1(R0)    ;LOAD RMCS1
10860  042336  012705  000002                  MOV     #2,R5           ;R5=CLOCK COUNT
10861  042342  052703  100000              30$:BIS     #DBCK,R3        ;SET CLOCK
10862
10863  042346  010360  000024                  MOV     R3,RMMR1(R0)    ;LOAD RMMR1
10864  042352  042703  100000                  BIC     #DBCK,R3        ;RESET CLOCK
10865
10866  042356  010360  000024                  MOV     R3,RMMR1(R0)    ;LOAD RMMR1
10867  042362  005305                          DEC     R5
10868  042364  001366                          BNE     30$             ;ISSUE 2 CLOCKS
10869                                      ;VERIFY THAT WRITE LOCK ERROR IS ACCORDING TO TABLE
10870
10871  042366  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
10872  042374  042737  173777  001142          BIC     #^CWLE,$BDDAT
10873  042402  026237  000004  001142          CMP     4(R2),$BDDAT
10874  042410  001417                          BEQ     40$             ;BRANCH IF WLE IS OK
10875  042412  016237  000004  001140          MOV     4(R2),$GDDAT    ;SAVE DATA FOR ERROR MSG
10876  042420  010037  001136                  MOV     R0,$BDADR
10877  042424  062737  000014  001136          ADD     #RMER1,$BDADR
```

```
10878  042432  011237  001174              MOV     (R2),$TMP0      ;$TMP0=FUNCTION CODE
10879  042436  016237  000002  001176      MOV     2(R2),$TMP1     ;$TMP1=WRITE PROTECT
10880  042444  104234                      ERROR   234             ;WLE INCORRECT
10881  042446  000406                      BR      50$
10882  042450                          40$:
10883                                  ;ADVANCE TABLE POINTER TO NEXT FUNCTION CODE-EXIT IF DONE
10884  042450  062702  000006              ADD     #6,R2
10885  042454  005762  000002              TST     2(R2)
10886  042460  100401                      BMI     50$
10887  042462  000705                      BR      10$             ;REPEAT TEST
10888
10889  042464  000416                  50$:BR      200$            ;JUMP OVER TABLE
10890
10891  042466                          100$:
10892
10893                                  ;TABLE OF FUNCTION CODES, WRITE PROTECT AND WRITE LOCK ERRORS
10894
10895  042466  000060                      .WORD   WD              ;WRITE DATA COMMAND
10896  042470  000010                      .WORD   MWP             ;WRITE PROTECT ON
10897  042472  004000                      .WORD   WLE             ;WRITE LOCK ERROR ONE
10898
10899  042474  000060                      .WORD   WD              ;WRITE DATA COMMAND
10900  042476  000000                      .WORD                   ;MWP OFF
10901  042500  000000                      .WORD                   ;WLE OFF
10902
10903  042502  000070                      .WORD   RD              ;READ DATA COMMAND
10904  042504  000010                      .WORD   MWP             ;MWP ON
10905  042506  000000                      .WORD                   ;WLE OFF
10906
10907  042510  000020                      .WORD   RIP             ;READ IN PRESET COMMAND
10908  042512  000010                      .WORD   MWP             ;MWP ON
10909  042514  000000                      .WORD                   ;WLE OFF
10910
10911  042516  000000                      .WORD                   ;END OF TABLE
10912  042520  177777                      .WORD   -1
10913
10914  042522                          200$:                       ;END OF TEST
10915
10916                                  ;;*****************************************************************
10917                                  ;*TEST 111      EXCEPTION TEST
10918
10919                                  ;;*****************************************************************
10920  042522  000004                  TST111: SCOPE
10921  042524  012737  000111  001226          MOV     #111,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
10922
10923                                  ;WITH OCCUPIED SET, EACH OF THE FOLLOWING ERRORS SHOULD CAUSE AN
10924                                  ;EXCEPTION:
10925                                  ;
10926                                  ;       PAR, IF RUN AND GO IS SET
10927                                  ;       RMR, IF RUN AND GO IS SET
10928                                  ;       ILR, IF RUN AND GO IS SET
10929                                  ;       DCK
10930                                  ;       HCE
10931                                  ;       HCRC
10932                                  ;       FER
10933                                  ;       OPI
```

```
10934
10935  042532  000240                          NOP
10936  042534  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
10937  042542  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
10938  042550  012737  042564  001122          MOV     #T111,$LPADR  ;LOAD LOOP ON TEST ADDRESS
10939  042556  012737  042564  001124          MOV     #T111,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
10940  042564                              T111:
10941  042564  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
10942  042570  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
10943  042574  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
10944  042600  012702  043060                  MOV     #100$,R2        ;INITIALIZE TABLE POINTER
10945  042604                              10$:
10946                                      ;INITIALIZE AND VERIFY THAT EXCEPTION IS RESET
10947  042604  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
10948  042612  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
10949
10950  042616  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
10951  042624  042737  167777  001142          BIC     #^CREX,$BDDAT
10952  042632  001410                          BEQ     20$             ;BRANCH IF EXCEPTION IS 0
10953  042634  010037  001136                  MOV     R0,$BDADR
10954  042640  062737  000024  001136          ADD     #RMMR1,$BDADR
10955  042646  005037  001140                  CLR     $GDDAT
10956  042652  104235                          ERROR   235             ;CANT CLEAR EXCEPTION
10957  042654                              20$:
10958                                      ;SET VOLUME VALID USING SUBROUTINE
10959  042654  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
10960  042660  000402                          BR      30$             ;BRANCH TO 30$ IF NO ERROR
10961  042662  104000                          ERROR                   ;RETURN HERE IF ERROR
10962  042664  000474                          BR      60$
10963  042666                              30$:
10964                                      ;EXECUTE A WRITE DATA COMMAND TO SET OCCUPIED, RUN AND GO
10965
10966  042666  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
10967
10968  042674  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
10969
10970  042702  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10971
10972  042710  012760  000061  000000          MOV     #WD!GO,RMCS1(R0)        ;LOAD RMCS1
10973  042716  012703  000002                  MOV     #2,R3
10974  042722                              40$:
10975
10976  042722  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
10977
10978  042730  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
10979  042736  005303                          DEC     R3
10980  042740  001370                          BNE     40$             ;ISSUE 2 CLOCKS
10981                                      ;LOAD ERROR REGISTER WITH ENTRY FROM TABLE AND VERIFY THAT EXCEPTION IS SET
10982
10983  042742  011260  000014                  MOV     (R2),RMER1(R0)  ;LOAD RMER1
10984  042746  012737  000200  001524          MOV     #200,WATCH      ;SET WATCHDOG TIMER VALUE
10985  042754  004777  136546                  JSR     PC,@CLOCK       ;START THE CLOCK
10986  042760                              45$:
10987
10988  042760  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
10989  042766  042737  167777  001142          BIC     #^CREX,$BDDAT
```

```
10990  042774  001021                          BNE     50$
10991  042776  005737  001524                  TST     WATCH           ;HAS CLOCK EXPIRED ??
10992  043002  001366                          BNE     45$             ;NO - TAKE ANOTHER SAMPLE
10993  043004  004777  136520                  JSR     PC,@STOP        ;STOP THE CLOCK
10994  043010  012737  010000  001140          MOV     #REX,$GDDAT
10995  043016  010037  001136                  MOV     R0,$BDADR
10996  043022  062737  000024  001136          ADD     #RMMR1,$BDADR
10997  043030  011237  001174                  MOV     (R2),$TMP0
10998  043034  104236                          ERROR   236             ;CANT SET EXCEPTION
10999  043036  000407                          BR      60$
11000  043040                          50$:
11001                                   ;ADVANCE TABLE POINTER-EXIT IF DONE
11002  043040  004777  136464                  JSR     PC,@STOP        ;STOP THE CLOCK
11003  043044  062702  000002                  ADD     #2,R2
11004  043050  005712                          TST     (R2)
11005  043052  001401                          BEQ     60$
11006  043054  000653                          BR      10$
11007  043056  000402                  60$:    BR      200$            ;JUMP OVER TABLE
11008
11009  043060                          100$:
11010                                   ;PRESENTLY, THE TABLE HAS ONLY ONE ENTRY, THE TEST USING RMR.  THE
11011                                   ;TABLE SHOULD BE EXPANDED TO TEST ALL THE CONDITIONS LISTED ABOVE IF
11012                                   ;A HARDWARE CHANGE IS MADE SUCH THAT IT IS POSSIBLE TO WRITE ERROR
11013                                   ;REGISTER 1 WITH GO SET.
11014
11015  043060  000004                          .WORD   RMR             ;RMR IS CAUSED BY HARDWARE
11016
11017  043062  000000                          .WORD                   ;END OF TABLE
11018  043064                          200$:                           ;END OF TEST
11019
11020                                   ;;*****************************************************************
11021                                   ;*TEST 112      RECALIBRATE TEST
11022
11023                                   ;;*****************************************************************
11024  043064  000004                  TST112: SCOPE
11025  043066  012737  000112  001226          MOV     #112,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
11026
11027  043074  000240                          NOP
11028  043076  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
11029  043104  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
11030  043112  012737  043126  001122          MOV     #T112,$LPADR  ;LOAD LOOP ON TEST ADDRESS
11031  043120  012737  043126  001124          MOV     #T112,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
11032  043126                          T112:
11033  043126  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
11034  043132  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
11035  043136  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
11036
11037                                   ;;*****************************************************************
11038                                   ;VERIFY THAT OPI SETS IF UNIT READY DROPS AFTER DECODE
11039
11040                                   ;SET VOLUME VALID USING SUBROUTINE
11041  043142  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
11042  043146  000403                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
11043  043150  104000                          ERROR                   ;RETURN HERE IF ERROR
11044  043152  000137  044450                  JMP     330$
11045  043156                          10$:
```

```
11046                                   ;       ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
11047
11048  043156  012760  041001  000024           MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11049
11050  043164  012760  000000  000014           MOV     #0,RMER1(R0)    ;LOAD RMER1
11051
11052  043172  012760  000000  000042           MOV     #0,RMER2(R0)    ;LOAD RMER2
11053
11054  043200  012760  000007  000000           MOV     #RECAL!GO,RMCS1(R0)     ;LOAD RMCS1
11055                                   ;       STEP COMMAND SEQUENCER TO RECAL COM (2 CLOCKS)
11056  043206  012702  000002                   MOV     #2,R2
11057  043212                          20$:
11058
11059  043212  012760  141001  000024           MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11060
11061  043220  012760  041001  000024           MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11062  043226  005302                           DEC     R2
11063  043230  001370                           BNE     20$
11064                                   ;       DROP UNIT READY AND STEP COMMAND SEQUENCER (2 CLOCKS)
11065
11066  043232  012760  040001  000024           MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
11067  043240  012702  000002                   MOV     #2,R2
11068  043244                          30$:
11069
11070  043244  012760  140001  000024           MOV     #DMD!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
11071
11072  043252  012760  040001  000024           MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
11073  043260  005302                           DEC     R2
11074  043262  001370                           BNE     30$
11075                                   ;       VERIFY THAT OPI IS SET
11076
11077  043264  016037  000014  001142           MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
11078  043272  042737  157777  001142           BIC     #^COPI,$BDDAT
11079  043300  001011                           BNE     40$
11080  043302  012737  020000  001140           MOV     #OPI,$GDDAT
11081  043310  010037  001136                   MOV     R0,$BDADR
11082  043314  062737  000014  001136           ADD     #RMER1,$BDADR
11083  043322  104241                           ERROR   241             ;OPI NOT SET DURING RECAL
11084
11085  043324  012737  043332  001124  40$:     MOV     #50$,$LPERR     ;CHANGE LOOP ON ERROR ADDRESS
11086
11087                                   ;;**************************************************************
11088                                   ;VERIFY THAT RECALIBRATE ABORTS DURING EXECUTION
11089
11090  043332                          50$:
11091                                   ;SET VOLUME VALID USING SUBROUTINE
11092  043332  004737  060556                   JSR     PC,SETVV        ;GO SET VOLUME VALID
11093  043336  000403                           BR      60$             ;BRANCH TO 60$ IF NO ERROR
11094  043340  104000                           ERROR                   ;RETURN HERE IF ERROR
11095  043342  000137  044450                   JMP     330$
11096
11097  043346                          60$:
11098                                   ;       ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
11099
11100  043346  012760  041001  000024           MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11101
```

```
11102  043354  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
11103
11104  043362  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
11105
11106  043370  012760  000007  000000          MOV     #RECAL!GO,RMCS1(R0)     ;LOAD RMCS1
11107                          ;               STEP THE COMMAND SEQUENCER TO FIRST TEST FOR ABORT (3 CLOCKS)
11108  043376  012702  000003                  MOV     #3,R2
11109  043402                  70$:
11110
11111  043402  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11112
11113  043410  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11114  043416  005302                          DEC     R2
11115  043420  001370                          BNE     70$
11116                          ;               SET DRIVE FAULT TO CAUSE ABORT CONDITION
11117
11118  043422  012760  041101  000024          MOV     #DMD!MUR!DBEN!MDF,RMMR1(R0)     ;LOAD RMMR1
11119                          ;               STEP 2 CLOCKS AND VERIFY GO IS RESET
11120  043430  012702  000002                  MOV     #2,R2
11121  043434                  80$:
11122
11123  043434  012760  141101  000024          MOV     #DMD!MUR!DBEN!MDF!DBCK,RMMR1(R0)        ;LOAD RMMR1
11124
11125  043442  012760  041101  000024          MOV     #DMD!MUR!DBEN!MDF,RMMR1(R0)     ;LOAD RMMR1
11126  043450  005302                          DEC     R2
11127  043452  001370                          BNE     80$
11128
11129  043454  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
11130  043462  042737  177776  001142          BIC     #^CGO,$BDDAT
11131  043470  001405                          BEQ     90$
11132  043472  010037  001136                  MOV     R0,$BDADR
11133  043476  005037  001140                  CLR     $GDDAT
11134  043502  104242                          ERROR   242             ;GO NOT RESET DUE TO ABORT
11135
11136  043504  012737  043512  001124  90$:    MOV     #100$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
11137
11138                          ;;*****************************************************************
11139                          ;VERIFY OPI SETS IF ON CYLINDER LATCH DOESNT CLEAR
11140
11141  043512                  100$:
11142                          ;SET VOLUME VALID USING SUBROUTINE
11143  043512  004737  060556          JSR     PC,SETVV        ;GO SET VOLUME VALID
11144  043516  000403                  BR      110$            ;BRANCH TO 110$ IF NO ERROR
11145  043520  104000                  ERROR                   ;RETURN HERE IF ERROR
11146  043522  000137  044450          JMP     330$
11147  043526                  110$:
11148                          ;               ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
11149
11150  043526  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11151
11152  043534  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
11153
11154  043542  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
11155
11156  043550  012760  000007  000000          MOV     #RECAL!GO,RMCS1(R0)     ;LOAD RMCS1
11157                          ;               STEP THE COMMAND SEQUENCER
```

```
11158  043556  012702  000017              MOV     #15.,R2
11159  043562                      120$:
11160
11161  043562  012760  141001  000024      MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11162
11163  043570  012760  041001  000024      MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11164  043576  005302                      DEC     R2
11165  043600  001370                      BNE     120$
11166                              ;       VERIFY THAT OPI IS SET
11167
11168  043602  016037  000014  001142      MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
11169  043610  042737  157777  001142      BIC     #^COPI,$BDDAT
11170  043616  001011                      BNE     130$
11171  043620  010037  001136              MOV     R0,$BDADR
11172  043624  062737  000014  001136      ADD     #RMER1,$BDADR
11173  043632  012737  020000  001140      MOV     #OPI,$GDDAT
11174  043640  104243                      ERROR   243             ;OPI NOT SET DUE TO ON LATCH
11175
11176  043642  012737  043650  001124  130$:  MOV     #150$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
11177
11178                              ;;**************************************************************
11179                              ;VERIFY ATA SETS IF DRIVE COMPLETES RECALIBRATE (ON CYLINDER SETS)
11180
11181  043650                      150$:
11182                              ;SET VOLUME VALID USING SUBROUTINE
11183  043650  004737  060556              JSR     PC,SETVV        ;GO SET VOLUME VALID
11184  043654  000403                      BR      160$            ;BRANCH TO 160$ IF NO ERROR
11185  043656  104000                      ERROR                   ;RETURN HERE IF ERROR
11186  043660  000137  044450              JMP     330$
11187  043664                      160$:
11188                              ;       ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
11189
11190  043664  012760  041401  000024      MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11191
11192  043672  012760  000000  000014      MOV     #0,RMER1(R0)    ;LOAD RMER1
11193
11194  043700  012760  000000  000042      MOV     #0,RMER2(R0)    ;LOAD RMER2
11195
11196  043706  012760  000007  000000      MOV     #RECAL!GO,RMCS1(R0)     ;LOAD RMCS1
11197                              ;       STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
11198  043714  012702  000015              MOV     #13.,R2
11199  043720                      170$:
11200
11201  043720  012760  141401  000024      MOV     #DMD!MUR!DBEN!DBCK!MOC,RMMR1(R0)        ;LOAD RMMR1
11202
11203  043726  012760  041401  000024      MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11204  043734  005302                      DEC     R2
11205  043736  001370                      BNE     170$
11206                              ;       DROP ON CYLINDER TO RESET LATCH, THEN SET ON CYLINDER
11207
11208  043740  012760  041001  000024      MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11209
11210  043746  012760  041401  000024      MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11211                              ;       STEP COMMAND SEQUENCER TO SET ATTENTION (3 CLOCKS)
11212  043754  012702  000003              MOV     #3,R2
11213  043760                      180$:
```

```
11214
11215  043760  012760  141401  000024          MOV     #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0)        ;LOAD RMMR1
11216
11217  043766  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11218  043774  005302                          DEC     R2
11219  043776  001370                          BNE     180$
11220                                  ;       VERIFY ATA IS SET
11221
11222  044000  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
11223  044006  042737  077777  001142          BIC     #^CATA,$BDDAT
11224  044014  001011                          BNE     190$
11225  044016  012737  100000  001140          MOV     #ATA,$GDDAT
11226  044024  010037  001136                  MOV     R0,$BDADR
11227  044030  062737  000012  001136          ADD     #RMDS,$BDADR
11228  044036  104244                          ERROR   244             ;ATA NOT SET BY RECAL
11229
11230  044040  012737  044046  001124  190$:   MOV     #200$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
11231
11232                                  ;;*****************************************************************
11233                                  ;VERIFY THAT RECALIBRATE ABORTS AFTER EXECUTION DURING WAIT LOOP
11234
11235  044046                          200$:
11236                                  ;SET VOLUME VALID USING SUBROUTINE
11237  044046  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
11238  044052  000402                          BR      210$            ;BRANCH TO 210$ IF NO ERROR
11239  044054  104000                          ERROR                   ;RETURN HERE IF ERROR
11240  044056  000574                          BR      330$
11241  044060                          210$:
11242                                  ;       ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
11243
11244  044060  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11245
11246  044066  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
11247
11248  044074  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
11249
11250  044102  012760  000007  000000          MOV     #RECAL!GO,RMCS1(R0)     ;LOAD RMCS1
11251                                  ;       STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
11252  044110  012702  000015                  MOV     #13.,R2
11253  044114                          220$:
11254
11255  044114  012760  141401  000024          MOV     #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0)        ;LOAD RMMR1
11256
11257  044122  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11258  044130  005302                          DEC     R2
11259  044132  001370                          BNE     220$
11260                                  ;       DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER 0
11261
11262  044134  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11263                                  ;       STEP COMMAND SEQUENCER THROUGH WAIT LOOP (7 CLOCKS)
11264  044142  012702  000007                  MOV     #7,R2
11265  044146                          230$:
11266
11267  044146  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11268
11269  044154  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
```

```
11270  044162  005302                          DEC     R2
11271  044164  001370                          BNE     230$
11272                                  ;       VERIFY THAT GO IS STILL SET
11273
11274  044166  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
11275  044174  042737  177776  001142          BIC     #^CGO,$BDDAT
11276  044202  001006                          BNE     240$
11277  044204  012737  000001  001140          MOV     #GO,$GDDAT
11278  044212  010037  001136                  MOV     R0,$BDADR
11279  044216  104245                          ERROR   245             ;GO RESET EARLY DURING RECAL
11280                                  ;       SET SEEK INCOMPLETE ERROR
11281  044220                          240$:
11282
11283  044220  012760  041201  000024          MOV     #DMD!MUR!DBEN!MSER,RMMR1(R0)    ;LOAD RMMR1
11284                                  ;       STEP COMMAND SEQUENCER AND VERIFY GO RESETS (3 CLOCKS)
11285  044226  012702  000003                  MOV     #3,R2
11286  044232                          250$:
11287
11288  044232  012760  141201  000024          MOV     #DMD!MUR!DBEN!MSER!DBCK,RMMR1(R0)       ;LOAD RMMR1
11289
11290  044240  012760  041201  000024          MOV     #DMD!MUR!DBEN!MSER,RMMR1(R0)    ;LOAD RMMR1
11291  044246  005302                          DEC     R2
11292  044250  001370                          BNE     250$
11293
11294  044252  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
11295  044260  042737  177776  001142          BIC     #^CGO,$BDDAT
11296  044266  001405                          BEQ     260$
11297  044270  010037  001136                  MOV     R0,$BDADR
11298  044274  005037  001140                  CLR     $GDDAT
11299  044300  104246                          ERROR   246             ;GO NOT RESET DUR TO WAIT ABORT
11300
11301  044302  012737  044310  001124  260$:   MOV     #300$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
11302
11303                          ;;*****************************************************************
11304                          ;VERIFY THE TAG BUS DURING RECALIBRATE
11305
11306  044310                  300$:
11307                          ;SET VOLUME VALID USING SUBROUTINE
11308  044310  004737  060556          JSR     PC,SETVV        ;GO SET VOLUME VALID
11309  044314  000402                  BR      310$            ;BRANCH TO 310$ IF NO ERROR
11310  044316  104000                  ERROR                   ;RETURN HERE IF ERROR
11311  044320  000453                  BR      330$
11312  044322                  310$:
11313                          ;       ENABLE DEBUG CLOCK AND LOAD RECALIBRATE COMMAND
11314
11315  044322  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11316
11317  044330  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
11318
11319  044336  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
11320
11321  044344  012760  000007  000000          MOV     #RECAL!GO,RMCS1(R0)     ;LOAD RMCS1
11322  044352  012702  044452                  MOV     #400$,R2        ;INITIALIZE TABLE POINTER
11323                                  ;       VERIFY TAG BUS ACCORDING TO TABLE
11324  044356                          315$:
11325
```

```
11326  044356  016037  000040  001142          MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
11327  044364  042737  150000  001142          BIC     #RQA!RQB!TST,$BDDAT
11328  044372  021237  001142                  CMP     (R2),$BDDAT
11329  044376  001411                          BEQ     320$
11330  044400  011237  001140                  MOV     (R2),$GDDAT
11331  044404  010037  001136                  MOV     R0,$BDADR
11332  044410  062737  000040  001136          ADD     #RMMR2,$BDADR
11333  044416  104247                          ERROR   247             ;INCORRECT TAG BUS DURING RECAL
11334  044420  000413                          BR      330$
11335  044422                          320$:
11336                                  ;       ADVANCE TO NEXT ENTRY IN TABLE-EXIT IF DONE
11337  044422  062702  000002                  ADD     #2,R2
11338  044426  005712                          TST     (R2)
11339  044430  100407                          BMI     330$            ;EXIT IF ENTRY NEGATIVE
11340                                  ;       STEP THE COMMAND SEQUENCER AND REPEAT VERIFICATION
11341
11342  044432  012760  151001  000024          MOV     #DMD!DBEN!MUR!MOV!DBCK,RMMR1(R0)        ;LOAD RMMR1
11343
11344  044440  012760  041401  000024          MOV     #DMD!DBEN!MUR!MOC,RMMR1(R0)     ;LOAD RMMR1
11345  044446  000743                          BR      315$
11346  044450  000416                  330$:   BR      500$            ;JUMP OVER TABLE
11347
11348  044452                          400$:
11349
11350                                  ;TABLE OF TAG BUS CONTROL AND BIT VALUES
11351
11352  044452  001777                          .WORD   1777            ;BUS BITS AT HIGH IMPEDANCE STATE
11353  044454  001777                          .WORD   1777
11354  044456  006100                          .WORD   CC!CH!BB06      ;CONTROL BITS ENABLED, BIT 6 ON
11355  044460  006100                          .WORD   CC!CH!BB06
11356  044462  026100                          .WORD   TAG!CC!CH!BB06  ;TAG COMES ON
11357  044464  026100                          .WORD   TAG!CC!CH!BB06
11358  044466  026100                          .WORD   TAG!CC!CH!BB06
11359  044470  026100                          .WORD   TAG!CC!CH!BB06
11360  044472  026100                          .WORD   TAG!CC!CH!BB06
11361  044474  026100                          .WORD   TAG!CC!CH!BB06
11362  044476  006100                          .WORD   CC!CH!BB06      ;TAG GOES OFF
11363  044500  006100                          .WORD   CC!CH!BB06
11364  044502  001777                          .WORD   1777            ;CONTROL BITS DISABLED
11365
11366  044504  177777                          .WORD   -1              ;END OF TABLE
11367
11368  044506                          500$:                           ;END OF TEST
11369
11370                                  ;;*****************************************************************
11371                                  ;*TEST 113       SEEK TEST
11372
11373                                  ;;*****************************************************************
11374  044506  000004                  TST113: SCOPE
11375  044510  012737  000113  001226          MOV     #113,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
11376
11377  044516  000240                          NOP
11378  044520  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
11379  044526  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
11380  044534  012737  044550  001122          MOV     #T113,$LPADR  ;LOAD LOOP ON TEST ADDRESS
11381  044542  012737  044550  001124          MOV     #T113,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
```

```
11382  044550                          T113:
11383  044550  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
11384  044554  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
11385  044560  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
11386
11387                                  ;;*****************************************************************
11388                                  ;VERIFY THAT OPI SETS IF UNIT READY DROPS DURING COMMAND EXECUTION
11389
11390                                  ;SET VOLUME VALID USING SUBROUTINE
11391  044564  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
11392  044570  000403                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
11393  044572  104000                          ERROR                   ;RETURN HERE IF ERROR
11394  044574  000137  046214                  JMP     330$
11395  044600                          10$:
11396                                  ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
11397                                  ;       ADDRESS,  AND LOAD SEEK COMMAND
11398
11399  044600  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11400
11401  044606  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
11402
11403  044614  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
11404
11405  044622  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
11406
11407  044630  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
11408
11409  044636  012760  000005  000000          MOV     #SEEK!GO,RMCS1(R0)      ;LOAD RMCS1
11410                                  ;       STEP COMMAND SEQUENCER TO SEEK COM (2 CLOCKS)
11411  044644  012702  000002                  MOV     #2,R2
11412  044650                          20$:
11413
11414  044650  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11415
11416  044656  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11417  044664  005302                          DEC     R2
11418  044666  001370                          BNE     20$
11419                                  ;       DROP UNIT READY AND STEP COMMAND SEQUENCER (2 CLOCKS)
11420
11421  044670  012760  040001  000024          MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
11422  044676  012702  000002                  MOV     #2,R2
11423  044702                          30$:
11424
11425  044702  012760  140001  000024          MOV     #DMD!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
11426
11427  044710  012760  040001  000024          MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
11428  044716  005302                          DEC     R2
11429  044720  001370                          BNE     30$
11430                                  ;       VERIFY THAT OPI IS SET
11431
11432  044722  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
11433  044730  042737  157777  001142          BIC     #^COPI,$BDDAT
11434  044736  001011                          BNE     40$
11435  044740  012737  020000  001140          MOV     #OPI,$GDDAT
11436  044746  010037  001136                  MOV     R0,$BDADR
11437  044752  062737  000014  001136          ADD     #RMER1,$BDADR
```

```
11438  044760  104250                          ERROR   250             ;OPI NOT SET DURING SEEK
11439
11440  044762  012737  044770  001124  40$:    MOV     #50$,$LPERR     ;CHANGE LOOP ON ERROR ADDRESS
11441
11442                                          ;;*****************************************************************
11443                                          ;VERIFY THAT SEEK ABORTS DURING EXECUTION
11444
11445  044770                                  50$:
11446                                          ;SET VOLUME VALID USING SUBROUTINE
11447  044770  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
11448  044774  000403                          BR      60$             ;BRANCH TO 60$ IF NO ERROR
11449  044776  104000                          ERROR                   ;RETURN HERE IF ERROR
11450  045000  000137  046214                  JMP     330$
11451
11452  045004                                  60$:
11453                                          ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
11454                                          ;       ADDRESS,  AND LOAD SEEK COMMAND
11455
11456  045004  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11457
11458  045012  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
11459
11460  045020  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
11461
11462  045026  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
11463
11464  045034  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
11465
11466  045042  012760  000005  000000          MOV     #SEEK!GO,RMCS1(R0)      ;LOAD RMCS1
11467                                          ;       STEP THE COMMAND SEQUENCER TO FIRST TEST FOR ABORT (3 CLOCKS)
11468  045050  012702  000003                  MOV     #3,R2
11469  045054                                  70$:
11470
11471  045054  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11472
11473  045062  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11474  045070  005302                          DEC     R2
11475  045072  001370                          BNE     70$
11476                                          ;       SET DRIVE FAULT TO CAUSE ABORT CONDITION
11477
11478  045074  012760  041101  000024          MOV     #DMD!MUR!DBEN!MDF,RMMR1(R0)     ;LOAD RMMR1
11479                                          ;       STEP 2 CLOCKS AND VERIFY GO IS RESET
11480  045102  012702  000002                  MOV     #2,R2
11481  045106                                  80$:
11482
11483  045106  012760  141101  000024          MOV     #DMD!MUR!DBEN!MDF!DBCK,RMMR1(R0)        ;LOAD RMMR1
11484
11485  045114  012760  041101  000024          MOV     #DMD!MUR!DBEN!MDF,RMMR1(R0)     ;LOAD RMMR1
11486  045122  005302                          DEC     R2
11487  045124  001370                          BNE     80$
11488
11489  045126  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
11490  045134  042737  177776  001142          BIC     #^CGO,$BDDAT
11491  045142  001405                          BEQ     90$
11492  045144  010037  001136                  MOV     R0,$BDADR
11493  045150  005037  001140                  CLR     $GDDAT
```

```
11494  045154  104251                          ERROR   251             ;GO NOT RESET DUE TO ABORT
11495
11496  045156  012737  045164  001124  90$:    MOV     #100$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
11497
11498                                          ;;*****************************************************************
11499                                          ;VERIFY OPI SETS IF ON CYLINDER LATCH DOESNT CLEAR
11500
11501  045164                                  100$:
11502                                          ;SET VOLUME VALID USING SUBROUTINE
11503  045164  004737  060556                 JSR     PC,SETVV        ;GO SET VOLUME VALID
11504  045170  000403                         BR      110$            ;BRANCH TO 110$ IF NO ERROR
11505  045172  104000                         ERROR                   ;RETURN HERE IF ERROR
11506  045174  000137  046214                 JMP     330$
11507  045200                                  110$:
11508                                          ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
11509                                          ;       ADDRESS,  AND LOAD SEEK COMMAND
11510
11511  045200  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11512
11513  045206  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
11514
11515  045214  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
11516
11517  045222  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
11518
11519  045230  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
11520
11521  045236  012760  000005  000000          MOV     #SEEK!GO,RMCS1(R0)      ;LOAD RMCS1
11522                                          ;       STEP THE COMMAND SEQUENCER
11523  045244  012702  000017                 MOV     #15.,R2
11524  045250                                  120$:
11525
11526  045250  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11527
11528  045256  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11529  045264  005302                         DEC     R2
11530  045266  001370                         BNE     120$
11531                                          ;       VERIFY THAT OPI IS SET
11532
11533  045270  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
11534  045276  042737  157777  001142          BIC     #^COPI,$BDDAT
11535  045304  001011                         BNE     130$
11536  045306  010037  001136                 MOV     R0,$BDADR
11537  045312  062737  000014  001136         ADD     #RMER1,$BDADR
11538  045320  012737  020000  001140         MOV     #OPI,$GDDAT
11539  045326  104252                         ERROR   252             ;OPI NOT SET DUE TO ON LATCH
11540
11541  045330  012737  045336  001124  130$:   MOV     #150$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
11542
11543                                          ;;*****************************************************************
11544                                          ;VERIFY ATA SETS IF DRIVE COMPLETES SEEK (ON CYLINDER SETS)
11545
11546  045336                                  150$:
11547                                          ;SET VOLUME VALID USING SUBROUTINE
11548  045336  004737  060556                 JSR     PC,SETVV        ;GO SET VOLUME VALID
11549  045342  000403                         BR      160$            ;BRANCH TO 160$ IF NO ERROR
```

```
11550  045344  104000                        ERROR                   ;RETURN HERE IF ERROR
11551  045346  000137  046214                JMP     330$
11552  045352                        160$:
11553                                ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
11554                                ;       ADDRESS,  AND LOAD SEEK COMMAND
11555
11556  045352  012760  041401  000024        MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11557
11558  045360  012760  000000  000006        MOV     #0,RMDA(R0)     ;LOAD RMDA
11559
11560  045366  012760  000000  000034        MOV     #0,RMDC(R0)     ;LOAD RMDC
11561
11562  045374  012760  000000  000014        MOV     #0,RMER1(R0)    ;LOAD RMER1
11563
11564  045402  012760  000000  000042        MOV     #0,RMER2(R0)    ;LOAD RMER2
11565
11566  045410  012760  000005  000000        MOV     #SEEK!GO,RMCS1(R0)      ;LOAD RMCS1
11567                                ;       STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
11568  045416  012702  000015                MOV     #13.,R2
11569  045422                        170$:
11570
11571  045422  012760  141401  000024        MOV     #DMD!MUR!DBEN!DBCK!MOC,RMMR1(R0)        ;LOAD RMMR1
11572
11573  045430  012760  041401  000024        MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11574  045436  005302                        DEC     R2
11575  045440  001370                        BNE     170$
11576                                ;       DROP ON CYLINDER TO RESET LATCH, THEN SET ON CYLINDER
11577
11578  045442  012760  041001  000024        MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11579
11580  045450  012760  041401  000024        MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11581                                ;       STEP COMMAND SEQUENCER TO SET ATTENTION (3 CLOCKS)
11582  045456  012702  000003                MOV     #3,R2
11583  045462                        180$:
11584
11585  045462  012760  141401  000024        MOV     #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0)        ;LOAD RMMR1
11586
11587  045470  012760  041401  000024        MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11588  045476  005302                        DEC     R2
11589  045500  001370                        BNE     180$
11590                                ;       VERIFY ATA IS SET
11591
11592  045502  016037  000012  001142        MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
11593  045510  042737  077777  001142        BIC     #^CATA,$BDDAT
11594  045516  001011                        BNE     190$
11595  045520  012737  100000  001140        MOV     #ATA,$GDDAT
11596  045526  010037  001136                MOV     R0,$BDADR
11597  045532  062737  000012  001136        ADD     #RMDS,$BDADR
11598  045540  104253                        ERROR   253             ;ATA NOT SET BY SEEK
11599
11600  045542  012737  045550  001124  190$:  MOV     #200$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
11601
11602                                ;;*****************************************************************
11603                                ;VERIFY THAT SEEK ABORTS AFTER EXECUTION DURING WAIT LOOP
11604
11605  045550                        200$:
```

```
11606                                   ;SET VOLUME VALID USING SUBROUTINE
11607  045550  004737  060556                JSR     PC,SETVV        ;GO SET VOLUME VALID
11608  045554  000403                        BR      210$            ;BRANCH TO 210$ IF NO ERROR
11609  045556  104000                        ERROR                   ;RETURN HERE IF ERROR
11610  045560  000137  046214                JMP     330$
11611  045564                          210$:
11612                                   ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
11613                                   ;       ADDRESS  AND LOAD SEEK COMMAND
11614
11615  045564  012760  041401  000024        MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11616
11617  045572  012760  000000  000006        MOV     #0,RMDA(R0)     ;LOAD RMDA
11618
11619  045600  012760  000000  000034        MOV     #0,RMDC(R0)     ;LOAD RMDC
11620
11621  045606  012760  000000  000014        MOV     #0,RMER1(R0)    ;LOAD RMER1
11622
11623  045614  012760  000000  000042        MOV     #0,RMER2(R0)    ;LOAD RMER2
11624
11625  045622  012760  000005  000000        MOV     #SEEK!GO,RMCS1(R0)      ;LOAD RMCS1
11626                                   ;       STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (13 CLOCKS)
11627  045630  012702  000015                MOV     #13.,R2
11628  045634                          220$:
11629
11630  045634  012760  141401  000024        MOV     #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0)        ;LOAD RMMR1
11631
11632  045642  012760  041401  000024        MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11633  045650  005302                        DEC     R2
11634  045652  001370                        BNE     220$
11635                                   ;       DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER 0
11636
11637  045654  012760  041001  000024        MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11638                                   ;       STEP COMMAND SEQUENCER THROUGH WAIT LOOP (7 CLOCKS)
11639  045662  012702  000007                MOV     #7,R2
11640  045666                          230$:
11641
11642  045666  012760  141001  000024        MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11643
11644  045674  012760  041001  000024        MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11645  045702  005302                        DEC     R2
11646  045704  001370                        BNE     230$
11647                                   ;       VERIFY THAT GO IS STILL SET
11648
11649  045706  016037  000000  001142        MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
11650  045714  042737  177776  001142        BIC     #^CGO,$BDDAT
11651  045722  001006                        BNE     240$
11652  045724  012737  000001  001140        MOV     #GO,$GDDAT
11653  045732  010037  001136                MOV     R0,$BDADR
11654  045736  104254                        ERROR   254             ;GO RESET EARLY DURING SEEK
11655                                   ;       SET SEEK INCOMPLETE ERROR
11656  045740                          240$:
11657
11658  045740  012760  041201  000024        MOV     #DMD!MUR!DBEN!MSER,RMMR1(R0)    ;LOAD RMMR1
11659                                   ;       STEP COMMAND SEQUENCER AND VERIFY GO RESETS (3 CLOCKS)
11660  045746  012702  000003                MOV     #3,R2
11661  045752                          250$:
```

```
11662
11663  045752  012760  141201  000024          MOV     #DMD!MUR!DBEN!MSER!DBCK,RMMR1(R0)       ;LOAD RMMR1
11664
11665  045760  012760  041201  000024          MOV     #DMD!MUR!DBEN!MSER,RMMR1(R0)    ;LOAD RMMR1
11666  045766  005302                          DEC     R2
11667  045770  001370                          BNE     250$
11668
11669  045772  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
11670  046000  042737  177776  001142          BIC     #^CGO,$BDDAT
11671  046006  001405                          BEQ     260$
11672  046010  010037  001136                  MOV     R0,$BDADR
11673  046014  005037  001140                  CLR     $GDDAT
11674  046020  104255                          ERROR   255             ;GO NOT RESET DUE TO WAIT ABORT
11675
11676  046022  012737  046034  001124  260$:  MOV     #300$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
11677  046030  012703  000001                  MOV     #1,R3           ;INITIALIZE CYLINDER ADDRESS
11678
11679                                  ;;*****************************************************************
11680                                  ;VERIFY THE TAG BUS DURING SEEK
11681
11682  046034                          300$:
11683                                  ;SET VOLUME VALID USING SUBROUTINE
11684  046034  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
11685  046040  000402                          BR      310$            ;BRANCH TO 310$ IF NO ERROR
11686  046042  104000                          ERROR                   ;RETURN HERE IF ERROR
11687  046044  000463                          BR      330$
11688  046046                          310$:
11689                                  ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
11690                                  ;       ADDRESS  AND LOAD SEEK COMMAND
11691
11692  046046  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11693
11694  046054  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
11695
11696  046062  010360  000034                  MOV     R3,RMDC(R0)     ;LOAD RMDC
11697
11698  046066  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
11699
11700  046074  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
11701
11702  046102  012760  000005  000000          MOV     #SEEK!GO,RMCS1(R0)      ;LOAD RMCS1
11703  046110  012702  046230                  MOV     #400$,R2        ;INITIALIZE TABLE POINTER
11704                                  ;
11705  046114                          315$:   VERIFY TAG BUS ACCORDING TO TABLE AND CYLINDER IN R3
11706
11707  046114  016037  000040  001142          MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
11708  046122  042737  150000  001142          BIC     #RQA!RQB!TST,$BDDAT
11709  046130  011237  001140                  MOV     (R2),$GDDAT
11710  046134  050337  001140                  BIS     R3,$GDDAT       ;OR CYLINDER ADDRESS IN
11711  046140  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;COMPARE EXPECTED AND RECEIVED
11712  046146  001407                          BEQ     320$            ;BRANCH IF TAG BUS OK
11713  046150  010037  001136                  MOV     R0,$BDADR
11714  046154  062737  000040  001136          ADD     #RMMR2,$BDADR
11715  046162  104256                          ERROR   256             ;INCORRECT TAG BUS DURING SEEK
11716  046164  000413                          BR      330$
11717  046166                          320$:
```

```
11718                                   ;       ADVANCE TO NEXT ENTRY IN TABLE-EXIT IF DONE
11719  046166  062702  000002                   ADD     #2,R2
11720  046172  005712                           TST     (R2)
11721  046174  100407                           BMI     330$            ;EXIT IF ENTRY NEGATIVE
11722                                   ;       STEP THE COMMAND SEQUENCER AND REPEAT VERIFICATION
11723
11724  046176  012760  151001  000024           MOV     #DMD!DBEN!MUR!MOV!DBCK,RMMR1(R0)        ;LOAD RMMR1
11725
11726  046204  012760  041401  000024           MOV     #DMD!DBEN!MUR!MOC,RMMR1(R0)     ;LOAD RMMR1
11727  046212  000740                           BR      315$
11728  046214                          330$:
11729                                   ;REPEAT TAG BUS TEST FOR EACH PRIME CYLINDER, I.E., 1,2,4,...
11730
11731  046214  006303                           ASL     R3              ;SHIFT TO NEXT CYLINDER
11732  046216  020327  000512                   CMP     R3,#512
11733  046222  101001                           BHI     340$            ;EXIT IF 512 WAS DONE
11734  046224  000703                           BR      300$            ;TEST NEXT CYLINDER
11735  046226  000416                  340$:    BR      500$            ;JUMP OVER TABLE
11736
11737  046230                          400$:
11738
11739                                   ;TABLE OF TAG BUS CONTROL AND BIT VALUES
11740
11741  046230  001777                           .WORD   1777            ;BUS BITS AT HIGH IMPEDANCE STATE
11742  046232  001777                           .WORD   1777
11743  046234  004000                           .WORD   CC      ;CONTROL BITS ENABLED, BIT 6 ON
11744  046236  004000                           .WORD   CC
11745  046240  024000                           .WORD   TAG!CC  ;TAG COMES ON
11746  046242  024000                           .WORD   TAG!CC
11747  046244  024000                           .WORD   TAG!CC
11748  046246  024000                           .WORD   TAG!CC
11749  046250  024000                           .WORD   TAG!CC
11750  046252  024000                           .WORD   TAG!CC
11751  046254  004000                           .WORD   CC      ;TAG GOES OFF
11752  046256  004000                           .WORD   CC
11753  046260  001777                           .WORD   1777            ;CONTROL BITS DISABLED
11754
11755  046262  177777                           .WORD   -1              ;END OF TABLE
11756
11757  046264                          500$:                           ;END OF TEST
11758
11759                                   ;;*****************************************************************
11760                                   ;*TEST 114       SEARCH TEST
11761
11762                                   ;;*****************************************************************
11763  046264  000004                  TST114: SCOPE
11764  046266  012737  000114  001226           MOV     #114,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
11765
11766  046274  000240                           NOP
11767  046276  012737  000024  001120           MOV     #20.,$ICNT      ;20 ITERATIONS
11768  046304  112737  000001  001131           MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
11769  046312  012737  046326  001122           MOV     #T114,$LPADR  ;LOAD LOOP ON TEST ADDRESS
11770  046320  012737  046326  001124           MOV     #T114,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
11771  046326                          T114:
11772  046326  012706  001100                   MOV     #STACK,SP       ;LOAD THE STACK POINTER
11773  046332  013700  001276                   MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
```

```
11774  046336  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
11775
11776                                  ;;*****************************************************************
11777                                  ;VERIFY THAT OPI SETS IF UNIT READY DROPS DURING COMMAND EXECUTION
11778
11779                                  ;SET VOLUME VALID USING SUBROUTINE
11780  046342  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
11781  046346  000403                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
11782  046350  104000                          ERROR                   ;RETURN HERE IF ERROR
11783  046352  000137  050410                  JMP     330$
11784  046356                          10$:
11785                                  ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
11786                                  ;       ADDRESS,  AND LOAD SEARCH COMMAND
11787
11788  046356  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11789
11790  046364  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
11791
11792  046372  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
11793
11794  046400  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
11795
11796  046406  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
11797
11798  046414  012760  000031  000000          MOV     #SEARCH!GO,RMCS1(R0)    ;LOAD RMCS1
11799                                  ;       STEP COMMAND SEQUENCER TO SEARCH COM (2 CLOCKS)
11800  046422  012702  000002                  MOV     #2,R2
11801  046426                          20$:
11802
11803  046426  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11804
11805  046434  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11806  046442  005302                          DEC     R2
11807  046444  001370                          BNE     20$
11808                                  ;       DROP UNIT READY AND STEP COMMAND SEQUENCER (2 CLOCKS)
11809
11810  046446  012760  040001  000024          MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
11811  046454  012702  000002                  MOV     #2,R2
11812  046460                          30$:
11813
11814  046460  012760  140001  000024          MOV     #DMD!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
11815
11816  046466  012760  040001  000024          MOV     #DMD!DBEN,RMMR1(R0)     ;LOAD RMMR1
11817  046474  005302                          DEC     R2
11818  046476  001370                          BNE     30$
11819                                  ;       VERIFY THAT OPI IS SET
11820
11821  046500  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
11822  046506  042737  157777  001142          BIC     #^COPI,$BDDAT
11823  046514  001011                          BNE     40$
11824  046516  012737  020000  001140          MOV     #OPI,$GDDAT
11825  046524  010037  001136                  MOV     R0,$BDADR
11826  046530  062737  000014  001136          ADD     #RMER1,$BDADR
11827  046536  104257                          ERROR   257             ;OPI NOT SET DURING SEARCH
11828
11829  046540  012737  046546  001124  40$:    MOV     #50$,$LPERR     ;CHANGE LOOP ON ERROR ADDRESS
```

```
11830
11831                                   ;;**********  **********
11832                                   ;VERIFY THAT SEARCH ABORTS DURING EXECUTION
11833
11834  046546                           50$:
11835                                   ;SET VOLUME VALID USING SUBROUTINE
11836  046546  004737  060556                 JSR     PC,SETVV        ;GO SET VOLUME VALID
11837  046552  000403                         BR      60$             ;BRANCH TO 60$ IF NO ERROR
11838  046554  104000                         ERROR                   ;RETURN HERE IF ERROR
11839  046556  000137  050410                 JMP     330$
11840
11841  046562                           60$:
11842                                   ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
11843                                   ;       ADDRESS,  AND LOAD SEARCH COMMAND
11844
11845  046562  012760  041001  000024         MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11846
11847  046570  012760  000000  000006         MOV     #0,RMDA(R0)     ;LOAD RMDA
11848
11849  046576  012760  000000  000034         MOV     #0,RMDC(R0)     ;LOAD RMDC
11850
11851  046604  012760  000000  000014         MOV     #0,RMER1(R0)    ;LOAD RMER1
11852
11853  046612  012760  000000  000042         MOV     #0,RMER2(R0)    ;LOAD RMER2
11854
11855  046620  012760  000031  000000         MOV     #SEARCH!GO,RMCS1(R0)    ;LOAD RMCS1
11856                                   ;       STEP THE COMMAND SEQUENCER TO FIRST TEST FOR ABORT (3 CLOCKS)
11857  046626  012702  000003                 MOV     #3,R2
11858  046632                           70$:
11859
11860  046632  012760  141001  000024         MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11861
11862  046640  012760  041001  000024         MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11863  046646  005302                         DEC     R2
11864  046650  001370                         BNE     70$
11865                                   ;       SET DRIVE FAULT TO CAUSE ABORT CONDITION
11866
11867  046652  012760  041101  000024         MOV     #DMD!MUR!DBEN!MDF,RMMR1(R0)     ;LOAD RMMR1
11868                                   ;       STEP 2 CLOCKS AND VERIFY GO IS RESET
11869  046660  012702  000002                 MOV     #2,R2
11870  046664                           80$:
11871
11872  046664  012760  141101  000024         MOV     #DMD!MUR!DBEN!MDF!DBCK,RMMR1(R0)        ;LOAD RMMR1
11873
11874  046672  012760  041101  000024         MOV     #DMD!MUR!DBEN!MDF,RMMR1(R0)     ;LOAD RMMR1
11875  046700  005302                         DEC     R2
11876  046702  001370                         BNE     80$
11877
11878  046704  016037  000000  001142         MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
11879  046712  042737  177776  001142         BIC     #^CGO,$BDDAT
11880  046720  001405                         BEQ     90$
11881  046722  010037  001136                 MOV     R0,$BDADR
11882  046726  005037  001140                 CLR     $GDDAT
11883  046732  104260                         ERROR   260             ;GO NOT RESET DUE TO ABORT
11884
11885                                   ;(THE OTHER TWO ABORT TESTS IN THE COMMAND SEQUENCER ARE TESTED
```

```
11886                                   ;DURING DATA COMMAND TESTS)
11887
11888  046734  012737  046742  001124   90$:    MOV     #100$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
11889
11890                                   ;;*****************************************************************
11891                                   ;VERIFY OPI SETS IF ON CYLINDER LATCH DOESNT CLEAR
11892
11893  046742                           100$:
11894                                   ;SET VOLUME VALID USING SUBROUTINE
11895  046742  004737  060556                   JSR     PC,SETVV        ;GO SET VOLUME VALID
11896  046746  000403                           BR      110$            ;BRANCH TO 110$ IF NO ERROR
11897  046750  104000                           ERROR                   ;RETURN HERE IF ERROR
11898  046752  000137  050410                   JMP     330$
11899  046756                           110$:
11900                                   ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
11901                                   ;       ADDRESS,  AND LOAD SEARCH COMMAND
11902
11903  046756  012760  041001  000024         MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11904
11905  046764  012760  000000  000006         MOV     #0,RMDA(R0)     ;LOAD RMDA
11906
11907  046772  012760  000000  000034         MOV     #0,RMDC(R0)     ;LOAD RMDC
11908
11909  047000  012760  000000  000014         MOV     #0,RMER1(R0)    ;LOAD RMER1
11910
11911  047006  012760  000000  000042         MOV     #0,RMER2(R0)    ;LOAD RMER2
11912
11913  047014  012760  000031  000000         MOV     #SEARCH!GO,RMCS1(R0)    ;LOAD RMCS1
11914                                   ;       STEP THE COMMAND SEQUENCER
11915  047022  012702  000023                 MOV     #19.,R2
11916  047026                           120$:
11917
11918  047026  012760  141001  000024         MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
11919
11920  047034  012760  041001  000024         MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
11921  047042  005302                         DEC     R2
11922  047044  001370                         BNE     120$
11923                                   ;       VERIFY THAT OPI IS SET
11924
11925  047046  016037  000014  001142         MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
11926  047054  042737  157777  001142         BIC     #^COPI,$BDDAT
11927  047062  001011                         BNE     130$
11928  047064  010037  001136                 MOV     R0,$BDADR
11929  047070  062737  000014  001136         ADD     #RMER1,$BDADR
11930  047076  012737  020000  001140         MOV     #OPI,$GDDAT
11931  047104  104261                         ERROR   261             ;OPI NOT SET DUE TO ON LATCH
11932
11933  047106  012737  047114  001124   130$:   MOV     #150$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
11934
11935                                   ;;*****************************************************************
11936                                   ;VERIFY ATA SETS IF DRIVE COMPLETES SEARCH (ON CYLINDER AND
11937                                   ;SECTOR COMPARE SETS)
11938
11939  047114                           150$:
11940                                   ;SET VOLUME VALID USING SUBROUTINE
11941  047114  004737  060556                   JSR     PC,SETVV        ;GO SET VOLUME VALID
```

```
11942  047120  000403                          BR      160$            ;BRANCH TO 160$ IF NO ERROR
11943  047122  104000                          ERROR                   ;RETURN HERE IF ERROR
11944  047124  000137  050410                  JMP     330$
11945  047130                          160$:
11946                                  ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
11947                                  ;       ADDRESS,  AND LOAD SEARCH COMMAND
11948
11949  047130  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11950
11951  047136  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
11952
11953  047144  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
11954
11955  047152  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
11956
11957  047160  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
11958
11959  047166  012760  000031  000000          MOV     #SEARCH!GO,RMCS1(R0)    ;LOAD RMCS1
11960                                  ;       STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (17 CLOCKS)
11961  047174  012702  000021                  MOV     #17.,R2
11962  047200                          170$:
11963
11964  047200  012760  141401  000024          MOV     #DMD!MUR!DBEN!DBCK!MOC,RMMR1(R0)        ;LOAD RMMR1
11965
11966  047206  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
11967  047214  005302                          DEC     R2
11968  047216  001370                          BNE     170$
11969                                  ;       DROP ON CYLINDER TO RESET LATCH,AND RAISE INDEX PULSE
11970                                  ;TO SET FORMAT CHANGE FLOP
11971
11972
11973  047220  012760  041005  000024          MOV     #DMD!MUR!DBEN!MI,RMMR1(R0)      ;LOAD RMMR1
11974                                  ;       RAISE ON CYLINDER AND INHIBIT SEARCH TIMEOUT
11975
11976  047226  012760  051401  000024          MOV     #DMD!MUR!DBEN!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
11977                                  ;       STEP COMMAND SEQUENCER TO SEARCH ENABLE (2 CLOCKS)
11978  047234  012702  000002                  MOV     #2,R2
11979  047240                          180$:
11980
11981  047240  012760  151401  000024          MOV     #DMD!MUR!DBEN!MOC!DTO!DBCK,RMMR1(R0)    ;LOAD RMMR1
11982
11983  047246  012760  051401  000024          MOV     #DMD!MUR!DBEN!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
11984  047254  005302                          DEC     R2
11985  047256  001370                          BNE     180$
11986                                  ;FORCE SECTOR COMPARE BY CLOCKING SECTOR PULSE WITH SECTOR COMPARE
11987                                  ;ACTIVE
11988
11989  047260  012760  051403  000024          MOV     #DMD!MUR!MOC!DBEN!DTO!MSC,RMMR1(R0)     ;LOAD RMMR1
11990
11991  047266  012760  051443  000024          MOV     #DMD!MUR!MOC!DBEN!DTO!MSC!MS,RMMR1(R0)  ;LOAD RMMR1
11992
11993  047274  012760  051403  000024          MOV     #DMD!MUR!MOC!DBEN!DTO!MSC,RMMR1(R0)     ;LOAD RMMR1
11994                                  ;CLOCK SEQUENCER TO SET ATA (3 CLOCKS)
11995  047302  012702  000003                  MOV     #3,R2           ;R2 = CLOCK COUNT
11996  047306                          185$:
11997
```

```
11998  047306  012760  151401  000024          MOV     #DMD!MUR!MOC!DBEN!DTO!DBCK,RMMR1(R0)    ;LOAD RMMR1
11999
12000  047314  012760  051401  000024          MOV     #DMD!MUR!MOC!DBEN!DTO,RMMR1(R0) ;LOAD RMMR1
12001  047322  005302                          DEC     R2
12002  047324  001370                          BNE     185$
12003                                  ;       VERIFY ATA IS SET
12004
12005  047326  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
12006  047334  042737  077777  001142          BIC     #^CATA,$BDDAT
12007  047342  001011                          BNE     190$
12008  047344  012737  100000  001140          MOV     #ATA,$GDDAT
12009  047352  010037  001136                  MOV     R0,$BDADR
12010  047356  062737  000012  001136          ADD     #RMDS,$BDADR
12011  047364  104262                          ERROR   262             ;ATA NOT SET BY SEARCH
12012
12013  047366  012737  047374  001124  190$:   MOV     #200$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
12014
12015                                  ;;**********************************************************************
12016                                  ;VERIFY THAT SEARCH ABORTS AFTER EXECUTION DURING SEARCH SEEK LOOP
12017
12018  047374                          200$:
12019                                  ;SET VOLUME VALID USING SUBROUTINE
12020  047374  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
12021  047400  000403                          BR      210$            ;BRANCH TO 210$ IF NO ERROR
12022  047402  104000                          ERROR                   ;RETURN HERE IF ERROR
12023  047404  000137  050410                  JMP     330$
12024  047410                          210$:
12025                                  ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
12026                                  ;       ADDRESS  AND LOAD SEARCH COMMAND
12027
12028  047410  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)      ;LOAD RMMR1
12029
12030  047416  012760  000000  000006          MOV     #0,RMDA(R0)      ;LOAD RMDA
12031
12032  047424  012760  000000  000034          MOV     #0,RMDC(R0)      ;LOAD RMDC
12033
12034  047432  012760  000000  000014          MOV     #0,RMER1(R0)     ;LOAD RMER1
12035
12036  047440  012760  000000  000042          MOV     #0,RMER2(R0)     ;LOAD RMER2
12037
12038  047446  012760  000031  000000          MOV     #SEARCH!GO,RMCS1(R0)     ;LOAD RMCS1
12039                                  ;       STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (17 CLOCKS)
12040  047454  012702  000021                  MOV     #17.,R2
12041  047460                          220$:
12042
12043  047460  012760  141401  000024          MOV     #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0)         ;LOAD RMMR1
12044
12045  047466  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)      ;LOAD RMMR1
12046  047474  005302                          DEC     R2
12047  047476  001370                          BNE     220$
12048                                  ;       DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER 0
12049
12050  047500  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
12051                                  ;       STEP COMMAND SEQUENCER THROUGH WAIT LOOP (7 CLOCKS)
12052  047506  012702  000007                  MOV     #7,R2
12053  047512                          230$:
```

```
12054
12055  047512  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
12056
12057  047520  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
12058  047526  005302                          DEC     R2
12059  047530  001370                          BNE     230$
12060                                  ;       VERIFY THAT GO IS STILL SET
12061
12062  047532  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
12063  047540  042737  177776  001142          BIC     #^CGO,$BDDAT
12064  047546  001006                          BNE     240$
12065  047550  012737  000001  001140          MOV     #GO,$GDDAT
12066  047556  010037  001136                  MOV     R0,$BDADR
12067  047562  104263                          ERROR   263             ;GO RESET EARLY DURING SEARCH
12068                                  ;       SET SEEK INCOMPLETE ERROR
12069  047564                          240$:
12070
12071  047564  012760  041201  000024          MOV     #DMD!MUR!DBEN!MSER,RMMR1(R0)    ;LOAD RMMR1
12072                                  ;       STEP COMMAND SEQUENCER AND VERIFY GO RESETS (3 CLOCKS)
12073  047572  012702  000003                  MOV     #3,R2
12074  047576                          250$:
12075
12076  047576  012760  141201  000024          MOV     #DMD!MUR!DBEN!MSER!DBCK,RMMR1(R0)       ;LOAD RMMR1
12077
12078  047604  012760  041201  000024          MOV     #DMD!MUR!DBEN!MSER,RMMR1(R0)    ;LOAD RMMR1
12079  047612  005302                          DEC     R2
12080  047614  001370                          BNE     250$
12081
12082  047616  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
12083  047624  042737  177776  001142          BIC     #^CGO,$BDDAT
12084  047632  001405                          BEQ     260$
12085  047634  010037  001136                  MOV     R0,$BDADR
12086  047640  005037  001140                  CLR     $GDDAT
12087  047644  104264                          ERROR   264             ;GO NOT RESET DUE TO WAIT ABORT
12088
12089  047646  012737  047654  001124  260$:   MOV     #265$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
12090
12091                                  ;;*****************************************************************
12092                                  ;VERIFY THAT SEARCH ABORTS DURING SECTOR COMPARE LOOP
12093
12094  047654                          265$:
12095                                  ;SET VOLUME VALID USING SUBROUTINE
12096  047654  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
12097  047660  000403                          BR      270$            ;BRANCH TO 270$ IF NO ERROR
12098  047662  104000                          ERROR                   ;RETURN HERE IF ERROR
12099  047664  000137  050410                  JMP     330$
12100  047670                          270$:
12101                                  ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
12102                                  ;       ADDRESS,  AND LOAD SEARCH COMMAND
12103
12104  047670  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
12105
12106  047676  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
12107
12108  047704  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
12109
```

```
12110  047712  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
12111
12112  047720  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
12113
12114  047726  012760  000031  000000          MOV     #SEARCH!GO,RMCS1(R0)    ;LOAD RMCS1
12115                          ;       STEP COMMAND SEQUENCER TO FIRST ON LATCH TEST (17 CLOCKS)
12116  047734  012702  000021                  MOV     #17.,R2
12117  047740                  275$:
12118
12119  047740  012760  141401  000024          MOV     #DMD!MUR!DBEN!DBCK!MOC,RMMR1(R0)        ;LOAD RMMR1
12120
12121  047746  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
12122  047754  005302                          DEC     R2
12123  047756  001370                          BNE     275$
12124                          ;       DROP ON CYLINDER TO RESET LATCH, AND RAISE INDEX PULSE
12125                          ;TO SET FORMAT CHANGE FLOP
12126
12127
12128  047760  012760  041005  000024          MOV     #DMD!MUR!DBEN!MI,RMMR1(R0)      ;LOAD RMMR1
12129                          ;       RAISE ON CYLINDER AND INHIBIT SEARCH TIMEOUT
12130
12131  047766  012760  051401  000024          MOV     #DMD!MUR!DBEN!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
12132                          ;       STEP COMMAND SEQUENCER TO SEARCH ENABLE (2 CLOCKS)
12133  047774  012702  000002                  MOV     #2,R2
12134  050000                  280$:
12135
12136  050000  012760  151401  000024          MOV     #DMD!MUR!DBEN!MOC!DTO!DBCK,RMMR1(R0)    ;LOAD RMMR1
12137
12138  050006  012760  051401  000024          MOV     #DMD!MUR!DBEN!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
12139  050014  005302                          DEC     R2
12140  050016  001370                          BNE     280$
12141                          ;       VERIFY THAT SEARCH ENABLE IS ON DURING SECTOR COMPARE LOOP
12142  050020  012702  000004                  MOV     #4,R2           ;R2 = CLOCK COUNT
12143  050024                  281$:
12144
12145  050024  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
12146  050032  042737  173777  001142          BIC     #^CESRC,$BDDAT
12147  050040  001411                          BEQ     282$            ;BRANCH IF SEARCH NOT ENABLED
12148
12149  050042  012760  151401  000024          MOV     #DMD!MUR!MOC!DBEN!DTO!DBCK,RMMR1(R0)    ;LOAD RMMR1
12150
12151  050050  012760  051401  000024          MOV     #DMD!MUR!MOC!DBEN!DTO,RMMR1(R0) ;LOAD RMMR1
12152  050056  005302                          DEC     R2
12153  050060  001361                          BNE     281$
12154  050062  000411                          BR      283$
12155  050064  012737  004000  001140  282$:   MOV     #ESRC,$GDDAT
12156  050072  010037  001136                  MOV     R0,$BDADR
12157  050076  062737  000024  001136          ADD     #RMMR1,$BDADR
12158  050104  104265                          ERROR   265             ;SEARCH NOT ENABLED
12159  050106                  283$:
12160                          ;       SET DRIVE FAULT TO CAUSE ABORT CONDITION
12161
12162  050106  012760  051501  000024          MOV     #DMD!MUR!DBEN!MOC!DTO!MDF,RMMR1(R0)     ;LOAD RMMR1
12163                          ;       STEP 2 CLOCKS AND VERIFY GO IS RESET
12164  050114  012702  000002                  MOV     #2,R2           ;R2 = CLOCK COUNT
12165  050120                  285$:
```

```
12166
12167  050120  012760  151501  000024          MOV     #DMD!MUR!DBEN!MOC!DTO!MDF!DBCK,RMMR1(R0)        ;LOAD RMMR1
12168
12169  050126  012760  051501  000024          MOV     #DMD!MUR!DBEN!MOC!DTO!MDF,RMMR1(R0)     ;LOAD RMMR1
12170  050134  005302                          DEC     R2
12171  050136  001370                          BNE     285$
12172
12173  050140  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
12174  050146  042737  177776  001142          BIC     #^CGO,$BDDAT
12175  050154  001406                          BEQ     290$
12176  050156  012737  000000  001140          MOV     #0,$GDDAT
12177  050164  010037  001136                  MOV     R0,$BDADR
12178  050170  104260                          ERROR   260             ;GO NOT RESET DURING SECTOR COMPARE
12179
12180  050172  012737  050204  001124  290$:   MOV     #300$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
12181  050200  012703  000001                  MOV     #1,R3           ;INITIALIZE CYLINDER ADDRESS
12182
12183                          ;;*****************************************************************
12184                          ;VERIFY THE TAG BUS DURING SEARCH
12185
12186  050204                  300$:
12187                          ;SET VOLUME VALID USING SUBROUTINE
12188  050204  004737  060556          JSR     PC,SETVV        ;GO SET VOLUME VALID
12189  050210  000402                  BR      310$            ;BRANCH TO 310$ IF NO ERROR
12190  050212  104000                  ERROR                   ;RETURN HERE IF ERROR
12191  050214  000475                  BR      330$
12192  050216                  310$:
12193                          ;       ENABLE DEBUG CLOCK, LOAD CYLINDER, TRACK AND SECTOR
12194                          ;       ADDRESS  AND LOAD SEARCH COMMAND
12195
12196  050216  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
12197
12198  050224  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
12199
12200  050232  010360  000034                  MOV     R3,RMDC(R0)     ;LOAD RMDC
```

```
12201
12202  050236  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
12203
12204  050244  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
12205
12206  050252  012760  000031  000000          MOV     #SEARCH!GO,RMCS1(R0)    ;LOAD RMCS1
12207                          ;       MOV     #400$,R2        ;INITIALIZE TABLE POINTER
12208                                                          ;THE HARDWARE ECO CHANGE THE PLA OF CS
12209                                                          ;BOARD, THE FOLLOWING 6 LINES
12210                                                          ;ADDED SO THAT THE PROGRAM CAN RUN
12211                                                          ;
12212                                                          ;ON THE BOARD WITH OR WITHOUT THE ECO
12213                                                          ;CHANGE.
12214  050260  012702  000011                  MOV     #11,R2          ;CLOCK THE COMMAND SEQ
12215                                                          ;TO STEP OVER THE FIRST 11 LOCATION
12216  050264  012760  141401  000024  101$:   MOV     #141401,RMMR1(R0)       ;CLOCK THE COMM SEQ
12217  050272  012760  041401  000024          MOV     #041401,RMMR1(R0)
12218  050300  005302                          DEC     R2              ;11 CLOCK PULSES ?
12219  050302  001370                          BNE     101$            ;LOOP BACK IF NOT DONE
12220  050304  012702  050444                  MOV     #400$+20,R2     ;ADJUST THE TABLE STARTING LOCATION
12221                          ;
                               VERIFY TAG BUS ACCORDING TO TABLE AND CYLINDER IN R3
12222  050310                  315$:
12223
12224  050310  016037  000040  001142          MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
12225  050316  042737  150000  001142          BIC     #RQA!RQB!TST,$BDDAT
12226  050324  011237  001140                  MOV     (R2),$GDDAT
12227  050330  050337  001140                  BIS     R3,$GDDAT       ;OR CYLINDER ADDRESS IN
12228  050334  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;COMPARE EXPECTED AND RECEIVED
12229  050342  001407                          BEQ     320$            ;BRANCH IF TAG BUS OK
12230  050344  010037  001136                  MOV     R0,$BDADR
12231  050350  062737  000040  001136          ADD     #RMMR2,$BDADR
12232  050356  104266                          ERROR   266             ;INCORRECT TAG BUS DURING SEARCH
12233  050360  000420                          BR      340$
12234  050362                  320$:
12235                          ;       ADVANCE TO NEXT ENTRY IN TABLE-EXIT IF DONE
12236  050362  062702  000002                  ADD     #2,R2
12237  050366  005712                          TST     (R2)
12238  050370  100407                          BMI     330$            ;EXIT IF ENTRY NEGATIVE
12239                          ;       STEP THE COMMAND SEQUENCER AND REPEAT VERIFICATION
12240
12241  050372  012760  141401  000024          MOV     #DMD!DBEN!MUR!MOC!DBCK,RMMR1(R0)        ;LOAD RMMR1
12242
12243  050400  012760  041401  000024          MOV     #DMD!DBEN!MUR!MOC,RMMR1(R0)     ;LOAD RMMR1
12244  050406  000740                          BR      315$
12245  050410                  330$:
12246                          ;REPEAT TAG BUS TEST FOR EACH PRIME CYLINDER, I.E., 1,2,4,...
12247
12248  050410  006303                          ASL     R3              ;SHIFT TO NEXT CYLINDER
12249  050412  020327  000512                  CMP     R3,#512
12250  050416  101001                          BHI     340$            ;EXIT IF 512 WAS DONE
12251  050420  000671                          BR      300$            ;TEST NEXT CYLINDER
12252  050422  000424          340$:   BR      500$            ;JUMP OVER TABLE
12253
12254  050424                  400$:
12255
12256                          ;TABLE OF TAG BUS CONTROL AND BIT VALUES
```

```
12257
12258  050424  001777                          .WORD   1777            ;BUS BITS AT HIGH IMPEDANCE STATE
12259  050426  001777                          .WORD   1777
12260  050430  001777                          .WORD   1777
12261  050432  001777                          .WORD   1777
12262  050434  001777                          .WORD   1777
12263  050436  004000                          .WORD   CC      ;CONTROL BITS ENABLED, BIT 6 ON
12264  050440  004000                          .WORD   CC
12265  050442  024000                          .WORD   TAG!CC  ;TAG COMES ON
12266  050444  024000                          .WORD   TAG!CC
12267  050446  024000                          .WORD   TAG!CC
12268  050450  024000                          .WORD   TAG!CC
12269  050452  024000                          .WORD   TAG!CC
12270                                  ;       .WORD   TAG!CC
12271                                                          ;THE TABLE IS CHANGE FOR HARDWWRE CHANGE
12272  050454  177777                          .WORD   -1              ;TABLE TERMINATED HERE
12273  050456  004000                          .WORD   CC      ;TAG GOES OFF
12274  050460  004000                          .WORD   CC
12275  050462  001777                          .WORD   1777            ;CONTROL BITS DISABLED
12276  050464  001777                          .WORD   1777
12277  050466  001777                          .WORD   1777
12278  050470  001777                          .WORD   1777
12279
12280  050472  177777                          .WORD   -1              ;END OF TABLE
12281
12282  050474                          500$:                           ;END OF TEST
12283
12284                                  ;;**********************************************************************
12285                                  ;*TEST 115       SEARCH TIMEOUT TEST
12286
12287                                  ;;**********************************************************************
12288  050474  000004                  TST115: SCOPE
12289  050476  012737  000115  001226          MOV     #115,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
12290
12291  050504  000240                          NOP
12292  050506  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
12293  050514  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
12294  050522  012737  050536  001122          MOV     #T115,$LPADR  ;LOAD LOOP ON TEST ADDRESS
12295  050530  012737  050536  001124          MOV     #T115,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
12296  050536                          T115:
12297  050536  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
12298  050542  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
12299  050546  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
12300                                  ;SET VOLUME VALID USING SUBROUTINE
12301  050552  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
12302  050556  000402                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
12303  050560  104000                          ERROR                   ;RETURN HERE IF ERROR
12304  050562  000550                          BR      90$
12305  050564                          10$:
12306                                  ;ENABLE DEBUG CLOCK AND LOAD SEARCH COMMAND
12307
12308  050564  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
12309
12310  050572  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
12311
12312  050600  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
```

```
12313
12314  050606  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
12315
12316  050614  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
12317
12318  050622  012760  000031  000000          MOV     #SEARCH!GO,RMCS1(R0)    ;LOAD RMCS1
12319                          ;EXECUTE SEARCH TO TEST FOR ON LATCH RESET (17 CLOCKS)
12320  050630  012702  000021                  MOV     #17.,R2
12321  050634                  20$:
12322
12323  050634  012760  141401  000024          MOV     #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0)        ;LOAD RMMR1
12324
12325  050642  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
12326  050650  005302                          DEC     R2
12327  050652  001370                          BNE     20$
12328                          ;DROP ON CYLINDER TO RESET LATCH, RAISE ON CYLINDER
12329
12330  050654  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
12331
12332  050662  012760  041401  000024          MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
12333                          ;STEP COMMAND SEQUENCER TO SECTOR COMPARE LOOP (2 CLOCKS)
12334  050670  012702  000002                  MOV     #2,R2
12335  050674                  30$:
12336
12337  050674  012760  151401  000024          MOV     #DMD!DBEN!MUR!MOC!DBCK!DTO,RMMR1(R0)    ;LOAD RMMR1
12338
12339  050702  012760  051401  000024          MOV     #DMD!DBEN!MUR!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
12340  050710  005302                          DEC     R2
12341  050712  001370                          BNE     30$
12342                          ;THE COMMAND SEQUENCER IS NOW IN THE SECTOR COMPARE LOOP.  STEP
12343                          ;THROUGH THE LOOP AND VERIFY SEARCH IS ENABLED.
12344  050714  012702  000005                  MOV     #5,R2
12345  050720                  40$:
12346
12347  050720  012760  151401  000024          MOV     #DMD!DBEN!MUR!MOC!DBCK!DTO,RMMR1(R0)    ;LOAD RMMR1
12348
12349  050726  012760  051401  000024          MOV     #DMD!DBEN!MUR!MOC!DTO,RMMR1(R0) ;LOAD RMMR1
12350
12351  050734  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
12352  050742  042737  173777  001142          BIC     #^CESRC,$BDDAT
12353  050750  001403                          BEQ     50$             ;BRANCH IF SEARCH NOT ENABLED
12354  050752  005302                          DEC     R2
12355  050754  001361                          BNE     40$
12356  050756  000412                          BR      60$
12357  050760  012737  004000  001140  50$:    MOV     #ESRC,$GDDAT    ;SETUP ERROR MSG
12358  050766  010037  001136                  MOV     R0,$BDADR
12359  050772  062737  000024  001136          ADD     #RMMR1,$BDADR
12360  051000  104265                          ERROR   265             ;SEARCH NOT ENABLED
12361  051002  000440                          BR      90$
12362  051004                  60$:
12363                          ;DROP DTO TO ENABLE SEARCH TIMEOUT AND WAIT FOR OPI TO SET.
12364
12365  051004  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12366  051012  012737  070000  001524          MOV     #70000,WATCH    ;SET WATCHDOG TIMER VALUE
12367  051020  004777  130502                  JSR     PC,@CLOCK       ;START THE CLOCK
12368  051024                  70$:
```

```
12369
12370  051024  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
12371  051032  042737  157777  001142          BIC     #^COPI,$BDDAT
12372  051040  001017                          BNE     80$
12373  051042  005737  001524                  TST     WATCH
12374  051046  001366                          BNE     70$
12375  051050  004777  130454                  JSR     PC,@STOP        ;STOP THE CLOCK
12376  051054  012737  020000  001140          MOV     #OPI,$GDDAT     ;SETUP ERROR MSG
12377  051062  010037  001136                  MOV     R0,$BDADR
12378  051066  062737  000014  001136          ADD     #RMER1,$BDADR
12379  051074  104267                          ERROR   267             ;OPI NOT SET BY SEARCH TIMEOUT
12380  051076  000402                          BR      90$
12381  051100                          80$:
12382  051100  004777  130424                  JSR     PC,@STOP        ;STOP THE CLOCK
12383
12384  051104                          90$:                            ;END OF TEST
12385
12386                                  ;;*******************************************************************
12387                                  ;*TEST 116       DATA COMMAND TESTS (1)
12388
12389                                  ;;*******************************************************************
12390  051104  000004                  TST116: SCOPE
12391  051106  012737  000116  001226          MOV     #116,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
12392
12393  051114  000240                          NOP
12394  051116  012737  000024  001120          MOV     #20.,$ICNT      ;20 ITERATIONS
12395  051124  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
12396  051132  012737  051146  001122          MOV     #T116,$LPADR  ;LOAD LOOP ON TEST ADDRESS
12397  051140  012737  051146  001124          MOV     #T116,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
12398  051146                          T116:
12399  051146  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
12400  051152  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
12401  051156  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
12402                                  ;;*******************************************************************
12403                                  ;VERIFY DATA COMMAND SETS OPI IF DRIVE NOT READY
12404                                  ;SET VOLUME VALID USING SUBROUTINE
12405  051162  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
12406  051166  000402                          BR      10$             ;BRANCH TO 10$ IF NO ERROR
12407  051170  104000                          ERROR                   ;RETURN HERE IF ERROR
12408  051172  000471                          BR      40$
12409  051174                          10$:
12410                                  ;       ENABLE DEBUG CLOCK AND LOAD READ COMMAND
12411
12412  051174  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12413
12414  051202  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
12415
12416  051210  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
12417
12418  051216  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
12419
12420  051224  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
12421
12422  051232  012760  000071  000000          MOV     #RD!GO,RMCS1(R0)        ;LOAD RMCS1
12423                                  ;       STEP COMMAND SEQUENCER TO UNIT READY TEST (3 CLOCKS)
12424  051240  012702  000003                  MOV     #3,R2
```

```
12425  051244                                  20$:
12426
12427  051244  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
12428
12429  051252  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12430  051260  005302                          DEC     R2
12431  051262  001370                          BNE     20$
12432                                  ;       DROP UNIT READY
12433
12434  051264  012760  040401  000024          MOV     #DMD!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
12435                                  ;       STEP SEQUENCER AND VERIFY OPI SETS (2 CLOCKS)
12436  051272  012702  000002                  MOV     #2,R2
12437  051276                                  30$:
12438
12439  051276  012760  140401  000024          MOV     #DMD!MOC!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
12440
12441  051304  012760  040401  000024          MOV     #DMD!MOC!DBEN,RMMR1(R0) ;LOAD RMMR1
12442  051312  005302                          DEC     R2
12443  051314  001370                          BNE     30$
12444
12445  051316  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
12446  051324  042737  157777  001142          BIC     #^COPI,$BDDAT
12447  051332  001011                          BNE     40$
12448  051334  012737  020000  001140          MOV     #OPI,$GDDAT
12449  051342  010037  001136                  MOV     R0,$BDADR
12450  051346  062737  000014  001136          ADD     #RMER1,$BDADR
12451  051354  104270                          ERROR   270             ;OPI NOT SET DURING DATA
12452
12453  051356  012737  051364  001124  40$:    MOV     #50$,$LPERR     ;CHANGE LOOP ON ERROR TEST
12454
12455                                  ;;*****************************************************************
12456                                  ;VERIFY DATA COMMAND ABORTS AT LOCATION 129
12457
12458  051364                                  50$:
12459                                  ;SET VOLUME VALID USING SUBROUTINE
12460  051364  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
12461  051370  000402                          BR      60$             ;BRANCH TO 60$ IF NO ERROR
12462  051372  104000                          ERROR                   ;RETURN HERE IF ERROR
12463  051374  000576                          BR      150$
12464  051376                                  60$:
12465                                  ;ENABLE DEBUG CLOCK AND LOAD READ COMMAND
12466
12467  051376  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12468
12469  051404  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
12470
12471  051412  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
12472
12473  051420  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
12474
12475  051426  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
12476
12477  051434  012760  000071  000000          MOV     #RD!GO,RMCS1(R0)        ;LOAD RMCS1
12478                                  ;STEP COMMAND SEQUENCER TO ABORT TEST AT LOCATION 129 (4 CLOCKS)
12479  051442  012702  000004                  MOV     #4,R2
12480  051446                                  70$:
```

```
12481
12482  051446  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
12483
12484  051454  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12485  051462  005302                          DEC     R2
12486  051464  001370                          BNE     70$
12487                                  ;SET DEVICE FAULT TO CAUSE ABORT CONDITION
12488
12489  051466  012760  041501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
12490                                  ;STEP THE SEQUENCER THROUGH THE TEST FOR ABORT (1 CLOCK)
12491  051474  012702  000001                  MOV     #1,R2
12492  051500                          80$:
12493
12494  051500  012760  141501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF!DBCK,RMMR1(R0)    ;LOAD RMMR1
12495
12496  051506  012760  041501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
12497  051514  005302                          DEC     R2
12498  051516  001370                          BNE     80$
12499
12500                                  ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
12501                                  ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
12502  051520  012702  000020                  MOV     #16.,R2          ;MAXIMUM NUMBER OF BIT CLOCKS
12503  051524                          85$:
12504
12505  051524  012760  045501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0)    ;LOAD RMMR1
12506
12507  051532  012760  041501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
12508
12509  051540  016037  000024  001142          MOV     RMMR1(R0),$BDDAT         ;STORE RMMR1 AT $BDDAT
12510  051546  042737  157777  001142          BIC     #^CEBL,$BDDAT
12511  051554  001014                          BNE     90$              ;BRANCH IF EBL IS SET
12512
12513  051556  005302                          DEC     R2
12514  051560  001361                          BNE     85$              ;CONTINUE BIT CLOCKS IF COUNT NOT 0
12515  051562  012737  020000  001140          MOV     #EBL,$GDDAT
12516  051570  010037  001136                  MOV     R0,$BDADR
12517  051574  062737  000024  001136          ADD     #RMMR1,$BDADR
12518  051602  104271                          ERROR   271              ;EBL NOT SET AT DATA ABORT
12519  051604  000472                          BR      150$
12520  051606                          90$:
12521                                  ;STEP THE SEQUENCER THROUGH ITS TEST FOR EBL (2 CLOCKS)
12522  051606  012702  000002                  MOV     #2,R2
12523  051612                          100$:
12524
12525  051612  012760  141501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF!DBCK,RMMR1(R0)     ;LOAD RMMR1
12526
12527  051620  012760  041501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
12528  051626  005302                          DEC     R2
12529  051630  001370                          BNE     100$
12530
12531                                  ;ABORT EBL SHOULD NOW BE INACTIVE - FORCE BIT CLOCK USING THE
12532                                  ;MAINTENANCE REGISTER TO RESET EBL (16 BIT CLOCKS)
12533  051632  012702  000020                  MOV     #16.,R2          ;MAXIMUM NUMBER OF BIT CLOCKS
12534  051636                          110$:
12535
12536  051636  012760  045501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0)     ;LOAD RMMR1
```

```
12537
12538  051644  012760  041501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
12539  051652  005302                          DEC     R2
12540  051654  001370                          BNE     110$            ;ISSUE 16 BIT CLOCKS THEN TEST
12541  051656                          120$:
12542                                  ;VERIFY EBL IS NOW RESET
12543
12544  051656  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
12545  051664  042737  157777  001142          BIC     #^CEBL,$BDDAT
12546  051672  001411                          BEQ     130$            ;BRANCH IF EBL IS RESET
12547  051674  005037  001140                  CLR     $GDDAT
12548  051700  010037  001136                  MOV     R0,$BDADR
12549  051704  062737  000024  001136          ADD     #RMMR1,$BDADR
12550  051712  104273                          ERROR   273             ;EBL DIDNT RESET ON TIME
12551  051714  000426                          BR      150$
12552  051716                          130$:
12553                                  ;VERIFY GO RESETS WITHIN 4 CLOCK CYCLES
12554  051716  012702  000004                  MOV     #4,R2
12555  051722                          140$:
12556
12557  051722  012760  141501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF!DBCK,RMMR1(R0)    ;LOAD RMMR1
12558
12559  051730  012760  041501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
12560  051736  005302                          DEC     R2
12561  051740  001370                          BNE     140$
12562
12563  051742  016037  000000  001142          MOV     RMCS1(R0),$BDDAT        ;STORE RMCS1 AT $BDDAT
12564  051750  042737  177776  001142          BIC     #^CGO,$BDDAT
12565  051756  001405                          BEQ     150$
12566  051760  005037  001140                  CLR     $GDDAT
12567  051764  010037  001136                  MOV     R0,$BDADR
12568  051770  104274                          ERROR   274             ;GO NOT RESET DURING DATA ABORT
12569
12570  051772  012737  052000  001124  150$:   MOV     #200$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
12571
12572                                  ;;*****************************************************************
12573                                  ;VERIFY SEQUENCER BRANCHES TO SEEK WHEN RUN AND GO FLOP SETS
12574
12575  052000                          200$:
12576                                  ;SET VOLUME VALID USING SUBROUTINE
12577  052000  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
12578  052004  000402                          BR      210$            ;BRANCH TO 210$ IF NO ERROR
12579  052006  104000                          ERROR                   ;RETURN HERE IF ERROR
12580  052010  000512                          BR      250$
12581  052012                          210$:
12582                                  ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
12583
12584  052012  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12585
12586  052020  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
12587
12588  052026  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
12589
12590  052034  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
12591
12592  052042  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
```

```
12593
12594  052050  012760  000071  000000          MOV     #RD!GO,RMCS1(R0)        ;LOAD RMCS1
12595                          ;MOVE SEQUENCER TO TEST FOR RUN AND GO AT LOCATION 130 (5 CLOCKS)
12596  052056  012702  000005                  MOV     #5,R2
12597  052062                  220$:
12598
12599  052062  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
12600
12601  052070  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12602  052076  005302                          DEC     R2
12603  052100  001370                          BNE     220$
12604                          ;VERIFY RUN AND GO IS SET
12605
12606  052102  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
12607  052110  042737  137777  001142          BIC     #^CRG,$BDDAT
12608  052116  001012                          BNE     230$
12609  052120  012737  040000  001140          MOV     #RG,$GDDAT
12610  052126  010037  001136                  MOV     R0,$BDADR
12611  052132  062737  000024  001136          ADD     #RMMR1,$BDADR
12612  052140  104275                          ERROR   275             ;RUN AND GO NOT SET
12613  052142  000435                          BR      250$
12614  052144                  230$:
12615                          ;VERIFY THAT CYLINDER TAG COMES UP IN ONE CLOCK CYCLE
12616  052144  012702  000001                  MOV     #1,R2
12617  052150                  240$:
12618
12619  052150  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
12620
12621  052156  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12622  052164  005302                          DEC     R2
12623  052166  001370                          BNE     240$
12624
12625  052170  016037  000040  001142          MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
12626                          ;       BIC     #RQA!RQB!TAG,$BDDAT
12627                                                  ;THE FOLLOWING LINE IS CHANGED FOR HW.
12628  052176  042737  162000  001142          BIC     #RQA!RQB!TAG!CH,$BDDAT  ;HARDWARE CHANGE :
12629                                                  ;CH AND CC WILL SET AT THE SAME TIME
12630  052204  012737  004000  001140          MOV     #CC,$GDDAT
12631  052212  023737  001140  001142          CMP     $GDDAT,$BDDAT
12632  052220  001406                          BEQ     250$
12633  052222  010037  001136                  MOV     R0,$BDADR
12634  052226  062737  000040  001136          ADD     #RMMR2,$BDADR
12635  052234  104276                          ERROR   276             ;CYLINDER TAG NOT SET DURING DATA
12636
12637  052236  012737  052250  001124  250$:   MOV     #260$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
12638
12639                          ;;*****************************************************************
12640                          ;VERIFY DATA COMMAND ABORTS AT COMMAND SEQUENCER LOCATIONS 144, 145
12641  052244  012702  000144                  MOV     #144,R2         ;INITIALIZE TEST LOCATION
12642  052250                  260$:
12643                          ;SET VOLUME VALID USING SUBROUTINE
12644  052250  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
12645  052254  000402                          BR      270$            ;BRANCH TO 270$ IF NO ERROR
12646  052256  104000                          ERROR                   ;RETURN HERE IF ERROR
12647  052260  000553                          BR      320$
12648  052262                  270$:
```

```
12649                                   ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
12650
12651  052262  012760  041401  000024           MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12652
12653  052270  012760  000000  000014           MOV     #0,RMER1(R0)    ;LOAD RMER1
12654
12655  052276  012760  000000  000042           MOV     #0,RMER2(R0)    ;LOAD RMER2
12656
12657  052304  012760  000000  000006           MOV     #0,RMDA(R0)     ;LOAD RMDA
12658
12659  052312  012760  000000  000034           MOV     #0,RMDC(R0)     ;LOAD RMDC
12660
12661  052320  012760  000071  000000           MOV     #RD!GO,RMCS1(R0)        ;LOAD RMCS1
12662                                   ;WAIT FOR RUN AND GO TO SET
12663  052326  012737  000310  001524           MOV     #200.,WATCH     ;SET WATCHDOG TIMER VALUE
12664  052334  004777  127166                   JSR     PC,@CLOCK       ;START THE CLOCK
12665  052340                          280$:
12666
12667  052340  016037  000024  001142           MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
12668  052346  042737  137777  001142           BIC     #^CRG,$BDDAT
12669  052354  001017                           BNE     290$
12670  052356  005737  001524                   TST     WATCH
12671  052362  001366                           BNE     280$
12672  052364  004777  127140                   JSR     PC,@STOP        ;STOP THE CLOCK
12673  052370  012737  040000  001140           MOV     #RG,$GDDAT
12674  052376  010037  001136                   MOV     R0,$BDADR
12675  052402  062737  000024  001136           ADD     #RMMR1,$BDADR
12676  052410  104275                           ERROR   275             ;RUN AND GO NOT SET
12677  052412  000476                           BR      320$
12678  052414                          290$:
12679  052414  004777  127110                   JSR     PC,@STOP        ;STOP THE CLOCK
12680                                   ;MOVE COMMAND SEQUENCER TO ABORT TEST (LOCATION 144 OR 145)
12681  052420  012703  000006                   MOV     #6,R3           ;SETUP CLOCK COUNT
12682  052424  022702  000144                   CMP     #144,R2
12683  052430  001402                           BEQ     300$
12684  052432  012703  000007                   MOV     #7,R3
12685  052436                          300$:
12686
12687  052436  012760  141401  000024           MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
12688
12689  052444  012760  041401  000024           MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12690  052452  005303                           DEC     R3
12691  052454  001370                           BNE     300$
12692                                   ;SET DRIVE FAULT TO FORCE ABORT CONDITION
12693
12694  052456  012760  041501  000024           MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
12695                                   ;CLOCK SEQUENCER THROUGH ITS TEST FOR ABORT (1 CLOCK)
12696  052464  012703  000001                   MOV     #1,R3
12697  052470                          305$:
12698
12699  052470  012760  141501  000024           MOV     #DMD!MUR!MOC!DBEN!MDF!DBCK,RMMR1(R0)    ;LOAD RMMR1
12700
12701  052476  012760  041501  000024           MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
12702  052504  005303                           DEC     R3
12703  052506  001370                           BNE     305$            ;ISSUE 2 CLOCKS
12704
```

```
12705                                   ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
12706                                   ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
12707  052510  012702  000020                 MOV     #16.,R2         ;MAXIMUM NUMBER OF BIT CLOCKS
12708  052514                          306$:
12709
12710  052514  012760  045501  000024         MOV     #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0)    ;LOAD RMMR1
12711
12712  052522  012760  041501  000024         MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
12713
12714  052530  016037  000024  001142         MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
12715  052536  042737  157777  001142         BIC     #^CEBL,$BDDAT
12716  052544  001013                         BNE     310$            ;BRANCH IF EBL IS SET
12717
12718  052546  005302                         DEC     R2
12719  052550  001361                         BNE     306$            ;CONTINUE BIT CLOCKS IF COUNT NOT 0
12720  052552  012737  020000  001140         MOV     #EBL,$GDDAT
12721  052560  010037  001136                 MOV     R0,$BDADR
12722  052564  062737  000024  001136         ADD     #RMMR1,$BDADR
12723  052572  104271                         ERROR   271             ;SEQ DID NOT ABORT AT DATA SEEK
12724  052574  022702  000144          310$:  CMP     #144,R2
12725  052600  001003                         BNE     320$
12726  052602  012702  000145                 MOV     #145,R2
12727  052606  000620                         BR      260$
12728
12729  052610  012737  052622  001124  320$:  MOV     #330$,$LPERR            ;CHANGE LOOP ON ERROR ADDRESS
12730
12731                                   ;;*****************************************************************
12732                                   ;VERIFY HEAD TAG DURING DATA COMMAND
12733  052616  012702  000400                 MOV     #400,R2         ;INITIALIZE TRACK ADDRESS
12734  052622                          330$:
12735                                   ;SET VOLUME VALID USING SUBROUTINE
12736  052622  004737  060556                 JSR     PC,SETVV        ;GO SET VOLUME VALID
12737  052626  000402                         BR      340$            ;BRANCH TO 340$ IF NO ERROR
12738  052630  104000                         ERROR                   ;RETURN HERE IF ERROR
12739  052632  000562                         BR      400$
12740  052634                          340$:
12741                                   ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
12742
12743  052634  012760  041401  000024         MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12744
12745  052642  012760  000000  000014         MOV     #0,RMER1(R0)    ;LOAD RMER1
12746
12747  052650  012760  000000  000042         MOV     #0,RMER2(R0)    ;LOAD RMER2
12748
12749  052656  010260  000006                 MOV     R2,RMDA(R0)     ;LOAD RMDA
12750
12751  052662  012760  000000  000034         MOV     #0,RMDC(R0)     ;LOAD RMDC
12752
12753  052670  012760  000071  000000         MOV     #RD!GO,RMCS1(R0)        ;LOAD RMCS1
12754                                   ;WAIT FOR RUN AND GO TO SET
12755  052676  012737  000310  001524         MOV     #200.,WATCH     ;SET WATCHDOG TIMER VALUE
12756  052704  004777  126616                 JSR     PC,@CLOCK       ;START THE CLOCK
12757  052710                          350$:
12758
12759  052710  016037  000024  001142         MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
12760  052716  042737  137777  001142         BIC     #^CRG,$BDDAT
```

```
12761  052724  001017                          BNE     360$
12762  052726  005737  001524                  TST     WATCH
12763  052732  001366                          BNE     350$
12764  052734  004777  126570                  JSR     PC,@STOP        ;STOP THE CLOCK
12765  052740  012737  040000  001140          MOV     #RG,$GDDAT
12766  052746  010037  001136                  MOV     R0,$BDADR
12767  052752  062737  000024  001136          ADD     #RMMR1,$BDADR
12768  052760  104275                          ERROR   275             ;RUN AND GO NOT SET
12769  052762  000506                          BR      400$
12770  052764                          360$:
12771  052764  004777  126540                  JSR     PC,@STOP        ;STOP THE CLOCK
12772                          ;STEP COMMAND SEQUENCER TO HEAD SEQUENCE, LOCATION 156 (17 CLOCKS)
12773  052770  012703  000021                  MOV     #17.,R3
12774  052774                          370$:
12775
12776  052774  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
12777
12778  053002  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12779  053010  005303                          DEC     R3
12780  053012  001370                          BNE     370$
12781                          ;DROP AND RAISE ON CYLINDER TO RESET ON LATCH
12782
12783  053014  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
12784
12785  053022  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12786                          ;       MOV     #450$,R3        ;INITIALIZE TABLE POINTER
12787                                                          ;THE FOLLOWING 6 LINES
12788                                                          ;FOR HW CHANGE: THE CH AND CC
12789                                                          ;WILL SET AT THE SAME TIME
12790                                                          ;ON THE SECOND CLOCK SEQ
12791  053030  012705  000004                  MOV     #4,R5           ;STEP OVER THE FIRST 4 LOC.
12792  053034  012760  141401  000024  101$:   MOV     #141401,RMMR1(R0)
12793  053042  012760  041401  000024          MOV     #041401,RMMR1(R0)       ;CLOCK IT THROUGH
12794  053050  005305                          DEC     R5
12795  053052  001370                          BNE     101$            ;LOOP, UNTIL DONE
12796  053054  012703  053212                  MOV     #450$+10,R3
12797                          ;VERIFY TAG BUS ACCORDING TO TABLE AND TRACK ADDRESS IN R2
12798  053060                  375$:
12799
12800  053060  016037  000040  001142          MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
12801  053066  042737  150000  001142          BIC     #RQA!RQB!TST,$BDDAT
12802  053074  011337  001140                  MOV     (R3),$GDDAT
12803  053100  010204                          MOV     R2,R4           ;GENERATE EXPECTED TAG BUS
12804  053102  000304                          SWAB    R4
12805  053104  042704  177770                  BIC     #^C7,R4
12806  053110  050437  001140                  BIS     R4,$GDDAT
12807  053114  023737  001140  001142          CMP     $GDDAT,$BDDAT
12808  053122  001020                          BNE     390$
12809
12810  053124  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
12811
12812  053132  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12813                          ;ADVANCE TO NEXT TABLE ENTRY
12814  053140  062703  000002                  ADD     #2,R3
12815  053144  005713                          TST     (R3)
12816  053146  100401                          BMI     380$
```

```
12817  053150  000743                          BR      375$
12818  053152                          380$:
12819                                  ;SHIFT TO NEXT TRACK ADDRESS-EXIT LOOP IF DONE
12820  053152  006302                          ASL     R2
12821  053154  020227  002000                  CMP     R2,#2000
12822  053160  101007                          BHI     400$
12823  053162  000617                          BR      330$
12824  053164                          390$:
12825                                  ;ERROR ON TAG BUS DURING HEAD SEQUENCE
12826  053164  010037  001136                  MOV     R0,$BDADR
12827  053170  062737  000040  001136          ADD     #RMMR2,$BDADR
12828  053176  104276                          ERROR   276             ;INCORRECT TAG BUS
12829
12830  053200  000440                  400$:   BR      500$            ;JUMP OVER TABLE
12831
12832  053202                          450$:
12833                                  ;TABLE OF TAG BUS DURING HEAD SEQUENCE
12834  053202  002000                          .WORD   CH
12835  053204  002000                          .WORD   CH
12836  053206  022000                          .WORD   CH!TAG
12837  053210  022000                          .WORD   CH!TAG
12838  053212  022000                          .WORD   CH!TAG
12839  053214  022000                          .WORD   CH!TAG
12840  053216  022000                          .WORD   CH!TAG
12841  053220  022000                          .WORD   CH!TAG
12842                                  ;       .WORD   CH
12843  053222  177777                          .WORD   -1      ;TABLE TERMINATED HERE
12844  053224  002000                          .WORD   CH
12845  053226  001777                          .WORD   1777
12846  053230  001777                          .WORD   1777
12847  053232  001777                          .WORD   1777
12848  053234  001777                          .WORD   1777
12849  053236  001777                          .WORD   1777
12850  053240  001777                          .WORD   1777
12851  053242  001777                          .WORD   1777
12852  053244  001777                          .WORD   1777
12853  053246  001777                          .WORD   1777
12854  053250  001777                          .WORD   1777
12855  053252  001777                          .WORD   1777
12856  053254  001777                          .WORD   1777
12857  053256  001777                          .WORD   1777
12858  053260  001777                          .WORD   1777
12859  053262  001777                          .WORD   1777
12860  053264  001777                          .WORD   1777
12861  053266  001777                          .WORD   1777
12862  053270  001777                          .WORD   1777
12863  053272  001777                          .WORD   1777
12864  053274  001777                          .WORD   1777
12865  053276  001777                          .WORD   1777
12866
12867  053300  177777                          .WORD   -1              ;END OF TABLE
12868
12869  053302                          500$:                           ;END OF TEST
12870
12871                                  ;;*******************************************************************
12872                                  ;*TEST 117       DATA COMMAND TESTS (2)
```

```
12873
12874                                   ;;*****************************************************************
12875  053302  000004                   TST117: SCOPE
12876  053304  012737  000117  001226           MOV     #117,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
12877
12878  053312  000240                           NOP
12879  053314  012737  000024  001120           MOV     #20.,$ICNT      ;20 ITERATIONS
12880  053322  112737  000001  001131           MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
12881  053330  012737  053344  001122           MOV     #T117,$LPADR  ;LOAD LOOP ON TEST ADDRESS
12882  053336  012737  053344  001124           MOV     #T117,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
12883  053344                           T117:
12884  053344  012706  001100                   MOV     #STACK,SP       ;LOAD THE STACK POINTER
12885  053350  013700  001276                   MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
12886  053354  013701  001456                   MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
12887                                   ;;*****************************************************************
12888                                   ;VERIFY OPI SETS IF ON CYLINDER LATCH DOESNT RESET
12889                                   ;SET VOLUME VALID USING SUBROUTINE
12890  053360  004737  060556                   JSR     PC,SETVV        ;GO SET VOLUME VALID
12891  053364  000402                           BR      10$             ;BRANCH TO 10$ IF NO ERROR
12892  053366  104000                           ERROR                   ;RETURN HERE IF ERROR
12893  053370  000514                           BR      60$
12894  053372                           10$:
12895                                   ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND DURING CYLINDER SEQUENCE
12896
12897  053372  012760  041401  000024           MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12898
12899  053400  012760  000000  000014           MOV     #0,RMER1(R0)    ;LOAD RMER1
12900
12901  053406  012760  000000  000042           MOV     #0,RMER2(R0)    ;LOAD RMER2
12902
12903  053414  012760  000000  000006           MOV     #0,RMDA(R0)     ;LOAD RMDA
12904
12905  053422  012760  000000  000034           MOV     #0,RMDC(R0)     ;LOAD RMDC
12906
12907  053430  012760  000071  000000           MOV     #RD!GO,RMCS1(R0)        ;LOAD RMCS1
12908                                   ;WAIT FOR RUN AND GO TO SET
12909  053436  012737  000310  001524           MOV     #200.,WATCH     ;SET WATCHDOG TIMER VALUE
12910  053444  004777  126056                   JSR     PC,@CLOCK       ;START THE CLOCK
12911  053450                           20$:
12912
12913  053450  016037  000024  001142           MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
12914  053456  042737  137777  001142           BIC     #^CRG,$BDDAT
12915  053464  001017                           BNE     30$
12916  053466  005737  001524                   TST     WATCH
12917  053472  001366                           BNE     20$
12918  053474  004777  126030                   JSR     PC,@STOP        ;STOP THE CLOCK
12919  053500  012737  040000  001140           MOV     #RG,$GDDAT
12920  053506  010037  001136                   MOV     R0,$BDADR
12921  053512  062737  000024  001136           ADD     #RMMR1,$BDADR
12922  053520  104275                           ERROR   275             ;RUN AND GO DIDNT SET
12923  053522  000437                           BR      60$
12924  053524                           30$:
12925  053524  004777  126000                   JSR     PC,@STOP        ;STOP THE CLOCK
12926                                   ;STEP COMMAND SEQUENCER AND VERIFY OPI SETS (19 CLOCKS)
12927  053530  012702  000023                   MOV     #19.,R2
12928  053534                           40$:
```

```
12929
12930  053534  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
12931
12932  053542  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12933  053550  005302                          DEC     R2
12934  053552  001370                          BNE     40$
12935
12936  053554  016037  000014  001142          MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
12937  053562  042737  157777  001142          BIC     #^COPI,$BDDAT
12938  053570  001011                          BNE     50$                     ;BRANCH IF OPI SET
12939  053572  012737  020000  001140          MOV     #OPE,$GDDAT
12940  053600  010037  001136                  MOV     R0,$BDADR
12941  053604  062737  000014  001136          ADD     #RMER1,$BDADR
12942  053612  104277                          ERROR   277                     ;OPI NOT SET BY ON LATCH SET
12943  053614  012737  053622  001124  50$:    MOV     #60$,$LPERR     ;CHANGE LOOP ON ERROR ADDRESS
12944                                  ;;**************************************************************
12945                                  ;VERIFY DATA COMMAND ABORTS DURING SEEK WAIT LOOP
12946  053622                          60$:
12947                                  ;SET VOLUME VALID USING SUBROUTINE
12948  053622  004737  060556                  JSR     PC,SETVV                ;GO SET VOLUME VALID
12949  053626  000402                          BR      70$                     ;BRANCH TO 70$ IF NO ERROR
12950  053630  104000                          ERROR                           ;RETURN HERE IF ERROR
12951  053632  000567                          BR      140$
12952  053634                          70$:
12953                                  ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
12954
12955  053634  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12956
12957  053642  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
12958
12959  053650  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
12960
12961  053656  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
12962
12963  053664  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
12964
12965  053672  012760  000071  000000          MOV     #RD!GO,RMCS1(R0)        ;LOAD RMCS1
12966                                  ;WAIT FOR RUN & GO TO SET
12967  053700  012737  000310  001524          MOV     #200.,WATCH     ;SET WATCHDOG TIMER VALUE
12968  053706  004777  125614                  JSR     PC,@CLOCK       ;START THE CLOCK
12969  053712                          80$:
12970
12971  053712  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
12972  053720  042737  137777  001142          BIC     #^CRG,$BDDAT
12973  053726  001017                          BNE     90$
12974  053730  005737  001524                  TST     WATCH
12975  053734  001366                          BNE     80$
12976  053736  004777  125566                  JSR     PC,@STOP        ;STOP THE CLOCK
12977  053742  012737  040000  001140          MOV     #RG,$GDDAT
12978  053750  010037  001136                  MOV     R0,$BDADR
12979  053754  062737  000024  001136          ADD     #RMMR1,$BDADR
12980  053762  104275                          ERROR   275                     ;RUN AND GO DIDNT SET
12981  053764  000512                          BR      140$
12982  053766                          90$:
12983  053766  004777  125536                  JSR     PC,@STOP        ;STOP THE CLOCK
12984                                  ;STEP COMMAND SEQUENCER TO ON LATCH TEST AT LOCATION 156 (17 CLOCKS)
```

```
12985  053772  012702  000021          MOV     #17.,R2
12986  053776                  100$:
12987
12988  053776  012760  141401  000024  MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
12989
12990  054004  012760  041401  000024  MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
12991  054012  005302          DEC     R2
12992  054014  001370          BNE     100$
12993                  ;DROP ON CYLINDER TO RESET LATCH
12994
12995  054016  012760  041001  000024  MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
12996                  ;MOVE COMMAND SEQUENCER TO SEEK WATI LOOP (31 CLOCKS)
12997  054024  012702  000037          MOV     #31.,R2
12998  054030                  110$:
12999
13000  054030  012760  141001  000024  MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
13001
13002  054036  012760  041001  000024  MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
13003  054044  005302          DEC     R2
13004  054046  001370          BNE     110$
13005                  ;STEP THROUGH SEEK WAIT LOOP (6 CLOCKS) 2 TIMES
13006  054050  012702  000006          MOV     #6.,R2
13007  054054                  120$:
13008
13009  054054  012760  141001  000024  MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
13010
13011  054062  012760  041001  000024  MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
13012  054070  005302          DEC     R2
13013  054072  001370          BNE     120$
13014                  ;SET SEEK INCOMPLETE ERROR TO CAUSE ABORT
13015
13016  054074  012760  041201  000024  MOV     #DMD!MUR!DBEN!MSER,RMMR1(R0)     ;LOAD RMMR1
13017                  ;CLOCK THE SEQUENCER THROUGH ITS TEST FOR ABORT (2 CLOCKS)
13018  054102  012702  000002          MOV     #2,R2
13019  054106                  130$:
13020
13021  054106  012760  141201  000024  MOV     #DMD!MUR!DBEN!MSER!DBCK,RMMR1(R0)        ;LOAD RMMR1
13022
13023  054114  012760  041201  000024  MOV     #DMD!MUR!DBEN!MSER,RMMR1(R0)     ;LOAD RMMR1
13024  054122  005302          DEC     R2
13025  054124  001370          BNE     130$
13026
13027                  ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
13028                  ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
13029  054126  012702  000020          MOV     #16.,R2          ;MAXIMUM NUMBER OF BIT CLOCKS
13030  054132                  135$:
13031
13032  054132  012760  045501  000024  MOV     #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0)     ;LOAD RMMR1
13033
13034  054140  012760  041501  000024  MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
13035
13036  054146  016037  000024  001142  MOV     RMMR1(R0),$BDDAT         ;STORE RMMR1 AT $BDDAT
13037  054154  042737  157777  001142  BIC     #^CEBL,$BDDAT
13038  054162  001013          BNE     140$
13039
13040  054164  005302          DEC     R2
```

```
13041  054166  001361                          BNE     135$            ;CONTINUE BIT CLOCKS IF COUNT NOT 0
13042  054170  012737  020000  001140          MOV     #EBL,$GDDAT
13043  054176  010037  001136                  MOV     R0,$BDADR
13044  054202  062737  000024  001136          ADD     #RMMR1,$BDADR
13045  054210  104300                          ERROR   300             ;EBL NOT SET DUE TO ABORT
13046  054212  012737  054220  001124  140$:   MOV     #150$,$LPERR            ;CHANGE LOOP ON ERROR ADDRESS
13047
13048                                  ;;*****************************************************************
13049                                  ;VERIFY DATA COMMAND ABORTS DURING OFFSET IF ON CYLINDER LATCH
13050                                  ;DOESNT RESET
13051  054220                          150$:
13052                                  ;SET VOLUME VALID USING SUBROUTINE
13053  054220  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
13054  054224  000402                          BR      160$            ;BRANCH TO 160$ IF NO ERROR
13055  054226  104000                          ERROR                   ;RETURN HERE IF ERROR
13056  054230  000536                          BR      220$
13057  054232                          160$:
13058                                  ;LOAD TRACK,SETCOR,CYLINDER ADDRESS
13059
13060  054232  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
13061
13062  054240  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
13063                                  ;SET OFFSET MODE USING SUBROUTINE
13064  054246  004737  060706                  JSR     PC,SETOM        ;GO SET OFFSET MODE
13065  054252  000402                          BR      170$            ;BRANCH TO 170$ IF NO ERROR
13066  054254  104000                          ERROR                   ;RETURN HERE IF ERROR
13067  054256  000523                          BR      220$
13068  054260                          170$:
13069                                  ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
13070
13071  054260  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13072
13073  054266  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
13074
13075  054274  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
13076
13077  054302  012760  000071  000000          MOV     #RD!GO,RMCS1(R0)        ;LOAD RMCS1
13078                                  ;WAIT FOR RUN AND GO TO SET
13079  054310  012737  000310  001524          MOV     #200.,WATCH     ;SET WATCHDOG TIMER VALUE
13080  054316  004777  125204                  JSR     PC,@CLOCK       ;START THE CLOCK
13081  054322                          180$:
13082
13083  054322  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
13084  054330  042737  137777  001142          BIC     #^CRG,$BDDAT
13085  054336  001017                          BNE     190$
13086  054340  005737  001524                  TST     WATCH
13087  054344  001366                          BNE     180$
13088  054346  004777  125156                  JSR     PC,@STOP        ;STOP THE CLOCK
13089  054352  012737  040000  001140          MOV     #RG,$GDDAT
13090  054360  010037  001140                  MOV     R0,$GDDAT
13091  054364  062737  000024  001136          ADD     #RMMR1,$BDADR
13092  054372  104275                          ERROR   275             ;RUN AND GO NOT SET
13093  054374  000454                          BR      220$
13094  054376                          190$:
13095  054376  004777  125126                  JSR     PC,@STOP        ;STOP THE CLOCK
13096                                  ;STEP COMMAND SEQUENCER TO ON LATCH TEST AT LOCATION 156 (17 CLOCKS)
```

```
13097  054402  012702  000021          MOV     #17.,R2
13098  054406                  200$:
13099
13100  054406  012760  141401  000024  MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
13101
13102  054414  012760  041401  000024  MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13103  054422  005302          DEC     R2
13104  054424  001370          BNE     200$
13105                  ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST
13106                  ;AT LOCATION 166
13107
13108  054426  012760  041001  000024  MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
13109
13110  054434  012760  041401  000024  MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
13111                  ;MOVE SEQUENCER TO SET OPI AND EBL (39 CLOCKS)
13112  054442  012702  000047          MOV     #39.,R2
13113  054446                  210$:
13114
13115  054446  012760  141401  000024  MOV     #DMD!MUR!DBEN!MOC!DBCK,RMMR1(R0)        ;LOAD RMMR1
13116
13117  054454  012760  041401  000024  MOV     #DMD!MUR!DBEN!MOC,RMMR1(R0)     ;LOAD RMMR1
13118  054462  005302          DEC     R2
13119  054464  001370          BNE     210$
13120                  ;VERIFY OPI IS SET
13121
13122  054466  016037  000014  001142  MOV     RMER1(R0),$BDDAT        ;STORE RMER1 AT $BDDAT
13123  054474  042737  157777  001142  BIC     #^COPI,$BDDAT
13124  054502  001011          BNE     220$
13125  054504  012737  020000  001140  MOV     #OPI,$GDDAT
13126  054512  010037  001136          MOV     R0,$BDADR
13127  054516  062737  000014  001136  ADD     #RMER1,$BDADR
13128  054524  104277          ERROR   277             ;OPI NOT SET DURING OFFSET
13129  054526  012737  054534  001124  220$:  MOV     #230$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
13130
13131                  ;;*****************************************************************
13132                  ;VERIFY DATA COMMAND ABORTS DURING OFFSET WAIT LOOP
13133
13134  054534                  230$:
13135                  ;SET VOLUME VALID USING SUBROUTINE
13136  054534  004737  060556          JSR     PC,SETVV        ;GO SET VOLUME VALID
13137  054540  000403          BR      240$            ;BRANCH TO 240$ IF NO ERROR
13138  054542  104000          ERROR                   ;RETURN HERE IF ERROR
13139  054544  000137  055154          JMP     310$
13140  054550                  240$:
13141                  ;LOAD SECTOR,TRACK AND CYLINDER ADDRESS
13142
13143  054550  012760  000000  000006  MOV     #0,RMDA(R0)     ;LOAD RMDA
13144
13145  054556  012760  000000  000034  MOV     #0,RMDC(R0)     ;LOAD RMDC
13146                  ;SET OFFSET MODE USING SUBROUTINE
13147  054564  004737  060706          JSR     PC,SETOM        ;GO SET OFFSET MODE
13148  054570  000402          BR      245$            ;BRANCH TO 245$ IF NO ERROR
13149  054572  104000          ERROR                   ;RETURN HERE IF ERROR
13150  054574  000567          BR      310$
13151                  ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
13152  054576                  245$:
```

```
13153
13154  054576  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13155
13156  054604  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
13157
13158  054612  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
13159
13160  054620  012760  000071  000000          MOV     #RD!GO,RMCS1(R0)        ;LOAD RMCS1
13161                          ;WAIT FOR RUN AND GO TO SET
13162  054626  012737  000310  001524          MOV     #200.,WATCH     ;SET WATCHDOG TIMER VALUE
13163  054634  004777  124666                  JSR     PC,@CLOCK       ;START THE CLOCK
13164  054640                  250$:
13165
13166  054640  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
13167  054646  042737  137777  001142          BIC     #^CRG,$BDDAT
13168  054654  001017                          BNE     260$
13169  054656  005737  001524                  TST     WATCH
13170  054662  001366                          BNE     250$
13171  054664  004777  124640                  JSR     PC,@STOP        ;STOP THE CLOCK
13172  054670  012737  040000  001140          MOV     #RG,$GDDAT
13173  054676  010037  001136                  MOV     R0,$BDADR
13174  054702  062737  000024  001136          ADD     #RMMR1,$BDADR
13175  054710  104275                          ERROR   275             ;RUN AND GO NOT SET
13176  054712  000520                          BR      310$
13177  054714                  260$:
13178  054714  004777  124610                  JSR     PC,@STOP        ;STOP THE CLOCK
13179                          ;STEP SEQUENCER TO LOCATION 156 (17 CLOCKS)
13180  054720  012702  000021                  MOV     #17.,R2
13181  054724                  270$:
13182
13183  054724  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
13184
13185  054732  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13186  054740  005302                          DEC     R2
13187  054742  001370                          BNE     270$
13188                          ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST
13189                          ;AT LOCATION 166
13190
13191  054744  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
13192
13193  054752  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13194                          ;MOVE SEQUENCER TO LOCATION  174 (37 CLOCKS)
13195  054760  012702  000045                  MOV     #37.,R2
13196  054764                  280$:
13197
13198  054764  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
13199
13200  054772  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13201  055000  005302                          DEC     R2
13202  055002  001370                          BNE     280$
13203                          ;DROP ON CYLINDER TO RESET LATCH, LEAVE ON CYLINDER RESET
13204
13205  055004  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
13206                          ;STEP SEQUENCER THROUGH OFFSET WAIT LOOP TWICE (7 CLOCKS)
13207  055012  012702  000007                  MOV     #7.,R2
13208  055016                  290$:
```

```
13209
13210  055016  012760  141001  000024          MOV     #DMD!MUR!DBEN!DBCK,RMMR1(R0)    ;LOAD RMMR1
13211
13212  055024  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
13213  055032  005302                          DEC     R2
13214  055034  001370                          BNE     290$
13215                                ;SET DRIVE FAULT TO CAUSE ABORT CONDITION
13216
13217  055036  012760  041101  000024          MOV     #DMD!MUR!DBEN!MDF,RMMR1(R0)     ;LOAD RMMR1
13218                                ;STEP SEQUENCER THROUGH ITS TEST FOR ABORT (2 CLOCKS)
13219  055044  012702  000002                   MOV     #2,R2
13220  055050                       300$:
13221
13222  055050  012760  141101  000024          MOV     #DMD!MUR!DBEN!MDF!DBCK,RMMR1(R0)        ;LOAD RMMR1
13223
13224  055056  012760  041101  000024          MOV     #DMD!MUR!DBEN!MDF,RMMR1(R0)     ;LOAD RMMR1
13225  055064  005302                          DEC     R2
13226  055066  001370                          BNE     300$
13227
13228                                ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
13229                                ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
13230  055070  012702  000020                   MOV     #16.,R2         ;MAXIMUM NUMBER OF BIT CLOCKS
13231  055074                       305$:
13232
13233  055074  012760  045501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0)    ;LOAD RMMR1
13234
13235  055102  012760  041501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
13236
13237  055110  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
13238  055116  042737  157777  001142          BIC     #^CEBL,$BDDAT
13239  055124  001013                          BNE     310$
13240
13241  055126  005302                          DEC     R2
13242  055130  001361                          BNE     305$            ;CONTINUE BIT CLOCKS IF COUNT NOT 0
13243  055132  012737  020000  001140          MOV     #EBL,$GDDAT
13244  055140  010037  001136                  MOV     R0,$BDADR
13245  055144  062737  000024  001136          ADD     #RMMR1,$BDADR
13246  055152  104271                          ERROR   271             ;EBL NOT SET DUE TO OFFSET WAIT ABORT
13247
13248  055154  012737  055162  001124  310$:   MOV     #320$,$LPERR    ;CHANGE LOOP ON ERROR ADDRESS
13249
13250                                ;;*****************************************************************
13251                                ;VERIFY THAT DATA COMMAND ABORTS DURING SECTOR WAIT LOOP AT LOCATION 179
13252
13253  055162                       320$:
13254                                ;SET VOLUME VALID USING SUBROUTINE
13255  055162  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
13256  055166  000403                          BR      330$            ;BRANCH TO 330$ IF NO ERROR
13257  055170  104000                          ERROR                   ;RETURN HERE IF ERROR
13258  055172  000137  055620                  JMP     420$
13259  055176                       330$:
13260                                ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
13261
13262  055176  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13263
13264  055204  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
```

```
13265
13266  055212  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
13267
13268  055220  012760  000000  000006          MOV     #0,RMDA(R0)     ;LOAD RMDA
13269
13270  055226  012760  000000  000034          MOV     #0,RMDC(R0)     ;LOAD RMDC
13271
13272  055234  012760  000071  000000          MOV     #RD!GO,RMCS1(R0)        ;LOAD RMCS1
13273                          ;WAIT FOR RUN AND GO TO SET
13274  055242  012737  000310  001524          MOV     #200.,WATCH     ;SET WATCHDOG TIMER VALUE
13275  055250  004777  124252                  JSR     PC,@CLOCK       ;START THE CLOCK
13276  055254                          340$:
13277
13278  055254  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
13279  055262  042737  137777  001142          BIC     #^CRG,$BDDAT
13280  055270  001017                          BNE     350$
13281  055272  005737  001524                  TST     WATCH
13282  055276  001366                          BNE     340$
13283  055300  004777  124224                  JSR     PC,@STOP        ;STOP THE CLOCK
13284  055304  012737  040000  001140          MOV     #RG,$GDDAT
13285  055312  010037  001136                  MOV     R0,$BDADR
13286  055316  062737  000024  001136          ADD     #RMMR1,$BDADR
13287  055324  104275                          ERROR   275             ;RUN AND GO NOT SET
13288  055326  000534                          BR      420$
13289  055330                          350$:
13290  055330  004777  124174                  JSR     PC,@STOP        ;STOP THE CLOCK
13291                          ;STEP SEQUENCER TO LOCATION 156 (17 CLOCKS)
13292  055334  012702  000021                  MOV     #17.,R2
13293  055340                          360$:
13294
13295  055340  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
13296
13297  055346  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13298  055354  005302                          DEC     R2
13299  055356  001370                          BNE     360$
13300                          ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST
13301                          ;AT LOCATION 166
13302
13303  055360  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
13304                          ;MOVE SEQUENCER TO SECTOR WAIT (34 CLOCKS)
13305  055366  012702  000042                  MOV     #34.,R2
13306  055372                          370$:
13307
13308  055372  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
13309
13310  055400  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13311  055406  005302                          DEC     R2
13312  055410  001370                          BNE     370$
13313                          ;STEP THROUGH SECTOR WAIT LOOP TWICE AND VERIFY SEARCH IS ENABLED
13314                          ;DURING THE LOOP (6 CLOCKS)
13315  055412  012702  000006                  MOV     #6.,R2
13316  055416                          380$:
13317
13318  055416  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
13319
13320  055424  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
```

```
13321
13322  055432  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
13323  055440  042737  173777  001142          BIC     #^CESRC,$BDDAT
13324  055446  001403                          BEQ     390$
13325  055450  005302                          DEC     R2
13326  055452  001361                          BNE     380$
13327  055454  000412                          BR      400$
13328  055456  012737  004000  001140  390$:   MOV     #ESRC,$GDDAT
13329  055464  010037  001136                  MOV     R0,$BDADR
13330  055470  062737  000024  001136          ADD     #RMMR1,$BDADR
13331  055476  104301                          ERROR   301             ;SEARCH NOT ENABLED DURING DATA
13332  055500  000447                          BR      420$
13333  055502                          400$:
13334                                  ;SET DRIVE FAULT TO CAUSE ABORT CONDITION
13335
13336  055502  012760  041501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
13337                                  ;STEP SEQUENCER THROUGH ITS TEST FOR ABORT (2 CLOCKS)
13338  055510  012702  000002                  MOV     #2,R2
13339  055514                          410$:
13340
13341  055514  012760  141501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF!DBCK,RMMR1(R0)    ;LOAD RMMR1
13342
13343  055522  012760  041501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
13344  055530  005302                          DEC     R2
13345  055532  001370                          BNE     410$
13346
13347                                  ;ABORT EBL SHOULD NOW BE ACTIVE - USE THE MAINTENANCE REGISTER TO
13348                                  ;FORCE BIT CLOCKS AND VERIFY THAT EBL SETS WITHIN 16 BIT CLOCKS
13349  055534  012702  000020                  MOV     #16.,R2         ;MAXIMUM NUMBER OF BIT CLOCKS
13350  055540                          415$:
13351
13352  055540  012760  045501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF!MCLK,RMMR1(R0)    ;LOAD RMMR1
13353
13354  055546  012760  041501  000024          MOV     #DMD!MUR!MOC!DBEN!MDF,RMMR1(R0) ;LOAD RMMR1
13355                                  ;VERIFY EBL IS SET
13356
13357  055554  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
13358  055562  042737  157777  001142          BIC     #^CEBL,$BDDAT
13359  055570  001013                          BNE     420$
13360
13361  055572  005302                          DEC     R2
13362  055574  001361                          BNE     415$            ;CONTINUE BIT CLOCKS IF COUNT NOT 0
13363  055576  012737  020000  001140          MOV     #EBL,$GDDAT
13364  055604  010037  001136                  MOV     R0,$BDADR
13365  055610  062737  000024  001136          ADD     #RMMR1,$BDADR
13366  055616  104271                          ERROR   271             ;NO ABORT AT SECTOR WAIT
13367
13368  055620                          420$:                           ;END OF TEST
13369
13370                                  ;;*****************************************************************
13371                                  ;*TEST 120       DATA COMMAND TESTS(3)
13372
13373                                  ;;*****************************************************************
13374  055620  000004                  TST120: SCOPE
13375  055622  012737  000120  001226          MOV     #120,$TESTN     ;;SET TEST NUMBER IN APT MAIL BOX
13376
```

```
13377  055630  000240                          NOP
13378  055632  012737  000024  001120          MOV     #20.,$:CNT      ;20 ITERATIONS
13379  055640  112737  000001  001131          MOVB    #1,$ERMAX       ;ONE ERROR ALLOWED
13380  055646  012737  055662  001122          MOV     #T120,$LPADR  ;LOAD LOOP ON TEST ADDRESS
13381  055654  012737  055662  001124          MOV     #T120,$LPERR  ;LOAD LOOP ON ERROR ADDRESS
13382  055662                          T120:
13383  055662  012706  001100                  MOV     #STACK,SP       ;LOAD THE STACK POINTER
13384  055666  013700  001276                  MOV     $BASE,R0        ;R0 = UNIBUS ADDRESS OF UUT
13385  055672  013701  001456                  MOV     TSTQUE,R1       ;R1 = POINTER TO DEVICE
13386                                  ;;*****************************************************************
13387                                  ;VERIFY THE TAG BUS DURING DATA COMMAND
13388
13389                                  ;;*****************************************************************
13390                                  ;FIRST PART USES OFFSET FORWARD
13391
13392                                  ;LOAD TEST PARAMETERS IN REGISTER OUTPUT BUFFER
13393  055676  112737  000035  001410          MOVB    #29.,RMDA0      ;SECTOR 29
13394  055704  112737  000004  001411          MOVB    #4,RMDA0+1      ;TRACK 4
13395  055712  012737  001466  001436          MOV     #822.,RMDC0     ;CYLINDER 822
13396  055720  012737  000200  001434          MOV     #OFD,RMOF0      ;FORWARD OFFSET
13397  055726  012737  000071  001402          MOV     #RD!GO,RMCS10   ;READ DATA
13398  055734  012737  177400  001404          MOV     #^C256.+1,RMWC0 ;WORD COUNT
13399  055742  012737  106724  001406          MOV     #BUFFER,RMBA0   ;BUFFER ADDRESS
13400
13401                                  ;EXECUTE COMMAND AND VERIFY TAG BUS USING SUBROUTINE
13402  055750  004737  055772                  JSR     PC,10$
13403
13404                                  ;;*****************************************************************
13405                                  ;SECOND PART USES OFFSET REVERSE
13406                                  ;LOAD TEST PARAMETERS IN REGISTER OUTPUT BUFFER
13407  055754  112737  000000  001434          MOVB    #0,RMOF0        ;REVERSE OFFSET
13408
13409                                  ;EXECUTE COMMAND AND VERIFY TAG BUS USING SUBROUTINE
13410  055762  004737  055772                  JSR     PC,10$
13411
13412  055766  000137  057100                  JMP     300$
13413
13414                                  ;;*****************************************************************
13415                                  ;SUBROUTINE USED DURING TEST
13416
13417  055772                          10$:
13418                                  ;SET VOLUME VALID USING SUBROUTINE
13419  055772  004737  060556                  JSR     PC,SETVV        ;GO SET VOLUME VALID
13420  055776  000403                          BR      20$             ;BRANCH TO 20$ IF NO ERROR
13421  056000  104000                          ERROR                   ;RETURN HERE IF ERROR
13422  056002  000137  056716                  JMP     160$
13423  056006                          20$:
13424                                  ;LOAD TRACK, SECTOR AND CYLINDER ADDRESS, LOAD OFFSET
13425
13426  056006  013760  001410  000006          MOV     RMDA0,RMDA(R0)  ;LOAD RMDA
13427
13428  056014  013760  001436  000034          MOV     RMDC0,RMDC(R0)  ;LOAD RMDC
13429
13430  056022  013760  001434  000032          MOV     RMOF0,RMOF(R0)  ;LOAD RMOF
13431                                  ;SET OFFSET MODE USING SUBROUTINE
13432  056030  004737  060706                  JSR     PC,SETOM        ;GO SET OFFSET MODE
```

```
13433  056034  000403                          BR      30$             ;BRANCH TO 30$ IF NO ERROR
13434  056036  104000                          ERROR                   ;RETURN HERE IF ERROR
13435  056040  000137  056716                  JMP     160$
13436  056044                          30$:
13437                                  ;LOAD BUFFER ADDRESS AND WORD COUNT
13438
13439  056044  013760  001404  000002          MOV     RMWC0,RMWC(R0)  ;LOAD RMWC
13440
13441  056052  013760  001406  000004          MOV     RMBA0,RMBA(R0)  ;LOAD RMBA
13442                                  ;ENABLE DEBUG CLOCK AND LOAD DATA COMMAND
13443
13444  056060  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13445
13446  056066  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
13447
13448  056074  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
13449
13450  056102  013760  001402  000000          MOV     RMCS10,RMCS1(R0)        ;LOAD RMCS1
13451                                  ;WAIT FOR RUN AND GO TO SET
13452  056110  012737  000310  001524          MOV     #200.,WATCH     ;SET WATCHDOG TIMER VALUE
13453  056116  004777  123404                  JSR     PC,@CLOCK       ;START THE CLOCK
13454  056122                          40$:
13455
13456  056122  016037  000024  001142          MOV     RMMR1(R0),$BDDAT        ;STORE RMMR1 AT $BDDAT
13457  056130  042737  137777  001142          BIC     #^CRG,$BDDAT
13458  056136  001020                          BNE     50$
13459  056140  005737  001524                  TST     WATCH
13460  056144  001366                          BNE     40$
13461  056146  004777  123356                  JSR     PC,@STOP        ;STOP THE CLOCK
13462  056152  012737  040000  001140          MOV     #RG,$GDDAT
13463  056160  010037  001136                  MOV     R0,$BDADR
13464  056164  062737  000024  001136          ADD     #RMMR1,$BDADR
13465  056172  104275                          ERROR   275             ;RUN AND GO NOT SET
13466  056174  000137  056716                  JMP     160$
13467  056200                          50$:
13468  056200  004777  123324                  JSR     PC,@STOP        ;STOP THE CLOCK
13469  056204  012704  056720                  MOV     #200$,R4        ;R4 = TABLE POINTER
13470
13471                                  ;STEP SEQUENCER TO HEAD SEQUENCE AT LOCATION 156 (17 CLOCKS)
13472  056210  012705  000021                  MOV     #17.,R5         ;R5 = CLOCK COUNT
13473  056214                          60$:
13474
13475  056214  016037  000040  001142          MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
13476  056222  016037  000040  001142          MOV     RMMR2(R0),$BDDAT        ;IF CH AND CC SET AT THE SAME TIME
13477                                                                          ;THE NEW HW ECO IS IMPLEMENTED.
13478  056230  013737  001142  001174          MOV     $BDDAT,$TMP0            ;CHECK IF CC AND CH SET
13479  056236  042737  171777  001174          BIC     #171777,$TMP0           ;CHOP OFF THE REST BITS
13480  056244  022737  006000  001174          CMP     #6000,$TMP0             ;CH AND CC BOTH SET THEN
13481  056252  001414                          BEQ     101$                    ;BRANCH TO NEW LOCATION
13482                                                                          ;
13483                                                                                  ;
13484  056254  042737  150000  001142          BIC     #RQA!RQB!TST,$BDDAT
13485  056262  012437  001140                  MOV     (R4)+,$GDDAT
13486  056266  053737  001436  001140          BIS     RMDC0,$GDDAT    ;OR CYLINDER ADDRESS
13487  056274  023737  001140  001142          CMP     $GDDAT,$BDDAT
13488  056302  001011                          BNE     70$             ;BRANCH IF TAG BUS WRONG
```

```
13489  056304                                  101$:
13490
13491  056304  012760  141401  000024                  MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
13492
13493  056312  012760  041401  000024                  MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13494  056320  005305                                  DEC     R5
13495  056322  001334                                  BNE     60$
13496  056324  000407                                  BR      80$
13497  056326  010037  001136                70$:     MOV     R0,$BDADR
13498  056332  062737  000040  001136                ADD     #RMMR2,$BDADR
13499  056340  104276                                  ERROR   276             ;INCORRECT TAG BUS DURING DATA
13500  056342  000565                                  BR      160$
13501  056344                                  80$:
13502                                          ;DROP ON CYLINDER TO RESET LATCH, SET ON CYLINDER TO PASS TEST AT
13503                                          ;LOCATION 166
13504
13505  056344  012760  041001  000024                  MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
13506
13507  056352  012760  041401  000024                  MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13508                                          ;STEP SEQUENCER TO END OF OFFSET AT LOCATION 174 (37 CLOCKS)
13509  056360  012705  000045                  MOV     #37.,R5         ;RELOAD CLOCK COUNT
13510  056364                                  90$:
13511
13512  056364  016037  000040  001142                  MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
13513  056372  016037  000040  001142                  MOV     RMMR2(R0),$BDDAT        ;IF CH AND CC SET AT THE SAME TIME
13514                                                                                  ;THE NEW HW ECO IS IMPLEMENTED
13515  056400  013737  001142  001174                  MOV     $BDDAT,$TMP0            ;TEMP STORAGE
13516  056406  042737  171777  001174                  BIC     #171777,$TMP0           ;CHECK CH AND CC SET ?
13517  056414  022737  006000  001174                  CMP     #6000,$TMP0
13518  056422  001003                                  BNE     102$                    ;IF IT IS AN OLD HW
13519  056424  162704  000002                          SUB     #2,R4                   ;ADJUST THE TABLE ADDRESS
13520  056430  000444                                  BR      107$                    ;TO OTHER LOC
13521  056432                                  102$:
13522  056432  042737  150000  001142                  BIC     #RQA!RQB!TST,$BDDAT
13523  056440  011437  001140                          MOV     (R4),$GDDAT
13524  056444  032714  002000                          BIT     #CH,(R4)
13525  056450  001430                                  BEQ     110$            ;BRANCH IF CONTROL/HEADER NOT ON
13526  056452  032714  004000                          BIT     #CC,(R4)
13527  056456  001416                                  BEQ     100$            ;BRANCH IF HEADER TAG
13528                                          ;CONTROL TAG SHOULD BE ON-SETUP EXPECTED OFFSET
13529  056460  052737  000010  001140                  BIS     #BB03,$GDDAT    ;ASSUME OFD IS NOT SET
13530  056466  032737  000200  001434                  BIT     #OFD,RMOFO
13531  056474  001406                                  BEQ     95$
13532  056476  042737  000010  001140                  BIC     #BB03,$GDDAT    ;RESET BUS BIT 3 - DIRECTION IS REV
13533  056504  052737  000004  001140                  BIS     #BB02,$GDDAT
13534  056512  000407                          95$:    BR      110$
13535
13536  056514                                  100$:
13537                                          ;HEADER TAG SHOULD BE ON - SETUP EXPECTED TRACK ADDRESS
13538  056514  013703  001410                          MOV     RMDAO,R3        ;GET TRACK
13539  056520  000303                                  SWAB    R3
13540  056522  042703  177770                          BIC     #^C7,R3
13541  056526  050337  001140                          BIS     R3,$GDDAT
13542  056532                                  110$:
13543                                          ;COMPARE EXPECTED AND RECEIVED TAG BUS DATA
13544  056532  023737  001140  001142                  CMP     $GDDAT,$BDDAT
```

```
13545  056540  001013                          BNE     120$
13546  056542                          107$:
13547
13548  056542  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
13549
13550  056550  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13551  056556  062704  000002                  ADD     #2,R4           ;MOVE TABLE POINTER
13552  056562  005305                          DEC     R5              ;DECREMENT CLOCK COUNT
13553  056564  001277                          BNE     90$
13554  056566  000407                          BR      130$
13555  056570  010037  001136          120$:   MOV     R0,$BDADR
13556  056574  062737  000040  001136          ADD     #RMMR2,$BDADR
13557  056602  104276                          ERROR   276             ;INCORRECT TAG BUS DURING DATA
13558  056604  000444                          BR      160$
13559  056606                          130$:
13560                                  ;DROP ON CYLINDER TO RESET LATCH, RAISE ON CYLINDER TO PASS TEST AT
13561                                  ;SEQUENCER LOCATION 175
13562
13563  056606  012760  041001  000024          MOV     #DMD!MUR!DBEN,RMMR1(R0) ;LOAD RMMR1
13564
13565  056614  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13566                                  ;STEP SEQUENCER TO SECTOR WAIT LOOP (8 CLOCKS)
13567  056622  012705  000010                  MOV     #8.,R5
13568  056626                          140$:
13569
13570  056626  016037  000040  001142          MOV     RMMR2(R0),$BDDAT        ;STORE RMMR2 AT $BDDAT
13571                                  ;       BIC     #RQA!RQB!TST,$BDDAT
13572  056634  042737  171777  001142          BIC     #171777,$BDDAT
13573
13574  056642  012737  006000  001140          MOV     #6000,$GDDAT    ;CH AND CC BOTH ARE SET ?
13575  056650  023737  001140  001142          CMP     $GDDAT,$BDDAT   ;GOOD DATAEXPECTED ?
13576  056656  001011                          BNE     150$            ;BRANCH IF ERROR
13577
13578  056660  012760  141401  000024          MOV     #DMD!MUR!MOC!DBEN!DBCK,RMMR1(R0)        ;LOAD RMMR1
13579
13580  056666  012760  041401  000024          MOV     #DMD!MUR!MOC!DBEN,RMMR1(R0)     ;LOAD RMMR1
13581  056674  005305                          DEC     R5
13582  056676  001353                          BNE     140$
13583  056700  000406                          BR      160$
13584  056702  010037  001136          150$:   MOV     R0,$BDADR
13585  056706  062737  000040  001136          ADD     #RMMR2,$BDADR
13586  056714  104276                          ERROR   276             ;INCORRECT TAG BUS DURING DATA
13587
13588  056716  000207                  160$:   RTS     PC
13589
13590  056720                          200$:
13591                                  ;TABLE OF TAG BUS CONTROL AND DATA VALUES
13592  056720  001777                          .WORD   1777            ;LOCATION 0
13593  056722  001777                          .WORD   1777            ;LOCATION 25
13594  056724  001777                          .WORD   1777            ;LOCATION 26
13595  056726  001777                          .WORD   1777            ;LOCATION 128
13596  056730  001777                          .WORD   1777            ;LOCATION 129
13597  056732  001777                          .WORD   1777            ;LOCATION 130
13598  056734  004000                          .WORD   CC              ;LOCATION 144
13599  056736  004000                          .WORD   CC              ;LOCATION 145
13600  056740  024000                          .WORD   CC!TAG          ;LOCATION 146
```

```
13601  056742  024000                    .WORD   CC!TAG          ;LOCATION 147
13602  056744  024000                    .WORD   CC!TAG          ;LOCATION 148
13603  056746  024000                    .WORD   CC!TAG          ;LOCATION 149
13604  056750  024000                    .WORD   CC!TAG          ;LOCATION 150
13605  056752  024000                    .WORD   CC!TAG          ;LOCATION 151
13606  056754  004000                    .WORD   CC              ;LOCATION 152
13607  056756  004000                    .WORD   CC              ;LOCATION 153
13608  056760  001777                    .WORD   1777            ;LOCATION 154
13609  056762  002000                    .WORD   CH              ;LOCATION 156
13610  056764  002000                    .WORD   CH              ;LOCATION 157
13611  056766  022000                    .WORD   CH!TAG          ;LOCATION 158
13612  056770  022000                    .WORD   CH!TAG          ;LOCATION 159
13613  056772  022000                    .WORD   CH!TAG          ;LOCATION 160
13614  056774  022000                    .WORD   CH!TAG          ;LOCATION 161
13615  056776  022000                    .WORD   CH!TAG          ;LOCATION 162
13616  057000  022000                    .WORD   CH!TAG          ;LOCATION 163
13617  057002  002000                    .WORD   CH              ;LOCATION 164
13618  057004  002000                    .WORD   CH              ;LOCATION 165
13619  057006  001777                    .WORD   1777            ;LOCATION 232
13620  057010  001777                    .WORD   1777            ;LOCATION 233
13621  057012  001777                    .WORD   1777            ;LOCATION 234
13622  057014  001777                    .WORD   1777            ;LOCATION 235
13623  057016  001777                    .WORD   1777            ;LOCATION 236
13624  057020  001777                    .WORD   1777            ;LOCATION 237
13625  057022  001777                    .WORD   1777            ;LOCATION 238
13626  057024  001777                    .WORD   1777            ;LOCATION 239
13627  057026  001777                    .WORD   1777            ;LOCATION 240
13628  057030  001777                    .WORD   1777            ;LOCATION 241
13629  057032  001777                    .WORD   1777            ;LOCATION 242
13630  057034  001777                    .WORD   1777            ;LOCATION 243
13631  057036  001777                    .WORD   1777            ;LOCATION 244
13632  057040  001777                    .WORD   1777            ;LOCATION 245
13633  057042  001777                    .WORD   1777            ;LOCATION 246
13634  057044  001777                    .WORD   1777            ;LOCATION 247
13635  057046  001777                    .WORD   1777            ;LOCATION 248
13636  057050  001777                    .WORD   1777            ;LOCATION 249
13637  057052  001777                    .WORD   1777            ;LOCATION 250
13638  057054  001777                    .WORD   1777            ;LOCATION 251
13639  057056  001777                    .WORD   1777            ;LOCATION 252
13640  057060  001777                    .WORD   1777            ;LOCATION 166
13641  057062  006000                    .WORD   CC!CH           ;LOCATION 169
13642  057064  006000                    .WORD   CC!CH           ;LOCATION 170
13643  057066  026000                    .WORD   CC!CH!TAG       ;LOCATION 171
13644  057070  026000                    .WORD   CC!CH!TAG       ;LOCATION 172
13645  057072  026000                    .WORD   CC!CH!TAG       ;LOCATION 1[illegible]
13646  057074  026000                    .WORD   CC!CH!TAG       ;LOCATION 174
13647  057076  026000                    .WORD   CC!CH!TAG       ;LOCATION 175,ETC
13648
13649  057100                  300$:                             ;END OF TEST
```

```
13650  057100                          $EOSP:
13651  057100  000240                          NOP
13652  057102  013700  001456                  MOV     TSTQUE,R0
13653  057106  062700  000002                  ADD     #2,R0
13654  057112  010037  001456                  MOV     R0,TSTQUE
13655  057116  005710                          TST     (R0)
13656  057120  001402                          BEQ     1$
13657  057122  000137  006326                  JMP     READY
13658  057126  012737  001460  001456  1$:     MOV     #TSTQUE+2,TSTQUE
13659                          .SBTTL  END OF PASS ROUTINE
13660
13661                          ;;**********************************************************************
13662                          ;*INCREMENT THE PASS NUMBER ($PASS)
13663                          ;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY''
13664                          ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
13665                          ;*IF THERES A MONITOR GO TO IT
13666                          ;*IF THERE ISN'T JUMP TO READY
13667
13668  057134                          $EOP:
13669  057134  000240                          NOP
13670  057136  005037  001116                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
13671  057142  005037  001206                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
13672  057146  005237  001230                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
13673  057152  042737  100000  001230          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
13674  057160  005327                          DEC     (PC)+           ;;LOOP?
13675  057162  000001                  $EOPCT: .WORD   1
13676  057164  003063                          BGT     $DOAGN          ;;YES
13677  057166  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
13678  057170  000001                  $ENDCT: .WORD   1
13679  057172  057162                          $EOPCT
13680  057174  104401  057202                  TYPE    ,65$            ;;TYPE ASCIZ STRING
13681  057200  000407                          BR      64$             ;;GET OVER THE ASCIZ
13682                          ;;65$:  .ASCIZ  <12><15>/END PASS #/
13683  057220                  64$:
13684  057220  013746  001230                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
13685                                                                  ;;TYPE PASS NUMBER
13686  057224  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
13687  057226  104401  057234                  TYPE    ,67$            ;;TYPE ASCIZ STRING
13688  057232  000421                          BR      66$             ;;GET OVER THE ASCIZ
13689                          ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
13690  057276                  66$:
13691  057276  013746  001126                  MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
13692                                                                  ;;TOTAL NUMBER OF ERRORS
13693  057302  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
13694  057304  104401  001217                  TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
13695  057310  005037  001126                  CLR     $ERTTL          ;;CLEAR ERROR TOTAL
13696  057314  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
13697  057320  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
13698  057322  000005                          RESET                   ;;CLEAR THE WORLD
13699  057324  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
13700  057326  000240                          NOP                     ;;SAVE ROOM
13701  057330  000240                          NOP                     ;;FOR
13702  057332  000240                          NOP                     ;;ACT11
13703  057334                          $DOAGN:
13704  057334  000137                          JMP     @(PC)+          ;;RETURN
13705  057336  006326                  $RTNAD: .WORD   READY
```

```
13706  057340      377      377      000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
13707           057344                            .EVEN
```

```
13708                                   .SBTTL  SUBROUTINES
13709
13710                                   ;;*****************************************************************
13711                                   .SBTTL  ERROR TYPEOUT ROUTINE
13712
13713                                   ;*THE ERROR TYPEOUT ROUTINE ASSEMBLES AND PRINTS INFORMATION
13714                                   ;*REGARDING THE DETECTION OF AN ERROR AS FOLLOWS:
13715                                   ;*
13716                                   ;*      .UNIT NUMBER, TEST NUMBER, ERROR NUMBER AND PROGRAM COUNTER ARE
13717                                   ;*PRINTED ON THE FIRST LINE;
13718                                   ;*      .ERROR MESSAGE IS ASSEMBLED, FORMATTED AND PRINTED ON
13719                                   ;*ONE OR MORE SUCCEEDING LINES;
13720                                   ;*      .PAIRED LINES OF ERROR HEADERS AND ERROR DATA
13721                                   ;*ARE PRINTED AFTER THE ERROR MESSAGE.
13722
13723  057344                           ERRTYP:
13724  057344  104414                           SAVREG
13725  057346  032777  020000  121600           BIT     #SW13,@SWR      ;INHIBIT TYPEOUTS??
13726  057354  001402                           BEQ     1$              ;NO!!
13727  057356  000137  060074                   JMP     21$             ;YES!!
13728                                   ;TYPE UNIT NUMBER, TEST NUMBER, ERROR NUMBER, AND PROGRAM COUNTER
13729  057362  104401  001217           1$:     TYPE    ,$CRLF
13730  057366  104401  060110                   TYPE    ,ERTY00         ;TYPE 'UNT#'
13731  057372  013746  001234                   MOV     $UNIT,-(SP)     ;;SAVE $UNIT FOR TYPEOUT
13732                                                                   ;;TYPE UNIT NUMBER
13733  057376  104403                           TYPOS                   ;;GO TYPE--OCTAL ASCII
13734  057400     003                           .BYTE   3               ;;TYPE 3 DIGIT(S)
13735  057401     000                           .BYTE   0               ;;SUPPRESS LEADING ZEROS
13736  057402  005037  060100                   CLR     TSTNMB          ;LOAD TEST NUMBER FOR
13737  057406  013737  001226  060100           MOV     $TESTN,TSTNMB
13738  057414  104401  060116                   TYPE    ,ERTY01         ;TYPE 'TST#'
13739  057420  013746  060100                   MOV     TSTNMB,-(SP)    ;;SAVE TSTNMB FOR TYPEOUT
13740                                                                   ;;TYPE TEST NUMBER
13741  057424  104403                           TYPOS                   ;;GO TYPE--OCTAL ASCII
13742  057426     003                           .BYTE   3               ;;TYPE 3 DIGIT(S)
13743  057427     000                           .BYTE   0               ;;SUPPRESS LEADING ZEROS
13744  057430  005037  060102                   CLR     ERRNMB          ;LOAD ERROR NUMBER FOR
13745  057434  113737  001130  060102           MOVB    $ITEMB,ERRNMB   ;TYPEOUT
13746  057442  001406                           BEQ     2$              ;SKIP IF NO ERROR CALLED
13747  057444  104401  060126                   TYPE    ,ERTY02         ;TYPE 'ERR#'
13748  057450  013746  060102                   MOV     ERRNMB,-(SP)    ;;SAVE ERRNMB FOR TYPEOUT
13749                                                                   ;;TYPE ERROR NUMBER
13750  057454  104403                           TYPOS                   ;;GO TYPE--OCTAL ASCII
13751  057456     003                           .BYTE   3               ;;TYPE 3 DIGIT(S)
13752  057457     000                           .BYTE   0               ;;SUPPRESS LEADING ZEROS
13753  057460  104401  060135           2$:     TYPE    ,ERTY03         ;TYPE 'PC='
13754  057464  013746  001132                   MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
13755                                                                   ;;TYPE PROGRAM COUNTER
13756  057470  104403                           TYPOS                   ;;GO TYPE--OCTAL ASCII
13757  057472     006                           .BYTE   6               ;;TYPE 6 DIGIT(S)
13758  057473     001                           .BYTE   1               ;;TYPE LEADING ZEROS
13759                                   ;GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
13760  057474  005737  060102           3$:     TST     ERRNMB          ;WAS AN ERROR CALLED?
13761  057500  001575                           BEQ     21$             ;NO!!
13762  057502  104401  001217                   TYPE    ,$CRLF          ;YES-TYPE CRLF
13763  057506  105037  060106                   CLRB    BOTFLG          ;CLEAR BOT FLAG
```

```
13764  057512  105037  060107                  CLRB    CHRCNT          ;CLEAR CHARACTER COUNTER
13765  057516  013700  060102                  MOV     ERRNMB,R0       ;R0 POINTS TO FIRST OF
13766  057522  006300                          ASL     R0              ;FOUR ENTRIES IN ERROR
13767  057524  006300                          ASL     R0              ;TABLE
13768  057526  006300                          ASL     R0
13769  057530  062700  001522                  ADD     #$ERRTB-8.,R0
13770  057534  011001                          MOV     (R0),R1         ;R1 POINTS TO ERROR MESSAGE
13771                                                                  ;TABLE
13772  057536  001507                          BEQ     12$             ;BRANCH IF NO ERROR MESSAGE
13773                                  ;TYPE THE ERROR MESSAGE
13774  057540  012102                  4$:     MOV     (R1)+,R2        ;R2=ADDRESS OF MESSAGE STRING
13775  057542  001505                          BEQ     12$             ;BRANCH IF END OF MESSAGE
13776  057544  010237  057712                  MOV     R2,11$          ;LOAD ADDRESS OF STRING
13777  057550  005037  060104                  CLR     BOTADR          ;CLEAR BOT ADDRESS
13778  057554  112203                  5$:     MOVB    (R2)+,R3        ;END OF STRING??
13779  057556  001454                          BEQ     10$             ;YES!!
13780  057560  122703  000015                  CMPB    #CR,R3          ;CARRIAGE RETURN??
13781  057564  001003                          BNE     6$              ;NO!!
13782  057566  105037  060107                  CLRB    CHRCNT          ;YES-CLEAR CHAR COUNT
13783  057572  000770                          BR      5$              ;GET NEXT CHARACTER
13784  057574  122703  000012          6$:     CMPB    #LF,R3          ;LINE FEED??
13785  057600  001765                          BEQ     5$              ;YES-GET NEXT CHARACTER
13786  057602  122703  000011                  CMPB    #HT,R3          ;HORIZONTAL TAB??
13787  057606  001007                          BNE     8$              ;NO!!
13788  057610  105237  060107          7$:     INCB    CHRCNT          ;ADJUST CHARACTER COUNT
13789  057614  132737  000007  060107          BITB    #7,CHRCNT
13790  057622  001372                          BNE     7$
13791  057624  000407                          BR      9$
13792  057626  105237  060107          8$:     INCB    CHRCNT          ;INCREMENT CHARACTER COUNT
13793  057632  122703  000040                  CMPB    #' ,R3          ;SPACE??
13794  057636  001002                          BNE     9$              ;NO!!
13795  057640  010237  060104                  MOV     R2,BOTADR       ;SAVE ADDRESS OF SPACE
13796  057644  122737  000100  060107  9$:     CMPB    #64.,CHRCNT     ;END OF LINE??
13797  057652  103340                          BHIS    5$              ;NO!!
13798  057654  013704  060104                  MOV     BOTADR,R4       ;GET ADDRESS OF LAST SPACE
13799  057660  001007                          BNE     90$             ;BRANCH IF SPACE DETECTED
13800  057662  104401  001217                  TYPE    ,$CRLF          ;TYPE CRLF
13801  057666  105037  060107                  CLRB    CHRCNT          ;CLEAR CHARACTER COUNT
13802  057672  013702  057712                  MOV     11$,R2          ;SET UP R2 FOR TESTING
13803  057676  000726                          BR      5$
13804  057700  105044                  90$:    CLRB    -(R4)           ;REPLACE SPACE
13805  057702  112737  177777  060106          MOVB    #-1,BOTFLG      ;SET BOT FLAG
13806  057710  104401                  10$:    TYPE                    ;TYPE ERROR MESSAGE STRING
13807  057712  000000                  11$:    .WORD                   ;STRING ADDRESS GOES HERE
13808  057714  105737  060106                  TSTB    BOTFLG          ;WAS STRING TRUNCATED??
13809  057720  001707                          BEQ     4$              ;NO!!
13810  057722  104401  001217                  TYPE    ,$CRLF          ;YES-TYPE CRLF
13811  057726  105037  060106                  CLRB    BOTFLG          ;CLEAR BOT FLAG
13812  057732  105037  060107                  CLRB    CHRCNT          ;CLEAR CHARACTER COUNT
13813  057736  013702  060104                  MOV     BOTADR,R2       ;SETUP R2 FOR TESTING
13814  057742  010237  057712                  MOV     R2,11$          ;SETUP 11$ FOR TYPING
13815  057746  112742  000040                  MOVB    #' ,-(R2)       ;RESTORE SPACE
13816  057752  105722                          TSTB    (R2)+           ;RESTORE R2
13817  057754  000677                          BR      5$              ;TYPE REST OF STRING
13818  057756                          12$:
13819                                  ;TYPE ERROR HEADER AND ERROR DATA
```

```
13820  057756                          13$:
13821  057756  016001  000002                  MOV     2(R0),R1        ;R1 POINTS TO ERROR HEADER TABLE
13822  057762  001444                          BEQ     21$             ;BRANCH IF NO HEADER
13823  057764  104401  001217                  TYPE    ,$CRLF          ;(ASSUME NO DATA)
13824  057770  016002  000004                  MOV     4(R0),R2        ;R2 POINTS TO DATA ADDRESS TABLE
13825  057774  016003  000006                  MOV     6(R0),R3        ;R3 POINTS TO FORMAT TABLE
13826  060000  012137  060010          14$:    MOV     (R1)+,15$       ;PUT HEADER ADDRESS FOR TYPE
13827  060004  001433                          BEQ     21$             ;BRANCH IF END OF HEADERS
13828                                                                  ;(ASSUME END OF DATA)
13829  060006  104401                          TYPE
13830  060010  000000          15$:    .WORD   0               ;HEADER ADDRESS GOES HERE
13831  060012  104401  001217                  TYPE    ,$CRLF
13832  060016  005702                          TST     R2              ;DATA WITH HEADER??
13833  060020  001767                          BEQ     14$             ;NO!!
13834  060022  012204                          MOV     (R2)+,R4        ;R4 POINTS TO DATA ADDRESS
13835  060024  012305                          MOV     (R3)+,R5        ;R5 POINTS TO FORMAT
13836  060026  105725          16$:    TSTB    (R5)+           ;WHAT KIND OF DATA??
13837  060030  100407                          BMI     18$             ;BINARY
13838  060032  001403                          BEQ     17$             ;OCTAL
13839  060034  013446                          MOV     @(R4)+,-(SP)    ;DECIMAL
13840  060036  104405                          TYPDS
13841  060040  000405                          BR      19$
13842  060042  013446          17$:    MOV     @(R4)+,-(SP)
13843  060044  104402                          TYPOC
13844  060046  000402                          BR      19$
13845  060050  013446          18$:    MOV     @(R4)+,-(SP)
13846  060052  104406                          TYPBN
13847  060054  005714          19$:    TST     (R4)            ;MORE DATA??
13848  060056  001403                          BEQ     20$             ;NO!!
13849  060060  104401  060143                  TYPE    ,ERTY04         ;YES-TYPE 2 SPACES
13850  060064  000760                          BR      16$             ;AND CONTINUE
13851  060066  104401  001217          20$:    TYPE    ,$CRLF          ;TYPE ONE BLANK LINE
13852  060072  000742                          BR      14$             ;BEFORE NEXT HEADER
13853  060074  104415          21$:    RESREG
13854  060076  000207                          RTS     PC
13855
13856  060100  000000          TSTNMB: .WORD   0               ;TEST NUMBER
13857  060102  000000          ERRNMB: .WORD   0               ;ERROR NUMBER
13858  060104  000000          BOTADR: .WORD   0               ;BEGINNING OF TEXT ADDRESS
13859  060106     000          BOTFLG: .BYTE   0               ;BOT FLAG
13860  060107     000          CHRCNT: .BYTE   0               ;CHARACTER COUNT
13861
13862  060110  047125  052111  000043  ERTY00: .ASCIZ  @UNIT#@
13863  060116  020054  042524  052123  ERTY01: .ASCIZ  @, TEST#@
13864  060124  000043
13865  060126  020054  051105  021522  ERTY02: .ASCIZ  @, ERR#@
13866  060134     000
13867  060135     054  050040  036503  ERTY03: .ASCIZ  @, PC=@
13868  060142     000
13869  060143     040  000040          ERTY04: .ASCIZ  @ @
13870                                  .EVEN
13871
```

```
13872                                   .SBTTL  CLOCK SUBROUTINES
13873
13874                                   ;ROUTINE TO SIZE FOR CLOCKS (KW11-L OR KW11-P)
13875                                   ;;*****************************************************************
13876  060146  000240                  SIZCLK: NOP
13877  060150  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
13878  060154  013746  000006                  MOV     ERRVEC+2,-(SP)          ;;PUSH ERRVEC+2 ON STACK
13879  060160  012737  060244  000004          MOV     #10$,ERRVEC     ;LOAD 04 TRAP VECTORS
13880  060166  012737  000300  000006          MOV     #PR6,ERRVEC+2
13881
13882                                   ;SEE IF A KW11-P CLOCK IS PRESENT - GO TO 10$ IF NOT PRESENT
13883  060174  005777  121302                  TST     @$LPCSR         ;TEST FOR P CLOCK
13884  060200  012737  060406  001526          MOV     #PCLOCK,CLOCK   ;LOAD SUBROUTINE ADDRESS
13885  060206  012737  060530  001530          MOV     #PSTOP,STOP     ;LOAD STOP ADDRESS
13886  060214  012777  060474  121264          MOV     #PCOUNT,@$LPVEC ;LOAD P CLOCK INTERRUPT VECTOR
13887  060222  012777  000300  121260          MOV     #PR6,@$LPVEC+2
13888  060230  013777  001516  121256          MOV     $LLVEC+2,@$LLVEC;CLEAR L CLOCK INTERRUPT VECTOR
13889  060236  005077  121254                  CLR     @$LLVEC+2
13890  060242  000454                  BR      30$
13891  060244  012716  060252          10$:    MOV     #15$,(SP)       ;DUMMY RTI ADDRESS
13892  060250  000002                  RTI                             ;RESTORE PRIORITY
13893  060252                          15$:
13894                                   ;NO P CLOCK-SEE IF L CLOCK IS PRESENT-GO TO 20$ IF NOT PRESET
13895  060252  012737  060330  000004          MOV     #20$,ERRVEC     ;CHANGE 04 TRAP VECTOR
13896  060260  005777  121226                  TST     @$LLCSR         ;TEST FOR L CLOCK
13897  060264  012737  060424  001526          MOV     #LCLOCK,CLOCK   ;LOAD SUBROUTINE ADDRESS
13898  060272  012737  060536  001530          MOV     #LSTOP,STOP     ;LOAD STOP ADDRESS
13899  060300  012777  060474  121206          MOV     #LCOUNT,@$LLVEC ;LOAD L CLOCK INTERRUPT VECTOR
13900  060306  012777  000300  121202          MOV     #PR6,@$LLVEC+2
13901  060314  013777  001510  121164          MOV     $LPVEC+2,@$LPVEC;CLEAR P CLOCK INTERRUPT VECTOR
13902  060322  005077  121162                  CLR     @$LPVEC+2
13903  060326  000422                  BR      30$
13904  060330  012716  060336          20$:    MOV     #25$,(SP)       ;DUMMY RTI ADDRESS
13905  060334  000002                  RTI                             ;RESTORE PRIORITY
13906  060336                          25$:
13907                                   ;NO CLOCK AVAILABLE - AUGMENT RETURN ADDRESS
13908  060336  005037  001526                  CLR     CLOCK           ;CLEAR SUBROUTINE ADDRESS
13909  060342  012737  001510  001506          MOV     #$LPVEC+2,$LPVEC;CLEAR P CLOCK INTERRUPT VECTOR
13910  060350  005037  001510                  CLR     $LPVEC+2
13911  060354  012737  001516  001514          MOV     #$LLVEC+2,$LLVEC;CLEAR L CLOCK INTERRUPT VECTOR
13912  060362  005037  001516                  CLR     $LLVEC+2
13913  060366  062766  000002  000004          ADD     #2,4(SP)        ;CHANGE RETURN ADDRESS
13914
13915  060374                          30$:
13916  060374  012637  000006                  MOV     (SP)+,ERRVEC+2          ;;POP STACK INTO ERRVEC+2
13917  060400  012637  000004                  MOV     (SP)+,ERRVEC    ;;POP STACK INTO ERRVEC
13918  060404  000207                  RTS     PC
13919                                   ;ROUTINES TO START THE CLOCK (KW11-L OR KW11-P)
13920                                   ;;*****************************************************************
13921
13922  060406  012777  177777  121070  PCLOCK: MOV     #-1,@$LPCSB     ;LOAD COUNT SET BUFFER
13923  060414  012777  000135  121060          MOV     #135,@$LPCSR    ;LOAD CONTROL REGISTER
13924  060422  000403                  BR      PLCLK           ;GO TO COMMON CODE
13925
13926  060424  012777  000100  121060  LCLOCK: MOV     #100,@$LLCSR    ;LOAD CONTROL REGISTER
13927
```

```
13928  060432  005037  001522          PLCLK:  CLR     TIME            ;CLEAR TIMER COUNT
13929  060436  104400                          TRAP                    ;;PUSH OLD PSW AND PC ON STACK
13930  060440  012605                          MOV     (SP)+,R5        ;;SAVE THE PSW IN R5
13931  060442  010537  001520                  MOV     R5,$PSW         ;SAVE PRIORITY
13932  060446  042705  177437                  BIC     #^CPR7,R5       ;MASK X
13933  060452  022705  000300                  CMP     #PR6,R5         ;IS PRIORITY TOO HIGH??
13934  060456  101005                          BHI     40$             ;NO!!
13935  060460  012746  000240                  MOV     #PR5,-(SP)      ;;PUT NEW PS ON STACK
13936  060464  012746  060472                  MOV     #30$,-(SP)      ;;PUT NEW PC ON  STACK
13937  060470  000002                          RTI                     ;;POP NEW PC AND PS
13938  060472                          30$:
13939  060472  000207                  40$:    RTS     PC
13940
13941                                  ;ROUTINES TO HANDLE CLOCK INTERRUPTS (KW11-L OR KW11-P)
13942
13943  060474                          PCOUNT:
13944  060474                          LCOUNT:
13945  060474  062737  000021  001522          ADD     #17.,TIME       ;ADD 17MS TO ELAPSED TIME
13946  060502  103003                          BCC     10$             ;BRANCH IF NO OVERFLOW
13947  060504  012737  177777  001522          MOV     #-1,TIME        ;RESTORE MAXIMUM COUNT
13948  060512  162737  000021  001524  10$:    SUB     #17.,WATCH      ;DECREMENT REMAINING TIME
13949  060520  100002                          BPL     20$             ;BRANCH IF POSITIVE
13950  060522  005037  001524                  CLR     WATCH           ;CLEAR REMAINING TIME
13951  060526  000002                  20$:    RTI                     ;RETURN TO USER
13952
13953                                  ;ROUTINES TO STOP THE CLOCK (KW11-L OR KW11-P)
13954
13955  060530  005077  120746          PSTOP:  CLR     @$LPCSR         ;STOP P CLOCK
13956  060534  000402                          BR      PLSTP           ;GO TO COMMON STOP CODE
13957
13958  060536  005077  120750          LSTOP:  CLR     @$LLCSR         ;STOP L CLOCK
13959
13960  060542                          PLSTP:
13961  060542  013746  001520                  MOV     $PSW,-(SP)      ;;PUT NEW PS ON STACK
13962  060546  012746  060554                  MOV     #10$,-(SP)      ;;PUT NEW PC ON  STACK
13963  060552  000002                          RTI                     ;;POP NEW PC AND PS
13964  060554                          10$:
13965  060554  000207                          RTS     PC
13966
13967                                  .SBTTL  SET VOLUME VAL!D SUBROUTINE
13968
13969                                  ;THIS SUBROUTINE INITIALIZES THE SUBSYSTEM AND SETS VOLUME VALID,
13970                                  ;RETURNING WITH THE DRIVE STILL IN DIAGNOSTIC MODE.  THE SUBROUTINE
13971                                  ;RETURNS TO THE WORD FOLLOWING THE CALL, EXCEPT WHEN AN ERROR IS
13972                                  ;DETECTED, IN WHICH CASE IT RETURNS TO THE SECOND WORD FOLLOWING THE
13973                                  ;CALL.
13974                                  ;
13975                                  ;CALL:  JSR     PC,SETVV        JUMP TO SUBROUTINE
13976                                  ;       BR      ??              RETURN HERE IF NO ERROR
13977                                  ;       ERROR                   RETURN HERE IF ERROR
13978
13979  060556                          SETVV:
13980  060556  012760  000040  000010          MOV     #CLR,RMCS2(R0)  ;CLEAR THE MASSBUS
13981  060564  111160  000010                  MOVB    (R1),RMCS2(R0)  ;SELECT UNIT
13982
13983  060570  012760  000001  000024          MOV     #DMD,RMMR1(R0)  ;LOAD RMMR1
```

```
13984
13985  060576  012760  001001  000024          MOV     #DMD!MUR,RMMR1(R0)      ;LOAD RMMR1
13986
13987  060604  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
13988
13989  060612  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
13990
13991  060620  012760  000023  000000          MOV     #PACACK!GO,RMCS1(R0)    ;LOAD RMCS1
13992
13993  060626  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
13994  060634  042737  177677  001142          BIC     #^CVV,$BDDAT
13995  060642  001020                          BNE     10$             ;BRANCH IF VOLUME VALID SET
13996  060644  010037  001136                  MOV     R0,$BDADR       ;SETUP FOR ERROR MSG
13997  060650  062737  000012  001136          ADD     #RMDS,$BDADR
13998  060656  012737  000100  001140          MOV     #VV,$GDDAT
13999  060664  062716  000002                  ADD     #2,(SP)         ;MOVE RETURN ADDRESS TO ERROR
14000  060670  112776  000170  000000          MOVB    #170,@(SP)      ;WRITE ERROR NUMBER
14001  060676  012737  000022  001174          MOV     #PACACK,$TMP0
14002  060704  000207                  10$:    RTS     PC              ;RETURN
14003
14004                                  .SBTTL  SET OFFSET MODE SUBROUTINE
14005
14006                                  ;THIS SUBROUTINE EXECUTES AN OFFSET COMMAND AND VERIFIES THAT OFFSET
14007                                  ;MODE SETS.  THE DRIVE SHOULD BE IN DIAGNOSTIC MODE WHEN CALLING THE
14008                                  ;SUBROUTINE, WHICH WILL LEAVE DMD ON.  THE SUBROUTINE RETURNS TO THE
14009                                  ;WORD FOLLOWING THE CALL UNLESS THERE IS AN ERROR, IN WHICH CASE IT
14010                                  ;RETURNS TO THE SECOND WORD FOLLOWING THE CALL
14011                                  ;
14012                                  ;CALL:  JSR     PC,SETOM        JUMP TO SUBROUTINE
14013                                  ;       BR      ??              RETURN HERE IF NO ERROR
14014                                  ;       ERROR                   RETURN HERE IF ERROR
14015
14016  060706                  SETOM:
14017
14018  060706  012760  001001  000024          MOV     #DMD!MUR,RMMR1(R0)      ;LOAD RMMR1
14019
14020  060714  012760  000000  000014          MOV     #0,RMER1(R0)    ;LOAD RMER1
14021
14022  060722  012760  000000  000042          MOV     #0,RMER2(R0)    ;LOAD RMER2
14023
14024  060730  012760  000015  000000          MOV     #OFFSET!GO,RMCS1(R0)    ;LOAD RMCS1
14025
14026  060736  016037  000012  001142          MOV     RMDS(R0),$BDDAT ;STORE RMDS AT $BDDAT
14027  060744  042737  177776  001142          BIC     #^COM,$BDDAT
14028  060752  001015                          BNE     10$             ;BRANCH IF OFFSET ON
14029  060754  012737  000001  001140          MOV     #OM,$GDDAT
14030  060762  010037  001136                  MOV     R0,$BDADR
14031  060766  062737  000012  001136          ADD     #RMDS,$BDADR
14032  060774  062716  000002                  ADD     #2,(SP)         ;MOVE RETURN ADDRESS TO ERROR
14033  061000  112776  000200  000000          MOVB    #200,@(SP)      ;WRITE ERROR NUMBER
14034  061006                  10$:
14035  061006  000207                  RTS     PC              ;RETURN TO USER
14036
```

```
14037                                   .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
14038
14039                                   ;;**********************************************************************
14040                                   ;*SAVE R0-R5
14041                                   ;*CALL:
14042                                   ;*      SAVREG
14043                                   ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
14044                                   ;*
14045                                   ;*TOP---(+16)
14046                                   ;* +2---(+18)
14047                                   ;* +4---R5
14048                                   ;* +6---R4
14049                                   ;* +8---R3
14050                                   ;*+10---R2
14051                                   ;*+12---R1
14052                                   ;*+14---R0
14053
14054  061010                           $SAVREG:
14055  061010  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
14056  061012  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
14057  061014  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
14058  061016  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
14059  061020  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
14060  061022  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
14061  061024  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
14062  061030  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
14063  061034  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
14064  061040  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
14065  061044  000002                          RTI
14066
14067                                   ;*RESTORE R0-R5
14068                                   ;*CALL:
14069                                   ;*      RESREG
14070  061046                           $RESREG:
14071  061046  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
14072  061052  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
14073  061056  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
14074  061062  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
14075  061066  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
14076  061070  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
14077  061072  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
14078  061074  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
14079  061076  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
14080  061100  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
14081  061102  000002                          RTI
14082                                   .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
14083
14084                                   ;;**********************************************************************
14085                                   ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
14086                                   ;*BINARY-ASCII NUMBER AND TYPE IT.
14087                                   ;*CALL:
14088                                   ;*      MOV     NUMBER,-(SP)    ;;NUMBER TO BE TYPED
14089                                   ;*      TYPBN                   ;;TYPE IT
14090
14091  061104  010146                   $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
14092  061106  016601  000006                  MOV     6(SP),R1        ;;GET THE INPUT NUMBER
```

```
14093  061112  000261                          SEC                     ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
14094  061114  112737  000060  061156  1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII "0".
14095  061122  006101                          ROL     R1              ;;GET THIS BIT
14096  061124  001406                          BEQ     2$              ;;DONE?
14097  061126  105537  061156                  ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
14098  061132  104401  061156                  TYPE    ,$BIN           ;;GO TYPE THIS BIT
14099  061136  000241                          CLC                     ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
14100  061140  000765                          BR      1$              ;;GO DO THE NEXT BIT
14101  061142  012601                  2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
14102  061144  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
14103  061152  012616                          MOV     (SP)+,(SP)
14104  061154  000002                          RTI                     ;;RETURN TO USER
14105  061156     000     000          $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
14106                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
14107
14108                                  ;;**************************************************************************
14109                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
14110                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
14111                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
14112                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
14113                                  ;*REPLACED WITH SPACES.
14114                                  ;*CALL:
14115                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
14116                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
14117
14118  061160                          $TYPDS:
14119  061160  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
14120  061162  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
14121  061164  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
14122  061166  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
14123  061170  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
14124  061172  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
14125  061176  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
14126  061202  100004                          BPL     1$              ;;BR IF INPUT IS POS.
14127  061204  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
14128  061206  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
14129  061214  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
14130  061216  012703  061374                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
14131  061222  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
14132  061226  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
14133  061230  016001  061364                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
14134  061234  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
14135  061236  002402                          BLT     4$              ;;BR IF DONE
14136  061240  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
14137  061242  000774                          BR      3$
14138  061244  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
14139  061246  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
14140  061250  001002                          BNE     5$              ;;FALL THROUGH IF 0
14141  061252  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
14142  061254  100407                          BMI     7$              ;;BR IF YES
14143  061256  106316                  5$:     ASLB    (SP)            ;;MSD?
14144  061260  103003                          BCC     6$              ;;BR IF NO
14145  061262  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
14146  061270  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
14147  061274  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
14148  061300  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
```

```
14149  061302  005720                          TST     (R0)+           ;;JUST INCREMENTING
14150  061304  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
14151  061310  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
14152  061312  003002                          BGT     8$              ;;GO TO EXIT
14153  061314  010502                          MOV     R5,R2           ;;GET THE LSD
14154  061316  000764                          BR      6$              ;;GO CHANGE TO ASCII
14155  061320  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
14156  061322  100003                          BPL     9$              ;;BR IF NO
14157  061324  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
14158  061332  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
14159  061334  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
14160  061336  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
14161  061340  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
14162  061342  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
14163  061344  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
14164  061346  104401  061374                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
14165  061352  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
14166  061360  012616                          MOV     (SP)+,(SP)
14167  061362  000002                          RTI                     ;;RETURN TO USER
14168  061364  023420                  $DTBL:  10000.
14169  061366  001750                          1000.
14170  061370  000144                          100.
14171  061372  000012                          10.
14172  061374  000004                  $DBLK:  .BLKW   4
14173                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
14174
14175                                  ;;**************************************************************
14176                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
14177                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
14178                                  ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
14179                                  ;*CALL:
14180                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
14181                                  ;*      TYPOS                   ;;CALL FOR TYPEOUT
14182                                  ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
14183                                  ;*      .BYTE   M               ;;M=1 OR 0
14184                                  ;*                                      ;;1=TYPE LEADING ZEROS
14185                                  ;*                                      ;;0=SUPPRESS LEADING ZEROS
14186                                  ;*
14187                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
14188                                  ;*$TYPOS OR $TYPOC
14189                                  ;*CALL:
14190                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
14191                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
14192                                  ;*
14193                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
14194                                  ;*CALL:
14195                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
14196                                  ;*      TYPOC                   ;;CALL FOR TYPEOUT
14197
14198  061404  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
14199  061410  116637  000001  061627          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
14200  061416  112637  061631                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
14201  061422  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
14202  061426  000406                          BR      $TYPON
14203  061430  112737  000001  061627  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
14204  061436  112737  000006  061631          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
```

```
14205  061444  112737  000005  061626  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
14206  061452  010346                   MOV     R3,-(SP)        ;;SAVE R3
14207  061454  010446                   MOV     R4,-(SP)        ;;SAVE R4
14208  061456  010546                   MOV     R5,-(SP)        ;;SAVE R5
14209  061460  113704  061631           MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
14210  061464  005404                   NEG     R4
14211  061466  062704  000006           ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
14212  061472  110437  061630           MOVB    R4,$OMODE       ;;SAVE IT FOR USE
14213  061476  113704  061627           MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
14214  061502  016605  000012           MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
14215  061506  005003                   CLR     R3              ;;CLEAR THE OUTPUT WORD
14216  061510  006105           1$:     ROL     R5              ;;ROTATE MSB INTO 'C'
14217  061512  000404                   BR      3$              ;;GO DO MSB
14218  061514  006105           2$:     ROL     R5              ;;FORM THIS DIGIT
14219  061516  006105                   ROL     R5
14220  061520  006105                   ROL     R5
14221  061522  010503                   MOV     R5,R3
14222  061524  006103           3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
14223  061526  105337  061630           DECB    $OMODE          ;;TYPE THIS DIGIT?
14224  061532  100016                   BPL     7$              ;;BR IF NO
14225  061534  042703  177770           BIC     #177770,R3      ;;GET RID OF JUNK
14226  061540  001002                   BNE     4$              ;;TEST FOR 0
14227  061542  005704                   TST     R4              ;;SUPPRESS THIS 0?
14228  061544  001403                   BEQ     5$              ;;BR IF YES
14229  061546  005204           4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
14230  061550  052703  000060           BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
14231  061554  052703  000040           BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
14232  061560  110337  061624           MOVB    R3,8$           ;;SAVE FOR TYPING
14233  061564  104401  061624           TYPE    ,8$             ;;GO TYPE THIS DIGIT
14234  061570  105337  061626   7$:     DECB    $OCNT           ;;COUNT BY 1
14235  061574  003347                   BGT     2$              ;;BR IF MORE TO DO
14236  061576  002402                   BLT     6$              ;;BR IF DONE
14237  061600  005204                   INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
14238  061602  000744                   BR      2$              ;;GO DO THE LAST DIGIT
14239  061604  012605           6$:     MOV     (SP)+,R5        ;;RESTORE R5
14240  061606  012604                   MOV     (SP)+,R4        ;;RESTORE R4
14241  061610  012603                   MOV     (SP)+,R3        ;;RESTORE R3
14242  061612  016666  000002  000004   MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
14243  061620  012616                   MOV     (SP)+,(SP)
14244  061622  000002                   RTI                     ;;RETURN
14245  061624     000   8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
14246  061625     000           .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
14247  061626     000   $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
14248  061627     000   $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
14249  061630  000000   $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
14250                    .SBTTL  TYPE ROUTINE
14251
14252                    ;;**********************************************************************
14253                    ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
14254                    ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
14255                    ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
14256                    ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
14257                    ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
14258                    ;*
14259                    ;*CALL:
14260                    ;*1) USING A TRAP INSTRUCTION
```

```
14261                                   ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
14262                                   ;*OR
14263                                   ;*      TYPE
14264                                   ;*      MESADR
14265                                   ;*
14266
14267  061632  105737  001173           $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
14268  061636  100002                           BPL     1$              ;;BR IF YES
14269  061640  000000                           HALT                    ;;HALT HERE IF NO TERMINAL
14270  061642  000430                           BR      3$              ;;LEAVE
14271  061644  010046                   1$:     MOV     R0,-(SP)        ;;SAVE R0
14272  061646  017600  000002                   MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
14273  061652  122737  000001  001242           CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
14274  061660  001011                           BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
14275  061662  132737  000100  001243           BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
14276  061670  001405                           BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
14277  061672  010037  061702                   MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
14278  061676  004737  064720                   JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
14279  061702  000000                   61$:    .WORD   0               ;;MESSAGE ADDRESS
14280  061704  132737  000040  001243   62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
14281  061712  001003                           BNE     60$             ;;YES,SKIP TYPE OUT
14282  061714  112046                   2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
14283  061716  001005                           BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
14284  061720  005726                           TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
14285  061722  012600                   60$:    MOV     (SP)+,R0        ;;RESTORE R0
14286  061724  062716  000002           3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
14287  061730  000002                           RTI                     ;;RETURN
14288  061732  122716  000011           4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
14289  061736  001430                           BEQ     8$
14290  061740  122716  000200                   CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
14291  061744  001006                           BNE     5$
14292  061746  005726                           TST     (SP)+           ;;POP  <CR><LF> EQUIV
14293  061750  104401                           TYPE                    ;;TYPE A CR AND LF
14294  061752  001217                           $CRLF
14295  061754  105037  062110                   CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
14296  061760  000755                           BR      2$              ;;GET NEXT CHARACTER
14297  061762  004737  062044           5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
14298  061766  123726  001172           6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
14299  061772  001350                           BNE     2$              ;;IF NO GO GET NEXT CHAR.
14300  061774  013746  001170                   MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
14301                                           ;;AND THE NULL CHAR.
14302  062000  105366  000001           7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
14303  062004  002770                           BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
14304  062006  004737  062044                   JSR     PC,$TYPEC       ;;GO TYPE A NULL
14305  062012  105337  062110                   DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
14306  062016  000770                           BR      7$              ;;LOOP
14307
14308                                   ;HORIZONTAL TAB PROCESSOR
14309
14310  062020  112716  000040           8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
14311  062024  004737  062044           9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
14312  062030  132737  000007  062110           BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
14313  062036  001372                           BNE     9$              ;;TAB STOP
14314  062040  005726                           TST     (SP)+           ;;POP SPACE OFF STACK
14315  062042  000724                           BR      2$              ;;GET NEXT CHARACTER
14316  062044  105777  117114           $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
```

```
14317  062050  100375                          BPL     $TYPEC
14318  062052  116677  000002  117106          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
14319  062060  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
14320  062066  001003                          BNE     1$              ;;BRANCH IF NO
14321  062070  105037  062110                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
14322  062074  000406                          BR      $TYPEX          ;;EXIT
14323  062076  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
14324  062104  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
14325  062106  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
14326  062110  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
14327  062112  000207                  $TYPEX: RTS     PC
14328
14329                                   .SBTTL  SCOPE HANDLER ROUTINE
14330
14331                                   ;;**********************************************************************
14332                                   ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
14333                                   ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
14334                                   ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
14335                                   ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
14336                                   ;*SW14=1        LOOP ON TEST
14337                                   ;*SW11=1        INHIBIT ITERATIONS
14338                                   ;*SW09=1        LOOP ON ERROR
14339                                   ;*SW08=1        LOOP ON TEST IN SWR<7:0>
14340                                   ;*CALL
14341                                   ;*      SCOPE           ;;SCOPE=IOT
14342
14343  062114                          $SCOPE:
14344  062114  104410                          CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
14345  062116  032777  040000  117030  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
14346  062124  001131                          BNE     $OVER           ;;YES IF SW14=1
14347                                   ;#####START OF CODE FOR THE XOR TESTER#####
14348  062126  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
14349                                                                   ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
14350  062130  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
14351  062134  012737  062154  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
14352  062142  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
14353  062146  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
14354  062152  000500                          BR      $SVLAD          ;;GO TO THE NEXT TEST
14355  062154  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
14356  062156  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
14357  062162  000440                          BR      7$              ;;LOOP ON THE PRESENT TEST
14358  062164                          6$:;#####END OF CODE FOR THE XOR TESTER#####
14359  062164  032777  000400  116762          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
14360  062172  001421                          BEQ     2$              ;;BR IF NO
14361  062174  005046                          CLR     -(SP)           ;;CLEAR A TEMP. LOCATION
14362  062176  117716  116752                  MOVB    @SWR,(SP)       ;;PICKUP THE DESIRED TEST NUMBER
14363  062202  001414                          BEQ     8$              ;;BRANCH IF BAD TEST NUMBER IN SWR
14364  062204  022716  000120                  CMP     #120,(SP)       ;;CHECK THE NUMBER IN THE SWR
14365  062210  002411                          BLT     8$              ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
14366  062212  011637  001116                  MOV     (SP),$TSTNM     ;;UPDATE THE TEST NUMBER
14367  062216  005316                          DEC     (SP)            ;;BACKUP BY ONE
14368  062220  006316                          ASL     (SP)            ;;SCALE THE TEST NUMBER AS AN INDEX
14369  062222  062716  062426                  ADD     #$SW08TBL,(SP)  ;;FORM THE ADDRESS OF TEST POINTER
14370  062226  013637  001122                  MOV     @(SP)+,$LPADR   ;;SET LOOP ADDRESS TO DESIRED TEST
14371  062232  000466                          BR      $OVER           ;;GO LOOP ON THE TEST
14372  062234  005726                  8$:     TST     (SP)+           ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
```

```
14373  062236  105737  001117          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
14374  062242  001421                  BEQ     3$              ;;BR IF NO
14375  062244  123737  001131  001117          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
14376  062252  101015                  BHI     3$              ;;BR IF NO
14377  062254  032777  001000  116672          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
14378  062262  001404                  BEQ     4$              ;;BR IF NO
14379  062264  013737  001124  001122  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
14380  062272  000446                  BR      $OVER
14381  062274  105037  001117          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
14382  062300  005037  001206                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
14383  062304  000415                  BR      1$              ;;ESCAPE TO THE NEXT TEST
14384  062306  032777  004000  116640  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
14385  062314  001011                  BNE     1$              ;;BR IF YES
14386  062316  005737  001230                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
14387  062322  001406                  BEQ     1$              ;;      INHIBIT ITERATIONS
14388  062324  005237  001120                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
14389  062330  023737  001206  001120          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
14390  062336  002024                  BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
14391  062340  012737  000001  001120  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
14392  062346  013737  062424  001206          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
14393  062354  105237  001116          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
14394  062360  113737  001116  001226          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
14395  062366  011637  001122                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
14396  062372  011637  001124                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
14397  062376  005037  001210                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
14398  062402  112737  000001  001131          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
14399  062410  013777  001116  116540  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
14400  062416  013716  001122                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
14401  062422  000002                  RTI                     ;;FIXES PS
14402  062424  000012          $MXCNT: 10.                     ;;MAX. NUMBER OF ITERATIONS
14403  062426                  $SW08TBL:
14404  062426  006344                  .WORD   TST1+2          ;;STARTING ADDRESS OF TEST 1
14405  062430  006720                  .WORD   TST2+2          ;;STARTING ADDRESS OF TEST 2
14406  062432  007140                  .WORD   TST3+2          ;;STARTING ADDRESS OF TEST 3
14407  062434  007354                  .WORD   TST4+2          ;;STARTING ADDRESS OF TEST 4
14408  062436  007542                  .WORD   TST5+2          ;;STARTING ADDRESS OF TEST 5
14409  062440  010700                  .WORD   TST6+2          ;;STARTING ADDRESS OF TEST 6
14410  062442  012060                  .WORD   TST7+2          ;;STARTING ADDRESS OF TEST 7
14411  062444  012240                  .WORD   TST10+2         ;;STARTING ADDRESS OF TEST 10
14412  062446  012362                  .WORD   TST11+2         ;;STARTING ADDRESS OF TEST 11
14413  062450  012660                  .WORD   TST12+2         ;;STARTING ADDRESS OF TEST 12
14414  062452  013272                  .WORD   TST13+2         ;;STARTING ADDRESS OF TEST 13
14415  062454  014120                  .WORD   TST14+2         ;;STARTING ADDRESS OF TEST 14
14416  062456  014654                  .WORD   TST15+2         ;;STARTING ADDRESS OF TEST 15
14417  062460  015012                  .WORD   TST16+2         ;;STARTING ADDRESS OF TEST 16
14418  062462  015366                  .WORD   TST17+2         ;;STARTING ADDRESS OF TEST 17
14419  062464  015526                  .WORD   TST20+2         ;;STARTING ADDRESS OF TEST 20
14420  062466  016050                  .WORD   TST21+2         ;;STARTING ADDRESS OF TEST 21
14421  062470  016224                  .WORD   TST22+2         ;;STARTING ADDRESS OF TEST 22
14422  062472  016410                  .WORD   TST23+2         ;;STARTING ADDRESS OF TEST 23
14423  062474  016720                  .WORD   TST24+2         ;;STARTING ADDRESS OF TEST 24
14424  062476  017256                  .WORD   TST25+2         ;;STARTING ADDRESS OF TEST 25
14425  062500  017736                  .WORD   TST26+2         ;;STARTING ADDRESS OF TEST 26
14426  062502  020076                  .WORD   TST27+2         ;;STARTING ADDRESS OF TEST 27
14427  062504  020422                  .WORD   TST30+2         ;;STARTING ADDRESS OF TEST 30
14428  062506  020774                  .WORD   TST31+2         ;;STARTING ADDRESS OF TEST 31
```

```
14429  062510  021322                    .WORD   TST32+2       ;;STARTING ADDRESS OF TEST 32
14430  062512  022076                    .WORD   TST33+2       ;;STARTING ADDRESS OF TEST 33
14431  062514  022446                    .WORD   TST34+2       ;;STARTING ADDRESS OF TEST 34
14432  062516  022776                    .WORD   TST35+2       ;;STARTING ADDRESS OF TEST 35
14433  062520  023356                    .WORD   TST36+2       ;;STARTING ADDRESS OF TEST 36
14434  062522  023744                    .WORD   TST37+2       ;;STARTING ADDRESS OF TEST 37
14435  062524  024464                    .WORD   TST40+2       ;;STARTING ADDRESS OF TEST 40
14436  062526  025014                    .WORD   TST41+2       ;;STARTING ADDRESS OF TEST 41
14437  062530  025256                    .WORD   TST42+2       ;;STARTING ADDRESS OF TEST 42
14438  062532  025716                    .WORD   TST43+2       ;;STARTING ADDRESS OF TEST 43
14439  062534  026124                    .WORD   TST44+2       ;;STARTING ADDRESS OF TEST 44
14440  062536  026542                    .WORD   TST45+2       ;;STARTING ADDRESS OF TEST 45
14441  062540  027310                    .WORD   TST46+2       ;;STARTING ADDRESS OF TEST 46
14442  062542  027642                    .WORD   TST47+2       ;;STARTING ADDRESS OF TEST 47
14443  062544  030302                    .WORD   TST50+2       ;;STARTING ADDRESS OF TEST 50
14444  062546  030574                    .WORD   TST51+2       ;;STARTING ADDRESS OF TEST 51
14445  062550  031040                    .WORD   TST52+2       ;;STARTING ADDRESS OF TEST 52
14446  062552  031730                    .WORD   TST53+2       ;;STARTING ADDRESS OF TEST 53
14447  062554  032240                    .WORD   TST54+2       ;;STARTING ADDRESS OF TEST 54
14448  062556  032472                    .WORD   TST55+2       ;;STARTING ADDRESS OF TEST 55
14449  062560  032734                    .WORD   TST56+2       ;;STARTING ADDRESS OF TEST 56
14450  062562  033166                    .WORD   TST57+2       ;;STARTING ADDRESS OF TEST 57
14451  062564  033404                    .WORD   TST60+2       ;;STARTING ADDRESS OF TEST 60
14452  062566  033660                    .WORD   TST61+2       ;;STARTING ADDRESS OF TEST 61
14453  062570  034056                    .WORD   TST62+2       ;;STARTING ADDRESS OF TEST 62
14454  062572  034332                    .WORD   TST63+2       ;;STARTING ADDRESS OF TEST 63
14455  062574  034476                    .WORD   TST64+2       ;;STARTING ADDRESS OF TEST 64
14456  062576  034700                    .WORD   TST65+2       ;;STARTING ADDRESS OF TEST 65
14457  062600  035166                    .WORD   TST66+2       ;;STARTING ADDRESS OF TEST 66
14458  062602  035434                    .WORD   TST67+2       ;;STARTING ADDRESS OF TEST 67
14459  062604  035712                    .WORD   TST70+2       ;;STARTING ADDRESS OF TEST 70
14460  062606  036200                    .WORD   TST71+2       ;;STARTING ADDRESS OF TEST 71
14461  062610  036442                    .WORD   TST72+2       ;;STARTING ADDRESS OF TEST 72
14462  062612  036704                    .WORD   TST73+2       ;;STARTING ADDRESS OF TEST 73
14463  062614  037170                    .WORD   TST74+2       ;;STARTING ADDRESS OF TEST 74
14464  062616  037334                    .WORD   TST75+2       ;;STARTING ADDRESS OF TEST 75
14465  062620  037516                    .WORD   TST76+2       ;;STARTING ADDRESS OF TEST 76
14466  062622  037676                    .WORD   TST77+2       ;;STARTING ADDRESS OF TEST 77
14467  062624  040060                    .WORD   TST100+2          ;;STARTING ADDRESS OF TEST 100
14468  062626  040242                    .WORD   TST101+2          ;;STARTING ADDRESS OF TEST 101
14469  062630  040462                    .WORD   TST102+2          ;;STARTING ADDRESS OF TEST 102
14470  062632  040722                    .WORD   TST103+2          ;;STARTING ADDRESS OF TEST 103
14471  062634  041106                    .WORD   TST104+2          ;;STARTING ADDRESS OF TEST 104
14472  062636  041350                    .WORD   TST105+2          ;;STARTING ADDRESS OF TEST 105
14473  062640  041536                    .WORD   TST106+2          ;;STARTING ADDRESS OF TEST 106
14474  062642  041770                    .WORD   TST107+2          ;;STARTING ADDRESS OF TEST 107
14475  062644  042216                    .WORD   TST110+2          ;;STARTING ADDRESS OF TEST 110
14476  062646  042524                    .WORD   TST111+2          ;;STARTING ADDRESS OF TEST 111
14477  062650  043066                    .WORD   TST112+2          ;;STARTING ADDRESS OF TEST 112
14478  062652  044510                    .WORD   TST113+2          ;;STARTING ADDRESS OF TEST 113
14479  062654  046266                    .WORD   TST114+2          ;;STARTING ADDRESS OF TEST 114
14480  062656  050476                    .WORD   TST115+2          ;;STARTING ADDRESS OF TEST 115
14481  062660  051106                    .WORD   TST116+2          ;;STARTING ADDRESS OF TEST 116
14482  062662  053304                    .WORD   TST117+2          ;;STARTING ADDRESS OF TEST 117
14483  062664  055622                    .WORD   TST120+2          ;;STARTING ADDRESS OF TEST 120
14484                          .SBTTL  ERROR HANDLER ROUTINE
```

```
14485
14486                                   ;;**********.**********************************************
14487                                   ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
14488                                   ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
14489                                   ;*AND GO TO ERRTYP ON ERROR
14490                                   ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
14491                                   ;*SW15=1        HALT ON ERROR
14492                                   ;*SW13=1        INHIBIT ERROR TYPEOUTS
14493                                   ;*SW10=1        BELL ON ERROR
14494                                   ;*SW09=1        LOOP ON ERROR
14495                                   ;*CALL
14496                                   ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
14497
14498  062666                           $ERROR:
14499  062666  104410                           CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
14500  062670  105237  001117           7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
14501  062674  001775                           BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
14502  062676  013777  001116  116252           MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
14503  062704  032777  002000  116242           BIT     #BIT10,@SWR     ;;BELL ON ERROR?
14504  062712  001402                           BEQ     1$              ;;NO - SKIP
14505  062714  104401  001212                   TYPE    ,$BELL          ;;RING BELL
14506  062720  005237  001126           1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
14507  062724  011637  001132                   MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
14508  062730  162737  000002  001132           SUB     #2,$ERRPC
14509  062736  117737  116170  001130           MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
14510  062744  032777  020000  116202           BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
14511  062752  001004                           BNE     20$             ;;SKIP TYPEOUTS
14512  062754  004737  057344                   JSR     PC,ERRTYP       ;;GO TO USER ERROR ROUTINE
14513  062760  104401  001217                   TYPE    ,$CRLF
14514  062764                           20$:
14515  062764  122737  000001  001242           CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
14516  062772  001007                           BNE     2$              ;;NO,SKIP APT ERROR REPORT
14517  062774  113737  001130  063006           MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
14518  063002  004737  064730                   JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
14519  063006     000           21$:    .BYTE   0
14520  063007     000                   .BYTE   0
14521  063010  000777           22$:    BR      22$             ;;APT ERROR LOOP
14522  063012  005777  116136           2$:     TST     @SWR            ;;HALT ON ERROR
14523  063016  100002                           BPL     3$              ;;SKIP IF CONTINUE
14524  063020  000000                           HALT                    ;;HALT ON ERROR!
14525  063022  104410                           CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
14526  063024  032777  001000  116122   3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
14527  063032  001402                           BEQ     4$              ;;BR IF NO
14528  063034  013716  001124                   MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
14529  063040  005737  001210           4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
14530  063044  001402                           BEQ     5$              ;;BR IF NONE
14531  063046  013716  001210                   MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
14532  063052                           5$:
14533  063052  022737  057324  000042           CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
14534  063060  001001                           BNE     6$              ;;BRANCH IF NO
14535  063062  000000                           HALT                    ;;YES
14536  063064                           6$:
14537  063064  000002                           RTI                     ;;RETURN
14538                                   .SBTTL  TTY INPUT ROUTINE
14539
14540                                   ;;**************************************************************************
```

```
14541                                   .ENABL  LSB
14542  063066  000000                   $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
14543  063070  000000                   $TKQIN: .WORD   0               ;;INPUT POINTER
14544  063072  000000                   $TKQOUT: .WORD  0               ;;OUTPUT POINTER
14545  063074  000001                   $TKQSRT: .BLKB  1               ;;TTY KEYBOARD QUEUE
14546          063075                   $TKQEND=.
14547          063076                   .EVEN
14548
14549                                   ;*TK INITIALIZE ROUTINE
14550                                   ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
14551                                   ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
14552                                   ;
14553                                   ;*CALL:
14554                                   ;*      JSR     PC,$TKINT
14555                                   ;*      RETURN
14556                                   ;
14557  063076  005037  063066           $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
14558  063102  012737  063074  063070           MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
14559  063110  013737  063070  063072           MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
14560  063116  012737  063146  000060           MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
14561  063124  012737  000200  000062           MOV     #200,@#TKVEC+2  ;;'BR' LEVEL 4
14562  063132  005777  116024                   TST     @$TKB           ;;CLEAR DONE FLAG
14563  063136  012777  000100  116014           MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
14564  063144  000207                           RTS     PC              ;;RETURN TO CALLER
14565
14566                                   ;*TK SERVICE ROUTINE
14567                                   ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
14568                                   ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
14569                                   ;*IT IN THE QUEUE.
14570                                   ;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
14571                                   ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (START)
14572                                   ;
14573  063146  117746  116010          $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
14574  063152  042716  177600                  BIC     #^C177,(SP)     ;;STRIP THE JUNK
14575  063156  021627  000003                   CMP     (SP),#3         ;;IS IT A CONTROL C?
14576  063162  001007                           BNE     1$              ;;BRANCH IF NO
14577  063164  104401  064262                   TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
14578  063170  004737  063076                   JSR     PC,$TKINT       ;;INIT THE KEYBOARD
14579  063174  005726                           TST     (SP)+           ;;CLEAN UP STACK
14580  063176  000137  004542                   JMP     START           ;;CONTROL C RESTART
14581  063202  021627  000007          1$:      CMP     (SP),#7         ;;IS IT A CONTROL G?
14582  063206  001004                           BNE     2$              ;;BRANCH IF NO
14583  063210  022737  000176  001154           CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
14584  063216  001500                           BEQ     6$              ;;GO TO SWR CHANGE
14585
14586  063220                           2$:
14587  063220  022737  000001  063066           CMP     #1,$TKCNT       ;;IS THE QUEUE FULL?
14588  063226  001004                           BNE     3$              ;;BRANCH IF NO
14589  063230  104401  001212                   TYPE    ,$BELL          ;;RING THE TTY BELL
14590  063234  005726                           TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
14591  063236  000451                           BR      5$              ;;EXIT
14592  063240  021627  000023          3$:      CMP     (SP),#23        ;;IS IT A CONTROL-S?
14593  063244  001021                           BNE     32$             ;;BRANCH IF NO
14594  063246  005077  115706                   CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
14595  063252  005726                           TST     (SP)+           ;;CLEAN CHAR OFF STACK
14596  063254  105777  115700          31$:     TSTB    @$TKS           ;;WAIT FOR A CHAR
```

```
14597  063260  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
14598  063262  117746  115674                  MOVB    @$TKB,-(SP)     ;;GET THE CHARACTER
14599  063266  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
14600  063272  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
14601  063276  001366                          BNE     31$             ;;BRANCH IF NO
14602  063300  012777  000100  115652          MOV     #100,@$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
14603  063306  000002                          RTI                     ;;RETURN
14604  063310  005237  063066          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
14605  063314  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
14606  063320  002405                          BLT     4$              ;;BRANCH IF YES
14607  063322  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
14608  063326  003002                          BGT     4$              ;;BRANCH IF YES
14609  063330  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
14610  063334  112677  177530          4$:     MOVB    (SP)+,@$TKQIN   ;;AND PUT IT IN QUEUE
14611  063340  005237  063070                  INC     $TKQIN          ;;UPDATE THE POINTER
14612  063344  023727  063070  063075          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
14613  063352  001003                          BNE     5$              ;;BRANCH IF NO
14614  063354  012737  063074  063070          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
14615  063362  000002                  5$:     RTI                     ;;RETURN
14616
14617                                  ;;**********************************************************************
14618                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
14619                                  ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
14620                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
14621                                  ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
14622  063364  022737  000176  001154  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
14623  063372  001124                          BNE     15$             ;;EXIT IF NOT
14624  063374  105777  115560                  TSTB    @$TKS           ;;IS A CHAR WAITING?
14625  063400  100121                          BPL     15$             ;;IF NOT, EXIT
14626  063402  117746  115554                  MOVB    @$TKB,-(SP)     ;;YES
14627  063406  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
14628  063412  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
14629  063416  001300                          BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
14630                                                                  ;;AND EXIT
14631
14632                                  ;;**********************************************************************
14633                                  ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
14634                                  ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
14635                                  ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
14636  063420  123727  001150  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
14637  063426  001674                          BEQ     2$              ;;BRANCH IF YES
14638  063430  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
14639  063432  004737  063076                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
14640  063436  005077  115516                  CLR     @$TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
14641  063442  112737  000001  001151          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR
14642
14643  063450  104401  064274                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
14644  063454  104401  064301  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
14645  063460  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
14646  063464  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
14647  063466  104401  064312                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
14648  063472  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
14649  063474  005046                          CLR     -(SP)           ;;THE NEW SWR
14650  063476  105777  115456          7$:     TSTB    @$TKS           ;;CHAR THERE?
14651  063502  100375                          BPL     7$              ;;IF NOT TRY AGAIN
14652
```

```
14653  063504  117746  115452                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
14654  063510  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
14655
14656  063514  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL-C?
14657  063520  001015                          BNE     9$              ;;BRANCH IF NOT
14658  063522  104401  064262                  TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
14659  063526  062706  000006                  ADD     #6,SP           ;;CLEAN UP STACK
14660  063532  123727  001151  000001          CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
14661  063540  001003                          BNE     8$              ;;BRANCH IF NO
14662  063542  012777  000100  115410          MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
14663  063550  000137  004542          8$:     JMP     START           ;;CONTROL-C RESTART
14664
14665
14666  063554  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
14667  063560  001005                          BNE     10$             ;;BRANCH IF NOT
14668  063562  104401  064267                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
14669  063566  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
14670  063572  000737                          BR      19$             ;;LET'S TRY IT AGAIN
14671
14672
14673  063574  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
14674  063600  001022                          BNE     16$             ;;BRANCH IF NO
14675  063602  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
14676  063606  001403                          BEQ     11$             ;;BRANCH IF YES
14677  063610  016677  000002  115336          MOV     2(SP),@SWR      ;;SAVE NEW SWR
14678  063616  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
14679  063622  104401  001217          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
14680  063626  123727  001151  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
14681  063634  001003                          BNE     15$             ;;BRANCH IF NOT
14682  063636  012777  000100  115314          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
14683  063644  000002                  15$:    RTI                     ;;RETURN
14684  063646  004737  062044          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
14685  063652  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
14686  063656  002420                          BLT     18$             ;;BRANCH IF YES
14687  063660  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
14688  063664  003015                          BGT     18$             ;;BRANCH IF YES
14689  063666  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
14690  063672  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
14691  063676  001403                          BEQ     17$             ;;BRANCH IF YES
14692  063700  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
14693  063702  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
14694  063704  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
14695  063706  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
14696  063712  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
14697  063716  000667                          BR      7$              ;;GET THE NEXT ONE
14698  063720  104401  001216          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
14699  063724  000720                          BR      20$             ;;SIMULATE CONTROL-U
14700                                  .DSABL  LSB
14701
14702
14703                                  ;;*******************************************************************
14704                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
14705                                  ;*CALL:
14706                                  ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
14707                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
14708                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
```

```
14709                                   ;
14710
14711  063726  011646                   $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
14712  063730  016666  000004  000002           MOV     4(SP),2(SP)     ;;THE PS
14713  063736  005066  000004                   CLR     4(SP)           ;;GET READY FOR A CHARACTER
14714  063742  005046                           CLR     -(SP)           ;;PUT NEW PS ON STACK
14715  063744  012746  063752                   MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
14716  063750  000002                           RTI                     ;;POP NEW PC AND PS
14717  063752                           64$:
14718  063752  005737  063066           1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
14719  063756  001775                           BEQ     1$
14720  063760  005337  063066                   DEC     $TKCNT          ;;DECREMENT THE COUNTER
14721  063764  117766  177102  000004           MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
14722  063772  005237  063072                   INC     $TKQOUT         ;;UPDATE THE POINTER
14723  063776  023727  063072  063075           CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
14724  064004  001003                           BNE     2$              ;;BRANCH IF NO
14725  064006  012737  063074  063072           MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
14726  064014  000002           2$:     RTI                             ;;RETURN
14727                                   ;;*****************************************************************
14728                                   ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
14729                                   ;*CALL:
14730                                   ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
14731                                   ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
14732                                   ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
14733
14734  064016  010346                   $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
14735  064020  005046                           CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
14736  064022  012703  064252           1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
14737  064026  022703  064262           2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
14738  064032  101456                           BLOS    4$              ;;BR IF YES
14739  064034  104411                           RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
14740  064036  112613                           MOVB    (SP)+,(R3)      ;;GET CHARACTER
14741  064040  122713  000177           10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
14742  064044  001022                           BNE     5$              ;;BR IF NO
14743  064046  005716                           TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
14744  064050  001007                           BNE     6$              ;;BR IF NO
14745  064052  112737  000134  064250           MOVB    #'\,9$          ;;TYPE A BACK SLASH
14746  064060  104401  064250                   TYPE    ,9$
14747  064064  012716  177777                   MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
14748  064070  005303                   6$:     DEC     R3              ;;BACKUP BY ONE
14749  064072  020327  064252                   CMP     R3,#$TTYIN      ;;STACK EMPTY?
14750  064076  103434                           BLO     4$              ;;BR IF YES
14751  064100  111337  064250                   MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
14752  064104  104401  064250                   TYPE    ,9$             ;;GO TYPE
14753  064110  000746                           BR      2$              ;;GO READ ANOTHER CHAR.
14754  064112  005716                   5$:     TST     (SP)            ;;RUBOUT KEY SET?
14755  064114  001406                           BEQ     7$              ;;BR IF NO
14756  064116  112737  000134  064250           MOVB    #'\,9$          ;;TYPE A BACK SLASH
14757  064124  104401  064250                   TYPE    ,9$
14758  064130  005016                           CLR     (SP)            ;;CLEAR THE RUBOUT KEY
14759  064132  122713  000025           7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
14760  064136  001003                           BNE     8$              ;;BR IF NO
14761  064140  104401  06 267                   TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
14762  064144  000726                           BR      1$              ;;GO START OVER
14763  064146  122713  000022           8$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
14764  064152  001011                           BNE     3$              ;;BRANCH IF NO
```

```
14765  064154  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
14766  064156  104401  001217                  TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
14767  064162  104401  064252                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
14768  064166  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
14769  064170  104401  001216          4$:     TYPE    ,$QUES          ;;TYPE A '?'
14770  064174  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
14771  064176  111337  064250          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
14772  064202  104401  064250                  TYPE    ,9$
14773  064206  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
14774  064212  001305                          BNE     2$              ;;LOOP IF NOT RETURN
14775  064214  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
14776  064220  104401  001220                  TYPE    ,$LF            ;;TYPE A LINE FEED
14777  064224  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
14778  064226  012603                          MOV     (SP)+,R3        ;;RESTORE R3
14779  064230  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
14780  064232  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
14781  064240  012766  064252  000004          MOV     #$TTYIN,4(SP)
14782  064246  000002                          RTI                     ;;RETURN
14783  064250     000          9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
14784  064251     000                  .BYTE   0               ;;TERMINATOR
14785  064252  000010          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
14786  064262  041536  005015     000  $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL "C"
14787  064267     136  006525  000012  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
14788  064274  043536  005015     000  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
14789  064301     015  051412  051127  $MSWR:  .ASCIZ  <15><12>/SWR = /
14790  064306  036440  000040
14791  064312  020040  042516  020127  $MNEW:  .ASCIZ  /  NEW = /
14792  064320  020075     000
14793          064324                  .EVEN
14794                                  .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
14795
14796                                  ;;**************************************************************************
14797                                  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
14798                                  ;*CHANGE IT TO BINARY.
14799                                  ;*CALL:
14800                                  ;*      RDOCT                   ;;READ AN OCTAL NUMBER
14801                                  ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
14802                                  ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
14803
14804  064324  011646          $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
14805  064326  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
14806  064334  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
14807  064336  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
14808  064340  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
14809  064342  104412          1$:     RDLIN                   ;;READ AN ASCIZ LINE
14810  064344  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
14811  064346  005001                          CLR     R1              ;;CLEAR DATA WORD
14812  064350  005002                          CLR     R2
14813  064352  112046          2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
14814  064354  001412                          BEQ     3$              ;;IF ZERO GET OUT
14815  064356  006301                          ASL     R1              ;;*2
14816  064360  006102                          ROL     R2
14817  064362  006301                          ASL     R1              ;;*4
14818  064364  006102                          ROL     R2
14819  064366  006301                          ASL     R1              ;;*8
14820  064370  006102                          ROL     R2
```

```
14821  064372  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
14822  064376  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
14823  064400  000764                          BR      2$              ;;LOOP
14824  064402  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
14825  064404  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
14826  064410  010237  064424                  MOV     R2,$HIOCT
14827  064414  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
14828  064416  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
14829  064420  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
14830  064422  000002                          RTI                     ;;RETURN
14831  064424  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
14832                                  .SBTTL  TRAP DECODER
14833
14834                                  ;;**************************************************************************
14835                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE ''TRAP'' INSTRUCTION
14836                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
14837                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
14838                                  ;*GO TO THAT ROUTINE.
14839
14840  064426  016646  000002          $TRAP:  MOV     2(SP),-(SP)     ;;ASSUME THE STATUS OF
14841  064432  042716  000020                  BIC     #20,(SP)        ;; THE CALLER--DO NOT ALLOW
14842  064436  012746  064444                  MOV     #1$,-(SP)       ;; T-BIT TRAPS
14843  064442  000002                          RTI                     ;;SET THE NEW STATUS
14844  064444  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
14845  064446  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
14846  064452  005740                          TST     -(R0)           ;;BACKUP BY 2
14847  064454  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
14848  064456  006300                          ASL     R0              ;;POSITION FOR INDEXING
14849  064460  016000  064500                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
14850  064464  000200                          RTS     R0              ;;GO TO ROUTINE
14851
14852
14853                                  ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO
14854
14855  064466  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
14856  064470  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
14857  064476  000002                          RTI                     ;;RESTORE THE PSW
14858
14859                                  .SBTTL  TRAP TABLE
14860
14861                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
14862                                  ;*BY THE ''TRAP'' INSTRUCTION.
14863
14864                                  ;       ROUTINE
14865                                  ;       -------
14866  064500  064466                  $TRPAD: .WORD   $TRAP2
14867  064502  061632                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
14868  064504  061430                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
14869  064506  061404                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
14870  064510  061444                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
14871  064512  061160                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
14872  064514  061104                          $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
14873
14874  064516  063454                          $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING
14875
14876  064520  063364                          $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
```

```
14877  064522  063726                          $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
14878  064524  064016                          $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
14879  064526  064324                          $RDOCT  ;;CALL=RDOCT    TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
14880  064530  061010                          $SAVREG ;;CALL=SAVREG   TRAP+14(104414) SAVE R0-R5 ROUTINE
14881  064532  061046                          $RESREG ;;CALL=RESREG   TRAP+15(104415) RESTORE R0-R5 ROUTINE
14882                                  .SBTTL  POWER DOWN AND UP ROUTINES
14883
14884                                  ;;**********************************************************************
14885                                  ;POWER DOWN ROUTINE
14886  064534  012737  064674  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
14887  064542  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
14888  064550  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
14889  064552  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
14890  064554  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
14891  064556  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
14892  064560  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
14893  064562  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
14894  064564  017746  114364                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
14895  064570  010637  064700                  MOV     SP,$SAVR6       ;;SAVE SP
14896  064574  012737  064606  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
14897  064602  000000                          HALT
14898  064604  000776                          BR      .-2             ;;HANG UP
14899
14900                                  ;;**********************************************************************
14901                                  ;POWER UP ROUTINE
14902  064606  012737  064674  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
14903  064614  013706  064700                  MOV     $SAVR6,SP       ;;GET SP
14904  064620  005037  064700                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
14905  064624  005237  064700          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
14906  064630  001375                          BNE     1$              ;;OF  WORD
14907  064632  012677  114316                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
14908  064636  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
14909  064640  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
14910  064642  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
14911  064644  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
14912  064646  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
14913  064650  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
14914  064652  012737  064534  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
14915  064660  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
14916  064666  104401                          TYPE                    ;REPORT THE POWER FAILURE
14917  064670  064702                  $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
14918  064672  000002                          RTI
14919  064674  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
14920  064676  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
14921  064700  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
14922  064702  005015  047520  042527  $POWER: .ASCIZ   <15><12>'POWER''
14923  064710  000122
14924                                          .EVEN
14925                                  .SBTTL  APT COMMUNICATIONS ROUTINE
14926
14927                                  ;;**********************************************************************
14928  064712  112737  000001  065156  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
14929  064720  112737  000001  065154  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
14930  064726  000403                          BR      $ATYC
14931  064730  112737  000001  065156  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
14932  064736                          $ATYC:
```

```
14933  064736  010046                      MOV     R0,-(SP)          ;;PUSH R0 ON STACK
14934  064740  010146                      MOV     R1,-(SP)          ;;PUSH R1 ON STACK
14935  064742  105737  065154              TSTB    $MFLG             ;;SHOULD TYPE A MESSAGE?
14936  064746  001450                      BEQ     5$                ;;IF NOT:  BR
14937  064750  122737  000001  001242      CMPB    #APTENV,$ENV      ;;OPERATING UNDER APT?
14938  064756  001031                      BNE     3$                ;;IF NOT:  BR
14939  064760  132737  000100  001243      BITB    #APTSPOOL,$ENVM   ;;SHOULD SPOOL MESSAGES?
14940  064766  001425                      BEQ     3$                ;;IF NOT:  BR
14941  064770  017600  000004              MOV     @4(SP),R0         ;;GET MESSAGE ADDR.
14942  064774  062766  000002  000004      ADD     #2,4(SP)                  ;;BUMP RETURN ADDR.
14943  065002  005737  001222      1$:     TST     $MSGTYPE          ;;SEE IF DONE W/ LAST XMISSION?
14944  065006  001375                      BNE     1$                ;;IF NOT:  WAIT
14945  065010  010037  001236              MOV     R0,$MSGAD         ;;PUT ADDR IN MAILBOX
14946  065014  105720              2$:     TSTB    (R0)+             ;;FIND END OF MESSAGE
14947  065016  001376                      BNE     2$
14948  065020  163700  001236              SUB     $MSGAD,R0         ;;SUB START OF MESSAGE
14949  065024  006200                      ASR     R0                ;;GET MESSAGE LNGTH IN WORDS
14950  065026  010037  001240              MOV     R0,$MSGLGT        ;;PUT LENGTH IN MAILBOX
14951  065032  012737  000004  001222      MOV     #4,$MSGTYPE       ;;TELL APT TO TAKE MSG.
14952  065040  000413                      BR      5$
14953  065042  017637  000004  065066  3$: MOV     @4(SP),4$         ;;PUT MSG ADDR IN JSR LINKAGE
14954  065050  062766  000002  000004      ADD     #2,4(SP)                  ;;BUMP RETURN ADDRESS
14955  065056  013746  177776              MOV     177776,-(SP)      ;;PUSH 177776 ON STACK
14956  065062  004737  061632              JSR     PC,$TYPE          ;;CALL TYPE MACRO
14957  065066  000000              4$:     .WORD   0
14958  065070                      5$:
14959  065070  105737  065156      10$:    TSTB    $FFLG             ;;SHOULD REPORT FATAL ERROR?
14960  065074  001416                      BEQ     12$               ;;IF NOT:  BR
14961  065076  005737  001242              TST     $ENV              ;;RUNNING UNDER APT?
14962  065102  001413                      BEQ     12$               ;;IF NOT:  BR
14963  065104  005737  001222      11$:    TST     $MSGTYPE          ;;FINISHED LAST MESSAGE?
14964  065110  001375                      BNE     11$               ;;IF NOT:  WAIT
14965  065112  017637  000004  001224      MOV     @4(SP),$FATAL     ;;GET ERROR #
14966  065120  062766  000002  000004      ADD     #2,4(SP)                  ;;BUMP RETURN ADDR.
14967  065126  005237  001222              INC     $MSGTYPE          ;;TELL APT TO TAKE ERROR
14968  065132  105037  065156      12$:    CLRB    $FFLG             ;;CLEAR FATAL FLAG
14969  065136  105037  065155              CLRB    $LFLG             ;;CLEAR LOG FLAG
14970  065142  105037  065154              CLRB    $MFLG             ;;CLEAR MESSAGE FLAG
14971  065146  012601                      MOV     (SP)+,R1          ;;POP STACK INTO R1
14972  065150  012600                      MOV     (SP)+,R0          ;;POP STACK INTO R0
14973  065152  000207                      RTS     PC                ;;RETURN
14974  065154     000      $MFLG:  .BYTE   0                 ;;MESSG. FLAG
14975  065155     000      $LFLG:  .BYTE   0                 ;;LOG FLAG
14976  065156     000      $FFLG:  .BYTE   0                 ;;FATAL FLAG
14977          065160              .EVEN
14978          000200      APTSIZE=200
14979          000001      APTENV=001
14980          000100      APTSPOOL=100
14981          000040      APTCSUP=040
14982
```

```
14983                                   .SBTTL  CONSOLE MESSAGES
14984
14985                                   .NLIST  BEX
065160      051                         CLSPRN: .ASCII  @)@
065161      075     000                 EQUALS: .ASCIZ  @=@
065163      015  025012     000         PROMPT: .ASCIZ  <CR><LF>@*@
065167      077     000                 QSTMRK: .ASCIZ  @?@
065171                                  HELPQST:
065171      015  052012  050131         .ASCIZ  <CR><LF>@TYPE HELP TEXT (Y OR N)??@
065225                                  UBUSQST:
065225      015  041412  040510         .ASCIZ  <CR><LF>@CHANGE RM03 UNIBUS ADDRESS OR VECTOR ADDRESS (Y OR N)<CR> ??@
065324   005015  051525  020105         CNSL00: .ASCIZ  <CR><LF>@USE SAME DEVICES (Y OR N) ??@
065363      015  051012  030115         CNSL01: .ASCIZ  <CR><LF>@RM03 BUS ADDRESS (@
065410   005015  047105  051124         CNSL02: .ASCII  <CR><LF>@ENTRY NOT IN I/O PAGE@
065437      015  040412  042104                 .ASCIZ  <CR><LF>@ADDRESS MUST BE >160000@
065471      015  051012  030115         CNSL03: .ASCIZ  <CR><LF>@RM03 VECTOR ADDRESS (@
065521      015  042412  052116         CNSL04: .ASCII  <CR><LF>@ENTRY OUT OF RANGE@
065545      015  040412  042104                 .ASCIZ  <CR><LF>@ADDRESS MUST BE <1000@
065575      015  051012  030115         CNSL05: .ASCIZ  <CR><LF>@RM03 INTERRUPT PRIORITY (@
065631      015  042412  052116         CNSL06: .ASCIZ  <CR><LF>@ENTRY OUT OF RANGE@
065656                                  CNSL07:
065656   005015  054524  042520         .ASCII  <CR><LF>@TYPE (A) TO TEST ALL DEVICES, OR TYPE DEVICE@
065734   047040  046525  042502         .ASCII  @ NUMBER(S)@
065746   005015  042524  046522         .ASCIZ  <CR><LF>@TERMINATE INPUT WITH CARRIAGE RETURN@
066015      015  047012  052117         NOTEX:  .ASCIZ  <CR><LF>/NOT EXIST DRIVE  /
                                        .LIST BEX
14986
14987           066042                  .EVEN
```

```
14988
14989  066042                   FNCDTB:                          ;FUNCTION CODE TABLE
14990
14991  066042  020000             .WORD   OPI                    ;NOP
14992  066044  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (2)
14993  066046  132000             .WORD   ATA!OPI!IVC!IAE        ;SEEK
14994  066050  130000             .WORD   ATA!OPI!IVC            ;RECALIBRATE
14995  066052  020000             .WORD   OPI                    ;DRIVE CLEAR
14996  066054  030000             .WORD   OPI!IVC                ;RELEASE
14997  066056  130000             .WORD   OPI!ATA!IVC            ;OFFSET
14998  066060  130000             .WORD   OPI!ATA!IVC            ;RETURN TO CENTERLINE
14999  066062  020000             .WORD   OPI                    ;READ IN PRESET
15000  066064  020000             .WORD   OPI                    ;PACK ACKNOWLEDGE
15001  066066  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (24)
15002  066070  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (26)
15003  066072  132000             .WORD   ATA!OPI!IVC!IAE        ;SEARCH
15004  066074  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (32)
15005  066076  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (34)
15006  066100  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (36)
15007  066102  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (40)
15008  066104  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (42)
15009  066106  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (44)
15010  066110  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (46)
15011  066112  073300             .WORD   WCE!OPI!IVC!IAE!AOE!HCE!ECH  ;WRITE CHECK DATA
15012  066114  073300             .WORD   WCE!OPI!IVC!IAE!AOE!HCE!ECH  ;WRITE CHECK HEADER AND DATA
15013  066116  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (54)
15014  066120  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (56)
15015  066122  037200             .WORD   OPI!IVC!WLE!IAE!AOE!HCE  ;WRITE DATA
15016  066124  037000             .WORD   OPI!IVC!WLE!IAE!AOE    ;WRITE HEADER AND DATA
15017  066126  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (64)
15018  066130  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (66)
15019  066132  033300             .WORD   OPI!IVC!IAE!AOE!HCE!ECH  ;READ DATA
15020  066134  033300             .WORD   OPI!IVC!IAE!AOE!HCE!ECH  ;READ HEADER AND DATA
15021  066136  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (74)
15022  066140  130001             .WORD   OPI!ATA!ILF!IVC        ;ILLEGAL FUNCTION (76)
15023
15024  066142      001          ATNTBL: .BYTE   1.
15025  066143      002                  .BYTE   2.
15026  066144      004                  .BYTE   4.
15027  066145      010                  .BYTE   8.
15028  066146      020                  .BYTE   16.
15029  066147      040                  .BYTE   32.
15030  066150      100                  .BYTE   64.
15031  066151      200                  .BYTE   128.
15032
15033  066152                   RGDTPT:
15034  066152  000000                   .WORD   0.
15035  066154  000001                   .WORD   1.
15036  066156  000003                   .WORD   3.
15037  066160  000007                   .WORD   7.
15038  066162  000017                   .WORD   15.
15039  066164  000037                   .WORD   31.
15040  066166  000077                   .WORD   63.
15041  066170  000177                   .WORD   127.
15042  066172  000377                   .WORD   255.
15043  066174  000777                   .WORD   511.
```

```
15044  066176  001777                    .WORD   1023.
15045  066200  003777                    .WORD   2047.
15046  066202  007777                    .WORD   4095.
15047  066204  017777                    .WORD   8191.
15048  066206  037777                    .WORD   16383.
15049  066210  077777                    .WORD   32767.
15050  066212  177777            ONES:   .WORD   65535.
15051  066214  177777                    .WORD   65535.
15052  066216  077777                    .WORD   32767.
15053  066220  037777                    .WORD   16383.
15054  066222  017777                    .WORD   8191.
15055  066224  007777                    .WORD   4095.
15056  066226  003777                    .WORD   2047.
15057  066230  001777                    .WORD   1023.
15058  066232  000777                    .WORD   511.
15059  066234  000377                    .WORD   255.
15060  066236  000177                    .WORD   127.
15061  066240  000077                    .WORD   63.
15062  066242  000037                    .WORD   31.
15063  066244  000017                    .WORD   15.
15064  066246  000007                    .WORD   7.
15065  066250  000003                    .WORD   3.
15066  066252  000001                    .WORD   1.
15067  066254  000000            ZEROS:  .WORD   0.
15068  066256  000000                    .WORD   0.
15069  066260  000001                    .WORD   1.
15070  066262  000002                    .WORD   2.
15071  066264  000004                    .WORD   4.
15072  066266  000010                    .WORD   8.
15073  066270  000020                    .WORD   16.
15074  066272  000040                    .WORD   32.
15075  066274  000100                    .WORD   64.
15076  066276  000200                    .WORD   128.
15077  066300  000400                    .WORD   256.
15078  066302  001000                    .WORD   512.
15079  066304  002000                    .WORD   1024.
15080  066306  004000                    .WORD   2048.
15081  066310  010000                    .WORD   4096.
15082  066312  020000                    .WORD   8192.
15083  066314  040000                    .WORD   16384.
15084  066316  100000                    .WORD   32768.
15085  066320  100000                    .WORD   32768.
15086  066322  040000                    .WORD   16384.
15087  066324  020000                    .WORD   8192.
15088  066326  010000                    .WORD   4096.
15089  066330  004000                    .WORD   2048.
15090  066332  002000                    .WORD   1024.
15091  066334  001000                    .WORD   512.
15092  066336  000400                    .WORD   256.
15093  066340  000200                    .WORD   128.
15094  066342  000100                    .WORD   64.
15095  066344  000040                    .WORD   32.
15096  066346  000020                    .WORD   16.
15097  066350  000010                    .WORD   8.
15098  066352  000004                    .WORD   4.
15099  066354  000002                    .WORD   2.
```

```
15100  066356  000001                    .WORD   1.
15101  066360  000000                    .WORD   0.
15102  066362  177777                    .WORD   65535.
15103  066364  177776                    .WORD   65534.
15104  066366  177774                    .WORD   65532.
15105  066370  177770                    .WORD   65528.
15106  066372  177760                    .WORD   65520.
15107  066374  177740                    .WORD   65504.
15108  066376  177700                    .WORD   65472.
15109  066400  177600                    .WORD   65408.
15110  066402  177400                    .WORD   65280.
15111  066404  177000                    .WORD   65024.
15112  066406  176000                    .WORD   64512.
15113  066410  174000                    .WORD   63488.
15114  066412  170000                    .WORD   61440.
15115  066414  160000                    .WORD   57344.
15116  066416  140000                    .WORD   49152.
15117  066420  100000                    .WORD   32768.
15118  066422  000000                    .WORD   0.
15119  066424  000000                    .WORD   0.
15120  066426  100000                    .WORD   32768.
15121  066430  140000                    .WORD   49152.
15122  066432  160000                    .WORD   57344.
15123  066434  170000                    .WORD   61440.
15124  066436  174000                    .WORD   63488.
15125  066440  176000                    .WORD   64512.
15126  066442  177000                    .WORD   65024.
15127  066444  177400                    .WORD   65280.
15128  066446  177600                    .WORD   65408.
15129  066450  177700                    .WORD   65472.
15130  066452  177740                    .WORD   65504.
15131  066454  177760                    .WORD   65520.
15132  066456  177770                    .WORD   65528.
15133  066460  177774                    .WORD   65532.
15134  066462  177776                    .WORD   65534.
15135  066464  177777                    .WORD   65535.
15136  066466  125252                    .WORD   43690.
15137  066470  152525                    .WORD   43690./2
15138  066472  125252                    .WORD   43690.
15139  066474  177777                    .WORD   65535.
15140  066476  177776                    .WORD   65534.
15141  066500  177775                    .WORD   65533.
15142  066502  177773                    .WORD   65531.
15143  066504  177767                    .WORD   65527.
15144  066506  177757                    .WORD   65519.
15145  066510  177737                    .WORD   65503.
15146  066512  177677                    .WORD   65471.
15147  066514  177577                    .WORD   65407.
15148  066516  177377                    .WORD   65279.
15149  066520  176777                    .WORD   65023.
15150  066522  175777                    .WORD   64511.
15151  066524  173777                    .WORD   63487.
15152  066526  167777                    .WORD   61439.
15153  066530  157777                    .WORD   57343.
15154  066532  137777                    .WORD   49151.
15155  066534  077777                    .WORD   32767.
```

```
15156  066536  077777                          .WORD   32767.
15157  066540  137777                          .WORD   49151.
15158  066542  157777                          .WORD   57343.
15159  066544  167777                          .WORD   61439.
15160  066546  173777                          .WORD   63487.
15161  066550  175777                          .WORD   64511.
15162  066552  176777                          .WORD   65023.
15163  066554  177377                          .WORD   65279.
15164  066556  177577                          .WORD   65407.
15165  066560  177677                          .WORD   65471.
15166  066562  177737                          .WORD   65503.
15167  066564  177757                          .WORD   65519.
15168  066566  177767                          .WORD   65527.
15169  066570  177773                          .WORD   65531.
15170  066572  177775                          .WORD   65533.
15171  066574  177776                          .WORD   65534.
15172  066576  177777                          .WORD   65535.
15173  066600                          ENRGDT:
```

```
15174                                       .EVEN
15175
15176  066600  101744  074626  000000  EMT1:   .WORD   EMS300,EMS1,0
15177  066606  101762  102005  102032  EMT2:   .WORD   EMS301,EMS302,EMS303,EMS1,EMS304
15178  066614  074626  102043
15179  066620  104750  104171  104331          .WORD   EMS511,EMS500,EMS501,EMS502,EMS503,0
15180  066626  104356  104425  000000
15181  066634  101762  102052  102005  EMT3:   .WORD   EMS301,EMS306,EMS302
15182  066642  104750  104475  104331          .WORD   EMS511,EMS505,EMS501,EMS502,0
15183  066650  104356  000000
15184  066654  101744  102005  102066  EMT4:   .WORD   EMS300,EMS302,EMS307,EMS2
15185  066662  074677
15186  066664  104750  104356  104331          .WORD   EMS511,EMS502,EMS501,EMS503,0
15187  066672  104425  000000
15188  066676  101762  102127  102144  EMT5:   .WORD   EMS301,EMS310,EMS311
15189  066704  104750  104356  104331          .WORD   EMS511,EMS502,EMS501,EMS503,EMS504
15190  066712  104425  104451
15191  066716  102210  000000                  .WORD   EMS312,0
15192  066722  101762  102052  102144  EMT6:   .WORD   EMS301,EMS306,EMS311
15193  066730  104750  104356  104331          .WORD   EMS511,EMS502,EMS501,EMS503,EMS504,0
15194  066736  104425  104451  000000
15195  066744  101762  102246  102005  EMT7:   .WORD   EMS301,EMS313,EMS302
15196  066752  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,EMS504,EMS503,0
15197  066760  104451  104425  000000
15198  066766  102354  102375  102277  EMT10:  .WORD   EMS316,EMS317,EMS314
15199  066774  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15200  067002  000000
15201  067004  102354  102375  102326  EMT11:  .WORD   EMS316,EMS317,EMS315
15202  067012  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15203  067020  000000
15204  067022  102354  102415  102277  EMT12:  .WORD   EMS316,EMS320,EMS314
15205  067030  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15206  067036  000000
15207  067040  102354  102415  102326  EMT13:  .WORD   EMS316,EMS320,EMS315
15208  067046  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15209  067054  000000
15210  067056  102354  102435  102277  EMT14:  .WORD   EMS316,EMS321,EMS314
15211  067064  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15212  067072  000000
15213  067074  102354  102435  102326  EMT15:  .WORD   EMS316,EMS321,EMS315
15214  067102  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15215  067110  000000
15216  067112  102354  102455  102277  EMT16:  .WORD   EMS316,EMS322,EMS314
15217  067120  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15218  067126  000000
15219  067130  102354  102455  102326  EMT17:  .WORD   EMS316,EMS322,EMS315
15220  067136  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15221  067144  000000
15222  067146  101762  102514  101232  EMT20:  .WORD   EMS301,EMS324,EMS250
15223  067154  104750  104451                  .WORD   EMS511,EMS504
15224  067160  102210  102545  000000          .WORD   EMS312,EMS326,0
15225  067166  101762  102475  101232  EMT21:  .WORD   EMS301,EMS323,EMS250
15226  067174  104750  104451                  .WORD   EMS511,EMS504
15227  067200  102210  102534  000000          .WORD   EMS312,EMS325,0
15228  067206  101762  102246  101232  EMT22:  .WORD   EMS301,EMS313,EMS250
15229  067214  104750  104451  000000          .WORD   EMS511,EMS504,0
```

```
15230  067222  101762  102514  101270  EMT23:  .WORD   EMS301,EMS324,EMS251
15231  067230  104750  104331          .WORD   EMS511,EMS501
15232  067234  102210  102545  000000  .WORD   EMS312,EMS326,0
15233  067242  101762  102475  101270  EMT24:  .WORD   EMS301,EMS323,EMS251
15234  067250  104750  104331          .WORD   EMS511,EMS501
15235  067254  102210  102534  000000  .WORD   EMS312,EMS325,0
15236  067262  101762  102246  101270  EMT25:  .WORD   EMS301,EMS313,EMS251
15237  067270  104750  104331  000000  .WORD   EMS511,EMS501,0
15238  067276  101762  102052  101334  EMT26:  .WORD   EMS301,EMS306,EMS252,EMS253,EMS327,EMS254
15239  067304  101367  102555  101422
15240  067312  104750  104425  104331  .WORD   EMS511,EMS503,EMS501,EMS502
15241  067320  104356
15242  067322  102563  102277  000000  .WORD   EMS330,EMS314,0
15243  067330  101744  101334  EMT27:  .WORD   EMS300,EMS252
15244  067334  104750  104331  000000  .WORD   EMS511,EMS501,0
15245  067342  101744  101334  EMT30:  .WORD   EMS300,EMS252
15246  067346  104750  104331  104451  .WORD   EMS511,EMS501,EMS504,0
15247  067354  000000
15248  067356  101744  101334  EMT31:  .WORD   EMS300,EMS252
15249  067362  104750  104331  104425  .WORD   EMS511,EMS501,EMS503,0
15250  067370  000000
15251  067372  101762  102514  101334  EMT32:  .WORD   EMS301,EMS324,EMS252
15252  067400  104750  104331  000000  .WORD   EMS511,EMS501,0
15253  067406  101762  102514  101334  EMT33:  .WORD   EMS301,EMS324,EMS252
15254  067414  104750  104331  104451  .WORD   EMS511,EMS501,EMS504,0
15255  067422  000000
15256  067424  101762  102514  101334  EMT34:  .WORD   EMS301,EMS324,EMS252
15257  067432  104750  104331  104425  .WORD   EMS511,EMS501,EMS503,0
15258  067440  000000
15259  067442  101762  102475  101334  EMT35:  .WORD   EMS301,EMS323,EMS252
15260  067450  104750  104331  000000  .WORD   EMS511,EMS501,0
15261  067456  101762  102246  101334  EMT36:  .WORD   EMS301,EMS313,EMS252
15262  067464  104750  104331  000000  .WORD   EMS511,EMS501,0
15263  067472  101762  102514  101463  EMT37:  .WORD   EMS301,EMS324,EMS255
15264  067500  102210  102545  000000  .WORD   EMS312,EMS326,0
15265  067506  101762  102475  101463  EMT40:  .WORD   EMS301,EMS323,EMS255
15266  067514  104750  104451          .WORD   EMS511,EMS504
15267  067520  102210  102534  000000  .WORD   EMS312,EMS325,0
15268  067526  101762  102246  101463  EMT41:  .WORD   EMS301,EMS313,EMS255
15269  067534  104750  104451  000000  .WORD   EMS511,EMS504,0
15270  067542  101762  102475  101232  EMT42:  .WORD   EMS301,EMS323,EMS250,EMS327,EMS255
15271  067550  102555  101463
15272  067554  104750  104451  104331  .WORD   EMS511,EMS504,EMS501,EMS503,0
15273  067562  104425  000000
15274  067566  101744  074746  EMT43:  .WORD   EMS300,EMS3
15275  067572  104750  104331  000000  .WORD   EMS511,EMS501,0
15276  067600  102602  075013  101367  EMT44:  .WORD   EMS331,EMS4,EMS253
15277  067606  104750  104331  000000  .WORD   EMS511,EMS501,0
15278
15279  067614  101744  101367  EMT45:  .WORD   EMS300,EMS253
15280  067620  104750  104331  104425  .WORD   EMS511,EMS501,EMS503,0
15281  067626  000000
15282
15283  067630  101744  101367  EMT46:  .WORD   EMS300,EMS253
15284  067634  104750  104331  104451  .WORD   EMS511,EMS501,EMS504,0
15285  067642  000000
```

```
15286
15287  067644  101762  102514  101367  EMT47:  .WORD   EMS301,EMS324,EMS253
15288  067652  104750  104331  104425          .WORD   EMS511,EMS501,EMS503
15289  067660  102210  102545  000000          .WORD   EMS312,EMS326,0
15290
15291  067666  101762  102514  101367  EMT50:  .WORD   EMS301,EMS324,EMS253
15292  067674  104750  104331  104451          .WORD   EMS511,EMS501,EMS504
15293  067702  102210  102545  000000          .WORD   EMS312,EMS326,0
15294
15295  067710  101762  102475  101367  EMT51:  .WORD   EMS301,EMS323,EMS253
15296  067716  104750  104331                  .WORD   EMS511,EMS501
15297  067722  102210  102534  000000          .WORD   EMS312,EMS325,0
15298
15299  067730  101762  102246  101367  EMT52:  .WORD   EMS301,EMS313,EMS253
15300  067736  104750  104331  000000          .WORD   EMS511,EMS501,0
15301
15302  067744  102602  075013  101525  EMT53:  .WORD   EMS331,EMS4,EMS256
15303  067752  104750  104331  000000          .WORD   EMS511,EMS501,0
15304
15305  067760  101762  102514  101525  EMT54:  .WORD   EMS301,EMS324,EMS256
15306  067766  104750  104331                  .WORD   EMS511,EMS501
15307  067772  102210  102545  000000          .WORD   EMS312,EMS326,0
15308
15309  070000  101762  102475  101525  EMT55:  .WORD   EMS301,EMS323,EMS256
15310  070006  104750  104331                  .WORD   EMS511,EMS501
15311  070012  102210  102534  000000          .WORD   EMS312,EMS325,0
15312
15313  070020  101762  102246  101525  EMT56:  .WORD   EMS301,EMS313,EMS256
15314  070026  104750  104331  000000          .WORD   EMS511,EMS501,0
15315
15316  070034  101555  102632          EMT57:  .WORD   EMS257,EMS332
15317  070040  104750  104550  104331          .WORD   EMS511,EMS506,EMS501,0
15318  070046  000000
15319
15320  070050  075044  102650          EMT60:  .WORD   EMS5,EMS333
15321  070054  104750  104605  104331          .WORD   EMS511,EMS507,EMS501,0
15322  070062  000000
15323
15324  070064  101762  102514  101611  EMT61:  .WORD   EMS301,EMS324,EMS260
15325  070072  104750  104331                  .WORD   EMS511,EMS501
15326  070076  102210  102545  000000          .WORD   EMS312,EMS326,0
15327
15328  070104  101762  102475  101611  EMT62:  .WORD   EMS301,EMS323,EMS260
15329  070112  104750  104331                  .WORD   EMS511,EMS501
15330  070116  102210  102534  000000          .WORD   EMS312,EMS325,0
15331
15332  070124  101762  102246  101611  EMT63:  .WORD   EMS301,EMS313,EMS260
15333  070132  104750  104331  000000          .WORD   EMS511,EMS501,0
15334
15335  070140  101744  075365          EMT64:  .WORD   EMS300,EMS12
15336  070144  104750  104331  000000          .WORD   EMS511,EMS501,0
15337
15338  070152  075365  102674  102772  EMT65:  .WORD   EMS12,EMS335,EMS342
15339  070160  104750  104331  000000          .WORD   EMS511,EMS501,0
15340
15341  070166  075365  102717  102772  EMT66:  .WORD   EMS12,EMS336,EMS342
```

```
15342  070174  104750  104331  000000          .WORD   EMS511,EMS501,0
15343
15344  070202  101744  075114          EMT67:  .WORD   EMS300,EMS6
15345  070206  104750  104331                  .WORD   EMS511,EMS501
15346  070212  075160  102650  000000          .WORD   EMS7,EMS333,0
15347
15348  070220  101744  075114  102555  EMT70:  .WORD   EMS300,EMS6,EMS327,EMS7
15349  070226  075160
15350  070230  104750  104331  104451          .WORD   EMS511,EMS501,EMS504,0
15351  070236  000000
15352
15353  070240  075114  102674  102752  EMT71:  .WORD   EMS6,EMS335,EMS340,EMS10,EMS333,EMS342
15354  070246  075231  102650  102772
15355  070254  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15356  070262  000000
15357
15358  070264  075114  102717  102752  EMT72:  .WORD   EMS6,EMS336,EMS340,EMS10,EMS334,EMS342
15359  070272  075231  102664  102772
15360  070300  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15361  070306  000000
15362
15363  070310  101762  101611  102032  EMT73:  .WORD   EMS301,EMS260,EMS303,EMS11
15364  070316  075274
15365  070320  104750  104331  104356          .WORD   EMS511,EMS501,EMS502,0
15366  070326  000000
15367
15368  070330  103024  103040  102772  EMT74:  .WORD   EMS343,EMS344,EMS342,0
15369  070336  000000
15370
15371  070340  101744  075443          EMT75:  .WORD   EMS300,EMS13
15372  070344  104750  104425  000000          .WORD   EMS511,EMS503,0
15373
15374  070352  103115  075443  103067  EMT76:  .WORD   EMS346,EMS13,EMS345
15375  070360  104750  104425  000000          .WORD   EMS511,EMS503,0
15376
15377  070366  102736  075443  103067  EMT77:  .WORD   EMS337,EMS13,EMS345
15378  070374  104750  104425  000000          .WORD   EMS511,EMS503,0
15379
15380  070402  101762  102246  101422  EMT100: .WORD   EMS301,EMS313,EMS254,EMS347,EMS13
15381  070410  103133  075443
15382  070414  104750  104425  000000          .WORD   EMS511,EMS503,0
15383
15384  070422  103115  075512  102763  EMT101: .WORD   EMS346,EMS14,EMS341,EMS15
15385  070430  075557
15386  070432  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15387  070440  000000
15388
15389  070442  102736  075512  102763  EMT102: .WORD   EMS337,EMS14,EMS341,EMS15
15390  070450  075557
15391  070452  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15392  070460  000000
15393
15394  070462  101762  102246  101422  EMT103: .WORD   EMS301,EMS313,EMS254,EMS347,EMS15
15395  070470  103133  075557
15396  070474  104750  104425                  .WORD   EMS511,EMS503
15397  070500  075512  102632  000000          .WORD   EMS14,EMS332,0
```

```
15398
15399  070506  103115  075716  102763  EMT104: .WORD   EMS346,EMS17,EMS341,EMS16
15400  070514  075635
15401  070516  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15402  070524  000000
15403
15404  070526  102736  075716  102763  EMT105: .WORD   EMS337,EMS17,EMS341,EMS16
15405  070534  075635
15406  070536  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15407  070544  000000
15408
15409  070546  101762  102246  101422  EMT106: .WORD   EMS301,EMS313,EMS254,EMS347,EMS16
15410  070554  103133  075635
15411  070560  104750  104425                  .WORD   EMS511,EMS503
15412  070564  075716  102632  000000          .WORD   EMS17,EMS332,0
15413
15414  070572  103115  075757  102763  EMT107: .WORD   EMS346,EMS20,EMS341,EMS21
15415  070600  076023
15416  070602  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15417  070610  000000
15418
15419  070612  075757  103161  076023  EMT110: .WORD   EMS20,EMS351,EMS21,EMS350,EMS22,EMS315
15420  070620  103154  076102  102326
15421  070626  104750  104331  000000          .WORD   EMS511,EMS501,0
15422
15423  070634  075757  102664  103154  EMT111: .WORD   EMS20,EMS334,EMS350,EMS22,EMS333
15424  070642  076102  102650
15425  070646  104750  104331  000000          .WORD   EMS511,EMS501,0
15426
15427  070654  102736  075757  102763  EMT112: .WORD   EMS337,EMS20,EMS341,EMS21
15428  070662  076023
15429  070664  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15430  070672  000000
15431
15432  070674  102736  075757  102763  EMT113: .WORD   EMS337,EMS20,EMS341,EMS21,EMS350,EMS22,EMS334
15433  070702  076023  103154  076102
15434  070710  102664
15435  070712  104750  104331  000000          .WORD   EMS511,EMS501,0
15436
15437  070720  075757  103177  076023  EMT114: .WORD   EMS20,EMS352,EMS21,EMS350,EMS22,EMS333
15438  070726  103154  076102  102650
15439  070734  104750  104331  000000          .WORD   EMS511,EMS501,0
15440
15441  070742  101762  102246  101422  EMT115: .WORD   EMS301,EMS313,EMS254,EMS347,EMS21
15442  070750  103133  076023
15443  070754  104750  104425                  .WORD   EMS511,EMS503
15444  070760  075757  102632  000000          .WORD   EMS20,EMS332,0
15445
15446  070766  103115  076147  102763  EMT116: .WORD   EMS346,EMS23,EMS341,EMS24
15447  070774  076225
15448  070776  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15449  071004  000000
15450
15451  071006  102736  076147  102763  EMT117: .WORD   EMS337,EMS23,EMS341,EMS24
15452  071014  076225
15453  071016  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
```

```
15454  071024  000000
15455
15456  071026  101762  102246  101422  EMT120: .WORD   EMS301,EMS313,EMS254,EMS347,EMS24
15457  071034  103133  076225
15458  071040  104750  104425                  .WORD   EMS511,EMS503
15459  071044  076147  102632  000000          .WORD   EMS23,EMS332,0
15460
15461  071052  103115  076304  102763  EMT121: .WORD   EMS346,EMS25,EMS341,EMS26
15462  071060  076362
15463  071062  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15464  071070  000000
15465
15466  071072  102736  076304  102763  EMT122: .WORD   EMS337,EMS25,EMS341,EMS26
15467  071100  076362
15468  071102  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15469  071110  000000
15470
15471  071112  101762  102246  101422  EMT123: .WORD   EMS301,EMS313,EMS254,EMS347,EMS26
15472  071120  103133  076362
15473  071124  104750  104425                  .WORD   EMS511,EMS503
15474  071130  076304  102632  000000          .WORD   EMS25,EMS332,0
15475
15476  071136  101744  076441  102066  EMT124: .WORD   EMS300,EMS27,EMS307,EMS2
15477  071144  074677
15478  071146  104750  104425  000000          .WORD   EMS511,EMS503,0
15479
15480  071154  102602  076441  103213  EMT125: .WORD   EMS331,EMS27,EMS353
15481  071162  104750  104425                  .WORD   EMS511,EMS503
15482  071166  076505  102326  000000          .WORD   EMS30,EMS315,0
15483
15484  071174  102736  076441  102763  EMT126: .WORD   EMS337,EMS27,EMS341,EMS30
15485  071202  076505
15486  071204  104750  104425  000000          .WORD   EMS511,EMS503,0
15487
15488  071212  101762  102246  101422  EMT127: .WORD   EMS301,EMS313,EMS254,EMS347,EMS30
15489  071220  103133  076505
15490  071224  104750  104425                  .WORD   EMS511,EMS503
15491  071230  076441  102632  000000          .WORD   EMS27,EMS332,0
15492
15493  071236  076565  103237  101232  EMT130: .WORD   EMS31,EMS354,EMS250
15494  071244  104750  104451  104425          .WORD   EMS511,EMS504,EMS503,0
15495  071252  000000
15496
15497  071254  076636  103237  101232  EMT131: .WORD   EMS32,EMS354,EMS250
15498  071262  104750  104451  104425          .WORD   EMS511,EMS504,EMS503,0
15499  071270  000000
15500
15501  071272  103272  076716  101232  EMT132: .WORD   EMS355,EMS33,EMS250,EMS341,EMS30
15502  071300  102763  076505
15503  071304  104750  104451  000000          .WORD   EMS511,EMS504,0
15504
15505  071312  103272  076755  101232  EMT133: .WORD   EMS355,EMS34,EMS250,EMS341,EMS30
15506  071320  102763  076505
15507  071324  104750  104451  000000          .WORD   EMS511,EMS504,0
15508
15509  071332  102602  075013  101463  EMT134: .WORD   EMS331,EMS4,EMS255
```

```
15510  071340  104750  104451  000000          .WORD   EMS511,EMS504,0
15511
15512  071346  077013  103334  103356  EMT135: .WORD   EMS35,EMS357,EMS360,EMS15
15513  071354  075557
15514  071356  104750  104331  000000          .WORD   EMS511,EMS501,0
15515
15516  071364  101642  103432          EMT136: .WORD   EMS261,EMS362
15517  071370  104750  104425  000000          .WORD   EMS511,EMS503,0
15518
15519  071376  101744  077055  102066  EMT137: .WORD   EMS300,EMS36,EMS307,EMS2
15520  071404  074677
15521  071406  104750  104331  000000          .WORD   EMS511,EMS501,0
15522
15523  071414  103272  077105  101463  EMT140: .WORD   EMS355,EMS37,EMS255,EMS341,EMS30
15524  071422  102763  076505
15525  071426  104750  104451  000000          .WORD   EMS511,EMS504,0
15526
15527  071434  103115  077145  103067  EMT141: .WORD   EMS346,EMS40,EMS345
15528  071442  104750  104451  000000          .WORD   EMS511,EMS504,0
15529
15530  071450  102736  077145  102763  EMT142: .WORD   EMS337,EMS40,EMS341,EMS30
15531  071456  076505
15532  071460  104750  104451  000000          .WORD   EMS511,EMS504,0
15533
15534  071466  103453  102127  077223  EMT143: .WORD   EMS363,EMS310,EMS41
15535  071474  104750  104331  000000          .WORD   EMS511,EMS501,0
15536
15537  071502  102736  077223  102763  EMT144: .WORD   EMS337,EMS41,EMS341,EMS252,EMS327,EMS253
15538  071510  101334  102555  101367
15539  071516  104750  104331  000000          .WORD   EMS511,EMS501,0
15540
15541  071524  077223  103237  103470  EMT145: .WORD   EMS41,EMS354,EMS364,EMS252,EMS365,EMS253
15542  071532  101334  103511  101367
15543  071540  104750  104331  000000          .WORD   EMS511,EMS501,0
15544
15545  071546  101762  102052  077055  EMT146: .WORD   EMS301,EMS306,EMS36
15546  071554  104750  104331  104425          .WORD   EMS511,EMS501,EMS503,0
15547  071562  000000
15548
15549  071564  103516  077271          EMT147: .WORD   EMS366,EMS42
15550  071570  104750  104425  000000          .WORD   EMS511,EMS503,0
15551
15552  071576  103541  103213  103511  EMT150: .WORD   EMS367,EMS353,EMS365,EMS42,EMS354,EMS3
15553  071604  077271  103237  074746
15554  071612  104750  104425  000000          .WORD   EMS511,EMS503,0
15555
15556  071620  102736  077055          EMT151: .WORD   EMS337,EMS36
15557  071624  104750  104331  000000          .WORD   EMS511,EMS501,0
15558
15559  071632  077353  103237  077055  EMT152: .WORD   EMS43,EMS354,EMS36
15560  071640  104750  104331  000000          .WORD   EMS511,EMS501,0
15561
15562  071646  103541  103213  103511  EMT153: .WORD   EMS367,EMS353,EMS365,EMS36,EMS370
15563  071654  077055  103611
15564  071660  104750  104425  000000          .WORD   EMS511,EMS503,0
15565
```

```
15566  071666  103541  103213  103511  EMT154: .WORD   EMS367,EMS353,EMS365,EMS36,EMS371
15567  071674  077055  103626
15568  071700  104750  104425  000000          .WORD   EMS511,EMS503,0
15569
15570  071706  101744  077424  102066  EMT155: .WORD   EMS300,EMS44,EMS307,EMS2
15571  071714  074677
15572  071716  104750  104425  000000          .WORD   EMS511,EMS503,0
15573
15574  071724  103541  103213  103511  EMT156: .WORD   EMS367,EMS353,EMS365,EMS44,EMS354,EMS3
15575  071732  077424  103237  074746
15576  071740  104750  104425  000000          .WORD   EMS511,EMS503,0
15577
15578  071746  101744  077465  102066  EMT157: .WORD   EMS300,EMS45,EMS307,EMS2
15579  071754  074677
15580  071756  104750  104425  000000          .WORD   EMS511,EMS503,0
15581
15582  071764  103541  103213  103511  EMT160: .WORD   EMS367,EMS353,EMS365,EMS45,EMS354,EMS3
15583  071772  077465  103237  074746
15584  072000  104750  104425  104331          .WORD   EMS511,EMS503,EMS501
15585  072006  077013  102650  000000          .WORD   EMS35,EMS333,0
15586
15587  072014  101744  077542  102066  EMT161: .WORD   EMS300,EMS46,EMS307,EMS2
15588  072022  074677
15589  072024  104750  104425  000000          .WORD   EMS511,EMS503,0
15590
15591  072032  102736  077542  103213  EMT162: .WORD   EMS337,EMS46,EMS353
15592  072040  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15593  072046  000000
15594
15595  072050  077013  102674  102736  EMT163: .WORD   EMS35,EMS335,EMS337,EMS41,EMS334,EMS372
15596  072056  077223  102664  103655
15597  072064  104750  104331  000000          .WORD   EMS511,EMS501,0
15598
15599  072072  077624  102674  102736  EMT164: .WORD   EMS47,EMS335,EMS337,EMS41,EMS335,EMS372
15600  072100  077223  102674  103655
15601  072106  104750  104331  000000          .WORD   EMS511,EMS501,0
15602
15603  072114  102736  077013  102555  EMT165: .WORD   EMS337,EMS35,EMS327,EMS47
15604  072122  077624
15605  072124  104750  104331                  .WORD   EMS511,EMS501
15606  072130  077223  102650  103655          .WORD   EMS41,EMS333,EMS372,0
15607  072136  000000
15608
15609  072140  101744  077624  102066  EMT166: .WORD   EMS300,EMS47,EMS307,EMS2
15610  072146  074677
15611  072150  104750  104331  104425          .WORD   EMS511,EMS501,EMS503,0
15612  072156  000000
15613
15614  072160  077664  102674  102752  EMT167: .WORD   EMS50,EMS335,EMS340,EMS36,EMS333
15615  072166  077055  102650
15616  072172  104750  104331  104425          .WORD   EMS511,EMS501,EMS503,0
15617  072200  000000
15618
15619  072202  102736  077013          EMT170: .WORD   EMS337,EMS35
15620  072206  104750  104331  000000          .WORD   EMS511,EMS501,0
15621
```

```
15622  072214  077664  076755  074746  EMT171: .WORD   EMS50,EMS34,EMS3
15623  072222  104750  104331  000000          .WORD   EMS511,EMS501,0
15624
15625  072230  101762  102052  077733  EMT172: .WORD   EMS301,EMS306,EMS51
15626  072236  104750  104331  104451          .WORD   EMS511,EMS501,EMS504,0
15627  072244  000000
15628
15629  072246  103541  103213  103511  EMT173: .WORD   EMS367,EMS353,EMS365,EMS47,EMS354,EMS3
15630  072254  077624  103237  074746
15631  072262  104750  104331  000000          .WORD   EMS511,EMS501,0
15632
15633  072270  101744  101232  102555  EMT174: .WORD   EMS300,EMS250,EMS327,EMS255,EMS327,EMS256
15634  072276  101463  102555  101525
15635  072304  102763  105047          .WORD   EMS341,EMS600
15636  072310  104750  104331  000000          .WORD   EMS511,EMS501,0
15637
15638  072316  101744  101525  102763  EMT175: .WORD   EMS300,EMS256,EMS341,EMS600
15639  072324  105047
15640  072326  104750  104331  000000          .WORD   EMS511,EMS501,0
15641
15642  072334  101744  101232  102763  EMT176: .WORD   EMS300,EMS250,EMS341,EMS600
15643  072342  105047
15644  072344  104750  104451  000000          .WORD   EMS511,EMS504,0
15645
15646  072352  101744  101463  102763  EMT177: .WORD   EMS300,EMS255,EMS341,EMS600
15647  072360  105047
15648  072362  104750  104451  000000          .WORD   EMS511,EMS504,0
15649
15650  072370  102736  100002  102763  EMT200: .WORD   EMS337,EMS52,EMS341,EMS601
15651  072376  105077
15652  072400  104750  104331  000000          .WORD   EMS511,EMS501,0
15653
15654  072406  103115  100002  102763  EMT201: .WORD   EMS346,EMS52,EMS341,EMS602
15655  072414  105117
15656  072416  104750  104331  000000          .WORD   EMS511,EMS501,0
15657
15658  072424  103115  077733  102763  EMT202: .WORD   EMS346,EMS51,EMS341,EMS602
15659  072432  105117
15660  072434  104750  104331  000000          .WORD   EMS511,EMS501,0
15661
15662  072442  103115  100002  103067  EMT203: .WORD   EMS346,EMS52,EMS345,EMS373,EMS255
15663  072450  103707  101463
15664  072454  104750  104451  104331          .WORD   EMS511,EMS504,EMS501,0
15665  072462  000000
15666
15667  072464  103115  100002  102763  EMT204: .WORD   EMS346,EMS52,EMS341,EMS27
15668  072472  076441
15669  072474  104750  104451  104331          .WORD   EMS511,EMS504,EMS501,0
15670  072502  000000
15671
15672  072504  100043  103237  074746  EMT205: .WORD   EMS53,EMS354,EMS3
15673  072512  104750  104425  104356          .WORD   EMS511,EMS503,EMS502,EMS510,0
15674  072520  104675  000000
15675
15676  072524  102736  100104  103713  EMT206: .WORD   EMS337,EMS54,EMS374,EMS250,EMS327,EMS255
15677  072532  101232  102555  101463
```

```
15678  072540  102555  074746                  .WORD   EMS327,EMS3
15679  072544  104750  104451  104331          .WORD   EMS511,EMS504,EMS501,0
15680  072552  000000
15681
15682  072554  100104  103237  074746  EMT207: .WORD   EMS54,EMS354,EMS3
15683  072562  104750  104331  000000          .WORD   EMS511,EMS501,0
15684
15685  072570  100104  103237  103470  EMT210: .WORD   EMS54,EMS354,EMS364,EMS250
15686  072576  101232
15687  072600  104750  104451  000000          .WORD   EMS511,EMS504,0
15688
15689  072606  100104  103237  103470  EMT211: .WORD   EMS54,EMS354,EMS364,EMS255
15690  072614  101463
15691  072616  104750  104451  000000          .WORD   EMS511,EMS504,0
15692
15693  072624  102736  100161          EMT212: .WORD   EMS337,EMS55
15694  072630  104750  104451  000000          .WORD   EMS511,EMS504,0
15695
15696  072636  100237  102664  103067  EMT213: .WORD   EMS56,EMS334,EMS345,EMS373,EMS262,EMS327,EMS251
15697  072644  103707  101701  102555
15698  072652  101270
15699  072654  104750  104331                  .WORD   EMS511,EMS501
15700  072660  100237  102674  000000          .WORD   EMS56,EMS335,0
15701
15702  072666  102736  100237          EMT214: .WORD   EMS337,EMS56
15703  072672  104750  104331  000000          .WORD   EMS511,EMS501,0
15704
15705  072700  100332  102650  103154  EMT215: .WORD   EMS57,EMS333,EMS350,EMS60,EMS334
15706  072706  100416  102664
15707  072712  104750  104331  000000          .WORD   EMS511,EMS501,0
15708
15709  072720  103453  102052  075044  EMT216: .WORD   EMS363,EMS306,EMS5
15710  072726  104750  104331  000000          .WORD   EMS511,EMS501,0
15711
15712  072734  103453  102052  100470  EMT217: .WORD   EMS363,EMS306,EMS61
15713  072742  104750  104331  000000          .WORD   EMS511,EMS501,0
15714
15715  072750  100543  102650  103744  EMT220: .WORD   EMS62,EMS333,EMS375,EMS251
15716  072756  101270
15717  072760  104750  104331  000000          .WORD   EMS511,EMS501,0
15718
15719  072766  100543  102650  103760  EMT221: .WORD   EMS62,EMS333,EMS376,EMS262
15720  072774  101701
15721  072776  104750  104331  000000          .WORD   EMS511,EMS501,0
15722
15723  073004  100543  102650  103760  EMT222: .WORD   EMS62,EMS333,EMS376,EMS250
15724  073012  101232
15725  073014  104750  104331  000000          .WORD   EMS511,EMS501,0
15726
15727  073022  103115  100543  102763  EMT223: .WORD   EMS346,EMS62,EMS341,EMS603
15728  073030  105160
15729  073032  104750  104331  000000          .WORD   EMS511,EMS501,0
15730
15731  073040  103115  100624  103760  EMT224: .WORD   EMS346,EMS63,EMS376,EMS262
15732  073046  101701
15733  073050  104750  104331  104425          .WORD   EMS511,EMS501,EMS503,0
```

```
15734  073056  000000
15735
15736  073060  100624  102650  103154  EMT225: .WORD   EMS63,EMS333,EMS350,EMS363,EMS310,EMS262
15737  073066  103453  102127  101701
15738  073074  104750  104331  000000          .WORD   EMS511,EMS501,0
15739
15740  073102  100624  103334  077055  EMT226: .WORD   EMS63,EMS357,EMS36,EMS372
15741  073110  103655
15742  073112  104750  104331  000000          .WORD   EMS511,EMS501,0
15743
15744  073120  100624  103314  103356  EMT227: .WORD   EMS63,EMS356,EMS360,EMS15
15745  073126  075557
15746  073130  104750  104331  000000          .WORD   EMS511,EMS501,0
15747
15748  073136  100624  103314  103404  EMT230: .WORD   EMS63,EMS356,EMS361,EMS15
15749  073144  075557
15750  073146  104750  104331  000000          .WORD   EMS511,EMS501,0
15751
15752  073154  100624  103314  077223  EMT231: .WORD   EMS63,EMS356,EMS41
15753  073162  104750  104331  000000          .WORD   EMS511,EMS501,0
15754
15755  073170  077223  103774  102772  EMT232: .WORD   EMS41,EMS377,EMS342,EMS365,EMS63,EMS332
15756  073176  103511  100624  102632
15757  073204  104750  104331  000000          .WORD   EMS511,EMS501,0
15758
15759  073212  103541  103213  103511  EMT233: .WORD   EMS367,EMS353,EMS365,EMS63,EMS401
15760  073220  100624  104022
15761  073224  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15762  073232  000000
15763
15764  073234  075635  103774  103655  EMT234: .WORD   EMS16,EMS377,EMS372,EMS365,EMS64,EMS354,EMS3
15765  073242  103511  100664  103237
15766  073250  074746
15767  073252  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15768  073260  000000
15769
15770  073262  101744  100734  102066  EMT235: .WORD   EMS300,EMS65,EMS307,EMS2
15771  073270  074677
15772  073272  104750  104356  104425          .WORD   EMS511,EMS502,EMS503,0
15773  073300  000000
15774
15775  073302  100237  103774  103760  EMT236: .WORD   EMS56,EMS377,EMS376,EMS252,EMS372,EMS350
15776  073310  101334  103655  103154
15777  073316  100734  104022                  .WORD   EMS65,EMS401
15778  073322  104750  104356  104425          .WORD   EMS511,EMS502,EMS503,EMS501,0
15779  073330  104331  000000
15780
15781  073334  101744  100775  102066  EMT237: .WORD   EMS300,EMS66,EMS307,EMS2
15782  073342  074677
```

```
15783  073344  104750  104331  104425          .WORD   EMS511,EMS501,EMS503,0
15784  073352  000000
15785
15786  073354  075557  104005  103655  EMT240: .WORD   EMS15,EMS400,EMS372,EMS350,EMS66,EMS401
15787  073362  103154  100775  104022
15788  073370  104750  104425  104331          .WORD   EMS511,EMS503,EMS501,0
15789  073376  000000
15790
15791  073400  100775  102717  102752  EMT241: .WORD   EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS604
15792  073406  075557  104136  104126
15793  073414  105201
15794  073416  104750  104425  000000          .WORD   EMS511,EMS503,0
15795
15796  073424  104046  105201  104037  EMT242: .WORD   EMS403,EMS604,EMS402,EMS21,EMS377
15797  073432  076023  103774
15798  073436  104750  104425                  .WORD   EMS511,EMS503
15799  073442  077055  102674  000000          .WORD   EMS36,EMS335,0
15800
15801  073450  100775  102717  102752  EMT243: .WORD   EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS604
15802  073456  076362  104107  104126
15803  073464  105201
15804  073466  104750  104425  000000          .WORD   EMS511,EMS503,0
15805
15806  073474  100624  104022  104126  EMT244: .WORD   EMS63,EMS401,EMS405,EMS604
15807  073502  105201
15808  073504  104750  104425  000000          .WORD   EMS511,EMS503,0
15809
15810  073512  077055  103611  104126  EMT245: .WORD   EMS36,EMS370,EMS405,EMS604
15811  073520  105201
15812  073522  104750  104425  000000          .WORD   EMS511,EMS503,0
15813
15814  073530  104046  105201  104037  EMT246: .WORD   EMS403,EMS604,EMS402,EMS24,EMS377
15815  073536  076225  103774
15816  073542  104750  104425                  .WORD   EMS511,EMS503
15817  073546  077055  102674  000000          .WORD   EMS36,EMS335,0
15818
15819  073554  101052  102632  104126  EMT247: .WORD   EMS67,EMS332,EMS405,EMS604
15820  073562  105201
15821  073564  104750  104425  000000          .WORD   EMS511,EMS503,0
15822
15823  073572  100775  102717  102752  EMT250: .WORD   EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS605
15824  073600  075557  104136  104126
15825  073606  105226
15826  073610  104750  104425  000000          .WORD   EMS511,EMS503,0
15827
15828  073616  104046  105226  104037  EMT251: .WORD   EMS403,EMS605,EMS402,EMS21,EMS377
15829  073624  076023  103774
15830  073630  104750  104425                  .WORD   EMS511,EMS503
15831  073634  077055  102674  000000          .WORD   EMS36,EMS335,0
15832
15833  073642  100775  102717  102752  EMT252: .WORD   EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS605
15834  073650  076362  104107  104126
15835  073656  105226
15836  073660  104750  104425  000000          .WORD   EMS511,EMS503,0
15837
15838  073666  100624  104022  104126  EMT253: .WORD   EMS63,EMS401,EMS405,EMS605
```

```
15839  073674  105226
15840  073676  104750  104425  000000          .WORD   EMS511,EMS503,0
15841
15842  073704  077055  103611  104126  EMT254: .WORD   EMS36,EMS370,EMS405,EMS605
15843  073712  105226
15844  073714  104750  104425  000000          .WORD   EMS511,EMS503,0
15845
15846  073722  104046  105226  104037  EMT255: .WORD   EMS403,EMS605,EMS402,EMS24,EMS377
15847  073730  076225  103774
15848  073734  104750  104425                  .WORD   EMS511,EMS503
15849  073740  077055  102674  000000          .WORD   EMS36,EMS335,0
15850
15851  073746  101052  102632  104126  EMT256: .WORD   EMS67,EMS332,EMS405,EMS605
15852  073754  105226
15853  073756  104750  104425  000000          .WORD   EMS511,EMS503,0
15854
15855  073764  100775  102717  102752  EMT257: .WORD   EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS606
15856  073772  075557  104136  104126
15857  074000  105244
15858  074002  104750  104425  000000          .WORD   EMS511,EMS503,0
15859
15860  074010  104046  105244  104037  EMT260: .WORD   EMS403,EMS606,EMS402,EMS21,EMS377
15861  074016  076023  103774
15862  074022  104750  104425                  .WORD   EMS511,EMS503
15863  074026  077055  102674  000000          .WORD   EMS36,EMS335,0
15864
15865  074034  100775  102717  102752  EMT261: .WORD   EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS606
15866  074042  076362  104107  104126
15867  074050  105244
15868  074052  104750  104425  000000          .WORD   EMS511,EMS503,0
15869
15870  074060  100624  104022  104126  EMT262: .WORD   EMS63,EMS401,EMS405,EMS606
15871  074066  105244
15872  074070  104750  104425  000000          .WORD   EMS511,EMS503,0
15873
15874  074076  077055  103611  104126  EMT263: .WORD   EMS36,EMS370,EMS405,EMS606
15875  074104  105244
15876  074106  104750  104425  000000          .WORD   EMS511,EMS503,0
15877
15878  074114  104046  105244  104037  EMT264: .WORD   EMS403,EMS606,EMS402,EMS24,EMS377
15879  074122  076225  103774
15880  074126  104750  104425                  .WORD   EMS511,EMS503
15881  074132  077055  102674  000000          .WORD   EMS36,EMS335,0
15882
15883  074140  101164  104022  104126  EMT265: .WORD   EMS70,EMS401,EMS405,EMS606
15884  074146  105244
15885  074150  104750  104425  000000          .WORD   EMS511,EMS503,0
15886
15887  074156  101052  102632  104126  EMT266: .WORD   EMS67,EMS332,EMS405,EMS606
15888  074164  105244
15889  074166  104750  104425  000000          .WORD   EMS511,EMS503,0
15890
15891  074174  100775  103314  104151  EMT267: .WORD   EMS66,EMS356,EMS407
15892  074202  104750  104425  000000          .WORD   EMS511,EMS503,0
15893
15894  074210  100775  102717  102752  EMT270: .WORD   EMS66,EMS336,EMS340,EMS15,EMS406,EMS405,EMS607
```

```
 15895  074216  075557  104136  104126
 15896  074224  105264
 15897  074226  104750  104425  000000          .WORD   EMS511,EMS503,0
 15898
 15899  074234  104046  105264  104037  EMT271: .WORD   EMS403,EMS607,EMS402,EMS21,EMS377
 15900  074242  076023  103774
 15901  074246  104750  104425                  .WORD   EMS511,EMS503
 15902  074252  076441  102717  000000          .WORD   EMS27,EMS336,0
 15903
 15904  074260  076441  103611  104126  EMT272: .WORD   EMS27,EMS370,EMS405,EMS607
 15905  074266  105264
 15906  074270  104750  104425  000000          .WORD   EMS511,EMS503,0
 15907
 15908  074276  076441  103626  104126  EMT273: .WORD   EMS27,EMS371,EMS405,EMS607
 15909  074304  105264
 15910  074306  104750  104425  000000          .WORD   EMS511,EMS503,0
 15911
 15912  074314  077055  104107  104126  EMT274: .WORD   EMS36,EMS404,EMS405,EMS607
 15913  074322  105264
 15914  074324  104750  104425  000000          .WORD   EMS511,EMS503,0
 15915
 15916  074332  100043  104022  104126  EMT275: .WORD   EMS53,EMS401,EMS405,EMS607
 15917  074340  105264
 15918  074342  104750  104425  104356          .WORD   EMS511,EMS503,EMS502,0
 15919  074350  000000
 15920
 15921  074352  101052  102632  104126  EMT276: .WORD   EMS67,EMS332,EMS405,EMS607
 15922  074360  105264
 15923  074362  104750  104425  000000          .WORD   EMS511,EMS503,0
 15924
 15925  074370  100775  102717  102752  EMT277: .WORD   EMS66,EMS336,EMS340,EMS26,EMS404,EMS405,EMS607
 15926  074376  076362  104107  104126
 15927  074404  105264
 15928  074406  104750  104425  000000          .WORD   EMS511,EMS503,0
 15929
 15930  074414  104046  105264  104037  EMT300: .WORD   EMS403,EMS607,EMS402,EMS24,EMS377
 15931  074422  076225  103774
 15932  074426  104750  104425                  .WORD   EMS511,EMS503
 15933  074432  076441  102717  000000          .WORD   EMS27,EMS336,0
 15934
 15935  074440  101164  104022  104126  EMT301: .WORD   EMS70,EMS401,EMS405,EMS607
 15936  074446  105264
 15937  074450  104750  104425  000000          .WORD   EMS511,EMS503,0
 15938
 15939  074456  105302  000000          EHT1:   .WORD   EH1,0
 15940  074462  105361  000000          EHT2:   .WORD   EH2,0
 15941  074466  105377  000000          EHT5:   .WORD   EH5,0
 15942  074472  105436  000000          EHT7:   .WORD   EH7,0
 15943  074476  105455  000000          EHT57:  .WORD   EH57,0
 15944  074502  105554  000000          EHT65:  .WORD   EH65,0
 15945  074506  105631  000000          EHT71:  .WORD   EH71,0
 15946  074512  105370  000000          EHT74:  .WORD   EH3,0
 15947  074516  105707  000000          EHT115: .WORD   EH115,0
 15948  074522  106005  000000          EHT130: .WORD   EH130,0
 15949  074526  106124  000000          EHT132: .WORD   EH132,0
 15950  074532  106223  000000          EHT145: .WORD   EH145,0
```

```
15951  074536  106341  000000          EHT150: .WORD   EH150,0
15952  074542  106440  000000          EHT213: .WORD   EH213,0
15953  074546  106536  000000          EHT220: .WORD   EH220,0
15954
15955  074552  106576                  EDT1:   .WORD   ED1
15956  074554  106606                  EDT2:   .WORD   ED2
15957  074556  106612                  EDT5:   .WORD   ED5
15958  074560  106620                  EDT57:  .WORD   ED57
15959  074562  106632                  EDT65:  .WORD   ED65
15960  074564  106642                  EDT71:  .WORD   ED71
15961  074566  106606                  EDT74:  .WORD   ED2
15962  074570  106652                  EDT115: .WORD   ED115
15963  074572  106664                  EDT130: .WORD   ED130
15964  074574  106652                  EDT132: .WORD   ED115
15965  074576  106700                  EDT220: .WORD   ED220
15966
15967  074600  106704                  EFT1:   .WORD   EF1
15968  074602  106707                  EFT2:   .WORD   EF2
15969  074604  106710                  EFT5:   .WORD   EF5
15970  074606  106712                  EFT57:  .WORD   EF57
15971  074610  106704                  EFT65:  .WORD   EF1
15972  074612  106704                  EFT71:  .WORD   EF1
15973  074614  106707                  EFT74:  .WORD   EF2
15974  074616  106712                  EFT115: .WORD   EF57
15975  074620  106716                  EFT130: .WORD   EF130
15976  074622  106712                  EFT132: .WORD   EF57
15977  074624  106710                  EFT220: .WORD   EF5
15978
15979                                  .NLIST  BEX
       074626  047516  042516  044530  EMS1:   .ASCIZ  @NONEXISTENT DEVICE 'NED' (RMCS2,BIT 12) @
       074677     103  047117  051124  EMS2:   .ASCIZ  @CONTROLLER CLEAR 'CLR' (RMCS2,BIT 05) @
       074746  052506  041516  044524  EMS3:   .ASCIZ  @FUNCTION CODE (RMCS1, BITS 01 - 05) @
       075013     125  052516  042523  EMS4:   .ASCIZ  @UNUSED BIT POSITIONS OF @
       075044  042504  044526  042503  EMS5:   .ASCIZ  @DEVICE AVAILABLE 'DVA' (RMCS1, BIT 11) @
       075114  040520  052122  054511  EMS6:   .ASCIZ  @PARTIY ERROR 'PAR' (RMER1, BIT 03) @
       075160  040504  040524  050040  EMS7:   .ASCIZ  @DATA PARITY ERROR 'DPE' (RMER2, BIT 03) @
       075231     120  051101  052111  EMS10:  .ASCIZ  @PARITY TEST 'PAT' (RMCS2, BIT 04) @
       075274  040515  051523  052502  EMS11:  .ASCII  @MASSBUS CONTROL BUS PARITY ERROR 'MCPE' @
       075344  051050  041515  030523          .ASCIZ  @(RMCS1, BIT 13) @
       075365     111  046114  043505  EMS12:  .ASCIZ  @ILLEGAL REGISTER ERROR 'ILR' (RMER1, BIT 01) @
       075443     104  040511  047107  EMS13:  .ASCIZ  @DIAGNOSTIC MODE 'DMD' (RMMR1, BIT 00) @
       075512  042515  044504  046525  EMS14:  .ASCIZ  @MEDIUM ON LINE 'MOL' (RMDS, BIT 12) @
       075557     115  044501  052116  EMS15:  .ASCIZ  @MAINTENANCE UNIT READY 'MUR' (RMMR1, BIT 09) @
       075635     115  044501  052116  EMS16:  .ASCIZ  @MAINTENANCE WRITE PROTECT 'MWP' (RMMR1, BIT 03) @
       075716  051127  052111  020105  EMS17:  .ASCIZ  @WRITE LOCK 'WRL' (RMDS, BIT 11) @
       075757     104  053105  041511  EMS20:  .ASCIZ  @DEVICE CHECK 'DVC' (RMER2, BIT 07) @
       076023     115  044501  052116  EMS21:  .ASCIZ  @MAINTENANCE DRIVE FAULT 'MDF' (RMMR1, BIT 06) @
       076102  047125  040523  042506  EMS22:  .ASCIZ  @UNSAFE STATUS 'UNS' (RMER1, BIT 14) @
       076147     123  042505  020113  EMS23:  .ASCIZ  @SEEK INCOMPLETE STATUS 'SKI' (RMER2, BIT 14) @
       076225     115  044501  052116  EMS24:  .ASCIZ  @MAINTENANCE SEEK ERROR 'MSER' (RMMR1, BIT 07) @
       076304  047520  044523  044524  EMS25:  .ASCIZ  @POSITIONING IN PROGRESS 'PIP' (RMDS, BIT 13) @
       076362  040515  047111  042524  EMS26:  .ASCIZ  @MAINTENANCE ON CYLINDER 'MOC' (RMMR1, BIT 08) @
       076441     105  042116  047440  EMS27:  .ASCIZ  @END OF BLOCK 'EBL' (RMMR1, BIT 13) @
       076505     104  040511  047107  EMS30:  .ASCIZ  @DIAGNOSTIC END OF BLOCK 'DEBL' (RMMR1, BIT 13) @
       076565     114  051501  020124  EMS31:  .ASCIZ  @LAST SECTOR STATUS 'LS' (RMMR1, BIT 02) @
       076636  040514  052123  051440  EMS32:  .ASCIZ  @LAST SECTOR/TRACK STATUS 'LST' (RMMR1, BIT 01) @
```

```
076716  042523  052103  051117  EMS33:  .ASCIZ  @SECTOR ADDRESS, BITS 00-04 OF @
076755     124  040522  045503  EMS34:  .ASCIZ  @TRACK ADDRESS, BITS 08-10 OF @
077013     126  046117  046525  EMS35:  .ASCIZ  @VOLUME VALID 'VV' (RMDS, BIT 06) @
077055     107  020117  044502  EMS36:  .ASCIZ  @GO BIT (RMCS1, BIT 00) @
077105     103  046131  047111  EMS37:  .ASCIZ  @CYLINDER ADDRESS, BIT 00-09 OF @
077145     114  051501  020124  EMS40:  .ASCIZ  @LAST BLOCK TAKEN STATUS 'LBT' (RMDS, BIT 10) @
077223     103  046517  047520  EMS41:  .ASCIZ  @COMPOSITE ERROR 'ERR' (RMDS, BIT 14) @
077271     103  046517  040515  EMS42:  .ASCIZ  @COMMAND SEQUENCER TEST BIT 'TST' (RMMR2, BIT 12) @
```

;

```
077353   104  044522  042526  EMS43:  .ASCIZ  @DRIVE READY STATUS 'DRY' (RMDS, BIT 07) @
077424 047503  052116  047111  EMS44:  .ASCIZ  @CONTINUE 'CONT' (RMMR1, BIT 06) @
077465   111  053116  046101  EMS45:  .ASCIZ  @INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) @
077542 047514  051523  047440  EMS46:  .ASCIZ  @LOSS OF SYSTEM CLOCK ERROR 'LSC' (RMER2, BIT 11) @
077624 041517  052503  044520  EMS47:  .ASCIZ  @OCCUPIED 'OCC' (RMMR1, BIT 15) @
077664 046111  042514  040507  EMS50:  .ASCIZ  @ILLEGAL FUNCTION 'ILF' (RMER1, BIT 0) @
077733   117  043106  042523  EMS51:  .ASCIZ  @OFFSET DIRECTION 'OFD' (RMOF, BIT 07) @
100002 043117  051506  052105  EMS52:  .ASCIZ  @OFFSET MODE 'OM' (RMDS, BIT 00) @
100043   122  047125  040440  EMS53:  .ASCIZ  @RUN AND GO 'RG' (RMMR1, BIT 14) @
100104 047111  040526  044514  EMS54:  .ASCIZ  @INVALID ADDRESS ERROR 'IAE' (RMER1, BIT 10) @
100161   101  042104  042522  EMS55:  .ASCIZ  @ADDRESS OVERFLOW ERROR 'AOE' (RMER1, BIT 09) @
100237   122  043505  051511  EMS56:  .ASCII  @REGISTER MODIFICATION REFUSED ERROR @
100303   042  046522  021122          .ASCIZ  @'RMR' (RMER1, BIT 02) @
100332 051104  053111  020105  EMS57:  .ASCIZ  @DRIVE REQUEST REQUIRED STATUS 'DRQ' (RMDT, BIT 11) @
100416 051120  043517  040522  EMS60:  .ASCIZ  @PROGRAMMABLE STATUS 'PGM' (RMDS, BIT 09) @
100470 051104  053111  020105  EMS61:  .ASCIZ  @DRIVE PRESENT STATUS 'DPR' (RMDS, BIT 08) @
100543   120  051117  020124  EMS62:  .ASCIZ  @PORT REQUEST FLOP 'RQA,RQB' (RMMR2, BITS 15,14) @
100624 052101  042524  052116  EMS63:  .ASCIZ  @ATTENTION 'ATA' (RMDS, BIT 15) @
100664 051127  052111  020105  EMS64:  .ASCIZ  @WRITE LOCK ERROR 'WLE' (RMER1, BIT 11) @
100734 054105  042503  052120  EMS65:  .ASCIZ  @EXCEPTION 'REX' (RMMR1, BIT 12) @
100775   111  053116  046101  EMS66:  .ASCIZ  @INVALID COMMAND ERROR 'IVC' (RMER2, BIT 12) @
101052 040524  020107  052502  EMS67:  .ASCIZ  @TAG BUS (RMMR2, BITS 00-09) OR TAG CONTROL @
101126 044514  042516  020123          .ASCIZ  @LINES (RMMR2, BITS 10,11,13) @
101164 042523  051101  044103  EMS70:  .ASCIZ  @SEARCH ENABLE 'ESRC' (RMMR1, BIT 11) @

101232 044504  045523  040440  EMS250: .ASCIZ  @DISK ADDRESS REGISTER (RMDA) @
101270 047503  052116  047522  EMS251: .ASCIZ  @CONTROL STATUS REGISTER #1 (RMCS1) @
101334 051105  047522  020122  EMS252: .ASCIZ  @ERROR REGISTER #1 (RMER1) @
101367   105  051122  051117  EMS253: .ASCIZ  @ERROR REGISTER #2 (RMER2) @
101422 040515  047111  042524  EMS254: .ASCIZ  @MAINTENANCE REGISTER #1 (RMMR1) @
101463   104  051505  051111  EMS255: .ASCIZ  @DESIRED CYLINDER REGISTER (RMDC) @
101525   117  043106  042523  EMS256: .ASCIZ  @OFFSET REGISTER (RMOF) @
101555   104  044522  042526  EMS257: .ASCIZ  @DRIVE TYPE REGISTER (RMDT) @
101611   110  046117  044504  EMS260: .ASCIZ  @HOLDING REGISTER (RMHR) @
101642 042523  044522  046101  EMS261: .ASCIZ  @SERIAL NUMBER REGISTER (RMSN) @
101701   101  052124  047105  EMS262: .ASCIZ  @ATTENTION SUMMARY REGISTER (RMAS) @

101744 040503  047116  052117  EMS300: .ASCIZ  @CANNOT CLEAR @
101762 040503  047116  052117  EMS301: .ASCIZ  @CANNOT WRITE/READ @
102005   101  054516  042040  EMS302: .ASCIZ  @ANY DEVICE REGISTER @
102032 044527  044124  052517  EMS303: .ASCIZ  @WITHOUT @
102043   105  051122  051117  EMS304: .ASCIZ  @ERROR @
102052 020101  047117  020105  EMS306: .ASCIZ  @A ONE FROM @
102066 051525  047111  020107  EMS307: .ASCIZ  @USING MASSBUS INITIALIZE, I.E., @
102127   101  055040  051105  EMS310: .ASCIZ  @A ZERO FROM @
102144 053105  051105  020131  EMS311: .ASCIZ  @EVERY DEVICE REGISTER BIT POSITION @
102210 044124  020105  047506  EMS312: .ASCIZ  @THE FOLLOWING BITS ARE STUCK @
102246 020101  044123  043111  EMS313: .ASCIZ  @A SHIFTING ONE BIT FROM @
102277   101  050120  040505  EMS314: .ASCIZ  @APPEARS STUCK AT ZERO @
102326 050101  042520  051101  EMS315: .ASCIZ  @APPEARS STUCK AT ONE @
102354 042522  044507  052123  EMS316: .ASCIZ  @REGISTER SELECT @
102375   061  024040  026061  EMS317: .ASCIZ  @1 (1,2,4,8,16) @
102415   062  024040  026061  EMS320: .ASCIZ  @2 (1,2,4,8,16) @
102435   064  024040  026061  EMS321: .ASCIZ  @4 (1,2,4,8,16) @
102455   070  024040  026061  EMS322: .ASCIZ  @8 (1,2,4,8,16) @
102475   101  046114  047440  EMS323: .ASCIZ  @ALL ONES FROM @
```

```
102514  046101  020114  042532  EMS324: .ASCIZ  @ALL ZEROS FROM @
102534  052101  055040  051105  EMS325: .ASCIZ  @AT ZERO @
102545     101  020124  047117  EMS326: .ASCIZ  @AT ONE @
102555     054  047440  020122  EMS327: .ASCIZ  @, OR @
102563     015  041412  020123  EMS330: .ASCIZ  <CR><LF>@CS MBA CLRL @
102602  040503  047116  052117  EMS331: .ASCIZ  @CANNOT READ ZEROS FROM @
102632  051511  044440  041516  EMS332: .ASCIZ  @IS INCORRECT @
102650  051511  047040  052117  EMS333: .ASCIZ  @IS NOT SET @
102664  051511  051440  052105  EMS334: .ASCIZ  @IS SET @
102674  044123  052517  042114  EMS335: .ASCIZ  @SHOULD NOT BE SET @
102717     123  047510  046125  EMS336: .ASCIZ  @SHOULD BE SET @
102736  040503  047116  052117  EMS337: .ASCIZ  @CANNOT SET @
102752  042502  040503  051525  EMS340: .ASCIZ  @BECAUSE @
102763     125  044523  043516  EMS341: .ASCIZ  @USING @
102772  052504  044522  043516  EMS342: .ASCIZ  @DURING REGISTER TRANSFER @
103024  047125  054105  042520  EMS343: .ASCIZ  @UNEXPECTED @
103040  052502  020123  044524  EMS344: .ASCIZ  @BUS TIMEOUT (04 TRAP) @
103067     102  020131  042522  EMS345: .ASCIZ  @BY REGISTER TRANSFER @
103115     103  047101  047516  EMS346: .ASCIZ  @CANNOT RESET @
103133     127  052111  047510  EMS347: .ASCIZ  @WITHOUT SETTING @
103154  052502  020124     000  EMS350: .ASCIZ  @BUT @
103161     127  051501  051040  EMS351: .ASCIZ  @WAS RESET BY @
103177     127  051501  051440  EMS352: .ASCIZ  @WAS SET BY @
103213     111  020116  044504  EMS353: .ASCIZ  @IN DIAGNOSTIC MODE @
103237     111  020123  047111  EMS354: .ASCIZ  @IS INCORRECT ACCORDING TO @
103272  040503  047116  052117  EMS355: .ASCIZ  @CANNOT INCREMENT @
103314  040527  020123  047516  EMS356: .ASCIZ  @WAS NOT SET BY @
103334  040527  020123  047516  EMS357: .ASCIZ  @WAS NOT RESET BY @
103356  020060  047524  030440  EMS360: .ASCIZ  @0 TO 1 TRANSITION OF @
103404  020061  047524  030040  EMS361: .ASCIZ  @1 TO 0 TRANSITION OF @
103432  051511  044440  041516  EMS362: .ASCIZ  @IS INCONSISTENT @
103453     103  047101  047516  EMS363: .ASCIZ  @CANNOT READ @
103470  042524  052123  050040  EMS364: .ASCIZ  @TEST PATTERN IN @
103511     101  042116  000040  EMS365: .ASCIZ  @AND @
103516  040503  047116  052117  EMS366: .ASCIZ  @CANNOT INITIALIZE @
103541     124  042510  041440  EMS367: .ASCIZ  @THE COMMAND SEQUENCER HAS BEEN CLOCKED @
```

```
103611   122  051505  052105  EMS370: .ASCIZ  @RESET EARLY @
103626  044504  020104  047516  EMS371: .ASCIZ  @DID NOT RESET ON TIME @
103655   104  051125  047111  EMS372: .ASCIZ  @DURING COMMAND EXECUTION @
103707   124  020117  000     EMS373: .ASCIZ  @TO @
103713   127  052111  020110  EMS374: .ASCIZ  @WITH ANY COMBINATION OF @
103744  054502  051040  040505  EMS375: .ASCIZ  @BY READING @
103760  054502  053440  044522  EMS376: .ASCIZ  @BY WRITING @
103774  040527  020123  042523  EMS377: .ASCIZ  @WAS SET @
104005   127  051501  047040  EMS400: .ASCIZ  @WAS NOT SET @
104022  044504  020104  047516  EMS401: .ASCIZ  @DID NOT SET @
104037   127  044510  042514  EMS402: .ASCIZ  @WHILE @
104046  047503  046515  047101  EMS403: .ASCIZ  @COMMAND SEQUENCER DID NOT ABORT @
104107   127  051501  047040  EMS404: .ASCIZ  @WAS NOT RESET @
104126  052504  044522  043516  EMS405: .ASCIZ  @DURING @
104136  040527  020123  042522  EMS406: .ASCIZ  @WAS RESET @
104151   123  040505  041522  EMS407: .ASCIZ  @SEARCH TIMEOUT @

104171   011  042504  044526  EMS500: .ASCII  @       DEVICE IS NONEXISTENT,@<CR><LF>
104222  042011  053105  041511          .ASCII  @       DEVICE IS SWITCHED TO OTHER PORT@<CR><LF>
104265   011  051124  047101          .ASCIZ  @       TRANSCEIVER ENABLE SWITCH IS OFF@<CR><LF>
104331   011  043111  046440  EMS501: .ASCIZ  @       IF MODULE, M7686,@<CR><LF>
104356  046411  051501  041123  EMS502: .ASCIZ  @       MASSBUS TRANSCEIVER,M5922 OR M5923      @<CR><LF>
104425   011  051503  046440  EMS503: .ASCIZ  @       CS MODULE,M7684,@<CR><LF>
104451   011  051504  046440  EMS504: .ASCIZ  @       DS MODULE,M7685,@<CR><LF>
104475   011  042504  044526  EMS505: .ASCIZ  @       DEVICE IS SWITCHED TO A/B PORT POSITION@<CR><LF>
104550  042011  053105  041511  EMS506: .ASCIZ  @       DEVICE IS NOT AN RM03, OR@<CR><LF>
104605   011  042504  044526  EMS507: .ASCIZ  @       DEVICE IS SWITCHED TO PROGRAMMABLE PORT POSITION, OR@<CR><LF>
104675   011  051501  052523  EMS510: .ASCIZ  @       ASSUMING THE RH CONTROLLER HAS NO FAULT@<CR><LF>
104750  005015  050011  047522  EMS511: .ASCII  <CR><LF>@       PROBABLE FAULT(S):@<CR><LF>
104777   011  047050  052117          .ASCIZ  @       (NOT INCLUDING CABLES OR CONNECTORS)@<CR><LF>

105047   122  040505  020104  EMS600: .ASCIZ  @READ IN PRESET COMMAND @
105077   117  043106  042523  EMS601: .ASCIZ  @OFFSET COMMAND @
105117   122  052105  051125  EMS602: .ASCIZ  @RETURN TO CENTER CENTER COMMAND @
105160  042522  042514  051501  EMS603: .ASCIZ  @RELEASE COMMAND @
105201   122  041505  046101  EMS604: .ASCIZ  @RECALIBRATE COMMAND @
105226  042523  045505  041440  EMS605: .ASCIZ  @SEEK COMMAND @
105244  042523  051101  044103  EMS606: .ASCIZ  @SEARCH COMMAND @
105264  040504  040524  041440  EMS607: .ASCIZ  @DATA COMMAND @

105302  054105  041520  042124  EH1:    .ASCII  @EXPCTD  RECEVD  REGSTR@<CR><LF>
105332  052123  052101  051525          .ASCIZ  @STATUS  STATUS  ADRESS@
105361   040  040502  042523  EH2:    .ASCII  @ BASE@<CR><LF>
105370  042101  042522  051523  EH3:    .ASCIZ  @ADRESS@
105377   105  050130  052103  EH5:    .ASCII  @EXPCTD   STUCK@<CR><LF>
105417   122  051505  046125          .ASCIZ  @RESULT  BIT(S)@
105436  054105  041520  042124  EH7:    .ASCIZ  @EXPCTD  RECEVD@
105455   123  043516  051120  EH57:   .ASCII  @SNGPRT  DULPRT  RECEVD  DRVTYP@<CR><LF>
105515   104  053122  054524          .ASCIZ  @DRVTYP  DRVTYP  DRVTYP  REGADR@
105554  054105  041520  042124  EH65:   .ASCII  @EXPCTD  RECEVD  TEST@<CR><LF>
105602  052123  052101  051525          .ASCIZ  @STATUS  STATUS  REGSTR@
105631   105  050130  052103  EH71:   .ASCII  @EXPCTD  RECEVD   TEST@<CR><LF>
105660  052123  052101  051525          .ASCIZ  @STATUS  STATUS  PATTRN@
105707   105  050130  052103  EH115:  .ASCII  @EXPCTD  RECEVD  REGSTR   TEST@<CR><LF>
105746  052123  052101  051525          .ASCIZ  @STATUS  STATUS  ADRESS  PATTRN@
106005   105  050130  052103  EH130:  .ASCII  @EXPCTD  RECEVD  REGSTR   TEST   OFFSET@<CR><LF>
```

```
        106055     123  040524  052524             .ASCIZ  @STATUS  STATUS  ADRESS  PATTRN  REGSTR@
        106124  054105  041520  042124  EH132:     .ASCII  @EXPCTD  ACTUAL  REGSTR  OFFSET@<CR><LF>
        106164  047503  047125  020124             .ASCIZ  @COUNT   COUNT   ADRESS  REGSTR@
        106223     105  050130  052103  EH145:     .ASCII  @EXPCTD  ACTUAL  REGSTR  RMER1   RMER2@<CR><LF>
        106272  046503  042520  051122             .ASCIZ  @CMPERR  CMPERR  ADRESS  PATTRN  PATTRN@

        106341     105  050130  052103  EH150:     .ASCII  @EXPCTD  ACTUAL  REGSTR FUNCTION@<CR><LF>
        106402  042522  052523  052114             .ASCIZ  @RESULT  RESULT  ADRESS   CODE@
        106440  054105  041520  042124  EH213:     .ASCII  @EXPCTD  ACTUAL  STATUS   TEST@<CR><LF>
        106477     122  051505  046125             .ASCIZ  @RESULT  RESULT  ADRESS  REGSTR@
        106536  041501  052524  046101  EH220:     .ASCII  @ACTUAL  REGSTR@<CR><LF>
        106556  042522  052523  052114             .ASCIZ  @RESULT  ADRESS@
                                             .LIST BEX
15980           106576                       .EVEN
15981   106576  001140  001142  001136  ED1:    .WORD   $GDDAT,$BDDAT,$BDADR,0
15982   106604  000000
15983   106606  001136  000000           ED2:    .WORD   $BDADR,0
15984   106612  001140  001142  000000   ED5:    .WORD   $GDDAT,$BDDAT,0
15985   106620  001174  001176  001142   ED57:   .WORD   $TMP0,$TMP1,$BDDAT,$BDADR,0
15986   106626  001136  000000
15987   106632  001140  001142  001174   ED65:   .WORD   $GDDAT,$BDDAT,$TMP0,0
15988   106640  000000
15989   106642  001140  001142  001440   ED71:   .WORD   $GDDAT,$BDDAT,RMHR0,0
15990   106650  000000
15991   106652  001140  001142  001136   ED115:  .WORD   $GDDAT,$BDDAT,$BDADR,$TMP0,0
15992   106660  001174  000000
15993   106664  001140  001142  001136   ED130:  .WORD   $GDDAT,$BDDAT,$BDADR,$TMP0,$TMP1,0
15994   106672  001174  001176  000000
15995   106700  001142  001136           ED220:  .WORD   $BDDAT,$BDADR
15996
15997   106704     000      000      000  EF1:    .BYTE   0,0,0
15998   106707     000                   EF2:    .BYTE   0
15999   106710     000      000          EF5:    .BYTE   0,0
16000   106712     000      000      000  EF57:   .BYTE   0,0,0,0
16001   106715     000
16002   106716     000      000      000  EF130:  .BYTE   0,0,0,0,0
16003   106721     000      000
16004
16005           106724                       .EVEN
16006   106724                               BUFFER:
16007   106724  000402                       BUFONE: .BLKW   258.
16008   107730  000402                       BUFTWO: .BLKW   258.
16009           106724                       .       =       BUFFER
16010                                        .NLIST BEX
        106724                               HELP:
        106724  005015                       .ASCII  <CR><LF>
        106726  044514  052123  047440       .ASCII  @LIST OF TESTS@<CR><LF>
        106745     055  026455  026455       .ASCII  @-------------@<CR><LF>
        106764  005015                       .ASCII  <CR><LF>
        106766  030524  052011  040522       .ASCII  @T1     TRANSFER TEST@<CR><LF>
        107010  031124  041411  047524       .ASCII  @T2     CTOD TEST@<CR><LF>
        107026  031524  046411  051501       .ASCII  @T3     MASSBUS INITIALIZE TEST@<CR><LF>
        107062  032124  041411  042514       .ASCII  @T4     CLEAR STUCK ACTIVE TEST@<CR><LF>
        107116  032524  052011  044522       .ASCII  @T5     TRISTATE TRANSFER TEST@<CR><LF>
        107151     124  004466  042522       .ASCII  @T6     REGISTER SELECT TEST@<CR><LF>
        107202  033524  042011  044522       .ASCII  @T7     DRIVE TYPE TEST@<CR><LF>
```

```
107226  030524  004460  042504  .ASCII  aT10    DEVICE AVAILABLE TEST@<CR><LF>
107261     124  030461  044011  .ASCII  aT11    HOLDING REGISTER TRANSFER TEST@<CR><LF>
107325     124  031061  041411  .ASCII  aT12    CONTROL STATUS #1 TRANSFER TEST@<CR><LF>
107372  030524  004463  051105  .ASCII  aT13    ERROR REGISTER 1 TRANSFER TEST@<CR><LF>
107436  030524  004464  051105  .ASCII  aT14    ERROR REGISTER 2 TRANSFER TEST@<CR><LF>
107502  030524  004465  046103  .ASCII  aT15    CLEAR OFFSET STUCK ACTIVE TEST@<CR><LF>
107546  030524  004466  043117  .ASCII  aT16    OFFSET REGISTER TRANSFER TEST@<CR><LF>
107611     124  033461  051411  .ASCII  aT17    SERIAL NUMBER TEST@<CR><LF>
107641     124  030062  041411  .ASCII  aT20    CONTROL BUS PARITY DETECTION TEST@<CR><LF>
107710  031124  004461  047503  .ASCII  aT21    CONTROL BUS PARITY GENERATION TEST@<CR><LF>
107760  031124  004462  046522  .ASCII  aT22    RMDA,RMDC FAULT TEST@<CR><LF>
110012  031124  004463  044504  .ASCII  aT23    DISK ADDRESS TRANSFER TEST@<CR><LF>
110052  031124  004464  042504  .ASCII  aT24    DESIRED CYLINDER TRANSFER TEST@<CR><LF>
110116  031124  004465  046111  .ASCII  aT25    ILLEGAL REGISTER TEST@<CR><LF>
110151     124  033062  051011  .ASCII  aT26    RESET GO BY INIT TEST@<CR><LF>
110204  031124  004467  044504  .ASCII  aT27    DIAGNOSTIC MODE TEST@<CR><LF>
110236  031524  004460  047515  .ASCII  aT30    MOL TEST@<CR><LF>
110254  031524  004461  051127  .ASCII  aT31    WRITE LOCK TEST@<CR><LF>
110301     124  031063  042011  .ASCII  aT32    DRIVE FAULT TEST@<CR><LF>
110327     124  031463  051411  .ASCII  aT33    SEEK ERROR TEST@<CR><LF>
110354  031524  004464  044520  .ASCII  aT34    PIP TEST@<CR><LF>
110372  031524  004465  041105  .ASCII  aT35    EBL TEST@<CR><LF>
110410  031524  004466  040514  .ASCII  aT36    LAST SECTOR, LAST TRACK TEST@<CR><LF>
110452  031524  004467  046522  .ASCII  aT37    RMDA COUNT TEST@<CR><LF>
110477     124  030064  051011  .ASCII  aT40    RMDC COUNT TEST@<CR><LF>
110524  032124  004461  041114  .ASCII  aT41    LBT TEST@<CR><LF>
110542  032124  004462  047503  .ASCII  aT42    COMPOSITE ERROR TEST@<CR><LF>
110574  032124  004463  051127  .ASCII  aT43    WRITE GO TEST@<CR><LF>
110617     124  032064  041011  .ASCII  aT44    BRANCH MULTIPLEXOR TEST@<CR><LF>
110654  032124  004465  042523  .ASCII  aT45    SET/RESET GO TEST@<CR><LF>
110703     124  033064  042411  .ASCII  aT46    END 1 RESET GO TEST@<CR><LF>
110734  032124  004467  042523  .ASCII  aT47    SET PULSE TEST@<CR><LF>
110760  032524  004460  042523  .ASCII  aT50    SET/RESET IVC TEST@<CR><LF>
111010  032524  004461  042523  .ASCII  aT51    SET LSC TEST@<CR><LF>
111032  032524  004462  042504  .ASCII  aT52    DECODE TEST@<CR><LF>
111053     124  031465  051411  .ASCII  aT53    SET/RESET VOLUME VALID TEST@<CR><LF>
111114  032524  004464  046111  .ASCII  aT54    ILLEGAL FUNCTION TEST@<CR><LF>
111147     124  032465  047411  .ASCII  aT55    OCCUPIED TEST@<CR><LF>
111172  032524  004466  042522  .ASCII  aT56    READ IN PRESET TEST@<CR><LF>
111223     124  033465  051011  .ASCII  aT57    RIP/RMOF TEST@<CR><LF>
111246  033124  004460  046522  .ASCII  aT60    RMDA/RMDC/RIP TEST@<CR><LF>
111276  033124  004461  043117  .ASCII  aT61    OFFSET COMMAND TEST@<CR><LF>
111327     124  031066  051011  .ASCII  aT62    RETURN TO CENTER TEST@<CR><LF>
111362  033124  004463  046522  .ASCII  aT63    RMDC CLEAR OFFSET TEST@<CR><LF>
111416  033124  004464  041105  .ASCII  aT64    EBL CLEAR OFFSET TEST@<CR><LF>
111451     124  032466  051011  .ASCII  aT65    RUN AND GO TEST@<CR><LF>
111476  033124  004466  042523  .ASCII  aT66    SET IAE TEST@<CR><LF>
111520  033124  004467  042523  .ASCII  aT67    SEARCH, SEEK, READ WRITE TEST@<CR><LF>
111563     124  030067  044411  .ASCII  aT70    INVALID SECTOR/TRACK TEST@<CR><LF>
111622  033524  004461  047111  .ASCII  aT71    INVALID CYLINDER TEST@<CR><LF>
111655     124  031067  051411  .ASCII  aT72    SET AOE TEST@<CR><LF>
111677     124  031467  051411  .ASCII  aT73    SET RMR TEST@<CR><LF>
111721     124  032067  050011  .ASCII  aT74    PGM STATUS CHECK@<CR><LF>
111747     124  032467  042011  .ASCII  aT75    DPR STATUS CHECK@<CR><LF>
111775     124  033067  050011  .ASCII  aT76    PORT REQUEST TEST, PART 1@<CR><LF>
112034  033524  004467  047520  .ASCII  aT77    PORT REQUEST TEST, PART 2@<CR><LF>
```

```
112073    124  030061  004460  .ASCII  @T100   PORT REQUEST TEST, PART 3@<CR><LF>
112133    124  030061  004461  .ASCII  @T101   RELEASE TEST@<CR><LF>
112156 030524  031060  053411  .ASCII  @T102   WRITE ATA TEST@<CR><LF>
112203    124  030061  004463  .ASCII  @T103   RESET ATA BY GO TEST@<CR><LF>
112236 030524  032060  052411  .ASCII  @T104   UNIT READY ATA TEST@<CR><LF>
112270 030524  032460  042411  .ASCII  @T105   ERROR ATA TEST@<CR><LF>
112315    124  030061  004466  .ASCII  @T106   REGISTER TRANSFER ATA TEST@<CR><LF>
112356 030524  033460  050011  .ASCII  @T107   P SET ATA TEST@<CR><LF>
112403    124  030461  004460  .ASCII  @T110   SET WLE TEST@<CR><LF>
112426 030524  030461  042411  .ASCII  @T111   EXCEPTION TEST@<CR><LF>
112453    124  030461  004462  .ASCII  @T112   RECALIBRATE TEST@<CR><LF>
112502 030524  031461  051411  .ASCII  @T113   SEEK TEST@<CR><LF>
112522 030524  032061  051411  .ASCII  @T114   SEARCH TEST@<CR><LF>
112544 030524  032461  051411  .ASCII  @T115   SEARCH TIMEOUT TEST@<CR><LF>
112576 030524  033061  042011  .ASCII  @T116   DATA COMMAND TESTS (1)@<CR><LF>
112633    124  030461  004467  .ASCII  @T117   DATA COMMAND TESTS (2)@<CR><LF>
112670 030524  030062  042011  .ASCII  @T120   DATA COMMAND TESTS(3)@<CR><LF>
112724 005015          .ASCII  <CR><LF>
112726 005015          .ASCII  <CR><LF>
112730 053523  052111  044103  .ASCII  @SWITCH          USE@<CR><LF>
112746 026455  026455  026455  .ASCII  @------          ------------------@<CR><LF>
113004 020040  032461  004411  .ASCII  @  15           HALT ON ERROR@<CR><LF>
113031    040  030440  004464  .ASCII  @  14           LOOP ON TEST@<CR><LF>
113055    040  030440  004463  .ASCII  @  13           INHIBIT ERROR TYPEOUTS@<CR><LF>
113113    040  030440  004461  .ASCII  @  11           INHIBIT ITERATIONS@<CR><LF>
113145    040  030440  004460  .ASCII  @  10           BELL ON ERROR@<CR><LF>
113172 020040  034440  004411  .ASCII  @   9           LOOP ON ERROR@<CR><LF>
113217    040  020040  004470  .ASCII  @   8           LOOP ON TEST IN SWR<7:0>@<CR><LF>
113257    040  020040  004467  .ASCII  @   7           TN128@<CR><LF>
113274 020040  033040  004411  .ASCII' @   6           TN64@<CR><LF>
113310 020040  032440  004411  .ASCII  @   5           TN32@<CR><LF>
113324 020040  032040  004411  .ASCII  @   4           TN16@<CR><LF>
113340 020040  031440  004411  .ASCII  @   3           TN8@<CR><LF>
113353    040  020040  004462  .ASCII  @   2           TN4@<CR><LF>
113366 020040  030440  004411  .ASCII  @   1           TN2@<CR><LF>
113401    040  020040  004460  .ASCII  @   0           TN1@<CR><LF>
113414 005015     000          .ASCIZ  <CR><LF>
                               .LIST BEX
16011          000001          .END
```

```
ABASE = 176700
ACDW1 = 000000
ACDW2 = 000000
ACPUOP= 000000
ADDW0 = 000000
ADDW1 = 000000
ADDW10= 000000
ADDW11= 000000
ADDW12= 000000
ADDW13= 000000
ADDW14= 000000
ADDW15= 000000
ADDW2 = 000000
ADDW3 = 000000
ADDW4 = 000000
ADDW5 = 000000
ADDW6 = 000000
ADDW7 = 000000
ADDW8 = 000000
ADDW9 = 000000
ADEVCT= 000000
ADEVM = 000000
AENV  = 000000
AENVM = 000000
AFATAL= 000000
AMADR1= 000000
AMADR2= 000000
AMADR3= 000000
AMADR4= 000000
AMAMS1= 000000
AMAMS2= 000000
AMAMS3= 000000
AMAMS4= 000000
AMSGAD= 000000
AMSGLG= 000000
AMSGTY= 000000
AMTYP1= 000000
AMTYP2= 000000
AMTYP3= 000000
AMTYP4= 000000
AOE   = 001000
APASS = 000000
APE   = 100000
APRIOR= 000000
APTCSU= 000040
APTENV= 000001
APTSIZ= 000200
APTSPO= 000100
ASWREG= 000000
ATA   = 100000
ATESTN= 000000
ATNMSK= 000377
ATNTBL  066142

AUNIT = 000000
AUSWR = 000000
AVECT1= 120254
AVECT2= 000000
A16   = 000400
A17   = 001000
BAI   = 000010
BB00  = 000001
BB01  = 000002
BB02  = 000004
BB03  = 000010
BB04  = 000020
BB05  = 000040
BB06  = 000100
BB07  = 000200
BB08  = 000400
BB09  = 001000
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BOTADR  060104
BOTFLG  060106
BPTVEC= 000014
BSE   = 100000
BUFFER  106724
BUFONE  106724
BUFTWO  107730
CC    = 004000
CH    = 002000
CHRCNT  060107

CKSWR = 104410
CLOCK   007526
CLR   = 000040
CLSPRN  065160
CMNSTA  006166
CNSL00  065324
CNSL01  065363
CNSL02  065410
CNSL03  065471
CNSL04  065521
CNSL05  065575
CNSL06  065631
CNSL07  065656
CONT  = 000100
CR    = 000015
CRLF  = 000200
CYLMSK= 001777
DBCK  = 100000
DBEN  = 040000
DBL   = 002000
DCK   = 100000
DDISP = 177570
DEBL  = 020000
DISPLA  001156
DISPRE  000174
DLT   = 100000
DMD   = 000001
DPE   = 000010
DPEHI = 040000
DPELO = 020000
DPR   = 000400
DRQ   = 004000
DRVCLR= 000010
DRY   = 000200
DSWR  = 177570
DTE   = 010000
DTO   = 010000
DULPRT= 024024
DVA   = 004000
DVC   = 000200
EBL   = 020000
ECH   = 000100
ECI   = 004000
ECRC  = 001000
EDT1    074552
EDT115  074570
EDT130  074572
EDT132  074574
EDT2    074554
EDT220  074576
EDT5    074556
EDT57   074560
EDT65   074562

EDT71   074564
EDT74   074566
ED1     106576
ED115   106652
ED130   106664
ED2     106606
ED220   106700
ED5     106612
ED57    106620
ED65    106632
ED71    106642
EECC  = 000020
EFT1    074600
EFT115  074616
EFT130  074620
EFT132  074622
EFT2    074602
EFT220  074624
EFT5    074604
EFT57   074606
EFT65   074610
EFT71   074612
EFT74   074614
EF1     106704
EF130   106716
EF2     106707
EF5     106710
EF57    106712
EHT1    074456
EHT115  074516
EHT130  074522
EHT132  074526
EHT145  074532
EHT150  074536
EHT2    074462
EHT213  074542
EHT220  074546
EHT5    074466
EHT57   074476
EHT65   074502
EHT7    074472
EHT71   074506
EHT74   074512
EH1     105302
EH115   105707
EH130   106005
EH132   106124
EH145   106223
EH150   106341
EH2     105361
EH213   106440
EH220   106536
EH3     105370

EH5     105377
EH57    105455
EH65    105554
EH7     105436
EH71    105631
EMS1    074626
EMS10   075231
EMS11   075274
EMS12   075365
EMS13   075443
EMS14   075512
EMS15   075557
EMS16   075635
EMS17   075716
EMS2    074677
EMS20   075757
EMS21   076023
EMS22   076102
EMS23   076147
EMS24   076225
EMS25   076304
EMS250  101232
EMS251  101270
EMS252  101334
EMS253  101367
EMS254  101422
EMS255  101463
EMS256  101525
EMS257  101555
EMS26   076362
EMS260  101611
EMS261  101642
EMS262  101701
EMS27   076441
EMS3    074746
EMS30   076505
EMS300  101744
EMS301  101762
EMS302  102005
EMS303  102032
EMS304  102043
EMS306  102052
EMS307  102066
EMS31   076565
EMS310  102127
EMS311  102144
EMS312  102210
EMS313  102246
EMS314  102277
EMS315  102326
EMS316  102354
EMS317  102375
EMS32   076636
```

```
EMS320  102415
EMS321  102435
EMS322  102455
EMS323  102475
EMS324  102514
EMS325  102534
EMS326  102545
EMS327  102555
EMS33   076716
EMS330  102563
EMS331  102602
EMS332  102632
EMS333  102650
EMS334  102664
EMS335  102674
EMS336  102717
EMS337  102736
EMS34   076755
EMS340  102752
EMS341  102763
EMS342  102772
EMS343  103024
EMS344  103040
EMS345  103067
EMS346  103115
EMS347  103133
EMS35   077013
EMS350  103154
EMS351  103161
EMS352  103177
EMS353  103213
EMS354  103237
EMS355  103272
EMS356  103314
EMS357  103334
EMS36   077055
EMS360  103356
EMS361  103404
EMS362  103432
EMS363  103453
EMS364  103470
EMS365  103511
EMS366  103516
EMS367  103541
EMS37   077105
EMS370  103611
EMS371  103626
EMS372  103655
EMS373  103707
EMS374  103713
EMS375  103744
EMS376  103760
EMS377  103774
EMS4    075013
EMS40   077145
EMS400  104005
EMS401  104022
EMS402  104037
EMS403  104046
EMS404  104107
EMS405  104126
EMS406  104136
EMS407  104151
EMS41   077223
EMS42   077271
EMS43   077353
EMS44   077424
EMS45   077465
EMS46   077542
EMS47   077624
EMS5    075044
EMS50   077664
EMS500  104171
EMS501  104331
EMS502  104356
EMS503  104425
EMS504  104451
EMS505  104475
EMS506  104550
EMS507  104605
EMS51   077733
EMS510  104675
EMS511  104750
EMS52   100002
EMS53   100043
EMS54   100104
EMS55   100161
EMS56   100237
EMS57   100332
EMS6    075114
EMS60   100416
EMS600  105047
EMS601  105077
EMS602  105117
EMS603  105160
EMS604  105201
EMS605  105226
EMS606  105244
EMS607  105264
EMS61   100470
EMS62   100543
EMS63   100624
EMS64   100664
EMS65   100734
EMS66   100775
EMS67   101052
EMS7    075160
EMS70   10:164
EMTVEC= 000030
EMT1    066600
EMT10   066766
EMT100  070402
EMT101  070422
EMT102  070442
EMT103  070462
EMT104  070506
EMT105  070526
EMT106  070546
EMT107  070572
EMT11   067004
EMT110  070612
EMT111  070634
EMT112  070654
EMT113  070674
EMT114  070720
EMT115  070742
EMT116  070766
EMT117  071006
EMT12   067022
EMT120  071026
EMT121  071052
EMT122  071072
EMT123  071112
EMT124  071136
EMT125  071154
EMT126  071174
EMT127  071212
EMT13   067040
EMT130  071236
EMT131  071254
EMT132  071272
EMT133  071312
EMT134  071332
EMT135  071346
EMT136  071364
EMT137  071376
EMT14   067056
EMT140  071414
EMT141  071434
EMT142  071450
EMT143  071466
EMT144  071502
EMT145  071524
EMT146  071546
EMT147  071564
EMT15   067074
EMT150  071576
EMT151  071620
EMT152  071632
EMT153  071646
EMT154  071666
EMT155  071706
EMT156  071724
EMT157  071746
EMT16   067112
EMT160  071764
EMT161  072014
EMT162  072032
EMT163  072050
EMT164  072072
EMT165  072114
EMT166  072140
EMT167  072160
EMT17   067130
EMT170  072202
EMT171  072214
EMT172  072230
EMT173  072246
EMT174  072270
EMT175  072316
EMT176  072334
EMT177  072352
EMT2    066606
EMT20   067146
EMT200  072370
EMT201  072406
EMT202  072424
EMT203  072442
EMT204  072464
EMT205  072504
EMT206  072524
EMT207  072554
EMT21   067166
EMT210  072570
EMT211  072606
EMT212  072624
EMT213  072636
EMT214  072666
EMT215  072700
EMT216  072720
EMT217  072734
EMT22   067206
EMT220  072750
EMT221  072766
EMT222  073004
EMT223  073022
EMT224  073040
EMT225  073060
EMT226  073102
EMT227  073120
EMT23   067222
EMT230  073136
EMT231  073154
EMT232  073170
EMT233  073212
EMT234  073234
EMT235  073262
EMT236  073302
EMT237  073334
EMT24   067242
EMT240  073354
EMT241  073400
EMT242  073424
EMT243  073450
EMT244  073474
EMT245  073512
EMT246  073530
EMT247  073554
EMT25   067262
EMT250  073572
EMT251  073616
EMT252  073642
EMT253  073666
EMT254  073704
EMT255  073722
EMT256  073746
EMT257  073764
EMT26   067276
EMT260  074010
EMT261  074034
EMT262  074060
EMT263  074076
EMT264  074114
EMT265  074140
EMT266  074156
EMT267  074174
EMT27   067330
EMT270  074210
EMT271  074234
EMT272  074260
EMT273  074276
EMT274  074314
EMT275  074332
EMT276  074352
EMT277  074370
EMT3    066634
EMT30   067342
EMT300  074414
EMT301  074440
EMT31   067356
EMT32   067372
EMT33   067406
EMT34   067424
EMT35   067442
EMT36   067456
```

```
EMT37   067472          F0     = 000002          IVC    = 010000          PIRQVE= 000240          RMDAI   001334
EMT4    066654          F1     = 000004          LBC    = 002000          PLCLK   060432          RMDAO   001410
EMT40   067506          F2     = 000010          LBT    = 002000          PLFS   = 002000          RMDB   = 000022
EMT41   067526          F3     = 000020          LCLOCK  060424          PLSTP   060542          RMDBI   001350
EMT42   067542          F4     = 000040          LCOUNT  060474          PROMPT  065163          RMDBO   001424
EMT43   067566          GO     = 000001          LF     = 000012          PR0    = 000000          RMDC   = 000034
EMT44   067600          GTSWR  = 104407          LS     = 000004          PR1    = 000040          RMDCI   001362
EMT45   067614          HCE    = 000200          LSC    = 004000          PR2    = 000100          RMDCO   001436
EMT46   067630          HCI    = 002000          LST    = 000002          PR3    = 000140          RMDS   = 000012
EMT47   067644          HCRC   = 000400          LSTOP   060536          PR4    = 000200          RMDSI   001340
EMT5    066676          HELP    106724          MCLK   = 004000          PR5    = 000240          RMDSO   001414
EMT50   067666          HELPQS  065171          MCPE   = 020000          PR6    = 000300          RMDT   = 000026
EMT51   067710          HT     = 000011          MDF    = 000100          PR7    = 000340          RMDTI   001354
EMT52   067730          IAE    = 002000          MDPE   = 000400          PS     = 177776          RMDTO   001430
EMT53   067744          IDXMSK= 000077           MI     = 000004          PSEL   = 002000          RMEC1  = 000044
EMT54   067760          IE     = 000100          MOC    = 000400          PSTOP   060530          RMEC1I  001372
EMT55   070000          ILF    = 000001          MOH    = 020000          PSW    = 177776          RMEC10  001446
EMT56   070020          ILF02  = 000002          MOL    = 010000          PWRVEC= 000024           RMEC2  = 000046
EMT57   070034          ILF24  = 000024          MRD    = 002000          QSTMRK  065167          RMEC2I  001374
EMT6    066722          ILF26  = 000026          MS     = 000040          RD     = 000070          RMEC20  001450
EMT60   070050          ILF30  = 000030          MSC    = 000002          RDCHR  = 104411          RMER1  = 000014
EMT61   070064          ILF32  = 000032          MSE    = 100000          RDLIN  = 104412          RMER1I  001342
EMT62   070104          ILF34  = 000034          MSER   = 000200          RDOCT  = 104413          RMER10  001416
EMT63   070124          ILF36  = 000036          MUR    = 001000          RDY    = 000200          RMER2  = 000042
EMT64   070140          ILF40  = 000040          MWD    = 000010          READY   006326          RMER2I  001370
EMT65   070152          ILF42  = 000042          MWP    = 000010          RECAL  = 000006          RMER20  001444
EMT66   070166          ILF44  = 000044          MXF    = 001000          RESREG= 104415           RMHR   = 000036
EMT67   070202          ILF46  = 000046          NDTMSK= 115760           RESVEC= 000010          RMHRI   001364
EMT7    066744          ILF54  = 000054          NED    = 010000          REX    = 010000          RMHRO   001440
EMT70   070220          ILF56  = 000056          NEM    = 004000          RG     = 040000          RMLA   = 000020
EMT71   070240          ILF64  = 000064          NOP    = 000000          RGDTPT  066152          RMLAI   001346
EMT72   070264          ILF66  = 000066          NOTEX   066015          RH     = 000072          RMLAO   001422
EMT73   070310          ILF74  = 000074          NSA    = 100000          RIP    = 000020          RMMR1  = 000024
EMT74   070330          ILF76  = 000076          OCC    = 100000          RLEASE= 000012           RMMR1I  001352
EMT75   070340          ILR    = 000002          OFD    = 000200          RMAS   = 000016          RMMR10  001426
EMT76   070352          ILRG50= 000050           OFFSET= 000014           RMASI   001344          RMMR2  = 000040
EMT77   070366          ILRG52= 000052           OM     = 000001          RMASO   001420          RMMR2I  001366
ENRGDT  066600          ILRG54= 000054           ONES    066277          RMBA   = 000004          RMMR20  001442
EQUALS  065161          ILRG56= 000056           OPE    = 020000          RMBAE  = 000050          RMOF   = 000032
ERR    = 040000         ILRG60= 000060           OPI    = 020000          RMBAEI  001376          RMOFI   001360
ERRNMB  060102          ILRG62= 000062           OR     = 000200          RMBAEO  001452          RMOFO   001434
ERRTYP  057344          ILRG64= 000064           PACACK= 000022           RMBAI   001332          RMR    = 000004
ERRVEC= 000004          ILRG66= 000066           PAKACK= 000022           RMBAO   001406          RMSN   = 000030
ERTY00  060110          ILRG70= 000070           PAR    = 000010          RMCS1  = 000000          RMSNI   001356
ERTY01  060116          ILRG72= 000072           PAT    = 000020          RMCS1I  001326          RMSNO   001432
ERTY02  060126          ILRG74= 000074           PCLOCK  060406          RMCS10  001402          RMWC   = 000002
ERTY03  060135          ILRG76= 000076           PCOUNT  060474          RMCS2  = 000010          RMWCI   001330
ERTY04  060143          IOTVEC= 000020           PDA    = 000400          RMCS2I  001336          RMWCO   001404
ESRC   = 004000         IPCK0  = 000001          PGE    = 002000          RMCS20  001412          RQA    = 100000
FER    = 000020         IPCK1  = 000002          PGM    = 001000          RMCS3  = 000052          RQB    = 040000
FMT16  = 010000         IPCK2  = 0000            PHA    = 000200          RMCS3I  001400          RTC    = 000016
FNCDTB  066042          IPCK3  = 0000            PIP    = 020000          RMCS30  001454          R6     =%000006
FNCMSK= 000077          IR     = 000100          PIRQ   = 177772          RMDA   = 000006          R7     =%000007
```

```
SADMSK= 000037
SAVREG= 104414
SA1   = 000001
SA16  = 000020
SA2   = 000002
SA4   = 000004
SA8   = 000010
SC    = 100000
SCTMSK= 003700
SC0   = 000100
SC1   = 000200
SC2   = 000400
SC3   = 001000
SC4   = 002000
SEARCH= 000030
SEEK  = 000004
SETOM   060706
SETVV   060556
SIZCLK  060146
SKI   = 040000
SNGPRT= 020024
STACK = 001100
STANDA  005340
START   004542
STKLMT= 177774
STOP    001530
SWR     001154
SWREG   000176
SW0   = 000001
SW00  = 000001
SW01  = 000002
SW02  = 000004
SW03  = 000010
SW04  = 000020
SW05  = 000040
SW06  = 000100
SW07  = 000200
SW08  = 000400
SW09  = 001000
SW1   = 000002
SW10  = 002000
SW11  = 004000
SW12  = 010000
SW13  = 020000
SW14  = 040000
SW15  = 100000
SW2   = 000004
SW3   = 000010
SW4   = 000020
SW5   = 000040
SW6   = 000100
SW7   = 000200
SW8   = 000400
SW9   = 001000
TADMSK= 003400
TAG   = 020000
TAGADR= 001114
TAP   = 040000
TA1   = 000400
TA2   = 001000
TA4   = 002000
TBITVE= 000014
TIME    001522
TKVEC = 000060
TPVEC = 000064
TRAPVE= 000034
TRE   = 040000
TRTVEC= 000014
TST   = 010000
TSTNMB  060100
TSTQUE  001456
TST1    006342
TST10   012236
TST100  040056
TST101  040240
TST102  040460
TST103  040720
TST104  041104
TST105  041346
TST106  041534
TST107  041766
TST11   012360
TST110  042214
TST111  042522
TST112  043064
TST113  044506
TST114  046264
TST115  050474
TST116  051104
TST117  053302
TST12   012656
TST120  055620
TST13   013270
TST14   014116
TST15   014652
TST16   015010
TST17   015364
TST2    006716
TST20   015524
TST21   016046
TST22   016222
TST23   016406
TST24   016716
TST25   017254
TST26   017734
TST27   020074
TST3    007136
TST30   020420
TST31   020772
TST32   021320
TST33   022074
TST34   022444
TST35   022774
TST36   023354
TST37   023742
TST4    007352
TST40   024462
TST41   025012
TST42   025254
TST43   025714
TST44   026122
TST45   026540
TST46   027306
TST47   027640
TST5    007540
TST50   030300
TST51   030572
TST52   031036
TST53   031726
TST54   032236
TST55   032470
TST56   032732
TST57   033164
TST6    010676
TST60   033402
TST61   033656
TST62   034054
TST63   034330
TST64   034474
TST65   034676
TST66   035164
TST67   035432
TST7    012056
TST70   035710
TST71   036176
TST72   036440
TST73   036702
TST74   037166
TST75   037332
TST76   037514
TST77   037674
TYPBN = 104406
TYPDS = 104405
TYPE  = 104401
TYPOC = 104402
TYPON = 104404
TYPOS = 104403
T1      006404
T10     012300
T100    040120
T101    040302
T102    040522
T103    040762
T104    041146
T105    041410
T106    041576
T107    042030
T11     012422
T110    042256
T111    042564
T112    043126
T113    044550
T114    046326
T115    050536
T116    051146
T117    053344
T12     012720
T120    055662
T13     013332
T14     014160
T15     014714
T16     015052
T17     015426
T2      006760
T20     015566
T21     016110
T22     016264
T23     016450
T24     016760
T25     017316
T26     017776
T27     020136
T3      007200
T30     020462
T31     021034
T32     021362
T33     022136
T34     022506
T35     023036
T36     023416
T37     024004
T4      007414
T40     024524
T41     025054
T42     025316
T43     025756
T44     026164
T45     026602
T46     027350
T47     027702
T5      007602
T50     030342
T51     030634
T52     031100
T53     031770
T54     032300
T55     032532
T56     032774
T57     033226
T6      010740
T60     033444
T61     033720
T62     034116
T63     034372
T64     034536
T65     034740
T66     035226
T67     035474
T7      012120
T70     035752
T71     036240
T72     036502
T73     036744
T74     037230
T75     037374
T76     037556
T77     037736
UBUSQS  065225
UNS   = 040000
UNTMSK= 000007
UPE   = 020000
USE   = 040000
U0    = 000001
U1    = 000002
U2    = 000004
VV    = 000100
WATCH   001524
WC    = 000040
WCD   = 000050
WCE   = 040000
WCEHI = 010000
WCELO = 004000
WCF   = 000040
WCH   = 000052
WD    = 000060
WH    = 000062
WLE   = 004000
WRL   = 004000
XNUDC = 176000
XNUER2= 001567
XNUOF = 161577
XSIZ    005216
ZEROS   066254
$APTHD  001100
$ATYC   064736
```

```
$ATY1   064712        $ENDCT  057170        $LPADR  001122        $PWRDN  064534        $TMP3   001202
$ATY3   064720        $ENULL  057340        $LPCSB  001504        $PWRMG  064670        $TMP4   001204
$ATY4   064730        $ENV    001242        $LPCSR  001502        $PWRUP  064606        $TN   = 000121
$AUTOB  001150        $ENVM   001243        $LPERR  001124        $QUES   001216        $TPB    001166
$BASE   001276        $EOP    057134        $LPVEC  001506        $RDCHR  063726        $TPFLG  001173
$BDADR  001136        $EOPCT  057162        $MADR1  001254        $RDLIN  064016        $TPS    001164
$BDDAT  001142        $EOSP   057100        $MADR2  001260        $RDOCT  064324        $TRAP   064426
$BELL   001212        $ERFLG  001117        $MADR3  001264        $RDSZ = 000010        $TRAP2  064466
$BIN    061156        $ERMAX  001131        $MADR4  001270        $RESRE  061046        $TRP  = 000016
$CDW1   001302        $ERROR  062666        $MAIL   001222        $RTNAD  057336        $TRPAD  064500
$CDW2   001304        $ERRPC  001132        $MAMS1  001252        $SAVRE  061010        $TSTM   001104
$CHARC  062110        $ERRTB  001532        $MAMS2  001256        $SAVR6  064700        $TSTNM  001116
$CKSWR  063364        $ERTTL  001126        $MAMS3  001262        $SCOPE  062114        $TTYIN  064252
$CMTAG  001114        $ESCAP  001210        $MAMS4  001266        $SETUP= 000137        $TYPBN  061104
$CM3  = 000000        $ETABL  001242        $MBADR  001102        $STUP = 177777        $TYPDS  061160
$CM4  = 000005        $ETEND  001326        $MFLG   065154        $SVLAD  062354        $TYPE   061632
$CNTLC  064262        $FATAL  001224        $MNEW   064312        $SVPC = 000200        $TYPEC  062044
$CNTLG  064274        $FFLG   065156        $MSGAD  001236        $SWR  = 167400        $TYPEX  062112
$CNTLU  064267        $FILLC  001172        $MSGLG  001240        $SWREG  001244        $TYPOC  061430
$CPUOP  001250        $FILLS  001171        $MSGTY  001222        $SWRMK= 000000        $TYPON  061444
$CRLF   001217        $GDADR  001134        $MSWR   064301        $SW08T  062426        $TYPOS  061404
$DBLK   061374        $GDDAT  001140        $MTYP1  001253        $TESTN  001226        $UNIT   001234
$DDW0   001306        $GET42  057314        $MTYP2  001257        $TIMES  001206        $UNITM  001110
$DDW1   001310        $GTSWR  063454        $MTYP3  001263        $TKB    001162        $USWR   001246
$DDW2   001312        $HD   = 000001        $MTYP4  001267        $TKCNT  063066        $VECT1  001272
$DDW3   001314        $HIBTS  001100        $MXCNT  062424        $TKINT  063076        $VECT2  001274
$DDW4   001316        $HIOCT  064424        $NULL   001170        $TKQEN= 063075        $XTSTR  062126
$DDW5   001320        $ICNT   001120        $NWTST= 000001        $TKQIN  063070        $$GET4= 000000
$DDW6   001322        $ILLUP  064674        $OCNT   061626        $TKQOU  063072        $$SW08= 000121
$DDW7   001324        $INTAG  001151        $OMODE  061630        $TKQSR  063074        $OFILL  061627
$DEVCT  001232        $ITEMB  001130        $OVER   062410        $TKS    001160        .     = 113417
$DEVM   001300        $LF     001220        $PASS   001230        $TKSRV  063146        .$X   = 001100
$DOAGN  057334        $LFLG   065155        $PASTM  001106        $TMP0   001174
$DTBL   061364        $LLCSR  001512        $POWER  064702        $TMP1   001176
$ENDAD  057324        $LLVEC  001514        $PSW    001520        $TMP2   001200

. ABS.  113417      000

 ERRORS DETECTED:  0

 DSKZ:CZRMJC.BIN,DSKZ:CZRMJC.SEQ/NL:MC:MD:CND:TOC/LI:ME/DOC/SOL=DSKM:CZRMJC.P11
 RUN-TIME: 54 50 1 SECONDS
 RUN-TIME RATIO: 278/107=2.5
 CORE USED:  33K  (65 PAGES)

 DOCUMENT PAGES:  330
```