

LPA11

LPA-11 EXERCISER
CRLPABO

AH-A857B-MC
FICHE 1 OF 2

FEB 1981
COPYRIGHT © 77-80
MADE IN USA



The main body of the document is a large grid of approximately 20 columns and 20 rows. Each cell in the grid contains a small, dense block of text, likely representing a specific exercise or data point. The text is too small to be legible in this scan, but the overall layout is a structured table.

LPA11

LPA-11 EXERCISER
CRLPABO

AH-A857B-MC
FICHE 2 OF 2

FEB 1981
COPYRIGHT © 77-80
MADE IN USA



[Faint, illegible text and markings, possibly bleed-through from the reverse side of the page.]

PRODUCT CODE: AC-A855B-MC
DIAG. CODE: MAINDEC-11-CRLPA-B-D
PRODUCT NAME: CRLPABO LPA-11 EXERCISER
DATE: DEC. 1980

COPYRIGHT (C) 1977, 1978, 1980
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

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

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

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

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
3.0	LOADING PROCEDURE
3.1	METHOD
3.2	NON-STANDARD ADDRESS, VECTOR, CONFIGURATION
4.0	STARTING PROCEDURE
4.1	CONTROL SWITCH SETTINGS
4.2	STARTING ADDRESS
4.3	PROGRAM AND/OR OPERATOR ACTION
5.0	OPERATING PROCEDURE
5.1	SWITCH REGISTER FUNCTION
5.2	SCOPE LOOPS
5.3	PROGRAM AND/OR OPERATION ACTION
6.0	ERRORS
6.1	ERROR PRINTOUT
7.0	RESTRICTIONS
8.0	MISCELLANEOUS
8.1	POWER FAIL
8.2	XXDP,ACT,APT
8.3	EXECUTION TIME
8.4	LPA-11 (SYSTEM) DIAGNOSTIC SUMMARY
8.5	LPA-11 VERSION 5 MICRO-CODE ERROR SUMMARY
8.6	LPA-11 VERSION 5 MICRO-CODE ERROR LIST
8.7	LPA-11 (KMC-11) REGISTER DEFINITIONS
8.8	MICRO-CODE PROGRAM
8.9	ILLEGAL INTERRUPTS ON PROGRAM START

S

1.0 ABSTRACT

THIS PROGRAM WAS DESIGNED TO EXERCISE THE LPA-11XX SUBSYSTEM. IT IS DIVIDED INTO TWO SECTIONS. THE FIRST SECTION EXERCISES EACH INDIVIDUAL HARDWARE COMPONENT ON THE SYSTEM. PLEASE NOTE THAT THE DEFAULT SYSTEM WHICH THE PROGRAM USES, HAS ONE AD11K AND ONE KW11K ON IT. THE USER MUST INFORM THE PROGRAM AS TO ANY DIFFERENCES IN CONFIGURATION. IF THE PROGRAM DETECTS A HARDWARE PROBLEM, IT INFORMS THE USER OF IT. THE USER SHOULD RUN THE DIAGNOSTIC DESIGNED TO DIAGNOSE THE SECTION OF HARDWARE THAT FAILED. (EXAMPLE IF "CRLPA" INFORMED THE USER THAT THE AD11K FAILED, THE USER SHOULD RUN THE LPA/AD11K DIAGNOSTIC.)

THE SECOND PART OF THE DIAGNOSTIC IS DESIGNED TO RUN WITH USER (KMC) MICRO-CODE. THIS IS THE FIRST (AND ONLY) TIME THAT M8254 (IPBM) INTERRUPT ARBITRATION LOGIC IS CHECKED. IF ANY PROBLEMS OCCUR HERE, ITS A GOOD BET THAT THE M8254 MODULE IS BAD. IF NOT, YOU'RE GOING TO HAVE TO CABLE THE I/O BUS TO THE UNIBUS IN ORDER TO RUN THE MORE DETAILED DIAGNOSTIC AVAILABLE FOR THE OPTIONS.

GOOD SCOPE LOOPS ARE HARD TO COME BY SINCE THIS PROGRAM MUST TRY TO KEEP THREE PROCESSORS IN SYNC.

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP-11 FAMILY COMPUTER WITH 16K OF MEMORY (OR MORE) AND CONSOLE I/O FACILITIES (I.E. TTY)
2. LPA-11X TO BE EXERCISED
3. (OPTIONAL) KW11L OR KW11P (IMPROVES PROGRAM EXECUTION)

2.2 STORAGE

THIS PROGRAM OCCUPIES AND USES 16K OF MEMORY.

NOTE

IF 20K OR MORE MEMORY IS AVAILABLE, THIS PROGRAM WILL RUN A/D SAMPLING AT MAXIMUM SPEED.

3.0 LOADING PROCEDURE

3.1 METHOD

STANDARD PROCEDURE FOR NORMAL BINARY PROGRAM SHOULD BE FOLLOWED. THIS PROGRAM IS SUPPLIED ON MULTI-MEDIA AND CAN BE LOADED BY XXDP,ACT, OR APT.

3.2 NON-STANDARD ADDRESS,VECTOR, CONFIGURATION

THIS PROGRAM IS SET UP TO CHECK AN LPA-11X WITH STANDARD SET-UP AS LISTED BELOW. IT IS IMPORTANT THAT IF THE EQUIPMENT ADDRESSES VARY, THAT YOU ONLY CHANGE THESE LOCATIONS.

TAG	ADDRESS	CONTENTS	COMMENTS
---	-----	-----	-----
\$BASE:	001250	170460	::BASE ADDRESS OF EQUIPMENT
\$VECT1:	001244	000300	::VECTOR LOCATION

THE FOLLOWING ARE A LIST OF DEVICE ADDRESSES. YOU MAY CHANGE THE ADDRESS OF A DEVICE ONLY BY MODIFYING THE FOLLOWING LIST. DO NOT CHANGE ANY OTHER ADDRESSES!

AD11K:	1566	170400	:AD11K ADDRESS.
KW11K:	1570	170404	:KW11K ADDRESS.
DR11K1:	1572	167770	:DR11K #1 ADDR.
AA11K:	1574	170416	:AA11 ADDRESS.
AD11K2:	1576	170440	:AD11K #2 ADDRESS.
DR11K2:	1600	167760	:DR11K #2 ADDRESS.
DR11K3:	1602	167750	:DR11K #3 ADDRESS.
DR11K4:	1604	167740	:DR11K #4 ADDRESS.
DR11K5:	1606	167730	:DR11K #5 ADDRESS.
AR11:	1610	170400	:AR11 ADDRESS.
LPS11:	1612	170400	:LPS11 BASE ADDRESS.

SR1 INFORMS THIS DIAGNOSTIC AS TO WHAT DEVICES ARE ON THE I/O BUS. SR1 DEFAULTS TO 1 KW11K AND 1 AD11K IF YOUR CONFIGURATION IS DIFFERENT, YOU MUST CHANGE THIS LOCATION.

WORD BIT=1	OCTAL	DEVICE
0	000001	1ST AD11K
1	000002	1ST KW11K
2	000004	1ST DR11K
3	000010	1ST AA11K
4	000020	2ND AD11K#2
5	000040	2ND DR11K
6	000100	RESERVED
7	000200	3RD DR11K
8	000400	4TH DR11K
9	001000	5TH DR11K
10	002000	AR11
11	004000	RESERVED
12	010000	LPSAD (LPS A/D)
13	020000	LPSKW (LPS REAL TIME CLOCK)
14	040000	LPSVC (LPS D/A)
15	100000	LPSDR (LPS DIGITAL I/O)

SR1: 1562 003 ;DEVICE PRESENT FLAG DEFAULT
;IS 1 KW11K, 1 AD11K

SR2 INFORMS THIS DIAGNOSTIC AS TO HOW THE DEVICES SELECTED BY SR1 ARE SET-UP FOR TEST. WHILE NO SPECIAL SETUP IS REQUIRED TO RUN THIS TEST, THE DEPTH OF COVERAGE WILL INCREASE IF SPECIAL SETUPS ARE PERFORMED.

WORD BIT=1	OCTAL	FUNCTION
0	000001	AD11K HAS G5036 WRAP-AROUND MODULE.
2	000004	DR11K #1 HAS LOOP BACK CABLE
3	000010	AA11K HAS SCOPE DISPLAY (VISUAL TEST).
4	000020	AD11K #2 HAS G5036 WRAP AROUND MODULE.
5	000040	DR11K #2 HAS LOOPBACK CABLE
7	000200	DR11K #3 HAS LOOP BACK CABLE.
8	000400	DR11K #4 HAS LOOP BACK CABLE.
9	001000	DR11K #5 HAS LOOP BACK CABLE.
10	002000	AR11 HAS G5034 WRAP AROUND MODULE.
11	004000	AR11 HAS SCOPE DISPLAY (VISUAL TEST)
14	040000	LPS-11 D/A HAS SCOPE DISPLAY (VISUAL TEST)
15	100000	LPS-11 DIGITAL I/O HAS LOOP BACK CABLE

SR2: 1564 0 ;DEVICE SETUP REG.

4.0 STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

BEFORE STARTING THE DIAGNOSTIC, SET ALL SWITCH REGISTER BITS AS DESIRED, SEE SECTION 5.

4.2 STARTING ADDRESS

200 START OF TEST

4.3 PROGRAM AND/OR OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY
2. SET UP LOCATIONS SR1: AND SR2: TO REFLECT THE SYSTEM CONFIGURATION.
3. LOAD ADDRESS 200
4. SET SWITCH REGISTER TO DESIRED SETTING
5. START PROGRAM

5.0 OPERATING PROCEDURE

5.1 SWITCH REGISTER FUNCTION

SWR BIT	OCTAL	FUNCTION WHEN SET
15	100000	HALT ON ERROR
14	040000	LOOP ON TEST
13	020000	INHIBIT ERROR TYPEOUT
11	004000	INHIBIT TEST ITERATIONS
10	002000	RUN ONLY DEDICATED A TO D MODE MICRO-CODE
9	001000	RUN ONLY MULTIUSER MODE MICRO-CODE
8	000400	RUN TEST SELECTED BY <7:0>

5.2 SCOPE LOOPS

IF AN ERROR OCCURS AND THE USER WISHES TO SCOPE THE ERROR SWITCH REG. BIT 15 SHOULD BE SET TO HALT ON ERROR. WHEN THE CPU IS HALTED ON ERROR, SWITCH REG BIT 14 (LOOP ON TEST) AND SWR BIT 13 (INHIBIT ERROR TYPEOUT) SHOULD BE SET. SWR BIT 15 SHOULD BE CLEARED AND CPU SHOULD BE CONTINUED.

NOTE

SOME SCOPE LOOPS MAY BE IMPOSSIBLE TO OBTAIN OR NOT REPEATIBLE DUE TO THE FACT THAT THREE ASYNCHRONOUS CPUS ARE RUNNING TO GENERATE THE ERROR.

5.3 PROGRAM AND/OR OPERATOR ACTION

1. WHEN THE PROGRAM IS INITIALLY STARTED IT WILL TYPE:

MD-11-CRLPA-B LPA-11 SYSTEM EXERCISER

THE FIRST 'PASS' THROUGH THE PROGRAM IS QUICK VERIFY OR A SHORT ONE. ALL OTHER PASSES WILL ITERATE ON EACH SUBTEST UNLESS INHIBITED.

2. THE PROGRAM PERFORMS TESTS ON THE KMC11.
3. THE PROGRAM PERFORMS TESTS ON THE M8200-YC AND M8254.
4. THE PROGRAM TESTS EACH OPTION ON THE I/O BUSS SELECTED BY THE OPERATOR.
5. THE PROGRAM LOADS USER MICRO-CODE INTO THE KMC11 AND EXERCISES TOTAL LPA11-KX SYSTEM.
6. PROGRAM END PASS.

NOTE

ON ALL EVEN PASSES THROUGH THE PROGRAM, THE PROGRAM SELECTS MULTIUSER MICRO-CODE FOR USER MICRO CODE; AND ON ALL ODD NUMBERED PASSES THE PROGRAM SELECTS DEDICATED MODE MICRO-CODE FOR USER MICRO-CODE. THE OPERATOR MAY INFORM THE PROGRAM TO ONLY RUN ON VERSION OF USER-MICRO-CODE BY USE OF THE SWITCH REGISTER (SEE SECTION 5.1)

6.0 ERRORS -----

6.1 ERROR PRINT-OUT

PRINTOUT VARIES WITH THE ERROR DETECTED. THE ERROR PC TYPED OUT IS THE ACTUAL LOCATION OF THE ERROR CALL.

THE FOLLOWING IS A LIST OF LABELS ASSOCIATED WITH LPA11 FUNCTIONAL ERROR.

TSTWD - TEST IN WHICH THE ERROR WAS DETECTED.

USJNO - USER JOB NUMBER ASSIGNED TO THE JOB BY THE MICRO-CODE.

ALPCO - LPA11 CONTROL OUT REGISTER CONTENTS.

ALPCI - LPA11 CONTROL IN REGISTER CONTENTS.

ALPSO - LPA11 STATUS OUT REGISTER CONTENTS.

NOTE

ERROR REPORTS MAY OR MAY NOT SPECIFY A DEVICE. IF SO, THAT DEVICE IS NOT NECESSARILY THE FAULTY UNIT. HOWEVER, IT IS THE DEVICE UNDER TEST WHEN THE ERROR WAS DETECTED. FOR FURTHER INFORMATION, THE OPERATOR MUST CONSULT THE LISTING.

7.0 RESTRICTIONS -----

NO 'USER' CONNECTIONS SHOULD BE MADE TO THE LPA-11 SYSTEM.

8.0 MISCELLANEOUS

8.1 POWER FAIL

THIS PROGRAM WILL NOT CONTINUE FROM A POWER FAILURE. IF A POWER FAILURE OCCURS, THE PROGRAM WILL BE RESTARTED.

8.2 XXDP,ACT,APT

THIS PROGRAM IS CHAINABLE UNDER XXDP, ACT, OR APT. ALTHOUGH "APT HOOKS" HAVE BEEN INSTALLED, THEY HAVE NOT BEEN TESTED.

8.3 EXECUTION TIME

THE EXECUTION TIME WILL VARY BETWEEN CPUS. EXECUTION TIME ALSO VARIES WITH THE NUMBER AND TYPE OF OPTIONS ON THE LPA-11XX SYSTEM. THE APPROXIMATE TIMES ARE LISTED BELOW:

1.0 MINUTE (60 SEC) -NO ERRORS-ITERATIONS INHIBITED
3.0 MINUTE (180 SEC) -NO ERRORS-WITH ITERATIONS.
(LISTED UNDER MIS.)

8.4 LPA11 (SYSTEM) DIAGNOSTIC SUMMARY

DIAGNOSTICS FOR THE LPA11 ARE WRITTEN AT THREE LEVELS: (1) TOTAL PDP-11 SYSTEM, (2) LPA11 SYSTEM; AND, (3) LPA11 OPTIONS.

LEVEL 1, IS DESIGNED TO ISOLATE A FAILURE TO THE LPA11 SYSTEM. ALL OPTIONS ON THE PDP-11 ARE EXERCISED.

LEVEL 2 DIAGNOSTICS ISOLATE A FAILURE TO THE INDIVIDUAL OPTION WITHIN THE LPA11. THE LEVEL 2 DIAGNOSTIC IS MD-11-CRLPA. WHEN THE USER RUNS "CRLPA" HE CAN GENERALLY TELL WHICH OPTION DIAGNOSTIC (LEVEL 3) TO RUN NEXT. M8254 AND M8200-YC ERRORS MAY "LOOK" ALIKE AND "CRLPA" MAY NOT BE ABLE TO DISTINGUISH BETWEEN THEM. ARBITRATION ERRORS WILL NOT BE DETECTED BY THIS DIAGNOSTIC.

LEVEL 3 DIAGNOSTICS AID IN DETERMINING IF THE ERROR WAS IN FACT ON THE OPTION THE "CRLPA" SPECIFIED. THE USER MAY "LOOP" ON THE ERROR. WITHIN LEVEL 3, THERE ARE TWO GROUPS OF DIAGNOSTICS. THE FIRST GROUP REQUIRES NO "EXTRA" WORK BY THE USER IN ORDER TO RUN. GROUP "A" DIAGNOSTICS DO NOT CHECK ARBITRATION, AND REQUIRE EXTRA TIME FOR EXECUTION. THE SECOND GROUP (GROUP "B") REQUIRES THAT THE USER RECONFIGURE THE PDP-11 SYSTEM. THIS RECONFIGURATION INVOLVES CABLING THE UNIBUS TO THE LPA'S I/O BUS.

THE DIAGNOSTIC FOR THE M8254 FALLS INTO THE GROUP "B" CATEGORY.

THE LPA11-KX DIAGNOSTIC KIT WILL INCLUDE:

OPTION -----	GROUP -----	DIAG. # -----	DIAG. TITLE -----
LPA11-KX	LEVEL 2	MD-11-CRLPA	LPA11-KX SYSTEM EXER.
M8254	'B'	MD-11-CRLPN	M8254 (IPBM) FIELD DIAG.
AA11-K	A	MD-11-CRLPB	LPA/AA11-K DIAG.
	B	MD-11-DZAAC	AA11-K DIAG.
AR11-K	A	MD-11-CRLPC	LPA/AR11 DIAG. #1
	A	MD-11-CRLPD	LPA/AR11 DIAG. #2
	A	MD-11-CRLPE	LPA/AR11 DIAG. #3
	B	MD-11-DZARA	AR11 DIAG. #1
	B	MD-11-DZARB	AR11 DIAG. #2
	B	MD-11-DZARC	AR11 DIAG. #3
DR11-K	A	MD-11-CRLPF	LPA/DR11-K DIAG.
	B	MD-11-DZDRG	DR11-K DIAG.

KW11-K	A	MD-11-CRLPG	LPA/KW11-K DIAG.
	B	MD-11-DZKWK	KW11-K DIAG.
LPS11	A	MD-11-CRLPH	LPA/LPS11 DIAG. #1
	A	MD-11-CRLPI	LPA/LPS11 DIAG. #2
	A	MD-11-CRLPJ	LPA/LPS11 DIAG. #3
	B	MD-11-DZLPC	LPS11 DIAG. #1
	B	MD-11-DZLPD	LPS11 DIAG. #2
	B	MD-11-DZLPI	LPS11 DIAG. #3
AD11-K	A	MD-11-CRLPK	LPA/AD11-K DIAG.
	B	MD-11-DZADL	AD11-K DIAG.
M8200-YC	B	MD-11-CRLPL	LPA-DMC11 DIAG. TST I
	B	MD-11-CRLPM	LPA-DMC11 DIAG. TST II

8.5 LPA11 VERSION 5 MICRO-CODE ERROR SUMMARY

MICRO-CODE ERRORS ARE DEFINED AS ANY ERROR RETURNED TO THE USER BY THE MICRO-CODE THROUGH THE LPA11 STATUS REGISTER.

WITHIN THE DIAGNOSTIC WE CHECK TO BE SURE THAT WE CAN GENERATE SOME OF THESE ERRORS. OTHERS MAY COME UP AT ANY TIME WHILE THE LPA11-XX MICRO-CODE IS BEING RUN.

8.5.1 FATAL HARDWARE ERRORS

THE FATAL HARDWARE ERRORS CONSIST OF DATA ERRORS, IMPROPER STATUS CONDITIONS, OR MALFUNCTIONS WHICH ARE DETECTED DURING THE INITIALIZATION OF THE LPA11 SUBSYSTEM OR DURING THE PROCESSING OF DATA. NO SUBSEQUENT DATA WILL BE TRANSFERRED AND THE LPA11 WILL NOT RESPOND TO ADDITIONAL COMMANDS FROM THE PDP-11. THE LPA11 MUST BE RE-INITIALIZED PRIOR TO ACCEPTING ANOTHER COMMAND FROM THE PDP-11.

THESE ERRORS MAY COME UP IF THERE IS AN ARBITRATION ERROR.

8.5.2 START REQUEST ERRORS

THE START REQUEST ERRORS CONSIST OF ERRORS DETECTED BY THE MASTER DURING THE PROCESSING OF A NEW PDP-11 COMMAND TO THE LPA11.

8.5.3 USER REQUEST ERRORS

THE USER REQUEST ERRORS ARE SPECIFIED BY ERROR CODE 0-4 AND ERROR STATUS BITS 0 AND 1 OF THE STATUS OUT, BYTE 3. THE REQUEST IS DEALLOCATED AND MUST BE REISSUED IN ORDER TO RESUME. THE USER REQUESTS ERROR CODES FROM 240 TO 247 ARE GENERATED BY THE MASTER MICROPROCESSOR AND ERROR CODES FROM 250 TO 254 AND 257 ARE GENERATED BY CONDITIONS DETECTED IN THE SLAVE MICROPROCESSOR AND TRANSFERRED TO THE CSR OF THE MASTER. IN THE MULTI REQUEST MODE THE USER FOR WHICH THE ERROR APPLIES IS IDENTIFIED BY THE USER INDEX CODE OF THE CONTROL OUT WORD BYTE 2.

8.5.4 NORMAL STATUS RETURNS

THE NORMAL STATUS RETURNS ARE INDICATED BY ERROR CODE 0-4. THE STATUS RETURNS INDICATE A FULL OR OVERRUN CONDITION OF THE PDP-11 MEMORY BUFFERS ASSIGNED TO STORE DATA FROM THE LPA11. THE BUFFER ADDRESSES ARE SPECIFIED BY THE RDA INFORMATION CONTAINED IN THE I/O DEVICE START COMMAND. THE BUFFER FULL AND OVERRUN STATUS CAN BE SPECIFIED AS A FATAL OR NON FATAL CONDITION BY THE RDA VALID BUFFER MASK CONFIGURATION OF THE USER STATUS WORD. ANY OF THE THREE NORMAL STATUS RETURNS WILL CAUSE AN LPA11 CONTROL OUT INTERRUPT REQUEST TO BE GENERATED. THE BUFFER FULL CONDITION EXISTS WHEN THE LPA11 HAS FILLED AN ASSIGNED BUFFER WITH DATA. THE BUFFER OVERRUN CONDITION IS A RESULT OF ALL THE ASSIGNED BUFFERS BEING LOADED BEFORE THE PDP-11 HAS PROCESSED OR TRANSFERRED THE DATA FROM THE BUFFER. IF THE BUFFER OVERRUN CONDITIONS OCCUR CONTINUALLY; A USER REQUEST ERROR WILL RESULT.

8.6 LPA-11 VERSION 5 MICRO-CODE ERROR LIST

***** FATAL HARDWARE ERRORS *****

322 --- ADDRESS OF NON-EXISTENT DEVICE
 340 --- SLAVE POWER LOW
 341 --- MASTER FIFO READ / WRITE ERROR
 342 --- I/O BUS SACK TIME OUT
 343 --- MASTER INITIAL CONDITION ERROR
 BYTE 6 --- EXPECTED STATUS
 BYTE 7 --- BAD STATUS
 344 --- MASTER / SLAVE VERSION ERROR
 BYTE 6 --- MASTER VERSION NUMBER
 BYTE 7 --- SLAVE VERSION NUMBER
 345 --- SLAVE COLD START TIME-OUT
 346 --- FATAL SLAVE ERROR
 BYTE 6 --- 200 --- SLAVE FIFO READ / WRITE ERROR
 201 --- KW11-K CLOCK 'A' OVERRUN
 202 --- SLAVE FIRMWARE FIFO SEQUENCE ERROR
 347 --- FPATH DATA ERROR
 BYTE 6 --- BAD VALUE
 BYTE 7 --- EXPECTED VALUE
 350 --- FIFO DATA ERROR
 BYTE 6 --- BAD VALUE
 BYTE 7 --- EXPECTED VALUE

***** START REQUEST ERRORS *****

300 --- NO ROOM FOR REQUEST
 301 --- 'GO' SET WITH 'RDY IN' CLEAR
 302 --- MULTI-USER REQUEST WITH DEDICATED MODE MICRO CODE LOADED
 304 --- DEDICATED MODE REQUEST WITH MULTI-USER MICRO CODE LOADED
 306 --- 'START COMMAND WITH NO 'INITIALIZE'
 310 --- MULTIPLE 'INITIALIZE' COMMANDS
 312 --- 'STOP' COMMAND WITH USER NOT ACTIVE
 314 --- ODD ADDRESS SPECIFIED FOR REQUEST DESCRIPTOR ARRAY
 316 --- 'INITIALIZE' WITH WRONG VERSION SPECIFIED
 320 --- 'START' FOR DEVICE NOT IN CONFIGURATION
 322 --- 'START' WITH ILLEGAL FUNCTION SPECIFIED
 324 --- NON EXISTANT MEMORY IN REQUEST DESCRIPTOR ARRAY
 325 --- ODD ADDRESS SPECIFIED FOR BUFFER OR USER STATUS WORD
 326 --- DEVICE NOT FOUND ON I/O BUS DURING 'INITIALIZE' COMMAND
 BYTE 6,7 --- ADDRESS OF NON EXISTANT DEVICE

***** USER REQUEST ERRORS *****

240 --- NON FATAL ERROR COUNT EXCEEDED
241 --- ERROR STATUS OVERRUN
242 --- NON EXISTANT MEMORY IN RANDOM CHANNEL LIST
243 --- BUFFER OVERRUN / UNDERRUN
244 --- NON EXISTANT MEMORY IN BUFFER
245 --- NON EXISTANT MEMORY IN USER STATUS WORD
246 --- INVALID BUFFER INDEX SPECIFIED IN USER STATUS WORD
247 --- MASTER FIFO 7/8 FULL
250 --- REQUEST TERMINATED BY USER STATUS WORD REQUEST
251 --- NO FIRMWARE FIFO BUFFER AVAILABLE FOR REQUEST
252 --- SLAVE FIFO 7/8 FULL
253 --- RANDOM CHANNEL ADDRESS UNDERRUN
254 --- DATA UNDERRUN
255 --- NON EXISTANT CHANNEL OR DEVICE
260 --- MULTIPLE EXTERNAL TRIGGER DIGITAL OUTPUT REQUESTS

***** NORMAL STATUS RETURNS *****

000 --- START REQUEST PROCESSED
001 --- BUFFER FULL
002 --- BUFFER OVERRUN / UNDERRUN

1

8.7 LPA-11 (KMC-11) REGISTER DEFINITIONS

THE CONTROL AND STATUS INFORMATION, TRANSFERRED BETWEEN THE PDP-11 UNIBUS AND THE LPA-11 SUBSYSTEM, IS STORED IN THE CONTROL AND STATUS REGISTERS (CSR) OF THE MASTER MICROPROCESSOR (M8204). THE CSR'S CONSIST OF EIGHT, 8-BIT BYTES IMPLEMENTED AS FOUR 16-BIT WORDS OF MULTIPOINT RANDOM ACCESS MEMORY (RAM'S). THE CSR LOCATIONS ARE ADDRESSABLE FROM EITHER THE PDP-11 OR THE M8204 MICROPROCESSOR PROGRAM. THE UNIBUS ADDRESSES FOR EACH CSR IS AS FOLLOWS:

MASTER MICROPROCESSOR CSR

BYTE	UNIBUS	NUM	WORD
0 (LOW)	76 XXX 0	LPCI	1
1 (HIGH)	76 XXX 1	MAIN	1
2 (LOW)	76 XXX 2	LPCO	2
3 (HIGH)	76 XXX 3	LPSO	2
4 (LOW)	76 XXX 4	LPADL	3
5 (HIGH)	76 XXX 5		4
6 (LOW)	76 XXX 6	LPMS1	4
7 (HIGH)	76 XXX 7		4

CONTROL IN (BYTE 0)

THE CONTROL IN BYTE CONSISTS OF 8 BITS AT ADDRESS 76XXX0 USED BY THE PDP-11 PROGRAM TO INITIATE THE TRANSFER OF COMMANDS TO THE LPA-11 SUBSYSTEM. THE FORMAT AND DESCRIPTION OF BYTE INFORMATION IS SHOWN ON FIGURE 8-1.

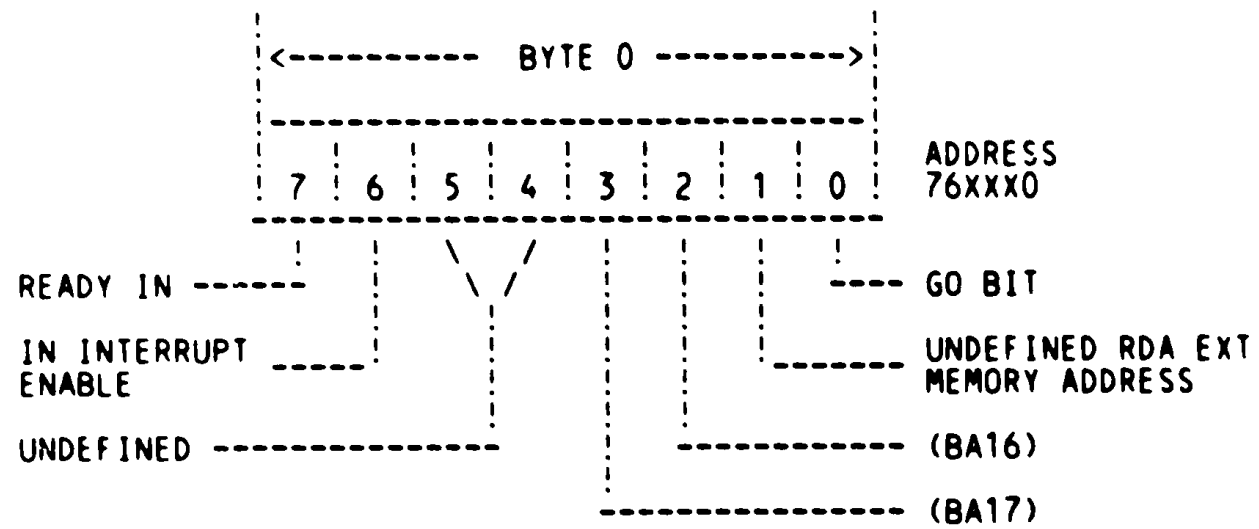


FIGURE 8-1
CONTROL IN (BYTE 0)

BIT	DESCRIPTION
0	<p>GO BIT- INITIAL CONDITION IS LOW (0). SET TO HIGH (1) BY THE PDP-11 PROGRAM TO INDICATE THAT A VALID REQUEST DESCRIPTOR ARRAY (RDA) ADDRESS IS AVAILABLE FOR PROCESSING BY THE LPA-11. CLEARED LOW (0) BY THE MASTER MICROPROGRAM AFTER THE USER'S REQUEST IS INITIATED.</p> <p>READ - LPA-11 MICROPROGRAM WRITE - PDP-11 PROGRAM</p>
1	NOT SPECIFIED
2,3	<p>BUS ADDRESS (BA16,BA17) - EXTENDED MEMORY BITS FOR REQUEST DESCRIPTOR ARRAY (RDA) ADDRESS (BYTES 4 AND 5) OF CSR.</p> <p>READ - PDP-11 PROGRAM WRITE - PDP-11 PROGRAM</p>
4,5	NOT SPECIFIED (MUST BE LOW 0)
6	<p>IN INTERRUPT ENABLE - INITIAL CONDITION IS LOW (0). SET TO HIGH (1) BY PDP-11 PROGRAM TO ENABLE INTERRUPT REQUEST TO BE GENERATED WHEN THE READY IN, BIT 07, IS HIGH (1) OR DURING A LOW TO HIGH TRANSITION.</p> <p>READ - PDP-11 PROGRAM WRITE - PDP-11 PROGRAM</p> <p>AN INTERRUPT OCCURS AT VECTOR ADDRESS +4. A READY IN INTERRUPT MAY OCCUR EITHER IMMEDIATELY OR AFTER A USEC DELAY. THE LPA-11 WILL CAUSE AN OUT INTERRUPT OR OUT STATUS BEFORE THE READY IN BIT IS SET IF BOTH ARE PENDING SIMULTANEOUSLY.</p>
7	<p>READY IN - INITIAL CONDITION IS LOW 0. SET TO HIGH (1) BY MICROPROCESSOR WHEN READY TO ACCEPT USER'S REQUEST. CLEARED LOW 0 AFTER REQUEST IS GRANTED.</p> <p>READ - PDP-11 PROGRAM WRITE - MICROPROCESSOR</p>

MAINTENANCE (BYTE 1)

THE MAINTENANCE BYTE IS AN EIGHT BIT, HIGH BYTE AT ADDRESS 76XXX1 WHICH CAN BE MONITORED BY THE PDP-11 PROGRAM DURING DIAGNOSTIC FUNCTIONS. REFER TO THE PROGRAMMING SECTION 4 OF THE KMC-11 MAINTENANCE MANUAL FOR A DESCRIPTION OF THE BITS NOT DEFINED. FIGURE 8-2 SHOWS THE BIT CONFIGURATION ON THE MAINTENANCE BYTE.

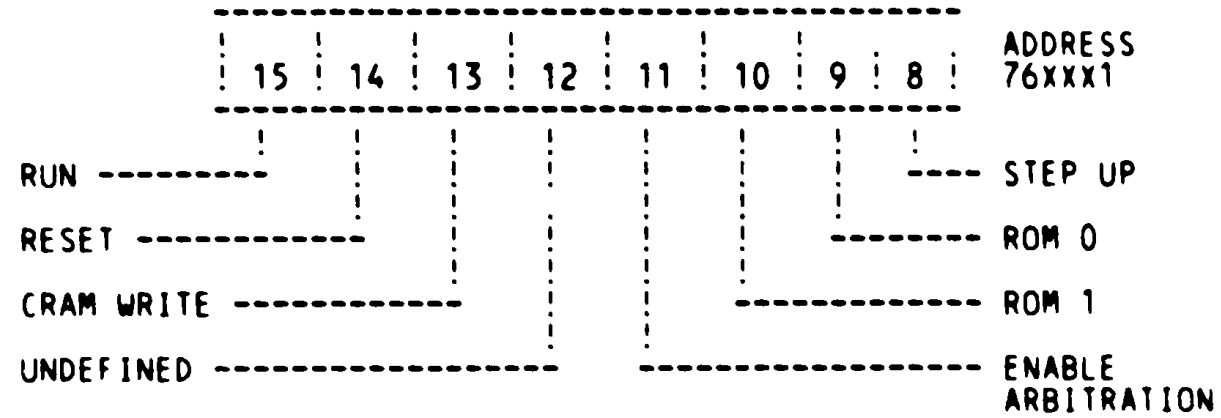


FIGURE 8-2
MAINTENANCE (BYTE 1)

BIT ---	DESCRIPTION -----
8,9,10	REFER TO KMC-11 MAINTENANCE MANUAL
11	ENABLE ARBITRATION - SET HIGH (1) BY PDP-11 PROGRAM TO ALLOW THE M8254 MODULE TO ARBITRATE NON PROCESSOR REQUESTS (NPR) AND BUS REQUESTS BR BETWEEN THE SLAVE PROCESSOR AND DEVICES ON THE I/O BUS. WHEN CLEARED LOW (0) BY PDP-11 PROGRAM THE ARBITRATION LOGIC IS INHIBITED. READ - PDP-11 PROGRAM AND MICROPROCESSOR WRITE - PDP-11 PROGRAM
12,13	REFER TO KMC-11 MAINTENANCE MANUAL
14	RESET - SET HIGH (1) BY PDP-11 PROGRAM TO CLEAR ALL PERTINENT REGISTERS IN THE LPA-11 SUBSYSTEM. READ - LPA-11 MICROPROGRAM WRITE - PDP-11 PROGRAM ONLY
15	REFER TO KMC-11 MAINTENANCE MANUAL

CONTROL OUT (BYTE 2)

THE CONTROL OUT BYTE CONSISTS OF 8 BITS AT ADDRESS 76XXX2 AND IS USED BY THE LPA-11 TO INDICATE TO THE PDP-11 THE AVAILABILITY OF THE LPA STATUS INFORMATION. THE FOMAT AND DESCRIPTION OF THE BYTE INFOMATION IS SHOWN ON FIGURE 8-3.

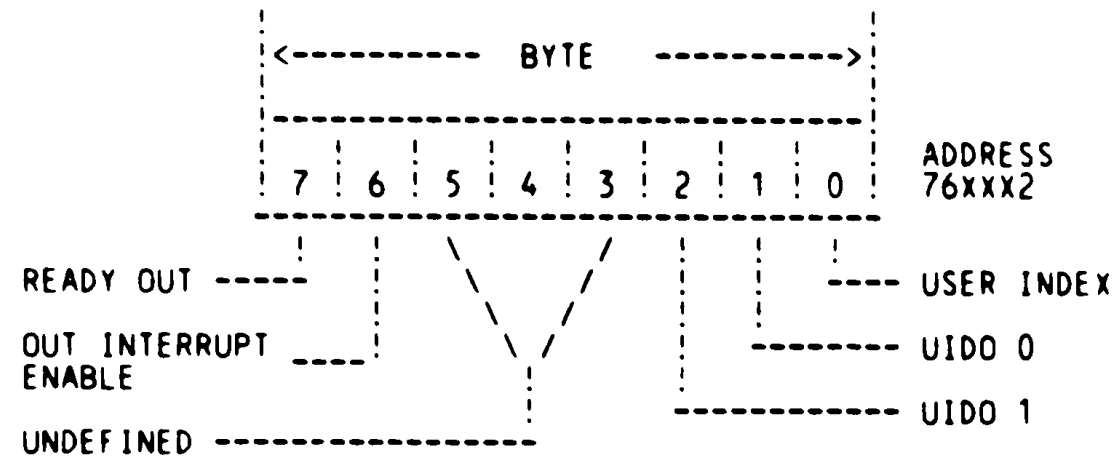


FIGURE 8-3

CONTROL OUT (BYTE 2)

BIT ---	DESCRIPTION -----
0,1,2	USER'S INDEX - AN OCTAL CODE (0-7) ASSIGNED BY THE MASTER MICROPROCESSOR IN THE MULTI REQUEST MODE TO IDENTIFY THE STATUS AS RELATED TO UP TO EIGHT USER REQUESTS.
3,4,5	NOT SPECIFIED
6	OUT INTERRUPT ENABLE - SET HIGH (1) BY MASTER MICROPROCESSOR PROGRAM TO ALLOW AN INTERRUPT REQUEST TO BE GENERATED AT VECTOR +00 WHEN THE READY OUT (BIT 07) IS SET. WRITE - PDP-11 PROGRAM
7	READY OUT - SET HIGH (1) BY THE MASTER MICROPROCESSOR PROGRAM TO INDICATE THAT THE LPA-11 SUBSYSTEM HAS STATUS INFORMATION FOR THE PDP-11. CLEARED LOW (0) BY THE PDP-11 PROGRAM TO ACKNOWLEDGE THE RECEPTION OF THE LPA-11 STATUS INFORMATION. READ - PDP-11 PROGRAM WRITE - PDP-11 PROGRAM

USER INDEX CODE

A USER INDEX CODE IS SPECIFIED BY THE LPA-11 TO IDENTIFY THE ERROR OR STATUS INFORMATION AS RELATED TO ONE OF EIGHT USER'S REQUESTS. THE USER'S INDEX IS AN OCTAL CODE FORM 0 - 7 AND IS ASSIGNED TO THE USER IN ORDER THAT THE REQUESTS ARE GRANTED. THIS EFFECTIVELY PROVIDES A VARIABLE INDEX TO IDENTIFY FURTHER COMMUNICATIONS WITH AN ASSOCIATED USER. THE USER'S INDEX CODE IS NOT VALID FOR FATAL HARDWARE ERROR CONDITION IN LPA-11 WHICH EFFECT ALL ACTIVE USER'S AND FOR START REQUEST ERRORS WHICH OCCUR PRIOR TO ESTABLISHING THE NEW USER'S REQUEST.

STATUS OUT (BYTE 3)

THE STATUS OUT IS AN 8-BIT, HIGH BYTE AT ADDRESS 76XXX3 WHICH INDICATES STATUS AND ERROR CONDITIONS TO THE PDP-11 PROGRAM DURING THE ESTABLISHMENT OF USER REQUESTS OR DURING THE TRANSFER OF DATA. THE ERRORS ARE CLASSIFIED AS FATAL HARDWARE ERRORS WHICH RESULT IN A HALT OF THE MICROPROCESSOR OR PROGRAM AND NON FATAL ERRORS WHICH INDICATE SPECIFIC STATUS CONDITIONS WHICH HAVE OCCURRED. THE HALT CONDITION PREVENTS THE LPA-11 FROM ACCEPTING ANY ADDITIONAL USER REQUESTS OR FROM TRANSFERRING ANY DATA TO OR FROM THE PDP-11. THE STATUS OUT BYTE CONTAINS FIVE ERROR CODE BITS 8 - 12, TWO ERROR STATUS BITS 13,14 AND FATAL ERROR INDICATOR, BIT 15, AS SHOWN ON FIGURE 8-4. THE ERROR CODES LISTED ON TABLE 8-1, 8-2 AND 8-3 ARE GROUPED AS SHOWN ON THE STATUS OUT BIT CONFIGURATION.

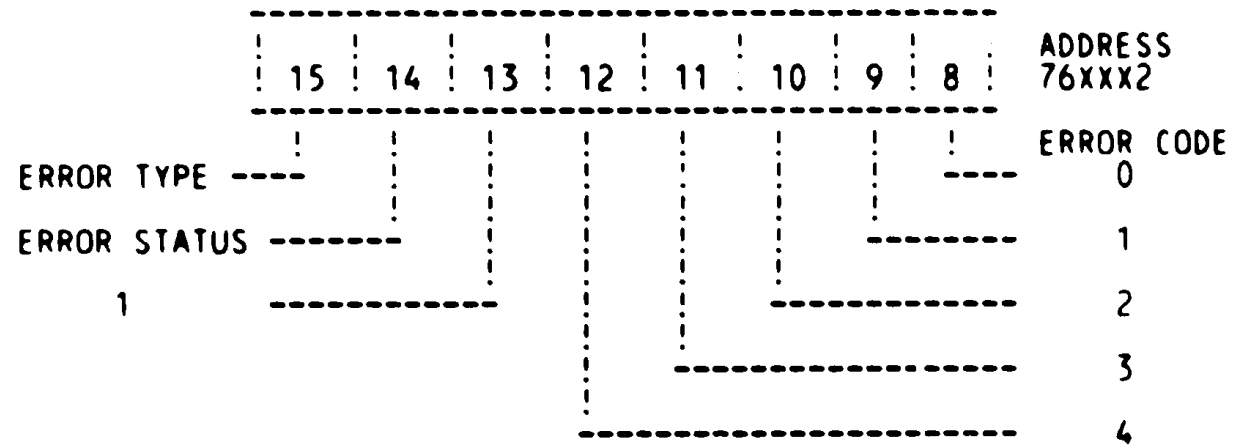


FIGURE 8-4
STATUS OUT (BYTE 3)

BIT ---	DESCRIPTION -----
8-12	<p>ERROR CODE 0-4 - A SUB CODE AS LISTED ON TABLE WHICH FURTHER DEFINES THE ERROR CODE INDICATED BY THE ERROR STATUS 0 (BIT 14) AND ERROR STATUS 1 (BIT 1).</p> <p> READ - PDP-11 PROGRAM WRITE - MASTER MICROPROCESSOR</p>
13,14	<p>ERROR STATUS 0 AND 1 - INDICATES THE CONDITIONS FROM WHICH THE ERROR OR STATUS ORIGINATED</p> <p> READ - PDP-11 PROGRAM WRITE - MASTER MICROPROCESSOR</p>
15	<p>STATUS/ERROR INDICATOR - A SINGLE BIT WHICH INDICATES STATUS OR FATAL ERROR CONDITIONS REQUIRING ATTENTION. A HIGH (1) INDICATES AN ERROR AND A LOW (0) INDICATES STATUS.</p> <p> READ - PDP-11 PROGRAM WRITE - MASTER MICROPROCESSOR</p>

ERROR/STATUS CODES

THE ERROR/STATUS CODES INDICATED BY THE STATUS OUT BYTE 3 ARE DEFINED AS FATAL HARDWARE ERRORS, START REQUEST ERRORS, USER REQUEST ERRORS AND NORMAL STATUS RETURNS.

THE ERROR/STATUS CODES AS LISTED ON THE FOLLOWING TABLES INCLUDE ALL EIGHT BITS STARTING AT BIT 0 TO BIT 7.

REQUEST DESCRIPTOR ARRAY ADDRESS (BYTE 4 AND 5)

THE BUS ADDRESSES OF THE REQUEST DESCRIPTOR ARRAYS (RDA) IN THE PDP-11 MEMORY ARE CONTAINED WITHIN BYTES 4 AND 5 OF THE LPA-11 CSR. BYTE 4 AND 5 EACH CONTAIN 8-ADDRESS BITS (BA00-BA07 AND BA08-BA15) AS SHOWN ON FIGURE 8-5 AND BYTE 0 CONTAINS TWO ADDRESS BITS (BA16 AND BA17).

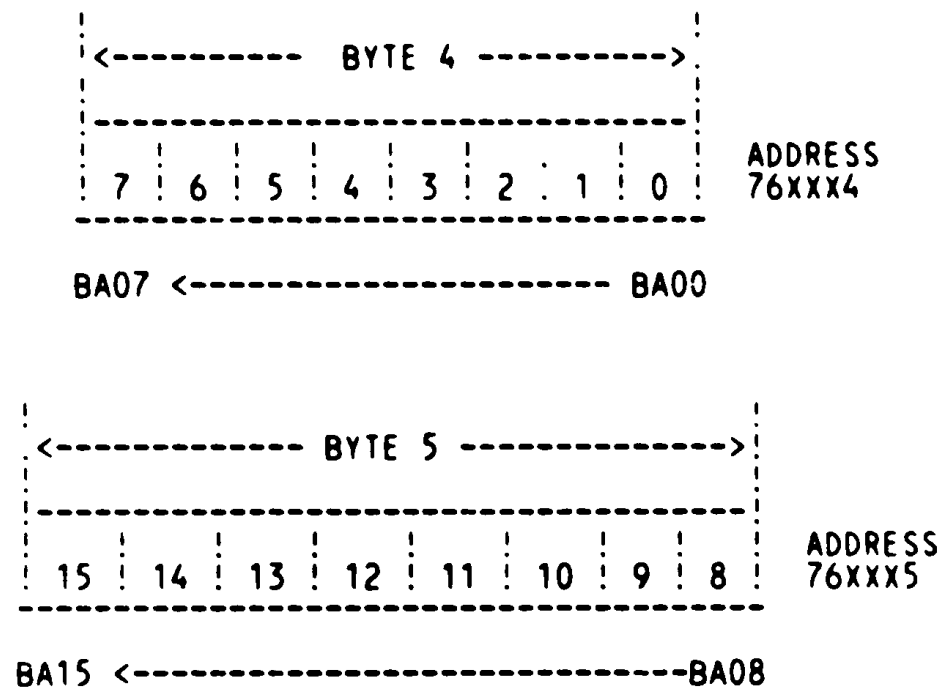


FIGURE 8-5
REQUEST DESCRIPTOR ARRAY ADDRESS

BIT ---	DESCRIPTION -----
0-7	BA00-BA07 - LOW BYTE OF THE RDA ADDRESS IN PDP-11 MEMORY
8-15	BA08-BA15 - HIGH BYTE OF THE RDA ADDRESS IN PDP-11 MEMORY.

MAINTENANCE STATUS (BYTE 6 AND 7)

THE MAINTENANCE STATUS, BYTES 6 AND 7, ARE LISTED ON TABLE 8-6 AND INDICATE THE SPECIFIC STATUS AND ERROR CHECK VALUES ASSOCIATED WITH THE FATAL HARDWARE ERRORS DEFINED ON TABLE 8-2. THE FORMAT OF THE LOW AND HIGH 8-BIT ERROR STATUS CODES ARE SHOWN ON FIGURE 8-6.

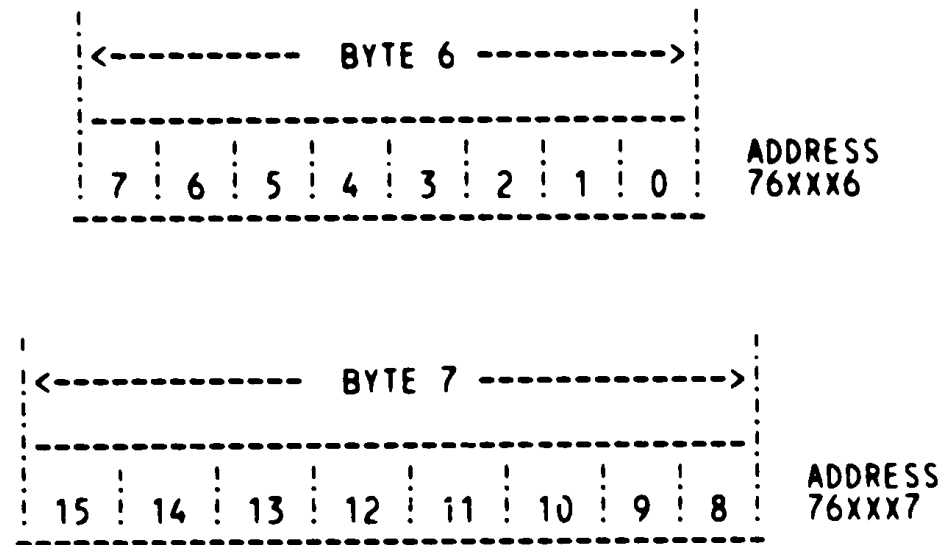


FIGURE 8-6
ERROR STATUS (BYTES 6 AND 7)

8.8 MICRO-CODE PROGRAMS

THERE ARE FOUR SEPARATE MICRO-CODE PROGRAMS LOADED INTO THE KMC11 DURING THE COURSE OF THIS DIAGNOSTIC. A BRIEF DESCRIPTION OF EACH FOLLOWS.

8.8.1 DRLPX1

THIS IS THE FIRST MICRO-CODE PROGRAM LOADED INTO THE KMC11. THE MICRO-CODE RESIDES AT LOCATION 65000 STARTING AT TAG 'DRLPX1'. ITS FUNCTION IS TO VERIFY PROPER OPERATION OF THE KMC11 BOARD. IT WILL PERFORM CROM READ/WRITE TESTS, BRANCH TESTS, ALU TESTS, NPR TESTS, AND INTERRUPT TESTS. IF THE KMC11 IS FOUND TO BE FAULTY, THE OPERATOR MUST ABORT 'CRLPA' AND RUN THE KMC11 DIAGNOSTICS.

8.8.2 DRLPX0

THE MICRO-CODE PROGRAM 'DRLPX0' IS LOADED INTO THE KMC11 AT THE ONSET OF TEST #3. THE MICRO-CODE RESIDES AT LOCATION 64000 STARTING AT TAG 'DRLPX0'. DRLPX0 PROVIDES COMMUNICATIONS FACILITIES BETWEEN THE KMC11 AND THE SLAVE MICRO-PROCESSOR. IT IS USED DURING TESTING OF THE SLAVE MICRO-PROCESSOR, THE IPBM, AND THE OPTIONS ON THE I/O BUS.

8.8.3 USER MICRO-CODE

THERE ARE TWO SEPARATE USER MICRO-CODE PROGRAMS LOADED INTO THE KMC11; DEDICATED AND MULTI-USER. THEIR FUNCTION IN RELATION TO THIS DIAGNOSTIC IS TO EXERCISE THE OPTIONS ON THE I/O BUS. A BRIEF DESCRIPTION OF EACH FOLLOWS.

DEDICATED A TO D MICRO-CODE

THIS PROGRAM ALLOWS HIGH SPEED SAMPLING OF A/D DEVICES ON THE I/O BUS. ONLY ONE JOB CAN BE RUN AT ANY GIVEN TIME. THE MICRO-CODE STARTS ABOUT LOCATION 50200 AT TAG 'DMAST'. THE MICRO-CODE IS STORED AS A TABLE OF OCTAL NUMBERS AND IS NOT PRINTED.

MULTI-USER MICRO-CODE

WHEN RUNNING MULTI-USER MICRO-CODE, ANY NUMBER OF JOBS FROM ONE TO EIGHT MAY BE RUN AT THE SAME TIME BY THE LPA11 SUBSYSTEM. THE JOBS MAY BE DIGITAL INPUT, A/D SAMPLING, AND/OR D/A CONVERSIONS. THE MICRO-CODE STARTS ABOUT LOCATION 44200 AT TAG 'MMAST'. THE MICRO-CODE IS STORED AS A TABLE OF OCTAL NUMBERS AND IS NOT PRINTED.

8.9 ILLEGAL INTERRUPTS ON START

SINCE THE KMC11 CAN NOT BE INITIALIZED UNTIL TEST #1 IS IN PROGRESS, THERE EXISTS THE POSSIBILITY THAT PREVIOUSLY LOADED MICRO-CODE WILL CAUSE A PROCESSOR INTERRUPT. THIS WILL BE INDICATED BY A PROCESSOR HALT AT THE KMC11 INTR. VECTOR LOCATION.

IN ORDER TO CORRECT THIS PROBLEM, THE OPERATOR MUST MANUALLY CLEAR ALL FOUR KMC11 REGISTERS SEEN BY THE UNIBUS. THIS IS DONE BY LOADING THE BASE ADDRESS OF THE KMC11 AND DEPOSITING ZERO'S IN FOUR CONSECUTIVE WORD LOCATIONS.

THE PROGRAM CAN THEN BE RESTARTED.

HISTORY FILE OF "CRLPAB"

THE 'B' VERSION WAS CAUSED BY A MICRO-CODE CHANGE TO THE FILE "CRLPA.MAC" WHICH REQUIRED THE PROGRAM TO BE REASSEMBLED. "CRLPAB" NOW SUPPORTS DMC CODE VERSION 4 AND 5.

CRLPABO ALSO INCLUDED THE FIXES THAT WERE PATCHED IN -A1 AND -A2.

LNKX11 V023 24-OCT-80 9:29

#CRLPAB.BIN/B:0,CRLPAB.MAP=CRLPAB,CRLPX0,CRLPX1/E

LOAD MAP

IDENT: LPA.03

TRANSFER ADDRESS: 000001

LOW LIMIT: 000000

HIGH LIMIT: 004000

MODULE	DMDT	SECTION ENTRY	ADDRESS	SIZE
<. ABS.>			000000	000000
	DRLPX0		064000	
	DRLPX2		064000	
	DRLPX1		065000	
	D.OCQ		054157	
	D.OCS		054134	
	D.OEQ		054111	
	D.OES		054066	
	D.TCQ		054273	
	D.TCQP		054407	
	D.TCS		054250	
	D.TCSP		054364	
	D.TEQ		054225	
	D.TEQP		054341	
	D.TES		054202	
	D.TESP		054316	
	FRECOR		056156	
<		>	000000	000000

MODULE	DMDT	SECTION ENTRY	ADDRESS	SIZE
<		>	000000	000000
<ABCODE>			000000	004000

MODULE	DMDT	SECTION ENTRY	ADDRESS	SIZE
<		>	004000	000000

RUN-TIME: 1 SECONDS
2K CORE USED

1163	OPERATIONAL SWITCH SETTINGS
1165	TRAP CATCHER
1179	BASIC DEFINITIONS
1183	ACT11 HOOKS
1186	APT PARAMETER BLOCK
1187	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
1323	*****
1324	SR1 DEVICE PRESENT REG
1325	*****
1359	*****
1360	SR2 DEVICE SET-UP REG
1361	*****
1390	*****
1391	LPA-11 I/O BUS DEVICE ADDRESS DEFINITION AND LIST
1392	*****
1416	AD11-K ADDRESSES
1426	AA11-K ADDRESSES
1442	DR11K #1 ADDRESS
1443	DR11K #2 ADDRESS
1444	DR11K #3 ADDRESS
1445	DR11K #4 ADDRESS
1446	DR11K #5 ADDRESS
1449	KW11K ADDRESS
1463	AR11 ADDRESSES
1476	LPS-11 ADDRESSES
1520	INITIALIZE THE COMMON TAGS
1644	TYPE PROGRAM NAME
(2)	GET VALUE FOR SOFTWARE SWITCH REGISTER
1650	T1 *TEST THE ADDRESSABILITY OF THE KMC-11
1695	T2 *TEST KMC-11 FUNCTIONS
1696	KMC-11 MICRO-INITIALIZATION.
1707	KMC-DIAG. SUB-TEST#1 CRAM WRITE ONES TEST
1758	KMC-DIAG SUB-TEST #2 CRAM WRITE ZEROES TEST
1807	KMC-DIAG SUB TEST #3 BRANCH TEST
1873	KMC-DIAG SUB-TEST #4 ALU TEST
1920	KMC-DIAG SUB-TEST #5 NPR TEST ZERO TRANSFER
1991	KMC-DIAG SUB-TEST #6 NPR TEST ONES TRANSFER.
2063	KMC-DIAG SUB-TEST #7 INTERRUPT TEST.
2157	T3 *TEST SLAVE MICRO PROCESSOR START
2223	**PHASE 2 IPBM/DMC TESTING.
2268	T4 *TEST DATA LOOP BACK THROUGH IPBM,M8254 MODULE
2430	PHASE 3 I/O BUS TESTING.
2578	T5 I/O DEVICE TEST-AD11-K OPTION
2581	T6 I/O DEVICE TEST-AD11-K OPTION
2622	T7 I/O DEVICE TEST-KW11K OPTION
2857	T10 *TEST THE DR11K OPTION,#1, IF "SR1" BIT 2=1(SET)
2859	T11 *TEST THE DR11K OPTION,#2, IF "SR1" BIT 5=1(SET)
2861	T12 *TEST THE DR11K OPTION,#3, IF "SR1" BIT 7=1(SET)
2863	T13 *TEST THE DR11K OPTION,#4, IF "SR1" BIT 8=1(SET)
2865	T14 *TEST THE DR11K OPTION,#5, IF "SR1" BIT 9=1(SET)
2875	T15 *TEST THE AA-11 OPTION, IF "SR1" BIT 3=1 (SET)
3032	T16 I/O DEVICE TEST-AR11 OPTION
3138	T17 I/O DEVICE TEST-AR11 CLOCK OPTION
3214	T20 *TEST THE AR11 DISPLAY OPTION, IF "SR1" BIT 10=1 (SET)

3326	T21	*TEST THAT THE AR11 CAN DISPLAY A SQUARE, SELECTED BY BIT 10 OF SR1,SR2
3356	T22	I/O DEVICE TEST LPS-11 A/D OPTION
3489	T23	I/O DEVICE TEST-LPS-CLOCK OPTION
3564	T24	*TEST THE LPS I/O, IF "SR1" BIT 15=1(SET)
3671	T25	*TEST THE LPS D/A OPTION, IF "SR1" BIT 14=1 (SET)
3790	T26	*TEST THAT THE LPS D/A CAN DISPLAY A SQUARE
3870	T27	*TEST THAT THE AA11K # CAN DISPLAY A SQUARE, SELECTED BIT 3 IN SR1,SR2
3879	T30	*TEST THAT USER MICRO-CODE CAN BE STARTED
3936	T31	*TEST FOR ERROR CODE 306
3979	T32	*TEST THAT WE CAN GENERATE ERROR CODE 314
4011	T33	*TEST THAT THE DEVICE LIST CAN BE VERIFIED
4155	T34	*TRY TO GENERATE ERROR CODE 312
4186	T35	*TRY TO GENERATE ERROR CODE 310
4243	T36	*TEST INTERRUPTS
4307	T37	*TEST THAT THE LPA-SYSTEM CLOCK CAN BE STARTED
4449	T40	*TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #1
4451	T41	*TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #2
4453	T42	*TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #3
4455	T43	*TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #4
4457	T44	*TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #5
4471	T45	*TEST THE LPA SYSTEM USER MICRO-CODE'S ABILITY TO TAKE SAMPLE SINGLE JOB
4545	T46	*TEST THE LPA SYSTEM USER MICRO-CODE'S ABILITY TO TAKE RANDOM SAMPLES SINGLE JOB
4640	T47	*TEST THAT MULTI JOBS CAN BE STARTED AND RUN
4761	T50	*HIGH SPEED A/D SAMPLE TEST (DEDICATED MODE,SPECIAL TEST
4822	T51	END OF TFSTS
4837		END OF PASS ROUTINE
4840		BINARY TO OCTAL (ASCII) AND TYPE
4841		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4847		ERROR HANDLER ROUTINE
4848		ERROR MESSAGE TYPEOUT ROUTINE
4849		SCOPE HANDLER ROUTINE
4850		TTY INPUT ROUTINE
4854		TYPE ROUTINE
4855		READ AN OCTAL NUMBER FROM THE TTY
4856		APT COMMUNICATIONS ROUTINE
4857		POWER DOWN AND UP ROUTINES
4863		: WIR ROUTINE TO WAIT FOR "DRLPX0" MICROCODE TO BECOME READY.
5308		TRAP DECODER
(3)		TRAP TABLE
5331		.ASCIZ MESSAGES.
5503		DMDT -- DEDICATED MODE DISPATCH TABLE

1
2
3
4
5
6
7
8
9
10
11
12
13
52
53
54
140
156
169
182
183
416
417
458
510
609
651
698
747

.REM [

CRLPAB.MAC

WELCOME, THIS DIAGNOSTIC IS ONE IN A SERIES OF DIAGNOSTIC
DESIGNED IN ORDER TO AID YOU IN TESTING THE LPA-11XX OPTION.
! HOPE THAT YOU HAVE READ THE DOCUMENTATION SECTION OF THIS
DIAGNOSTIC. IF YOU HAVE, YOU KNOW ABOUT ALL OF THE DIAGNOSTICS
THAT ARE AVAILABLE FOR TESTING THE LPA SYSTEM.

GOOD LUCK .

[
.GLOBL DRLPX2

763
764
765
766
767
768
769
770
771
905
906
907
908
909
910
911
912
913
1047

.TITLE MMAST.MAC
.IDENT /4.01/

:
: LPA11-K MICRO CODE
:
: CHARLES A. SAMUELSON
: NOVEMBER, 1977
:

.TITLE DMAST.MAC
.IDENT /4.01/

:
: LPA11-K MICRO CODE
:
: CHARLES A. SAMUELSON
: NOVEMBER, 1977
:

```

1161      .TITLE MAINDEC -11- CRLPA-B
(1)      :*COPYRIGHT (C) 1980
(1)      :*DIGITAL EQUIPMENT CORP.
(1)      :*MAYNARD, MASS. 01754
(1)      :*
(1)      :*PROGRAM BY EDWARD C. BADGER, R SHGOP
(1)      :*
(1)      :*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)      :*PACKAGE (MAINDEC-11-DZQAC-(3), JAN 19, 1977.
(1)
(1)      000001      $TN=1
1162
1163      .SBTTL OPERATIONAL SWITCH SETTINGS
(1)
(1)      :
(1)      :      SWITCH      USE
(1)      :      -----
(1)      :      15      HALT ON ERROR
(1)      :      14      LOOP ON TEST
(1)      :      13      INHIBIT ERROR TYPEOUTS
(1)      :      11      INHIBIT ITERATIONS
(1)      :      10      BELL ON ERROR
(1)      :      9      LOOP ON ERROR
(1)      :      8      LOOP ON TEST IN SWR<7:0>
1164      .ENABL GBL
1165      .SBTTL TRAP CATCHER
(1)
(1)      000000      .=0
(1)      :*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)      :*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)      :*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)      .=174
(1)      000174      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
(1)      000176      000000      SWREG: .WORD 0      ;;SOFTWARE SWITCH REGISTER
1166      000100      000100      .=100
1167      000100      000104      000200      000002      .WORD 104,200,2
1168
1169      000200      000200      .=200
1170      000200      000137      002016      JMP START
1171      000214      000214      .=214
1172      000214      000137      040646      JMP $UTK      ;JUMP TO USER LINK
1173      000220      000220      .=220
1174      000220      000137      034646      JMP MAKEI
1175
1176      000240      000240      .=240
1177      000240      000137      041234      JMP RBUFR
1178
1179      .SBTTL BASIC DEFINITIONS
(1)
(1)      :*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
(1)      001100      STACK= 1100
(1)      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
(1)      .EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)      :*MISCELLANEOUS DEFINITIONS
(1)      000011      HT= 11      ;;CODE FOR HORIZONTAL TAB
(1)      000012      LF= 12      ;;CODE FOR LINE FEED

```

```

(1) 000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
(1) 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1) 177776 PS= 177776 ;;PROCESSOR STATUS WORD
(1) .EQUIV PS,PSW
(1) 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
(1) 177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
(1) 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
(1) 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1)
(1) ;*GENERAL PURPOSE REGISTER DEFINITIONS
(1) 000000 R0= X0 ;;GENERAL REGISTER
(1) 000001 R1= X1 ;;GENERAL REGISTER
(1) 000002 R2= X2 ;;GENERAL REGISTER
(1) 000003 R3= X3 ;;GENERAL REGISTER
(1) 000004 R4= X4 ;;GENERAL REGISTER
(1) 000005 R5= X5 ;;GENERAL REGISTER
(1) 000006 R6= X6 ;;GENERAL REGISTER
(1) 000007 R7= X7 ;;GENERAL REGISTER
(1) 000006 SP= X6 ;;STACK POINTER
(1) 000007 PC= X7 ;;PROGRAM COUNTER
(1)
(1) ;*PRIORITY LEVEL DEFINITIONS
(1) 000000 PR0= 0 ;;PRIORITY LEVEL 0
(1) 000040 PR1= 40 ;;PRIORITY LEVEL 1
(1) 000100 PR2= 100 ;;PRIORITY LEVEL 2
(1) 000140 PR3= 140 ;;PRIORITY LEVEL 3
(1) 000200 PR4= 200 ;;PRIORITY LEVEL 4
(1) 000240 PR5= 240 ;;PRIORITY LEVEL 5
(1) 000300 PR6= 300 ;;PRIORITY LEVEL 6
(1) 000340 PR7= 340 ;;PRIORITY LEVEL 7
(1)
(1) ;*'SWITCH REGISTER' SWITCH DEFINITIONS
(1) 100000 SW15= 100000
(1) 040000 SW14= 40000
(1) 020000 SW13= 20000
(1) 010000 SW12= 10000
(1) 004000 SW11= 4000
(1) 002000 SW10= 2000
(1) 001000 SW09= 1000
(1) 000400 SW08= 400
(1) 000200 SW07= 200
(1) 000100 SW06= 100
(1) 000040 SW05= 40
(1) 000020 SW04= 20
(1) 000010 SW03= 10
(1) 000004 SW02= 4
(1) 000002 SW01= 2
(1) 000001 SW00= 1
(1) .EQUIV SW09,SW9
(1) .EQUIV SW08,SW8
(1) .EQUIV SW07,SW7
(1) .EQUIV SW06,SW6
(1) .EQUIV SW05,SW5
(1) .EQUIV SW04,SW4
(1) .EQUIV SW03,SW3
(1) .EQUIV SW02,SW2
    
```

```

(1) .EQUIV SW01,SW1
(1) .EQUIV SW00,SW0
(1)
(1) ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1) 100000 BIT15= 100000
(1) 040000 BIT14= 40000
(1) 020000 BIT13= 20000
(1) 010000 BIT12= 10000
(1) 004000 BIT11= 4000
(1) 002000 BIT10= 2000
(1) 001000 BIT09= 1000
(1) 000400 BIT08= 400
(1) 000200 BIT07= 200
(1) 000100 BIT06= 100
(1) 000040 BIT05= 40
(1) 000020 BIT04= 20
(1) 000010 BIT03= 10
(1) 000004 BIT02= 4
(1) 000002 BIT01= 2
(1) 000001 BIT00= 1
(1) .EQUIV BIT09,BIT9
(1) .EQUIV BIT08,BIT8
(1) .EQUIV BIT07,BIT7
(1) .EQUIV BIT06,BIT6
(1) .EQUIV BIT05,BIT5
(1) .EQUIV BIT04,BIT4
(1) .EQUIV BIT03,BIT3
(1) .EQUIV BIT02,BIT2
(1) .EQUIV BIT01,BIT1
(1) .EQUIV BIT00,BIT0
(1)
(1) ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1) 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
(1) 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
(1) 000014 TBITVEC=14 ;: "T" BIT
(1) 000014 TRTVEC= 14 ;:TRACE TRAP
(1) 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
(1) 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1) 000024 PWRVEC= 24 ;:POWER FAIL
(1) 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
(1) 000034 TRAPVEC=34 ;:"TRAP" TRAP
(1) 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
(1) 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
(1) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
1180 170460 ABASE= 170460
1181 000300 AVECT1= 300
1182 000001 $TN=1
1183 .SBTTL ACT11 HOOKS
(1)
(2) ;*****
(1) ;HOOKS REQUIRED BY ACT11
(1) 000244 $SVPC= ;SAVE PC
(1) 000046 .=46 ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SECP
(1) 000046 031030 $ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SECP
(1) 000052 .=52 ;:2)SET LOC.52 TO ZERO
(1) 000052 000000 .WORD 0
  
```

```

(1)          000244          .=$SVPC          ;; RESTORE PC
1184
1185          001000          .=1000
1186          .SBTTL  APT PARAMETER BLOCK
(1)
(2)          ;*****
(1)          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)          ;*****
(1)          001000          .$X=.          ;;SAVE CURRENT LOCATION
(1)          000024          .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200          200          ;;FOR APT START UP
(1)          000044          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000          $APTHDR ;;POINT TO APT HEADER BLOCK
(1)          001000          .=.$X          ;;RESET LOCATION COUNTER
(2)          ;*****
(1)          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)          ;INTERFACE SPEC.
(1)
(1) 001000          $APTHD:
(1) 001000 000000          $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001174          $MBADR: .WORD $MAIL          ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 000002          $TSTM: .WORD 2          ;;RUN TIM OF LONGEST TEST
(1) 001006 000170          $PASTM: .WORD 120.          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000170          $UNITM: .WORD 120.          ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000031          .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
  
```


(2)	001212	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
(2)	001214		\$ETABLE:		::APT ENVIRONMENT TABLE
(2)	001214	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
(2)	001215	000	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
(2)	001216	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
(2)	001220	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
(2)	001222	000000	\$CPUOP: .WORD	ACPUOP	::CPU TYPE,OPTIONS
(2)			.*		BITS 15-11=CPU TYPE
(2)			.*		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)			.*		11/70=06,PDQ=07,Q=10
(2)			.*		BIT 10=REAL TIME CLOCK
(2)			.*		BIT 9=FLOATING POINT PROCESSOR
(2)			.*		BIT 8=MEMORY MANAGEMENT
(2)	001224	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
(2)	001225	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
(2)			.*		MEM.TYPE BYTE -- (HIGH BYTE)
(2)			.*		900 NSEC CORE=001
(2)			.*		300 NSEC BIPOLAR=002
(2)			.*		500 NSEC MOS=003
(2)	001226	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
(2)			.*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2)	001230	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
(2)	001231	000	\$MTYP2: .BYTE	AMTYP2	::MEM.TYPE,BLK#2
(2)	001232	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
(2)	001234	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
(2)	001235	000	\$MTYP3: .BYTE	AMTYP3	::MEM.TYPE,BLK#3
(2)	001236	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
(2)	001240	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
(2)	001241	000	\$MTYP4: .BYTE	AMTYP4	::MEM.TYPE,BLK#4
(2)	001242	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
(2)	001244	000300	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)	001246	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
(2)	001250	170460	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)	001252	000000	\$DEVN: .WORD	ADEVN	::DEVICE MAP
(2)	001254	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
(2)	001256		\$ETEND:		
(2)			.MEXIT		

```

(1) .SBTTL ERROR POINTER TABLE
(1)
(1) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) ;* EM ;:POINTS TO THE ERROR MESSAGE
(1) ;* DH ;:POINTS TO THE DATA HEADER
(1) ;* DT ;:POINTS TO THE DATA
(1) ;* DF ;:POINTS TO THE DATA FORMAT
(1)
(1) $ERRTB:
(1) 001256
1188
1192
1193 ;ITEM 1
1194
1195 001256 041414 EM1 ;LPA-11 ADDRESS ER OR
1196 001260 042150 DH1 ;ERRPC ADDRESS
1197 001262 042426 DT1 ;$ERRPC, KMADO
1198 001264 042516 DF0 ;ALL NUMBER ARE IN OCTAL FORM.
1199
1200 ;ITEM 2
1201
1202 001266 041437 EM2 ;LPA(KMC-11) DATA ERROR
1203 001270 042171 DH2 ;ERRPC EXP'D REC'D
1204 001272 042434 DT2 ;$ERRPC, $GDDAT, $BDDAT
1205 001274 042516 DF0 ;ALL NUMBER ARE IN OCTAL FORM.
1206
1207 ;ITEM 3
1208
1209 001276 041470 EM3 ;LPA (KMC-11) INSTRUCTION ERROR
1210 001300 042222 DH3 ;ERRPC
1211 001302 042444 DT3 ;$ERRPC
1212 001304 042516 DF0 ;ALL NUMBER ARE IN OCTAL FORM.
1213
1214 ;ITEM 4
1215
1216 001306 041530 EM4 ;LPA (M8254) INIT.ERROR
1217 001310 042222 DH3 ;ERRPC
1218 001312 042444 DT3 ;$ERRPC
1219 001314 042516 DF0 ;ALL NUMBER ARE IN OCTAL FORM.
1220
1221 ;ITEM 5
1222
1223 001316 041530 EM4 ;LPA (M8254) INIT.ERROR
1224 001320 042171 DH2 ;ERRPC EXP'ED REC'ED
1225 001322 042434 DT2 ;$ERRPC, $GDDAT, $BDDAT
1226 001324 042516 DF0 ;ALL NUMBER ARE IN OCTAL FORM.
1227
1228 ;ITEM 6
1229
1230 001326 041560 EM6 ;LPA (M8254) SILO DATA ERROR
1231 001330 042171 DH2 ;ERRPC EXP'ED REC'ED

```

1232	001332	042434	DT2	:SERRPC,\$GDDAT,\$BDDAT
1233	001334	042516	DF0	:ALL NUMBER ARE IN OCTAL FORM.
1234				
1235			:ITEM 7	
1236				
1237	001336	041615	EM7	:LPA (M8254) FAST PATH DATA ERROR
1238	001340	042171	DH2	:ERRPC EXP'ED REC'ED
1239	001342	042434	DT2	:SERRPC,\$GDDAT,\$BDDAT
1240	001344	042516	DF0	:ALL NUMBER ARE IN OCTAL FORM.
1241				
1242			:ITEM 10	
1243				
1244	001346	041657	EM10	:LPA (AD11K) CSR ERROR
1245	001350	042171	DH2	:ERRPC EXP'ED REC'ED
1246	001352	042434	DT2	:SERRPC,\$GDDAT,\$BDDAT
1247	001354	042516	DF0	:ALL NUMBER ARE IN OCTAL FORM.
1248				
1249			:ITEM 11	
1250				
1251	001356	041706	EM11	:LPA (KW11K) CSR ERROR
1252	001360	042171	DH2	:ERRPC EXP'ED REC'ED
1253	001362	042434	DT2	:SERRPC,\$GDDAT,\$BDDAT
1254	001364	042516	DF0	:ALL NUMBER ARE IN OCTAL FORM.
1255				
1256			:ITEM 12	
1257				
1258	001366	041735	EM12	:LPA (DR11K) ERROR
1259	001370	042171	DH2	:ERRPC EXP'ED REC'ED
1260	001372	042434	DT2	:SERRPC,\$GDDAT,\$BDDAT
1261	001374	042516	DF0	:ALL NUMBER ARE IN OCTAL FORM.
1262				
1263			:ITEM 13	
1264				
1265	001376	000000	0	:NO-ERROR 13 (BAD LUCK!)
1266	001400	000000	0	
1267	001402	000000	0	
1268	001404	000000	0	
1269				
1270			:ITEM 14	
1271				
1272	001406	041760	EM14	:LPA (AA-11K) ERROR
1273	001410	042171	DH2	:ERRPC EXP'ED REC'ED
1274	001412	042434	DT2	:SERRPC,\$GDDAT,\$BDDAT
1275	001414	042516	DF0	:ALL NUMBER ARE IN OCTAL FORM.
1276				
1277			:ITEM 15	
1278				
1279	001416	042004	EM15	:LPA-11 (AR11K) ERROR
1280	001420	042171	DH2	:ERRPC EXP'ED REC'ED
1281	001422	042434	DT2	:SERRPC,\$GDDAT,\$BDDAT
1282	001424	042516	DF0	:ALL NUMBER ARE IN OCTAL FORM.
1283				
1284			:ITEM 16	
1285				
1286	001426	042026	EM16	:LPA-11 (LPS-11) ERROR
1287	001430	042171	DH2	

```

1288 001432 042434      DT2
1289 001434 042516      DF0      ;ALL NUMBER ARE IN OCTAL FORM.
1290
1291      ;ITEM 17
1292
1293 001436 042052      EM17      ;LPA-11 FUNCTIONAL ERROR
1294 001440 042231      DH17
1295 001442 042450      DT17
1296 001444 042516      DF0      ;ALL NUMBER ARE IN OCTAL FORM.
1297
1298      ;ITEM 20
1299
1300 001446 042103      EM20
1301 001450 042271      DH20
1302 001452 042462      DT20
1303 001454 042516      DF0      ;ALL NUMBER ARE IN OCTAL FORM.
1304
1305      ;ITEM 21
1306 001456 042052      EM17      ;LPA FUNCTIONAL ERROR
1307 001460 042337      DH21      ;ERRPC TSTNO USJNO USRDA ALPCO ALPCI ALPSO
1308 001462 042476      DT21      ;ERRPC,$TESTN,USJNO,USRDA,ALPCO,ALPCI,APSO
1309 001464 042516      DF0      ;ALL NUMBER ARE IN OCTAL FORM.
1310
1311      ;DEVICE ADDRESSES AS USED BY PROGRAM.
1312      ;DO NOT "PATCH" THESE LOCATIONS
1313      ;LOCATIONS ARE ALTERED BY PROGRAM BASED ON YOUR INFO IN $BASE
1314
1315 001466      KMCSR:
1316      ;
(1)      ;ADDRESS OF KMC-11 OF LPA-11      THE ADDR FOR KMADO MAY BE
(1)      ;                               CHANGED BY THE USER TO REFLECT
(1)      ;                               A DIFFERENT KMC-11 ADDR. THE
(1)      ;                               REST OF THE ADDRESSES WILL
(1)      ;                               BE CHANGED BY THE PROGRAM.
(1)      ;
(1)      ;
(1)      ;
(1) 001466      LPCI:
(1) 001466 170460      KMADO: .WORD  ABASE      ;BASE KMC ADDR. MAY BE PATCHED BY USER.
(1)      ;
(1) 001470      LPMR:
(1) 001470 170461      KMAD1: .WORD  ABASE+1      ;>DO NOT      <;KMC-CSR ADDR
(1) 001472      LPCO:
(1) 001472 170462      KMAD2: .WORD  ABASE+2      ;>PATCH      <;
(1) 001474      LPSO:
(1) 001474 170463      KMAD3: .WORD  ABASE+3      ;>THIS AREA      <
(1) 001476      LPADL:
(1) 001476 170464      KMAD4: .WORD  ABASE+4      ;
(1) 001500      LPADH:
(1) 001500 170465      KMAD5: .WORD  ABASE+5      ;>DO NOT      <
(1) 001502      LPMS1:
(1) 001502 170466      KMAD6: .WORD  ABASE+6      ;>PATCH      <
(1) 001504      LPMS2:
(1) 001504 170467      KMAD7: .WORD  ABASE+7      ;>THIS AREA      <
(1)      ;
(1) 001506 000300      VECTOR: .WORD  AVECT18777      ;BASE VECTOR OF KMC
(1) 001510 000304      VECTPS: .WORD  4+AVECT18777      ;VECTR ADDR.+2
    
```

(1)
(1) 001512 000005 VERSN: .WORD 5 ;CURRENT VERSION NUMBER OF MICROCODE.
(1)
(1) 001514 000000 .DVLS: .WORD 0 ;/DEVICE LIST OF I/O ADDR. DEFINED
(1) 001516 000020 .BLKW 16. ;/BY INIT.

(1)
1317
1318
1319 001556 000300 VECT1: .WORD AVECT1
1320
1321 001560 000304 VECT2: .WORD AVECT1+4
1322
1323 .SBTTL *****
1324 .SBTTL SR1 DEVICE PRESENT REG
1325 .SBTTL *****
1326
1327 .REM %
1328
1329 SR1 INFORMS THIS DIAGNOSTIC AS TO WHAT DEVICES
1330 ARE ON THE I/O BUS.
1331 SR1 DEFAULTS TO 1 KW11K, AND 1 AD11K.
1332 IF YOUR CONFIGURATION IS DIFFERENT, YOU MUST CHANGE
1333 THIS LOCATION

WORD BIT=1	OCTAL	DEVICE
0	000001	1ST AD11K
1	000002	1ST KW11K
2	000004	1ST DR11K
3	000010	1ST AA11K
4	000020	2ND AD11K#2
5	000040	2ND DR11K
6	000100	RESERVED
7	000200	3RD DR11K
8	000400	4TH DR11K
9	001000	5TH DR11K
10	002000	AR11
11	004000	RESERVED
12	010000	LPSAD (LPS A/D)
13	020000	LPSKW (LPS REAL TIME CLOCK)
14	040000	LPSVC (LPS D/A)
15	100000	LPSDR (LPS DIGITAL J/O)

1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357 001562 000003 SR1: .WORD 003 ;DEVICE PRESENT FLAG DEFAULT
1358 ;IS 1 KW11K, 1 AD11K
1359 .SBTTL *****
1360 .SBTTL SR2 DEVICE SET-UP REG
1361 .SBTTL *****
1362
1363 .REM %
1364
1365 SR2 INFORMS THIS DIAGNOSTIC AS TO HOW THE
1366 DEVICES SELECTED BY SR1 ARE SET-UP FOR TEST.

WHILE NO SPECIAL SETUP IS REQUIRED TO RUN THIS TEST,
 THE DEPTH OF COVERAGE WILL INCREASE IF SPECIAL SETUPS
 ARE PERFORMED.

WORD BIT=1	OCTAL	FUNCTION
0	000001	AD11K HAS G5034 WRAP-AROUND MODULE.
2	000004	DR11K #1 HAS LOOP BACK CABLE
3	000010	AA11K HAS SCOPE DISPLAY (VISUAL TEST)
4	000020	AD11K #2 HAS G5034 WRAP AROUND MODULE.
5	000040	DR11K #2 HAS LOOPBACK CABLE
7	000200	DR11K #3 HAS LOOP BACK CABLE.
8	000400	DR11K #4 HAS LOOP BACK CABLE.
9	001000	DR11K #5 HAS LOOP BACK CABLE.
10	002000	AR11 HAS G5034 WRAP AROUND MODULE.
11	004000	AR11 HAS SCOPE DISPLAY (VISUAL TEST)
14	040000	LPS-11 D/A HAS SCOPE DISPLAY (VISUAL TEST)
15	100000	LPS-11 DIGITAL I/O HAS LOOP BACK CABLE

1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386

1387
1388 001564 000000

SR2: .WORD 0

1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399

.SBTTL *****
 .SBTTL LPA-11 I/O BUS DEVICE ADDRESS DEFINITION AND LIST
 .SBTTL *****

.REM %
 THE FOLLOWING ARE A LIST OF DEVICE ADDRESSES. YOU MAY
 CHANGE THE ADDRESS OF A DEVICE ONLY BY MODIFYING THE FOLLOWING LIST.
 DO NOT CHANGE ANY OTHER ADDRESSES!
 %

1400 001566 170400
 1401 001570 170404
 1402 001572 167770
 1403 001574 170416
 1404 001576 170440
 1405 001600 167760
 1406 001602 167750
 1407 001604 167740
 1408 001606 167730
 1409 001610 170400
 1410 001612 170400

AD11K: .WORD 170400 ;AD11K ADDRESS.
 KW11K: .WORD 170404 ;KW11K ADDRESS.
 DR11K1: .WORD 167770 ;DR11K #1 ADDR.
 AA11K: .WORD 170416 ;AA11 ADDRESS.
 AD11K2: .WORD 170440 ;AD11K #2 ADDRESS.
 DR11K2: .WORD 167760 ;DR11K #2 ADDRESS.
 DR11K3: .WORD 167750 ;DR11K #3 ADDRESS.
 DR11K4: .WORD 167740 ;DR11K #4 ADDRESS.
 DR11K5: .WORD 167730 ;DR11K #5 ADDRESS.
 AR11K: .WORD 170400 ;AR11 ADDRESS.
 LPS11: .WORD 170400 ;LPS11 BASE ADDRESS.

1411
1412
1413
1414
1415
1416
1417
1418

.REM \$
 END OF PATCHABLE DEVICE ADDRESSES.

\$
 .SBTTL AD11-K ADDRESSES

1419 001614 170400
 1420
 1421 001616
 1422 001616 170402

STREG1: .WORD 170400 ;CONTROL AND STATUS REGISTER.
 ADBUF1: ;ADDRESS OF BUFFER REG. AND
 ADAC: .WORD 170402 ;DIGITAL OUTPUT REG.

```

1423 001620 170440 STREG2: .WORD 170440 ;AD11K #2 CSR
1424 001622 ADBUF2:
1425 001622 170442 ADAC2: .WORD 170442 ;AD11K #2 OUTPUT/INPUT
1426 .SBTTL AA11-K ADDRESSES
1427
1428 001624 170416 AACSR: .WORD 170416 ;AA11K CSR
1429 001626 170420 DAC0: .WORD 170420
1430 001630 170422 DAC1: .WORD 170422
1431 001632 170424 DAC2: .WORD 170424
1432 001634 170426 DAC3: .WORD 170426
1433
1441
1442 .SBTTL DR11K #1 ADDRESS
(1) 001636 167770 DRCR1: .WORD 167770 ;DR11K #1 CSR.
(1) 001640 167774 DROA1: .WORD 167770+4 ;DR11K OUTPUT REG.
(1) 001642 167772 DRIA1: .WORD 167770+2 ;DR11K INPUT REG.
(1)
1443 .SBTTL DR11K #2 ADDRESS
(1) 001644 167760 DRCR2: .WORD 167760 ;DR11K #2 CSR.
(1) 001646 167764 DROA2: .WORD 167760+4 ;DR11K OUTPUT REG.
(1) 001650 167762 DRIA2: .WORD 167760+2 ;DR11K INPUT REG.
(1)
1444 .SBTTL DR11K #3 ADDRESS
(1) 001652 167750 DRCR3: .WORD 167750 ;DR11K #3 CSR.
(1) 001654 167754 DROA3: .WORD 167750+4 ;DR11K OUTPUT REG.
(1) 001656 167752 DRIA3: .WORD 167750+2 ;DR11K INPUT REG.
(1)
1445 .SBTTL DR11K #4 ADDRESS
(1) 001660 167740 DRCR4: .WORD 167740 ;DR11K #4 CSR.
(1) 001662 167744 DROA4: .WORD 167740+4 ;DR11K OUTPUT REG.
(1) 001664 167742 DRIA4: .WORD 167740+2 ;DR11K INPUT REG.
(1)
1446 .SBTTL DR11K #5 ADDRESS
(1) 001666 167730 DRCR5: .WORD 167730 ;DR11K #5 CSR.
(1) 001670 167734 DROA5: .WORD 167730+4 ;DR11K OUTPUT REG.
(1) 001672 167732 DRIA5: .WORD 167730+2 ;DR11K INPUT REG.
(1)
1447
1448
1449 .SBTTL KW11K ADDRESS
1450
1451 001674 170404 KWADR0: .WORD 170404 ;CLOCK A CSR
1452
1453 001676 170406 KWADR1: .WORD 170406 ;CLOCK A PRESENT REG.
1454
1455 001700 170430 KWADR2: .WORD 170430 ;CLOCK A COUNTER REG.
1456
1457 001702 170432 KWADR4: .WORD 170432 ;CLOCK B CSR
1458
1459 001704 170434 KWADR5: .WORD 170434 ;CLOCK B PRESENT REG.
1460
1461 001706 170436 KWADR6: .WORD 170436 ;CLOCK B COUNT REG.
1462
1463 .SBTTL AR11 ADDRESSES
1464
1465

```

1466	001710	170400	ARADS:	.WORD	170400	;A/D STATUS REG
1467	001712	170402	ARADB:	.WORD	170402	;A/D BUFFER
1468	001714	170404	ARCS:	.WORD	170404	;CLOCK STATUS
1469	001716	170406	ARCB:	.WORD	170406	;CLOCK BUFFER/PRESET
1470	001720	170410	ARDS:	.WORD	170410	;DISPLAY STATUS
1471	001722	170412	ARXB:	.WORD	170412	;X BUFFER
1472	001724	170414	ARYB:	.WORD	170414	;Y BUFFER
1473	001726	170416	ARCC:	.WORD	170416	;CLOCK COUNT REG.

1474
1475
1476
1477

.SBTTL LPS-11 ADDRESSES

1478	001730	170400	LPADSR:	.WORD	170400	;A/D STATUS REG.
1479	001732	170402	LPADBR:	.WORD	170402	;A/D BUFFER REG.
1480	001734	170404	LPCKSR:	.WORD	170404	;CLOCK STATUS REG.
1481	001736	170406	LPCKBR:	.WORD	170406	;CLOCK BUFFER REG.
1482	001740	170410	LPIOSR:	.WORD	170410	;I/O STATUS REG.
1483	001742	170412	LPIOIR:	.WORD	170412	;I/O INPUT REG.
1484	001744	170414	LPIOOR:	.WORD	170414	;I/O OUTPUT REG.
1485	001746	170416	LPDASR:	.WORD	170416	;D/A STATUS REG.
1486	001750	170420	LPDAXR:	.WORD	170420	;D/A X REG.
1487	001752	170422	LPDAYR:	.WORD	170422	;D/A Y REG.

1488
1489
1490

;MISC. STORAGE LOCATIONS

1491	001754	000000	MYTEMP:	.WORD	0	;TEMP STORAGE
1492	001756	000000	USJNO:	.WORD	0	;USERS JOB NUMBER (IN R FLASH)
1493	001760	000000	USRDA:	.WORD	0	;USERS RDA ADDR.
1494	001762	000000	ALPLI:	.WORD	0	;CONTENTS OF CONTROL IN REG.
1495	001764	000000	ALPCO:	.WORD	0	;CONTENTS OF CONTROL OUT REG.
1496	001766	000000	ALPSO:	.WORD	0	;CONTENTS OF HIGH BYT OF ABOVE,(STATUS REG)
1497	001770	000000	ALPADL:	.WORD	0	;CONTENTS OF ADDR. REG.
1498	001772	000	\$VERSN:	.BYTE	0,3	;CONTAIN UCODE SYMBOLS
1499	001774	000000	TAXING:	.WORD	0	
1500	001776	000000	ERCNT:	.WORD	0	;NO OF ERRORS.

1501
1502
1503
1504

;TO BE KW11P ADDR. OR NO ADDR. AT ALL

1505		000073			STPCOD=73	
1506		000074			LOPC=74	
1507		000007			VBM=7	
1508		000007			USWH=7	
1509	002000	177546	KW11L:	.WORD	177546	;ADDR. OF KW11L
1510	002002	172540	KW11P:	.WORD	172540	;ADDR. OF KW11P.
1511						
1512	002004	002006	NOCLK:	.WORD	CLKTMP	;ADDR. OF NO CLOCK.
1513	002006	000000	CLKTMP:	.WORD	0	
1514	002010	000100	KWVECL:	.WORD	100	;VECTR OF KW11L
1515	002012	000104	KWVECP:	.WORD	104	;VECTOR OF KW11L
1516	002014	000000	FUDGE:	.WORD	0	;FUDGE FACTOR

1517
1519
1520
(1)
(1)

START:
 .SBTTL INITIALIZE THE COMMON TAGS
 ;:CLEAR THE COMMON TAGS (\$CMTAG) AREA


```

(1) 002016 012706 001100      MOV    #SCMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
(1) 002022 005026             CLR    (R6)+           ;;CLEAR MEMORY LOCATION
(1) 002024 022706 001140      CMP    #SWR,R6 ;;DONE?
(1) 002030 001374             BNE   -6               ;;LOOP BACK IF NO
(1) 002032 012706 001100      MOV    #STACK,SP      ;;SETUP THE STACK POINTER
(1)                               ;;INITIALIZE A FEW VECTORS
(1) 002036 012737 032052 000020  MOV    $$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 002044 012737 000340 000022  MOV    #340,@#IOTVEC+2 ;;LEVEL 7
(1) 002052 012737 031522 000030  MOV    #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 002060 012737 000340 000032  MOV    #340,@#EMTVEC+2 ;;LEVEL 7
(1) 002066 012737 041152 000034  MOV    #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 002074 012737 000340 000036  MOV    #340,@#TRAPVEC+2;LEVEL 7
(1) 002102 012737 033742 000024  MOV    #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1) 002110 012737 000340 000026  MOV    #340,@#PWRVEC+2 ;;LEVEL 7
(1) 002116 005037 001160             CLR    $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
(1) 002122 005037 001162             CLR    $ESCAPE        ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 002126 112737 000001 001115  MOV    #1,$ERMAX      ;;ALLOW ONE ERROR PER TEST
(1) 002134 012737 002134 001106  MOV    #,$SLPADR      ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002142 012737 002142 001110  MOV    #,$SLPERR      ;;SETUP THE ERROR LOOP ADDRESS
(2)                               ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2)                               ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 002150 013746 000004             MOV    @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
(2) 002154 012737 002210 000004  MOV    #64$,@#ERRVEC  ;;SET UP ERROR VECTOR
(2) 002162 012737 177570 001140  MOV    #DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 002170 012737 177570 001142  MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 002176 022777 177777 176734  CMP    #-1,@SWR       ;;TRY TO REFERENCE HARDWARE SWR
(2) 002204 001012             BNE   66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)                               ;;AND THE HARDWARE SWR IS NOT = -1
(2) 002206 000403             BR    65$             ;;BRANCH IF NO TIMEOUT
(2) 002210 012716 002216 64$:      MOV    #65$,(SP)      ;;SET UP FOR TRAP RETURN
(2) 002214 000002             RTI
(2) 002216 012737 000176 001140 65$:  MOV    #SWREG,SWR     ;;POINT TO SOFTWARE SWR
(2) 002224 012737 000174 001142  MOV    #DISPREG,DISPLAY
(2) 002232 012637 000004 66$:  MOV    (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 002236 005037 001202             CLR    $PASS          ;;CLEAR PASS COUNT
(2) 002242 132737 000200 001215  BITB  #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
(2) 002250 001403             BEQ   67$            ;;YES,USE NON-APT SWITCH
(2) 002252 012737 001216 001140  MOV    #$$SWREG,SWR   ;;NO,USE APT SWITCH REGISTER
(2) 002260 67$:
1521
1522 002260 013700 001250             MOV    $BASE,R0       ;;GET BASE KMC-ADDR.
1523 002264 012701 001466             MOV    #KMAD0,R1     ;;STORAGE OF ADDR. LIST
1524 002270 010021 1$:      MOV    R0,(1)+        ;;STORE ADDR.
1525 002272 005200             INC    R0             ;;UPDATE ADDR.
1526 002274 020127 001506             CMP    R1,#KMAD7+2   ;;DONE WHOLE LIST?
1527 002300 001373             BNE   1$
1528
1529 002302 005037 001514             CLR    .DVLS
1530
1531 002306 013737 001244 001556  MOV    $VECT1,VECT1   ;;GET VECTOR ADDR.
1532 002314 042737 170000 001556  BIC    #170000,VECT1
1533 002322 013737 001556 001560  MOV    VECT1,VECT2
1534 002330 052737 000004 001560  BIS    #4,VECT2
1535
1536                               ;;THE FOLLOWING SECTION OF CODE SETS UP DEVICE ADDRESSES
    
```

```

1537                                     ;BASED ON DEFAULT OR OPERATOR MODIFIED DATA IN LOCATIONS
1538                                     ;"AD11K" THROUGH "LPS11"
1539                                     ;
1540 002336 013700 001566                MOV     AD11K,RO           ;FIX AD11K'S ADDRESS
1541 002342 010037 001614                MOV     RO,STREG1
1542 002346 010037 001616                MOV     RO,ADAC
1543 002352 062737 000002 001616        ADD     #2,ADAC
1544
1545 002360 013700 001570                MOV     KW11K,RO         ;FIX KW11K ADDRESS
1546 002364 010037 001674                MOV     RO,KWADR0
1547 002370 010037 001676                MOV     RO,KWADR1
1548 002374 010037 001700                MOV     RO,KWADR2
1549 002400 010037 001702                MOV     RO,KWADR4
1550 002404 010037 001704                MOV     RO,KWADR5
1551 002410 010037 001706                MOV     RO,KWADR6
1552 002414 062737 000002 001676        ADD     #2,KWADR1
1553 002422 062737 000024 001700        ADD     #24,KWADR2
1554 002430 062737 000026 001702        ADD     #26,KWADR4
1555 002436 062737 000030 001704        ADD     #30,KWADR5
1556 002444 062737 000032 001706        ADD     #32,KWADR6
1557
1567
1568 (1) 002452 013700 001572                MOV     DR11K1,RO       ;FIX DR11K #1 ADDRESS
1568 (1) 002456 010037 001636                MOV     RO,DRCR1
1568 (1) 002462 010037 001640                MOV     RO,DROA1
1568 (1) 002466 010037 001642                MOV     RO,DRIA1
1568 (1) 002472 062737 000004 001640        ADD     #4,DROA1
1568 (1) 002500 062737 000002 001642        ADD     #2,DRIA1
1569
1569 (1) 002506 013700 001600                MOV     DR11K2,RO       ;FIX DR11K #2 ADDRESS
1569 (1) 002512 010037 001644                MOV     RO,DRCR2
1569 (1) 002516 010037 001646                MOV     RO,DROA2
1569 (1) 002522 010037 001650                MOV     RO,DRIA2
1569 (1) 002526 062737 000004 001646        ADD     #4,DROA2
1569 (1) 002534 062737 000002 001650        ADD     #2,DRIA2
1570
1570 (1) 002542 013700 001602                MOV     DR11K3,RO       ;FIX DR11K #3 ADDRESS
1570 (1) 002546 010037 001652                MOV     RO,DRCR3
1570 (1) 002552 010037 001654                MOV     RO,DROA3
1570 (1) 002556 010037 001656                MOV     RO,DRIA3
1570 (1) 002562 062737 000004 001654        ADD     #4,DROA3
1570 (1) 002570 062737 000002 001656        ADD     #2,DRIA3
1571
1571 (1) 002576 013700 001604                MOV     DR11K4,RO       ;FIX DR11K #4 ADDRESS
1571 (1) 002602 010037 001660                MOV     RO,DRCR4
1571 (1) 002606 010037 001662                MOV     RO,DROA4
1571 (1) 002612 010037 001664                MOV     RO,DRIA4
1571 (1) 002616 062737 000004 001662        ADD     #4,DROA4
1571 (1) 002624 062737 000002 001664        ADD     #2,DRIA4
1572
1572 (1) 002632 013700 001606                MOV     DR11K5,RO       ;FIX DR11K #5 ADDRESS
1572 (1) 002636 010037 001666                MOV     RO,DRCR5
1572 (1) 002642 010037 001670                MOV     RO,DROA5
1572 (1) 002646 010037 001672                MOV     RO,DRIA5
1572 (1) 002652 062737 000004 001670        ADD     #4,DROA5

```

(1)	002660	062737	000002	001672	ADD	#2,DRIA5	
1573							
1574	002666	013700	001574		MOV	AA11K,RO	;FIX AA11K ADDRESS
1575	002672	010037	001624		MOV	RO,AACSR	
1576	002676	010037	001626		MOV	RO,DACO	
1577	002702	010037	001630		MOV	RO,DAC1	
1578	002706	010037	001632		MOV	RO,DAC2	
1579	002712	010037	001634		MOV	RO,DAC3	
1580	002716	062737	000002	001626	ADD	#2,DACO	
1581	002724	062737	000004	001630	ADD	#4,DAC1	
1582	002732	062737	000006	001632	ADD	#6,DAC2	
1583	002740	062737	000010	001634	ADD	#10,DAC3	
1584							
1585	002746	013700	001576		MOV	AD11K2,RO	;FIX SECOND AD11K'S ADDR.
1586	002752	010037	00162C		MOV	RO,STREG2	
1587	002756	010037	001622		MOV	RO,ADAC2	
1588	002762	062737	000002	001622	ADD	#2,ADAC2	
1589							
1590	002770	013700	001610		MOV	AR11K,RO	;FIX AR11 ADDR.
1591	002774	010037	001710		MOV	RO,ARADS	
1592	003000	010037	001712		MOV	RO,ARADB	
1593	003004	010037	001714		MOV	RO,ARCS	
1594	003010	010037	001716		MOV	RO,ARCB	
1595	003014	010037	001720		MOV	RO,ARDS	
1596	003020	010037	001722		MOV	RO,ARXB	
1597	003024	010037	001724		MOV	RO,ARYB	
1598	003030	010037	001726		MOV	RO,ARCC	
1599	003034	062737	000002	001712	ADD	#2,ARADB	
1600	003042	062737	000004	001714	ADD	#4,ARCS	
1601	003050	062737	000006	001716	ADD	#6,ARCB	
1602	003056	062737	000010	001720	ADD	#10,ARDS	
1603	003064	062737	000012	001722	ADD	#12,ARXB	
1604	003072	062737	000014	001724	ADD	#14,ARYB	
1605	003100	062737	000016	001726	ADD	#16,ARCC	
1606							
1607	003106	013700	001612		MOV	LPS11,RO	;FIX LPS-11 ADDRESSES
1608	003112	010037	001730		MOV	RO,LPADSR	
1609	003116	010037	001732		MOV	RO,LPADBR	
1610	003122	010037	001734		MOV	RO,LPCKSR	
1611	003126	010037	001736		MOV	RO,LPCKBR	
1612	003132	010037	001740		MOV	RO,LPIOSR	
1613	003136	010037	001742		MOV	RO,LPIOIR	
1614	003142	010037	001744		MOV	RO,LPIOOR	
1615	003146	010037	001746		MOV	RO,LPDASR	
1616	003152	010037	001750		MOV	RO,LPDAXR	
1617	003156	010037	001752		MOV	RO,LPDAYR	
1618	003162	062737	000002	001732	ADD	#2,LPADBR	
1619	003170	062737	000004	001734	ADD	#4,LPCKSR	
1620	003176	062737	000006	001736	ADD	#6,LPCKBR	
1621	003204	062737	000010	001740	ADD	#10,LPIOSR	
1622	003212	062737	000012	001742	ADD	#12,LPIOIR	
1623	003220	062737	000014	001744	ADD	#14,LPIOOR	
1624	003226	062737	000016	001746	ADD	#16,LPDASR	
1625	003234	062737	000020	001750	ADD	#20,LPDAXR	
1626	003242	062737	000022	001752	ADD	#22,LPDAYR	
1627	003250	005037	002014		CLR	FUDGE	

```

1628
1629
1630 003254 012737 003276 000004      MOV    #10$,ERRVEC      ;SET UP FOR TIME-OUT IF NO KW11L
1631 003262 005777 176512                TST    @KW11L           ;TRAP HERE IF NO L
1632 003266 013737 002000 040534      MOV    KW11L,RTCCSR    ;MUST BE HERE,RECORD ADDR.
1633 003274 000414                BR     50$
1634 003276 012737 003320 000004 10$:   MOV    #20$,ERRVEC    ;SET UP FOR TIME-OUT IF NO KW11P
1635 003304 005777 176472                TST    @KW11P          ;TRAP HERE IF NO P
1636 003310 013737 002002 040534      MOV    KW11P,RTCCSR    ;MUST BE HERE,RECORD ADDR.
1637 003316 000403                BR     50$
1638 003320 013737 002004 040534 20$:   MOV    NOCLK,RTCCSR    ;NO CLOCK! OHOH.
1639 003326 012706 001100 50$:   MOV    #STACK,SP      ;RESTORE STACK
1640 003332 012737 000006 000004      MOV    #6,ERRVEC      ;RESTORE ERROR VECTOR.
1641
1642 003340 012777 040526 176442      MOV    #CLKINT,@KWVECL ;SET UP CLOCK VECTOR.
1643 003346 012777 040526 176436      MOV    #CLKINT,@KWVECP
1644
(1)                                     .SBTTL  TYPE PROGRAM NAME
(1)                                     ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 003354 005227 177777                INC    #-1              ;;FIRST TIME?
(1) 003360 001046                BNE    68$              ;;BRANCH IF NO
(1) 003362 104401 003430                TYPE   ,69$            ;;TYPE ASCIZ STRING
(2)                                     .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 003366 005737 000042                TST    @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 003372 001012                BNE    70$              ;;BRANCH IF YES
(2) 003374 123727 001214 000001        CMPB   $ENV,#1         ;;ARE WE RUNNING UNDER APT?
(2) 003402 001406                BEQ    70$              ;;BRANCH IF YES
(2) 003404 023727 001140 000176        CMP    SWR,#SWREG     ;;SOFTWARE SWITCH REG SELECTED?
(2) 003412 001005                BNE    71$              ;;BRANCH IF NO
(2) 003414 104406                GTSWR                    ;;GET SOFT-SWR SETTINGS
(2) 003416 000403                BR     71$
(2) 003420 112737 000001 001134 70$:   MOVB   #1,$AUTOB      ;;SET AUTO-MODE INDICATOR
(2) 003426 71$:
(1) 003426 000423                BR     68$              ;;GET OVER THE ASCIZ
(1)                                     ;;69$: .ASCIZ <CRLF>#CRLPAB LPA-11 SYSTEM EXERCISER #<CRLF>
(1) 003476                68$:
1645
1646 003476 005037 001776                CLR    ERCNT
1647 003502                RSTART:
1648 003502 005037 001514                CLR    .DVLS
1649
1650                                     ;;*****
(3)                                     ;;*TEST 1 *TEST THE ADDRESSABILITY OF THE KMC-11
(3)                                     ;;*****
(2) 003506 000240                TST1:  NOP
(1) 003510 012737 000050 001160        MOV    #50,$TIMES     ;;DO 50 ITERATIONS
(1) 003516 012737 003554 001106        MOV    #1$,SLPADR     ;;SET SCOPE LOOP ADDRESS
1651
1652 003524 012737 000001 001102        MOV    #1,$STNM
1653 003532 012737 000001 001200        MOV    #1,$TESTN
1654 003540 012737 003554 001106        MOV    #1$,SLPADR
1655 003546 012737 003554 001110        MOV    #1$,SLPERR
1656
1657 003554 013746 000004 1$:   MOV    ERRVEC, -(SP)   ;SAVE CONTENTS OF LOC 4.
1658 003560 012737 003630 000004        MOV    #2$,ERRVEC     ;SET TIME-OUT TO ERROR REPORTER
1659 003566 013746 000006        MOV    ERRVEC+2, -(SP)
1660 003572 012737 000340 000006        MOV    #340,ERRVEC+2

```

```
1661  
1662 003600 042777 000000 175660      BIC      #0,@KMADO      ;CAUSE A DATA IN, DATA OUT  
1663  
1664 003606 005077 175654      CLR      @KMADO        ;CLEAR KMC  
1665 003612 005077 175654      CLR      @KMAD2  
1666 003616 005077 175654      CLR      @KMAD4  
1667 003622 005077 175654      CLR      @KMAD6  
1668  
1669 003626 000403      BR       3$  
1670  
1671  
1672 003630 104001      2$:      ERROR      1      ;TIME-OUT TRAP TO LOC. 4 WHEN  
1673  
1674  
1675  
1676  
1677  
1678 003632 062706 000004      ADD      #4,SP        ;TAKE CARE OF TRAP SP.  
1679  
1680 003636 012637 000006      3$:      MOV      (SP)+,ERRVEC+2 ;RESTORE ERROR VECTOR.  
1681 003642 012637 000004      MOV      (SP)+,ERRVEC  
1682  
1694  
1695  
(3) :*****  
(4) :*TEST 2      *TEST KMC-11 FUNCTIONS  
(4) :*  
(4) :*IN THIS TEST WE ATTEMPT TO RUN AN INSTRUCTION TEST ON THE KMC-11.  
(4) :*YOU WILL NOT BE ABLE TO LOOP ON AN INDIVIDUAL SUBTEST, BUT MUST LOOP ON  
(4) :*THIS WHOLE TEST SINCE ONCE THE MICRO-CODE IS STARTED, IT EXECUTES IN-LINE  
(4) :*WITH NO-LOOP BACK CAPABILITIES.  
(4) :*IF THIS TEST FAILS YOU SHOULD RUN THE REGULAR KMC-11 DIAGNOSTICS  
(4) :*AVAILABLE FOR THE KMC-11  
(4) :*FOR EACH TEST, WE WILL LIST THE MICRO CODE THAT THE KMC EXECUTES  
(3) :*****  
(2) 003646 000004      TST2:      SCOPE  
(1) 003650 012737 000050 001160      MOV      #50,$TIMES      ;;DO 50 ITERATIONS  
1696      .SBTTL      KMC-11 MICRO-INITIALIZATION.  
1697 003656 004537 037444      JSR      R5,$LOAD      ;LOAD KMC-DIAG. MICRO-CODE  
1698      .GLOBL      DRLPX1  
1699 003662 000000G      .WORD      DRLPX1  
1700 003664 012777 040000 175574      MOV      #BIT14,@KMADO ;INITAILIZE LPA  
1701 003672 005000      CLR      R0  
1702 003674 005200      10$:      INC      R0      ;SHORT DELAY.  
1703 003676 001376      BNE      10$  
1704  
1705 003700 112777 000210 175562      MOV      #BIT7!BIT3,@KMAD1 ;SET RUN BIT ON KMC.  
1706  
1707      .SBTTL      KMC-DIAG. SUB-TEST#1 CRAM WRITE ONES TEST  
1708      .REM      %  
1709      ;TEST #1  
1710      ;  
1711      ;          CRAM ONES WRITE TEST  
1712      ;          ENTERED ON INIT.  
1713      TST1:      MOVE      # 377,BREG      ;GET ALL ONES  
1714      MOVE      BREG,OUT1 <0> ;PUT ALL ONES INTO CRAM
```

```

1715 MOVE BREG,OUT1 <2>
1716 MOVE BREG,OUT1 <3>
1717 MOVE BREG,OUT1 <4>
1718 MOVE BREG,OUT1 <5>
1719 MOVE BREG,OUT1 <6>
1720 MOVE BREG,OUT1 <7>
1721
1722 A1: MOVE INP1 <0>,BREG ;WHEN PDP-11 DONE VERIFICATION,
1723 BZ A1 ;IT WILL CLEAR CRAM ADDR#0
1724
1725 %
1726
1727 003706 005037 001126 CLR $BDDAT
1728 003712 012737 000377 001124 MOV #377,$GDDAT ;EXPECT 377 IN ADDR. #0
1729 003720 117737 175542 001126 MOV @KMADO,$BDDAT ;READ ADDR 0
1730 003726 023737 001124 001126 CMP $GDDAT,$BDDAT ;GOOD READ?
1731 003734 001401 BEQ 11$ ;YES-NEXT SUB-TEST.
1732
1733 003736 104002 ERROR 2 ;CRAM WRITE ONES TEST
1734
1735 003740 012737 177777 001124 11$: MOV #177777,$GDDAT ;EXPECT 377,377, FROM ADDRS.2,3,4,5,6,AND 7
1736 003746 017737 175520 001126 MOV @KMAD2,$BDDAT ;READ ADDRS 2 AND 3
1737 003754 023737 001126 001124 CMP $BDDAT,$GDDAT ;READ ALL ONES?
1738 003762 001401 BEQ 12$ ;YES-CHECK NEXT REG.
1739
1740 003764 104002 ERROR 2 ;CRAM WRITE ONES TEST, ADDR. 2 OR 3
1741
1742 003766 017737 175504 001126 12$: MOV @KMAD4,$BDDAT ;READ ADDRS. 4 AND 5.
1743 003774 023737 001126 001124 CMP $BDDAT,$GDDAT ;READ ALL ONES?
1744 004002 001401 BEQ 13$ ;YES-NEXT REGS.
1745
1746 004004 104002 ERROR 2 ;CRAM WRITE ONES TEST, ADDR. 4 OR 5
1747
1748 004006 017737 175470 001126 13$: MOV @KMAD6,$BDDAT ;READ ADDRS. 6 AND 7.
1749 004014 023737 001126 001124 CMP $BDDAT,$GDDAT ;READ ALL ONES?
1750 004022 001401 BEQ 14$ ;YES-NEXT TEST
1751
1752 004024 104002 ERROR 2 ;CRAM WRITE ONES TEST ADDR 6 OR 7.
1753
1754 004026 105077 175434 14$: CLRB @KMADO ;TELL THE MICRO-PROGRAM TO
1755 ;DO TEST #2.
1756
1757
1758 .SBTTL KMC-DIAG SUB-TEST #2 CRAM WRITE ZEROES TEST
1759
1760 .REM %
1761 ;TEST #2
1762 ; CRAM ZEROES WRITE TEST
1763 ;
1764 ; ENTERED WHEN PDP-11 ZEROES CRAM ADDR 0
1765
1766 TST2: MOVE # 0,BREG ;GET ZERO.
1767 MOVE BREG,OUT1 <0> ;PUT INTO ALL CRAM.
1768 MOVE BREG,OUT1 <2>
1769 MOVE BREG,OUT1 <3>
1770 MOVE BREG,OUT1 <4>
    
```

```

1771          MOVE      BREG,OUT1 <5>
1772          MOVE      BREG,OUT1 <6>
1773          MOVE      BREG,OUT1 <7>
1774
1775          A2:  MOVE      INP1 <0>,BREG      ;GET ZERO
1776          BZ          TST3                    ;WHEN =377,PDP-11 READY FOR NEXT TEST.
1777          BR          A2
1778
1779          *
1780
1781 004032 000240          NOP
1782 004034 000240          NOP
1783 004036 005037 001124  CLR      $GDDAT      ;EXPECT ZERO REGISTERS.
1784 004042 117737 175420 001126  MOVVB  @KMADO,$BDDAT  ;READ ADR.0
1785 004050 001404          BEQ      21$          ;IF ZERO, GOOD.
1786 004052 042737 177400 001126  BIC      #177400,$BDDAT
1787
1788 004060 104002          ERROR  2              ;CRAM WRITE ZERO TEST, ADDR.0
1789
1790 004062 017737 175404 001126 21$:  MOV      @KMAD2,$BDDAT  ;READ ADDRS. 2,3
1791 004070 001401          BEQ      22$          ;GOOD IF=0
1792
1793 004072 104002          ERROR  2              ;CRAM WRITE ZERO TEST, ADDR 2 OR 3
1794
1795 004074 017737 175376 001126 22$:  MOV      @KMAD4,$BDDAT  ;READ ADDRS 4,5
1796 004102 001401          BEQ      23$          ;GOOD IF=0
1797
1798 004104 104002          ERROR  2              ;CRAM WRITE ZERO TEST, ADDR. 4 OR 5
1799
1800 004106 017737 175370 001126 23$:  MOV      @KMAD6,$BDDAT  ;READ ADDRS. 6,7
1801 004114 001401          BEQ      24$          ;GOOD IF=0
1802
1803 004116 104002          ERROR  2              ;CRAM WRITE ZERO TEST ADDR. 6,7.
1804
1805 004120 112777 000377 175340 24$:  MOVVB  #377, @KMADO  ;TELL MICRO-CODE WE'RE READY FOR NEXT TEST.
1806
1807          .SBTTL  KMC-DIAG SUB TEST #3 BRANCH TEST
1808          .REM      *
1809          :
1810          :TEST #3
1811          :
1812          :          BRANCH TEST
1813          :
1814          :          ENTERED WHEN PDP-11 RETURNS 377 TO ADDR.0
1815          :
1816          :          NOTE WE HAVE TO ASSUME BR AND BZ WORK
1817          :          OR WE COULDN'T HAVE GOT THIS FAR.
1818          :          UPON SUCCESSFULL COMPLETION OF THIS TEST WE
1819          :          PUT CODE 377 INTO CRAM ADDR 2
1820          :          AND 0 INTO CRAM ADDR.0. ON FAILURE
1821          :          WE ONLY ZERO CRAM ADDR.0.
1822          TST3:  MOVE      # 376,BREG      ;GET ZERO BIT0
1823          BBO      T3E                    ;ERROR IF BR
1824          MOVE      # 375,BREG      ;BIT 1=0
1825          BB1      T3E                    ;ERROR IF BR
1826          MOVE      # 357,BREG      ;BIT 4=0
  
```

```

1827          BB4      T3E          ;ERROR IF BR
1828          MOVE     # 177,BREG    ;BIT 7=0
1829          BB7      T3E          ;ERROR IF BR
1830
1831          MOVE     # 1,BREG        ;POSITIVE BR TEST
1832          BB0      A3            ;SHOULD BRANCH
1833          BR       T3E          ;IF NOT, ERROR.
1834          A3:    MOVE     # 2,BREG    ;BR OR BIT 1
1835          BB1      B3            ;SHOULD BR.
1836          BR       T3E          ;IF NOT ERROR.
1837          B3:    MOVE     # 20,BREG   ;BR ON BIT 4
1838          BB4      C3            ;SHOULD BR.
1839          BR       T3E          ;IF NOT ERROR.
1840          C3:    MOVE     # 200,BREG  ;BR OF BIT 7
1841          BB7      D3            ;SHOULD BR.
1842          BR       T3E          ;IF NOT ERROR.
1843          D3:    MOVE     # 0,BREG    ;TEST BZ NEGATIVE
1844          BZ       T3E          ;ERROR IF BR.
1845          MOVE     # 377,BREG      ;POS BR
1846          BZ       E3            ;SHOULD BR.
1847          BR       T3E          ;ERROR IF NOT.
1848
1849          E3:    MOVE     BREG,OUT1 <2> ;PUT 377 IN OUTPUT REG #2
1850          BR       F3            ;IF BR INSTR FAILS TO WORK
1851          MOVE     # 0,BREG        ;THEN A ZERO GETS PUT IN #2
1852          MOVE     BREG,OUT1 <2>    ;A SIGN OF AN ERROR
1853          F3:
1854          T3E:   MOVE     # 0,BREG    ;SIGNAL PDP-11 THAT WE ARE
1855          MOVE     BREG,OUT1 <0>    ;THROUGH BR-TEST.
1856
1857          X
1858          004126 005000          CLR      R0          ;TIME OUT COUNTER.
1859          004130 105777 175332 30$:  TSTB   @KMA0        ;MICRO CODE DONE?
1860          004134 001404          BEQ     31$         ;YES-EXIT
1861          004136 005200          INC     R0          ;NO-CHECK FOR TIME OUT.
1862          004140 001373          BNE    30$         ;NO-TIMEOUT, LOOP.
1863
1864          004142 104003          ERROR   3           ;KMC-11 FAILED TO FINISH BRANCH TEST.
1865          004144 000550          BR      TST3        ;;
1866
1867          004146 105777 175320 31$:  TSTB   @KMA2        ;DID ALL BRANCHES WORK?,CODE 377 IN ADDR 2
1868          004152 0C1002          BNE    40$
1869
1870          004154 104003          ERROR   3           ;KMC-11 FAILED BRANCH TEST.
1871          004156 000543          BR      TST3        ;;
1872
1873          .SBTTL  KMC-DIAG SUB-TEST #4 ALU TEST
1874          .REM   X
1875          :TEST #4
1876          :
1877          :       ALU TEST
1878          :
1879          :       ENTERED WHEN PDP-11 PUTS A CODE 377 IN I/O
1880          :       CRAM ADDR 0
1881          :       ADDR 2=0 IF BAD
1882          :
  
```



```

1883          TST4:  MOVE  !NP1 <0>,BREG  ;GET CRAM ADDR0
1884          BZ      A4                ;WHEN =377 DO TEST
1885          BR      TST4
1886
1887          A4:     MOV     # 0,BREG      ;GET=0
1888          MOV     BREG,OUT1 <2>
1889          MOVE    BREG,SPAD <4>
1890          DEC     SPAD <4>           ;MAKE = 377
1891          BZ      B4                ;BR IF GOOD SEC.
1892          BR      T4E
1893          B4:     DEC     SPAD <4>     ;SEC = 376
1894          INC     SPAD <4>           ;INC = 377
1895          BZ      C4                ;BR IF = 377
1896          BR      T4E
1897          C4:     MOVE    # 377,BREG
1898          MOVE    BREG,OUT1 <2>     ;RETURN GOOD CODE
1899
1900
1901          T4E:    MOVE    # 0,.BREG    ;RETURN CODE 0
1902          MOVE    BREG,OUT1 <0>
1903          %
1904
1905 004160 112777 000377 175300 40$:  MOVB  #377,@KMADO  ;START TEST.
1906 004166 005000                CLR    R0          ;TIME OUT COUNTER
1907 004170 105777 175272 41$:  TSTB@KMADO  ;DONE?
1908 004174 001404                BEQ   42$         ;YES-CHECK RESULTS
1909 004176 005200                INC   R0          ;NO-TIME OUT?
1910 004200 001373                BNE  41$         ;NO-LOOP.
1911
1912 004202 104003                ERROR 3          ;KMC-11 FAILED TO FINISH ALU TEST.
1913 004204 000530                BR    TST3       ;;
1914
1915 004206 122777 000377 175256 42$:  CMPB  #377,@KMAD2 ;DID IT DO IT RIGHT?
1916 004214 001401                BEQ   50$
1917
1918 004216 104003                ERROR 3          ;KMC-1? ALU TEST FAILED.
1919
1920          .SBTTL  KMC-DIAG SUB-TEST #5 NPR TEST ZERO TRANSFER
1921
1922          .REM    %
1923
1924          ;
1925          ;TEST 5
1926          ;
1927          ;      NPR TEST
1928          ;
1929          ;      ENTERED IN LINE WHEN PDP-11 PUTS CODE 377 INTO
1930          ;      CRAM ADDR.0
1931          ;      DOES AN INPUT DATA XFERR FROM LOC 1124
1932          ;      IN PDP-11 MEM TO LOC 1126
1933          ;      DOES NO DATA CHECKING RETURNS
1934          ;      CODE 0 TO CRAM ADDR.0 WHEN DONE.
1935          ;
1936          ;
1937          TST5:  MOVE    INP! <0>,BREG  ;WAIT FOR PDP-11
1938          BZ      A5
  
```

```
1939 BR TST5
1940
1941 A5: MOVE # 2,BREG ;SET TO DO NPR IN
1942 MOVE BREG,OUT0 <5> ;SET HIGH ADDR.
1943 MOVE # 124,BREG ;GET LOW ADDR OF ADDR. 1124
1944 MOVE BREG,OUT0 <4> ;SET LOW ADDR.
1945 MOVE # 1,BREG ;SET TO NPR IN
1946 MOVE BREG,OUT1 <10> ;CAUSE NPR IN
1947 B5: MOVE INP1 <10>,BREG ;WAIT FOR NPR DONE.
1948 BBO B5
1949
1950 MOVE INP0 <0>,BREG ;GET LOW BYTE DATA.
1951 MOVE BREG,OUT0 <2> ;OUTPUT NPR DATA REG.
1952 MOVE INP0 <1>,BREG ;GET HIGH BYTE.
1953 MOVE BREG,OUT0 <3> ;SAVE FOR OUTPUT.
1954 MOVE # 2,BREG ;GET OUTPUT ADDR.=1126
1955 MOVE BREG,OUT0 <7>
1956 MOVE # 126,BREG ;LOW PART.
1957 MOVE BREG,OUT0 <6>
1958 MOVE INP1 <11>,SPAD <4> ;/-STN-
1959 AND SPAD <4>,BREG,SPAD
1960 MOVE # 0,BREG
1961 OR SPAD <4>,BREG,BREG
1962 MOVE BREG,OUT1 <11>
1963 MOVE # 21,BREG
1964 MOVE BREG,OUT1 <10> ;SET NPR OUT.
1965
1966 C5: MOVE INP1 <10>,BREG ;WAIT TILL DONE
1967 BBO C5
1968
1969 MOVE # 0,BREG ;TELL PDP-11 WE'RE DONE.
1970 MOVE BREG,OUT1 <0>
1971
1972 %
1973
1974 004220 005037 001124 50$: CLR $GDDAT ;CLEAR $GDDAT (LOC 1124)
1975 004224 012737 177777 001126 MOV #-1,$BDDAT
1976 004232 112777 000377 175226 MOVB #377,@KMADO ;TELL MICRO-CODE TO GO!
1977 004240 005000 CLR R0 ;CLEAR TIME OUT COUNTER.
1978
1979 004242 105777 175220 51$: TSTB @KMADO ;KMC-MICROCODE DONE?
1980 004246 001403 BEQ 52$
1981 004250 005200 INC R0 ;NO-TIME OUT OCCUR?
1982 004252 001373 BNE 51$
1983
1984 004254 104003 ERROR 3 ;KMC-11 FAILED TO FINISH NPR TEST.
1985
1986 004256 023737 001124 001126 52$: CMP $GDDAT,$BDDAT ;NPR ALL ZEROS CORRECTLY?
1987 004264 001401 BEQ 60$ ;YES-NEXT TEST.
1988
1989 004266 104002 ERROR 2 ;KMC-11 NPR TEST ZERO TRANSFER
1990
1991 .SBTTL KMC-DIAG SUB-TEST #6 NPR TEST ONES TRANSFER.
1992 .REM %
1993 ;
1994 ;TEST 6
```

```

1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006 TST6: MOVE INP1 <0>,BREG ;WAIT FOR PDP-11
2007 BZ A6
2008 BR TST6
2009
2010 A6: MOVE # 2,BREG ;SET TO DO NPR IN
2011 MOVE BREG,OUT0 <5> ;SET HIGH ADDR.
2012 MOVE # 124,BREG ;GET LOW ADDR OF ADDR. 1124
2013 MOVE BREG,OUT0 <4> ;SET LOW ADDR.
2014 MOVE # 1,BREG ;SET TO NPR IN
2015 MOVE BREG,OUT1 <10> ;CAUSE NPR IN
2016 B6: MOVE INP1 <10>,BREG ;WAIT FOR NPR DONE.
2017 BBO B6
2018
2019 MOVE INP0 <0>,BREG ;GET LOW BYTE DATA.
2020 MOVE BREG,OUT0 <2> ;OUTPUT NPR DATA REG.
2021 MOVE INP0 <1>,BREG ;GET HIGH BYTE.
2022 MOVE BREG,OUT0 <3> ;SAVE FOR OUTPUT.
2023 MOVE # 2,BREG ;GET OUTPUT ADDR.=1126
2024 MOVE BREG,OUT0 <7>
2025 MOVE # 126,BREG ;LOW PART.
2026 MOVE BREG,OUT0 <6>
2027 MOVE INP1 <11>,SPAD <4> ;/-STN-
2028 AND SPAD <4>,BREG,SPAD
2029 MOVE # 0,BREG
2030 OR SPAD <4>,BREG,BREG
2031 MOVE BREG,OUT1 <11>
2032 MOVE # 21,BREG
2033 MOVE BREG,OUT1 <10> ;SET NPR OUT.
2034
2035 C6: MOVE INP1 <10>,BREG ;WAIT TILL DONE
2036 BBO C6
2037
2038 MOVE # 0,BREG ;TELL PDP-11 WE'RE DONE
2039 MOVE BREG,OUT1 <0>
2040
2041 %
2042
2043 004270 005037 001126 60$: CLR $BDDAT ;NOW XFERR $GDDAT TO $BDDAT
2044 004274 012737 177777 001124 MOV #177777,$GDDAT
2045
2046 004302 112777 000377 175156 MOVB #377,@KMADO ;START NPR XFERR
2047
2048 004310 005000 CLR RO ;TIME OUT COUNTER
2049
2050 004312 105777 175150 61$: TSTB @KMADO ;DONE NPR!
    
```

```

2051 004316 001404 BEQ 62$
2052 004320 005200 INC R0 ;NO-TIME OUT?
2053 004322 001373 BNE 61$
2054
2055 004324 104003 ERROR 3 ;KMC-11 NPR TEST FAILED TO FINISH.
2056 004326 000457 BR TST3 ;:
2057
2058 004330 023737 001124 001126 62$: CMP $GDDAT,$BDDAT ;ONES XFERR OK?
2059 004336 001401 BEQ 70$ ;YES-NEXT TEST.
2060
2061 004340 104002 ERROR 2 ;KMC-11 NPR ONES XFERR FAILED.
2062
2063 .SBTTL KMC-DIAG SUB-TEST #7 INTERRUPT TEST.
2064 .REM %
2065
2066 ;TEST #7
2067 :
2068 : INTERRUPT TEST
2069 :
2070 : THIS TEST GENERATES TWO INTERRUPTS
2071 :
2072 : THE 1ST INTERRUPT TO VECTOR XX0 WHEN
2073 : CRAM ADDR.0 =377 BY PDP-11 CONTROL.
2074 : THE 2ND INTERRUPT TO VECTOR XX4 WHEN
2075 : CRAM ADDR0 =377 (1ST MODE=0 AGAIN).
2076 :
2077 :
2078 TST7: MOVE INP1 <0>,BREG ;WAIT TILL PDP-11 READY:
2079 BZ A7
2080 BR TST7
2081
2082 A7: MOVE INP1 <11>,SPAD <4>
2083 AND SPAD <4>,BREG,SPAD
2084 MOVE # 200,BREG
2085 OR SPAD <4>,BREG,BREG
2086 MOVE BREG,OUT1 <11> ;SET INTR TO ADDR. XX0
2087 B7: MOVE INP1 <11>,BREG ;WAIT TILL DONE
2088 BB7 B7
2089
2090 MOVE # 0,BREG ;TELL PDP-11 WE INTERRUPTED
2091 MOVE BREG,OUT1 <0> ;IN CASE WE DIDN'T.
2092
2093 C7: MOVE INP1 <0>,BREG ;NOW WAIT FOR PDP-11 TO TELL
2094 BZ D7 ;US TO INTR. TO VECTOR XX4
2095 BR C7
2096
2097 D7: MOVE INP1 <11>,SPAD <4>
2098 AND SPAD <4>,BREG,SPAD
2099 MOVE # 300,BREG
2100 OR SPAD <4>,BREG,BREG
2101 MOVE BREG,OUT1 <11> ;SET INTR TO ADDR. XX4
2102
2103 E7: MOVE INP1 <11>,BREG ;WAIT TILL DONE.
2104 BB7 E7
2105
2106 MOVE # 0,BREG ;TELL PDP-11 WE THOUGHT

```

```

2107             MOVE      BREG,OUT1 <0>      ;WE HAD INTERRUPTED.
2108
2109             BR          .
2110             %
2111
2112 004342 012777 004374 175206 70$: MOV      #72$,@VECT1      ;SET-UP FOR 1ST INTERRUPT.
2113 004350 005037 177776             CLR      PS          ;LOWER PROCESSOR STATUS.
2114 004354 112777 000377 175104     MOVB     #377,@KMADO    ;TELL MICRO-CODE WE'RE READY!
2115 004362 005000             CLR      RO
2116 004364 105200 71$: INCB     RO          ;TIME-OUT COUNT IN CASE OF NO INTERRUPT.
2117 004366 100376             BPL      71$
2118
2119 004370 104003             ERROR    3          ;KMC-FAILED TO INTERRUPT AT XX0
2120 004372 000435             BR          TST3      ;;
2121
2122 004374 062706 000004 72$: ADD      #4,SP        ;RESTORE STACK.
2123 004400 012777 004446 175152     MOV      #74$,@VECT2  ;SET NEW INTERRUPT VECTOR
2124 004406 013777 001556 175142     MOV      VECT1,@VEL.1 ;RESTORE OLD VECTOR.
2125 004414 062777 000002 175134     ADD      #2,@VECT1
2126 004422 005037 177776             CLR      PS          ;ALLOW INTERRUPTS
2127 004426 112777 000377 175032     MOVB     #377,@KMADO  ;TELL MICRO CODE WE'RE READY AGAIN.
2128 004434 005000             CLR      RO
2129
2130 004436 105200 73$: INCB     RO          ;TIME OUT COUNTER IN CASE OF NO INTR
2131 004440 100376             BPL      73$
2132
2133 004442 104003             ERROR    3          ;KMC-FAILED TO INTERRUPT AT 004
2134 004444 000410             BR          TST3      ;;
2135
2136 004446 062706 000004 74$: ADD      #4,SP        ;RESTORE STACK
2137 004452 013777 001560 175100     MOV      VECT2,@VECT2 ;RESTORE VECTOR.
2138 004460 062777 000002 175072     ADD      #2,@VECT2
2139

```

2156
2157
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(2)
(1)
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193

```
004466 000004  
004470 012737 000010 001160  
004476 004537 037444  
004502 000000G  
004504 052777 040000 174754  
004512 005000  
004514 005200  
004516 001376  
004520 012777 104000 174740  
004526 005000  
004530 005200  
004532 001376  
004534 032777 000040 174724  
004542 001410  
004544 032777 000002 174714  
004552 001002  
004554 104004  
004556 000432  
004560 104004  
004562 000430
```

```
*****  
: *TEST 3 *TEST SLAVE MICRO PROCESSOR START  
*****  
: *  
: *IN THIS TEST WE'LL LOAD IN MICROCODE INTO THE KMC-11  
: *THAT WILL ALLOW US TO TALK TO THE SLAVE MICRO-PROCESSOR.  
: *  
: *WHEN THE SLAVE MICRO-PROCESSOR IS INTIALIZED (IT GETS INITED  
: *WHEN BIT 14 OF THE KMC'S CSR GET SET) IT SENDS US A VERSION  
: *NUMBER THROUGH THE IPBM'S FAST PATH REGISTER IF THE  
: *INITIAL CSR OF IPBM (AS SEEN BY THE SLAVE) IS CORRECT.  
: *IF BAD, IT SEND CODE 377 THOUGH BOTH FAST PATH AND  
: *SILO TO INDICATE AN ERROR.  
*****  
TST3: SCOPE  
MOV #10,$TIMES ;DO 10 ITERATIONS  
JSR R5,$LOAD ;LOAD MICRO CODE INTO KMC-11  
.GLOBL DRLPX0  
.WORD DRLPX0 ;FILE 'DRLPX0'  
BIS #BIT14,@KMADO ;SET INIT BIT, WHEN THIS BIT  
;CLEARS IN SLAVE, RUN STARTS.  
1$: CLR R0  
INC R0 ;LITTLE DELAY LOOP.  
BNE 1$  
MOV #BIT15!BIT11,@KMADO ;CLEAR INTIT, SET RUN  
CLR R0  
2$: INC R0 ;LITTLE DELAY.  
BNE 2$  
BIT #BIT5,@KMADO ;DID SLAVE SEND DATA THROUGH F.P.?  
BEQ 4$ ;BR IF YES (LOOKING AT IPBM CSR BIT 5)  
;SENDS EITHER VERSION OR CODE 377.  
BIT #BIT1,@KMADO ;RIGHT NOW WE KNOW AN ERROR  
BNE 3$ ;HAS OCCURRED. IF THERE IS DATA  
;IN THE SILO, MASTER'S IPBM  
;CSR BIT 1=0  
;IPBM INIT ERROR  
ERROR 4 ;ERROR (IPBM) M8254 FAILED TO  
;INITIALIZE PROPERLY. AS SEEN FROM  
;;  
BR TST4  
3$: ERROR 4 ;ERROR IPBM OR (MOST LIKELY)  
;SLAVE MICRO PROCESSOR FAILED  
;TO START PROPERLY.  
BR TST4 ;;
```

```

2194 004564 112777 000004 174700 4$: MOVB #4,@KMAD2 ;ISSUE CMMD TO READ FAST PATH
2195
2196 004572 122777 000377 174672 5$: CMPB #377,@KMAD2 ;WAIT FOR READ.
2197 004600 001374 BNE 5$
2198 004602 013737 001512 001124 MOV VERSN,$GDDAT ;GET MICRO-CODE VERSION NUMBER.
2199 004610 117737 174662 001126 MOVB @KMAD4,$BDDAT ;IF=MINUS, ERROR CODE RETURNED,
2200 004616 100002 BPL 6$ ;IF PLUS, NEXT TEST
2201
2202 004620 104005 ERROR 5 ;ERROR-SLAVE MICRO PROCESSOR
2203 ;INDICATED M8254 FAILED TO
2204 ;INITIALIZE PROPERLY.
2205 004622 000410 BR TST4 ;;
2206
2207 004624 6$: BR TST4 ;;
(2) 004624 000407 CLPB $BDDAT+1 ;CLEAR HIGH BYTE SO THE OPERATOR IS
2208 004626 105037 001127 ;NOT CONFUSED
2209
2210 004632 123737 001124 001126 CMPB $GDDAT,$BDDAT ;DOES THE MICRO-CODE VERSION
2211 ;NUMBER AGREE WITH THE
2212 ;VERSION WE EXPECT?
2213 004640 001401 BEQ TST4 ;;
2214
2215 004642 104005 ERROR 5 ;WRONG MICRO-CODE VERSION #
2216 ;RETURNED BY SLAVE MICRO-PROCESSOR.
2217 ;NOTE:(1) YOU COULD BE RUNNING THE
2218 ;WRONG VERSION OF THE DIAGNOSTIC
2219 ;(2) SLAVE MICRO COULD BE BAD
2220 ;OR (3) M8254 MIGHT BE BAD
2221 ;(FAST PATH)
2222
2223 .SBTTL **PHASE 2 IPBM/DMC TESTING.
  
```



```

2390 005220 104004          ERROR 4          ;SLAVE MICRO FAILED TO LOOP BACK
2391                                     ;DATA THROUGH SILO..
2392
2393 005222 112777 000002 174242 14$:  MOVB  #2,@KMAD2      ;DATA IN SILO READY TO BE READ, TELL
2394                                     ;DRLPX0 TO READ IT.
2395 005230 004737 034162          JSR    PC,WIR      ;WAIT FOR DRLPX0 READY
2396
2397 005234 117737 174236 001126    MOVB  @KMAD4,$BDDAT ;GET DATA.
2398 005242 113737 005300 001124    MOVB  21$,$GDDAT   ;GET S/B
2399
2400 005250 123737 001124 001126    CMPB  $GDDAT,$BDDAT ;DID DATA LOOP BACK OK?
2401 005256 001401          BEQ    15$
2402
2403 005260 104006          ERROR 6          ;SILO DATA ERROR.
2404
2405 005262 105237 005300 15$:  INCB  21$          ;UPDATE LOOP BACK PATTERN
2406                                     ;NOTE: IF YOU REALLY WANT TO,
2407                                     ;YOU CAN MODIFY THIS
2408                                     ;SECTION OF TEST TO SEND ONLY A
2409                                     ;SPECIFIC PATTERN.
2410                                     ;TO DO THIS, CHANGE THE
2411                                     ;INSTRUCTION "INCB 21$" (2 WORDS)
2412                                     ;ALONG WITH THE "MOV#0,21$"
2413
2414 005266 000710          BR     11$          ;IF NOT DONE 256 TIMES, LOOP.
2415                                     ;WARNING: THIS PROGRAM MUST
2416                                     ;SEND 256 DATA LOOP BACK PATTERNS
2417                                     ;THROUGH THE SILO REG SO
2418                                     ;THAT THE SLAVE MICRO CODE
2419                                     ;WILL EXIT FROM LOOP BACK
2420                                     ;MODE TO COMMAND MODE.
2421
2422
2423 005270 005037 005276 18$:  CLR    20$          ;EXIT, ALL DONE!
2424 005274 000402          BR     TST5        ;;
2425
2426 005276 000000 20$:  .WORD  0          ;CONTAINS CURRENT LOOP COUNT.
2427 005300 000000 21$:  .WORD  0          ;CONTAINS CURRENT LOOP BACK PATTERN.
2428
  
```

2430
 2578
 (1)
 (5)
 (4)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (5)
 (4)
 (3)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (2)
 (2)
 (1)
 (1)
 (1)
 (1)
 (1)

.SBTTL PHASE 3 I/B BUS TESTING.

```

:*****
: *TEST 5      I/O DEVICE TEST-AD11-K OPTION
:
: *IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.
: *WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE
: *KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS
: *NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.
: *THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE
: *I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE
: *IT FROM "SR1" (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN
: *HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE
: *I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO "SR1"
: *IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE
: *INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.
:
: IN THIS TEST, WE WILL EXERCISE THE AD11-K
: IF DVID1 IS UNALTERED, WE WILL DO THIS TEST SENCE THE
: AD11-K IS A DEFAULT I/O DEVICE. IF AN AD11-K IS NOT ON THE
: I/O BUS, YOU MUST DELETE IT FROM SR.
:
: TESTS:
: 1.  MAKE SURE AD11-K RESPONDS TO ADDRESS.
:     (NOTE IF THE AD11-K ON YOU I/O BUS HAS NON-STANDARD
:     ADDRESS, PLEASE REFLECT THE NEW ADDRESS IN ADDRESS "STREG")
:
: 2.  MAKE SURE WE CAN WRITE A ZERO INTO CSR (EXCEPT BIT 7:15).
:
: 3.  WRITE BITS 13,11,9,6 MAKE SURE THEY WRITE.
:
: 4.  WRITE BITS 12,10,8,4 MAKE SURE THEY WRITE.
:
: 5.  SET A/D START, MAKE SURE IT CLEARS, DONE FLAG SETS.
:
: 6.  SET A/D START AGAIN, MAKE SURE ERROR FLAG SETS.
:
: 7.  MAKE SURE ERROR FLAG CAN BE CLEARED.
:*****

```

```

005302  000004
005304  032737  000001  001562
005312  001002
005314  000137  005760
005320
005320  005037  001126

```

```

:TST5:  SCOPE
      BIT      #BIT0,SR1      ;IS THIS DEVICE SELECTED FOR TEST?
      BNE      10$            ;NOTE: DEFAULT=YES.
      JMP      7$
10$:
      CLR      $BDDAT
: *      MOV      $BDDAT,@STREG1  ;/ PUT DATA FROM $BDDAT TO DEVICE REG STREG1
;OK-WHAT WE JUST DID IS ASKED THE
;SLAVE MICRO-CODE TO WRITE THE CONTENTS
;OF THE AD11K'S CSR TO ZEROES.
;IF THE AD11K FAILS TO RESPOND TO ITS
;ADDRESS-THE SLAVE WILL SEND US AN

```

```

(1)                                     ;ERROR CODE THAT CAUSES US TO SET $AERR=1
(1)                                     ;IN THE SUPPORT ROUTINES.
(1)
(1) 005334 005737 037442             TST     $AERR             ;DEVICE PRESENT?
(1) 005340 001403                     BEQ     11$              ;YES-CONTINUE
(1) 005342 104010                     ERROR   10              ;A/D #1 FAILED TO RESPOND
(1) 005344 000137 005760             JMP     7$              ;TO ADDR.
(1) 005350                               11$:
(2)
(2) ;*      MOV     @STREG1,$BDDAT ;/READ DEVICE REG STREG1,PUT DATA IN $BDDAT.
(1) 005360 042737 100200 001126      BIC     #BIT15!BIT7,$BDDAT ;CLEAR OUT FLAGS
(1) 005366 005037 001124             CLR     $GDDAT
(1) 005372 005737 001126             TST     $BDDAT          ;IS CSR ZERO?
(1) 005376 001402                     BEQ     1$
(1)
(1) 005400 104010                     ERROR   10              ;FAILED TO ZERO AD11K CSR #1
(3) 005402 000572                     BR      TST6            ;;
(1) 005404 012737 025100 001124      1$:  MOV     #BIT13!BIT11!BIT9!BIT6,$GDDAT ;NOW TRY A NEW BIT PATTERN
(1)
(2)
(2) ;*      MOV     $GDDAT,@STREG1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG STREG1
(2)
(2) ;*      MOV     @STREG1,$BDDAT ;/READ DEVICE REG STREG1,PUT DATA IN $BDDAT.
(1) 005432 042737 100200 001126      BIC     #BIT15!BIT7,$BDDAT ;IGNORE FLAGS
(1) 005440 023737 001124 001126      CMP     $GDDAT,$BDDAT   ;BIT SET OK?
(1) 005446 001402                     BEQ     2$
(1)
(1) 005450 104010                     ERROR   10              ;FAILED TO WRITE #1 AD11K CSR CORRECTLY
(3) 005452 000546                     BR      TST6            ;;
(1)
(1) 005454 012737 012420 001124      2$:  MOV     #BIT12!BIT10!BIT8!BIT4,$GDDAT ;NOW TRY A NEW BIT PATTERN.
(1)
(2)
(2) ;*      MOV     $GDDAT,@STREG1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG STREG1
(2)
(2) ;*      MOV     @STREG1,$BDDAT ;/READ DEVICE REG STREG1,PUT DATA IN $BDDAT.
(1) 005502 042737 100200 001126      BIC     #BIT15!BIT7,$BDDAT ;IGNORE FLAGS
(1) 005510 023737 001124 001126      CMP     $GDDAT,$BDDAT   ;BITS SET OK?
(1) 005516 001402                     BEQ     3$
(1)
(1) 005520 104010                     ERROR   10              ;FAILED TO WRITE #1 AD11K CSR CORRECTLY.
(3) 005522 000522                     BR      TST6            ;;
(1)
(1) 005524 005037 001124             3$:  CLR     $GDDAT          ;NOW CLEAR ALL BITS SET
(2)
(2) ;*      MOV     $GDDAT,@STREG1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG STREG1
(2)
(2) ;*      MOV     @STREG1,$BDDAT ;/READ DEVICE REG STREG1,PUT DATA IN $BDDAT.
(1) 005550 042737 000200 001126      BIC     #BIT7,$BDDAT    ;IGNORE DONE FLAG.
(1) 005556 005737 001126             TST     $BDDAT          ;CSR CLEAR?
(1) 005562 001402                     BEQ     4$
(1)
(1) 005564 104010                     ERROR   10              ;#1 AD11K CSR FAILED TO CLEAR.
(3) 005566 000500                     BR      TST6            ;;
(1)
(1) 005570                               4$:
    
```

```

(2)
(2)
(1) 005600 012737 000001 001124 ;* MOV @ADBUF1,MYTEMP ;/READ DEVICE REG ADBUF1,PUT DATA IN MYTEMP.
MOV #BIT0,$GDDAT ;NOW WE'LL SET THE A/D START BIT.
(2)
(2)
(1) 005616 012737 000200 001124 ;* MOV $GDDAT,@STREG1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG STREG1
MOV #BIT7,$GDDAT ;WHEN WE READ CSR BACK EXPECT
;START BIT TO CLEAR, DONE TO SET.
(2)
(2)
(1) ;* MOV @STREG1,$BDDAT ;/READ DEVICE REG STREG1,PUT DATA IN $BDDAT.
(1)
(1) 005634 023737 001124 001126 CMP $GDDAT,$BDDAT ;START OK?
(1) 005642 001402 BEQ 5$
(1)
(1) 005644 104010 ERROR 10 ;CSR BAD AFTER #1 A/D START
;BIT 0 SHOULD CLEAR ONLY A/D DONE SET.
(1)
(3) 005646 000450 BR TST6 ;:
(1)
(1) 005650 012737 000001 001124 5$: MOV #BIT0,$GDDAT ;OK-NOW WE'RE GONNA START ANOTHER
;A/D CONVERSION WITHOUT READING
;THE RESULTS OF THE LAST ONE. THIS
;SHOULD CAUSE THE ERROR FLAG
;AS WELL AS THE DONE FLAG TO SET.
(1)
(2)
(2) ;* MOV $GDDAT,@STREG1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG STREG1
(1)
(2)
(2)
(1) 005676 012737 100200 001124 ;* MOV @STREG1,$BDDAT ;/READ DEVICE REG STREG1,PUT DATA IN $BDDAT.
MOV #BIT15.BIT7,$GDDAT ;S/B
(1)
(1) 005704 023737 001124 001126 CMP $GDDAT,$BDDAT ;IS 15 AND 7 SET?
(1)
(1) 005712 001401 BEQ 6$
(1)
(1) 005714 104010 ERROR 10 ;ERROR #1 AD11K BIT 15 AND 7 SHOULD BE SET.
(1)
(1)
(1) 005716 005037 001124 6$: CLR $GDDAT ;MAKE SURE ERROR FLAG CLEARS
(2)
(2) ;* MOV $GDDAT,@STREG1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG STREG1
(2)
(2) ;* MOV @STREG1,$BDDAT ;/READ DEVICE REG STREG1,PUT DATA IN $BDDAT.
(1) 005742 042737 000200 001126 BIC #BIT7,$BDDAT ;IGNORE DONE FLAG.
(1) 005750 005737 001126 TST $BDDAT ;IS CSR CLEAR?
(1) 005754 001401 BEQ 7$
(1) 005756 104010 ERROR 10 ;AD11K CSR FAILED TO CLEAR.
(1)
(1) 005760 7$:
(2)
(2) ;* MOV @ADBUF1,MYTEMP ;/READ DEVICE REG ADBUF1,PUT DATA IN MYTEMP.
  
```



```

(1)                                     ;ERROR CODE THAT CAUSES US TO SET $AERR=1
(1)                                     ;IN THE SUPPORT ROUTINES.
(1)
(1) 006022 005737 037442               TST    $AERR                ;DEVICE PRESENT?
(1) 006026 001403                       BEQ    11$                  ;YES-CONTINUE
(1) 006030 104010                       ERROR  10                   ;A/D #2 FAILED TO RESPOND
(1) 006032 000137 006446               JMP    7$                   ;TO ADDR.
(1) 006036                               11$:
(2)
(2)                                     ;*
(1) 006046 042737 100200 001126       MOV    @STREG2,$BDDAT      ;/READ DEVICE REG STREG2,PUT DATA IN $BDDAT.
(1) 006054 005037 001124 001126       BIC   #BIT15!BIT7,$BDDAT ;CLEAR OUT FLAGS
(1) 006060 005737 001126               CLR    $GDDAT
(1) 006064 001402                       TST    $BDDAT              ;IS CSR ZERO?
(1)                                     BEQ    1$
(1) 006066 104010                       ERROR  10                   ;FAILED TO ZERO AD11K CSR #2
(3) 006070 000572                       BR     TST7
(1) 006072 012737 025100 001124       1$: MOV    #BIT13!BIT11!BIT9!BIT6,$GDDAT ;NOW TRY A NEW BIT PATTERN
(1)
(2)
(2)                                     ;*
(2)                                     ;*
(1) 006120 042737 100200 001126       MOV    @STREG2,$BDDAT      ;/READ DEVICE REG STREG2,PUT DATA IN $BDDAT.
(1) 006126 023737 001124 001126       BIC   #BIT15!BIT7,$BDDAT ;IGNORE FLAGS
(1) 006134 001402                       CMP    $GDDAT,$BDDAT      ;BIT SET OK?
(1)                                     BEQ    2$
(1) 006136 104010                       ERROR  10                   ;FAILED TO WRITE #2 AD11K CSR CORRECTLY
(3) 006140 000546                       BR     TST7
(1)
(1) 006142 012737 012420 001124       2$: MOV    #BIT12!BIT10!BIT8!BIT4,$GDDAT ,NOW TRY A NEW BIT PATTERN.
(1)
(2)
(2)                                     ;*
(2)                                     ;*
(1) 006170 042737 100200 001126       MOV    @STREG2,$BDDAT      ;/READ DEVICE REG STREG2,PUT DATA IN $BDDAT.
(1) 006176 023737 001124 001126       BIC   #BIT15!BIT7,$BDDAT ;IGNORE FLAGS
(1) 006204 001402                       CMP    $GDDAT,$BDDAT      ;BITS SET OK?
(1)                                     BEQ    3$
(1) 006206 104010                       ERROR  10                   ;FAILED TO WRITE #2 AD11K CSR CORRECTLY.
(3) 006210 000522                       BR     TST7
(1)
(1) 006212 005037 001124               3$: CLR    $GDDAT          ;NOW CLEAR ALL BITS SET
(2)
(2)                                     ;*
(2)                                     ;*
(1) 006236 042737 000200 001126       MOV    @STREG2,$BDDAT      ;/READ DEVICE REG STREG2,PUT DATA IN $BDDAT.
(1) 006244 005737 001126 001126       BIC   #BIT7,$BDDAT        ;IGNORE DONE FLAG.
(1) 006250 001402                       TST    $BDDAT              ;CSR CLEAR?
(1)                                     BEQ    4$
(1) 006252 104010                       ERROR  10                   ;#2 AD11K CSR FAILED TO CLEAR.
(3) 006254 000500                       BR     TST7
(1)
(1) 006256                               4$:

```



```

(2)
(2)
(1) 006266 012737 000001 001124 :* MOV @ADBUF2,MYTEMP ;/READ DEVICE REG ADBUF2,PUT DATA IN MYTEMP.
MOV #BIT0,$GDDAT ;NOW WE'LL SET THE A/D START BIT.
(2)
(2)
(1) 006304 012737 000200 001124 :* MOV $GDDAT,@STREG2 ;/ PUT DATA FROM $GDDAT TO DEVICE REG STREG2
MOV #BIT7,$GDDAT ;WHEN WE READ CSR BACK EXPECT
;START BIT TO CLEAR, DONE TO SET.
(1)
(2)
(2)
(1) 006322 023737 001124 001126 :* MOV @STREG2,$BDDAT ;/READ DEVICE REG STREG2,PUT DATA IN $BDDAT.
(1) 006330 001402 CMP $GDDAT,$BDDAT ;START OK?
BEQ 5$
(1)
(1) 006332 104010 ERROR 10 ;CSR BAD AFTER #2 A/D START
;BIT 0 SHOULD CLEAR ONLY A/D DONE SET.
(1)
(3) 006334 000450 BR TST7 ;:
(1)
(1) 006336 012737 000001 001124 5$: MOV #BIT0,$GDDAT ;OK-NOW WE'RE GONNA START ANOTHER
;A/D CONVERSION WITHOUT READING
;THE RESULTS OF THE LAST ONE. THIS
;SHOULD CAUSE THE ERROR FLAG
;AS WELL AS THE DONE FLAG TO SET.
(1)
(2)
(2)
(1) :* MOV $GDDAT,@STREG2 ;/ PUT DATA FROM $GDDAT TO DEVICE REG STREG2
(2)
(2)
(1) 006364 012737 100200 001124 :* MOV @STREG2,$BDDAT ;/READ DEVICE REG STREG2,PUT DATA IN $BDDAT.
MOV #BIT15!BIT7,$GDDAT ;S/B
(1)
(1) 006372 023737 001124 001126 CMP $GDDAT,$BDDAT ;IS 15 AND 7 SET?
(1)
(1) 006400 001401 BEQ 6$
(1)
(1) 006402 104010 ERROR 10 ;ERROR #2 AD11K BIT 15 AND 7 SHOULD BE SET.
(1)
(1)
(1) 006404 005037 001124 6$: CLR $GDDAT ;MAKE SURE ERROR FLAG CLEARS
(2)
(2)
(2)
(1) :* MOV $GDDAT,@STREG2 ;/ PUT DATA FROM $GDDAT TO DEVICE REG STREG2
(2)
(2)
(1) 006430 042737 000200 001126 :* MOV @STREG2,$BDDAT ;/READ DEVICE REG STREG2,PUT DATA IN $BDDAT.
BIC #BIT7,$BDDAT ;IGNORE DONE FLAG.
(1) 006436 005737 001126 TST $BDDAT ;IS CSR CLEAR?
(1) 006442 001401 BEQ 7$
(1) 006444 104010 ERROR 10 ;AD11K CSR FAILED TO CLEAR.
(1)
(1) 006446 7$:
(2)
(2)
(2)
(1) :* MOV @ADBUF2,MYTEMP ;/READ DEVICE REG ADBUF2,PUT DATA IN MYTEMP.
2582
  
```



```

2627 006470 000137 007264          JMP      11$
2628 006474          10$:
2629
2630 006474 012737 041110 001124      MOV      #BIT14.BIT9!BIT6.BIT3,$GDDAT ;TEST THESE BITS
2631
2632
(1)          ;*      MOV      $GDDAT,@KWADRO  ;/ PUT DATA FROM $GDDAT TO DEVICE REG KWADRO
2633
2634 006512 005737 037442          TST      $AERR          ;DID KW11K RESPOND
2635 006516 001403          BEQ      12$          ;TO ADDR? $AERR=0?
2636          ;IF YES-NEXT TEST.
2637 006520 104010          ERROR   10          ;KW11K FAILED TO RESPOND TO ADDR.
2638 006522 000137 007264          JMP      11$
2639 006526          12$:
2640
2641
(1)          ;*      MOV      @KWADRO,$BDDAT  ;/READ DEVICE REG KWADRO,PUT DATA IN $BDDAT.
2642
2643 006536 023737 001124 001126      CMP      $GDDAT,$BDDAT ;DID THESE BITS SET?
2644 006544 001403          BEQ      1$
2645
2646 006546 104011          ERROR   11          ;KW11K R/W CSR ERROR.
2647 006550 000137 007264          JMP      11$
2648
2649 006554 012737 020404 001124 1$:      MOV      #BIT13!BIT8!BIT2,$GDDAT ;GET NEW PATTERN
2650
(1)          ;*      MOV      $GDDAT,@KWADRO  ;/ PUT DATA FROM $GDDAT TO DEVICE REG KWADRO
2651
(1)          ;*      MOV      @KWADRO,$BDDAT  ;/READ DEVICE REG KWADRO,PUT DATA IN $BDDAT.
2652
2653 006602 023737 001124 001126      CMP      $GDDAT,$BDDAT ;DID THE BITS SET?
2654 006610 001403          BEQ      2$
2655
2656 006612 104011          ERROR   11          ;KW11K R/W CSR ERROR.
2657 006614 000137 007264          JMP      11$
2658
2659 006620 005037 001124          CLR      $GDDAT          ;WRITE ALL ZEROS.
2660
(1)          ;*      MOV      $GDDAT,@KWADRO  ;/ PUT DATA FROM $GDDAT TO DEVICE REG KWADRO
2661
(1)          ;*      MOV      @KWADRO,$BDDAT  ;/READ DEVICE REG KWADRO,PUT DATA IN $BDDAT.
2662 006644 005737 001126          TST      $BDDAT          ;DID CSR CLEAR?
2663 006650 001403          BEQ      3$          ;YES-GOOD.
2664
2665 006652 104011          ERROR   11          ;KW11K FAILED TO ZERO CSR.
2666 006654 000137 007264          JMP      11$
2667
2668 006660 012737 004124 001124 3$:      MOV      #BIT11!BIT6!BIT4!BIT2,$GDDAT ;TEST CLOCK B'S CSR.
2669
(1)          ;*      MOV      $GDDAT,@KWADR4  ;/ PUT DATA FROM $GDDAT TO DEVICE REG KWADR4
2670
(1)          ;*      MOV      @KWADR4,$BDDAT  ;/READ DEVICE REG KWADR4,PUT DATA IN $BDDAT.
2671
2672 006706 023737 001126 001124      CMP      $BDDAT,$GDDAT ;DID THE BITS SET OK?
2673 006714 001402          BEQ      4$
2674

```

```

2675 006716 104011 ERROR 11 ;KW11K CLK B CSR FAILURE.
2676 006720 000561 BR TST10 ;;
2677
2678 006722 012737 000050 001124 4$: MOV #BITS!BIT3,$GDDAT ;TRY THESE BITS
2679
2680
2681 ;* MOV $GDDAT,@KWADR4 ;/ PUT DATA FROM $GDDAT TO DEVICE REG KWADR4
2682 ;* MOV @KWADR4,$BDDAT ;/READ DEVICE REG KWADR4,PUT DATA IN $BDDAT.
2683 006750 023737 001124 001126 ;* CMP $GDDAT,$BDDAT ;DID BITS SET OK?
2684 006756 001402 BEQ 5$
2685 006760 104011 ERROR 11 ;KW11K-CLK B CSR ERROR
2686 006762 000540 BR TST10 ;;
2687
2688 006764 005037 001124 5$: CLR $GDDAT ;ZERO CSR
2689
2690 ;* MOV $GDDAT,@KWADR4 ;/ PUT DATA FROM $GDDAT TO DEVICE REG KWADR4
2691 ;* MOV @KWADR4,$BDDAT ;/READ DEVICE REG KWADR4,PUT DATA IN $BDDAT.
2692 007010 005737 001126 ;* TST $BDDAT ;DID CSR CLEAR?
2693 007014 001402 BEQ 6$
2694 007016 104011 ERROR 11 ;KW11K-CLKB CSR ERROR.
2695 007020 000521 BR TST10 ;;
2696
2697 007022 012737 000003 001124 6$: MOV #BIT0!BIT1,$GDDAT ;PUT 'ENABL CNTR A' AND RATE 1MHZ IN CSR
2698 007030 012737 177777 001126 MOV #177777,$BDDAT ;PRELOAD CNTR.
2699
2700 ;* MOV $BDDAT,@KWADR1 ;/ PUT DATA FROM $BDDAT TO DEVICE REG KWADR1
2701 ;* MOV $GDDAT,@KWADRO ;/ PUT DATA FROM $GDDAT TO DEVICE REG KWADRO
2702 ;* MOV @KWADRO,$BDDAT ;/READ DEVICE REG KWADRO,PUT DATA IN $BDDAT.
2703 ;CLOCK SHOULD OVERFLOW, CLEAR
2704 007066 012737 000242 001124 MOV #BITS!BIT7!BIT1,$GDDAT ;BIT 0 ('ENABLE CNTR A) SET
2705 ;BIT 5 AND 7
2706 007074 023737 001124 001126 CMP $GDDAT,$BDDAT ;HAPPEN OK?
2707 007102 001401 BEQ 7$
2708 ;KW11K
2709 007104 104011 ERROR 11 ;CLOCK CSR A PROBLEM
2710 ;TRY CLEARING CSR
2711 007106 005037 001124 7$: CLR $GDDAT
2712 ;* MOV $GDDAT,@KWADRO ;/ PUT DATA FROM $GDDAT TO DEVICE REG KWADRO
2713 ;* MOV @KWADRO,$BDDAT ;/READ DEVICE REG KWADRO,PUT DATA IN $BDDAT.
2714 007132 005737 001126 ;* TST $BDDAT ;DID CSR CLEAR?
2715 007136 001402 BEQ 8$
2716 007140 104011 ERROR 11 ;CSR FAILED TO CLEAR
2717 007142 000450 BR TST10 ;;
2718
2719 007144 012737 000003 001124 8$: MOV #BIT1!BIT0,$GDDAT ;SET 'ENABL CNTR B' AND 1MHZ IN CSR B
2720 007152 012737 000377 001126 MOV #377,$BDDAT ;PRESENT ENTR.
2721

```

```

(1)          ;*      MOV      $BDDAT,@KWADR5  ;/ PUT DATA FROM $BDDAT TO DEVICE REG KWADR5
2722
(1)          ;*      MOV      $GDDAT,@KWADR4  ;/ PUT DATA FROM $GDDAT TO DEVICE REG KWADR4
2723
2724          ;CLOCK SHOULD OVERFLOW,NOT CLEAR
2725          ;'ENB CNTR B' AND SET
2726          ;'B OVERFL FLAG'
2727
(1)          ;*      MOV      @KWADR4,$BDDAT  ;/READ DEVICE REG KWADR4,PUT DATA IN $BDDAT.
2728 007210 012737 000203 001124  ;*      MOV      #BIT7!BIT0:$GDDAT  ;EXPECT ONLY OVERFL FLAG SET.
2729 007216 023737 001124 001126  ;*      CMP      $GDDAT,$BDDAT  ;CSR OK?
2730 007224 001401          ;*      BEQ      9$
2731
2732 007226 104011          ;*      ERROR   11              ;KW11K CLKB COUNT UP ERROR.
2733
2734
2735 007230 005037 001124  ;*      CLR      $GDDAT          ;TRY CLEARING CSR
2736
(1)          ;*      MOV      $GDDAT,@KWADR4  ;/ PUT DATA FROM $GDDAT TO DEVICE REG KWADR4
2737
(1)          ;*      MOV      @KWADR4,$BDDAT  ;/READ DEVICE REG KWADR4,PUT DATA IN $BDDAT.
2738 007254 005737 001126  ;*      TST      $BDDAT          ;DID CSR CLEAR?
2739 007260 001401          ;*      BEQ      TST10          ;:
2740
2741 007262 104011          ;*      ERROR   11              ;KW11K CLOCK B SCR FAILED TO CLEAR.
2742
2743 007264          ;*      11$:

```

2745
 2856
 2857

(5)
 (4)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (5)
 (5)
 (5)
 (5)
 (4)
 (3) 007264 000004
 (1) 007266 032737 000004 001562
 (1) 007274 001002
 (1) 007276 000137 007772
 (1) 007302
 (1) 007302 005037 001514
 (1) 007306 005037 001124
 (2)
 (2)
 (1) 007322 005737 037442
 (1) 007326 001403
 (1)
 (1) 007330 104012
 (1)
 (1)
 (1)
 (1)
 (1) 007332 000137 007772
 (1)
 (1) 007336
 (2)
 (2)
 (1) 007346 005737 001126
 (1) 007352 001403
 (1)
 (1) 007354 104012
 (1) 007356 000137 007772
 (1)
 (1) 007362
 (2)
 (2)
 (2)
 (2)

```

:*****
:*TEST 10      *TEST THE DR11K OPTION,#1, IF "SR1" BIT 2=1(SET)
:
:*IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.
:*WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE
:*KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS
:*NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.
:*THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE
:*I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE
:*IT FROM "SR1" (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN
:*HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE
:*I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO "SR1"
:*IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE
:*INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.
:
:*IN THIS TEST WE WILL CHECK OUT DR11 #1 IF BIT 2 OF SR1=1
:*(SET). ALSO, DATA LOOP BACK (OUTPUT DATA BACK TO INPUT) WILL BE CHECKED
:*IF "SR2" BIT 2 =1 (SET). FOR THIS, YOU MUST HAVE INSTALLED A LOOP-BACK
:*CABLE.
:*****
TST10:  SCOPE
        BIT      #BIT2,SR1
        BNE      10$      ;/-TDRT-
        JMP      11$
10$:
        CLR      .DVLS
        CLR      $GDDAT      ;/SET TO XFERR ZEROS.
:*      MOV      $GDDAT,@DRCR1      ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR1
        TST      $AERR      ;=1 IF DEVICE NOT PRESENT,
        BEQ      1$      ;IF 0, PRESENT, (CON'T).
        ERROR    12      ;DR11K #1 DID NOT RESPOND WHEN
                        ;ADDRESSED. "SR1" BIT 2 SELECTED
                        ;THIS OPTION. ADDRESS 'DRCR'1
                        ;CONTAINS ITS ADDRESS
        JMP      11$
1$:
:*      MOV      @DRCR1,$BDDAT      ;/READ DEVICE REG DRCR1,PUT DATA IN $BDDAT.
        TST      $BDDAT      ;DID CSR CLEAR?
        BEQ      2$
        ERROR    12      ;DR11K #1 CSR FAILED TO CLEAR.
        JMP      11$
2$:
:*      MOV      @DROA1,$GDDAT      ;/READ DEVICE REG DROA1,PUT DATA IN $GDDAT.
:*      MOV      @DRIA1,$GDDAT      ;/READ DEVICE REG DRIA1,PUT DATA IN $GDDAT.
    
```

```

(1) 007402 012737 040100 001124      MOV      #BIT6:BIT14,$GDDAT ;LOAD WRITEABLE BITS.
(2)
(2) ;*      MOV      $GDDAT,@DRCR1  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR1
(2) ;*      MOV      @DRCR1,$BDDAT  ;/READ DEVICE REG DRCR1,PUT DATA IN $BDDAT.
(1) 007430 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;DID BITS SET?
(1) 007436 001403      BEQ      3$
(1) 007440 104012      ERROR    12              ;DR11K #1 CSR BIT(S) FAILED.
(1) 007442 000137 007772      JMP      11$
(1) 007446 005037 001124      3$:     CLR      $GDDAT      ;TRY CLEARING CSR.
(2) ;*      MOV      $GDDAT,@DRCR1  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR1
(2) ;*      MOV      @DRCR1,$BDDAT  ;/READ DEVICE REG DRCR1,PUT DATA IN $BDDAT.
(1) 007472 005737 001126      TST     $BDDAT          ;DID CSR CLEAR?
(1) 007476 001403      BEQ      4$              ;YES-NEXT TEST.
(1) 007500 104012      ERROR    12              ;DR11K #1 CSR FAILED TO CLEAR.
(1) 007502 000137 007772      JMP      11$
(1) 007506 032737 000004 001564 4$:     BIT      #BIT2,SR2      ;DOES THIS DR11 HAVE A LOOP BACK
(1) ;CABLE CONNECTED TO IT?
(3) 007514 001526      BEQ      TST11          ;:
(1) ;IF SR2 BIT2=1 THEN LOOP BACK.
(1) 007516 012737 052525 001124      MOV      #052525,$GDDAT ;SET FIRST PATTERN
(2) ;*      MOV      $GDDAT,@DPOA1  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DPOA1
(2) ;*      MOV      @DRCR1,$BDDAT  ;/READ DEVICE REG DRCR1,PUT DATA IN $BDDAT.
(1) 007544 012737 000200 001124      MOV      #BIT7,$GDDAT  ;EXPECT FLAGS TO SET.
(1) 007552 023737 001126 001124      CMP      $BDDAT,$GDDAT ;DID THEY?
(1) 007560 001402      BEQ      5$              ;YES-CON'T.
(1) 007562 104012      ERROR    12              ;DR11K #1 FLAG(S) FAILED TO SET
(1) ;WHEN DATA XFERRD. IS THE OUTPUT
(1) ;REALLY CABLED BACK TO THE INPUT?
(3) 007564 000502      BR       TST11          ;:
(1) 007566 012737 052525 001124 5$:     MOV      #052525,$GDDAT ;PATTERN THAT SHOULD HAVE XFERRD.
(2) ;*      MOV      @DRIA1,$BDDAT  ;/READ DEVICE REG DRIA1,PUT DATA IN $BDDAT.
(1) 007604 023737 001126 001124      CMP      $BDDAT,$GDDAT ;DATA SENT=DATA RECEIVED?
(1) 007612 001402      BEQ      6$
(1) 007614 104012      ERROR    12              ;DR11K #1 DATA XFERR ERROR.
(3) 007616 000465      BR       TST11          ;:
(1) 007620      6$:
(2) ;*      MOV      @DRCR1,$BDDAT  ;/READ DEVICE REG DRCR1,PUT DATA IN $BDDAT.
(1) 007630 005737 001126      TST     $BDDAT          ;DID 'OUTPUT FLAG' SET.
(1) 007634 100402      BMI     7$              ;SHOULD HAVE.
(1)

```

```

(1) 007636 104012          ERROR 12          ;DR11K #1, 'OUTPUT FLAG' FAILED TO SET.
(3) 007640 000454          BR    TST11       ;;
(1)
(1) 007642                7$:
(2)
(2)
(1) 007652 012737 125252 001124 ;*   MOV    $GDDAT,@DRIA1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRIA1
(2)   MOV    #125252,$GDDAT ;NEW PATTERN
(2)
(2)   ;*   MOV    $GDDAT,@DROA1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA1
(2)
(2)   ;*   MOV    @DRIA1,$BDDAT ;/READ DEVICE REG DRIA1,PUT DATA IN $BDDAT.
(1) 007700 023737 001126 001124  CMP    $BDDAT,$GDDAT ;PATTERN XFERR OK?
(1) 007706 001402          BEQ    8$
(1)
(1) 007710 104012          ERROR 12          ;DR11K #1 DATA XFERR ERROR
(3) 007712 000427          BR    TST11       ;;
(1) 007714                8$:
(2)
(2)   ;*   MOV    $GDDAT,@DRIA1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRIA1
(1)
(1) 007724 005037 001124          CLR    $GDDAT      ;NOW XFERR ZERO PATTERN.
(2)
(2)   ;*   MOV    $GDDAT,@DROA1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA1
(2)
(2)   ;*   MOV    @DRIA1,$BDDAT ;/READ DEVICE REG DRIA1,PUT DATA IN $BDDAT.
(1) 007750 005737 001126          TST    $BDDAT      ;DID ZERO PATTERN GET XFERRERD?
(1) 007754 001402          BEQ    9$
(1)
(1) 007756 104012          ERROR 12          ;DR11K #1 DATA XFERR ERROR
(3) 007760 000404          BR    TST11       ;;
(1)
(1) 007762                9$:
(2)
(2)   ;*   MOV    $GDDAT,@DPCR1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DPCR1
(1)
(1) 007772                11$:
  
```


2859
(5)
(4)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(5)
(5)
(5)
(5)
(4)
(3)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(2)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(2)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(2)
(2)
(2)
(1)
(1)
(1)
(1)
(1)
(1)

007772 000004
007774 032737 000040 001562
010002 001002
010004 000137 010500
010010
010010 005037 001514
010014 005037 001124

010030 005737 037442
010034 001403

010036 104012

010040 000137 010500
010044

010054 005737 001126
010060 001403

010062 104012
010064 000137 010500

010070

010110 012737 040100 001124

```
*****  
*TEST 11 *TEST THE DR11K OPTION,#2, IF 'SR1' BIT 5=1(SET)  
*  
*IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.  
*WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE  
*KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS  
*NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.  
*THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE  
*I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE  
*IT FROM 'SR1' (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN  
*HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE  
*I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO 'SR1'  
*IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE  
*INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.  
*  
*IN THIS TEST WE WILL CHECK OUT DR11 #2 IF BIT 5 OF SR1=1  
*(SET). ALSO, DATA LOOP BACK (OUTPUT DATA BACK TO INPUT) WILL BE CHECKED  
*IF 'SR2' BIT 5 =1 (SET). FOR THIS, YOU MUST HAVE INSTALLED A LOOP-BACK  
*CABLE.  
*****  
TST11: SCOPE  
BIT #BITS,SR1  
BNE 10$ ;/-TDR1-  
JMP 11$  
  
10$:  
CLR .DVLS  
CLR $GDDAT ;/SET TO XFERR ZEROS.  
  
;* MOV $GDDAT,@DRCR2 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR2  
TST $AERR ;=1 IF DEVICE NOT PRESENT.  
BEQ 1$ ;IF 0, PRESENT, (CON'T).  
  
ERROR 12 ;DR11K #2 DID NOT RESPOND WHEN  
;ADDRESSED. 'SR1' BIT 5 SELECTED  
;THIS OPTION. ADDRESS 'DRCR'2  
;CONTAINS ITS ADDRESS  
  
JMP 11$  
  
1$:  
;* MOV @DRCR2,$BDDAT ;/READ DEVICE REG DRCR2,PUT DATA IN $BDDAT.  
TST $BDDAT ;DID CSR CLEAR?  
BEQ 2$  
  
ERROR 12 ;DR11K #2 CSR FAILED TO CLEAR.  
JMP 11$  
  
2$:  
;* MOV @DROA2,$GDDAT ;/READ DEVICE REG DROA2,PUT DATA IN $GDDAT.  
;* MOV @DRIA2,$GDDAT ;/READ DEVICE REG DRIA2,PUT DATA IN $GDDAT.  
MOV #BIT6!BIT14,$GDDAT ;LOAD WRITEABLE BITS.
```

```

(2) ;* MOV $GDDAT,@DRCR2 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR2
(2) ;* MOV @DRCR2,$BDDAT ;/READ DEVICE REG DRCR2,PUT DATA IN $BDDAT.
(2) ;* CMP $GDDAT,$BDDAT ;DID BITS SET?
(1) 010136 023737 001124 001126 ;* BEQ 3$
(1) 010144 001403
(1) 010146 104012 ERROR 12 ;DR11K #2 CSR BIT(S) FAILED.
(1) 010150 000137 010500 JMP 11$
(1) 010154 005037 001124 3$: CLR $GDDAT ;TRY CLEARING CSR.
(2) ;* MOV $GDDAT,@DRCR2 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR2
(2) ;* MOV @DRCR2,$BDDAT ;/READ DEVICE REG DRCR2,PUT DATA IN $BDDAT.
(1) 010200 005737 001126 ;* TST $BDDAT ;DID CSR CLEAR?
(1) 010204 001403 ;* BEQ 4$ ;YES-NEXT TEST.
(1) 010206 104012 ERROR 12 ;DR11K #2 CSR FAILED TO CLEAR.
(1) 010210 000137 010500 JMP 11$
(1) 010214 032737 000040 001564 4$: BIT #BIT5,SR2 ;DOES THIS DR11 HAVE A LOOP BACK
(1) ;* BEQ TST12 ;CABLE CONNECTED TO IT?
(3) 010222 001526 ;* ;IF SR2 BIT5=1 THEN LOOP BACK.
(1) 010224 012737 052525 001124 MOV #052525,$GDDAT ;SET FIRST PATTERN
(2) ;* MOV $GDDAT,@DROA2 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA2
(2) ;* MOV @DRCR2,$BDDAT ;/READ DEVICE REG DRCR2,PUT DATA IN $BDDAT.
(1) 010252 012737 000200 001124 ;* MOV #BIT7,$GDDAT ;EXPECT FLAGS TO SET.
(1) 010260 023737 001126 001124 ;* CMP $BDDAT,$GDDAT ;DID THEY?
(1) 010266 001402 ;* BEQ 5$ ;YES-CON'T.
(1) 010270 104012 ERROR 12 ;DR11K #2 FLAG(S) FAILED TO SET
(1) ;* WHEN DATA XFERRD. IS THE OUTPUT
(1) ;* REALLY CABLED BACK TO THE INPUT?
(3) 010272 000502 ;* BR TST12 ;*
(1) 010274 012737 052525 001124 5$: MOV #052525,$GDDAT ;PATTERN THAT SHOULD HAVE XFERRD.
(2) ;* MOV @DRIA2,$BDDAT ;/READ DEVICE REG DRIA2,PUT DATA IN $BDDAT.
(1) 010312 023737 001126 001124 ;* CMP $BDDAT,$GDDAT ;DATA SENT=DATA RECEIVED?
(1) 010320 001402 ;* BEQ 6$
(1) 010322 104012 ERROR 12 ;DR11K #2 DATA XFERR ERROR.
(3) 010324 000465 ;* BR TST12 ;*
(1) 010326 6$:
(2) ;* MOV @DRCR2,$BDDAT ;/READ DEVICE REG DRCR2,PUT DATA IN $BDDAT.
(1) 010336 005737 001126 ;* TST $BDDAT ;DID "OUTPUT FLAG" SET.
(1) 010342 100402 ;* BMI 7$ ;SHOULD HAVE.
(1) 010344 104012 ERROR 12 ;DR11K #2, "OUTPUT FLAG" FAILED TO SET.
(3) 010346 000454 ;* BR TST12 ;*
    
```

```

(1)
(1) 010350          7$:
(2)
(2)
(1) 010360 012737 125252 001124 ;*   MOV   $GDDAT,@DRIA2  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRIA2
(2)   MOV   #125252,$GDDAT ;NEW PATTERN
(2)
(2) ;*   MOV   $GDDAT,@DROA2  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA2
(2)
(2) ;*   MOV   @DRIA2,$BDDAT  ;/READ DEVICE REG DRIA2,PUT DATA IN $BDDAT.
(1) 010406 023737 001126 001124   CMP   $BDDAT,$GDDAT  ;PATTERN XFERR OK?
(1) 010414 001402          BEQ   8$
(1)
(1) 010416 104012          ERROR  12          ;DR11K #2 DATA XFERR ERROR
(3) 010420 000427          BR    TST12         ;;
(1) 010422          8$:
(2)
(2) ;*   MOV   $GDDAT,@DRIA2  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRIA2
(1)
(1) 010432 005037 001124          CLR   $GDDAT        ;NOW XFERR ZERO PATTERN.
(2)
(2) ;*   MOV   $GDDAT,@DROA2  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA2
(2)
(2) ;*   MOV   @DRIA2,$BDDAT  ;/READ DEVICE REG DRIA2,PUT DATA IN $BDDAT.
(1) 010456 005737 001126          TST   $BDDAT        ;DID ZERO PATTERN GET XFERRERD?
(1) 010462 001402          BEQ   9$
(1)
(1) 010464 104012          ERROR  12          ;DR11K #2 DATA XFERR ERROR
(3) 010466 000404          BR    TST12         ;;
(1)
(1) 010470          9$:
(2)
(2) ;*   MOV   $GDDAT,@DRLR2  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR2
(1)
(1) 010500          11$:
  
```

2861
(5)
(4)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(5)
(5)
(5)
(5)
(4)
(3)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(2)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(2)
(2)
(2)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(2)
(2)
(2)
(2)
(1)
(1)
(1)
(1)
(2)

010500 000004
010502 032737 000200 001562
010510 001002
010512 000137 011206
010516
010516 005037 001514
010522 005037 001124

010536 005737 037442
010542 001403

010544 104012

010546 000137 011206

010552

010562 005737 001126
010566 001403

010570 104012
010572 000137 011206

010576

010616 012737 040100 001124

```
*****
*TEST 12 *TEST THE DR11K OPTION,#3, IF 'SR1' BIT 7=1(SET)
*
*IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.
*WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE
*KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS
*NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.
*THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE
*I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE
*IT FROM 'SR1' (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN
*HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE
*I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO 'SR1'
*IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE
*INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.
*
*IN THIS TEST WE WILL CHECK OUT DR11 #3 IF BIT 7 OF SR1=1
*(SET). ALSO, DATA LOOP BACK (OUTPUT DATA BACK TO INPUT) WILL BE CHECKED
*IF 'SR2' BIT 7 =1 (SET). FOR THIS, YOU MUST HAVE INSTALLED A LOOP-BACK
*CABLE.
*****
TST12: SCOPE
        BIT      #BIT7,SR1
        BNE      10$      ;/-TDRT-
        JMP      11$
10$:
        CLR      .DVLS
        CLR      $GDDAT      ;/SET TO XFERR ZEROS.
;*
        MOV      $GDDAT,@DRCR3 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR3
        TST      $AERR        ;=1 IF DEVICE NOT PRESENT,
        BEQ      1$          ;IF 0, PRESENT, (CON'T).
        ERROR   12          ;DR11K #3 DID NOT RESPOND WHEN
                           ;ADDRESSED. 'SR1' BIT 7 SELECTED
                           ;THIS OPTION. ADDRESS 'DRCR'3
                           ;CONTAINS ITS ADDRESS
        JMP      11$
1$:
;*
        MOV      @DRCR3,$BDDAT ;/READ DEVICE REG DRCR3,PUT DATA IN $BDDAT.
        TST      $BDDAT        ;DID CSR CLEAR?
        BEQ      2$
        ERROR   12          ;DR11K #3 CSR FAILED TO CLEAR.
        JMP      11$
2$:
;*
        MOV      @DROA3,$GDDAT ;/READ DEVICE REG DROA3,PUT DATA IN $GDDAT.
;*
        MOV      @DRIA3,$GDDAT ;/READ DEVICE REG DRIA3,PUT DATA IN $GDDAT.
        MOV      #BIT6!BIT14,$GDDAT ;LOAD WRITEABLE BITS.
```



```

(1)
(1) 011056          7$:
(2)
(2)
(1) 011066 012737 125252 001124 ;*   MOV   $GDDAT,@DRIA3  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRIA3
      MOV   #125252,$GDDAT ;NEW PATTERN
(2)
(2)
(2) ;*   MOV   $GDDAT,@DROA3  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA3
(2)
(1) 011114 023737 001126 001124 ;*   MOV   @DRIA3,$BDDAT  ;/READ DEVICE REG DRIA3,PUT DATA IN $BDDAT.
(1) 011122 001402          CMP   $BDDAT,$GDDAT  ;PATTERN XFERR OK?
      BEQ   8$
(1)
(1) 011124 104012          ERROR 12          ;DR11K #3 DATA XFERR ERROR
(3) 011126 000427          BR    TST13         ;;
(1) 011130          8$:
(2)
(2) ;*   MOV   $GDDAT,@DRIA3  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRIA3
(1)
(1) 011140 005037 001124          CLR   $GDDAT          ;NOW XFERR ZERO PATTERN.
(2)
(2) ;*   MOV   $GDDAT,@DROA3  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA3
(2)
(2) ;*   MOV   @DRIA3,$BDDAT  ;/READ DEVICE REG DRIA3,PUT DATA IN $BDDAT.
(1) 011164 005737 001126          TST   $BDDAT          ;DID ZERO PATTERN GET XFERRERD?
(1) 011170 001402          BEQ   9$
(1)
(1) 011172 104012          ERROR 12          ;DR11K #3 DATA XFERR ERROR
(3) 011174 000404          BR    TST13         ;;
(1)
(1) 011176          9$:
(2)
(2) ;*   MOV   $GDDAT,@DRCLR3  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCLR3
(1)
(1) 011206          11$:
  
```

2863

```
(5)
(4)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(5)
(5)
(5)
(5)
(4)
(3) 011206 000004
(1) 011210 032737 000400 001562
(1) 011216 001002
(1) 011220 000137 011714
(1) 011224
(1) 011224 005037 001514
(1) 011230 005037 001124
(2)
(2)
(1) 011244 005737 037442
(1) 011250 001403
(1)
(1) 011252 104012
(1)
(1)
(1)
(1)
(1) 011254 000137 011714
(1)
(1) 011260
(2)
(2)
(1) 011270 005737 001126
(1) 011274 001403
(1)
(1) 011276 104012
(1) 011300 000137 011714
(1)
(1) 011304
(2)
(2)
(2)
(1) 011324 012737 040100 001124
(2)
```

```
*****
*TEST 13 *TEST THE DR11K OPTION,#4, IF 'SR1' BIT 8=1(SET)
*
*IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.
*WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE
*KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS
*NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.
*THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE
*I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE
*IT FROM 'SR1' (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN
*HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE
*I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO 'SR1'
*IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE
*INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.
*
*IN THIS TEST WE WILL CHECK OUT DR11 #4 IF BIT 8 OF SR1=1
*(SET). ALSO, DATA LOOP BACK (OUTPUT DATA BACK TO INPUT) WILL BE CHECKED
*IF 'SR2' BIT 8 =1 (SET). FOR THIS, YOU MUST HAVE INSTALLED A LOOP-BACK
*CABLE.
*****
TST13: SCOPE
BIT #BIT8,SR1
BNE 10$ ;/-TDRT-
JMP 11$
10$:
CLR .DVLS
CLR $GDDAT ;/SET TO XFERR ZEROS.
;* MOV $GDDAT,@DRCR4 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR4
TST $AERR ;=1 IF DEVICE NOT PRESENT,
BEQ 1$ ;IF 0, PRESENT, (CON'T).
ERROR 12 ;DR11K #4 DID NOT RESPOND WHEN
;ADDRESSED. 'SR1' BIT 8 SELECTED
;THIS OPTION. ADDRESS 'DRCR'4
;CONTAINS ITS ADDRESS
JMP 11$
1$:
;* MOV @DRCR4,$BDDAT ;/READ DEVICE REG DRCR4,PUT DATA IN $BDDAT.
TST $BDDAT ;DID CSR CLEAR?
BEQ 2$
ERROR 12 ;DR11K #4 CSR FAILED TO CLEAR.
JMP 11$
2$:
;* MOV @DROA4,$GDDAT ;/READ DEVICE REG DROA4,PUT DATA IN $GDDAT.
;* MOV @DRIA4,$GDDAT ;/READ DEVICE REG DRIA4,PUT DATA IN $GDDAT.
MOV #BIT6!BIT14,$GDDAT ;LOAD WRITEABLE BITS.
```

```

(2)          :*      MOV      $GDDAT,@DRCR4      ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR4
(2)
(2)          :*      MOV      @DRCR4,$BDDAT      ;/READ DEVICE REG DRCR4,PUT DATA IN $BDDAT.
(1) 011352 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;DID BITS SET?
(1) 011360 001403                      BEQ      3$
(1)
(1) 011362 104012                      ERROR   12                      ;DR11K #4 CSR BIT(S) FAILED.
(1) 011364 000137 011714                      JMP      11$
(1)
(1) 011370 005037 001124      3$:    CLR      $GDDAT                      ;TRY CLEARING CSR.
(2)
(2)          :*      MOV      $GDDAT,@DRCR4      ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR4
(2)
(2)          :*      MOV      @DRCR4,$BDDAT      ;/READ DEVICE REG DRCR4,PUT DATA IN $BDDAT.
(1) 011414 005737 001126      TST      $BDDAT                      ;DID CSR CLEAR?
(1) 011420 001403                      BEQ      4$                      ;YES-NEXT TEST.
(1)
(1) 011422 104012                      ERROR   12                      ;DR11K #4 CSR FAILED TO CLEAR.
(1) 011424 000137 011714                      JMP      11$
(1)
(1) 011430 032737 000400 001564 4$:    BIT      #BIT8,SR2                      ;DOES THIS DR11 HAVE A LOOP BACK
(1)                                     ;CABLE CONNECTED TO IT?
(3) 011436 001526                      BEQ      TST14                      ;:
(1)                                     ;IF SR2 BIT8=1 THEN LOOP BACK.
(1)
(1) 011440 012737 052525 001124      MOV      #052525,$GDDAT              ;SET FIRST PATTERN
(2)
(2)          :*      MOV      $GDDAT,@DROA4      ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA4
(2)
(2)          :*      MOV      @DRCR4,$BDDAT      ;/READ DEVICE REG DRCR4,PUT DATA IN $BDDAT.
(1) 011466 012737 000200 001124      MOV      #BIT7,$GDDAT              ;EXPECT FLAGS TO SET.
(1) 011474 023737 001126 001124      CMP      $BDDAT,$GDDAT              ;DID THEY?
(1) 011502 001402                      BEQ      5$                      ;YES-CON'T.
(1)
(1) 011504 104012                      ERROR   12                      ;DR11K #4 FLAG(S) FAILED TO SET
(1)                                     ;WHEN DATA XFERRD. IS THE OUTPUT
(1)                                     ;REALLY CABLED BACK TO THE INPUT?
(3) 011506 000502                      BR       TST14                      ;:
(1)
(1) 011510 012737 052525 001124 5$:    MOV      #052525,$GDDAT              ;PATTERN THAT SHOULD HAVE XFERRD.
(2)
(2)          :*      MOV      @DRIA4,$BDDAT      ;/READ DEVICE REG DRIA4,PUT DATA IN $BDDAT.
(1) 011526 023737 001126 001124      CMP      $BDDAT,$GDDAT              ;DATA SENT=DATA RECEIVED?
(1) 011534 001402                      BEQ      6$
(1)
(1) 011536 104012                      ERROR   12                      ;DR11K #4 DATA XFERR ERROR.
(3) 011540 000465                      BR       TST14                      ;:
(1)
(1) 011542          6$:
(2)
(2)          :*      MOV      @DRCR4,$BDDAT      ;/READ DEVICE REG DRCR4,PUT DATA IN $BDDAT.
(1) 011552 005737 001126      TST      $BDDAT                      ;DID 'OUTPUT FLAG' SET.
(1) 011556 100402                      BMI      7$                      ;SHOULD HAVE.
(1)
(1) 011560 104012                      ERROR   12                      ;DR11K #4, 'OUTPUT FLAG' FAILED TO SET.
(3) 011562 000454                      BR       TST14                      ;:

```



```

(1)
(1) 011564          7$:
(2)
(2)                :*      MOV      $GDDAT,@DRIA4    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRIA4
(1) 011574 012737 125252 001124      MOV      #125252,$GDDAT    ;NEW PATTERN
(2)
(2)                :*      MOV      $GDDAT,@DROA4    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA4
(2)
(2)                :*      MOV      @DRIA4,$BDDAT    ;/READ DEVICE REG DRIA4,PUT DATA IN $BDDAT.
(1) 011622 023737 001126 001124      CMP      $BDDAT,$GDDAT    ;PATTERN XFERR OK?
(1) 011630 001402      BEQ      8$
(1)
(1) 011632 104012      ERROR   12                ;DR11K #4 DATA XFERR ERROR
(3) 011634 000427      BR      TST14              ;;
(1) 011636          8$:
(2)
(2)                :*      MOV      $GDDAT,@DRIA4    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRIA4
(1) 011646 005037 001124      CLR      $GDDAT          ;NOW XFERR ZERO PATTERN.
(2)
(2)                :*      MOV      $GDDAT,@DROA4    ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA4
(2)
(2)                :*      MOV      @DRIA4,$BDDAT    ;/READ DEVICE REG-DRIA4,PUT DATA IN $BDDAT.
(1) 011672 005737 001126      TST      $BDDAT          ;DID ZERO PATTERN GET XFERRERD?
(1) 011676 001402      BEQ      9$
(1)
(1) 011700 104012      ERROR   12                ;DR11K #4 DATA XFERR ERROR
(3) 011702 000404      BR      TST14              ;;
(1)
(1) 011704          9$:
(2)
(2)                :*      MOV      $GDDAT,@DRCLR4   ;/ PUT DATA FROM $CDDAT TO DEVICE REG DRCLR4
(1)
(1) 011714          11$:
  
```

2865
 (5)
 (4)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (6)
 (5)
 (5)
 (5)
 (5)
 (4)
 (3) 011714 000004
 (1) 011716 032737 001000 001562
 (1) 011724 001002
 (1) 011726 000137 012422
 (1) 011732
 (1) 011732 005037 001514
 (1) 011736 005037 001124
 (2)
 (2)
 (1) 011752 005737 037442
 (1) 011756 001403
 (1)
 (1) 011760 104012
 (1)
 (1)
 (1)
 (1)
 (1)
 (1) 011762 000137 012422
 (1)
 (1) 011766
 (2)
 (2)
 (1) 011776 005737 001126
 (1) 012002 001403
 (1)
 (1) 012004 104012
 (1) 012006 000137 012422
 (1)
 (1) 012012
 (2)
 (2)
 (2)
 (2)
 (1) 012032 012737 040100 001124
 (2)

```

*****
*TEST 14 *TEST THE DR11K OPTION,#5, IF 'SR1' BIT 9=1(SET)
*
*IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.
*WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE
*KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS
*NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.
*THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE
*I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE
*IT FROM 'SR1' (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN
*HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE
*I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO 'SR1'
*IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE
*INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.
*
*IN THIS TEST WE WILL CHECK OUT DR11 #5 IF BIT 9 OF SR1=1
*(SET). ALSO, DATA LOOP BACK (OUTPUT DATA BACK TO INPUT) WILL BE CHECKED
*IF 'SR2' BIT 9 =1 (SET). FOR THIS, YOU MUST HAVE INSTALLED A LOOP-BACK
*CABLE.
*****
TST14: SCOPE
BIT #BIT9,SR1
BNE 10$ ;/-TDRT-
JMP 11$
10$:
CLR .DVLS
CLR $GDDAT ;/SET TO XFERR ZEROS.
;* MOV $GDDAT,@DRCR5 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR5
TST $AERR ;=1 IF DEVICE NOT PRESENT,
BEQ 1$ ;IF 0, PRESENT, (CON'T).
ERROR 12 ;DR11K #5 DID NOT RESPOND WHEN
;ADDRESSED. 'SR1' BIT 9 SELECTED
;THIS OPTION. ADDRESS 'DRCR'S
;CONTAINS ITS ADDRESS
JMP 11$
1$:
;* MOV @DRCR5,$BDDAT ;/READ DEVICE REG DRCR5,PUT DATA IN $BDDAT.
TST $BDDAT ;DID CSR CLEAR?
BEQ 2$
ERROR 12 ;DR11K #5 CSR FAILED TO CLEAR.
JMP 11$
2$:
;* MOV @DROA5,$GDDAT ;/READ DEVICE REG DROA5,PUT DATA IN $GDDAT.
;* MOV @DRIA5,$GDDAT ;/READ DEVICE REG DRIA5,PUT DATA IN $GDDAT.
MOV #BIT6!BIT14,$GDDAT ;LOAD WRITEABLE BITS.
    
```

```

(2) ;* MOV $GDDAT,@DRCR5 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR5
(2) ;* MOV @DRCR5,$BDDAT ;/READ DEVICE REG DRCR5,PUT DATA IN $BDDAT.
(2) ;* MOV @DRCR5,$BDDAT ;/READ DEVICE REG DRCR5,PUT DATA IN $BDDAT.
(1) 012060 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID BITS SET?
(1) 012066 001403 BEQ 3$
(1) 012070 104012 ERROR 12 ;DR11K #5 CSR BIT(S) FAILED.
(1) 012072 000137 012422 JMP 11$
(1) 012076 005037 001124 3$: CLR $GDDAT ;TRY CLEARING CSR.
(2) ;* MOV $GDDAT,@DRCR5 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCR5
(2) ;* MOV @DRCR5,$BDDAT ;/READ DEVICE REG DRCR5,PUT DATA IN $BDDAT.
(1) 012122 005737 001126 TST $BDDAT ;DID CSR CLEAR?
(1) 012126 001403 BEQ 4$ ;YES-NEXT TEST.
(1) 012130 104012 ERROR 12 ;DR11K #5 CSR FAILED TO CLEAR.
(1) 012132 000137 012422 JMP 11$
(1) 012136 032737 001000 001564 4$: BIT #BIT9,SR2 ;DOES THIS DR11 HAVE A LOOP BACK
(1) ;CABLE CONNECTED TO IT?
(3) 012144 001526 BEQ TST15 ;:
(1) ;IF SR2 BIT9=1 THEN LOOP BACK.
(1) 012146 012737 052525 001124 MOV #052525,$GDDAT ;SET FIRST PATTERN
(2) ;* MOV $GDDAT,@DROA5 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA5
(2) ;* MOV @DRCR5,$BDDAT ;/READ DEVICE REG DRCR5,PUT DATA IN $BDDAT.
(1) 012174 012737 000200 001124 MOV #BIT7,$GDDAT ;EXPECT FLAGS TO SET.
(1) 012202 023737 001126 001124 CMP $BDDAT,$GDDAT ;DID THEY?
(1) 012210 001402 BEQ 5$ ;YES-CON'T.
(1) 012212 104012 ERROR 12 ;DR11K #5 FLAG(S) FAILED TO SET
(1) ;WHEN DATA XFERRERD. IS THE OUTPUT
(1) ;REALLY CABLED BACK TO THE INPUT?
(3) 012214 000502 BR TST15 ;:
(1) 012216 012737 052525 001124 5$: MOV #052525,$GDDAT ;PATTERN THAT SHOULD HAVE XFERRERD.
(2) ;* MOV @DRIA5,$BDDAT ;/READ DEVICE REG DRIA5,PUT DATA IN $BDDAT.
(1) 012234 023737 001126 001124 CMP $BDDAT,$GDDAT ;DATA SENT=DATA RECEIVED?
(1) 012242 001402 BEQ 6$
(1) 012244 104012 ERROR 12 ;DR11K #5 DATA XFERR ERROR.
(3) 012246 000465 BR TST15 ;:
(1) 012250 6$:
(2) ;* MOV @DRCR5,$BDDAT ;/READ DEVICE REG DRCR5,PUT DATA IN $BDDAT.
(1) 012260 005737 001126 TST $BDDAT ;DID 'OUTPUT FLAG' SET.
(1) 012264 100402 BMI 7$ ;SHOULD HAVE.
(1) 012266 104012 ERROR 12 ;DR11K #5, 'OUTPUT FLAG' FAILED TO SET.
(3) 012270 000454 BR TST15 ;:
  
```

```

(1)
(1) 012272          7$:
(2)
(2)
(1) 012302 012737 125252 001124 ;*   MOV   $GDDAT,@DRIAS  ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRIAS
(2)   MOV   #125252,$GDDAT ;NEW PATTERN
(2)
(2) ;*   MOV   $GDDAT,@DROA5 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA5
(2)
(2) ;*   MOV   @DRIAS,$BDDAT ;/READ DEVICE REG DRIAS,PUT DATA IN $BDDAT.
(1) 012330 023737 001126 001124   CMP   $BDDAT,$GDDAT ;PATTERN XFERR OK?
(1) 012336 001402
(1)
(1) 012340 104012          ERROR 12          ;DR11K #5 DATA XFERR ERROR
(3) 012342 000427          BR    TST15          ;;
(1) 012344          8$:
(2)
(2) ;*   MOV   $GDDAT,@DRIAS ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRIAS
(1)
(1) 012354 005037 001124          CLR   $GDDAT          ;NOW XFERR ZERO PATTERN.
(2)
(2) ;*   MOV   $GDDAT,@DROA5 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DROA5
(2)
(2) ;*   MOV   @DRIAS,$BDDAT ;/READ DEVICE REG DRIAS,PUT DATA IN $BDDAT.
(1) 012400 005737 001126          TST   $BDDAT          ;DID ZERO PATTERN GET XFERRERD?
(1) 012404 001402          BEQ   9$
(1)
(1) 012406 104012          ERROR 12          ;DR11K #5 DATA XFERR ERROR
(3) 012410 000404          BR    TST15          ;;
(1)
(1) 012412          9$:
(2)
(2) ;*   MOV   $GDDAT,@DRCLR5 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DRCLR5
(1)
(1) 012422          11$:
  
```

2867
2874
2875
(3)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(4)
(4)
(4)
(3)
(2)
2876
2877
2878
2879
2880
2881
2882
(1)
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
(1)
2893
2894
2895
2896
2897
2898
2899
2900
(1)
2901
(1)
2902
2903
2904
2905

012422 000004
012424 032737 000010 001562
012432 001002
012434 000137 013452
012440
012440 005037 001124
012454 005737 037442
012460 001403
012462 104014
012464 000137 013452
012470
012500 042737 000200 001126
012506 001403
012510 104014
012512 000137 013452
012516 012737 005124 001124
012544 042737 000200 001126
012552 023737 001124 001126
012560 001403

```
*****  
: *TEST 15 *TEST THE AA-11 OPTION, IF 'SR1' BIT 3=1 (SET)  
:  
: *IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.  
: *WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE  
: *KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS  
: *NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.  
: *THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE  
: *I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE  
: *IT FROM 'SR1' (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN  
: *HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE  
: *I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO 'SR1'  
: *IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE  
: *INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.  
:  
: *IN THIS TEST WE WILL CHECK OUT THE AA-11 OPTION,  
: *IF BIT 3 OF 'SR1' IS SET.  
:  
:*****  
TST15: SCOPE  
BIT #BIT3,SR1 ;IS AA-11 SELECTED  
BNE 20$ ;IF =0, NO, SKIP THIS TEST.  
JMP 19$  
20$:  
CLR $GDDAT  
;* MOV $GDDAT,@AACSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG AACSR  
TST $AERR ;DID WE SUCCESSFULLY WRITE THE AA-11?  
BEQ 1$ ;YES IF $AERR=0  
ERROR 14 ;FAILED TO ADDRESS AA-11. 'SR1' BIT 3=1  
;SELECTED THIS TEST. AA-11 ADDRESS IS  
;IN LOC 'AACSR'  
JMP 19$  
1$:  
;* MOV @AACSR,$BDDAT ;/READ DEVICE REG AACSR,PUT DATA IN $BDDAT.  
BIC #BIT7,$BDDAT ;CLEAR FLAG  
BEQ 2$ ;DID CSR CLEAR?  
ERROR 14 ;CLEAR CSR FAILED (AA-11K)  
JMP 19$  
2$:  
MOV #5124,$GDDAT ;TRY WRITING THESE BITS  
;* MOV $GDDAT,@AACSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG AACSR  
;* MOV @AACSR,$BDDAT ;/READ DEVICE REG AACSR,PUT DATA IN $BDDAT.  
;READ THEM BACK  
BIC #BIT7,$BDDAT ;GET RID OF FLAG  
CMP $GDDAT,$BDDAT ;DID PATTERN W/R OK?  
BEQ 3$ ;YES-NEXT TEST.
```

```

2906
2907 012562 104014          ERROR 14          ;AA11-K PATTERN 5124 FAILED TO W/R PROPERLY.
2908 012564 000137 013452    JMP 19$
2909
2910 012570 012737 002012 001124 3$: MOV #2012,$GDDAT ;TRY NEW PATTERN.
2911
(1) ;* MOV $GDDAT,@AACSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG AACSR
2912
(1) ;* MOV @AACSR,$BDDAT ;/READ DEVICE REG AACSR,PUT DATA IN $BDDAT.
2913 012616 042737 000200 001126 BIC #BIT7,$BDDAT ;GET RID OF FLAG.
2914 012624 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID PATTERN XFERR OK?
2915 012632 001403 BEQ 4$ ;YES-NEXT TEST.
2916
2917 012634 104014          ERROR 14          ;AA-11K PATTERN 2012 FAILED TO W/R PROPERLY.
2918 012636 000137 013452    JMP 19$
2919
2920 012642 005037 001124      4$: CLR $GDDAT ;TRY TO CLEAR CSR.
2921
(1) ;* MOV $GDDAT,@AACSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG AACSR
2922
(1) ;* MOV @AACSR,$BDDAT ;/READ DEVICE REG AACSR,PUT DATA IN $BDDAT.
2923 012666 042737 000200 001126 BIC #BIT7,$BDDAT ;CLEAR FLAG BIT.
2924 012674 001403 BEQ 5$
2925
2926 012676 104014          ERROR 14          ;AA-11K CSR FAILED TO CLEAR.
2927 012700 000137 013452    JMP 19$
2928
2929 012704 012737 005252 001124 5$: MOV #5252,$GDDAT
2930
(1) ;* MOV $GDDAT,@DAC0 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC0
2931
(1) ;* MOV @DAC0,$BDDAT ;/READ DEVICE REG DAC0,PUT DATA IN $BDDAT.
2932 012732 023737 001124 001126 CMP $GDDAT,$BDDAT
2933 012740 001403 BEQ 6$ ;OK?
2934
2935 012742 104014          ERROR 14          ;AA-11K DAC0 FAILED TO SET PATTERN 5252.
2936 012744 000137 013452    JMP 19$
2937
2938 012750      6$:
(1)
(1) ;* MOV $GDDAT,@DAC1 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC1
2939
(1) ;* MOV @DAC1,$BDDAT ;/READ DEVICE REG DAC1,PUT DATA IN $BDDAT.
2940 012770 023737 001126 001124 CMP $BDDAT,$GDDAT
2941 012776 001403 BEQ 7$ ;OK
;YES-
2942
2943 013000 104014          ERROR 14          ;AA-11K DAC1 FAILED PATTERN 5252
2944 013002 000137 013452    JMP 19$
2945
2946 013006      7$:
(1)
(1) ;* MOV $GDDAT,@DAC2 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC2
2947
(1) ;* MOV @DAC2,$BDDAT ;/READ DEVICE REG DAC2,PUT DATA IN $BDDAT.
2948 013026 023737 001126 001124 CMP $BDDAT,$GDDAT
2949 013034 001403 BEQ 8$ ;OK?
  
```

```

2950
2951 013036 104014          ERROR 14          ;AA-11K DAC2 FAILED PATTERN 5252
2952 013040 000137 013452  JMP     19$
2953
2954 013044          8$:
(1)
(1)
2955          ;*   MOV     $GDDAT,@DAC3      ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC3
(1)          ;*   MOV     @DAC3,$BDDAT      ;/READ DEVICE REG DAC3,PUT DATA IN $BDDAT.
2956 013064 023737 001126 001124  CMP     $BDDAT,$GDDAT ;PATTERN OK?
2957 013072 001402          BEQ     9$
2958
2959 013074 104014          ERROR 14          ;AA-11K DAC3 FAILED PATTERN 5252
2960 013076 000565          BR      TST16      ;;
2961
2962 013100 012737 002525 001124 9$:   MOV     #2525,$GDDAT
2963          ;*   MOV     $GDDAT,@DAC0      ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC0
(1)          ;*   MOV     @DAC0,$BDDAT      ;/READ DEVICE REG DAC0,PUT DATA IN $BDDAT.
2964          ;*   MOV     $GDDAT,$BDDAT
(1)          ;*   CMP     $GDDAT,$BDDAT
2965 013126 023737 001124 001126  BEQ     10$      ;OK?
2966 013134 001402          ;OK?
2967
2968 013136 104014          ERROR 14          ;AA-11K DAC0 FAILED TO SET PATTERN 2525
2969 013140 000544          BR      TST16      ;;
2970
2971 013142          10$:
(1)
(1)
2972          ;*   MOV     $GDDAT,@DAC1      ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC1
(1)          ;*   MOV     @DAC1,$BDDAT      ;/READ DEVICE REG DAC1,PUT DATA IN $BDDAT.
2973 013162 023737 001126 001124  CMP     $BDDAT,$GDDAT ;OK?
2974 013170 001402          BEQ     11$      ;YES-
2975
2976 013172 104014          ERROR 14          ;AA-11K DAC1 FAILED PATTERN 2525
2977 013174 000526          BR      TST16      ;;
2978
2979 013176          11$:
(1)
(1)
2980          ;*   MOV     $GDDAT,@DAC2      ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC2
(1)          ;*   MOV     @DAC2,$BDDAT      ;/READ DEVICE REG DAC2,PUT DATA IN $BDDAT.
2981 013216 023737 001126 001124  CMP     $BDDAT,$GDDAT ;OK?
2982 013224 001402          BEQ     12$
2983
2984 013226 104014          ERROR 14          ;AA-11K DAC2 FAILED PATTERN 2525.
2985 013230 000510          BR      TST16      ;;
2986
2987 013232          12$:
(1)
(1)
2988          ;*   MOV     $GDDAT,@DAC3      ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC3
(1)          ;*   MOV     @DAC3,$BDDAT      ;/READ DEVICE REG DAC3,PUT DATA IN $BDDAT.
2989 013252 023737 001126 001124  CMP     $BDDAT,$GDDAT ;PATTERN OK?
2990 013260 001402          BEQ     13$
2991
  
```

```

2992 013262 104014          ERROR 14          :AA-11K DAC3 FAILED PATTERN.
2993                                     :2525
2994 013264 000472          BR    TST16         ;;
2995
2996 013266 012737 000000 001124 13$: MOV    #0,$GDDAT
2997                                     :/ PUT DATA FROM $GDDAT TO DEVICE REG DAC0
(1)                                     ;* MOV    $GDDAT,@DAC0
2998                                     :/READ DEVICE REG DAC0,PUT DATA IN $BDDAT.
(1)                                     ;* MOV    @DAC0,$BDDAT
2999 013314 023737 001124 001126  :* CMP    $GDDAT,$BDDAT
3000 013322 001402          BEQ    14$          :OK?
3001
3002 013324 104014          ERROR 14          :AA-11K DAC0 FAILED TO SET PATTERN 0
3003 013326 000451          BR    TST16         ;;
3004
3005 013330          14$:
(1)
(1)                                     ;* MOV    $GDDAT,@DAC1
3006                                     :/ PUT DATA FROM $GDDAT TO DEVICE REG DAC1
(1)                                     ;* MOV    @DAC1,$BDDAT
3007 013350 023737 001126 001124  :* CMP    $BDDAT,$GDDAT
3008 013356 001402          BEQ    15$          :OK?
3009                                     :YES-
3010 013360 104014          ERROR 14          :AA-11K DAC1 FAILED PATTERN 0
3011 013362 000433          BR    TST16         ;;
3012
3013 013364          15$:
(1)
(1)                                     ;* MOV    $GDDAT,@DAC2
3014                                     :/ PUT DATA FROM $GDDAT TO DEVICE REG DAC2
(1)                                     ;* MOV    @DAC2,$BDDAT
3015 013404 023737 001126 001124  :* CMP    $BDDAT,$GDDAT
3016 013412 001402          BEQ    16$          :OK?
3017
3018 013414 104014          ERROR 14          :AA-11K DAC2 FAILED PATTERN 0
3019 013416 000415          BR    TST16         ;;
3020
3021 013420          16$:
(1)
(1)                                     ;* MOV    $GDDAT,@DAC3
3022                                     :/ PUT DATA FROM $GDDAT TO DEVICE REG DAC3
(1)                                     ;* MOV    @DAC3,$BDDAT
3023 013440 023737 001126 001124  :* CMP    $BDDAT,$GDDAT
3024                                     :/READ DEVICE REG DAC3,PUT DATA IN $BDDAT.
3025 013446 001401          BEQ    TST16         ;;
3026 013450 104014          ERROR 14          :AA-11K DAC3 FAILED PATTERN 0.
3027
3028
3029
3030 013452          19$:
  
```



```

3032          ;:*****
(3)          ;*TEST 16      I/O DEVICE TEST-AR11 OPTION
(3)          ;:*****
(2) 013452 000004 TST16: SCOPE
3033
3034 013454 032737 002000 001562      BIT      #BIT10,SR1      ;IS THIS DEVICE SELECTED FOR TEST?
3035                                          ;NOTE: DEFAULT=YES.
3036
3037 013462 001002      BNE      10$
3038 013464 000137 014022      JMP      7$
3039 013470          10$:
3040 013470 005037 001126      CLR      $BDDAT
3041 013474 005037 001124      CLR      $GDDAT
3042
(1)          ;*      MOV      $BDDAT,@ARADS      ;/ PUT DATA FROM $BDDAT TO DEVICE REG ARADS
3043                                          ;OK-WHAT WE JUST DID IS ASKED THE
3044                                          ;SLAVE MICRO-CODE TO WRITE THE CONTENTS
3045                                          ;OF THE AR11 CSR TO ZEROES.
3046                                          ;IF THE AR11 FAILS TO RESPOND TO ITS
3047                                          ;ADDRESS-THE SLAVE WILL SEND US AN
3048                                          ;ERROR CODE THAT CAUSES US TO SET $AERR-1
3049                                          ;IN THE SUPPORT ROUTINES.
3050
3051 013510 005737 037442      TST      $AERR      ;DEVICE PRESENT?
3052 013514 001403      BEQ      11$      ;YES-CONTINUE
3053 013516 104015      ERROR   15      ;AR11 FAILED TO RESPOND
3054 013520 000137 014022      JMP      7$      ;TO ADDR.
3055 013524          11$:
3056
(1)          ;*      MOV      @ARADS,$BDDAT      ;/READ DEVICE REG ARADS,PUT DATA IN $BDDAT.
3057 013534 042737 000200 001126      BIC      #BIT7,$BDDAT      ;CLEAR OUT FLAG
3058 013542 005037 001124      CLR      $GDDAT
3059 013546 005737 001126      TST      $BDDAT      ;IS CSR ZERO?
3060 013552 001402      BEQ      1$
3061
3062 013554 104015      ERROR   15      ;FAILED TO ZERO AR11 A/D CSR
3063 013556 000521      BR      TST17      ;
3064 013560 012737 025100 001124 1$:      MOV      #BIT13!BIT11!BIT9!BIT6,$GDDAT ;NOW TRY A NEW BIT PATTERN
3065
3066
(1)          ;*      MOV      $GDDAT,@ARADS      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARADS
3067
(1)          ;*      MOV      @ARADS,$BDDAT      ;/READ DEVICE REG ARADS,PUT DATA IN $BDDAT.
3068 013606 042737 000200 001126      BIC      #BIT7,$BDDAT      ;IGNORE FLAG
3069 013614 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;BIT SET OK?
3070 013622 001402      BEQ      2$
3071
3072 013624 104015      ERROR   15      ;FAILED TO WRITE AR11 A/D CSR CORRECTLY
3073 013626 000475      BR      TST17      ;
3074
3075 013630 012737 002420 001124 2$:      MOV      #BIT10!BIT8!BIT4,$GDDAT ;NOW TRY A NEW BIT PATTERN.
3076
3077
(1)          ;*      MOV      $GDDAT,@ARADS      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARADS
3078
(1)          ;*      MOV      @ARADS,$BDDAT      ;/READ DEVICE REG ARADS,PUT DATA IN $BDDAT.

```

```

3079 013656 042737 000200 001126      BIC    #BIT7,$BDDAT    ;IGNORE FLAG
3080 013664 023737 001124 001126      CMP    $GDDAT,$BDDAT  ;BITS SET OK?
3081 013672 001402                      BEQ    3$
3082
3083 013674 104015                      ERROR  15              ;FAILED TO WRITE #'NY' AD11K CSR CORRECTLY
3084 013676 000451                      BR     TST17           ;;
3085
3086 013700 005037 001124      3$:   CLR    $GDDAT      ;NOW CLEAR ALL BITS SET
3087
(1) ;*   MOV    $GDDAT,@ARADS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARADS
3088
(1) ;*   MOV    @ARADS,$BDDAT ;/READ DEVICE REG ARADS,PUT DATA IN $BDDAT.
3089 013724 042737 000200 001126      BIC    #BIT7,$BDDAT  ;IGNORE DONE FLAG.
3090 013732 005737 001126      TST    $BDDAT        ;CSR CLEAR
3091 013736 001402                      BEQ    4$
3092
3093 013740 104015                      ERROR  15              ;AR11 A/D CSR FAILED TO CLEAR.
3094 013742 000427                      BR     TST17           ;;
3095
3096 013744                      4$:
(1)
(1) ;*   MOV    @ARADB,MYTEMP ;/READ DEVICE REG ARADB,PUT DATA IN MYTEMP.
3097 013754 012737 000001 001124      MOV    #BIT0,$GDDAT  ;NOW WE'LL SET THE A/D START BIT.
3098
(1) ;*   MOV    $GDDAT,@ARADS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARADS
3099 013772 012737 000200 001124      MOV    #BIT7,$GDDAT ;WHEN WE READ CSR BACK EXPECT
3100 ;START BIT TO CLEAR, DONE TO SET.
3101
(1) ;*   MOV    @ARADS,$BDDAT ;/READ DEVICE REG ARADS,PUT DATA IN $BDDAT.
3102
3103 014010 023737 001124 001126      CMP    $GDDAT,$BDDAT ;START OK?
3104 014016 001401                      BEQ    TST17           ;;
3105
3106 014020 104015                      ERROR  15              ;CSR BAD AFTER A/D START
3107 ;BIT 0 SHOULD CLEAR ONLY A/D DONE SET.
3108 014022                      7$:
3109
3137
3138

```

```

(3) ;*****
(4) ;*TEST 17      I/O DEVICE TEST-AR11 CLOCK OPTION
(5)
(5) ;*
(5) ;*IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.
(5) ;*WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE
(5) ;*KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS
(5) ;*NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.
(5) ;*THE DIAGNOSTIC IS SET UP FOR CERTAIN DFFAULT DEVICES ON THE
(5) ;*I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE
(5) ;*IT FROM 'SR1' (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN
(5) ;*HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE
(5) ;*I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO 'SR1'
(5) ;*IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE
(5) ;*INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.
(5) ;*
(4) ;*
(4) ;*      IN THIS TEST WE WILL EXERCISE THE AR11 CLOCK. IF
(4) ;*      SR1 IS UNALTERED, WE DO THIS TEST SINCE THE

```

(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(2) 014022 000004
3139
3140 014024 032737 002000 001562
3141
3142 014032 001002
3143 014034 000137 014364
3144 014040
3145
3146 014040 012737 040130 001124
3147
3148
(1)
3149
3150 014056 005737 037442
3151 014062 001403
3152
3153 014064 104015
3154 014066 000137 014364
3155 014072
3156
(1)
3157
3158 014102 023737 001124 001126
3159 014110 001403
3160
3161 014112 104015
3162 014114 000137 014364
3163
3164 014120 012737 000424 001124 1\$
3165
(1)
3166
(1)
3167

```

: * KW11K IS A DEFAULT I/O DEVICE.
: * PLEASE NOTE: THIS IS ONLY A BRIEF TEST OF THE AR11,
: * TO SEE THAT IT ROUGHLY WORKS, FOR TESTING, YOU
: * MUST RUN THE AR11 DIAGNOSTIC, COMPLETE
: *
: * TESTS:
: *      1. MAKE SURE AR11 RESPONDS TO ADDRESS
: *        (NOTE: IF THE AR11 ON YOU I/O BUS USES A NON
: *        STANDARD ADDRESS, MODIFY "ARADS" WITH THE CORR
: *
: *      2. CLOCK CSR BITS 14,6,3, AND 1 SET
: *
: *      3. CLOCK CSR BITS 8 AND 2 SET
: *
: *      4. CLOCK CSR ZEROS.
: *
: *      5. START CLOCK A MODE 0, RATE 1MHZ,
: *        CHECK "ENABLE CNTR A" CLEARS,
: *        "A OVERFLOW FLAG" SETS
: *
: *      6. CLOCK A CSR ZEROS
: *
: * *****
TST17: SCOPE
      BIT #BIT10,SR1 ;IS THIS DEVICE SELECTED FOR TEST?
      ;NOTE: DEFAUT=YES.
      BNE 10$
      JMP 11$
10$:
      MOV #BIT14!BIT6!BIT4!BIT3,$GDDAT ;TEST THESE BITS
: *
      MOV $GDDAT,@ARCS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARCS
: *
      TST $AERR ;DID AR11 RESPOND
      BEQ 12$ ;TO ADDR? $AERR=0?
      ;IF YES-NEXT TEST.
      ;AR11 CLOCK FAILED TO RESPOND TO ADDR.
      ERROR 15
      JMP 11$
12$:
: *
      MOV @ARCS,$BDDAT ;/READ DEVICE REG ARCS,PUT DATA IN $BDDAT.
: *
      CMP $GDDAT,$BDDAT ;DID THESE BITS SET?
      BEQ 1$
: *
      ERROR 15 ;AR11 CLOCK R/W CSR ERROR.
      JMP 11$
1$:
      MOV #BIT8!BIT4!BIT2,$GDDAT ;GET NEW PATTERN
: *
      MOV $GDDAT,@ARCS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARCS
: *
      MOV @ARCS,$BDDAT ;/READ DEVICE REG ARCS,PUT DATA IN $BDDAT.

```

```

3168 014146 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;DID THE BITS SET?
3169 014154 001403                    BEQ      2$
3170
3171 014156 104015                    ERROR   15                    ;AR11 CLOCK R/W CSR ERROR.
3172 014160 000137 014364              JMP      11$
3173
3174 014164 005037 001124              2$:    CLR      $GDDAT          ;WRITE ALL ZEROS.
3175
3176      (1)                          ;*    MOV      $GDDAT,@ARCS      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARCS
3177      (1)                          ;*    MOV      @ARCS,$BDDAT      ;/READ DEVICE REG ARCS,PUT DATA IN $BDDAT.
3177 014210 012737 000020 001124      MOV      #BIT4,$GDDAT      ;BIT 4 ALWAYS SET
3178 014216 023737 001124 001126      CMP      $GDDAT,$BDDAT
3179 014224 001403                    BEQ      6$                    ;YES-GOOD.
3180
3181 014226 104015                    ERROR   15                    ;AR11 CLOCK FAILED TO ZERO CSR.
3182 014230 000137 014364              JMP      11$
3183
3184 014234 012737 000003 001124      6$:    MOV      #BIT0!BIT1,$GDDAT ;PUT 'ENABL CNTR A' AND RATE 1MHZ IN CSR
3185 014242 012737 177777 001126      MOV      #177777,$BDDAT    ;PRELOAD CNTR.
3186
3187      (1)                          ;*    MOV      $BDDAT,@ARCB     ;/ PUT DATA FROM $BDDAT TO DEVICE REG ARCB
3188      (1)                          ;*    MOV      $GDDA1,@ARCS     ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARCS
3189      (1)                          ;*    MOV      @ARCS,$BDDAT     ;/READ DEVICE REG ARCS,PUT DATA IN $BDDAT.
3190 014300 012737 000222 001124      MOV      #BIT7!BIT4.BIT1,$GDDAT ;BIT 0 ('ENABLE CNTR A') SET
3191      (1)                          ;BIT 7
3192 014306 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;HAPPEN OK?
3193 014314 001401                    BEQ      7$
3194
3195 014316 104015                    ERROR   15                    ;CLOCK CSR PROBLEM
3196
3197 014320 005037 001124              7$:    CLR      $GDDAT          ;TRY CLEARING CSR
3198
3199      (1)                          ;*    MOV      $GDDAT,@ARCS     ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARCS
3200      (1)                          ;*    MOV      @ARCS,$BDDAT     ;/READ DEVICE REG ARCS,PUT DATA IN $BDDAT.
3200 014344 012737 000020 001124      MOV      #BIT4,$GDDA1      ;BIT 4 ALWAYS SET
3201 014352 023737 001124 001126      CMP      $GDDAT,$BDDAT
3202 014360 001401                    BEQ      TST20                ;;
3203
3204 014362 104015                    ERROR   15                    ;CSR FAILED TO CLEAR
3205 014364              11$:
3206
3207
3208
3209
3210
3211
3212
3213
3214

```

```

:*****
:*TEST 20      *TEST THE AR11 DISPLAY OPTION, IF 'SR1' BIT 10=1 (SET)
:*
:*IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.
:*WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE
:*KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS
:*NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.
:*THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE
:*I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE

```

(3)
(5)
(5)
(5)
(5)
(5)
(5)
(5)

```

(5) ;*IT FROM 'SR1' (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN
(5) ;*HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE
(5) ;*I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO 'SR1'
(5) ;*IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE
(5) ;*INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.
(5) ;*
(4) ;*IN THIS TEST WE WILL CHECK OUT THE AR11 DISPLAY OPTION,
(4) ;*IF BIT 10 OF 'SR1' IS SET.
(4) ;*
(3) ;*****
(2) 014364 000004 TST20: SCOPE
3215
3216 014366 032737 002000 001562 BIT #BIT10,SR1 ;IS AR-11 SELECTED
3217 014374 001002 BNE 20$ ;IF =0, NO, SKIP THIS TEST.
3218 014376 000137 015142 JMP 19$
3219 014402 20$:
3220 014402 005037 001124 CLR $GDDAT
3221
(1) ;* MOV $GDDAT,@ARDS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARDS
3222 014416 005737 037442 TST $AERR ;DID WE SUCCESSFULLY WRITE THE AR-11?
3223 014422 001403 BEQ 1$ ;YES IF $AERR=0
3224
3225 014424 104015 ERROR 15 ;FAILED TO ADDRESS AR-11. 'SR1' BIT 3=1
3226 ;SELECTED THIS TEST. AR-11 ADDRESS IS
3227 ;IN LOC 'ARADS'
3228 014426 000137 015142 JMP 19$
3229
3230 014432 1$:
3231
(1) ;* MOV @ARDS,$BDDAT ;/READ DEVICE REG ARDS,PUT DATA IN $BDDAT.
3232 014442 042737 000200 001126 BIC #BIT7,$BDDAT ;CLEAR FLAG
3233 014450 001403 BEQ 2$ ;DID CSR CLEAR?
3234
3235 014452 104015 ERROR 15 ;CLEAR CSR FAILED (AR-11)
3236 014454 000137 015142 JMP 19$
3237
3238 014460 012737 005104 001124 2$: MOV #5104,$GDDAT ;TRY WRITING THESE BITS
3239
(1) ;* MOV $GDDAT,@ARDS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARDS
3240
(1) ;* MOV @ARDS,$BDDAT ;/READ DEVICE REG ARDS,PUT DATA IN $BDDAT.
3241 ;READ THEM BACK
3242 014506 042737 000200 001126 BIC #BIT7,$BDDAT ;GET RID OF FLAG
3243 014514 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID PATTERN W/R OK?
3244 014522 001403 BEQ 3$ ;YES-NEXT TEST.
3245
3246 014524 104015 ERROR 15 ;AR11 D/A PATTERN 5024 FAILED TO W/R PROPERLY.
3247 014526 000137 015142 JMP 19$
3248
3249 014532 012737 002010 001124 3$: MOV #2010,$GDDAT ;TRY NEW PATTERN.
3250
(1) ;* MOV $GDDAT,@ARDS ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARDS
3251
(1) ;* MOV @ARDS,$BDDAT ;/READ DEVICE REG ARDS,PUT DATA IN $BDDAT.
3252 014560 042737 000200 001126 BIC #BIT7,$BDDAT ;GET RID OF FLAG.
3253 014566 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID PATTERN XFERR OK?
  
```

```

3254 014574 001403          BEQ      4$          ;YES-NEXT TEST.
3255
3256 014576 104015          ERROR    15          ;AR-11 DISPLAY REG PATTERN 2010 FAILED TO W/R PROPERLY.
3257 014600 000137 015142      JMP      19$
3258
3259 014604 005037 001124      4$:      CLR      $GDDAT      ;TRY TO CLEAR CSR.
3260
3261      (1)          ;*      MOV      $GDDAT,@ARDS      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARDS
3262      (1)          ;*      MOV      @ARDS,$BDDAT      ;/READ DEVICE REG ARDS,PUT DATA IN $BDDAT.
3262 014630 042737 000200 001126      BIC      #BIT7,$BDDAT      ;CLEAR FLAG BIT.
3263 014636 001403          BEQ      5$
3264
3265 014640 104015          ERROR    15          ;AR-11 DISPLAY REG. CSR FAILED TO CLEAR.
3266 014642 000137 015142      JMP      19$
3267
3268 014646 012737 001252 001124      5$:      MOV      #1252,$GDDAT
3269      (1)          ;*      MOV      $GDDAT,@ARXB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARXB
3270      (1)          ;*      MOV      @ARXB,$BDDAT      ;/READ DEVICE REG ARXB,PUT DATA IN $BDDAT.
3271 014674 023737 001124 001126      CMP      $GDDAT,$BDDAT
3272 014702 001403          BEQ      6$          ;OK?
3273
3274 014704 104015          ERROR    15          ;AR-11 DACX FAILED TO SET PATTERN 1252.
3275 014706 000137 015142      JMP      19$
3276
3277 014712          6$:
3278      (1)          ;*      MOV      $GDDAT,@ARYB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARYB
3279      (1)          ;*      MOV      @ARYB,$BDDAT      ;/READ DEVICE REG ARYB,PUT DATA IN $BDDAT.
3279 014732 023737 001126 001124      CMP      $BDDAT,$GDDAT      ;OK
3280 014740 001403          BEQ      7$          ;YES-
3281
3282 014742 104015          ERROR    15          ;AR-11 DACY FAILED PATTERN 1252
3283 014744 000137 015142      JMP      19$
3284
3285 014750 012737 000525 001124      7$:      MOV      #0525,$GDDAT
3286      (1)          ;*      MOV      $GDDAT,@ARXB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARXB
3287      (1)          ;*      MOV      @ARXB,$BDDAT      ;/READ DEVICE REG ARXB,PUT DATA IN $BDDAT.
3288 014776 023737 001124 001126      CMP      $GDDAT,$BDDAT
3289 015004 001402          BEQ      10$         ;OK?
3290
3291 015006 104015          ERROR    15          ;AR-11 DACX FAILED TO SET PATTERN 0525
3292 015010 000454          BR      TST21        ;:
3293 015012          10$:
3294      (1)          ;*      MOV      $GDDAT,@ARYB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARYB
3295      (1)          ;*      MOV      @ARYB,$BDDAT      ;/READ DEVICE REG ARYB,PUT DATA IN $BDDAT.
3295 015032 023737 001126 001124      CMP      $BDDAT,$GDDAT
3296 015040 001402          BEQ      13$         ;OK?
3297          ;YES-

```

```

3298 015042 104015      ERROR 15      ;AR-11 DACY FAILED PATTERN 0525
3299 015044 000436      BR      TST21      ;;
3300
3301 015046 012737 000000 001124 13$:  MOV      #0,$GDDAT
3302
(1) ;*      MOV      $GDDAT,@ARXB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARXB
3303
(1) ;*      MOV      @ARXB,$BDDAT      ;/READ DEVICE REG ARXB,PUT DATA IN $BDDAT.
3304 015074 023737 001124 001126  CMP      $GDDAT,$BDDAT
3305 015102 001402      BEQ      14$      ;OK?
3306
3307 015104 104015      ERROR 15      ;AR-11 DACX FAILED TO SET PATTERN 0
3308 015106 000415      BR      TST21      ;;
3309
3310 015110      14$:
(1)
(1) ;*      MOV      $GDDAT,@ARYB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARYB
3311
(1) ;*      MOV      @ARYB,$BDDAT      ;/READ DEVICE REG ARYB,PUT DATA IN $BDDAT.
3312 015130 023737 001126 001124  CMP      $BDDAT,$GDDAT      ;OK?
3313 015136 001401      BEQ      19$      ;YES-
3314
3315 015140 104015      ERROR 15      ;AR-11 DACY FAILED PATTERN 0
3316 015142      19$:
3317
3325
3326
(3) ;:*****
(4) ;*TEST 21      *TEST THAT THE AR11 CAN DISPLAY A SQUARE, SELECTED BY BIT 10 OF SR1,SR2
(4) ;*IN THIS TEST WE'LL DISPLAY A SQUARE ON THE
(4) ;*DISPLAY SCOPE VIA THE AR-11.
(4) ;*IF YOU HAVE AN AR11 AND SCOPE YOU MUST SELECT THIS
(4) ;*TEST BY SETTING THE APPROPRIATE BITS IN "SR1" (INDICATING YOU HAVE
(4) ;*AN AR-11) AND "SR2" (INDICATING YOU HAVE A SCOPE).
(3) ;:*****
(2) 015142 000004      TST21:  SCOPE
3327
3328 015144 032737 002000 001562      BIT      #BIT10,SR1      ;/AR-11 SELECTED?
3329 015152 001475      BEQ      TST22      ;;
3330 015154 032737 010000 001564      BIT      #BIT12,SR2      ;/SCOPE DISPLAY?
3331 015162 001471      BEQ      TST22      ;;
3332
3333 015164 005037 001124      CLR      $GDDAT
3334 015170 005037 001126      CLR      $BDDAT
3335 015174 012737 000001 001754      MOV      #1,MYTEMP
3336 015202      1$:
(1)
(1) ;*      MOV      $GDDAT,@ARXB      ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARXB
3337
(1) ;*      MOV      MYTEMP,@ARDS      ;/ PUT DATA FROM MYTEMP TO DEVICE REG ARDS
3338 015222 005237 001124      INC      $GDDAT
3339 015226 032737 002000 001124      BIT      #BIT10,$GDDAT
3340 015234 001762      BEQ      1$
3341      2$:
(1)
(1) ;*      MOV      $BDDAT,@ARYB      ;/ PUT DATA FROM $BDDAT TO DEVICE REG ARYB
3342
  
```

```

(1)
3343 015256 005237 001126 ;* MOV MYTEMP,@ARDS ;/ PUT DATA FROM MYTEMP TO DEVICE REG ARDS
3344 015262 032737 002000 001126 INC $BDDAT
3345 015270 001762 BIT #BIT10,$BDDAT
3346 015272 3$: BEQ 2$

(1)
(1) ;* MOV $GDDAT,@ARXB ;/ PUT DATA FROM $GDDAT TO DEVICE REG ARXB
3347 (1) ;* MOV MYTEMP,@ARDS ;/ PUT DATA FROM MYTEMP TO DEVICE REG ARDS
3348 015312 005337 001124 DEC $GDDAT
3349 015316 001365 BNE 3$
3350 015320 4$:

(1)
(1) ;* MOV $BDDAT,@ARYB ;/ PUT DATA FROM $BDDAT TO DEVICE REG ARYB
3351 (1) ;* MOV MYTEMP,@ARDS ;/ PUT DATA FROM MYTEMP TO DEVICE REG ARDS
3352 015340 005337 001126 DEC $BDDAT
3353 015344 001365 BNE 4$
3354
  
```



```

3356      ;*****
(3)      ;*TEST 22      I/O DEVICE TEST LPS-11 A/D OPTION
(3)      ;*****
(2) 015346 000004      TST22: SCOPE
3357
3358 015350 032737 010000 001562      BIT      #BIT12,SR1      ;IS THIS DEVICE SELECTED FOR TEST?
3359                                     ;NOTE: DEFAULT=YES.
3360
3361 015356 001002      BNE      10$
3362 015360 000137 016024      JMP      7$
3363 015364                                     10$:
3364 015364 005037 001126      CLR      $BDDAT
3365
(1)      ;*      MOV      $BDDAT,@LPADSR      ;/ PUT DATA FROM $BDDAT TO DEVICE REG LPADSR
3366                                     ;OK-WHAT WE JUST DID IS ASKED THE
3367                                     ;SLAVE MICRO-CODE TO WRITE THE CONTENTS
3368                                     ;OF THE A/D 'S CSR TO ZEROES.
3369                                     ;IF THE A/D FAILS TO RESPOND TO ITS
3370                                     ;ADDRESS-THE SLAVE WILL SEND US AN
3371                                     ;ERROR CODE THAT CAUSES US TO SET $AERR=1
3372                                     ;IN THE SUPPORT ROUTINES.
3373
3374 015400 005737 037442      TST      $AERR      ;DEVICE PRESENT?
3375 015404 001403      BEQ      11$      ;YES-CONTINUE
3376 015406 104016      ERROR   16      ;A/D FAILED TO RESPOND
3377 015410 000137 016024      JMP      7$      ;TO ADDR.
3378
3379 015414                                     11$:
3380
(1)      ;*      MOV      @LPADSR,$BDDAT      ;/READ DEVICE REG LPADSR,PUT DATA IN $BDDAT.
3381 015424 042737 100200 001126      BIC      #BIT15!BIT7,$BDDAT ;CLEAR OUT FLAGS
3382 015432 005037 001124      CLR      $GDDAT
3383 015436 005737 001126      TST      $BDDAT      ;IS CSR ZERO?
3384 015442 001402      BEQ      1$
3385
3386 015444 104016      ERROR   16      ;FAILED TO ZERO A/D CSR
3387 015446 000572      BR      TST23
3388 015450 012737 025100 001124 1$:      MOV      #BIT13!BIT11!BIT9!BIT6,$GDDAT ;NOW TRY A NEW BIT PATTERN
3389
3390
(1)      ;*      MOV      $GDDAT,@LPADSR      ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPADSR
3391
(1)      ;*      MOV      @LPADSR,$BDDAT      ;/READ DEVICE REG LPADSR,PUT DATA IN $BDDAT.
3392 015476 042737 100200 001126      BIC      #BIT15!BIT7,$BDDAT ;IGNORE FLAGS
3393 015504 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;BIT SET OK?
3394 015512 001402      BEQ      2$
3395
3396 015514 104016      ERROR   16      ;FAILED TO WRITE A/D CSR CORRECTLY
3397 015516 000546      BR      TST23
3398
3399 015520 012737 012404 001124 2$:      MOV      #BIT12!BIT10!BIT8!BIT2,$GDDAT ;NOW TRY A NEW BIT PATTERN.
3400
3401
(1)      ;*      MOV      $GDDAT,@LPADSR      ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPADSR
3402
(1)      ;*      MOV      @LPADSR,$BDDAT      ;/READ DEVICE REG LPADSR,PUT DATA IN $BDDAT.
    
```

```

3403 015546 042737 100200 001126      BIC    #BIT15!BIT7,$BDDAT ;IGNORE FLAGS
3404 015554 023737 001124 001126      CMP    $GDDAT,$BDDAT ;BITS SET OK?
3405 015562 001402                      BEQ    3$
3406
3407 015564 104016                      ERROR  16 ;FAILED TO WRITE A/D CSR CORRECTLY
3408 015566 000522                      BR     TST23 ;:
3409
3410 015570 005037 001124      3$:   CLR    $GDDAT ;NOW CLEAR ALL BITS SET
3411
3412  (1) ;*   MOV    $GDDAT,@LPADSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPADSR
3413  (1) ;*   MOV    @LPADSR,$BDDAT ;/READ DEVICE REG LPADSR,PUT DATA IN $BDDAT.
3413 015614 042737 000200 001126      BIC    #BIT7,$BDDAT ;IGNORE DONE FLAG.
3414 015622 005737 001126      .ST   $BDDAT ;CSR CLEAR?
3415 015626 001402                      BEQ    4$
3416
3417 015630 104016                      ERROR  16 ;A/D CSR FAILED TO CLEAR.
3418 015632 000500                      BR     TST23 ;:
3419
3420 015634      4$:
3421  (1) ;*   MOV    @LPADBR,MYTEMP ;/READ DEVICE REG LPADBR,PUT DATA IN MYTEMP.
3421 015644 012737 000001 001124      MOV    #BIT0,$GDDAT ;NOW WE'LL SET THE A/D START BIT.
3422  (1) ;*   MOV    $GDDAT,@LPADSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPADSR
3423 015662 012737 000200 001124      MOV    #BIT7,$GDDAT ;WHEN WE READ CSR BACK EXPECT
3424 ;START BIT TO CLEAR, DONE TO SET.
3425  (1) ;*   MOV    @LPADSR,$BDDAT ;/READ DEVICE REG LPADSR,PUT DATA IN $BDDAT.
3426
3427 015700 023737 001124 001126      CMP    $GDDAT,$BDDAT ;START OK?
3428 015706 001402                      BEQ    5$
3429
3430 015710 104016                      ERROR  16 ;CSR BAD AFTER A/D START
3431 ;BIT 0 SHOULD CLEAR ONLY A/D DONE SET.
3432 015712 000450                      BR     TST23 ;:
3433
3434 015714 012737 000001 001124 5$:   MOV    #BIT0,$GDDAT ;OK-NOW WE'RE GONNA START ANOTHER
3435 ;A/D CONVERSION WITHOUT READING
3436 ;THE RESULTS OF THE LAST ONE. THIS
3437 ;SHOULD CAUSE THE ERROR FLAG
3438 ;AS WELL AS THE DONE FLAG TO SET.
3439  (1) ;*   MOV    $GDDAT,@LPADSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPADSR
3440
3441  (1) ;*   MOV    @LPADSR,$BDDAT ;/READ DEVICE REG LPADSR,PUT DATA IN $BDDAT.
3442 015742 012737 100200 001124      MOV    #BIT15!BIT7,$GDDAT ;S/B
3443
3444 015750 023737 001124 001126      CMP    $GDDAT,$BDDAT ;IS 15 AND 7 SET?
3445
3446 015756 001401                      BEQ    6$
3447
3448 015760 104016                      ERROR  16 ;ERROR A/D BIT 15 AND 7 SHOULD BE
3449
3450 015762 005037 001124      6$:   CLR    $GDDAT ;MAKE SURE ERROR FLAG CLEARS

```

```
3451 (1) ;* MOV $GDDAT,@LPADSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPADSR
3452 (1) ;* MOV @LPADSR,$BDDAT ;/READ DEVICE REG LPADSR,PUT DATA IN $BDDAT.
3453 016006 042737 000200 001126 BIC #BIT7,$BDDAT ;IGNORE DONE FLAG.
3454 016014 005737 001126 TST $BDDAT ;IS CSR CLEAR?
3455 016020 001401 BEQ 7$
3456 016022 104016 ERROR 16 ;A/D CSR FAILED TO CLEAR.
```

```
3457 3458 016024 7$:
3459 (1) ;* MOV @LPADBR,MYTEMP ;/READ DEVICE REG LPADBR,PUT DATA IN MYTEMP.
```

```
3460
3488
3489 ;*****
; *TEST 23 I/O DEVICE TEST-LPS-CLOCK OPTION
```

```
(3)
(4)
(5)
(5) ;*IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.
(5) ;*WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE
(5) ;*KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS
(5) ;*NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.
(5) ;*THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE
(5) ;*I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE
(5) ;*IT FROM 'SR1' (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN
(5) ;*HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE
(5) ;*I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO 'SR1'
(5) ;*IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE
(5) ;*INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.
```

```
(4)
(4) ;*
(4) ;* IN THIS TEST WE WILL EXERCISE THE LPS-11 CLOCK. IF
(4) ;* SR1 IS UNALTERED, WE DO THIS TEST SINCE THE
(4) ;* LPS-11 CLOCK IS A DEFAULT I/O DEVICE.
(4) ;* PLEASE NOTE: THIS IS ONLY A BRIEF TEST OF THE LPS-11 CLOCK,
(4) ;* TO SEE THAT IT ROUGHLY WORKS. FOR TESTING, YOU
(4) ;* MUST RUN THE LPS-11 CLOCK DIAGNOSTIC. COMPLETE
```

- ```
(4)
(4) ;* TESTS:
(4) ;* 1. MAKE SURE LPS-11 CLOCK RESPONDS TO ADDRESS
(4) ;* (NOTE: IF THE LPS-11 CLOCK ON YOU I/O BUS USES A NON
(4) ;* STANDARD ADDRESS, MODIFY 'LPADSR' WITH THE CORR
(4) ;*
(4) ;* 2. CLOCK CSR BITS 14,9,6,3, AND 1 SET
(4) ;*
(4) ;* 3. CLOCK CSR BITS 13,8 AND 2 SET
(4) ;*
(4) ;* 4. CLOCK CSR ZEROS.
(4) ;*
(4) ;* 5. START CLOCK A MODE 0, RATE 1MHZ,
(4) ;* CHECK 'ENABLE CNTR A' CLEARS,
(4) ;* 'OVERFLOW FLAG' SETS
(4) ;*
(4) ;* 6. CLOCK A CSR ZEROS
```

```
(3)
(2) 016034 000004 ;*****
3490 ;TST23: SCOPE
```

```

3491 016036 032737 020000 001562 BIT #BIT13,SR1 ;IS THIS DEVICE SELECTED FOR TEST?
3492 ;NOTE: DEFAULT=YES.
3493 016044 001002 BNE 10$
3494 016046 000137 016356 JMP 11$
3495 016052 10$:
3496
3497 016052 012737 041110 001124 MOV #BIT14!BIT9!BIT6!BIT3,$GDDAT ;TEST THESE BITS
3498
3499 ;* MOV $GDDAT,@LPCKSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPCKSR
(1)
3500
3501 016070 005737 037442 TST $AERR ;DID CLOCK RESPOND
3502 016074 001403 BEQ 12$;TO ADDR? $AERR=0?
3503 ;IF YES-NEXT TEST.
3504 016076 104016 ERROR 16 ;CLOCK FAILED TO RESPOND TO ADDR.
3505 016100 000137 016356 JMP 11$
3506 016104 12$:
3507
3508 ;* MOV @LPCKSR,$BDDAT ;/READ DEVICE REG LPCKSR,PUT DATA IN $BDDAT.
(1)
3509
3510 016114 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID THESE BITS SET?
3511 016122 001403 BEQ 1$
3512
3513 016124 104016 ERROR 16 ;CLOCK R/W CSR ERROR.
3514 016126 000137 016356 JMP 11$
3515
3516 016132 012737 020404 001124 1$: MOV #BIT13!BIT8!BIT2,$GDDAT ;GET NEW PATTERN
3517 ;* MOV $GDDAT,@LPCKSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPCKSR
(1)
3518 ;* MOV @LPCKSR,$BDDAT ;/READ DEVICE REG LPCKSR,PUT DATA IN $BDDAT.
(1)
3519
3520 016160 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID THE BITS SET?
3521 016166 001403 BEQ 2$
3522
3523 016170 104016 ERROR 16 ;CLOCK R/W CSR ERROR.
3524 016172 000137 016356 JMP 11$
3525
3526 016176 005037 001124 2$: CLR $GDDAT ;WRITE ALL ZEROS.
3527 ;* MOV $GDDAT,@LPCKSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPCKSR
(1)
3528 ;* MOV @LPCKSR,$BDDAT ;/READ DEVICE REG LPCKSR,PUT DATA IN $BDDAT.
(1)
3529 016222 005737 001126 TST $BDDAT ;DID CSR CLEAR?
3530 016226 001403 BEQ 3$;YES-GOOD.
3531
3532 016230 104016 ERROR 16 ;CLOCK FAILED TO ZERO CSR.
3533 016232 000137 016356 JMP 11$
3534
3535 016236 012737 000003 001124 3$: MOV #BIT0!BIT1,$GDDAT ;PUT 'ENABL CNTR A' AND RATE 1MHZ IN CSR
3536 016244 012737 177777 001126 MOV #177777,$BDDAT ;PRELOAD CNTR.
3537 ;* MOV $BDDAT,@LPCKBR ;/ PUT DATA FROM $BDDAT TO DEVICE REG LPCKBR
(1)
3538 ;* MOV $GDDAT,@LPCKSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPCKSR
(1)

```

```

3539
(1) ;* MOV @LPCKSR,$BDDAT ;/READ DEVICE REG LPCKSR,PUT DATA IN $BDDAT.
3540 ;CLOCK SHOULD OVERFLOW, CLEAR
3541 016302 012737 000202 001124 MOV #BIT7:BIT1,$GDDAT ;BIT 0 ('ENABLE CNTR A) SET
3542 ;BIT 7
3543 016310 023737 001124 001126 CMP $GDDAT,$BDDAT ;HAPPEN OK?
3544 016316 001401 BEQ 7$
3545
3546 016320 104016 ERROR 16 ;CLOCK CSR PROBLEM
3547
3548 016322 005037 001124 7$: CLR $GDDAT ;TRY CLEARING CSR
3549
(1) ;* MOV $GDDAT,@LPCKSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPCKSR
3550
(1) ;* MOV @LPCKSR,$BDDAT ;/READ DEVICE REG LPCKSR,PUT DATA IN $BDDAT.
3551 016346 005737 001126 TST $BDDAT ;DID CSR CLEAR?
3552 016352 001401 BEQ TST24 ;:
3553
3554 016354 104016 ERROR 16 ;CSR FAILED TO CLEAR
3555 016356 11$:
3556
3564
(3) ;*****
(5) ;*TEST 24 *TEST THE LPS I/O, IF 'SR1' BIT 15-1(SET)
(5) ;*
(5) ;*IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.
(5) ;*WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE
(5) ;*KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS
(5) ;*NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.
(5) ;*THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE
(5) ;*I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE
(5) ;*IT FROM 'SR1' (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN
(5) ;*HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE
(5) ;*I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO 'SR1'
(5) ;*IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE
(5) ;*INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.
(5) ;*
(4) ;*IN THIS TEST WE WILL CHECK OUT LPS-I/O IF BIT 15 OF SR1=1
(4) ;*(SET). ALSO, DATA LOOP BACK (OUTPUT DATA BACK TO INPUT) WILL BE CHECKED
(4) ;*IF 'SR2' BIT 15 =1 (SET). FOR THIS, YOU MUST HAVE INSTALLED A LOOP-BACK
(4) ;*CABLE.
(3) ;*****
(2) 016356 000004 TST24: SCOPE
3565 016360 032737 100000 001562 BIT #BIT15,SR1
3566 016366 001002 BNE 10$;1-TDRT-
3567 016370 000137 017054 JMP 11$
3568 016374 10$:
3569 016374 005037 001124 CLR $GDDAT ;/SET TO XFERR ZEROS.
3570
(1) ;* MOV $GDDAT,@LPIOSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPIOSR
3571 016410 005737 037442 TST $AERR ;=1 IF DEVICE NOT PRESENT,
3572 016414 001403 BEQ 1$;IF 0, PRESENT, (CON'T).
3573
3574 016416 104016 ERROR 16 ;DR11K #'DRN' DID NOT RESPOND WHEN
3575 ;ADDRESSED. 'SR1' BIT 'DRB' SELECTED
3576 ;THIS OPTION. ADDRESS 'DRCR' 'DRN'
3577 ;CONTAINS ITS ADDRESS

```

```

3578
3579 016420 000137 017054 JMP 11$
3580
3581 016424 1$:
(1)
(1)
3582 016434 005737 001126 ;* MOV @LPIOSR,$BDDAT ;/READ DEVICE REG LPIOSR,PUT DATA IN $BDDAT.
3583 016440 001403 TST $BDDAT ;DID CSR CLEAR?
3584
3585 016442 104016 ERROR 16 ;LPS-I/O CSR FAILED TO CLEAR.
3586 016444 000137 017054 JMP 11$
3587 016450 2$:
3588
(1)
3589 ;* MOV @LPIOOR,$GDDAT ;/READ DEVICE REG LPIOOR,PUT DATA IN $GDDAT.
(1)
3590 ;* MOV @LPIOIR,$GDDAT ;/READ DEVICE REG LPIOIR,PUT DATA IN $GDDAT.
3591 016470 012737 040100 001124 MOV #BIT6!BIT14,$GDDAT ;LOAD WRITEABLE BITS.
3592
(1)
3593 ;* MOV $GDDAT,@LPIOSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPIOSR
(1)
3594 016516 023737 001124 001126 ;* MOV @LPIOSR,$BDDAT ;/READ DEVICE REG LPIOSR,PUT DATA IN $BDDAT.
3595 016524 001402 CMP $GDDAT,$BDDAT ;DID BITS SET?
3596
3597 016526 104016 ERROR 16 ;LPS-I/O CSR BIT(S) FAILED.
3598 016530 000551 BR TST25 ;;
3599
3600 016532 005037 001124 3$: CLR $GDDAT ;TRY CLEARING CSR.
3601
(1)
3602 ;* MOV $GDDAT,@LPIOSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPIOSR
(1)
3603 016556 005737 001126 ;* MOV @LPIOSR,$BDDAT ;/READ DEVICE REG LPIOSR,PUT DATA IN $BDDAT.
3604 016562 001402 TST $BDDAT ;DID CSR CLEAR?
3605
3606 016564 104016 ERROR 16 ;LPS-I/O CSR FAILED TO CLEAR.
3607 016566 000532 BR TST25 ;;
3608
3609 016570 032737 100000 001564 4$: BIT #BIT15,SR2 ;DOES THIS I/O HAVE A LOOP BACK
3610
3611 016576 001526 BEQ TST25 ;CABLE CONNECTED TO IT?
3612
3613
3614 016600 012737 052525 001124 MOV #052525,$GDDAT ;SET FIRST PATTERN
3615
(1)
3616 ;* MOV $GDDAT,@LPIOOR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPIOOR
(1)
3617 016626 012737 000200 001124 ;* MOV @LPIOSR,$BDDAT ;/READ DEVICE REG LPIOSR,PUT DATA IN $BDDAT.
3618 016634 023737 001126 001124 MOV #BIT7,$GDDAT ;EXPECT FLAGS TO SET.
3619 016642 001402 CMP $BDDAT,$GDDAT ;DID THEY?
3620
3621 016644 104016 ERROR 16 ;LPS-I/O CSR FLAG(S) FAILED TO SET
3622
3623

```

;WHEN DATA XFERRD. IS THE OUTPUT  
;REALLY CALLED BACK TO THE INPUT?

```

3624 016646 000502 BR TST25 ;;
3625
3626 016650 012737 052525 001124 5$: MOV #052525,$GDDAT ;PATTERN THAT SHOULD HAVE XFERRERD.
3627
(1) * MOV @LPIOIR,$BDDAT ;/READ DEVICE REG LPIOIR,PUT DATA IN $BDDAT.
3628 016666 023737 001126 001124 ;* CMP $BDDAT,$GDDAT ;DATA SENT=DATA RECEIVED?
3629 016674 001402 BEQ 6$
3630
3631 016676 104016 ERROR 16 ;LPS-11 I/O DATA XFERR ERROR.
3632 016700 000465 BR TST25 ;;
3633
3634 016702 6$:
(1)
(1) ;* MOV @LPIOSR,$BDDAT ;/READ DEVICE REG LPIOSR,PUT DATA IN $BDDAT.
3635 016712 005737 001126 ;* TST $BDDAT ;DID "OUTPUT FLAG" SET.
3636 016716 100402 BMI 7$;SHOULD HAVE.
3637
3638 016720 104016 ERROR 16 ;LPS-11 I/O "OUTPUT FLAG" FAILED TO SET.
3639 016722 000454 BR TST25 ;;
3640
3641 016724 7$:
(1)
(1) ;* MOV $GDDAT,@LPIOOR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPIOOR
3642 016734 012737 125252 001124 ;* MOV #125252,$GDDAT ;NEW PATTERN
3643
(1) ;* MOV $GDDAT,@LPIOOR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPIOOR
3644
(1) ;* MOV @LPIOIR,$BDDAT ;/READ DEVICE REG LPIOIR,PUT DATA IN $BDDAT.
3645 016762 023737 001126 001124 ;* CMP $BDDAT,$GDDAT ;PATTERN XFERR OK?
3646 016770 001402 BEQ 8$
3647
3648 016772 104016 ERROR 16 ;LPS I/O DATA XFERR ERROR
3649 016774 000427 BR TST25 ;;
3650
3651 016776 8$:
(1)
(1) ;* MOV $GDDAT,@LPIOIR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPIOIR
3652 017006 005037 001124 ;* CLR $GDDAT ;NOW XFERR ZERO PATTERN.
3653
(1) ;* MOV $GDDAT,@LPIOOR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPIOOR
3654
(1) ;* MOV @LPIOIR,$BDDAT ;/READ DEVICE REG LPIOIR,PUT DATA IN $BDDAT.
3655 017032 005737 001126 ;* TST $BDDAT ;DID ZERO PATTERN GET XFERRERD?
3656 017036 001402 BEQ 9$
3657
3658 017040 104016 ERROR 16 ;LPS I/O DATA XFERR ERROR
3659 017042 000404 BR TST25 ;;
3660
3661 017044 9$:
(1)
(1) ;* MOV $GDDAT,@LPIOSR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPIOSR
3662 017054 ;*
3663
3670
3671
(3)
;*****
;*TEST 25 *TEST THE LPS D/A OPTION, IF "SR1" BIT 14=1 (SET)

```

(5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (4)  
 (4)  
 (4)  
 (3)  
 (2)  
 3672  
 3673  
 3674  
 3675  
 3676  
 3677  
 3678  
 (1)  
 3679  
 3680  
 3681  
 3682  
 3683  
 3684  
 3685  
 3686  
 3687  
 3688  
 (1)  
 3689  
 3690  
 3691  
 3692  
 3693  
 3694  
 3695  
 3696  
 (1)  
 3697  
 (1)  
 3698  
 3699  
 3700  
 3701  
 3702  
 3703  
 3704  
 3705

017054 000004  
 017056 032737 040000 001562  
 017064 001002  
 017066 000137 017632  
 017072  
 017072 005037 001124  
 017106 005737 037442  
 017112 001403  
 017114 104016  
 017116 000137 017632  
 017122  
 017132 042737 000200 001126  
 017140 001403  
 017142 104014  
 017144 000137 017632  
 017150 012737 005124 001124  
 017176 042737 000200 001126  
 017204 023737 001124 001126  
 017212 001403  
 017214 104016  
 017216 000137 017632

```

:*
:* IN THIS TEST WE ARE CHECKING OUT AN I/O DEVICE ON THE LPA-11'S I/O BUS.
:* WE HAVE A CERTAIN AMOUNT OF CONFIDENCE THAT THE M8254 (IPBM), THE
:* KMC-11 AND THE DMC-11 WORK. IN THIS TEST, THE ARBITRATION CIRCUITRY IS
:* NOT CHECK, NOR THE DEVICE'S INTERRUPT CAPABILITIES.
:* THE DIAGNOSTIC IS SET UP FOR CERTAIN DEFAULT DEVICES ON THE
:* I/O BUS. IF THE DEFAULT DEVICE IS NOT ON THE I/O BUS, DELETE
:* IT FROM "SR1" (NOTE: AT THAT LOCATION, WE LIST CODES, AND EXPLAIN
:* HOW TO ADD AND DELETE OPTIONS). IF YOU HAVE A DEVICE ON THE
:* I/O BUS THAT IS NOT A DEFAULT DEVICE, AD IT TO "SR1"
:* IF A FAILURE OCCURS IN THIS TEST, YOU SHOULD RUN THE
:* INDIVIDUAL DIAGNOSTIC THAT EXERCISES THIS OPTION.
:*
:* IN THIS TEST WE WILL CHECK OUT THE LPS D/A OPTION.
:* IF BIT 14 OF "SR1" IS SET.
:*
:*****
TST25: SCOPE
 BIT #BIT14,SR1 ;IS LPS D/A SELECTED?
 BNE 20$;IF =0, NO, SKIP THIS TEST.
 JMP 19$
20$: CLR $GDDAT
 ;* MOV $GDDAT,@LPDASR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDASR
 TST $AERR ;DID WE SUCCESSFULLY WRITE THE LPS D/A?
 BEQ 1$;YES IF $AERR=0
 ERROR 16 ;FAILED TO ADDRESS LPS D/A. 'SR1' BIT 14=1
 ;SELECTED THIS TEST. LPS D/A ADDRESS IS
 ;IN LOC 'LPDASR'
 JMP 19$
1$: ;* MOV @LPDASR,$BDDAT ;/READ DEVICE REG LPDASR,PUT DATA IN $BDDAT.
 BIC #BIT7,$BDDAT ;CLEAR FLAG
 BEQ 2$;DID CSR CLEAR?
 ERROR 14 ;CLEAR CSR FAILED (LPS-D/A)
 JMP 19$
2$: MOV #5124,$GDDAT ;TRY WRITING THESE BITS
 ;* MOV $GDDAT,@LPDASR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDASR
 ;* MOV @LPDASR,$BDDAT ;/READ DEVICE REG LPDASR,PUT DATA IN $BDDAT.
 BIC #BIT7,$BDDAT ;READ THEM BACK.
 CMP $GDDAT,$BDDAT ;GET RID OF FLAG
 BEQ 3$;DID PATTERN
 ;
 ERROR 16 ;LPS D/A PATTERN 5124 FAILED TO W/R PROPERLY.
 JMP 19$

```



```

3706 017222 012737 002012 001124 3$: MOV #2012,$GDDAT ;TRY NEW PATTERN.
3707 (1) ;* MOV $GDDAT,@LPDASR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDASR
3708 (1) ;* MOV @LPDASR,$BDDAT ;/READ DEVICE REG LPDASR,PUT DATA IN $BDDAT.
3709 017250 042737 000200 001126 BIC #BIT7,$BDDAT ;GET RID OF FLAG.
3710 017256 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID PATTERN XFERR OK?
3711 017264 001403 BEQ 4$;YES-NEXT TEST.
3712 (1)
3713 017266 104016 ERROR 16 ;LPS D/A PATTERN 2012 FAILED TO W/R PROPERLY.
3714 017270 000137 017632 JMP 19$
3715 (1)
3716 017274 005037 001124 4$: CLR $GDDAT ;TRY TO CLEAR CSR.
3717 (1) ;* MOV $GDDAT,@LPDASR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDASR
3718 (1) ;* MOV @LPDASR,$BDDAT ;/READ DEVICE REG LPDASR,PUT DATA IN $BDDAT.
3719 017320 042737 000200 001126 BIC #BIT7,$BDDAT ;CLEAR FLAG BIT.
3720 017326 001403 BEQ 5$
3721 (1)
3722 017330 104016 ERROR 16 ;LPS D/A CSR FAILED TO CLEAR.
3723 017332 000137 017632 JMP 19$
3724 (1)
3725 017336 012737 005252 001124 5$: MOV #5252,$GDDAT
3726 (1) ;* MOV $GDDAT,@LPDAXR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDAXR
3727 (1) ;* MOV @LPDAXR,$BDDAT ;/READ DEVICE REG LPDAXR,PUT DATA IN $BDDAT.
3728 017364 023737 001124 001126 CMP $GDDAT,$BDDAT
3729 017372 001403 BEQ 6$;OK?
3730 (1)
3731 017374 104016 ERROR 16 ;D/A DACX FAILED TO SET PATTERN 5252.
3732 017376 000137 017632 JMP 19$
3733 (1)
3734 017402 6$:
3735 (1) ;* MOV $GDDAT,@LPDAYR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDAYR
3736 (1) ;* MOV @LPDAYR,$BDDAT ;/READ DEVICE REG LPDAYR,PUT DATA IN $BDDAT.
3737 017422 023737 001126 001124 CMP $BDDAT,$GDDAT ;OK
3738 017430 001403 BEQ 7$;YES-
3739 017432 104016 ERROR 16 ;DACX FAILED PATTERN 5252
3740 017434 000137 017632 JMP 19$
3741 (1)
3742 017440 7$:
3743 017440 012737 002525 001124 MOV #2525,$GDDAT
3744 (1) ;* MOV $GDDAT,@LPDAXR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDAXR
3745 (1) ;* MOV @LPDAXR,$BDDAT ;/READ DEVICE REG LPDAXR,PUT DATA IN $BDDAT.
3746 017466 023737 001124 001126 CMP $GDDAT,$BDDAT
3747 017474 001402 BEQ 10$;OK?
3748 (1)
3749 017476 104016 ERROR 16 ;D/A DACY FAILED TO SET PATTEN 2525
3750 017500 000454 BR TST26 ;:

```

```

3751
3752 017502 10$:
(1)
(1) :* MOV $GDDAT,@LPDAYR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDAYR
3753 :* MOV @LPDAYR,$BDDAT ;/READ DEVICE REG LPDAYR,PUT DATA IN $BDDAT.
(1) CMP $BDDAT,$GDDAT ;OK?
3754 017522 023737 001126 001124 BEQ 11$;YES-
3755 017530 001402
3756
3757 017532 104016 ERROR 16 ;D/A DACY FAILED PATTERN 2525
3758 017534 000436 BR TST26 ;;
3759
3760 017536 11$:
3761
3762 017536 012737 000000 001124 13$: MOV #0,$GDDAT
3763 :* MOV $GDDAT,@LPDAXR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDAXR
(1) :* MOV @LPDAXR,$BDDAT ;/READ DEVICE REG LPDAXR,PUT DATA IN $BDDAT.
3764 CMP $GDDAT,$BDDAT
(1) BEQ 14$;OK?
3765 017564 023737 001124 001126
3766 017572 001402
3767
3768 017574 104016 ERROR 16 ;D/A DACX FAILED TO SET PATTERN 0
3769 017576 000415 BR TST26 ;;
3770
3771 017600 14$:
(1)
(1) :* MOV $GDDAT,@LPDAYR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDAYR
3772 :* MOV @LPDAYR,$BDDAT ;/READ DEVICE REG LPDAYR,PUT DATA IN $BDDAT.
(1) CMP $BDDAT,$GDDAT ;OK?
3773 017620 023737 001126 001124 BEQ TST26 ;;
3774 017626 001401
3775
3776 017630 104016 ERROR 16 ;D/A DACY FAILED PATTERN 0
3777 017632 19$:
3778
3779 017632 15$:

```

```

3781
3789
3790 ;*****
(3) ;*TEST 26 *TEST THAT THE LPS D/A CAN DISPLAY A SQUARE
(4) ;*IN THIS TEST WE'LL DISPLAY A SQUARE ON A
(4) ;*DISPLAY SCOPE VIA THE LPS-D/A
(4) ;*IF YOU HAVE AN LPS-D/A AND SCOPE YOU MUST SELECT THIS
(4) ;*TEST BY SETTING THE APPROPRIATE BITS IN 'SR1' (INDICATING YOU HAVE
(4) ;*AN LPS D/A) AND 'SR2' (INDICATING YOU HAVE A SCOPE).
(3) ;*****
(2) 017632 000004 TST26: SCOPE
3791
3792 ;--DAAT-
3793 017634 032737 040000 001562 BIT #BIT14,SR1 ;/LPS D/A SELECTED?
3794 017642 001475 BEQ TST27 ;;
3795 017644 052737 040000 001564 BIT #BIT14,SR2 ;/SCOPE DISPLAY?
3796 017652 001471 BEQ TST27 ;;
3797
3798 017654 005037 001124 CLR $GDDAT
3799 017660 005037 001126 CLR $BDDAT
3800 017664 012737 000001 001754 MOV #1,MYTEMP
3801 017672 1$:
(1) ;*
(1) ;* MOV $GDDAT,@LPDAXR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDAXR
3802 ;*
(1) ;* MOV MYTEMP,@LPDASR ;/ PUT DATA FROM MYTEMP TO DEVICE REG LPDASR
3803 017712 005237 001124 INC $GDDAT
3804 017716 032737 010000 001124 BIT #BIT12,$GDDAT
3805 017724 001762 BEQ 1$
3806 017726 2$:
(1) ;*
(1) ;* MOV $BDDAT,@LPDAYR ;/ PUT DATA FROM $BDDAT TO DEVICE REG LPDAYR
3807 ;*
(1) ;* MOV MYTEMP,@LPDASR ;/ PUT DATA FROM MYTEMP TO DEVICE REG LPDASR
3808 017746 005237 001126 INC $BDDAT
3809 017752 032737 010000 001126 BIT #BIT12,$BDDAT
3810 017760 001762 BEQ 2$
3811 017762 3$:
(1) ;*
(1) ;* MOV $GDDAT,@LPDAXR ;/ PUT DATA FROM $GDDAT TO DEVICE REG LPDAXR
3812 ;*
(1) ;* MOV MYTEMP,@LPDASR ;/ PUT DATA FROM MYTEMP TO DEVICE REG LPDASR
3813 020002 005337 001124 DEC $GDDAT
3814 020006 001365 BNE 3$
3815 020010 4$:
(1) ;*
(1) ;* MOV $BDDAT,@LPDAYR ;/ PUT DATA FROM $BDDAT TO DEVICE REG LPDAYR
3816 ;*
(1) ;* MOV MYTEMP,@LPDASR ;/ PUT DATA FROM MYTEMP TO DEVICE REG LPDASR
3817 020030 005337 001126 DEC $BDDAT
3818 020034 001365 BNE 4$
3819

```

3821  
 3869  
 3870  
 (5)  
 (4)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (4)  
 (3)  
 (2)  
 (1)  
 (1)  
 (1)  
 (3)  
 (1)  
 (3)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (2)  
 (2)  
 (2)  
 (2)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (2)  
 (2)  
 (2)  
 (2)  
 (1)  
 (1)  
 (1)  
 (1)  
 (2)  
 (2)  
 (2)  
 (2)  
 (1)  
 (1)  
 (1)  
 (1)  
 (2)  
 (2)  
 (2)  
 (1)  
 (1)  
 3871

```

*TEST 27 *TEST THAT THE AA11K # CAN DISPLAY A SQUARE, SELECTED BIT 3 IN SR1,SR2
*IN THIS TEST WE'LL DISPLAY A SQUARE ON THE
*DISPLAY SCOPE VIA THE AA-11K.
*IF YOU HAVE AN AA11-K AND SCOPE YOU MUST SELECT THIS
*TEST SETTING THE APPROPRIATE BITS IN 'SR1' (INDICATING YOU HAVE
*AN AA-11K) AND 'SR2' (INDICATING YOU HAVE A SCOPE).

TST27: SCOPE
MOV #1,$TIMES ;:DO 1 ITERATION
;:
;:-DAAT-
;/AA-11K SELECTED?
;:
;/SCOPE DISPLAY?
;:
CLR $GDDAT
CLR $BDDAT
MOV #1,MYTEMP
1$:
;* MOV $GDDAT,@DAC0 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC0
;* MOV MYTEMP,@AACSR ;/ PUT DATA FROM MYTEMP TO DEVICE REG AACSR
INC $GDDAT
BIT #BIT12,$GDDAT
BEQ 1$
2$:
;* MOV $BDDAT,@DAC1 ;/ PUT DATA FROM $BDDAT TO DEVICE REG DAC1
;* MOV MYTEMP,@AACSR ;/ PUT DATA FROM MYTEMP TO DEVICE REG AACSR
INC $BDDAT
BIT #BIT12,$BDDAT
BEQ 2$
3$:
;* MOV $GDDAT,@DAC0 ;/ PUT DATA FROM $GDDAT TO DEVICE REG DAC0
;* MOV MYTEMP,@AACSR ;/ PUT DATA FROM MYTEMP TO DEVICE REG AACSR
DEC $GDDAT
BNE 3$
4$:
;* MOV $BDDAT,@DAC1 ;/ PUT DATA FROM $BDDAT TO DEVICE REG DAC1
;* MOV MYTEMP,@AACSR ;/ PUT DATA FROM MYTEMP TO DEVICE REG AACSR
DEC $BDDAT
BNE 4$
;:
DAAT 6,2

```

```

3873
3879 ;*****
(3) ;*TEST 30 *TEST THAT USER MICRO-CODE CAN BE STARTED
(4) ;*IN THIS TEST, WE ARE GOING TO MAKE SURE
(4) ;*THAT RUN SETS, READY IN SETS, READY OUT CLEARS.
(4) ;*CALL INITIAL CONDITIONS
(3) ;*****
(2) 020250 000004 TST30: SCOPE
3880
3881 020252 004737 040414 JSR PC,$RESET ;ISSUE LPA-11 RESET.
3882 020256 032737 000001 001202 BIT #1,$PASS ;ON ALTERNATE PASSES,WE ALTERNATE MICROCODE.
3883 020264 001013 BNE 10$;
3884 020266 032777 002000 160644 BIT #BIT10,@SWR ;RUN ONLY DEDICATED
3885 020274 001007 BNE 10$;CODE?
3886 ;YES DO THAT LOAD.
3887 020276 004537 037444 13$: JSR R5,$LOAD
3888 020302 044030 MAST
3889 020304 012737 000115 001772 MOV #'M,$VERSN
3890 020312 000412 BR 11$
3891 020314 004537 037444 10$: JSR R5,$LOAD
3892 020320 050034 DMAST
3893 020322 012737 000104 001772 MOV #'D,$VERSN
3894 020330 032777 001000 160602 BIT #BIT9,@SWR ;RUN ONLY MULITE-SER CODE?
3895 020336 001357 BNE 13$;YES-GO BACK AND RELOAD
3896 020340 113737 001512 001773 11$: MOVB VERSN,$VERSN+1
3897 020346 052777 040000 161112 BIS #BIT14,@KMADO ;SET INIT.
3898 020354 012701 000004 MOV #4,R1
3899 020360 004737 040452 12$: JSR PC,$DELAY
3900 020364 005301 DEC R1
3901 020366 001374 BNE 12$
3902 020370 012777 104000 161070 MOV #BIT15!BIT11,@KMADO ;BIT 15 OF THE KMC CSR SHOULD BE
3903 ;SET AT THIS TIME.
3904 020376 012701 000062 MOV #50,R1
3905 020402 032777 000200 161056 1$: BIT #BIT7,@LPCI ;BIT 7 OF CONTROL IN REG. (READY IN)
3906 020410 001014 BNE 2$;SHOULD BE SET AT INIT.
3907 020412 004737 040452 JSR PC,$DELAY ;DELAY FOR WHILE.
3908 020416 005301 DEC R1
3909 020420 001370 BNE 1$
3910 020422 012737 000200 001124 MOV #BIT7,$GDDAT
3911 020430 017737 161032 001126 MOV @LPCI,$BDDAT
3912 020436 104017 ERROR 17 ;LPA-11 ERROR - READY IN NOT SET AT INIT.
3913 020440 000424 BR TST31 ;
3914 020442 032777 000200 161022 2$: BIT #BIT7,@LPCO ;CHECK READY OUT, IT SHOULD
3915 020450 001407 BEQ 3$;BE CLEAR AT INIT.
3916 020452 017737 161014 001126 MOV @LPCO,$BDDAT
3917 020460 005037 001124 CLR $GDDAT
3918 020464 104017 ERROR 17 ;LPW-11 ERROR - READY OUT SET AT INIT.
3919 020466 000411 BR TST31 ;
3920
3921 020470 105777 161000 3$: TSTB @LPSO ;CHECK THE STATUS OUT REGISTER
3922 ;IT SHOULD BE CLEAR AT INIT
3923 020474 001406 BEQ TST31 ;;<IF CLEAR, NEXT TEST>
3924 020476 117737 160772 001126 MOVB @LPSO,$BDDAT
3925 020504 005077 160414 CLR @SGDDAT
3926 020510 104017 ERROR 17 ;LPA-11 ERROR STATUS OUT REGISTER NOT CLEAR AT INIT.
3927

```

```

3936 ;:*****
(3) ;*TEST 31 *TEST FOR ERROR CODE 306
(4) ;*IN THIS TEST, WE ARE GOING TO ISSUE A CLOCK START
(4) ;*COMMAND. THE FIRST THING THAT SHOULD TAKE PLACE
(4) ;*IS 'READY OUT' SHOULD SET. NEXT WE SHOULD GET
(4) ;*ERROR #306 (NO INIT COMMAND) SINCE WE HADN'T
(4) ;*INITIALIZED FIRST. AFTER, WE'LL CLEAR 'READY OUT'
(4) ;*AND MAKE SURE THE STATUS OUT REG. CLEARS.
(3) ;:*****
(2) 020512 000004 TST31: SCOPE
(1) 020514 012737 000001 001160 MOV #1,$TIMES ;;DO 1 ITERATION
3937
3938 020522 012737 000322 044022 MOV #BIT1!BIT4!BIT6!BIT7,KWT ;SET START CLOCK IN CLOCK TABLE
3939 ;(REQUEST DESCRIPTOR ARRAY)
3940
3941 020530 012700 044022 MOV #KWT,R0 ;GET ADDR. OF ARRAY
3942 020534 012760 000000 000002 MOV #0,2(0) ;SET CLOCK STATUS.
3943 020542 005060 000004 CLR 4(0) ;SET CLOCK PRESET
3944 020546 010077 160724 MOV R0,@LPADL
3945 020552 152777 000001 160706 BISB #1,@LPCI ;SET GO.
3946 020560 004737 040452 JSR PC,SDELAY
3947 020564 105777 160702 TSTB @LPCO ;DID 'READY OUT' SET?
3948 020570 100410 BMI
3949 020572 017737 160674 001126 MOV @LPCO,$BDDAT
3950 020600 012737 000200 0C1124 MOV #200,$GDDAT
3951 020606 104017 ERROR 17 ;'READY OUT' FAILED TO SET (LPA-11 ERROR)
3952 020610 000432 BR TST32 ;;
3953
3954 020612 005037 001126 1$: CLR $BDDAT
3955 020616 012737 000306 001124 MOV #306,$GDDAT ;/EXPECT 306.
(1) 020624 117737 160644 001126 MOVB @LPSO,$BDDAT ;/SEE WHAT WE GET.
3956 020632 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID LPA-11 RETURN ERROR CODE
3957 ;#306 - 'NO INITIALIZE COMMAND'?
3958 020640 001402 BEQ 2$
3959
3960 020642 104017 ERROR 17 ;LPA-11 ERROR - ERROR CODE #306 NOT RETURNED
3961 020644 000414 BR TST32 ;;
3962
3963 020646 005077 160620 2$: CLR @LPCO ;CLEAR 'READY OUT'
3964 ;ERROR OR STATUS OUT REG. SHOULD CLEAR.
3965 020652 105777 160616 TSTB @LPSO ;DID IT CLEAR?
3966 020656 001407 BEQ TST32 ;;<YES - NEXT TEST>
3967
3968 020660 012737 000000 001124 MOV #0,$GDDAT ;/EXPECT 0.
(1) 020666 117737 160602 001126 MOVB @LPSO,$BDDAT ;/SEE WHAT WE GET.
3969 020674 104017 ERROR 17 ;STATUS OUT REG. NOT CLEARFD WHEN 'READY OUT' CLEARED (L

```

3971  
 3979  
 (3)  
 (4)  
 (4)  
 (4)  
 (4)  
 (4)  
 (3)  
 (2)  
 3980  
 3981  
 3982  
 3983  
 3984  
 3985  
 3986  
 3987  
 3988  
 3989  
 3990  
 3991  
 (1)  
 3992  
 3993  
 3994  
 3995  
 3996  
 (1)  
 3997  
 3998  
 3999  
 4000  
 4001  
 4002

020676 000004  
 020700 012700 054040  
 020704 005200  
 020706 010077 160564  
 020712 152777 000001 160546  
 020720 005000  
 020722  
 020722 105777 160544  
 020726 100412  
 020730 105200  
 020732 100373  
 020734 012737 000200 001124  
 020742 017737 160524 001126  
 020750 104017  
 020752 000414  
 020754  
 020754 012737 000314 001124  
 020762 117737 160506 001126  
 020770 122737 000314 001126  
 020776 001402  
 021000 104017  
 021002 000400

```

;*TEST 32 *TEST THAT WE CAN GENERATE ERROR CODE 314
;
; *IN THIS TEST WE WILL TRY TO GENERATE ERROR CODE 314
; *(ODD ADDRESS SPECIFIED) FOR REQUEST DESCRIPTOR ARRAY <RDA>
;

TST32: SCOPE
 MOV #DEVLST,RO ;GET AN ADDR.
 INC RO ;MAKE IT ODD.
 MOV RO,@LPADL ;STORE ADDR. AS ADDR OF A
 ;REQUEST DESCRIPTOR ARRAY.
 BISB #1,@LPCI ;SET GO
 CLR RO ;TIME OUT COUNTER
1$:
 TSTB @LPCO ;READY OUT SET?
 BMI 2$;YES-EXIT LOOP
 INCB RO ;NO-LOOP OVERFLOW OCCUR?
 BPL 1$;NO-LOOP.
 MOV #200,$GDDAT ;/EXPECT 200.
 MOV @LPCO,$BDDAT ;/SEE WHAT WE GET.
 ERROR 17 ;READY OUT NOT SET ON INIT CMND. (LPC-ERROR)
 BR TST33 ;;

2$:
 MOV #314,$GDDAT ;/EXPECT 314.
 MOVB @LPSO,$BDDAT ;/SEE WHAT WE GET.
 CMPB #314,$BDDAT ;DID CORRECT ERROR CODE GET RETURNED?
 BEQ TST33 ;;
 ERROR 17 ;ERROR CODE 314 NOT RETURNED FOR ODD ADDRESS FOR RDA
 BR TST33 ;;

```

```
4004
4011 ;*****
(3) ;*TEST 33 *TEST THAT THE DEVICE LIST CAN BE VERIFIED
(4) ;
(4) ; INIT TEST IN THIS TEST WE MAKE SURE THE KMC MICRO-
(4) ; CODE CAN FIND ALL ADDRESSES OF DEVICES TO BE TESTED.
(4) ;
(3) ;*****
(2) 021004 000004 TST33: SCOPE
4012
4013 021006 012700 000012 MOV #10.,R0
4014 021012 012701 054040 MOV #DEVLST,R1 ;CLEAR DEVICE LIST AREA
4015 021016 005021 CLR (1)+
4016 021020 005003 CLR R3
4017 021022 012721 000001 1$: MOV #1,(1)+ ;PUT A 1 INTO ALL POSTIONS OF
4018 ; DEVICE LIST SO THAT LPA WON'T RETURN
4019 ; ARR ERROR UNLESS BAD ADDR.
4020 ; OTHERWISE IT WOULD RETURN ADDR. ERROR
4021 ; CODE ON ADDR. ZERO.
4022 021026 005300 DEC R0
4023 021030 001374 BNE 1$
4024 021032 012701 054042 MOV #DEVLST+2,R1
4025 021036 032737 000002 001562 BIT #BIT1,SR1 ;KW11K CLOCK??
4026 021044 001403 BEQ 10$;NO
4027 021046 013721 001570 MOV KW11K,(1)+ ;YES FIX ADDR.
4028 021052 000425 BR 20$
4029 021054 032737 002000 001562 10$: BIT #BIT10,SR1 ;AR11 CLOCK??
4030 021062 001405 BEQ 11$;NO
4031 021064 013711 001610 MOV AR11K,(1) ;YES FIX AR11K CLOCK ADDR.
4032 021070 062721 000004 ADD #4,(1)+
4033 021074 000414 BR 20$
4034 021076 032737 020000 001562 11$: BIT #BIT13,SR1 ;LPS CLOCK?
4035 021104 001405 BEQ 12$;NO-ERROR!!!
4036 021106 013711 001612 MOV LPS11,(1) ;YES FIX LPS11 CLOCK ADDR.
4037 021112 062721 000004 ADD #4,(1)+
4038 021116 000403 BR 20$
4039 021120 12$:
4040 021120 104017 ERROR 17 ;NO CLOCK SELECTED FOR TEST...
4041 ;CAN NOT DO ANY MORE TESTS WITHOUT
4042 ;CLOCK PRESENT!
4043 ;YOU MUST SELECT A CLOCK.
4044 021122 000137 030634 JMP ETEST
4045
4046 021126 012721 000001 20$: MOV #1,(1)+
4047 021132 032737 000001 001562 BIT #BIT0,SR1 ;AD11K FOR TEST??
4048 021140 001403 BEQ 21$
4049 021142 013721 001566 MOV AD11K,(1)+ ;YES FIX AD11K ADDR.
4050 021146 000420 BR 30$
4051 021150 032737 002000 001562 21$: BIT #BIT10,SR1 ;AR11 A/D?
4052 021156 001403 BEQ 22$
4053 021160 013721 001610 MOV AR11K,(1)+
4054 021164 000411 BR 30$
4055 021166 032737 010000 001562 22$: BIT #BIT12,SR1 ;LPS A/D?
4056 021174 001403 BEQ 23$
4057 021176 013721 001612 MOV LPS11,(1)+
4058 021202 000402 BR 30$
```



|      |        |        |        |        |       |     |             |                      |
|------|--------|--------|--------|--------|-------|-----|-------------|----------------------|
| 4059 | 021204 | 012721 | 000001 |        | 23\$: | MOV | #1,(1)+     |                      |
| 4060 |        |        |        |        |       |     |             |                      |
| 4061 | 021210 | 032737 | 000020 | 001562 | 30\$: | BIT | #BIT4,SR1   | ;2ND AD11K?          |
| 4062 | 021216 | 001403 |        |        |       | BEQ | 31\$        |                      |
| 4063 | 021220 | 013721 | 001576 |        |       | MOV | AD11K2,(1)+ |                      |
| 4064 | 021224 | 000402 |        |        |       | BR  | 32\$        |                      |
| 4065 | 021226 | 012721 | 000001 |        | 31\$: | MOV | #1,(1)+     |                      |
| 4066 |        |        |        |        |       |     |             |                      |
| 4067 | 021232 | 032737 | 000010 | 001562 | 32\$: | BIT | #BIT3,SR1   | ;AA11K?              |
| 4068 | 021240 | 001403 |        |        |       | BEQ | 33\$        |                      |
| 4069 | 021242 | 013721 | 001574 |        |       | MOV | AA11K,(1)+  |                      |
| 4070 | 021246 | 000424 |        |        |       | BR  | 40\$        |                      |
| 4071 | 021250 | 032737 | 002000 | 001562 | 33\$: | BIT | #BIT10,SR1  | ;AR11?               |
| 4072 | 021256 | 001405 |        |        |       | BEQ | 34\$        |                      |
| 4073 | 021260 | 013711 | 001610 |        |       | MOV | AR11K,(1)   |                      |
| 4074 | 021264 | 062721 | 000016 |        |       | ADD | #16,(1)+    |                      |
| 4075 | 021270 | 000413 |        |        |       | BR  | 40\$        |                      |
| 4076 | 021272 | 032737 | 040000 | 001562 | 34\$: | BIT | #BIT14,SR1  | ;LPS D/A?            |
| 4077 | 021300 | 001405 |        |        |       | BEQ | 35\$        |                      |
| 4078 | 021302 | 013711 | 001612 |        |       | MOV | LPS11,(1)   |                      |
| 4079 | 021306 | 062721 | 000016 |        |       | ADD | #16,(1)+    |                      |
| 4080 | 021312 | 000402 |        |        |       | BR  | 40\$        |                      |
| 4081 | 021314 | 012721 | 000001 |        | 35\$: | MOV | #1,(1)+     |                      |
| 4082 |        |        |        |        |       |     |             |                      |
| 4083 | 021320 | 032737 | 000004 | 001562 | 40\$: | BIT | #BIT2,SR1   | ;DR11K?              |
| 4084 | 021326 | 001403 |        |        |       | BEQ | 41\$        |                      |
| 4085 | 021330 | 013721 | 001572 |        |       | MOV | DR11K1,(1)+ |                      |
| 4086 | 021334 | 000413 |        |        |       | BR  | 50\$        |                      |
| 4087 | 021336 | 032737 | 100000 | 001562 | 41\$: | BIT | #BIT15,SR1  | ;LPS I/O?            |
| 4088 | 021344 | 001405 |        |        |       | BEQ | 42\$        |                      |
| 4089 | 021346 | 013711 | 001612 |        |       | MOV | LPS11,(1)   |                      |
| 4090 | 021352 | 062721 | 000010 |        |       | ADD | #10,(1)+    | ;FIX LPS11 I/O ADDR. |
| 4091 | 021356 | 000402 |        |        |       | BR  | 50\$        |                      |
| 4092 | 021360 | 012721 | 000001 |        | 42\$: | MOV | #1,(1)+     |                      |
| 4093 |        |        |        |        |       |     |             |                      |
| 4094 | 021364 | 032737 | 000040 | 001562 | 50\$: | BIT | #BIT5,SR1   | ;DR11K#2?            |
| 4095 | 021372 | 001403 |        |        |       | BEQ | 51\$        |                      |
| 4096 | 021374 | 013721 | 001600 |        |       | MOV | DR11K2,(1)+ |                      |
| 4097 | 021400 | 000402 |        |        |       | BR  | 52\$        |                      |
| 4098 | 021402 | 012721 | 000001 |        | 51\$: | MOV | #1,(1)+     |                      |
| 4099 | 021406 | 032737 | 000200 | 001562 | 52\$: | BIT | #BIT7,SR1   | ;DR11K #3 ?          |
| 4100 | 021414 | 001403 |        |        |       | BEQ | 53\$        |                      |
| 4101 | 021416 | 013721 | 001602 |        |       | MOV | DR11K3,(1)+ |                      |
| 4102 | 021422 | 000402 |        |        |       | BR  | 54\$        |                      |
| 4103 | 021424 | 012721 | 000001 |        | 53\$: | MOV | #1,(1)+     |                      |
| 4104 | 021430 | 032737 | 000400 | 001562 | 54\$: | BIT | #BIT8,SR1   | ;DR11K #4 ?          |
| 4105 | 021436 | 001403 |        |        |       | BEQ | 55\$        |                      |
| 4106 | 021440 | 013721 | 001604 |        |       | MOV | DR11K4,(1)+ |                      |
| 4107 | 021444 | 000402 |        |        |       | BR  | 56\$        |                      |
| 4108 | 021446 | 012721 | 000001 |        | 55\$: | MOV | #1,(1)+     |                      |
| 4109 | 021452 | 032737 | 001000 | 001562 | 56\$: | BIT | #BIT9,SR1   | ;DR11K #5 ?          |
| 4110 | 021460 | 001403 |        |        |       | BEQ | 57\$        |                      |
| 4111 | 021462 | 013721 | 001606 |        |       | MOV | DR11K5,(1)+ |                      |
| 4112 | 021466 | 000402 |        |        |       | BR  | 60\$        |                      |
| 4113 | 021470 | 012711 | 000001 |        | 57\$: | MOV | #1,(1)      |                      |
| 4114 | 021474 |        |        |        | 60\$: |     |             |                      |



```

4148
4155 ;*****
(3) ;*TEST 34 *TRY TO GENERATE ERROR CODE 312
(4) ;
(4) ;*TEST THAT WE CAN GENERATE ERROR CODE 312
(4) ;*'USER NOT ACTIVE FOR STOP COMMAND'
(4) ;
(3) ;*****
(2) 021626 000004 TST34: SCOPE
4156
4157 021630 012737 000403 054776 MOV #403,JOBO ;SET USER#1, STOP COMMAND IN MODE
4158 ;WORD OF RDA #0
4159 021636 012737 055074 055002 MOV #JOB0U,JOBO+4 ;SET UP USW ADDR.
4160 021644 012777 054776 157624 MOV #JOB0,@LPADL ;LOAD ADDR OF RDA.
4161 021652 152777 000001 157606 BISB #1,@LPCI ;SET GO.
4162 021660 005000
4163 021662
4164 021662 105777 157604 1$: TSTB @LPCO ;READ OUT SET?
4165 021666 100412 BMI 2$;YES-EXIT LOOP
4166 021670 105200 INCB RO ;NO-LOOP OVERFLOW OCCUR?
4167 021672 100373 BPL 1$;NO-LOOP
4168
4169 021674 012737 000200 001124 MOV #200,$GDDAT ;/EXPECT 200.
(1) 021702 117737 157564 001126 MOVB @LPCO,$BDDAT ;/SEE WHAT WE GET.
4170 021710 104017 ERROR 17 ;READY OUT NOT SET ON CMND (KMC ERROR)
4171 021712 000413 BR TST35 ;;
4172
4173 021714
(1) 021714 012737 000312 001124 2$: MOV #312,$GDDAT ;/EXPECT 312.
(1) 021722 117737 157546 001126 MOVB @LPSO,$BDDAT ;/SEE WHAT WE GET.
4174 021730 122737 000312 001126 CMPB #312,$BDDAT ;DID CORRECT ERROR CODE GET RETURNED
4175 021736 001401 BEQ TST35 ;;
4176 021740 104017 ERROR 17 ;ERROR CODE 312 NOT RETURNED STOP USER ISSUED AND NO USE

```

4178  
4185  
4186  
(3)  
(4)  
(4)  
(4)  
(4)  
(3)  
(2)  
4187  
4188  
4189  
4198  
(1)  
(1)  
(1)  
(1)  
4199  
4200  
4201  
4202  
(1)  
4203  
4204  
4205  
4206  
4207  
4208  
4209  
4210  
(1)  
4211  
4212  
4213  
4214  
4215  
4216  
4217  
4218  
4219  
4220  
4221  
4222  
4223  
4224  
4225  
(1)  
4226  
4227  
4228  
4229  
(1)  
(1)  
4230

021742 000004  
021744 004737 040536  
021750 012777 054040 157520  
021756 005037 054040  
021762 113737 001512 054041  
021770 152777 000001 157470  
021776 004737 040452  
022002  
022002 105777 157460  
022006 100410  
022010 012737 000200 001124  
022016 017737 157444 001126  
022024 104017  
022026 000456  
022030  
022030 012737 000000 001124  
022036 117737 157430 001126  
022044 122737 000000 001126  
022052 001405  
022054 117737 157414 001126  
022062 104017  
022064 000437  
022066 012777 054040 157402  
022074 152777 000001 157364  
022102 005000  
022104 004737 040452  
022110 105777 157356  
022114 100410  
022116 012737 000200 001124  
022124 017737 157342 001126  
022132 104017  
022134 000413  
022136  
022136 012737 000310 001124  
022144 117737 157324 001126  
022152 122737 000310 001126

```

; *TEST 35 *TRY TO GENERATE ERROR CODE 310
; *
; *MAKE USRE WE CAN GENERATE ERROR CODE 310
; *'MULTIPLE INITIALIZE COMMAND'.
; *

TST35: SCOPE
JSR PC,SRESET ;LPA-11 RESET
MOV #DEVLST,@LPADL ;SET ADDR. OF DEVICE LIST IN RDA WORD
CLR DEVLST ;INDICATE SIZING
MOVB VERSN,DEVLST+1 ;SET VERSION NUMBER
BISB #1,@LPCI ;SET GO.
JSR PC, SDELAY
1$:
TSTB @LPCI ;READY IN SET?
BMI 2$
MOV #200,$GDDAT ;/EXPECT 200.
MOV @LPCI,$BDDAT ;/SEE WHAT WE GET.
ERROR 17 ;READY IN NOT SET ON INIT CMND
BR TST36 ;;
2$:
;NOTE-WE HAVE ALREADY DETERMINED
;DEVICES IN LIST ARE VALID
;SO WE SHOULD GET A '000'
;STATUS BACK FOR STAT REG.
MOV #0,$GDDAT ;/EXPECT 0.
MOVB @LPCO,$BDDAT ;/SEE WHAT WE GET.
CMPB #0,$BDDAT ;DID AN ERROR CODE GET RETURNED?
BEQ 3$;NO-NEXT
MOVB @LPCO,$BDDAT ;PUT ERROR CODE IN FOR TYPEOUT.
ERROR 17 ;KNOWN ERROR CODE RETURN ON INIT COMMAND
BR TST36 ;;
3$:
MOV #DEVLST,@LPADL ;RESET ADDR.
BISB #1,@LPCI ;SET GO-DO ANOTHER INITIALIZE CMND.
CLR RO
4$:
JSR PC,SDELAY
TSTB @LPCO ;READY OUT SET
BMI 5$
MOV #200,$GDDAT ;/EXPECT 200.
MOV @LPCO,$BDDAT ;/SEE WHAT WE GET.
ERROR 17 ;READY OUT NOT SET ON INIT COMMAND
BR TST36 ;;
5$:
MOV #310,$GDDAT ;/EXPECT 310.
MOVB @LPCO,$BDDAT ;/SEE WHAT WE GET.
CMPB #310,$BDDAT ;NOW WE EXPECT ERROR CODE 310
```

```

4231 022160 001401 BEQ TST36 ;;
4232
4233 022162 104017 ERROR 17 ;ERROR CODE 310 NOT RETURN ON DOUBLE INIT.
4234
4243 ;*****
(3) ;*TEST 36 *TEST INTERRUPTS
(4)
(4) ;*
(4) ;*IN THIS TEST WE'LL MAKE SURE READY IN AND READY OUT
(4) ;*CAN GENERATE INTERRUPTS
(4) ;*
(3) ;*****
(2) 022164 000004 TST36: SCOPE
4244
4245 022166 004737 040536 JSR PC,SRESET
4246 022172 013700 001556 MOV VECT1,R0 ;GET VECTOR ADDR.
4247 022176 012710 022360 MOV #4$, (0) ;SET ADDR FOR READY OUT INTERRUPT.
4248 022202 012760 022300 000004 MOV #1$,4(0) ;SET ADDR FOR READY IN INTERRUPT
4249 022210 012777 054040 157260 MOV #DEVLST,@LPADL ;SET ADDR. OF DEVICE LIST IN RDA WORD
(1) 022216 005037 054040 CLR DEVLST ;INDICATE SIZING
(1) 022222 113737 001512 054041 MOV VERSN,DEVLST+1 ;SET VERSION NUMBER
(1)
(1) 022230 152777 000001 157230 BISB #1,@LPCI ;SET GO.
(1) 022236 004737 040452 JSR PC, SDELAY
4250 022242 052777 000100 157216 BIS #100,@LPCI
4251
4252 022250 004737 040452 JSR PC, SDELAY ;INTERRUPT RIGHT AWAY.
4253 022254 012737 000300 001124 MOV #300,$GDDAT ;/EXPECT 300.
(1) 022262 017737 157200 001126 MOV @LPCI,$BDDAT ;/SEE WHAT WE GET.
4254 022270 005077 157172 CLR @LPCI ;CLEAR INTERRUPT ENABLE.
4255 022274 104017 ERROR 17 ;READY IN FAILED TO CAUSE AN INTERRUPT
4256 022276 000436 BR TST37 ;;
4257 022300 1$: ;DELAY SERVICING
4258
4259 022300 012777 054040 157170 2$: MOV #DEVLST,@LPADL ;SET FOR INIT.
4260 022306 152777 000001 157152 BISB #1,@LPCI
4261 022314 052777 000100 157150 BIS #BIT6,@LPCO ;SET READY OUT INTERRUPT ENABLE
4262 022322 005037 177776 CLR PS
4263 022326 3$:
4264 022326 004737 040452 JSR PC, SDELAY
4265 022332 100375 BPL 3$
4266 022334 012737 000300 001124 MOV #300,$GDDAT ;/EXPECT 300.
(1) 022342 017737 157124 001126 MOV @LPCO,$BDDAT ;/SEE WHAT WE GET.
4267 022350 105077 157116 CLRB @LPCO ;DISABLE ON ERROR.
4268 022354 104017 ERROR 17 ;READY OUT FAILED TO GENERATE AN ERROR
4269 022356 000406 BR TST37 ;;
4270 022360 4$:
4271 022360 105077 157102 5$: CLRB @LPCI
4272 022364 105077 157102 CLRB @LPCO
4273 022370 012706 001100 MOV #STACK,SP
4274
;STATE LOGIC TESTS DONE.

```

4299  
 4306  
 4307  
 (3)  
 (4)  
 (4)  
 (4)  
 (4)  
 (3)  
 (2)  
 (1)  
 4308  
 4309  
 4310  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 4311  
 4312  
 4313  
 4314  
 4315  
 4316  
 4317  
 4318  
 4319  
 4320  
 4321  
 4322  
 4323  
 4324  
 4325  
 4326  
 4327  
 4328  
 4329  
 4330  
 4331  
 4332  
 4333  
 4334  
 (1)  
 4335  
 4336  
 4337  
 4338  
 4339  
 4340  
 4341  
 (1)  
 4342  
 4343  
 4344  
 4447

022374 000004  
 022376 012737 000001 001160  
 022404 004737 040536  
 022410 012777 054040 157060  
 022416 005037 054040  
 022422 113737 001512 054041  
 022430 152777 000001 157030  
 022436 004737 040452  
 022442 032737 022002 001562  
 022450 001007  
 022452 032737 010001 001562  
 022460 001461  
 022462 104017  
 022464 000000  
 022466 000765  
 022470 012737 000001 044022  
 022476 123727 001772 000115  
 022504 001003  
 022506 052737 000010 044022  
 022514  
 022514 012737 000503 044024  
 022522 012737 177766 044026  
 022530 012777 044022 156740  
 022536 052777 000901 156722  
 022544 004737 040452  
 022550 032777 000200 156710  
 022556 001010  
 022560 012737 000200 001124  
 022566 017737 156674 001126  
 022574 104017  
 022576 000412  
 022600 105777 156666  
 022604 100007  
 022606 012737 000000 001124  
 022614 117737 156654 001126  
 022622 104017

```

*TEST 37 *TEST THAT THE LPA-SYSTEM CLOCK CAN BE STARTED
*IN THIS TEST WE WILL MAKE SURE WE CAN ORDERLY START
*THE SYSTEM CLOCK IF ONE EXISTS. IF NO CLOCK SPECIFIED BY
*SR1, WE'LL SEE IF AN ANOLOG DEVICE IS SPECIFIED, IF SO AN ERROR WILL
*BE REPORTED SINCE WE DO NEED A CLOCK.

TST37: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
JSR PC,SRESET
MOV #DEVLST,@LPADL ;SET ADDR. OF DEVICE LIST IN RDA WORD
CLR DEVLST ;INDICATE SIZING
MOVB VERSN,DEVLST+1 ;SET VERSION NUMBER
BISB #1,@LPCI ;SET GO.
JSR PC,SDELAY
1$: BIT #BIT1!BIT10!BIT13,SR1 ;IS A CLOCK SELECTED?
BNE 2$;YES-CONTINUE.
BIT #BIT0!BIT12,SR1 ;IS A CLOCK NEEDED?
BEQ TST40 ;;
ERROR 17 ;ERROR-NO CLOCK SPECIFIED FOR ANOLOG
HALT ;USER MAY NOT CONTINUE-FATAL ERROR
BR 1$;IF A CLOCK IS IN SYSTEM, THEN SELEC IT.
;IF YOU DESIRE TO RUN TIS DIAG. WITH NO CLOCK,
;Deselect ANOLOG OPTIONS.

2$: MOV #BIT0,KWT ;SELECT MODE:START CLOCK
CMPB $VERSN,#'M ;MULTI-REQUEST MODE?
BNE 4$
BIS #BIT3,KWT

4$: MOV #BIT1!BIT8!BIT0!BIT6,KWT+2 ;RATE:1MHZ,GO BIT, REPEATED INTERRUPT,IN
MOV #-10.,KWT+4 ;CLOCK INTERRUPT EVERY 10 USEC.
MOV #KWT,@LPADL ;PUT ADDR. IN LPA ADDR. REG
BIS #BIT0,@LPCI ;SET GO.

JSR PC,SDELAY
BIT #BIT7,@LPCI ;READY IN SET
BNE 3$
MOV #200,$GDDAT ;/EXPECT 200.
MOV @LPCI,$BDDAT ;/SEE WHAT WE GET.
ERROR 17 ;READY IN FAILED TO SET WHEN
;CLOCK STARTED.
;;

3$: TSTB @LPCO ;ERROR CODE RETURNED?
BPL TST40 ;;
MOV #0,$GDDAT ;/EXPECT 0.
MOVB @LP50,$BDDAT ;/SEE WHAT WE GET.

ERROR 17 ;ERROR CODE RETURNED ON
;CLOCK START.

```

4449

(5)  
(4)  
(5)  
(5)  
(5)  
(5)  
(5)  
(5)  
(5)  
(5)  
(5)  
(5)  
(6)  
(6)  
(6)  
(6)  
(6)  
(6)  
(6)  
(6)  
(4)

\*\*\*\*\*  
: \*TEST 40 \*TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #1  
: \*  
: \*IN THIS TEST WE'LL MAKE SURE THAT DR11K #1 WILL PASS DATA.  
: \*SR1 BIT2 SELECTS THIS DR11K FOR TEST. SR2 BIT2 INDICTES  
: \*THAT THE INPUT IS CALLED BACK TO THE OUTPUT.  
: \*  
: \* NOTE: THIS TEST IS A MULTI-USER MICRO-CODE TEST ONLY!  
: \* IF DEDICATE USER MICRO-CODE, THIS TEST WILL BE BY-PASSED.  
: \*  
: \*THIS TEST WAS DESIGNED IN ORDER TO FIND A WILD LPA-11 SYSTEM FAULT.  
: \*I DON'T REALLY EXPECT AN ERROR HERE UNLESS YOU'VE GOT A HEAVY FAULT.  
: \*ARBITRATION LOGIC IS ONE BIG REASON WHY YOU COULD FAIL. ANOTHER  
: \*IS THAT INTERRUPT VECTOR COULD BE WRONG. ALSO THE DMC MICRO-CODE MIGHT  
: \*NOT BE EXECUTING PROPERLY.  
: \*  
: \*\*\*\*\*

(3) 022624 000004  
(2) 022626 012737 000001 001160  
(1)  
(1) 022634 122737 000115 001772  
(1) 022642 001402  
(1) 022644 000137 023350  
(1) 022650 012700 054776  
(1) 022654 005020  
(1) 022656 020027 055466  
(1) 022662 001374  
(1)  
(1) 022664 032737 000004 001562  
(1) 022672 001002  
(1) 022674 000137 023350  
(1)  
(1) 022700 012700 055232  
(1) 022704 012710 000012  
(1) 022710 052710 000420  
(1) 022714 012760 000401 000002  
(1) 022722 012760 055330 000004  
(1) 022730 105060 000006  
(1) 022734 112760 000201 000007  
(1) 022742 012760 056156 000010  
(1) 022750 012760 057160 000014  
(1) 022756 005060 000012  
(1) 022762 012760 000200 000054  
(1) 022770 012760 000000 000056  
(1) 022776 012760 000001 000060  
(1) 023004 012760 000001 000062  
(1) 023012 005060 000064  
(1) 023016 005060 000066  
(1) 023022 005060 000070  
(1)  
(1) 023026 012701 055114  
(1) 023032 012711 000012  
(1) 023036 052711 000620  
(1) 023042 012761 000401 000002  
(1) 023050 012761 055212 000004

TST40: SCOPE  
MOV #1,\$TIMES ;:DO 1 ITERATION  
;/-DRLT-  
CMPB #'M,\$VERSN ;:RUNNING MULTI-USER MICRO-CODE?  
BEQ 1\$ ;:YES-CONTINUE.  
JMP 20\$ ;:NO-EXIT THIS TEST.  
1\$: MOV #JOB0,R0 ;:CLEAR OUT THE RDA TABLES  
25\$: CLR (R0)+  
CMP R0,#JOB4  
BNE 25\$  
  
BIT #BIT2,SR1 ;:IS THIS DR11K SELECTED FOR TEST?  
BNE 2\$ ;:YES-TEST IT.  
JMP 20\$ ;:NO-EXIT THIS TEST.  
  
2\$: MOV #JOB2,R0 ;:PICK UP THIS JOB ADDRESS.  
MOV #BIT1!BIT3,(0) ;:SET START AND MULTI-USER.  
BIS #BIT4!BIT8,(0) ;:SELECT DIGITAL INPUT SINGLE CHAN.  
3\$: MOV #257,,2(0) ;:SET WORD COUNT  
MOV #JOB2U,4(0) ;:USW ADDR.  
CLRB 6(0) ;:NO EXT.  
MOVB #201,7(0) ;:MAKE TWO BUFFERS AVAIL.  
MOV #BUFF0,10(0) ;:1ST BUFFER ADDR.  
MOV #BUFF1,14(0) ;:2ND BUFFER ADDR.  
CLR 12(0) ;:NO EXT.  
MOV #200,54(0) ;:DELAY =200 TICKS.  
MOV #1-1,56(0) ;:SET CHAN #=DR11K#-1.  
MOV #1,60(0) ;:SAMPLING ONLY ONE CHANNEL.  
MOV #1,62(0) ;:ON PULSE BETWEEN SAMPLE IS NO EXTERNAL TRIG.  
CLR 64(0) ;:NO START/EVENT MAKR WORD  
CLR 66(0) ;:NO START MASK  
CLR 70(0) ;:NO EVENT MASK.  
  
MOV #JOB1,R1 ;:OK-NOW LETS SET UP THE OUTPUT.  
MOV #BIT1!BIT3,(1) ;:SET START AND MULTI USER.  
BIS #BIT4!BIT7!BIT8,(1) ;:SELECT DIGITAL OUT.  
MOV #257,,2(1) ;:WORD COUNT  
MOV #JOB1U,4(1) ;:SET USW ADDR.

J 10

MAINDEC -11- CRLPA-B MACY11 30G(1063) 24-OCT-80 09:27 PAGE 26-1  
 CRLPAB.P11 20-OCT-80 13:17 T40 \*TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #1 SEQ 0126

```

(1) 023056 105061 000006 CLRB 6(1) ;NO EXT.
(1) 023062 112761 000201 000007 MOVB #201,7(1)
(1) 023070 012761 060162 000010 MOV #BUFF2,10(1) ;SET BUFFER #1 ADDR.
(1) 023076 012761 061164 000014 MOV #BUFF3,14(1) ;SET BUFFER #2 ADDR.
(1) 023104 012761 060162 000020 MOV #BUFF2,20(1) ;THE THIRD OUTPUT IS NECESSARY
(1) ;TO COMPENSATE FOR SAMPLES LOST
(1) 023112 105061 000012 CLRB 12(1) ;NO EXT.
(1) 023116 012761 000200 000054 MOV #200,54(1) ;DELAY 1 TICK.
(1) 023124 012761 000000 000056 MOV #1-1,56(1) ;SET CHAN #= DR#-1
(1) 023132 012761 000001 000060 MOV #1,60(1) ;SAMPLE ONLY ONE CHAN.
(1) 023140 012761 000001 000062 MOV #1,62(1) ;ONE TICK BETWEEN SAMPLE
(1) 023146 005061 000064 CLR 64(1) ;NO START/EVENT
(1) 023152 005061 000066 CLR 66(1) ;NO START MASK.
(1) 023156 005061 000070 CLR 70(1) ;NO EVENT MASK.
(1)
(1) 023162 012700 125252 MOV #125252,R0 ;FILL OUTPUT BUFFER W/PATTERN
(1) 023166 012701 001002 MOV #514.,R1
(1) 023172 012702 060162 MOV #BUFF2,R2
(1) 023176 010022 4$: MOV R0,(2)+
(1) 023200 005100 COM R0
(1) 023202 005301 DEC R1
(1) 023204 001374 BNE 4$
(1)
(1) 023206 012700 056156 MOV #BUFF0,R0 ;CLEAR INPUT BUFFER
(1) 023212 012701 001002 MOV #514.,R1
(1) 023216 005020 5$: CLR (0)+
(1) 023220 005301 DEC R1
(1) 023222 001375 BNE 5$
(1)
(1) 023224 012737 000011 054776 MOV #11,JOB0 ;OK,NOW WE GOTTO STOP THE CLOCK
(1) 023232 012737 000000 055000 MOV #0,JOB0+2 ;SO THAT IT DOESN'T START UNTIL
(1) 023240 012737 000011 055350 MOV #11,JOB3 ;DRIIK PARAMETER SET UP,THEN AS
(1) 023246 012737 000503 055352 MOV #503,JOB3+2 ;JOB #4,WE START CLOCK.
(1) 023254 012737 177000 055354 MOV #177000,JOB3+4
(1) 023262 012737 000004 001124 MOV #4,$GDDAT ;FOUR JOBS.
(1) 023270 012737 000002 002014 MOV #2,$FUDGE ;/SET UP A FUDGE FACTOR TO
(1) ;/COMPENSATE FOR THE TWO KW11K JOBS
(1) 023276 004737 036126 JSR PC, FLASH ;/GO DO THEM!
(1)
(1) 023302 032737 000004 001564 BIT #BIT2,SR2 ;/CABLED TOGETHER?
(3) 023310 001417 BEQ TST41
(1) 023312 012700 001003 MOV #515.,R0 ;/NOW CHECK DATA IF CABLED.
(1) 023316 012701 060162 MOV #BUFF2,R1
(1) 023322 012702 056156 MOV #BUFF0,R2
(1) 023326 005300 6$: DEC R0
(3) 023330 001407 BEQ TST41 ;;
(1) 023332 022221 CMP (R2)+,(R1)+
(1) 023334 001774 BEQ 6$
(1) 023336 014137 001124 MOV -(R1),$GDDAT
(1) 023342 014237 001126 MOV -(R2),$BDDAT
(1) 023346 104012 ERROR 12
(1) 023350 20$:

```



4451

```
(5)
(4)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(6)
(4)
(3) 023350 000004
(2) 023352 012737 000001 001160
(1)
(1) 023360 122737 000115 001772
(1) 023366 001402
(1) 023370 000137 024074
(1) 023374 012700 054776
(1) 023400 005020
(1) 023402 020027 055466
(1) 023406 001374
(1)
(1) 023410 032737 000040 001562
(1) 023416 001002
(1) 023420 000137 024074
(1)
(1) 023424 012700 055232
(1) 023430 012710 000012
(1) 023434 052710 000420
(1) 023440 012760 000401 000002
(1) 023446 012760 055330 000004
(1) 023454 105060 000006
(1) 023460 112760 000201 000007
(1) 023466 012760 056156 000010
(1) 023474 012760 057160 000014
(1) 023502 005060 000012
(1) 023506 012760 000200 000054
(1) 023514 012760 000001 000056
(1) 023522 012760 000001 000060
(1) 023530 012760 000001 000062
(1) 023536 005060 000064
(1) 023542 005060 000066
(1) 023546 005060 000070
(1)
(1) 023552 012701 055114
(1) 023556 012711 000012
(1) 023562 052711 000620
(1) 023566 012761 000401 000002
(1) 023574 012761 055212 000004
```

```

*TEST 41 *TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #2
*
*IN THIS TEST WE'LL MAKE SURE THAT DR11K #2 WILL PASS DATA.
*SR1 BITS SELECTS THIS DR11K FOR TEST. SR2 BITS INDICATES
*THAT THE INPUT IS CALLED BACK TO THE OUTPUT.
*
* NOTE: THIS TEST IS A MULTI-USER MICRO-CODE TEST ONLY!
* IF DEDICATE USER MICRO-CODE, THIS TEST WILL BE BY-PASSED.
*
*THIS TEST WAS DESIGNED IN ORDER TO FIND A WILD LPA-11 SYSTEM FAULT.
*I DON'T REALLY EXPECT AN ERROR HERE UNLESS YOU'VE GOT A HEAVY FAULT.
*ARBITRATION LOGIC IS ONE BIG REASON WHY YOU COULD FAIL. ANOTHER
*IS THAT INTERRUPT VECTOR COULD BE WRONG. ALSO THE DMC MICRO-CODE MIGHT
*NOT BE EXECUTING PROPERLY.
*

TST41: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
;/-DRLT-
CMPB #'M,$VERSN ;RUNNING MULTI-USER MICRO-CODE?
BEQ 1$;YES-CONTINUE.
JMP 20$;NO-EXIT THIS TEST.
1$: MOV #JOB0,R0 ;CLEAR OUT THE RDA TABLES
25$: CLR (R0)+
CMP R0,#JOB4
BNE 25$
(1)
(1) 023410 032737 000040 001562 BIT #BIT5,SR1 ;IS THIS DR11K SELECTED FOR TEST?
(1) 023416 001002 BNE 2$;YES-TEST IT.
(1) 023420 000137 024074 JMP 20$;NO-EXIT THIS TEST.
(1)
(1) 023424 012700 055232 2$: MOV #JOB2,R0 ;PICK UP THIS JOB ADDRESS.
(1) 023430 012710 000012 MOV #BIT1!BIT3,(0) ;SET START AND MULTI-USER.
(1) 023434 052710 000420 BIS #BIT4!BIT8,(0) ;SELECT DIGITAL INPUT SINGLE CHAN.
(1) 023440 012760 000401 000002 3$: MOV #257,,2(0) ;SET WORD COUNT
(1) 023446 012760 055330 000004 MOV #JOB2U,4(0) ;USW ADDR.
(1) 023454 105060 000006 CLRB 6(0) ;NO EXT.
(1) 023460 112760 000201 000007 MOVB #201,7(0) ;MAKE TWO BUFFERS AVAIL.
(1) 023466 012760 056156 000010 MOV #BUFF0,10(0) ;1ST BUFFER ADDR.
(1) 023474 012760 057160 000014 MOV #BUFF1,14(0) ;2ND BUFFER ADDR.
(1) 023502 005060 000012 CLR 12(0) ;NO EXT.
(1) 023506 012760 000200 000054 MOV #200,54(0) ;DELAY =200 TICKS.
(1) 023514 012760 000001 000056 MOV #2-1,56(0) ;SET CHAN #=DR11K#-1.
(1) 023522 012760 000001 000060 MOV #1,60(0) ;SAMPLING ONLY ONE CHANNEL.
(1) 023530 012760 000001 000062 MOV #1,62(0) ;ON PULSE BETWEEN SAMPLE IS NO EXTERNAL TRIG.
(1) 023536 005060 000064 CLR 64(0) ;NO START/EVENT MAKR WORD
(1) 023542 005060 000066 CLR 66(0) ;NO START MASK
(1) 023546 005060 000070 CLR 70(0) ;NO EVENT MASK.
(1)
(1) 023552 012701 055114 MOV #JOB1,R1 ;OK-NOW LETS SET UP THE OUTPUT.
(1) 023556 012711 000012 MOV #BIT1!BIT3,(1) ;SET START AND MULTI USER.
(1) 023562 052711 000620 BIS #BIT4!BIT7!BIT8,(1) ;SELECT DIGITAL OUT.
(1) 023566 012761 000401 000002 MOV #257,,2(1) ;WORD COUNT
(1) 023574 012761 055212 000004 MOV #JOB1U,4(1) ;SET USW ADDR.
```

```

(1) 023602 105061 000006 CLRB 6(1) ;NO EXT.
(1) 023606 112761 000201 000007 MOVB #201,7(1)
(1) 023614 012761 060162 000010 MOV #BUFF2,10(1) ;SET BUFFER #1 ADDR.
(1) 023622 012761 061164 000014 MOV #BUFF3,14(1) ;SET BUFFER #2 ADDR.
(1) 023630 012761 060162 000020 MOV #BUFF2,20(1) ;THE THIRD OUTPUT IS NECESSARY
(1) ;TO COMPENSATE FOR SAMPLES LOST
(1) 023636 105061 000012 CLRB 12(1) ;NO EXT.
(1) 023642 012761 000200 000054 MOV #200,54(1) ;DELAY 1 TICK.
(1) 023650 012761 000001 000056 MOV #2-1,56(1) ;SET CHAN #= DR#-1
(1) 023656 012761 000001 000060 MOV #1,60(1) ;SAMPLE ONLY ONE CHAN.
(1) 023664 012761 000001 000062 MOV #1,62(1) ;ONE TICK BETWEEN SAMPLE
(1) 023672 005061 000064 CLR 64(1) ;NO START/EVENT
(1) 023676 005061 000066 CLR 66(1) ;NO START MASK.
(1) 023702 005061 000070 CLR 70(1) ;NO EVENT MASK.

(1) 023706 012700 125252 MOV #125252,R0 ;FILL OUTPUT BUFFER W/PATTERN
(1) 023712 012701 001002 MOV #514.,R1
(1) 023716 012702 060162 MOV #BUFF2,R2
(1) 023722 010022 4$: MOV R0,(2)+
(1) 023724 005100 COM R0
(1) 023726 005301 DEC R1
(1) 023730 001374 BNE 4$

(1) 023732 012700 056156 MOV #BUFF0,R0 ;CLEAR INPUT BUFFER
(1) 023736 012701 001002 MOV #514.,R1
(1) 023742 005020 5$: CLR (0)+
(1) 023744 005301 DEC R1
(1) 023746 001375 BNE 5$

(1) 023750 012737 000011 054776 MOV #11,JOB0 ;OK,NOW WE GOTTO STOP THE CLOCK
(1) 023756 012737 000000 055000 MOV #0,JOB0+2 ;SO THAT IT DOESN'T START UNTIL
(1) 023764 012737 000011 055350 MOV #11,JOB3 ;DRIK PARAMETER SET UP,THEN AS
(1) 023772 012737 000503 055352 MOV #503,JOB3+2 ;JOB #4,WE START CLOCK.
(1) 024000 012737 177000 055354 MOV #177000,JOB3+4
(1) 024006 012737 000004 001124 MOV #4,$GDDAT ;FOUR JOBS.
(1) 024014 012737 000002 002014 MOV #2,FUDGE ;/SET UP A FUDGE FACTOR TO
(1) ;/COMPENSATE FOR THE TWO KW11K JOBS
(1) 024022 004737 036126 JSR PC, FLASH ;/GO DO THEM!

(1) 024026 032737 000040 001564 BIT #BITS,SR2 ;/CABLED TOGETHER?
(3) 024034 001417 BEQ TST42
(1) 024036 012700 001003 MOV #515.,R0 ;/NOW CHECK DATA IF CABLED.
(1) 024042 012701 060162 MOV #BUFF2,R1
(1) 024046 012702 056156 MOV #BUFF0,R2
(1) 024052 005300 6$: DEC R0
(3) 024054 001407 BEQ TST42 ;;
(1) 024056 022221 CMP (R2)+,(R1)+
(1) 024060 001774 BEQ 6$
(1) 024062 014137 001124 MOV -(R1),$GDDAT
(1) 024066 014237 001126 MOV -(R2),$BDDAT
(1) 024072 104012 ERROR 12
(1) 024074 20$:

```

4453

(5)  
 (4)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (6)  
 (6)  
 (6)  
 (6)  
 (6)  
 (6)  
 (6)  
 (6)  
 (4)  
 (3) 024074 000004  
 (2) 024076 012737 000001 001160  
 (1)  
 (1) 024104 122737 000115 001772  
 (1) 024112 001402  
 (1) 024114 000137 024620  
 (1) 024120 012700 054776  
 (1) 024124 005020  
 (1) 024126 020027 055466  
 (1) 024132 001374  
 (1)  
 (1) 024134 032737 000200 001562  
 (1) 024142 001002  
 (1) 024144 000137 024620  
 (1)  
 (1) 024150 012700 055232  
 (1) 024154 012710 000012  
 (1) 024160 052710 000420  
 (1) 024164 012760 000401 000002  
 (1) 024172 012760 055330 000004  
 (1) 024200 105060 000006  
 (1) 024204 112760 000201 000007  
 (1) 024212 012760 056156 000010  
 (1) 024220 012760 057160 000014  
 (1) 024226 005060 000012  
 (1) 024232 012760 000200 000054  
 (1) 024240 012760 000002 000056  
 (1) 024246 012760 000001 000060  
 (1) 024254 012760 000001 000062  
 (1) 024262 005060 000064  
 (1) 024266 005060 000066  
 (1) 024272 005060 000070  
 (1)  
 (1) 024276 012701 055114  
 (1) 024302 012711 000012  
 (1) 024306 052711 000620  
 (1) 024312 012761 000401 000002  
 (1) 024320 012761 055212 000004

```

*TEST 42 *TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #3
*
*IN THIS TEST WE'LL MAKE SURE THAT DR11K #3 WILL PASS DATA.
*SR1 BIT7 SELECTS THIS DR11K FOR TEST. SR2 BIT7 INDICTES
*THAT THE INPUT IS CALLED BACK TO THE OUTPUT.
*
* NOTE: THIS TEST IS A MULTI-USER MICRO-CODE TEST ONLY!
* IF DEDICATE USER MICRO-CODE, THIS TEST WILL BE BY-PASSED.
*
*THIS TEST WAS DESIGNED IN ORDER TO FIND A WILD LPA-11 SYSTEM FAULT.
*I DON'T REALLY EXPECT AN ERROR HERE UNLESS YOU'VE GOT A HEAVY FAULT.
*ARBITRATION LOGIC IS ONE BIG REASON WHY YOU COULD FAIL. ANOTHER
*IS THAT INTERRUPT VECTOR COULD BE WRONG. ALSO THE DMC MICRO-CODE MIGHT
*NOT BE EXECUTING PROPERLY.

TST42: SCOPE
 MOV #1,$TIMES ;;DO 1 ITERATION
 ;/-DRLT-
 CMPB #'M,$VERSN ;RUNNING MULTI-USER MICRO-CODE?
 BEQ 1$;YES-CONTINUE.
 JMP 20$;NO-EXIT THIS TEST.
1$: MOV #JOB0,R0 ;CLEAR OUT THE RDA TABLES
25$: CLR (R0)+
 CMP R0,#JOB4
 BNE 25$

 BIT #BIT7,SR1 ;IS THIS DR11K SELECTED FOR TEST?
 BNE 2$;YES-TEST IT.
 JMP 20$;NO-EXIT THIS TEST.

2$: MOV #JOB2,R0 ;PICK UP THIS JOB ADDRESS.
 MOV #BIT1!BIT3,(0) ;SET START AND MULTI-USER.
 BIS #BIT4!BIT8,(0) ;SELECT DIGITAL INPUT SINGLE CHAN.
3$: MOV #257,,2(0) ;SET WORD COUNT
 MOV #JOB2U,4(0) ;USW ADDR.
 CLRB 6(0) ;NO EXT.
 MOVB #201,7(0) ;MAKE TWO BUFFERS AVAIL.
 MOV #BUFF0,10(0) ;1ST BUFFER ADDR.
 MOV #BUFF1,14(0) ;2ND BUFFER ADDR.
 CLR 12(0) ;NO EXT.
 MOV #200,54(0) ;DELAY =200 TICKS.
 MOV #3-1,56(0) ;SET CHAN #=DR11K#-1.
 MOV #1,60(0) ;SAMPLING ONLY ONE CHANNEL.
 MOV #1,62(0) ;ON PULSE BETWEEN SAMPLE IS NO EXTERNAL TRIG.
 CLR 64(0) ;NO START/EVENT MAKR WORD
 CLR 66(0) ;NO START MASK
 CLR 70(0) ;NO EVENT MASK.

 MOV #JOB1,R1 ;OK-NOW LETS SET UP THE OUTPUT.
 MOV #BIT1!BIT3,(1) ;SET START AND MULTI USER.
 BIS #BIT4!BIT7!BIT8,(1) ;SELECT DIGITAL OUT.
 MOV #257,,2(1) ;WORD COUNT
 MOV #JOB1U,4(1) ;SET USW ADDR.

```

|     |        |        |        |        |       |                |                                                                   |
|-----|--------|--------|--------|--------|-------|----------------|-------------------------------------------------------------------|
| (1) | 024326 | 105061 | 000006 |        | CLRB  | 6(1)           | ;NO EXT.                                                          |
| (1) | 024332 | 112761 | 000201 | 000007 | MOVB  | #201,7(1)      |                                                                   |
| (1) | 024340 | 012761 | 060162 | 000010 | MOV   | #BUFF2,10(1)   | ;SET BUFFER #1 ADDR.                                              |
| (1) | 024346 | 012761 | 061164 | 000014 | MOV   | #BUFF3,14(1)   | ;SET BUFFER #2 ADDR.                                              |
| (1) | 024354 | 012761 | 060162 | 000020 | MOV   | #BUFF2,20(1)   | ;THE THIRD OUTPUT IS NECESSARY<br>;TO COMPENSATE FOR SAMPLES LOST |
| (1) | 024362 | 105061 | 000012 |        | CLRB  | 12(1)          | ;NO EXT.                                                          |
| (1) | 024366 | 012761 | 000200 | 000054 | MOV   | #200,54(1)     | ;DELAY 1 TICK.                                                    |
| (1) | 024374 | 012761 | 000002 | 000056 | MOV   | #3-1,56(1)     | ;SET CHAN # = DR#-1                                               |
| (1) | 024402 | 012761 | 000001 | 000060 | MOV   | #1,60(1)       | ;SAMPLE ONLY ONE CHAN.                                            |
| (1) | 024410 | 012761 | 000001 | 000062 | MOV   | #1,62(1)       | ;ONE TICK BETWEEN SAMPLE                                          |
| (1) | 024416 | 005061 | 000064 |        | CLR   | 64(1)          | ;NO START/EVENT                                                   |
| (1) | 024422 | 005061 | 000066 |        | CLR   | 66(1)          | ;NO START MASK.                                                   |
| (1) | 024426 | 005061 | 000070 |        | CLR   | 70(1)          | ;NO EVENT MASK.                                                   |
| (1) | 024432 | 012700 | 125252 |        | MOV   | #125252,R0     | ;FILL OUTPUT BUFFER W/PATTERN                                     |
| (1) | 024436 | 012701 | 001002 |        | MOV   | #514.,R1       |                                                                   |
| (1) | 024442 | 012702 | 060162 |        | MOV   | #BUFF2,R2      |                                                                   |
| (1) | 024446 | 010022 |        | 4\$:   | MOV   | R0,(2)+        |                                                                   |
| (1) | 024450 | 005100 |        |        | COM   | R0             |                                                                   |
| (1) | 024452 | 005301 |        |        | DEC   | R1             |                                                                   |
| (1) | 024454 | 001374 |        |        | BNE   | 4\$            |                                                                   |
| (1) | 024456 | 012700 | 056156 |        | MOV   | #BUFF0,R0      | ;CLEAR INPUT BUFFER                                               |
| (1) | 024462 | 012701 | 001002 |        | MOV   | #514.,R1       |                                                                   |
| (1) | 024466 | 005020 |        | 5\$:   | CLR   | (0)+           |                                                                   |
| (1) | 024470 | 005301 |        |        | DEC   | R1             |                                                                   |
| (1) | 024472 | 001375 |        |        | BNE   | 5\$            |                                                                   |
| (1) | 024474 | 012737 | 000011 | 054776 | MOV   | #11,JOB0       | ;OK,NOW WE GOTTO STOP THE CLOCK                                   |
| (1) | 024502 | 012737 | 000000 | 055000 | MOV   | #0,JOB0+2      | ;SO THAT IT DOESN'T START UNTIL                                   |
| (1) | 024510 | 012737 | 000011 | 055350 | MOV   | #11,JOB3       | ;DRIIK PARAMETER SET UP,THEN AS                                   |
| (1) | 024516 | 012737 | 000503 | 055352 | MOV   | #503,JOB3+2    | ;JOB #4,WE START CLOCK.                                           |
| (1) | 024524 | 012737 | 177000 | 055354 | MOV   | #177000,JOB3+4 |                                                                   |
| (1) | 024532 | 012737 | 000004 | 001124 | MOV   | #4,\$GDDAT     | ;FOUR JOBS.                                                       |
| (1) | 024540 | 012737 | 000002 | 002014 | MOV   | #2,FUDGE       | ;SET UP A FUDGE FACTOR TO                                         |
| (1) | 024546 | 004737 | 036126 |        | JSR   | PC, FLASH      | ;COMPENSATE FOR THE TWO KW11K JOBS                                |
| (1) | 024552 | 032737 | 000200 | 001564 |       |                | ;GO DO THEM!                                                      |
| (3) | 024560 | 001417 |        |        | BIT   | #BIT7,SR2      | ;CABLED TOGETHER?                                                 |
| (1) | 024562 | 012700 | 001003 |        | BEQ   | TST43          | ::                                                                |
| (1) | 024566 | 012701 | 060162 |        | MOV   | #515.,R0       | ;NOW CHECK DATA IF CABLED.                                        |
| (1) | 024572 | 012702 | 056156 |        | MOV   | #BUFF2,R1      |                                                                   |
| (1) | 024576 | 005300 |        | 6\$:   | MOV   | #BUFF0,R2      |                                                                   |
| (3) | 024600 | 001407 |        |        | DEC   | R0             |                                                                   |
| (1) | 024602 | 022221 |        |        | BEQ   | TST43          | ::                                                                |
| (1) | 024604 | 001774 |        |        | CMP   | (R2)+,(R1)+    |                                                                   |
| (1) | 024606 | 014137 | 001124 |        | BEQ   | 6\$            |                                                                   |
| (1) | 024612 | 014237 | 001126 |        | MOV   | -(R1),\$GDDAT  |                                                                   |
| (1) | 024616 | 104012 |        |        | MOV   | -(R2),\$BDDAT  |                                                                   |
| (1) | 024620 |        |        | 20\$:  | ERROR | 12             |                                                                   |

4455

(5)  
(4)  
(5)  
(5)  
(5)  
(5)  
(5)  
(5)  
(5)  
(5)  
(6)  
(6)  
(6)  
(6)  
(6)  
(6)  
(6)  
(6)  
(6)  
(4)

```

*TEST 43 *TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #4
*
*IN THIS TEST WE'LL MAKE SURE THAT DR11K #4 WILL PASS DATA.
*SR1 BIT8 SELECTS THIS DR11K FOR TEST. SR2 BIT8 INDICTES
*THAT THE INPUT IS CALLED BACK TO THE OUTPUT.
*
* NOTE: THIS TEST IS A MULTI-USER MICRO-CODE TEST ONLY!
* IF DEDICATE USER MICRO-CODE, THIS TEST WILL BE BY-PASSED.
*
*THIS TEST WAS DESIGNED IN ORDER TO FIND A WILD LPA-11 SYSTEM FAULT.
*I DON'T REALLY EXPECT AN ERROR HERE UNLESS YOU'VE GOT A HEAVY FAULT.
*ARBITRATION LOGIC IS ONE BIG REASON WHY YOU COULD FAIL. ANOTHER
*IS THAT INTERRUPT VECTOR COULD BE WRONG. ALSO THE DMC MICRO-CODE MIGHT
*NOT BE EXECUTING PROPERLY.

```

```

(3) 024620 000004
(2) 024622 012737 000001 001160
(1)
(1) 024630 122737 000115 001772
(1) 024636 001402
(1) 024640 000137 025344
(1) 024644 012700 054776
(1) 024650 005020
(1) 024652 020027 055466
(1) 024656 001374
(1)
(1) 024660 032737 000400 001562
(1) 024666 001002
(1) 024670 000137 025344
(1)
(1) 024674 012700 055232
(1) 024700 012710 000012
(1) 024704 052710 000420
(1) 024710 012760 000401 000002
(1) 024716 012760 055330 000004
(1) 024724 105060 000006
(1) 024730 112760 000201 000007
(1) 024736 012760 056156 000010
(1) 024744 012760 057160 000014
(1) 024752 005060 000012
(1) 024756 012760 000200 000054
(1) 024764 012760 000003 000056
(1) 024772 012760 000001 000060
(1) 025000 012760 000001 000062
(1) 025006 005060 000064
(1) 025012 005060 000066
(1) 025016 005060 000070
(1)
(1) 025022 012701 055114
(1) 025026 012711 000012
(1) 025032 052711 000620
(1) 025036 012761 000401 000002
(1) 025044 012761 055212 000004

```

```

TST43: SCOPE
MOV #1,$TIMES ;:DO 1 ITERATION
;/-DRLT-
CMPB #'M,$VERSN ;:RUNNING MULTI-USER MICRO-CODE?
BEQ 1$;:YES-CONTINUE.
JMP 20$;:NO-EXIT THIS TEST.
1$: MOV #JOB0,R0 ;:CLEAR OUT THE RDA TABLES
25$: CLR (R0)+
CMP R0,#JOB4
BNE 25$

BIT #BIT8,SR1 ;:IS THIS DR11K SELECTED FOR TEST?
BNE 2$;:YES-TEST IT.
JMP 20$;:NO-EXIT THIS TEST.

2$: MOV #JOB2,R0 ;:PICK UP THIS JOB ADDRESS.
MOV #BIT1!BIT3,(0) ;:SET START AND MULTI-USER.
BIS #BIT4!BIT8,(0) ;:SELECT DIGITAL INPUT SINGLE CHAN.
3$: MOV #257,,2(0) ;:SET WORD COUNT
MOV #JOB2U,4(0) ;:USW ADDR.
CLRB 6(0) ;:NO EXT.
MOVB #201,7(0) ;:MAKE TWO BUFFERS AVAIL.
MOV #BUFF0,10(0) ;:1ST BUFFER ADDR.
MOV #BUFF1,14(0) ;:2ND BUFFER ADDR.
CLR 12(0) ;:NO EXT.
MOV #200,54(0) ;:DELAY =200 TICKS.
MOV #4-1,56(0) ;:SET CHAN #=DR11K#-1.
MOV #1,60(0) ;:SAMPLING ONLY ONE CHANNEL.
MOV #1,62(0) ;:ON PULSE BETWEEN SAMPLE IS NO EXTERNAL TRIG.
CLR 64(0) ;:NO START/EVENT MAKR WORD
CLR 66(0) ;:NO START MASK
CLR 70(0) ;:NO EVENT MASK.

MOV #JOB1,R1 ;:OK-NOW LETS SET UP THE OUTPUT.
MOV #BIT1!BIT3,(1) ;:SET START AND MULTI USER.
BIS #BIT4!BIT7!BIT8,(1) ;:SELECT DIGITAL OUT.
MOV #257,,2(1) ;:WORD COUNT
MOV #JOB1U,4(1) ;:SET USW ADDR.

```

```

(1) 025052 105061 000006 CLR B 6(1) ;NO EXT.
(1) 025056 112761 000201 000007 MOV B #201,7(1)
(1) 025064 012761 060162 000010 MOV #BUFF2,10(1) ;SET BUFFER #1 ADDR.
(1) 025072 012761 061164 000014 MOV #BUFF3,14(1) ;SET BUFFER #2 ADDR.
(1) 025100 012761 060162 000020 MOV #BUFF2,20(1) ;THE THIRD OUTPUT IS NECESSARY
(1) ;TO COMPENSATE FOR SAMPLES LOST
(1) 025106 105061 000012 CLR B 12(1) ;NO EXT.
(1) 025112 012761 000200 000054 MOV #200,54(1) ;DELAY 1 TICK.
(1) 025120 012761 000003 000056 MOV #4-1,56(1) ;SET CHAN #= DR#-1
(1) 025126 012761 000001 000060 MOV #1,60(1) ;SAMPLE ONLY ONE CHAN.
(1) 025134 012761 000001 000062 MOV #1,62(1) ;ONE TICK BETWEEN SAMPLE
(1) 025142 005061 000064 CLR 64(1) ;NO START/EVENT
(1) 025146 005061 000066 CLR 66(1) ;NO START MASK.
(1) 025152 005061 000070 CLR 70(1) ;NO EVENT MASK.
(1)
(1) 025156 012700 125252 MOV #125252,R0 ;FILL OUTPUT BUFFER W/PATTERN
(1) 025162 012701 001002 MOV #514.,R1
(1) 025166 012702 060162 MOV #BUFF2,R2
(1) 025172 010022 4$: MOV R0,(2)+
(1) 025174 005100 COM R0
(1) 025176 005301 DEC R1
(1) 025200 001374 BNE 4$
(1)
(1) 025202 012700 056156 MOV #BUFF0,R0 ;CLEAR INPUT BUFFER
(1) 025206 012701 001002 MOV #514.,R1
(1) 025212 005020 5$: CLR (0)+
(1) 025214 005301 DEC R1
(1) 025216 001375 BNE 5$
(1)
(1) 025220 012737 000011 054776 MOV #11,JOB0 ;OK,NOW WE GOTTO STOP THE CLOCK
(1) 025226 012737 000000 055000 MOV #0,JOB0+2 ;SO THAT IT DOESN'T START UNTIL
(1) 025234 012737 000011 055350 MOV #11,JOB3 ;DRIK PARAMETER SET UP,THEN AS
(1) 025242 012737 000503 055352 MOV #503,JOB3+2 ;JOB #4,WE START CLOCK.
(1) 025250 012737 177000 055354 MOV #177000,JOB3+4
(1) 025256 012737 000004 001124 MOV #4,$GDDAT ;FOUR JOBS.
(1) 025264 012737 000002 002014 MOV #2,FUDGE ;/SET UP A FUDGE FACTOR TO
(1) ;/COMPENSATE FOR THE TWO KW11K JOBS
(1) 025272 004737 036126 JSR PC, FLASH ;/GO DO THEM!
(1)
(1) 025276 032737 000400 001564 BIT #BIT8,SR2 ;/CABLED TOGETHER?
(3) 025304 001417 BEQ TST44 ;:
(1) 025306 012700 001003 MOV #515.,R0 ;/NOW CHECK DATA IF CABLED.
(1) 025312 012701 060162 MOV #BUFF2,R1
(1) 025316 012702 056156 MOV #BUFF0,R2
(1) 025322 005300 6$: DEC R0
(3) 025324 001407 BEQ TST44 ;:
(1) 025326 022221 CMP (R2)+,(R1)+
(1) 025330 001774 BEQ 6$
(1) 025332 014137 001124 MOV -(R1),$GDDAT
(1) 025336 014237 001126 MOV -(R2),$BDDAT
(1) 025342 104012 ERROR 12
(1) 025344 20$:

```

4457

(5)  
 (4)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (6)  
 (6)  
 (6)  
 (6)  
 (6)  
 (6)  
 (6)  
 (4)  
 (3)  
 (2)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1)

```

025344 000004
025346 012737 000001 001160
025354 122737 000115 001772
025362 001402
025364 000137 026070
025370 012700 054776
025374 005020
025376 020027 055466
025402 001374
025404 032737 001000 001562
025412 001002
025414 000137 026070
025420 012700 055232
025424 012710 000012
025430 052710 000420
025434 012760 000401 000002
025442 012760 055330 000004
025450 105060 000006
025454 112760 000201 000007
025462 012760 056156 000010
025470 012760 057160 000014
025476 005060 000012
025502 012760 000200 000054
025510 012760 000004 000056
025516 012760 000001 000060
025524 012760 000001 000062
025532 005060 000064
025536 005060 000066
025542 005060 000070
025546 012701 055114
025552 012711 000012
025556 052711 000620
025562 012761 000401 000002
025570 012761 055212 000004

```

```

*TEST 44 *TEST THAT THE MULTI-USER MICRO-CODE WILL EXERCISE DR11K #5
*
*IN THIS TEST WE'LL MAKE SURE THAT DR11K #5 WILL PASS DATA.
*SR1 BIT9 SELECTS THIS DR11K FOR TEST. SR2 BIT9 INDICTES
*THAT THE INPUT IS CALLED BACK TO THE OUTPUT.
*
* NOTE: THIS TEST IS A MULTI-USER MICRO-CODE TEST ONLY.
* IF DEDICATE USER MICRO-CODE, THIS TEST WILL BE BY-PASSED.
*
*THIS TEST WAS DESIGNED IN ORDER TO FIND A WILD LPA-11 SYSTEM FAULT.
*I DON'T REALLY EXPECT AN ERROR HERE UNLESS YOU'VE GOT A HEAVY FAULT.
*ARBITRATION LOGIC IS ONE BIG REASON WHY YOU COULD FAIL. ANOTHER
*IS THAT INTERRUPT VECTOR COULD BE WRONG. ALSO THE DMC MICRO-CODE MIGHT
*NOT BE EXECUTING PROPERLY.

```

```

TST44: SCOPE
 MOV #1,$TIMES ;;DO 1 ITERATION
 ;/-DRLT-
 CMPB #'M,$VERSN ;RUNNING MULTI-USER MICRO-CODE?
 BEQ 1$;YES-CONTINUE.
 JMP 20$;NO-EXIT THIS TEST.
1$: MOV #JOB0,R0 ;CLEAR OUT THE RDA TABLES
25$: CLR (R0)+
 CMP R0,#JOB4
 BNE 25$
 BIT #BIT9,SR1 ;IS THIS DR11K SELECTED FOR TEST?
 BNE 2$
 JMP 20$;NO-EXIT THIS TEST.

2$: MOV #JOB2,R0 ;PICK UP THIS JOB ADDRESS.
 MOV #BIT1!BIT3,(0) ;SET START AND MULTI-USER.
 BIS #BIT4!BIT8,(0) ;SELECT DIGITAL INPUT SINGLE CHAN.
3$: MOV #257,,2(0) ;SET WORD COUNT
 MOV #JOB2U,4(0) ;USW ADDR.
 CLRB 6(0) ;NO EXT.
 MOVB #201,7(0) ;MAKE TWO BUFFERS AVAIL.
 MOV #BUFF0,10(0) ;1ST BUFFER ADDR.
 MOV #BUFF1,14(0) ;2ND BUFFER ADDR.
 CLR 12(0) ;NO EXT.
 MOV #200,54(0) ;DELAY =200 TICKS.
 MOV #5-1,56(0) ;SET CHAN #=DR11K#-1.
 MOV #1,60(0) ;SAMPLING ONLY ONE CHANNEL.
 MOV #1,62(0) ;ON PULSF BETWEEN SAMPLE IS NO EXTERNAL TRIG.
 CLR 64(0) ;NO START/EVENT MAKR WORD
 CLR 66(0) ;NO START MASK
 CLR 70(0) ;NO EVENT MASK.

 MOV #JOB1,R1 ;OK-NOW LETS SET UP THE OUTPUT.
 MOV #BIT1!BIT3,(1) ;SET START AND MULTI USER.
 BIS #BIT4!BIT7!BIT8,(1) ;SELECT DIGITAL OUT.
 MOV #257,,2(1) ;WORD COUNT
 MOV #JOB1U,4(1) ;SET USW ADDR.

```

```

(1) 025576 105061 000006 CLRB 6(1) ;NO EXT.
(1) 025602 112761 000201 000007 MOVB #201,7(1)
(1) 025610 012761 060162 000010 MOV #BUFF2,10(1) ;SET BUFFER #1 ADDR.
(1) 025616 012761 061164 000014 MOV #BUFF3,14(1) ;SET BUFFER #2 ADDR.
(1) 025624 012761 060162 000020 MOV #BUFF2,20(1) ;THE THIRD OUTPUT IS NECESSARY
(1) ;TO COMPENSATE FOR SAMPLES LOST
(1) 025632 105061 000012 CLRB 12(1) ;NO EXT.
(1) 025636 012761 000200 000054 MOV #200,54(1) ;DELAY 1 TICK.
(1) 025644 012761 000004 000056 MOV #5-1,56(1) ;SET CHAN #= DR#-1
(1) 025652 012761 000001 000060 MOV #1,60(1) ;SAMPLE ONLY ONE CHAN.
(1) 025660 012761 000001 000062 MOV #1,62(1) ;ONE TICK BETWEEN SAMPLE
(1) 025666 005061 000064 CLR 64(1) ;NO START/EVENT
(1) 025672 005061 000066 CLR 66(1) ;NO START MASK.
(1) 025676 005061 000070 CLR 70(1) ;NO EVENT MASK.
(1)
(1) 025702 012700 125252 MOV #125252,R0 ;FILL OUTPUT BUFFER W/PATTERN
(1) 025706 012701 001002 MOV #514.,R1
(1) 025712 012702 060162 MOV #BUFF2,R2
(1) 025716 010022 4$: MOV R0,(2)+
(1) 025720 005100 COM R0
(1) 025722 005301 DEC R1
(1) 025724 001374 BNE 4$
(1)
(1) 025726 012700 056156 MOV #BUFF0,R0 ;CLEAR INPUT BUFFER
(1) 025732 012701 001002 MOV #514.,R1
(1) 025736 005020 5$: CLR (0)+
(1) 025740 005301 DEC R1
(1) 025742 001375 BNE 5$
(1)
(1) 025744 012737 000011 054776 MOV #11,JOB0 ;OK,NOW WE GOTTO STOP THE CLOCK
(1) 025752 012737 000000 055000 MOV #0,JOB0+2 ;SO THAT IT DOESN'T START UNTIL
(1) 025760 012737 000011 055350 MOV #11,JOB3 ;DRIIK PARAMETER SET UP,THEN AS
(1) 025766 012737 000503 055352 MOV #503,JOB3+2 ;JOB #4,WE START CLOCK.
(1) 025774 012737 177000 055354 MOV #177000,JOB3+4
(1) 026002 012737 000004 001124 MOV #4,$GDDAT ;FOUR JOBS.
(1) 026010 012737 000002 002014 MOV #2,$FUDGE ;/SET UP A FUDGE FACTOR TO
(1) ;/COMPENSATE FOR THE TWO KW11K JOBS
(1) 026016 004737 036126 JSR PC, FLASH ;/GO DO THEM!
(1)
(1) 026022 032737 001000 001564 BIT #BIT9,SR2 ;/CABLED TOGETHER?
(3) 026030 001417 BEQ TST45 ;;
(1) 026032 012700 001003 MOV #515.,R0 ;/NOW CHECK DATA IF CABLED.
(1) 026036 012701 060162 MOV #BUFF2,R1
(1) 026042 012702 056156 MOV #BUFF0,R2
(1) 026046 005300 6$: DEC R0
(3) 026050 001407 BEQ TST45 ;;
(1) 026052 022221 CMP (R2)+,(R1)+
(1) 026054 001774 BEQ 6$
(1) 026056 014137 001124 MOV -(R1),$GDDAT
(1) 026062 014237 001126 MOV -(R2),$BDDAT
(1) 026066 104012 ERROR 12
(1) 026070 20$:

```



4459  
 4470  
 4471  
 (3)  
 (4)  
 (4)  
 (4)  
 (4)  
 (4)  
 (4)  
 (4)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (5)  
 (3)  
 (2)  
 4472  
 4473  
 4474  
 4475  
 4476  
 4477  
 4478  
 4479  
 4480  
 4481  
 4482  
 4483  
 4484  
 4485  
 4486  
 4487  
 4488  
 4489  
 4490  
 4491  
 4492  
 4493  
 4494  
 4495  
 4496  
 4497  
 4498  
 4499  
 4500  
 4501  
 4502  
 4503  
 4504  
 4505  
 4506  
 4507

026070 000004  
 026072 032737 012001 001562  
 026100 001002  
 026102 000137 026426  
 026106 012737 177000 044026  
 026114 012777 044022 153354  
 026122 052777 000001 153336  
 026130 012700 054776  
 026134 005020  
 026136 022700 056140  
 026142 001374  
 026144 012700 054776  
 026150 012710 000002  
 026154 052710 001000  
 026160 122737 000104 001772  
 026166 001402  
 026170 052710 000010  
 026174 012760 000401 000002  
 026202 012760 055074 000004  
 026210 005070 000004  
 026214 012760 056156 000010  
 026222 012760 057160 000014  
 026230 012760 060162 000020  
 026236 112760 000002 000007  
 026244 122737 000104 001772  
 026252 001003  
 026254 152760 000200 000007  
 026262  
 026270 012760 000144 000054  
 026276 012760 000001 000062  
 026304 012760 000400 000056  
 026312 016037 000060 034554  
 026320 012737 000001 001124

```

*TEST 45 *TEST THE LPA SYSTEM USER MICRO-CODE'S ABILITY TO TAKE SAMPLE SINGLE JOB
*
*IN THIS TEST WE'LL VERIFY ANOLOG SAMPLING ABILITIES.
*DATA WILL BE TAKEN ON ALL CHANNELLS THROUGH ONE JOB.
*AT THE CONCLUSION, IF A WRAP AROUND MODULE WAS ON THE ANOLOG
*MODULE, A CHECK WILL BE MADE ON SPECIFIC CHANNELLS
*FOR SPECIFIC VALUES.
*
*THIS TEST WAS DESIGNED IN ORDER TO FIND A WILD LPA-11 SYSTEM FAULT.
*I DON'T REALLY EXPECT AN ERROR HERE UNLESS YOU'VE GOT A HEAVY FAULT.
*ARBITRATION LOGIC IS ONE BIG REASON WHY YOU COULD FAIL. ANOTHER
*IS THAT INTERRUPT VECTOR COULD BE WRONG. ALSO THE DMC MICRO-CODE MIGHT
*NOT BE EXECUTING PROPERLY.
*

TST45: SCOPE
 BIT #BIT12!BIT10!BIT0,SR1 ;IS THERE AN ANOLOG DEVICE?
 BNE 1$
 JMP 20$
 MOV #177000,KWT+4 ;RESTART CLOCK AT 1 MFGHZ RATE
 MOV #KWT,@LPADL
 BIS #BIT0,@LPCI
 MOV #JOB0,RO ;CLEAR OUT JOB AREA.
 CLR (0)+
 CMP #JOB7R,RO
 BNE 10$
 MOV #JOB0,RO ;GET SET TO DO SINGLE JOB RDA
 MOV #2,(0) ;SET "START" OP-CODE
 BIS #BIT9,(0) ;SET SEQUENTIAL CHAN SAMPLE
 CMPB #'D,$VERSN ;MULTI USER MICRO-CODE?
 BEQ 3$
 BIS #BIT3,(0) ;YES-TELL RDA MODE WORD - MULTI REQUEST
 MOV #257,2(0) ;SET BUFFER SIZE (WORD COUNT)
 MOV #JOB0U,4(0) ;SET USW WORD.
 CLR @4(0) ;CLEAR USW WORD.
 MOV #BUFF0,10(0) ;NOW SET BUFFER ADDRESS
 MOV #BUFF1,14(0)
 MOV #BUFF2,20(0)
 MOV #2,7(0) ;INDICATE HOW MANY BUFFERS USED.
 CMPB #'D,$VERSN ;IF THIS IS DEDICATED MODE,
 BNE 31$;WE MUST LET BUFFER OVERRUN BE NON-FATAL.
 BISB #BIT7,7(0) ;DEDICATED MODE IS JUST TO FAST FOR
 ;THE SIZE BUFFER AND STYLE WE USE HERE.
 MOV #100,54(0) ;SET DELAY BEFORE START
 MOV #16,60(0) ;SET # OF SAMPLES (CHANNELS)
 MOV #+1,62(0) ;# OF TICKS BETWEEN SAMPLES (DWELL)
 MOV #400,56(0) ;START CH 0, INC=1
 MOV 60(0),CHANU
 MOV #1,$GDDAT ;ONLY ONE JOB.

```

|      |        |        |        |        |     |                      |                                       |
|------|--------|--------|--------|--------|-----|----------------------|---------------------------------------|
| 4508 | 026326 | 004737 | 036126 |        | JSR | PC,FLASH             | :GO-DO IT.                            |
| 4509 | 026332 | 032737 | 002021 | 001564 | BIT | #BIT0.BIT4!BIT10,SR2 | :G5034 WRAPAROUND MODULE?             |
| 4510 | 026340 | 001432 |        |        | BEQ | TST46                | :                                     |
| 4511 |        |        |        |        |     |                      | :YES-LETS CHECK OUT THE RESULTS!      |
| 4512 | 026342 | 005237 | 034640 |        | INC | REPOR                |                                       |
| 4513 | 026346 | 012700 | 054776 |        | MOV | #JOB0,R0             |                                       |
| 4514 | 026352 | 012703 | 054716 |        | MOV | #LIST10,R3           | :GET LIST OF S/B RESULTS FOR 10 BITS. |
| 4515 | 026356 | 032737 | 002000 | 001562 | BIT | #BIT10,SR1           | :SEC IF 12 BIT A/D                    |
| 4516 | 026364 | 001002 |        |        | BNE | 5\$                  |                                       |
| 4517 | 026366 | 012703 | 054746 |        | MOV | #LIST12,R3           | :GET LIST OF 12 BIT A/D RESULTS.      |
| 4518 | 026372 | 012702 | 000004 |        | MOV | #4.,R2               | :CHECK 4 CHANS                        |
| 4519 | 026376 | 011337 | 034574 |        | MOV | (3),CHANS            | :PICK UP CHAN #                       |
| 4520 | 026402 | 012337 | 034576 |        | MOV | (3)+,CHANF           | :                                     |
| 4521 | 026406 | 012337 | 034614 |        | MOV | (3)+,AVEXP           |                                       |
| 4522 | 026412 | 012337 | 034612 |        | MOV | (3)+,TOLER           |                                       |
| 4523 | 026416 | 004737 | 034246 |        | JSR | PC,AVERR             |                                       |
| 4524 | 026422 | 005302 |        |        | DEC | R2                   |                                       |
| 4525 | 026424 | 001364 |        |        | BNE | 6\$                  |                                       |
| 4526 |        |        |        |        |     |                      |                                       |
| 4527 | 026426 |        |        |        |     |                      | 20\$:                                 |

```

4529
4544
4545
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(5)
(3)
(2) 026426 000004

```

```

*TEST 46 *TEST THE LPA SYSTEM USER MICRO-CODE'S ABILITY TO TAKE RANDOM SAMPLES SI
*
*IN THIS TEST WE'LL VERIFY ANOLOG SAMPLING ABILITIES.
*DATA WILL BE TAKEN AN ALL CHANNELLS THROUGH ONE JOB.
*
*THE DIFFERENCE BETWEEN THIS TEST AND THE ONE BEFORE IF IS THAT
*DATA IS TAKEN IN RANDOM CHANNELL MODE INSTEAD OF SEQUENCAL.
*
*AT THE CONCLUSION, IF A WRAP AROUND MODULE WAS ON THE ANOLOG
*MODULE, A CHECK WILL BE MADE ON SPECIFIC CHANNELLS
*FOR SPECIFIC VALUES.
*
*THIS TEST WAS DESIGNED IN ORDER TO FIND A WILD LPA-11 SYSTEM FAULT.
*I DON'T REALLY EXPECT AN ERROR HERE UNLESS YOU'VE GOT A HEAVY FAULT.
*ARBITRATION LOGIC IS ONE BIG REASON WHY YOU COULD FAIL. ANOTHER
*IS THAT INTERRUPT VECTOR COULD BE WRONG. ALSO THE DMC MICRO-CODE MIGHT
*NOT BE EXECUTING PROPERLY.

```

```

TST46: SCOPE
BIT #BIT12!BIT10!BIT0,SR1 ;IS THERE AN ANOLOG DEVICE?
BNE 1$
JMP 20$
1$: MOV #177000,KWT+4 ;RESTART CLOCK AT 1 MEGHZ RATE
MOV #KWT,@LPADL
BIS #BIT0,@LPCI
MOV #JOB0,RO ;CLEAR OUT JOB AREA.
10$: CLR (0)+
CMP #JOB7R,RO
BNE 10$

MOV #JOB0,RO ;GET SET TO DO SINGLE JOB RDA
MOV #2,(0)

2$: CMPB #'D,$VERSN ;MULTI USER MICRO-CODE?
BEQ 3$
BIS #BIT3,(0) ;YES-TELL RDA MODE WORD.
3$: MOV #257.,2(0) ;SET BUFFER SIZE.
MOV #JOB0U,4(0) ;SET USW WORD.
CLR @4(0) ;CLEAR USW WORD.
MOV #BUFF0,10(0) ;NOW SET BUFFER ADDRESS
MOV #BUFF1,14(0)
MOV #BUFF2,20(0)
MOVB #2,7(0) ;INDICATE HOW MANY BUFFERS USED.
CMPB #'D,$VERSN ;IF THIS IS DEDICATED MODE,
BNE 31$;WE MUST LET BUFFER OVERRUN BE NON-FATAL.
BISB #BIT7,7(0) ;DEDICATED MODE IS JUST TO FAST FOR
 ;THE SIZE BUFFER AND STYLE WE USE HERE.

31$: MOV #100.,54(0) ;SET DELAY BEFORE START
MOV #7.,60(0) ;SET # OF CHANNELS

```

|      |        |        |        |        |      |     |                      |   |                                      |
|------|--------|--------|--------|--------|------|-----|----------------------|---|--------------------------------------|
| 4578 | 026630 | 012760 | 000002 | 000062 | 48:  | MOV | #+2,62(0)            | ; | # OF TICKS BETWEEN SAMPLES           |
| 4579 | 026636 | 012760 | 055076 | 000050 |      | MOV | #JOBOR,50(0)         | ; | SET ADDR OF RANDOM CHANNEL LIST      |
| 4580 | 026644 | 016037 | 000060 | 034554 |      | MOV | 60(0),CHANU          |   |                                      |
| 4581 | 026652 | 012700 | 055076 |        |      | MOV | #JOBOR,RO            | ; | PICK UP POINT TO RANDOM LIST.        |
| 4582 | 026656 | 012720 | 000000 |        |      | MOV | #0,(0)+              | ; | CHAN 0 FIRST                         |
| 4583 | 026662 | 012720 | 000003 |        |      | MOV | #3,(0)+              | ; | NEXT CHAN 3                          |
| 4584 | 026666 | 012720 | 000002 |        |      | MOV | #2,(0)+              | ; | CHAN 2 NEXT                          |
| 4585 | 026672 | 012720 | 000003 |        |      | MOV | #3,(0)+              | ; | CHAN 3                               |
| 4586 | 026676 | 012720 | 000004 |        |      | MOV | #4,(0)+              | ; | NEXT CHAN 4                          |
| 4587 | 026702 | 012720 | 000015 |        |      | MOV | #15,(0)+             | ; | CHAN 15                              |
| 4588 | 026706 | 012710 | 000003 |        |      | MOV | #3,(0)               | ; | CHAN 3 NEXT.                         |
| 4589 | 026712 | 052720 | 140000 |        |      | BIS | #BIT15!BIT14,(0)+    | ; | NOP AND END OF LIST.                 |
| 4590 | 026716 | 012737 | 000001 | 001124 |      | MOV | #1,\$GDDAT           | ; | ONLY ONE JOB.                        |
| 4591 | 026724 | 004737 | 036126 |        |      | JSR | PC,FLASH             | ; | GO-DO IT.                            |
| 4592 | 026730 | 032737 | 002021 | 001564 |      | BIT | #BIT0!BIT4!BIT10,SR2 | ; | G5034 WRAPAROUND MODULE?             |
| 4593 | 026736 | 001432 |        |        |      | BEQ | TST47                | ; |                                      |
| 4594 |        |        |        |        |      |     |                      | ; | YES-LETS CHECK OUT THE RESULTS!      |
| 4595 | 026740 | 005237 | 034640 |        |      | INC | REPOR                |   |                                      |
| 4596 | 026744 | 012700 | 054776 |        |      | MOV | #JOB0,RO             |   |                                      |
| 4597 | 026750 | 012703 | 054716 |        |      | MOV | #LIST10,R3           | ; | GET LIST OF S/B RESULTS FOR 10 BITS. |
| 4598 | 026754 | 032737 | 002000 | 001562 |      | BIT | #BIT10,SR1           | ; | SEC IF 12 BIT A/D                    |
| 4599 | 026762 | 001002 |        |        |      | BNE | 58                   |   |                                      |
| 4600 | 026764 | 012703 | 054746 |        |      | MOV | #LIST12,R3           | ; | GET LIST OF 12 BIT A/D RESULTS.      |
| 4601 | 026770 | 012702 | 000004 |        | 58:  | MOV | #4.,R2               | ; | CHECK 4 CHANS                        |
| 4602 | 026774 | 011337 | 034574 |        | 68:  | MOV | (3),CHANS            | ; | PICK UP CHAN #                       |
| 4603 | 027000 | 012337 | 034576 |        |      | MOV | (3)+,CHANF           | ; |                                      |
| 4604 | 027004 | 012337 | 034614 |        |      | MOV | (3)+,AVEXP           | ; |                                      |
| 4605 | 027010 | 012337 | 034612 |        |      | MOV | (3)+,TOLER           | ; |                                      |
| 4606 | 027014 | 004737 | 034246 |        |      | JSR | PC,AVERR             |   |                                      |
| 4607 | 027020 | 005302 |        |        |      | DEC | R2                   |   |                                      |
| 4608 | 027022 | 001364 |        |        |      | BNE | 68                   |   |                                      |
| 4609 |        |        |        |        |      |     |                      |   |                                      |
| 4610 | 027024 |        |        |        | 208: |     |                      |   |                                      |

4612  
4639  
4640  
(3)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(4)  
(3)  
(2)  
(1)  
4641  
4642  
4643  
4644  
4645  
4646  
4647  
4648  
4649  
4650  
4651  
4652  
4653  
4654  
4655  
4656  
4657  
4658  
4659  
4660  
4661  
4662  
4663  
4664  
4665

```

027024 000004
(1) 027026 012737 000001 001160
4641 027034 122737 000115 001772
027042 001413
4643 027044 000137 030256
4644 027050 012737 170000 044026
4645 027056 012777 044022 152412
4646 027064 052777 000001 152374
4647 027072 012700 054776
4648 027076 012703 055114
4649 027102 160003
4650 027104 005020
4651 027106 020027 056140
4652 027112 001374
4653 027114 005037 001124
4654 027120 012700 054776
4655 027124 160300
4656 027126 032737 012001 001562
4657 027134 001441
4658 027136 060300
4659 027140 012710 000012
4660 027144 052710 000400
4661 027150 012760 000512 000002
4662 027156 012760 055074 000004
4663 027164 005070 000004
4664 027170 152760 000200 000007
4665 027176 012760 056156 000010

```

```

:*****
*TEST 47 *TEST THAT MULTI JOBS CAN BE STARTED AND RUN
*IN THIS TEST, WE'LL ATTEMPT TO STARTUP UP TO EIGHT JOBS WITH
*MULTIUSER MICRO CODE. THIS TEST WILL ONLY BE EXECUTED IF
*MULTIUSER MICRO-CODE HAS BEEN LOADED INTO THE KMC.
*LESS THAN EIGHT JOBS MAY BE RUN, DEPENDING UPON THE
*PARTICULAR CONFIGURATION OF YOUR SYSTEM.
*BELOW IS A LIST OF JOBS AND HOW THEY ARE SET UP.
*
* JOB#0 AD11K#1 (IF CONFIGURED) OR LPS A/D OR AR11
* JOB#1 DR11K#1 (IF CONFIGURED) OR LPS I/O OR AR11 I/O
* JOB#2 AA1K (IF CONFIGURED) OR LPS OR AR11
* JOB#3 AD11K#2 (IF CONFIGURED)
* JOB#4 DR11K#2 (IF CONFIGURED)
* JOB#5 DR11K#3 (IF CONFIGURED)
* JOB#6 DR11K#4 (IF CONFIGURED)
* JOB#7 DR11K#5 (IF CONFIGURED)
*
*WHEN WE SET UP THESE JOBS, WE'LL SET THEM UP TO START ROUGHLY
*AT THE SAME TIME BY VARYING THE START UP DELAY.
*
*WHAT WE'RE GONNA LOOK FOR AN ERROR INDICATION
*IS FOR ERRORS ON JOBS, OR SOME JOB TO FAIL TO FINISH.
*
:*****

```

```

:*****
TST47: SCOPE
 MOV #1,$TIMES ;;DO 1 ITERATION
 CMPB #'M,$VERSN
 BEQ 1$
 JMP 20$
 MOV #170000,$KWT+4 ;RESTART CLOCK AT 1 MEGHZ RATE
 MOV #KWT,@LPC1
 BIS #BIT0,@LPCI
1$: MOV #JOB0,$R0
 MOV #JOB1,$R3 ;GET DIF IN ADDR. BETWEEN JOBS.
 SUB $R0,$R3
2$: CLR ($R0)+
 CMP $R0,#JOB7R
 BNE 2$
 CLR $GDDAT
 MOV #JOB0,$R0 ;GET JOB POINTER
 SUB $R3,$R0 ;BACK UP POINTER
 BIT #BIT0!BIT10!BIT12,$SR1 ;ANY A/D'S?
 BEQ 3$;NO-EXIT A/D'S.
 ADD $R3,$R0 ;ADJUST POINTER
 MOV #12,$(R0) ;MODE WORD-START A/D.
 BIS #BIT8,$(R0) ;SINGLE CHAN.
 MOV #512,$2(R0) ;TAKE 512 SAMPLES
 MOV #JOB0U,4(R0)
 CLR @4(R0) ;CLEAR USW
 BISB #BIT7,7(R0) ;BUFFER OVERRUN NOT FATAL, ONE BUFFER.
 MOV #BUFF0,10(R0);SET BUFFER ADDR.

```

```

4666 027204 012760 000010 000054 MOV #10,54(R0) ;SET DELAY BEFORE START.
4667 027212 012760 000016 000060 MOV #16,60(R0) ;SAMPLE 14 CHANNELS
4668 027220 012760 000010 000062 MOV #10,62(R0) ;SET SAMPLE RATE.
4669 027226 005237 001124 INC $GDDAT
4670 027232 012760 000400 000056 MOV #400,56(R0) ;START CH0, INC=1
4671 027240 3$:
4702
4703
(1)
(1)
(1) 027240 032737 100004 001562 BIT #BIT2!BIT15,SR1 ;/THIS DR11K #1 OR LPS I/O SEL?
(1) 027246 001442 BEQ 64$
(1) 027250 060300 ADD R3,R0 ;/UPDAT JOB STORAGE AREA POINTER.
(1) 027252 012710 000432 MOV #432,(R0) ;/SET I/O START, MULTI-USER, INPUT SINGLE CHAN.
(1) 027256 012760 000256 000002 MOV #256,2(R0) ;/WORD COUNT
(1) 027264 012760 055212 000004 MOV #JOB1U,4(0)
(1) 027272 012770 000000 000004 MOV #0,24(R0) ;/CLEAR USW
(1) 027300 012760 000000 000006 MOV #0,6(R0)
(1) 027306 012760 060162 000010 MOV #BUFF2,10(R0) ;/SET BUFFER ADDR.
(1) 027314 005060 000012 CLR 12(R0)
(1) 027320 005237 001124 INC $GDDAT ;RECORD THIS JOB
(1) 027324 012760 000100 000054 MOV #100,54(R0) ;DELAY BEFORE START.
(1) 027332 012760 000000 000056 MOV #1-1,56(R0) ;/SELECT CHAN #.
(1) 027340 012760 000001 000060 MOV #1,60(R0) ;/SAMPLING ONLY ONE CHAN.
(1) 027346 012760 000001 000062 MOV #1,62(R0) ;/SAMPLE RATE
(1)
(1) 027354 64$:
4704
4705 027354 032737 042010 001562 BIT #BIT3!BIT10!BIT14,SR1 ;D/A OUTPUT SELECTED?
4706 027362 001440 BEQ 5$;NO-NEXT SET-UP
4707 027364 005237 001124 INC $GDDAT
4708
4709
4710
4711
4712 027370 060300 ADD R3,R0 ;ALTHOUGH WE'RE GONNA WORK
4713 027372 012710 001212 MOV #BIT1!BIT3!BIT7!BIT9,(0) ;WITH D/A, NO OUTPUT
4714 027376 012760 000256 000002 MOV #256,2(0) ;WILL RESULT. D/A THROWN IN
4715 027404 012760 055330 000004 MOV #JOB2U,4(0) ;JUST TO TRY AND CONFUSE LPA.
4716 027412 012770 000000 000004 MOV #0,24(0) ;UPDATE JOB STORAGE AREA POINTER
4717 027420 012760 000000 000006 MOV #0,6(0) ;START, D/A, SEQ. CHAN.
4718 027426 012760 061164 000010 MOV #BUFF3,10(0) ;SET XFERR COUNT
4719 027434 012760 000100 000054 MOV #100,54(0) ;CLR USW.
4720 027442 012760 000000 000056 MOV #0,56(0) ;BUFFER ADDR.
4721 027450 012760 000001 000060 MOV #1,60(0) ;DELAY BEFORE START.
4722 027456 012760 000001 000062 MOV #1,62(0) ;/SAMPLE CHAN 0
4723
4724 027464 032737 000020 001562 BIT #BIT4,SR1 ;/SAMPLE ONLY ONE CHAN
4725 027472 001437 BEQ 5$;/SAMPLE RATE
4726 027474 060300 ADD R3,R0 ;SECOND A/D?
4727 027476 012710 000012 MOV #12,(0) ;UPDATE JOB STORAGE ARE POINTER.
4728 027502 012760 000512 000002 MOV #512,2(0) ;MODE WORD-START A/D.
4729 027510 012760 055446 000004 MOV #JOB3U,4(0) ;TAKE 512 SAMPLES
4730 027516 005070 000004 CLR 24(0) ;CLEAR USW
4731 027522 152760 000200 000007 BISB #BIT7,7(0) ;BUFFER OVERRUN NOT FATAL, ONE BUFFER.
4732 027530 012760 056156 000010 MOV #BUFF0,10(0) ;SET BUFFER ADDR.

```

```

4733 027536 012760 000010 000054 MOV #10,54(0) ;SET DELAY BEFORE START.
4734 027544 012760 000420 000056 MOV #420,56(0)
4735 027552 012760 000016 000060 MOV #16,60(0) ;SAMPLE 14 CHANNELS.
4736 027560 012760 000010 000062 MOV #10,62(0) ;SET SAMPLE RATE.
4737 027566 005237 001124 INC $GDDAT
4738
4739 027572 68:
4740
(1)
(1)
(1) 027572 032737 000040 001562 BIT #BIT5,SR1 ;/THIS DR11K #2 SELECTED?
(1) 027600 001442 BEQ 65$
(1) 027602 060300 ADD R3,R0 ;/UPDAT JOB STORAGE AREA POINTER.
(1) 027604 012710 000432 MOV #432,(R0) ;/SET I/O START, MULTI-USER, INPUT SINGLE CHAN.
(1) 027610 012760 000256 000002 MOV #256,2(R0) ;/WORD COUNT
(1) 027616 012760 055564 000004 MOV #JOB4U,4(0)
(1) 027624 012770 000000 000004 MOV #0,24(R0) ;/CLEAR USW
(1) 027632 012760 000000 000006 MOV #0,6(R0)
(1) 027640 012760 060162 000010 MOV #BUFF2,10(R0) ;/SET BUFFER ADDR.
(1) 027646 005060 000012 CLR 12(R0)
(1) 027652 005237 001124 INC $GDDAT ;RECORD THIS JOB
(1) 027656 012760 000100 000054 MOV #100,54(R0) ;DELAY BEFORE START.
(1) 027664 012760 000001 000056 MOV #2-1,56(R0) ;/SELECT CHAN #.
(1) 027672 012760 000001 000060 MOV #1,60(R0) ;/SAMPLING ONLY ONE CHAN.
(1) 027700 012760 000001 000062 MOV #1,62(R0) ;/SAMPLE RATE
(1)
(1) 027706 65$:
4741
(1)
(1)
(1) 027706 032737 000200 001562 BIT #BIT7,SR1 ;/THIS DR11K #3 SELECTED?
(1) 027714 001442 BEQ 66$
(1) 027716 060300 ADD R3,R0 ;/UPDAT JOB STORAGE AREA POINTER.
(1) 027720 012710 000432 MOV #432,(R0) ;/SET I/O START, MULTI-USER, INPUT SINGLE CHAN.
(1) 027724 012760 000256 000002 MOV #256,2(R0) ;/WORD COUNT
(1) 027732 012760 055702 000004 MOV #JOB5U,4(0)
(1) 027740 012770 000000 000004 MOV #0,24(R0) ;/CLEAR USW
(1) 027746 012760 000000 000006 MOV #0,6(R0)
(1) 027754 012760 060162 000010 MOV #BUFF2,10(R0) ;/SET BUFFER ADDR.
(1) 027762 005060 000012 CLR 12(R0)
(1) 027766 005237 001124 INC $GDDAT ;RECORD THIS JOB
(1) 027772 012760 000100 000054 MOV #100,54(R0) ;DELAY BEFORE START.
(1) 030000 012760 000002 000056 MOV #3-1,56(R0) ;/SELECT CHAN #.
(1) 030006 012760 000001 000060 MOV #1,60(R0) ;/SAMPLING ONLY ONE CHAN.
(1) 030014 012760 000001 000062 MOV #1,62(R0) ;/SAMPLE RATE
(1)
(1) 030022 66$:
4742
(1)
(1)
(1) 030022 032737 000400 001562 BIT #BIT8,SR1 ;/THIS DR11K #4 SELECTED?
(1) 030030 001442 BEQ 67$
(1) 030032 060300 ADD R3,R0 ;/UPDAT JOB STORAGE AREA POINTER.
(1) 030034 012710 000432 MOV #432,(R0) ;/SET I/O START, MULTI-USER, INPUT SINGLE CHAN.
(1) 030040 012760 000256 000002 MOV #256,2(R0) ;/WORD COUNT
(1) 030046 012760 056020 000004 MOV #JOB6U,4(0)

```

```

(1) 030054 012770 000000 000004 MOV #0,24(R0) ;/CLEAR USW
(1) 030062 012760 000000 000006 MOV #0,6(R0) ;/SET BUFFER ADDR.
(1) 030070 012760 060162 000010 MOV #BUFF2,10(R0)
(1) 030076 005060 000012 CLR 12(R0)
(1) 030102 005237 001124 INC $GDDAT ;RECORD THIS JOB
(1) 030106 012760 000100 000054 MOV #100,54(R0) ;DELAY BEFORE START.
(1) 030114 012760 000003 000056 MOV #4-1,56(R0) ;/SELECT CHAN #.
(1) 030122 012760 000001 000060 MOV #1,60(R0) ;/SAMPLING ONLY ONE CHAN.
(1) 030130 012760 000001 000062 MOV #1,62(R0) ;/SAMPLE RATE
(1)
(1) 030136 67$:
4743
(1)
(1)
(1) 030136 032737 001000 001562 BIT #BIT9,SR1 ;/THIS DR11K #5 SELECTED?
(1) 030144 001442 BEQ 68$
(1) 030146 060300 ADD R3,R0 ;/UPDAT JOB STORAGE AREA POINTER.
(1) 030150 012710 000432 MOV #432,(R0) ;/SET I/O START, MULTI-USER, INPUT SINGLE CHAN.
(1) 030154 012760 000256 000002 MOV #256,2(R0) ;/WORD COUNT
(1) 030162 012760 056136 000004 MOV #JOB7U,4(0)
(1) 030170 012770 000000 000004 MOV #0,24(R0) ;/CLEAR USW
(1) 030176 012760 000000 000006 MOV #0,6(R0)
(1) 030204 012760 060162 000010 MOV #BUFF2,10(R0) ;/SET BUFFER ADDR.
(1) 030212 005060 000012 CLR 12(R0)
(1) 030216 005237 001124 INC $GDDAT ;RECORD THIS JOB
(1) 030222 012760 000100 000054 MOV #100,54(R0) ;DELAY BEFORE START.
(1) 030230 012760 000004 000056 MOV #5-1,56(R0) ;/SELECT CHAN #.
(1) 030236 012760 000001 000060 MOV #1,60(R0) ;/SAMPLING ONLY ONE CHAN.
(1) 030244 012760 000001 000062 MOV #1,62(R0) ;/SAMPLE RATE
(1)
(1) 030252 68$:
4744
4745 030252 004737 036126 JSR PC,FLASH ;FLASH WILL START JOBS
4746 ;AND REPORT ERRORS,WAIT TILL DONE.
4747 030256 20$:
4748
4761
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(2) 030256 000004
4762
4763 030260 122737 000104 001772 CMPB #'D,$VERSN ;RUNNING DEDICATED MODE U-CODE?
4764 030266 001014 BNE ETRF ;NO-THEN EXIT.
4765 030270 032737 000020 001564 BIT #BIT4,SR2 ;SECOND A/D WITH WRAPARROUND MODULE?
4766 030276 001410 BEQ ETRF ;NO-EXIT.
4767 030300 012737 030320 000004 MOV #ETDF,2#4 ;SET FOR TIMEOUT IF NOT ENOUGH CORE.

```

```

*TEST 50 *HIGH SPEED A/D SAMPLE TEST (DEDICATED MODE,SPECIAL TEST)
*
*
* IN THIS TEST WE WILL TAKE A/D SAMPLES AS FAST AS WE CAN.
* THIS TEST REQUIRES MORE MEMORY THAN THE REST SO UPON ENTRY
* WE WILL CHECK TO MAKE SURE THAT WE HAVE 20K OF MEMORY.
* ALSO DEDICATED MODE MICRO-CODE MUST BE IN THE KMC. LAST
* WE MUST HAVE TWO A/D S ON THE I/O BUSS,BOTH HAVEING A G5036
* WRAP-ARROUND MODULE INSTALLED AND INFORMED TO PROGRAM VIA SR2.
* IF ALL REQUIREMENTS ARE MENT,WE WILL DO THIS TEST AND TAKE
* SAMPLES AT A RATE OF APP. 150 KILOHERTZ.

TST50: SCOPE

```



```

4768 030306 005037 000006 CLR @#6
4769 030312 005737 070000 TST @#70000 ;ADDR. MEMORY.
4770 030316 000407 BR 1$;GO TO START OF TEST.IF NOT ENOUGH MEMORY
4771 ;WE WOULD HAVE TRAPPED.
4772
4773
4774 030320 012737 000006 000004 ETDF=. MOV #6,@#4 ;RESTORE LOC 4
4775 030326 012706 001100 MOV #STACK,SP
4776 030332 000137 030634 JMP 20$
4777
4778 030336 012737 000006 000004 1$: MOV #6,@#4
4779 030344 012737 000503 044024 MOV #503,KWT+2 ;START CLOCK AT RATE 1MHZ.
4780 030352 012737 177763 044026 MOV #-13.,KWT+4 ;INTR. EVERY 13 USEC.
4781 030360 012700 054776 MOV #JOB0,R0
4782 030364 012710 004442 MOV #BIT11!BIT8!BIT5!BIT1,(0) ;START WORD.
4783 030370 012760 001750 000002 MOV #1000.,2(0) ;BUFFER SIZE
4784 030376 012760 055074 000004 MOV #JOB0U,4(0) ;USW ARR.
4785 030404 005070 000004 CLR @4(0)
4786 030410 012760 070000 000010 MOV #70000,10(0) ;SET UP BUFFER ADDRS.
4787 030416 012760 070000 000014 MOV #70000,14(0)
4788 030424 062760 001750 000014 ADD #1000.,14(0)
4789 030432 012760 070000 000020 MOV #70000,20(0)
4790 030440 062760 003720 000020 ADD #2000.,20(0)
4791 030446 112760 000002 000007 MOV #2,7(0)
4792 030454 112760 000144 000054 MOV #100.,54(0) ;DELAY BEFORE START.
4793 030462 112760 000001 000060 MOV #1.,60(0) ;NUMBER OF CHANS.
4794 030470 112737 000002 034554 MOV #2.,CHANU
4795 030476 012760 000001 000062 MOV #1,62(0) ;TICKS BETWEEN SAMPLES.
4796 030504 012760 000400 000056 MOV #400,56(0) ;START CH0,INC=1
4797 030512 012737 000001 001124 MOV #1,$GDDAT ;ONE JOB.
4798 030520 004737 036126 JSR PC,FLASH ;DO IT.
4799
4800 030524 005237 034640 INC REPOR ;LOOK AT AD11K #1
4801 030530 012703 054746 MOV #LIST12,R3
4802 030534 011337 034574 2$: MOV (3),CHANS
4803 030540 012337 034576 MOV (3)+,CHANF
4804 030544 012337 034614 MOV (3)+,AVEXP
4805 030550 012337 034612 MOV (3)+,TOLER
4806 030554 012701 070000 MOV #70000,R1
4807 030560 013737 034574 034566 MOV CHANS,CHAN
4808 030566 004737 034260 JSR PC,SPEP
4809
4810 030572 012703 054746 MOV #LIST12,R3 ;LOOK AT AD11K #2
4811 030576 011337 034574 MOV (3),CHANS
4812 030602 012337 034576 MOV (3)+,CHANF
4813 030606 012337 034614 MOV (3)+,AVEXP
4814 030612 012337 034612 MOV (3)+,TOLER
4815 030616 012701 070002 MOV #70002,R1
4816 030622 013737 034574 034566 MOV CHANS,CHAN
4817 030630 004737 034260 JSR PC,SPEP
4818
4819 030634 20$:
4820
4821 030634 ETEST:
4822
(3) ;*TEST 51 END OF TESTS

```

```

(3)
(2) 030634 000004
4837
(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1) 030636
(2) 030636 000240
(1) 030640 005037 001102
(1) 030644 005037 001160
(1) 030650 005237 001202
(1) 030654 042737 100000 001202
(1) 030662 005327
(1) 030664 000001
(1) 030666 003064
(1) 030670 012737
(1) 030672 000001
(1) 030674 030664
(2)
(2)
(2) 030676 122737 000104 001772
(2) 030704 001006
(3) 030706 104401 030714
(3) 030712 000402
(3)
(3) 030720
64$:
(2) 030720 000405
(2) 030722
1$:
(3) 030722 104401 030730
(3) 030726 000402
(3)
(3) 030734
66$:
(2) 030734
2$:
(3) 030734 104401 030742
(3) 030740 000405
(3)
(3) 030754
68$:
(3) 030754 013746 001202
(3)
(3) 030760 104405
(3) 030762 104401 030770
(3) 030766 000411
(3)
(3) 031012
70$:
(3) 031012 013746 001776
(3) 031016 104405
(1) 031020 013700 000042
(1) 031024 001405
(1) 031026 000005
(1) 031030 004710
(1) 031032 000240
(1) 031034 000240
(1) 031036 000240

::*****
TST51: SCOPE
.SBTTL END OF PASS ROUTINE
::*****
*INCREMENT THE PASS NUMBER ($PASS)
*IF THERES A MONITOR GO TO IT
*IF THERE ISN'T JUMP TO RSTART

$EOP:
NOP
CLR $TSTNM ;;ZERO THE TEST NUMBER
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS ;;INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?
$EOPCT: .WORD 1
BGT $DOAGN ;;YES
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
$ENDCT: .WORD 1
$EOPCT

;/-ENDPAS-
CMPB #'D,$VERSN
BNE 1$
TYPE ,65$;;TYPE ASCIZ STRING
BR 64$;;GET OVER THE ASCIZ
65$: .ASCIZ <200>#D-#
64$:
BR 2$
1$:
TYPE ,67$;;TYPE ASCIZ STRING
BR 66$;;GET OVER THE ASCIZ
67$: .ASCIZ <200>#M-#
66$:
2$:
TYPE ,69$;;TYPE ASCIZ STRING
BR 68$;;GET OVER THE ASCIZ
69$: .ASCIZ #END PASS #
68$:
MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
;;TYPE PASS NUMBER.
;;GO TYPE--DECIMAL ASCII WITH SIGN
TYPDS
TYPE ,71$;;TYPE ASCIZ STRING
BR 70$;;GET OVER THE ASCIZ
71$: .ASCIZ # ; TOTAL ERRORS #
70$:
MOV ERCNT,-(SP) ;;SAVE ERCNT FOR TYPEOUT
;;GO TYPE--DECIMAL ASCII WITH SIGN
$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
BEQ $DOAGN ;;BRANCH IF NO MONITOR
RESET ;;CLEAR THE WORLD
$ENDAD: JSR PC,(R0) ;;GO TO MONITOR
NOP ;;SAVE ROOM
NOP ;;FOR
NOP ;;ACT11

```

```

(1) 031040 $DOAGN:
(1) 031040 000137 JMP @ (PC)+ ;;RETURN
(1) 031042 003502 $RTNAD: .WORD RSTART
(1) 031044 377 377 000 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
(1) 031050 .EVEN

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

(1)
(2) *****
(1) *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1) *OCTAL (ASCII) NUMBER AND TYPE IT.
(1) *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1) *CALL:
(1) * MOV NUM,-(SP) ;;NUMBER TO BE TYPED
(1) * TYPOS ;;CALL FOR TYPEOUT
(1) * .BYTE N ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1) * .BYTE M ;;M=1 OR 0
(1) * ;;1=TYPE LEADING ZEROS
(1) * ;;0=SUPPRESS LEADING ZEROS
(1) *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1) *$TYPOS OR $TYPOC
(1) *CALL:
(1) * MOV NUM,-(SP) ;;NUMBER TO BE TYPED
(1) * TYPON ;;CALL FOR TYPEOUT
(1) *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1) *CALL:
(1) * MOV NUM,-(SP) ;;NUMBER TO BE TYPED
(1) * TYPOC ;;CALL FOR TYPEOUT
(1) 031050 017646 000000 $TYPOS: MOV @ (SP),-(SP) ;;PICKUP THE MODE
(1) 031054 116637 000001 031273 MOVBL 1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
(1) 031062 112637 031275 MOVBL (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
(1) 031066 062716 000002 ADD #2,(SP) ;;ADJUST RETURN ADDRESS
(1) 031072 000406 BR $TYPON
(1) 031074 112737 000001 031273 $TYPOC: MOVBL #1,$OFILL ;;SET THE ZERO FILL SWITCH
(1) 031102 112737 000006 031275 MOVBL #6,$OMODE+1 ;;SET FOR SIX(6) DIGITS
(1) 031110 112737 000005 031272 $TYPON: MOVBL #5,$OCNT ;;SET THE ITERATION COUNT
(1) 031116 010346 MOV R3,-(SP) ;;SAVE R3
(1) 031120 010446 MOV R4,-(SP) ;;SAVE R4
(1) 031122 010546 MOV R5,-(SP) ;;SAVE R5
(1) 031124 113704 031275 MOVBL $OMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
(1) 031130 005404 NEG R4
(1) 031132 062704 000006 ADD #6,R4 ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 031136 110437 031274 MOVBL R4,$OMODE ;;SAVE IT FOR USE
(1) 031142 113704 031273 MOVBL $OFILL,R4 ;;GET THE ZERO FILL SWITCH
(1) 031146 016605 000012 MOV 12(SP),R5 ;;PICKUP THE INPUT NUMBER
(1) 031152 005003 CLR R3 ;;CLEAR THE OUTPUT WORD
(1) 031154 006105 1$: ROL R5 ;;ROTATE MSB INTO 'C'
(1) 031156 000404 BR 3$;;GO DO MSB
(1) 031160 006105 2$: ROL R5 ;;FORM THIS DIGIT
(1) 031162 006105 ROL R5
(1) 031164 006105 ROL R5
(1) 031166 0105C3 MOV R5,R3

```

```

(1) 031170 006103 3$: ROL R3 ;;GET LSB OF THIS DIGIT
(1) 031172 105337 031274 DECB $OMODE ;;TYPE THIS DIGIT?
(1) 031176 100016 BPL 7$;;BR IF NO
(1) 031200 042703 177770 BIC #177770,R3 ;;GET RID OF JUNK
(1) 031204 001002 BNE 4$;;TEST FOR 0
(1) 031206 005704 TST R4 ;;SUPPRESS THIS 0?
(1) 031210 001403 BEQ 5$;;BR IF YES
(1) 031212 005204 4$: INC R4 ;;DON'T SUPPRESS ANYMORE 0'S
(1) 031214 052703 000060 BIS #'0,R3 ;;MAKE THIS DIGIT ASCII
(1) 031220 052703 000040 5$: BIS #' ,R3 ;;MAKE ASCII IF NOT ALREADY
(1) 031224 110337 031270 MOV R3,8$;;SAVE FOR TYPING
(1) 031230 104401 031270 TYPE ,8$;;GO TYPE THIS DIGIT
(1) 031234 105337 031272 7$: DECB $OCNT ;;COUNT BY 1
(1) 031240 003347 BGT 2$;;BR IF MORE TO DO
(1) 031242 002402 BLT 6$;;BR IF DONE
(1) 031244 005204 INC R4 ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 031246 000744 BR 2$;;GO DO THE LAST DIGIT
(1) 031250 012605 6$: MOV (SP)+,R5 ;;RESTORE R5
(1) 031252 012604 MOV (SP)+,R4 ;;RESTORE R4
(1) 031254 012603 MOV (SP)+,R3 ;;RESTORE R3
(1) 031256 016666 000002 000004 MOV 2(SP),4(SP) ;;SET THE STACK FOR RETURNING
(1) 031264 012616 MOV (SP)+,(SP)
(1) 031266 000002 RTI ;;RETURN
(1) 031270 000 8$: .BYTE 0 ;;STORAGE FOR ASCII DIGIT
(1) 031271 000 .BYTE 0 ;;TERMINATOR FOR TYPE ROUTINE
(1) 031272 000 $OCNT: .BYTE 0 ;;OCTAL DIGIT COUNTER
(1) 031273 000 $OFILL: .BYTE 0 ;;ZERO FILL SWITCH
(1) 031274 000000 $OMODE: .WORD 0 ;;NUMBER OF DIGITS TO TYPE
. SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

(1)
(2) ;;*****
(1) ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1) ;;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1) ;;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1) ;;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1) ;;*REPLACED WITH SPACES.
(1) ;;*CALL:
(1) ;;* MOV NUM,-(SP) ;;PUT THE BINARY NUMBER ON THE STACK
(1) ;;* TYPDS ;;GO TO THE ROUTINE
(1)
(1) $TYPDS:
(3) 031276 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 031300 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(3) 031302 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
(3) 031304 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
(3) 031306 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
(1) 031310 012746 020200 MOV #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
(1) 031314 016605 000020 MOV 20(SP),R5 ;;GET THE INPUT NUMBER
(1) 031320 100004 BPL 1$;;BR IF INPUT IS POS.
(1) 031322 005405 NEG R5 ;;MAKE THE BINARY NUMBER POS.
(1) 031324 112766 000055 000001 MOVB #'-,1(SP) ;;MAKE THE ASCII NUMBER NEG.
(1) 031332 005000 1$: CLR R0 ;;ZERO THE CONSTANTS INDEX
(1) 031334 012703 031512 MOV # $DBLK,R3 ;;SETUP THE OUTPUT POINTER
(1) 031340 112723 000040 MOVB #' ,(R3)+ ;;SET THE FIRST CHARACTER TO A BLANK
(1) 031344 005002 2$: CLR R2 ;;CLEAR THE BCD NUMBER
(1) 031346 016001 031502 MOV $DTBL(R0),R1 ;;GET THE CONSTANT

```

```

(1) 031352 160105 3$: SUB R1,R5 ;;FORM THIS BCD DIGIT
(1) 031354 002402 BLT 4$;;BR IF DONE
(1) 031356 005202 INC R2 ;;INCREASE THE BCD DIGIT BY 1
(1) 031360 000774 BR 3$
(1) 031362 060105 4$: ADD R1,R5 ;;ADD BACK THE CONSTANT
(1) 031364 005702 TST R2 ;;CHECK IF BCD DIGIT=0
(1) 031366 001002 BNE 5$;;FALL THROUGH IF 0
(1) 031370 105716 TSTB (SP) ;;STILL DOING LEADING 0'S?
(1) 031372 100407 BMI 7$;;BR IF YES
(1) 031374 106316 5$: ASLB (SP) ;;MSD?
(1) 031376 103003 BCC 6$;;BR IF NO
(1) 031400 116663 000001 177777 MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
(1) 031406 052702 000060 6$: BIS #'0,R2 ;;MAKE THE BCD DIGIT ASCII
(1) 031412 052702 000040 7$: BIS #' ,R2 ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 031416 110223 MOVB R2,(R3)+ ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 031420 005720 TST (R0)+ ;;JUST INCREMENTING
(1) 031422 020027 000010 CMP R0,#10 ;;CHECK THE TABLE INDEX
(1) 031426 002746 BLT 2$;;GO DO THE NEXT DIGIT
(1) 031430 003002 BGT 8$;;GO TO EXIT
(1) 031432 010502 MOV R5,R2 ;;GET THE LSD
(1) 031434 000764 BR 6$;;GO CHANGE TO ASCII
(1) 031436 105726 8$: TSTB (SP)+ ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 031440 100003 BPL 9$;;BR IF NO
(1) 031442 116663 177777 177776 MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 031450 105013 9$: CLRB (R3) ;;SET THE TERMINATOR
(3) 031452 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
(3) 031454 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
(3) 031456 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
(3) 031460 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 031462 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 031464 104401 031512 TYPE ,SDBLK ;;NOW TYPE THE NUMBER
(1) 031470 016666 000002 000004 MOV 2(SP),4(SP) ;;ADJUST THE STACK
(1) 031476 012616 MOV (SP)+,(SP)
(1) 031500 000002 RTI ;;RETURN TO USER
(1) 031502 023420 $DTBL: 10000.
(1) 031504 001750 1000.
(1) 031506 000144 100.
(1) 031510 000012 10.
(1) 031512 000004 $DBLK: .BLKW 4
4847 .SBTTL ERROR HANDLER ROUTINE
(1)
(2)
(1)
(1) *****
(1) *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
(1) *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1) *AND GO TO $ERRTYP ON ERROR
(1) *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) *SW15=1 HALT ON ERROR
(1) *SW13=1 INHIBIT ERROR TYPEOUTS
(1) *SW10=1 BELL ON ERROR
(1) *SW09=1 LOOP ON ERROR
(1) *CALL
(1) * ERROR N ;;ERROR-EMT AND N=ERROR ITEM NUMBER
(1)
(1) 031522 $ERROR:
(1) 031522 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(1) 031524 105237 001103 7$: INCB $ERFLG ;;SET THE ERROR FLAG

```

```

(1) 031530 001775 BEQ 7$;; DON'T LET THE FLAG GO TO ZERO
(1) 031532 013777 001102 147402 MOV $TSTNM,@DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
(1) 031540 032777 002000 147372 BIT #BIT10,@SWR ;; BELL ON ERROR?
(1) 031546 001402 BEQ 1$;; NO - SKIP
(1) 031550 104401 001164 TYPE $BELL ;; RING BELL
(1) 031554 005237 001112 1$: INC $ERTTL ;; COUNT THE NUMBER OF ERRORS
(1) 031560 011637 001116 MOV (SP),$ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION
(1) 031564 162737 000002 001116 SUB #2,$ERRPC
(1) 031572 117737 147320 001114 MOVB @$ERRPC,$ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
(1) 031600 032777 020000 147332 BIT #BIT13,@SWR ;; SKIP TYPEOUT IF SET
(1) 031606 001004 BNE 20$;; SKIP TYPEOUTS
(1) 031610 004737 031716 JSR PC,$ERRTYP ;; GO TO USER ERROR ROUTINE
(1) 031614 104401 001171 TYPE ,SRLF
(1) 031620 20$:
(1) 031620 122737 000001 001214 CMPB #APTENV,$ENV ;; RUNNING IN APT MODE
(1) 031626 001007 BNE 2$;; NO, SKIP APT ERROR REPORT
(1) 031630 113737 001114 031642 MOVB $ITEMB,21$;; SET ITEM NUMBER AS ERROR NUMBER
(1) 031636 004737 033512 JSR PC,$ATY4 ;; REPORT FATAL ERROR TO APT
(1) 031642 000 21$: .BYTE 0
(1) 031643 000 .BYTE 0
(1) 031644 000777 BR 22$;; APT ERROR LOOP
(1) 031646 005777 147266 2$: TST @SWR ;; HALT ON ERROR
(1) 031652 100002 BPL 3$;; SKIP IF CONTINUE
(1) 031654 000000 HALT
(1) 031656 104407 CKSWR ;; HALT ON ERROR!
(1) 031660 032777 001000 147252 3$: BIT #BIT09,@SWR ;; TEST FOR CHANGE IN SOFT-SWR
(1) 031666 001402 BEQ 4$;; LOOP ON ERROR SWITCH SET?
(1) 031670 013716 001110 MOV $LPERR,(SP) ;; BR IF NO
(1) 031674 005737 001162 4$: TST $ESCAPE ;; FUDGE RETURN FOR LOOPING
(1) 031700 001402 BEQ 5$;; CHECK FOR AN ESCAPE ADDRESS
(1) 031702 013716 001162 MOV $ESCAPE,(SP) ;; BR IF NONE
(1) 031706 5$:
(5) 031706 005237 001776 10$: INC ERCNT
(3) 031712 001775 BEQ 10$
(3) 031714 000002 RTI

```

4848

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

(1)
(2)
(1) *****
(1) *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(1) *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(1) *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)

```

```

(1) 031716 $ERRTYP:
(1) 031716 104401 001171 TYPE ,SRLF ;; "CARRIAGE RETURN" & "LINE FEED"
(1) 031722 010046 MOV RO,-(SP) ;; SAVE RO
(1) 031724 005000 CLR RO ;; PICKUP THE ITEM INDEX
(1) 031726 153700 001114 BISB @#$ITEMB,RO
(1) 031732 001004 BNE 1$;; IF ITEM NUMBER IS ZERO, JUST
(1) ;; TYPE THE PC OF THE ERROR
(2) 031734 013746 001116 MOV $ERRPC,-(SP) ;; SAVE $ERRPC FOR TYPEOUT
(2) ;; ERROR ADDRESS
(2) 031740 104402 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 031742 000426 BR 6$;; GET OUT
(1) 031744 005300 1$: DEC RO ;; ADJUST THE INDEX SO THAT IT WILL
(1) 031746 006300 ASL RO ;; WORK FOR THE ERROR TABLE
(1) 031750 006300 ASL RO

```

```

(1) 031752 006300 ASL R0
(1) 031754 062700 001256 ADD #ERRTB,R0 ;;FORM TABLE POINTER
(1) 031760 012037 031770 MOV (R0)+,2$;;PICKUP "ERROR MESSAGE" POINTER
(1) 031764 001404 BEQ 3$;;SKIP TYPEOUT IF NO POINTER
(1) 031766 104401 TYPE ;;TYPE THE "ERROR MESSAGE"
(1) 031770 000000 2$: .WORD 0 ;;"ERROR MESSAGE" POINTER GOES HERE
(1) 031772 104401 001171 TYPE ,SCRLF ;;"CARRIAGE RETURN" & "LINE FEED"
(1) 031776 012037 032006 3$: MOV (R0)+,4$;;PICKUP "DATA HEADER" POINTER
(1) 032002 001404 BEQ 5$;;SKIP TYPEOUT IF 0
(1) 032004 104401 TYPE ;;TYPE THE "DATA HEADER"
(1) 032006 000000 4$: .WORD 0 ;;"DATA HEADER" POINTER GOES HERE
(1) 032010 104401 001171 TYPE ,SCRLF ;;"CARRIAGE RETURN" & "LINE FEED"
(1) 032014 011000 5$: MOV (R0),R0 ;;PICKUP "DATA TABLE" POINTER
(1) 032016 001004 BNE 7$;;GO TYPE THE DATA
(1) 032020 012600 6$: MOV (SP)+,R0 ;;RESTORE R0
(1) 032022 104401 001171 TYPE ,SCRLF ;;"CARRIAGE RETURN" & "LINE FEED"
(1) 032026 000207 RTS PC ;;RETURN
(1) 032030 7$:
(2) 032030 013046 MOV @ (R0)+,-(SP) ;;SAVE @ (R0)+ FOR TYPEOUT
(2) 032032 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 032034 005710 TST (R0) ;;IS THERE ANOTHER NUMBER?
(1) 032036 001770 BEQ 6$;;BR IF NO
(1) 032040 104401 032046 TYPE ,8$;;TYPE TWO(2) SPACES
(1) 032044 000771 BR 7$;;LOOP
(1) 032046 020040 000 8$: .ASCIZ / / ;;TWO(2) SPACES
(1) 032052 .EVEN
4849 .SBTTL SCOPE HANDLER ROUTINE
(1)
(2)
(1)
(1) *****
(1) *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(1) *AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(1) *AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(1) *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) *SW14=1 LOOP ON TEST
(1) *SW11=1 INHIBIT ITERATIONS
(1) *SW09=1 LOOP ON ERROR
(1) *SW08=1 LOOP ON TEST IN SWR<7:0>
(1) *CALL
(1) * SCOPE ;;SCOPE=10T
(1)
(1) $SCOPE:
(1) 032052 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
(2) 032054 104407 CKSWR
(1) 032056 032777 040000 147054 1$: BIT #BIT14,@SWR ;;LOOP ON PRESENT TEST?
(1) 032064 001114 BNE $OVER ;;YES IF SW14=1
(1) ;*****START OF CODE FOR THE XOR TESTER*****
(1) 032066 000416 $XTSTR: BR 6$;;IF RUNNING ON THE "XOR" TESTER CHANGE
(1) ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
(1) 032070 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 032074 012737 032114 000004 MOY #5$,@#ERRVEC ;;SET FOR TIMEOUT
(1) 032102 005737 177060 TST @#177060 ;;TIME OUT ON XOR?
(1) 032106 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
(1) 032112 000463 BR $$VLAD ;;GO TO THE NEXT TEST
(1) 032114 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
(1) 032116 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
(1) 032122 000423 BR 7$;;LOOP ON THE PRESENT TEST

```

```
(1) 032124 6$:;##### OF CODE FOR THE XOR TESTER#####
(1) 032124 032777 000400 147006 BIT #BIT08,@SWR ;;LOOP ON SPEC. TEST?
(1) 032132 001404 BEQ 2$;;BR IF NO
(1) 032134 127737 147000 001102 CMPB @SWR,$STNM ;;ON THE RIGHT TEST? SWR<7:0>
(1) 032142 001465 BEQ $OVER ;;BR IF YES
(1) 032144 105737 001103 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
(1) 032150 001421 BEQ 3$;;BR IF NO
(1) 032152 123737 001115 001103 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 032160 101015 BHI 3$;;BR IF NO
(1) 032162 032777 001000 146750 BIT #BIT09,@SWR ;;LOOP ON ERROR?
(1) 032170 001404 BEQ 4$;;BR IF NO
(1) 032172 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
(1) 032200 000446 BR $OVER
(1) 032202 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
(1) 032206 005037 001160 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 032212 000415 BR 1$;;ESCAPE TO THE NEXT TEST
(1) 032214 032777 004000 146716 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
(1) 032222 001011 BNE 1$;;BR IF YES
(1) 032224 005737 001202 TST $PASS ;;IF FIRST PASS OF PROGRAM
(1) 032230 001406 BEQ 1$;; INHIBIT ITERATIONS
(1) 032232 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
(1) 032236 023737 001160 001104 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
(1) 032244 002024 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
(1) 032246 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
(1) 032254 013737 032332 001160 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(1) 032262 105237 001102 $SVLAD: INCB $STNM ;;COUNT TEST NUMBERS
(1) 032266 113737 001102 001200 MOVB $STNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
(1) 032274 011637 001106 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
(1) 032300 011637 001110 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
(1) 032304 005037 001162 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 032310 112737 000001 001115 MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 032316 013777 001102 146616 $OVER: MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1) 032324 013716 001106 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
(1) 032330 000002 RTI ;;FIXES PS
(1) 032332 000010 $MXCNT: 10 ;;MAX. NUMBER OF ITERATIONS
4850 .SBTTL TTY INPUT ROUTINE
(1)
(2) ;:*****
(1) .ENABL LSB
(1)
(2) ;:*****
(1) ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
(1) ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
(1) ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
(1) ;*WHEN OPERATING IN TTY FLAG MODE.
(1) 032334 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
(1) 032342 001074 BNE 15$;;BRANCH IF NO
(1) 032344 105777 146574 TSTB @STKS ;;CHAR THERE?
(1) 032350 100071 BPL 15$;;IF NO, DON'T WAIT AROUND
(1) 032352 117746 146570 MOVB @STKB,-(SP) ;;SAVE THE CHAR
(1) 032356 042716 177600 BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
(1) 032362 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
(1) 032366 001062 BNE 15$;;NO, RETURN TO USER
(1) 032370 123727 001134 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
(1) 032376 001456 BEQ 15$;;BRANCH IF YES
(1)
```



```

(1) 032400 104401 033061 TYPE ,SCNTLG ;; ECHO THE CONTROL-G (^G)
(1) 032404 104401 033066 SGTSWR: TYPE ,SMSWR ;; TYPE CURRENT CONTENTS
(2) 032410 013746 000176 MOV SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
(2) 032414 104402 TYP0C ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 032416 104401 033077 TYPE ,SMNEW ;; PROMPT FOR NEW SWR
(1) 032422 005046 19$: CLR -(SP) ;; CLEAR COUNTER
(1) 032424 005046 CLR -(SP) ;; THE NEW SWR
(1) 032426 105777 146512 7$: TSTB @STKS ;; CHAR THERE?
(1) 032432 100375 BPL 7$;; IF NOT TRY AGAIN
(1) 032434 117746 146506 MOVB @STKB,-(SP) ;; PICK UP CHAR
(1) 032440 042716 177600 BIC #^C177,(SP) ;; MAKE IT 7-BIT ASCII
(1)
(1)
(1)
(1) 032444 021627 000025 9$: CMP (SP),#25 ;; IS IT A CONTROL-U?
(1) 032450 001005 BNE 10$;; BRANCH IF NOT
(1) 032452 104401 033054 TYPE ,SCNTLU ;; YES, ECHO CONTROL-U (^U)
(1) 032456 062706 000006 20$: ADD #6,SP ;; IGNORE PREVIOUS INPUT
(1) 032462 000757 BR 19$;; LET'S TRY IT AGAIN
(1)
(1)
(1) 032464 021627 000015 10$: CMP (SP),#15 ;; IS IT A <CR>?
(1) 032470 001022 BNE 16$;; BRANCH IF NO
(1) 032472 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?
(1) 032476 001403 BEQ 11$;; BRANCH IF YES
(1) 032500 016677 000002 146432 MOV 2(SP),@SWR ;; SAVE NEW SWR
(1) 032506 062706 000006 11$: ADD #6,SP ;; CLEAR UP STACK
(1) 032512 104401 001171 14$: TYPE ,SCRLF ;; ECHO <CR> AND <LF>
(1) 032516 123727 001135 000001 CMPB $INTAG,#1 ;; RE-ENABLE TTY KBD INTERRUPTS?
(1) 032524 001003 BNE 15$;; BRANCH IF NOT
(1) 032526 012777 000100 146410 MOV #100,@STKS ;; RE-ENABLE TTY KBD INTERRUPTS
(1) 032534 000002 RTI ;; RETURN
(1) 032536 004737 033322 16$: JSR PC,$TYPEC ;; ECHO CHAR
(1) 032542 021627 000060 CMP (SP),#60 ;; CHAR < 0?
(1) 032546 002420 BLT 18$;; BRANCH IF YES
(1) 032550 021627 000067 CMP (SP),#67 ;; CHAR > 7?
(1) 032554 003015 BGT 18$;; BRANCH IF YES
(1) 032556 042726 000060 BIC #60,(SP)+ ;; STRIP-OFF ASCII
(1) 032562 005766 000002 TST 2(SP) ;; IS THIS THE FIRST CHAR
(1) 032566 001403 BEQ 17$;; BRANCH IF YES
(1) 032570 006316 ASL (SP) ;; NO, SHIFT PRESENT
(1) 032572 006316 ASL (SP) ;; CHAR OVER TO MAKE
(1) 032574 006316 ASL (SP) ;; ROOM FOR NEW ONE.
(1) 032576 005266 000002 17$: INC 2(SP) ;; KEEP COUNT OF CHAR
(1) 032602 056616 177776 BIS -2(SP),(SP) ;; SET IN NEW CHAR
(1) 032606 000707 BR 7$;; GET THE NEXT ONE
(1) 032610 104401 001170 18$: TYPE ,SGUES ;; TYPE ?<CR><LF>
(1) 032614 000720 BR 20$;; SIMULATE CONTROL-U
(1) .DSABL LSB

```

```

(1)
(1)
(2) *****
(1) *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1) *CALL:
(1) * RDCHR ;; INPUT A SINGLE CHARACTER FROM THE TTY

```

```

(1) ;* RETURN HERE ;; CHARACTER IS ON THE STACK
(1) ;* ;; WITH PARITY BIT STRIPPED OFF
(1) ;
(1) ;
(1) 032616 011646 $RDCHR: MOV (SP),-(SP) ;; PUSH DOWN THE PC
(1) 032620 016666 000004 000002 MOV 4(SP),2(SP) ;; SAVE THE PS
(1) 032626 105777 146312 1$: TSTB @STKS ;; WAIT FOR
(1) 032632 100375 BPL 1$;; A CHARACTER
(1) 032634 117766 146306 000004 MOVB @STKB,4(SP) ;; READ THE TTY
(1) 032642 042766 177600 000004 BIC #'C<177>,4(SP) ;; GET RID OF JUNK IF ANY
(1) 032650 026627 000004 000023 CMP 4(SP),#23 ;; IS IT A CONTROL-S?
(1) 032656 001013 BNE 3$;; BRANCH IF NO
(1) 032660 105777 146260 2$: TSTB @STKS ;; WAIT FOR A CHARACTER
(1) 032664 100375 BPL 2$;; LOOP UNTIL ITS THERE
(1) 032666 117746 146254 MOVB @STKB,-(SP) ;; GET CHARACTER
(1) 032672 042716 177600 BIC #'C177,(SP) ;; MAKE IT 7-BIT ASCII
(1) 032676 022627 000021 CMP (SP)+,#21 ;; IS IT A CONTROL-Q?
(1) 032702 001366 BNE 2$;; IF NOT DISCARD IT
(1) 032704 000750 BR 1$;; YES, RESUME
(1) 032706 026627 000004 000140 3$: CMP 4(SP),#140 ;; IS IT UPPER CASE?
(1) 032714 002407 BLT 4$;; BRANCH IF YES
(1) 032716 026627 000004 000175 CMP 4(SP),#175 ;; IS IT A SPECIAL CHAR?
(1) 032724 003003 BGT 4$;; BRANCH IF YES
(1) 032726 042766 000040 000004 BIC #40,4(SP) ;; MAKE IT UPPER CASE
(1) 032734 000002 4$: RTI ;; GO BACK TO USER
(2) ;*****
(1) ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1) ;*CALL:
(1) ;* RDLIN ;; INPUT A STRING FROM THE TTY
(1) ;* RETURN HERE ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1) ;* ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
(1) ;
(1) 032736 010346 $RDLIN: MOV R3,-(SP) ;; SAVE R3
(1) 032740 012703 033044 1$: MOV #$TTYIN,R3 ;; GET ADDRESS
(1) 032744 022703 033054 2$: CMP #$TTYIN+8.,R3 ;; BUFFER FULL?
(1) 032750 101405 BLOS 4$;; BR IF YES
(1) 032752 104410 RDCHR ;; GO READ ONE CHARACTER FROM THE TTY
(1) 032754 112613 MOVB (SP)+,(R3) ;; GET CHARACTER
(1) 032756 122713 000177 10$: CMPB #177,(R3) ;; IS IT A RUBOUT
(1) 032762 001003 BNE 3$;; SKIP IF NOT
(1) 032764 104401 001170 4$: TYPE ,8QUES ;; TYPE A '?'
(1) 032770 000763 BR 1$;; CLEAR THE BUFFER AND LOOP
(1) 032772 111337 033042 3$: MOVB (R3),9$;; ECHO THE CHARACTER
(1) 032776 104401 033042 TYPE ,9$
(1) 033002 122723 000015 CMPB #15,(R3)+ ;; CHECK FOR RETURN
(1) 033006 001356 BNE 2$;; LOOP IF NOT RETURN
(1) 033010 105063 177777 CLRB -1(R3) ;; CLEAR RETURN (THE 15)
(1) 033014 104401 001172 TYPE ,8LF ;; TYPE A LINE FEED
(1) 033020 012603 MOV (SP)+,R3 ;; RESTORE R3
(1) 033022 011646 MOV (SP),-(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 033024 016666 000004 000002 MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
(1) 033032 012766 033044 000004 MOV #$TTYIN,4(SP)
(1) 033040 000002 RTI ;; RETURN
(1) 033042 000 9$: .BYTE 0 ;; STORAGE FOR ASCII CHAR. TO TYPE
(1) 033043 000 .BYTE 0 ;; TERMINATOR
(1) 033044 000010 $TTYIN: .BLKB 8. ;; RESERVE 8 BYTES FOR TTY INPUT

```

```

(1) 033054 052536 005015 000 $CNTLU: .ASCIZ / ^U/<15><12> ;;CONTROL 'U'
(1) 033061 136 006507 000012 $CNTLG: .ASCIZ / ^G/<15><12> ;;CONTROL 'G'
(1) 033066 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
(1) 033074 020075 000
(1) 033077 040 047040 053505 $MNEW: .ASCIZ / NEW = /
(1) 033104 036440 000040
4851 : .$$SB2D
4852 : .$$DB2D
4853
4854 .SBTTL TYPE ROUTINE
(1)
(2) ;*****
(1) ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1) ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1) ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1) ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1) ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1) ;*
(1) ;*CALL:
(1) ;*1) USING A TRAP INSTRUCTION
(1) ;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1) ;*OR
(1) ;* TYPE
(1) ;* MESADR
(1) ;*
(1)
(1) 033110 105737 001157 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
(1) 033114 100002 BPL 1$;;BR IF YES
(1) 033116 000000 HALT ;;HALT HERE IF NO TERMINAL
(1) 033120 000430 BR 3$;;LEAVE
(1) 033122 010046 1$: MOV RO,-(SP) ;;SAVE RO
(1) 033124 017600 000002 MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
(1) 033130 122737 000001 001214 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 033136 001011 BNE 62$;;NO,GO CHECK FOR APT CONSOLE
(1) 033140 132737 000100 001215 BITB #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1) 033146 001405 BEQ 62$;;NO,GO CHECK FOR CONSOLE
(1) 033150 010037 033160 MOV RO,61$;;SETUP MESSAGE ADDRESS FOR API
(1) 033154 004737 033502 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
(1) 033160 000000 61$: .WORD 0 ;;MESSAGE ADDRESS
(1) 033162 132737 000040 001215 62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(1) 033170 001003 BNE 60$;;YES,SKIP TYPE OUT
(1) 033172 112046 2$: MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 033174 001005 BNE 4$;;BR IF IT ISN'T THE TERMINATOR
(1) 033176 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
(1) 033200 012600 60$: MOV (SP)+,RO ;;RESTORE RO
(1) 033202 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
(1) 033206 000002 RTI ;;RETURN
(1) 033210 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
(1) 033214 001430 BEQ 8$
(1) 033216 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
(1) 033222 001006 BNE 5$
(1) 033224 005726 TST (SP)+ ;;POP <CR><'F> EQUIV
(1) 033226 104401 TYPE ;;TYPE A CR AND LF
(1) 033230 001171 $CRLF
(1) 033232 105037 033366 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
(1) 033236 000755 BR 2$;;GET NEXT CHARACTER

```

```

(1) 033240 004737 033322 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
(1) 033244 123726 001156 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
(1) 033250 001350 BNE 2$;;IF NO GO GET NEXT CHAR.
(1) 033252 013746 001154 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
(1) ;;AND THE NULL CHAR.
(1) 033256 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
(1) 033262 002770 BLT 6$;;BR IF NO--GO POP THE NULL OFF OF STACK
(1) 033264 004737 033322 JSR PC,$TYPEC ;;GO TYPE A NULL
(1) 033270 105337 033366 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
(1) 033274 000770 BR 7$;;LOOP

```

;HORIZONTAL TAB PROCESSOR

```

(1) 033276 112716 000040 8$: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE
(1) 033302 004737 033322 9$: JSR PC,$TYPEC ;;TYPE A SPACE
(1) 033306 132737 000007 033366 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
(1) 033314 001372 BNE 9$;;TAB STOP
(1) 033316 005726 TST (SP)+ ;;POP SPACE OFF STACK
(1) 033320 000724 BR 2$;;GET NEXT CHARACTER
(1) 033322 105777 145622 $TYPEC: TSTB @STPS ;;WAIT UNTIL PRINTER IS READY
(1) 033326 100375 BPL $TYPEC
(1) 033330 116677 000002 145614 MOVB 2(SP),@$TPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 033336 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
(1) 033344 001003 BNE 1$;;BRANCH IF NO
(1) 033346 105037 033366 CLR $CHARCNT ;;YES--CLEAR CHARACTER COUNT
(1) 033352 00406 BR $TYPEX ;;EXIT
(1) 033354 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
(1) 033362 001402 BEQ $TYPEX ;;BRANCH IF YES
(1) 033364 105227 INCB (PC)+ ;;COUNT THE CHARACTER
(1) 033366 000000 $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
(1) 033370 000207 $TYPEX: RTS PC

```

4855

.SBTTI READ AN OCTAL NUMBER FROM THE TTY

```

(1) ;;*****
(2) ;;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1) ;;CHANGE IT TO BINARY.
(1) ;;CALL:
(1) ;; * RDOCT ;;READ AN OCTAL NUMBER
(1) ;; * RETURN HERE ;;LOW ORDER BITS ARE ON TOP OF THE STACK
(1) ;; * ;;HIGH ORDER BITS ARE IN $HIOCT
(1)
(1) 033372 011646 000004 000002 $RDOCT: MOV (SP),-(SP) ;;PROVIDE SPACE FOR THE
(1) 033374 016666 MOV 4(SP),2(SP) ;;INPUT NUMBER
(3) 033402 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 033404 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(3) 033406 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
(1) 033410 104411 1$: RDLIN ;;READ AN ASCII LINE
(1) 033412 012600 MOV (SP)+,R0 ;;GET ADDRESS OF 1ST CHARACTER
(1) 033414 005001 CLR R1 ;;CLEAR DATA WORD
(1) 033416 005002 CLR R2
(1) 033420 112046 2$: MOVB (R0)+,-(SP) ;;PICKUP THIS CHARACTER
(1) 033422 001412 BEQ 3$;;IF ZERO GET OUT
(1) 033424 006301 ASL R1 ;;*2
(1) 033426 006102 ROL R2
(1) 033430 006301 ASL R1 ;;*4

```

```

(1) 033432 006102 ROL R2
(1) 033434 006301 ASL R1 ;;*8
(1) 033436 006102 ROL R2
(1) 033440 042716 177770 BIC #^C7,(SP) ;;STRIP THE ASCII JUNK
(1) 033444 062601 ADD (SP)+,R1 ;;ADD IN THIS DIGIT
(1) 033446 000764 BR 2$;;LOOP
(1) 033450 005726 3$: TST (SP)+ ;;CLEAN TERMINATOR FROM STACK
(1) 033452 010166 000012 MOV R1,12(SP) ;;SAVE THE RESULT
(1) 033456 010237 033472 MOV R2,$HIOCT
(3) 033462 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
(3) 033464 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 033466 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 033470 000002 RTI
(1) 033472 000000 $HIOCT: .WORD 0 ;;RETURN
 .SBTTL APT COMMUNICATIONS ROUTINE ;;HIGH ORDER BITS GO HERE

4856
(1)
(2)
(1) 033474 112737 000001 033740 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
(1) 033502 112737 00000i 033736 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
(1) 033510 000403 BR $ATYC
(1) 033512 112737 000001 033740 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
(1) 033520 $ATYC:
(3) 033520 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 033522 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(1) 033524 105737 033736 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
(1) 033530 001450 BEQ 5$;;IF NOT: BR
(1) 033532 122737 000001 001214 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
(1) 033540 001031 BNE 3$;;IF NOT: BR
(1) 033542 132737 000100 001215 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 033550 001425 BEQ 3$;;IF NOT: BR
(1) 033552 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
(1) 033556 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 033564 005737 001174 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
(1) 033570 001375 BNE 1$;;IF NOT: WAIT
(1) 033572 010037 001210 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
(1) 033576 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
(1) 033600 001376 BNE 2$
(1) 033602 163700 001210 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
(1) 033606 006200 ASR R0 ;;GET MESSAGE LNGTH IN WORDS
(1) 033610 010037 001212 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
(1) 033614 012737 000004 001174 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
(1) 033622 000413 BR 5$
(1) 033624 017637 000004 033650 3$: MOV @4(SP),4$;;PUT MSG ADDR IN JSR LINKAGE
(1) 033632 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
(3) 033640 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
(1) 033644 004737 033110 JSR PC,$TYPE ;;CALL TYPE MACRO
(1) 033650 000000 4$: .WORD 0
(1) 033652 5$:
(1) 033652 105737 033740 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
(1) 033656 001416 BEQ 12$;;IF NOT: BR
(1) 033660 005737 001214 TST $ENV ;;RUNNING UNDER APT?
(1) 033664 001413 BEQ 12$;;IF NOT: BR
(1) 033666 005737 001174 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
(1) 033672 001375 BNE 11$;;IF NOT: WAIT
(1) 033674 017637 000004 001176 MOV @4(SP),$FATAL ;;GET ERROR #
(1) 033702 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.

```

```

(1) 033710 005237 001174 INC $MSGTYPE ;; TELL APT TO TAKE ERROR
(1) 033714 105037 033740 12$: CLR $FFLG ;; CLEAR FATAL FLAG
(1) 033720 105037 033737 CLR $LFLG ;; CLEAR LOG FLAG
(1) 033724 105037 033736 CLR $MFLG ;; CLEAR MESSAGE FLAG
(3) 033730 012601 MOV (SP)+,R1 ;; POP STACK INTO R1
(3) 033732 012600 MOV (SP)+,R0 ;; POP STACK INTO R0
(1) 033734 000207 RTS PC ;; RETURN
(1) 033736 000 $MFLG: .BYTE 0 ;; MESSG. FLAG
(1) 033737 000 $LFLG: .BYTE 0 ;; LOG FLAG
(1) 033740 000 $FFLG: .BYTE 0 ;; FATAL FLAG
(1) 033742 .EVEN
(1) 000700 APTSIZE=200
(1) 000001 APTENV=001
(1) 000100 APTSPOOL=100
(1) 000040 APTCSUP=040
4857 .SBTTL POWER DOWN AND UP ROUTINES
(1)
(2)
(1)
(1) 033742 012737 034106 000024 $PWRDN: MOV #SILLUP,@PWRVEC ;; SET FOR FAST UP
(1) 033750 012737 000340 000026 MOV #340,@PWRVEC+2 ;; Prio:7
(3) 033756 010046 MOV R0,-(SP) ;; PUSH R0 ON STACK
(3) 033760 010146 MOV R1,-(SP) ;; PUSH R1 ON STACK
(3) 033762 010246 MOV R2,-(SP) ;; PUSH R2 ON STACK
(3) 033764 010346 MOV R3,-(SP) ;; PUSH R3 ON STACK
(3) 033766 010446 MOV R4,-(SP) ;; PUSH R4 ON STACK
(3) 033770 010546 MOV R5,-(SP) ;; PUSH R5 ON STACK
(3) 033772 017746 145142 MOV @SWR,-(SP) ;; PUSH @SWR ON STACK
(1) 033776 010637 034112 MOV SP,$SAVR6 ;; SAVE SP
(1) 034002 012737 034014 000024 MOV #PWRUP,@PWRVEC ;; SET UP VECTOR
(1) 034010 000000 HALT
(1) 034012 000776 BR .-2 ;; HANG UP
(1)
(2)
(1)
(1) 034014 012737 034106 000024 $PWRUP: MOV #SILLUP,@PWRVEC ;; SET FOR FAST DOWN
(1) 034022 013706 034112 MOV $SAVR6,SP ;; GET SP
(1) 034026 005037 034112 CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
(1) 034032 005237 034112 1$: INC $SAVR6 ;; WAIT FOR THE INC
(1) 034036 001375 BNE 1$;; OF WORD
(3) 034040 012677 145074 MOV (SP)+,@SWR ;; POP STACK INTO @SWR
(3) 034044 012605 MOV (SP)+,R5 ;; POP STACK INTO R5
(3) 034046 012604 MOV (SP)+,R4 ;; POP STACK INTO R4
(3) 034050 012603 MOV (SP)+,R3 ;; POP STACK INTO R3
(3) 034052 012602 MOV (SP)+,R2 ;; POP STACK INTO R2
(3) 034054 012601 MOV (SP)+,R1 ;; POP STACK INTO R1
(3) 034056 012600 MOV (SP)+,R0 ;; POP STACK INTO R0
(1) 034060 012737 033742 000024 MOV #PWRDN,@PWRVEC ;; SET UP THE POWER DOWN VECTOR
(1) 034066 012737 000340 000026 MOV #340,@PWRVEC+2 ;; Prio:7
(1) 034074 104401 TYPE PWRMSG ;; REPORT THE POWER FAILURE
(1) 034076 034114 $PWRMG: .WORD PWRMSG ;; POWER FAIL MESSAGE POINTER
(1) 034100 012716 MOV (PC)+,(SP) ;; RESTART AT START
(1) 034102 002016 $PWRAD: .WORD START ;; RESTART ADDRESS
(1) 034104 000002 RTI
(1) 034106 000000 $SILLUP: HALT
(1) 034110 000776 BR .-2 ;; THE POWER UP SEQUENCE WAS STARTED
;; BEFORE THE POWER DOWN WAS COMPLETE

```

```

(1) 034112 000000 $SAVR6: 0 ;:PUT THE SP HERE
4858 034114 015 012 PWRMSG. .BYTE 15,12
4859 034116 042522 052123 051101 .ASCII /RESTARTING AFTER A POWER FAILURE/
 034124 044524 043516 040440
 034132 052106 051105 040440
 034140 050040 053517 051105
 034146 043040 044501 052514
 034154 042522
4860 034156 015 012 012 .BYTE 15,12,12,0
 034161 000 .EVEN
4861
4862
4863 .SBTTL ; WIR ROUTINE TO WAIT FOR 'DRLPX0' MICROCODE TO BECOME READY.
4864
4865 034162 005037 034210 WIR: CLR WIRT
4866 034166 122777 000377 145276 1$: CMPB #377,@KMD2
4867 034174 001404 BEQ 2$
4868 034176 005237 034210 INC WIRT ;TIME OUT?
4869 034202 001371 BNE 1$
4870
4871 034204 104003 ERROR 3 ;TIME OUT ERROR,KMC-11 ERROR.
4872
4873 034206 2$:
4874 034206 000207 RTS PC
4875 034210 000000 WIRT: .WORD 0

```

```

4877
4878
4879
4880
4881
4882
4883
4884
4885
4886 034212 004737 040536 SYNC: JSR PC,SRESET ;RESET SYSTEM
4887
4888 034216 012777 044022 145252 MOV #KWT,@LPADL ;CLOCK INFO
4889 034224 052777 000001 145234 BIS #BIT0,@LPCI ;START
4890 034232 004737 040452 JSR PC,SDELAY ;DELAY
4891 034236 117737 145232 001126 MOVB @LPS0,$BDDAT ;DUMB READ OF STAT REG.
4892 034244 000207 RTS PC
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908 034246 013737 034574 034566 AVERR: MOV CHANS,CHAN ;SET TO FIRST CHAN.
4909 034254 012701 056156 MOV #BUFFER,R1
4910 034260 013737 034554 034610 SPEP: MOV CHANU,SAMOFF
4911 034266 006337 034610 ASL SAMOFF
4912 034272 013737 034566 034564 MOV CHAN,CHAN1
4913 034300 006337 034564 ASL CHAN1
4914 034304 012737 000020 034606 AVERRL: MOV #16,SAMCNT
4915 034312 063701 034564 ADD CHAN1,R1 ;PUT CHAN OFFSET IN R1.
4916 034316 005037 034616 CLR AVTKN ;SET INITIAL CONDITIONS FOR THIS CHAN.
4917 034322 005037 034624 CLR RTEMP
4918 034326 011137 034620 MOV (1),RLOW
4919 034332 011137 034622 MOV (1),RHIGH
4920 034336 023711 034622 2$: CMP RHIGH,(1) ;FIND REAL HIGH VALUE.
4921 034342 003002 BGT 3$
4922 034344 011137 034622 MOV (1),RHIGH
4923 034350 021137 034620 3$: CMP (1),RLOW ;FIND REAL LOW VALUE.
4924 034354 003002 BGT 4$
4925 034356 011137 034620 MOV (1),RLOW
4926 034362 011137 034626 4$: MOV (1),RTEMP1 ;GET CURRENT SAMPLE.
4927 034366 063737 034626 034616 ADD RTEMP1,AVTKN ;'BOOT' ADD ALL SAMPLES.
4928 034374 005537 034624 ADC RTEMP
4929 034400 063701 034610 ADD SAMOFF,R1
4930 034404 005337 034606 DEC SAMCNT
4931 034410 001352 BNE 2$
4932 034412 013737 034614 034626 MOV AVEXP,RTEMP1

```



```

4933 034420 063737 034612 034626 ADD TOLER,RTEMP1
4934 034426 023737 034626 034622 CMP RTEMP1,RHIGH
4935 034434 002427 BLT ERAV1 ;NO-THEN REPORT ERROR.
4936 034436 163737 034612 034626 SUB TOLER,RTEMP1 ;YES-OK-CHECK LOWEST READING.
4937 034444 163737 034612 034626 SUB TOLER,RTEMP1
4938 034452 023737 034620 034626 CMP RLOW,RTEMP1
4939 034460 002415 BLT ERAV1 ;NO-REPORT ERROR.
4940 034462 005737 034612 TST TOLER ;DOES OPERATOR WISH 'FORCED' TYPEOUT?
4941 034466 001412 BEQ ERAV2 ;IF SO-DO IT
4942 034470 062737 000002 034564 AVERRN: ADD #2,CHAN1 ;SET TO DO NEXT CHAN-BUT
4943 034476 005237 034566 INC CHAN ;IF DOEN ALL CHANS-EXIT.
4944 034502 023737 034566 034576 CMP CHAN,CHANF ;OTHERWISE LOOP.
4945 034510 003675 BLE AVERRL
4946 034512 000207 RTS PC
4947
4948 ;*ERROR REPORTER
4949
4950 034514 ERAV1:
4951 034514 ERAV2:
4952 034514 005737 034640 TST REPOR ;REPORT ERROR?
4953 034520 001002 BNE 1$
4954 034522 000137 034470 JMP AVERRN
4955 034526 1$:
4956 ;GO TYPE--OCTAL ASCII(AL)
4957 034526 042737 170000 034620 BIC #170000,RLOW ;GO TYPE--OCTAL ASCII(AL)
4958 ;GO TYPE--OCTAL ASCII(AL)
4959 034534 042737 170000 034616 BIC #170000,AVTKN ;GO TYPE--OCTAL ASCII(AL)
4960 ;GO TYPE--OCTAL ASCII(AL)
4961 034542 042737 170000 034622 BIC #170000,RHIGH
4962 034550 104020 ERROR 20
4963 034552 000746 BR AVERRN
4964
4965 ;*
4966 ;*POINTERS USED BY REPEATIBILITY TEST
4967 ;*
4968
4969 034554 000000 CHANU: 0
4970 034556 000000 CHAN7:0
4971 034560 000000 NA07:0
4972 034562 000000 NA17:0
4973 034564 000000 CHAN1: 0 ;LEFT JUSTIFIED CURRENT CHANNELL.
4974 034566 000000 CHAN: 0 ;CURRENT CHANNELL
4975 034570 000000 GAIN: 0 ;CURRENT GAIN
4976 034572 000000 ADWD: 0 ;WORD SENT TO A/D
4977 034574 000000 CHANS: 0 ;STARTING CHANNELL
4978 034576 000000 CHANF: 0 ;LAST CHANNELL
4979 034600 000000 CHANSR: 0
4980 034602 000000 CHANFR: 0
4981 034604 000000 CHANNO: 0
4982 034606 000000 SAMCNT: 0 ;SAMPLE COUNT
4983 034610 000000 SAMOFF: 0 ;SAMPLE OFFSET
4984 034612 000000 TOLER: 0 ;TOLERANCE BEFORE ERROR IS REPORT
4985 034614 000000 AVEXP: 0 ;EXPECTED AVERAGE
4986 034616 000000 AVTKN: 0 ;AVERAGE OF SAMPLES TAKEN
4987 034620 000000 RLOW: 0 ;LOWEST SAMPLE TAKEN
4988 034622 000000 RHIGH: 0 ;HIGHEST SAMPLE TAKEN

```

|      |        |        |        |        |         |          |                                    |
|------|--------|--------|--------|--------|---------|----------|------------------------------------|
| 4989 | 034624 | 000000 |        |        | RTEMP:  | 0        |                                    |
| 4990 | 034626 | 000000 |        |        | RTEMP1: | 0        |                                    |
| 4991 | 034630 | 000000 |        |        | REPMAN: | 0        |                                    |
| 4992 | 034632 | 000000 |        |        |         | 0        | :SECTION RESERVED FOR              |
| 4993 | 034634 | 000000 |        |        |         | 0        | :MANUAL REPEATIBILITY              |
| 4994 | 034636 | 000000 |        |        |         | 0        | :WHERE OPERETOR ENTERS             |
| 4995 | 034640 | 000000 |        |        | REPOR:  | 0        | :IN INFORMATION                    |
| 4996 | 034642 | 000    | 000    |        | .BYTE   | 0,0      |                                    |
| 4997 | 034644 | 000000 |        |        |         | 0        |                                    |
| 4998 |        |        |        |        |         |          |                                    |
| 4999 | 034646 | 104401 | 043612 |        | MAKEI:  | TYPE,M0  | :WELCOME MESSAGE                   |
| 5000 | 034652 | 104411 |        |        |         | RDLIN    |                                    |
| 5001 | 034654 | 005037 | 001200 |        |         | CLR      | \$TESTN                            |
| 5002 | 034660 | 012600 |        |        |         | MOV      | (SP)+,R0                           |
| 5003 | 034662 | 121027 | 000117 |        |         | CMPB     | (R0),#'0                           |
| 5004 | 034666 | 001002 |        |        |         | BNE      | 1\$                                |
| 5005 | 034670 | 000137 | 035616 |        |         | JMP      | LOOPJ                              |
| 5006 | 034674 | 104401 | 042525 |        | 1\$:    | TYPE,M1  |                                    |
| 5007 | 034700 | 104411 |        |        |         | RDLIN    |                                    |
| 5008 | 034702 | 012600 |        |        |         | MOV      | (SP)+,R0                           |
| 5009 | 034704 | 111037 | 001772 |        |         | MOVB     | (0),\$VERSN                        |
| 5010 | 034710 | 121027 | 000115 |        |         | CMPB     | (0),#'M                            |
| 5011 | 034714 | 001403 |        |        |         | BEQ      | 2\$                                |
| 5012 | 034716 | 121027 | 000104 |        |         | CMPB     | (0),#'D                            |
| 5013 | 034722 | 001364 |        |        |         | BNE      | 1\$                                |
| 5014 | 034724 | 104401 | 042740 |        | 2\$:    | TYPE,M2  |                                    |
| 5015 | 034730 | 104412 |        |        |         | RDOCT    |                                    |
| 5016 | 034732 | 012637 | 044024 |        |         | MOV      | (SP)+,KWT+2                        |
| 5017 | 034736 | 104401 | 042761 |        |         | TYPE,M3  |                                    |
| 5018 | 034742 | 104412 |        |        |         | RDOCT    |                                    |
| 5019 | 034744 | 012637 | 044026 |        |         | MOV      | (SP)+,KWT+4                        |
| 5020 | 034750 | 012737 | 000001 | 001124 | 3\$:    | MOV      | #1,\$GDDAT                         |
| 5021 | 034756 | 122737 | 000104 | 001772 |         | CMPB     | #'D,\$VERSN                        |
| 5022 | 034764 | 001403 |        |        |         | BEQ      | 4\$                                |
| 5023 | 034766 | 104401 | 043024 |        |         | TYPE,M4  |                                    |
| 5024 | 034772 | 104412 |        |        |         | RDOCT    |                                    |
| 5025 | 034774 | 012637 | 001124 |        | 4\$:    | MOV      | (SP)+,\$GDDAT                      |
| 5026 | 035000 | 123727 | 001124 | 000011 |         | CMPB     | \$GDDAT,#11                        |
| 5027 | 035006 | 103360 |        |        |         | BHS      | 3\$                                |
| 5028 | 035010 | 013737 | 001124 | 001754 |         | MOV      | \$GDDAT,MYTEMP                     |
| 5029 | 035016 | 012737 | 000001 | 001126 |         | MOV      | #1,\$BDDAT                         |
| 5030 | 035024 | 012701 | 054776 |        |         | MOV      | #JOB0,R1                           |
| 5031 | 035030 | 012702 | 066000 |        |         | MOV      | #66000,R2                          |
| 5032 | 035034 | 010237 | 056156 |        |         | MOV      | R2,FRECOR                          |
| 5033 | 035040 | 013702 | 056156 |        |         | MOV      | FRECOR,R2                          |
| 5034 | 035044 | 104401 | 043044 |        | L1PED:  | TYPE,M5  |                                    |
| 5035 | 035050 | 013746 | 001126 |        |         | MOV      | \$BDDAT,-(SP)                      |
| (1)  | 035054 | 104402 |        |        |         | TYPOC    | ::SAVE \$BDDAT FOR TYPEOUT         |
| 5036 | 035056 | 104401 | 043054 |        |         | TYPE,M6  | ::GO TYPE--OCTAL ASCII(ALL DIGITS) |
| 5037 | 035062 | 104401 | 043064 |        |         | TYPE,M7  |                                    |
| 5038 | 035066 | 104412 |        |        |         | RDOCT    |                                    |
| 5039 | 035070 | 012611 |        |        |         | MOV      | (SP)+,(1)                          |
| 5040 | 035072 | 104401 | 043122 |        |         | TYPE,M8  | :STORE MODE WORD                   |
| 5041 | 035076 | 104412 |        |        |         | RDOCT    |                                    |
| 5042 | 035100 | 012661 | 000002 |        |         | MOV      | (SP)+,2(1)                         |
| 5043 | 035104 | 104401 | 043257 |        | 1\$:    | TYPE,M10 | :# OF BUFFERS                      |

```

5044 035110 104412 RDOCT
5045 035112 112661 000007 MOVB (SP)+,7(1)
5046 035116 001772 BEQ 1$
5047 035120 012737 035764 000004 MOV #ERTOUT,@#4 ;IN CASE OF BAD BUFFER AREA
5048 035126 116137 000007 001202 MOVB 7(1),$PASS
5049 035134 105361 000007 DECB 7(1)
5050 035140 010103 MOV R1,R3
5051 035142 016104 000002 MOV 2(1),R4
5052 035146 006304 ASL R4
5053 035150 010261 000010 2$: MOV R2,10(1)
5054 035154 104401 035162 TYPE ,65$;:TYPE ASCIZ STRING
(1) 035160 000410 BR 64$;:GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <200>#BUFFER ADDR. #
(1) 64$:
5055 035202 010246 MOV R2,-(SP) ;:SAVE R2 FOR TYPEOUT
(1) 035204 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
5056 035206 060402 ADD R4,R2
5057 035210 005712 TST (2)
5058 035212 005337 001202 DEC $PASS
5059 035216 001403 BEQ 3$
5060 035220 062701 000004 ADD #4,R1
5061 035224 000751 BR 2$
5062 035226 010301 3$: MOV R3,R1
5063 035230 104401 043154 TYPE ,M9 ;:USER SW=
5064 035234 016346 000004 MOV 4(R3),-(SP) ;:SAVE 4(R3) FOR TYPEOUT
(1) 035240 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
5065 035242 104401 043336 5$: TYPE,M12 ;:BUFFER OVERRUN FATAL?
5066 035246 104411 RDLIN
5067 035250 012600 MOV (SP)+,R0
5068 035252 142710 000240 BICB #240,(0)
5069 035256 122710 000131 CMPB #'Y,(0)
5070 035262 001004 BNE 6$
5071 035264 042761 100000 000006 BIC #BIT15,6(1)
5072 035272 000406 BR 7$
5073 035274 121027 000116 6$: CMPB (0),#'N
5074 035300 001360 BNE 5$
5075 035302 052761 100000 000006 BIS #BIT15,6(1)
5076 035310 005071 000004 7$: CLR @4(1) ;:CLEAR USER WORD.
5077 035314 104401 043461 TYPE,M14 ;:DELAY BEFORE START
5078 035320 104412 RDOCT
5079 035322 012661 000054 MOV (SP)+,54(1)
5080 035326 104401 043667 TYPE,M20 ;:CHAN ADDR WORD
5081 035332 104412 RDOCT
5082 035334 012661 000056 MOV (SP)+,56(1) ;:# OF CHAN
5083 035340 104401 043505 TYPE,M15
5084 035344 104412 RDOCT
5085 035346 012661 000060 MOV (SP)+,60(1) ;:CLOCK RATE
5086 035352 104401 043713 TYPE,M21
5087 035356 104412 RDOCT
5088 035360 012661 000062 MOV (SP)+,62(1)
5089 035364 032711 140000 BIT #BIT14!BIT15,(1) ;:EVENT OR START MASK
5090 035370 001453 BEQ 9$
5091 035372 104401 043730 TYPE,M22
5092 035376 104412 RDOCT
5093 035400 012661 000064 MOV (SP)+,64(1)
5094 035404 104401 043761 TYPE,M23

```

```

5095 035410 104412 RDOCT
5096 035412 012661 000066 MOV (SP)+,66(1)
5097 035416 104401 043777 TYPE,M24
5098 035422 104412 RDOCT
5099 035424 012661 000070 MOV (SP)+,70(1)
5100 035430 032711 001400 BIT #BIT8!BIT9,(1) ;RANDOM CHANNEL?
5101 035434 001031 BNE 9$;NO EXIT THIS LOOP
5102 035435 016037 000060 001202 MOV 60(0),$PASS ;YES GET NUMBER OF CHANS.
5103 035444 012703 000062 MOV #<JOB1-JOB0-34>,R3
5104 035450 060103 ADD R1,R3
5105 035452 010361 000050 MOV R3,50(1)
5106
5107 035456 13$:
(1) 035456 104401 035464 TYPE ,67$;;TYPE ASCIZ STRING
(1) 035462 000407 BR 66$;;GET OVER THE ASCIZ
(1) ;;67$: .ASCIZ <200>#RANDOM CH. =#
(1) 66$:
5108 035502 RDOCT
5109 035504 104412 MOV (SP)+,(3)+
5110 035506 012623 DEC $PASS
5111 035506 005337 001202 BNE 13$
5112 035512 001361 MOV #BIT15!BIT14,(3) ;ADD EOC MARKER.
5113 140000
5114 035520 9$:
5115
5116 035520 104401 043205 TYPE,M97
5117 035524 104412 RDOCT
5118 035526 012661 000074 MOV (SP)+,74(1)
5119 035532 032711 000200 BIT #BIT7,(1) ;OUTPUT DEVICE?
5120 035536 001404 BEQ 10$;NO-CONTINUE
5121 035540 104401 043570 TYPE, M18 ;MAKE BUFFER
5122 035544 104401 043634 TYPE, M19 ;PRESS CONTINUE
5123 035550 010237 056156 MOV R2,FRECOR
5124 035554 023727 056156 000000G 10$: CMP FRECOR,#DRLPX0 ;CORE EXCEEDED?
5125 035562 103100 BHIS ERTOUT ;YES-ERROR MESSAGE!
5126 035564 123737 001126 001124 CMPB $BDDAT,$GDDAT ;DONE ALL?
5127 035572 001411 BEQ 11$
5128 035574 005237 001126 INC $BDDAT
5129 035600 012703 055114 MOV #JOB1,R3 ;GET JOB SIZ
5130 035604 162703 054776 SUB #JOB0,R3
5131 035610 060301 ADD R3,R1 ;UPDATE JOB POINTER
5132 035612 000137 035044 JMP L1PED ;LOOP FOR NEXT JOBS DATA
5133
5134 035616 11$:
5135 035616 004737 036126 LOOPJ: JSR PC,FLASH
5136 ;;REPORT DATA
5137
5138 035622 104401 035630 TYPE ,65$;;TYPE ASCIZ STRING
(1) 035626 000414 BR 64$;;GET OVER THE ASCIZ
(1) ;;65$: .ASCIZ <200>#REPORT DATA FOR JOB 0?#
(1) 64$:
5139 035660 RDLIN
5140 035662 104411 MOV (SP)+,R0
5141 035664 012600 CMPB (0),#Y
5142 035670 001402 BEQ 2$
5143 035672 000137 034646 1$: JMP MAKEI

```

```

5144
5145 035676 012701 054776 2$: MOV #JOB0,R1
5146 035702 013700 001754 MOV MYTEMP,R0
5147 035706 016137 000060 034554 3$: MOV 60(1),CHANU ;GET NU. OF CHANS.
5148 035714 005003 CLR R3
5149
5150 035716 013702 034554 MOV CHANU,R2 ;NU OF CHANS
5151 035722 010337 034574 4$: MOV R3,CHANS
5152 035726 010337 034576 MOV R3,CHANF
5153
5154 035732 005237 034640 INC REPOR
5155 035736 004737 034246 JSR PC,AVERR
5156 035742 005203 INC R3
5157 035744 005302 DEC R2
5158 035746 001365 BNE 4$
5159 035750 062701 000116 ADD #<JOB1-JOB0>,R1
5160 035754 005300 DEC R0
5161 035756 001353 BNE 3$
5162 035760 000744 BR 1$
5163 035762 000000 HALT
5164
5165
5166 035764 ERTOUT:
(1) 035764 104401 035772 TYPE ,65$;;TYPE ASCIZ STRING
(1) 035770 000434 BR 64$;;GET OVER THE ASCIZ
(1) ;;65$: .ASCIZ <200>#INSUFFICIENT CORE TO DO REQUEST,PLEASE SPECIFY SMALLER#
(1) 64$:
5167 036062 TYPE ,67$;;TYPE ASCIZ STRING
(1) 036066 104401 036070 BR 66$;;GET OVER THE ASCIZ
(1) 036066 000415 ;;67$: .ASCIZ <200>#BUFFER OR LESS BUFFERS#
(1) 66$:
5168 036122 000137 034646 JMP MAKEI
5169
5170
5171 ;*
5172 ;*FLASH! ROUTINE TO EXERCISE 1-8 JOBS UNTIL DONE.
5173 ;* REQUIRED:
5174 ;* $BERSN MUST CONTAIN ASCII DOR M FOR DEDICATED OR MULTIUSER.
5175 ;* KWT PRE SET UP TO CLOCK STATS
5176 ;* JOB0-JOB7 PRE SET UP
5177 ;* $GDDAT= NO. OF JOBS TO EXERCISE
5178 ;*
5179 ;* CALL= JSR PC, FLASH
5180 ;* RETURNS WHEN ALL DONE.
5181 036126 FLASH:
5182
5183 036126 004737 040536 2$: JSR PC,SRESET
5184 036132 012777 054040 143336 MOV #DEVLST,@LPADL ;SET ADDR. OF DEVICE LIST IN RDA WORD
(1) 036140 005037 054040 CLR DEVLST ;INDICATE SIZING
(1) 036144 113737 001512 054041 MOVB VERSN,DEVLST+1 ;SET VERSION NUMBER
(1)
(1) 036152 152777 000001 143306 BISB #1,@LPCI ;SET GO.
(1) 036160 004737 040452 JSR PC,SDELAY
5185 036164 012777 044022 143304 MOV #KWT,@LPADL ;START CLOCK
5186 036172 052777 000001 143266 BIS #BIT0,@LPCI
5187 036200 012700 054430 MOV #TODOQ,R0

```

```

5188
5189 036204 005020 3$: CLR (0)+ ;CLEAR QUE AREA.
5190 036206 020027 054714 CMP RO,#DONEC
5191 036212 001374 BNE 3$
5192 036214 013700 001124 MOV $GDDAT,RO ;GET NO. OF JOBS
5193 036220 012701 054430 MOV #TODOQ,R1 ;GET QUES AREA
5194 036224 012702 054776 MOV #JOB0,R2 ;PICK UP 1ST JOB ADDRESS
5195 036230 012703 055114 MOV #JOB1,R3 ;CALCULATE ADDR DIF.
5196 036234 162703 054776 SUB #JOB0,R3
5197 036240 010037 054550 MOV RO,TODOC ;REMEMBER HOW MANY JOBS ARE QUEUED
5198 036244 012700 000010 MOV #8.,RO
5199 036250 005712 4$: TST (2) ;ANYTHING IN THIS JOB??
5200 036252 001401 BEQ 6$
5201 036254 010221 MOV R2,(1)+ ;STORE FIRST JOB
5202 036256
5203 036256 060302 6$: ADD R3,R2 ;POINT TO NEXT JOB
5204 036260 005300 DEC RO ;DONE ALL JOBS?
5205 036262 001372 BNE 4$;NO-LOOP.
5206 036264 005037 001126 CLR $BDDAT
5207 036270 063737 002014 001126 ADD FUDGE,$BDDAT ;DON'T WAIT FOR CLOCK JOBS(DR11K ONLY)
5208 036276 005037 002014 CLR FUDGE
5209 036302 013700 001556 MOV VECT1,RO
5210 036306 012720 036510 MOV #OUPSRV,(0)+
5211 036312 012720 000340 MOV #340,(0)+
5212 036316 012720 036434 MOV #INPSRV,(0)+
5213 036322 012720 000340 MOV #340,(0)+
5214 036326 052777 000100 143132 BIS #BIT6,@LPCI
5215 036334 052777 000100 143130 BIS #BIT6,@LPCO
5216 036342 013737 001124 001774 MOV $GDDAT,TAXING ;GET NO OF REQUESTS.
5217 036350 062737 000200 001774 ADD #200,TAXING
5218 036356 006337 001774 ASL TAXING ;MUL X2 TO GET TIME.
5219 036362 005037 177776 CLR PS
5220 036366 023737 001124 001126 5$: CMP $GDDAT,$BDDAT ;DONE ALL JOBS?
5221 036374 001414 BEQ 7$
5222 036376 004737 040452 JSR PC,$DELAY
5223 036402 005337 001774 DEC TAXING
5224 036406 001367 BNE 5$
5225 036410 017737 143052 001124 MOV @LPCI,$GDDAT
5226 036416 017737 143050 001126 MOV @LPCO,$BDDAT
5227 036424 104017 ERROR 17 ;JOB(S) FAILED TO FINISH (LPA FAULT)
5228 036426
5229 036426 004737 040536 7$: JSR PC,$RESET
5230 036432 000207 RTS PC
5231 036434 005737 054550 INPSRV: TST TODOC ;ANYTHING TO DO?
5232 036440 001422 BEQ IEX
5233 036442 010046 MOV RO,-(SP)
5234 036444 012700 054430 MOV #TODOQ,RO
5235 036450 005720 1$: TST (0)+
5236 036452 001776 BEQ 1$
5237 036454 014077 143016 MOV -(0),@LPADL ;SAVE RDA ADDR.
5238 036460 005010 CLR (0) ;ZERO THIS JOB
5239 036462 005337 054550 DEC TODOC
5240 036466 001003 BNE 2$
5241 036470 042777 000100 142770 BIC #BIT6,@LPCI ;DON'T ALLOW ANOTHER INTERRUPT.
5242 036476
5243 036476 052777 000001 142762 2$: BIS #BIT0,@LPCI ;SET GO.

```

```

5244 036504 012600 MOV (SP)+,R0
5245 036506 000002 IEX: RTI
5246
5247 036510 010046 OUPSRV: MOV R0,-(SP) ;SAVE REGS.
5248 036512 010146 MOV R1,-(SP)
5249 036514 010246 MOV R2,-(SP)
5250 036516 010346 MOV R3,-(SP)
5251 036520 010446 MOV R4,-(SP)
5252 036522 005237 001774 INC TAXING ;NO TIME OUT IF ACTIVE.
5253
5254 036526 017737 142744 001770 MOV @LPADL,ALPADL ;SAVE RDA ADDR.
5255 036534 017757 142732 001764 MOV @LPCO,ALPCO ;GET CONTROL OUT REG.
5256 036542 013700 001764 MOV ALPCO,R0 ;NOW LETS GET USER NUMBER.
5257 036546 042700 177770 BIC #^C<7>,R0 ;RID OF REST OF JUNK.
5258 036552 010037 001756 MOV R0,USJNO ;GET USJNO (USER INDEX)
5259 036556 010003 MOV R0,R3
5260 036560 006300 ASL R0
5261 036562 013701 001770 MOV ALPADL,R1
5262 036566 105737 001765 TSTB ALPCO+1 ;SEE IF BIT15 SET SHOWING ERROR
5263 036572 100024 BPL NER ;IF NO ERROR,PROCEED.
5264
5265 036574 017737 142666 001762 MOV @LPCI,ALPCI ;ELSE REPORT AN ERROR.
5266 036602 113737 001765 001766 MOV ALPCO+1,ALPSO ;FIX STATUS REG SO ITS EASIER TO READ.
5267 036610 105037 001767 CLRB ALPSO+1 ;ONLY LOW BYTE.
5268 036614 122737 000250 001766 CMPB #250,ALPSO ;LEGAL ERROR. BIT 14 SET IN USW
5269 036622 001003 BNE 40$
5270 036624 005237 001126 INC $BDDAT
5271 036630 000440 BR OUPEX
5272
5273 036632 104021 40$: ERROR 21 ;LPA-11 MICRO CODE REPORTED AN ERROR
5274
5275 036634 013737 001124 001126 MOV $GDDAT,$BDDAT ;FIX COUNT FOR EXIT TO MAIN LINE FLASH.
5276 036642 000433 BR OUPEX
5277
5278 036644 001005 NER: BNE 10$
5279 036646 005071 000004 CLR @4(1) ;CLEAR USW
5280 036652 013760 001770 054552 MOV ALPADL,TABLE(0) ;SAVE RDA ADDR. FOR FUTURE REF.
5281
5282 036660 10$:
5283 036660 016001 054552 MOV TABLE(0),R1 ;GET RDA ADDR FROM TABLE
5284 036664 016103 000004 MOV 4(1),R3 ;GET ADDR OF USW
5285 036670 116302 000001 MOV 1(3),R2 ;GET LAST BUFFER USED
5286 036674 005202 INC R2 ;UPDATE FOR NEXT BUFFER
5287 036676 042702 177770 BIC #177770,R2 ;CLEAN OFF JUNK
5288 036702 116104 000007 MOV VBM(1),R4 ;GET LAST BUFFER MASK.
5289 036706 042704 177770 BIC #177770,R4 ;CLEAN OFF JUNK
5290
5291 036712 042702 177770 30$: BIC #^C<7>,R2 ;VALID OFFSET.
5292 036716 110263 000001 MOV R2,1(3)
5293 036722 120204 CMPB R2,R4
5294 036724 003402 BLE OUPEX ;LAST BUFFER ?
5295 036726 052713 040000 BIS #40000,(3) ;NO-BRANCH
5296 036732 042777 000200 142532 OUPEX: BIC #BIT7,@LPCO ;SET STOP JOB BIT
5297
5298 036740 012604 MOV (SP)+,R4 ;CLEAR READY OUT
5299 036742 012603 MOV (SP)+,R3

```

```
5300
5301 036744 012602 MOV (SP)+,R2
5302 036746 012601 MOV (SP)+,R1
5303 036750 012600 MOV (SP)+,R0
5304 036752 000002 RTI
5305
```





```

(2) 037054 001376 BNE 25$
(2)
(2) 037056 032777 000040 142402 BIT #BITS,@KMADO ;SLAVE READY? (READING IPBM SR)
(2) 037064 001401 BEQ 3$;FATAL LPA-11 ERROR SLAVE NOT READY.
(2)
(2) 037066 104000 ERROR
(2)
(2) 037070 012777 000004 142374 3$: MOV #4,@KMAD2 ;READ FAST PATH
(2) 037076 4$:
(3) 037076 004537 040354 JSR R5, $TOUT ;-TOUT-CHECK FOR TIMEOUT
(3)
(3) 037102 104000 ERROR ;/TIME-OUT ERROR
(3) ;/WE FAILED TO COMPLETE
(3) ;/CURRENT OPERATION.
(3) ;/CONTINUES IN THIS LOOP
(3) ;/WOULD MAKE US 'HANG' HERE
(3)
(3) 037104 000774 BR 4$
(3)
(3) ;/RETURNS HERE-FROM-TIMED OUT.
(2) 037106 122777 000377 142356 CMPB #377,@KMAD2 ;WAIT TILL KMC DONE COMMAND.
(2) 037114 001370 BNE 4$
(2) 037116 122777 000377 142352 CMPB #377,@KMAD4 ;IF FAST PATH=377 THEN ERROR.
(2) 037124 001001 BNE 35$
(2) 037126 104000 ERROR ;IPBM ERROR (SLAVE SIDE)
(2) ;YOU MUST RUN IPBM DIAGNOSTIC.
(2)
(2) 037130 117737 142342 037410 35$: MOVB @KMAD4,11$;GET THE VERSION NUMBER FROM DMC-11
(2) 037136 005227 177777 INC #-1
(2) 037142 001045 BNE 5$
(2) 037144 005227 177777 INC #-1
(2) 037150 001042 BNE 5$
(3) 037152 104401 037160 TYPE ,67$;:TYPE ASCIZ STRING
(3) 037156 000426 BR 66$;:GET OVER THE ASCIZ
(3) ;:67$: .ASCIZ <200>'MB200-YC (DMC) MICROCODE VERSION NUMBER - ''
(3) 66$:
(2) 037234 013746 037410 MOV 11$,-(SP)
(2) 037240 104403 TYPOS
(2) 037242 002 000 .BYTE 2,0
(3) 037244 104401 037252 TYPE ,69$;:TYPE ASCIZ STRING
(3) 037250 000402 BR 68$;:GET OVER THE ASCIZ
(3) ;:69$: .ASCIZ <200>' ' '
(3) 68$:
(2) 037256 112737 177777 037410 5$: MOVB #0-1,11$;DAC CODE FOR SLAVE.
(2) 037264 012501 MOV (5)+,R1 ;GET NEXT DEVICE ADDR.
(2) 037266 021127 000000 6$: CMP (R1),#0 ;TERM REACHED?
(2) 037272 001444 BEQ 10$
(2) 037274 105237 037410 INCB 11$
(2) 037300 113777 037410 142170 MOVB 11$,@KMAD4 ;FIFO DATA
(2) 037306 004737 037412 JSR PC,20$;ISSUE SEND
(2) 037312 112177 142160 MOVB (R1)+,@KMAD4 ;SEND LOW BYTE OF DEVICE ADDR TO SLAVE.
(2) 037316 004737 037412 JSR PC,20$;ISSUE SEND
(2) 037322 112177 142150 MOVB (R1)+,@KMAD4 ;SEND HIGH BYTE OF DEVICE ADDR. TO SLAVE.
(2) 037326 004737 037412 JSR PC,20$
(2)

```

```

(2) 037332 032777 000002 142126 7$: BIT #BIT1,@KMADO ;WAIT FOR FIFO DATA
(2) 037340 001374 BNE 7$;=1 NO DATA. =0 DATA.
(2) 037342 112777 000002 142122 MOVB #2,@KMAD2 ;READ FIFO.
(2)
(2) 037350 8$: JSR R5, $TOUT ;-TOUT-CHECK FOR TIMEOUT
(3) 037350 004537 040354 ERROR ;/TIME-OUT ERROR
(3) ;/WE FAILED TO COMPLETE
(3) ;/CURRENT OPERATION.
(3) ;/CONTINUES IN THIS LOOP
(3) ;/WOULD MAKE US 'HANG' HERE
(3)
(3) 037356 000774 BR 8$
(3) ;/RETURNS HERE-FROM-TIMED OUT.
(2) 037360 122777 000377 142104 CMPB #377,@KMAD2 ;WAIT FOR READ.
(2) 037366 001370 BNE 8$
(2) 037370 105777 142102 TSTB @KMAD4 ;WAS A ZERO RETURNED?
(2) 037374 001734 BEQ 6$;YES GET NEXT ADDR.
(2) ;SLAVE WILL RETURN CODE 0 IF
(2) 037376 005237 037442 INC $AERR ;DEV PRESENT. ELSE
(2) ;EXIT $AERR=1 IF SLAVE GIVES ERROR.
(2) 037402 005041 CLR -(1) ;GET RID OF REFERENCE TO BAD ADDR.
(2) 037404 012601 10$: MOV (SP)+,R1
(2) 037406 000205 RTS ;RETURN ALL ADDR. CHECKED.
(2)
(2) 037410 000000 11$: .WORD 0 ;HOLDS DAC CODE PLUS OFFSET
(2) ;TO SLAVES ADDR. TABLE.
(2)
(2) 037412 112777 000003 142052 20$: MOVB #3,@KMAD2 ;ISSUE FIFO WRITE
(2) 037420 21$: JSR R5, $TOUT ;-TOUT-CHECK FOR TIMEOUT
(3) 037420 004537 040354 ERROR ;/TIME-OUT ERROR
(3) ;/WE FAILED TO COMPLETE
(3) ;/CURRENT OPERATION.
(3) ;/CONTINUES IN THIS LOOP
(3) ;/WOULD MAKE US 'HANG' HERE
(3)
(3) 037426 000774 BR 21$
(3) ;/RETURNS HERE-FROM-TIMED OUT.
(2) 037430 122777 000377 142034 CMPB #377,@KMAD2 ;KMC CODE WILL RETURN A '377'
(2) 037436 001370 BNE 21$;WHEN DONE COMMAND.
(2) 037440 000207 RTS PC
(2)
(2) 037442 000000 $AERR: .WORD 0 ;=0 IF ADDR. LIST OK,=1 IF BAD.
(2)
(2) ;*
(2) ;*THIS SUB CODE USED TO LOAD MICRO-CODE INTO LPA-11.
(2) ;* CALL = JSR R5,$LOAD
(2) ;* .WORD XX ;ADDR. OF MICRO CODE.
(2) ;* ;RETURNS HERE
(2) ;* NOTE: MICRO CODE FILE MUST END IN -1 DATA.
(2) ;*

```

```

(2)
(2) 037444 010446 $LOAD: MOV R4,-(SP) ;SAVE R4.
(2) 037446 010046 MOV R0,-(SP) ;SAVE R0.
(2) 037450 012500 1$: MOV (5)+,R0 ;GET PROG. ADDR.
(2) 037452 005077 142010 CLR @KMAD0 ;CLEAR CSR
(2) 037456 005077 142014 CLR @KMAD4 ;CLEAR CRAM ADDR.
(2) 037462 052777 002000 141776 2$: BIS #2000,@KMAD0 ;SELECT CRAM.
(2) 037470 012077 142006 MOV (0)+,@KMAD6 ;WRITE DATA.
(2) 037474 052777 020000 141764 BIS #20000,@KMAD0 ;SET CRAM WRITE
(2) 037502 005077 141760 CLR @KMAD0 ;DISABLE CRAM.
(2) 037506 005277 141764 INC @KMAD4 ;UPDATE CRAM ADDR.
(2) 037512 021027 177777 CMP (0),#-1 ;ALL DONE?
(2) 037516 001361 BNE 2$;NO LOOP.
(2) 037520 005077 141752 CLR @KMAD4 ;CLEAR CRAM ADDR.
(2) 037524 016500 177776 MOV -2(5),R0 ;GET MICRO CODE ADDR.
(2)
(2) 037530 052777 002000 141730 3$: BIS #2000,@KMAD0 ;SELECT CRAM
(2) 037536 022077 141740 CMP (R0)+,@KMAD6 ;DATA OK?
(2) 037542 001013 BNE 5$;NO - REPORT AN ERROR.
(2) 037544 021027 177777 CMP (0),#-1 ;ALL DONE?
(2) 037550 001405 BEQ 4$;YES - EXIT
(2) 037552 005077 141710 CLR @KMAD0 ;NO - DESELECT CRAM.
(2) 037556 005277 141714 INC @KMAD4 ;UPDATE CRAM ADDR.
(2) 037562 000762 BR 3$
(2)
(2) 037564 012600 4$: MOV (SP)+,R0 ;RESTORE R0
(2) 037566 012604 MOV (SP)+,R4 ;RESTORE R4
(2) 037570 000205 RTS R5 ;EXIT
(2)
(2) 037572 5$: ;COME HERE ON LOAD ERROR
(2) 037572 005745 TST -(5)
(2) 037574 105204 INCB R4 ;UPDATE ERROR COUNTER.
(2) 037576 100324 BPL 1$;IF NOT TOO MANY, TRY AGAIN.
(2) 037600 000000 HALT ;MICRO CODE LOAD ERROR.
(2)
(2) 037602 000722 BR 1$;KMC-11 FAULT. YOU COULD TRY
(2) ;TO PRESS CONTINUE TO GIVE IT
(2) ;ANOTHER CHANCE, BUT I DOUBT
(2) ;THAT THAT WOULD WORK. SINCE I'VE
(2) ;ALREADY GIVEN IT 177 (OCTAL) CHANCES.
(2) ;TRY RUNNING THE KMC-11 DIAGNOSTIC.
(2)
(2)
(2) ;*THIS ROUTINE ISSUES A WRITE COMMAND TO THE LPA-11
(2) ;*
(2) ;* CALL = JSR R5,$TLKW
(2) ;* .WORD 0 ;OFFSET OF DEVICE ADDR.
(2) ;* .WORD 0 ;DATA TO BE WRITTEN
(2)
(2) 037604 010046 $TLKW: MOV R0,-(SP) ;SAVE R0
(2) 037606 012500 MOV (5)+,R0 ;GET DEVICE OFFSET
(2) 037610 052700 000340 BIS #340,R0 ;ADD WRITE CODE.
(2) 037614 004737 040066 JSR PC,$LPW ;WAIT FOR FAST PATH READY
(2) 037620 010037 037712 MOV R0,W1
(2) 037624 010077 141646 MOV R0,@KMAD4
(2) 037630 112777 000005 141634 MOVB #5,@KMAD2 ;ISSUE FAST PATH WRITE

```



```

(3) ;/CONTINUES IN THIS LOOP
(3) ;/WOULD MAKE US 'HANG' HERE
(3) 040024 000774 BR 2$
(3)
(3) ;/RETURNS HERE-FROM-TIMED OUT.
(2) 040026 032777 000040 141432 BIT #BIT5,@KMADO ;FAST PATH READY?
(2) 040034 001370 BNE 2$
(2) 040036 112777 000004 141426 MOVB #4,@KMAD2 ;ISSUE FAST PATH READ
(2) 040044 004737 040066 JSR PC,$LPW
(2) 040050 117737 141422 040065 MOVB @KMAD4,$DATR+1 ;SAVE HIGH BYTE
(2) 040056 012600 MOV (SP)+,R0
(2) 040060 000205 RTS R5
(2) 040062 000000 RD1: 0
(2) 040064 000000 $DATR: .WORD 0
(2)
(2) ;THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
(2) ;AS FAST PATH TO BE READ.
(2)
(2) ;
(2) ; CALL = JSR PC,$LPW
(2) ;
(2) ;IT WILL TIME OUT IF TOO MUCH TIME IS TAKEN BY
(2) ;THE MICRO-PROCESSORS AND REPORT AN ERROR, THEN HALT.
(2)
(2)
(2) 040066 010146 $LPW: MOV R1,-(SP) ;SAVE R1
(2) 040070 005001 CLR R1
(2) 040072 122777 000377 141372 1$: CMPB #377,@KMAD2 ;FINISHED INSTRUCTION?
(2) 040100 001403 BEQ 2$
(2) 040102 005201 INC R1 ;TIME OUT?
(2) 040104 001372 BNE 1$
(2) 040106 000411 BR 10$
(2)
(2) 040110 032777 000020 141350 2$: BIT #BIT4,@KMADO ;FAST PATH READ?
(2) 040116 001403 BEQ 3$
(2) 040120 005201 INC R1 ;NO - TIME OUT?
(2) 040122 001372 BNE 2$
(2) 040124 000402 BR 10$;YES - REPORT AN ERROR
(2)
(2) 040126 012601 3$: MOV (SP)+,R1 ;RESTORE R1
(2) 040130 000207 RTS PC ;EXIT
(2)
(2) 040132
(3) 040132 104401 040140 10$: TYPE ,65$;:TYPE ASCIZ STRING
(3) 040136 000407 BR 64$;:GET OVER THE ASCIZ
(3) ;:65$: .ASCIZ <200>#LPA-11 FAULT#
(3) 040156 64$:
(2)
(2) 040156 000000 11$: HALT
(2) 040160 000776 BR 11$;LPA-11 FAULT RUN LPA-11
(2) ;DIAGNOSTICS.
(2)
(2)
(2) ;*
(2) ;*THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE TO
(2) ;*A DEVICE ADDRESS ON THE I/O BUSS FOR WRITE ONLY.

```

```
(2) ;*
(2) ;* FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN USED
(2) ;* BEFORE. IF NOT WE HAVE TO INITIALIZE THE LPA WITH
(2) ;* THAT ADDRESS.
(2) ;* WHEN THE ADDR. IS KNOWN BY THE LPA, DO THE OUTPUT BY
(2) ;* $TLKW
(2) ;*
(2) 040162 010046 $OUTLP: MOV R0,-(SP) ;SAVE R0
(2) 040164 010146 MOV R1,-(SP) ;SAVE R1
(2)
(2) 040166 012700 001514 MOV #.DVLS,R0 ;PROGRAM DEFINED LIST.
(2) 040172 005001 CLR R1
(2) 040174 005710 1$: TST (0) ;TERMINATOR REACHED?
(2) 040176 001421 BEQ :0$;YES NEXT STEP.
(2) 040200 027520 000000 CMP @ (5),(0)+ ;MATCH WITH ADDR IN LIST?
(2) 040204 001402 BEQ 2$
(2) 040206 005201 INC R1
(2) 040210 000771 BR 1$
(2)
(2) 040212 010137 040230 2$: MOV R1,3$;SAVE OFFSET, DEVICE KNOWN.
(2) 040216 005725 TST (5)+
(2) 040220 013537 040232 MOV @ (5)+,4$;GET DATA TO BE WRITTEN
(2) 040224 004537 037604 JSR R5,$TLKW ;DO WRITE
(2) 040230 000000 3$: .WORD 0 ;DEVICE OFFSET
(2) 040232 000000 4$: .WORD 0 ;DATA TO BE WRITTEN.
(2) 040234 012601 MOV (SP)+,R1
(2) 040236 012600 MOV (SP)+,R0
(2) 040240 000205 RTS R5
(2) 040242 017520 000000 10$: MOV @ (5),(0)+ ;SAVE ADDR.
(2) 040246 005010 CLR (0)
(2) 040250 004537 036754 JSR R5,$LPAI
(2) 040254 001514 .WORD .DVLS
(2) 040256 000755 BR 2$
(2)
(2) ;*
(2) ;* THIS ROUTINE PROVIDES THE LINKAGE FROM USER CODE
(2) ;* TO A DEVICE ADDR. ON THE I/O BUSS FOR READ ONLY.
(2) ;*
(2) ;* FIRST WE WILL DETERMINE IF THE ADDRESS HAS BEEN
(2) ;* USED BEFORE. IF NOT, WE HAVE TO INITIALIZE THE LPA
(2) ;* WITH THE NEW ADDR.
(2) ;* WHEN THE ADDR IS KNOWN WE CAN DO OUTPUT THROUGH
(2) ;* $TLKR
(2) ;* CALL THROUGH MOVEI DATA,ADDR.
(2) ;* WHICH EQUALS:
(2) ;* JSR R5,$INLP
(2) ;* .WORD XX ADDR OF DEVICE
(2) ;* .WORD YY ADDR TO STORE READ DATA.
(2) ;*
(2) 040260 010046 $INLP: MOV R0,-(SP) ;SAVE R0
(2) 040262 010146 MOV R1,-(SP) ;SAVE R1
(2)
(2) 040264 012700 001514 MOV #.DVLS,RC ;PROG DEFINED ADDR. LIST.
(2) 040270 005001 CLR R1
(2) 040272 005710 1$: TST (0) ;EOL REACHED?
```





```

(3) ;* MOV @2$,1$;/READ DEVICE REG 2$,PUT DATA IN 1$.
(2) 040426 005737 037442 TST $AERR ;IF NO ERROR,LOOP
(2) 040432 001004 BNE 10$;THERE WAS AN ERROR.
(2) 040434 062737 000002 040450 ADD #2,2$;UPDATE DEVICE ADDR.
(2) ;YOU SEE ,WE HAVE TO PROTECT OUR SELF!
(2) ;IF 2$ CONTAINED A VALID ADDR,WE
(2) ;MUST KEEP TRYING UNTIL WE GENERATE
(2) ;AN INVALID ADDR.
(2) 040442 000764 BR $RESET
(2) 040444 10$:
(2) 040444 000207 RTS PC
(2) 040446 000000 1$: .WORD 0 ;JUNK LOC.
(2) 040450 160000 2$: .WORD 160000 ;DUMB ADDR. FORCES INIT OF DMC/KMC.
(2)
(2)
(2) ;SDELAY- ROUTINE TO GIVE A MINOR DELAY.
(2) ;IS NOT TIME DEPENDENT CODE SENCE
(2) ;NOT USED TO GET SPECIFIC TIME BUT
(2) ;JUST A LITTLE DELAY.
(2)
(2) ;
(2) ; THAT IS UNLESS A REAL TIME CLOCK IS PRESENT!
(2) ; THEN WE'LL GENERATE A TIME BETWEEN 16MS TO 32 MS
(2)
(2) ;
(2) ; CALL= JSR PC, SDELAY
(2)
(2) SDELAY:
(2) 040452 TST RTCCSR ;CLOCK PRESENT?
(2) 040452 005737 040534 BPL 10$
(2) 040456 100016 MOV #2,TIME
(2) 040460 012737 000002 040524 BIS #115,@RTCCSR ;START CLOCK
(2) 040466 052777 000115 000040 CLR PS
(2) 040474 005037 177776 1$: TST TIME
(2) 040500 005737 040524 BNE 1$
(2) 040504 001375 CLR @RTCCSR ;STOP CLOCK
(2) 040506 005077 000022
(2)
(2) 040512 000207 RTS PC
(2) 040514 105237 040524 10$: INCB TIME
(2) 040520 001375 BNE 10$
(2) 040522 000207 RTS PC
(2)
(2) 040524 000000 TIME: .WORD 0
(2)
(2) 040526 005337 040524 CLKINT: DEC TIME
(2) 040532 000002 RTI
(2) 040534 000000 RTCCSR: .WORD 0 ;CLOCK CSR IF USED.
(2)
(2) ;SRESET WILL RESET WHOLE LPA SYSTEM. WITH USER MICRO-CODE.
(2) ;CALL= JSR PC,SRESET
(2)
(2) SRESET:
(2) 040536 052777 040000 140722 BIS #BIT14,@KMADO ;SET INIT BIT
(2) 040544 012700 000144 MOV #100.,R0
(2) 040550 004737 040452 1$: JSR PC,SDELAY ;DELAY
(2) 040554 005300 DEC R0
(2) 040556 001374 BNE 1$

```

|     |        |        |        |        |         |                                                          |                                    |
|-----|--------|--------|--------|--------|---------|----------------------------------------------------------|------------------------------------|
| (2) | 040560 | 012737 | 044030 | 040610 | MOV     | #MMAST,10\$                                              |                                    |
| (2) | 040566 | 122737 | 000115 | 001772 | CM PB   | #'M,\$VERSN                                              |                                    |
| (2) | 040574 | 001403 |        |        | BEQ     | 9\$                                                      |                                    |
| (2) | 040576 | 012737 | 050034 | 040610 | MOV     | #DMAST,10\$                                              |                                    |
| (2) | 040604 | 004537 | 037444 |        | 9\$:    | JSR                                                      | R5,\$LOAD ;RELOAD MICRO CODE.      |
| (2) | 040610 | 044030 |        |        | 10\$:   | .WORD                                                    | MMAST ;CURREN VERSION.             |
| (2) |        |        |        |        |         |                                                          |                                    |
| (2) | 040612 | 012777 | 104000 | 140646 | MOV     | #BIT15!BIT11,@KMADO                                      | ;START KMC.                        |
| (2) | 040620 | 012700 | 000005 |        | MOV     | #5.,R0                                                   |                                    |
| (2) | 040624 | 004737 | 040452 |        | 2\$:    | JSR                                                      | PC,\$DELAY                         |
| (2) | 040630 | 005300 |        |        | DEC     | R0                                                       |                                    |
| (2) | 040632 | 001374 |        |        | BNE     | 2\$                                                      |                                    |
| (2) | 040634 | 005077 | 140632 |        | CLR     | @KMAD2                                                   |                                    |
| (2) | 040640 | 005077 | 140632 |        | CLR     | @KMAD4                                                   |                                    |
| (2) | 040644 | 000207 |        |        | RTS     | PC                                                       |                                    |
| (2) |        |        |        |        | :*      |                                                          |                                    |
| (2) |        |        |        |        | :       | *THIS MACRO ALLOWS THE OPERATOR TO TALK TO               |                                    |
| (2) |        |        |        |        | :       | *ANY DEVICE ON THE I/O BUS                               |                                    |
| (2) |        |        |        |        | :       | *USER MUST START AT THIS ADDR.                           |                                    |
| (2) |        |        |        |        | :       | *HE MUST SAY EITHER 'E' FOR EXAMINE, OR 'D' FOR DEPOSIT. |                                    |
| (2) |        |        |        |        | :       | *'E' IS DEFAULT.                                         |                                    |
| (2) |        |        |        |        | :       | *NEXT, HE MUST SUPPLY AN ADDR.                           |                                    |
| (2) |        |        |        |        | :       | *NOTE IF ADDR. IS NOT FOUND ON I/O BUS, A HALT           |                                    |
| (2) |        |        |        |        | :       | *WILL OCCUR.                                             |                                    |
| (2) |        |        |        |        | \$UTK:  |                                                          |                                    |
| (2) | 040646 |        |        |        | CLR     | .DVL\$                                                   |                                    |
| (2) | 040646 | 005037 | 001514 |        | 21\$:   |                                                          |                                    |
| (2) | 040652 |        |        |        | TYPE    | .65\$                                                    | ::TYPE ASCIZ STRING                |
| (3) | 040652 | 104401 | 040660 |        | BR      | 64\$                                                     | ::GET OVER THE ASCIZ               |
| (3) | 040656 | 000405 |        |        | ::65\$: | .ASCIZ                                                   | <200>#E OR D?#                     |
| (3) | 040672 |        |        |        | 64\$:   |                                                          |                                    |
| (2) | 040672 | 105777 | 140246 |        | 1\$:    | TSTB                                                     | @\$TKS                             |
| (2) | 040676 | 100375 |        |        | BPL     | 1\$                                                      |                                    |
| (2) | 040700 | 117737 | 140242 | 041022 | MOV B   | @\$TKB,20\$                                              | ;GET INPUT                         |
| (2) | 040706 | 104401 | 041022 |        | TYPE    | 20\$                                                     | ;ECHO, NEXT MESSAGE.               |
| (2) | 040712 | 142737 | 000240 | 041022 | BIC B   | #240,20\$                                                | ;STRIP PARITY, LC                  |
| (2) | 040720 | 104412 |        |        | RDOCT   |                                                          | ;GET ADDR.                         |
| (2) | 040722 | 012637 | 041020 |        | MOV     | (SP)+,14\$                                               |                                    |
| (2) | 040726 | 123727 | 041022 | 000104 | CM PB   | 20\$,#D                                                  | ;DEPOSIT?                          |
| (2) | 040734 | 001411 |        |        | BEQ     | 10\$                                                     |                                    |
| (2) |        |        |        |        |         |                                                          |                                    |
| (2) | 040736 | 004537 | 040260 |        | JSR     | R5,\$INLP                                                | ;GET DATA                          |
| (2) | 040742 | 041020 |        |        | 2\$:    | .WORD                                                    | 14\$                               |
| (2) | 040744 | 040756 |        |        | .WORD   | 5\$                                                      |                                    |
| (2) |        |        |        |        |         |                                                          |                                    |
| (3) | 040746 | 013746 | 040756 |        | MOV     | 5\$,-(SP)                                                | ::SAVE 5\$ FOR TYPEOUT             |
| (3) | 040752 | 104402 |        |        | TYPOC   |                                                          | ::GO TYPE--OCTAL ASCII(ALL DIGITS) |
| (2) | 040754 | 000736 |        |        | BR      | 21\$                                                     | ;LOOP.                             |
| (2) | 040756 | 000000 |        |        | 5\$:    | .WORD                                                    | 0                                  |
| (2) |        |        |        |        |         |                                                          |                                    |
| (2) | 040760 |        |        |        | 10\$:   |                                                          |                                    |
| (3) | 040760 | 104401 | 040766 |        | TYPE    | .67\$                                                    | ::TYPE ASCIZ STRING                |
| (3) | 040764 | 000404 |        |        | BR      | 66\$                                                     | ::GET OVER THE ASCIZ               |
| (3) |        |        |        |        | ::67\$: | .ASCIZ                                                   | <200>#DATA- #                      |



J 14

5308

.SBTTL TRAP DECODER

(1)  
(2)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)

```

;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;*GO TO THAT ROUTINE.
```

(1) 041152 010046  
(1) 041154 016600 000002  
(1) 041160 005740  
(1) 041162 111000  
(1) 041164 006300  
(1) 041166 016000 041206  
(1) 041172 000200

```
$TRAP: MOV R0,-(SP) ;;SAVE R0
 MOV 2(SP),R0 ;;GET TRAP ADDRESS
 TST -(R0) ;;BACKUP BY 2
 MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
 ASL R0 ;;POSITION FOR INDEXING
 MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
 RTS R0 ;;GO TO ROUTINE
```

(1)  
(1)  
(1)

```
;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
```

(1) 041174 011646  
(1) 041176 016666 000004 000002  
(1) 041204 000002

```
$TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
 RTI ;;RESTORE THE PSW
```

(1)  
(3)  
(3)

.SBTTL TRAP TABLE

(3)  
(3)  
(3)  
(3)

```
;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;*BY THE 'TRAP' INSTRUCTION.
```

(3) 041206 041174  
(3) 041210 033110  
(3) 041212 031074  
(3) 041214 031050  
(3) 041216 031110  
(3) 041220 031276  
(1)  
(3) 041222 032404  
(1)  
(3) 041224 032334  
(3) 041226 032616  
(3) 041230 032736  
(3) 041232 033372

```
ROUTINE

$TRPAD: .WORD $TRAP2
 $TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
 $TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIM. NUMBER (WITH SIGN)

$GTSWR ;;CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING

$CKSWR ;;CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
$RDCHR ;;CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
$RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
$RDOCT ;;CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
```

5309

5310

5311

5312 041234  
(1) 041234 104401 041242  
(1) 041240 000410  
(1)  
(1) 041262  
5313 041262 104412  
5314 041264 012600  
5315 041266 001002  
5316 041270 012700 056156  
5317 041274

```
;;ROUTINE TO READ NUMBERS FROM A BUFFER
```

```
RBUFR: TYPE 65$;;TYPE ASCIZ STRING
 BR 64$;;GET OVER THE ASCIZ
 .ASCIZ <200>#BUFFER START? #

 RDOCT
 MOV (SP)+,R0
 BNE 1$
 MOV #BUFFER,R0

1$:
```

```

(1) 041274 104401 041302 TYPE ,67$;;TYPE ASCIZ STRING
(1) 041300 000406 BR 66$;;GET OVER THE ASCIZ
(1) ;;67$: .ASCIZ #CHAN INC? #
(1) 041316 66$:
5318 041316 104412 RDOCT
5319 041320 012601 MOV (SP)+,R1
5320 041322 001002 BNE 2$
5321 041324 012701 000020 MOV #16.,R1
5322 041330 006301 2$: ASL R1
5323 041332 104401 041340 TYPE ,69$;;TYPE ASCIZ STRING
(1) 041336 000407 BR 68$;;GET OVER THE ASCIZ
(1) ;;69$: .ASCIZ <200>#ADDR. DATA#
(1) 041356 68$:
5324 041356 3$:
(1) 041356 104401 041364 TYPE ,71$;;TYPE ASCIZ STRING
(1) 041362 000401 BR 70$;;GET OVER THE ASCIZ
(1) ;;71$: .ASCIZ <200>##
(1) 041366 70$:
5325 041366 010046 MOV R0,-(SP) ;;SAVE R0 FOR TYPEOUT
(1) 041370 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
5326 041372 104401 041400 TYPE ,73$;;TYPE ASCIZ STRING
(1) 041376 000402 BR 72$;;GET OVER THE ASCIZ
(1) ;;73$: .ASCIZ # #
(1) 041404 72$:
5327 041404 011046 MOV (0),-(SP) ;;SAVE (0) FOR TYPEOUT
(1) 041406 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
5328 041410 060100 ADD R1,R0
5329 041412 000761 BR 3$
5330
5331 .SBTTL .ASCIZ MESSAGES.
5332 041414 046200 040520 040440 EM1: .ASCIZ <200>#LPA ADDRESS ERROR#
041422 042104 042522 051523
041430 042440 051122 051117
041436 000
5333
5334 041437 200 050114 020101 EM2: .ASCIZ <200>#LPA (KMC-11) DATA ERROR#
041444 045450 041515 030455
041452 024461 042040 052101
041460 020101 051105 047522
041466 000122
5335
5336 041470 046200 040520 024040 EM3: .ASCIZ <200>#LPA (KMC-11) INSTRUCTION ERROR#
041476 046513 026503 030461
041504 020051 047111 052123
041512 052522 052103 047511
041520 020116 051105 047522
041526 000122
5337
5338 041530 046200 040520 024040 EM4: .ASCIZ <200>#LPA (M8254) INIT ERROR#
041536 034115 032462 024464
041544 044440 044516 020124
041552 051105 047522 000122
5339
5340 041560 046200 040520 024040 EM6: .ASCIZ <200>#LPA (M8254) SILO DATA ERROR#
041566 034115 032462 024464
041574 051440 046111 020117

```

|      |        |        |        |        |       |                                                   |
|------|--------|--------|--------|--------|-------|---------------------------------------------------|
|      | 041602 | 040504 | 040524 | 042440 |       |                                                   |
|      | 041610 | 051122 | 051117 | 000    |       |                                                   |
| 5341 |        |        |        |        |       |                                                   |
| 5342 | 041615 | 200    | 050114 | 020101 | EM7:  | .ASCIZ <200>#LPA (M8254) FAST PATH DATA ERROR#    |
|      | 041622 | 046450 | 031070 | 032065 |       |                                                   |
|      | 041630 | 020051 | 040506 | 052123 |       |                                                   |
|      | 041636 | 050040 | 052101 | 020110 |       |                                                   |
|      | 041644 | 040504 | 040524 | 042440 |       |                                                   |
|      | 041652 | 051122 | 051117 | 000    |       |                                                   |
| 5343 |        |        |        |        |       |                                                   |
| 5344 | 041657 | 200    | 050114 | 020101 | EM10: | .ASCIZ <200>#LPA (AD11K) CSR ERROR#               |
|      | 041664 | 040450 | 030504 | 045461 |       |                                                   |
|      | 041672 | 020051 | 051503 | 020122 |       |                                                   |
|      | 041700 | 051105 | 047522 | 000122 |       |                                                   |
| 5345 |        |        |        |        |       |                                                   |
| 5346 | 041706 | 046200 | 040520 | 024040 | EM11: | .ASCIZ <200>#LPA (KW11K) CSR ERROR#               |
|      | 041714 | 053513 | 030461 | 024513 |       |                                                   |
|      | 041722 | 041440 | 051123 | 042440 |       |                                                   |
|      | 041730 | 051122 | 051117 | 000    |       |                                                   |
| 5347 |        |        |        |        |       |                                                   |
| 5348 | 041735 | 200    | 050114 | 020101 | EM12: | .ASCIZ <200>#LPA (DR11K) ERROR#                   |
|      | 041742 | 042050 | 030522 | 045461 |       |                                                   |
|      | 041750 | 020051 | 051105 | 047522 |       |                                                   |
|      | 041756 | 000122 |        |        |       |                                                   |
| 5349 |        |        |        |        |       |                                                   |
| 5350 | 041760 | 046200 | 040520 | 024040 | EM14: | .ASCIZ <200>#LPA (AA-11K) ERROR#                  |
|      | 041766 | 040501 | 030455 | 045461 |       |                                                   |
|      | 041774 | 020051 | 051105 | 047522 |       |                                                   |
|      | 042002 | 000122 |        |        |       |                                                   |
| 5351 |        |        |        |        |       |                                                   |
| 5352 | 042004 | 046200 | 040520 | 024040 | EM15: | .ASCIZ <200>#LPA (AR11) ERROR#                    |
|      | 042012 | 051101 | 030461 | 020051 |       |                                                   |
|      | 042020 | 051105 | 047522 | 000122 |       |                                                   |
| 5353 |        |        |        |        |       |                                                   |
| 5354 | 042026 | 046200 | 040520 | 024040 | EM16: | .ASCIZ <200>#LPA (LPS-11) ERROR#                  |
|      | 042034 | 050114 | 026523 | 030461 |       |                                                   |
|      | 042042 | 020051 | 051105 | 047522 |       |                                                   |
|      | 042050 | 000122 |        |        |       |                                                   |
| 5355 |        |        |        |        |       |                                                   |
| 5356 | 042052 | 046200 | 040520 | 030455 | EM17: | .ASCIZ <200>#LPA-11 FUNCTIONAL ERROR#             |
|      | 042060 | 020061 | 052506 | 041516 |       |                                                   |
|      | 042066 | 044524 | 047117 | 046101 |       |                                                   |
|      | 042074 | 042440 | 051122 | 051117 |       |                                                   |
|      | 042102 | 000    |        |        |       |                                                   |
| 5357 |        |        |        |        |       |                                                   |
| 5358 | 042103 | 200    | 042522 | 042520 | EM20: | .ASCIZ <200>#REPEATIBILITY OR CHAN AVERAGE ERROR# |
|      | 042110 | 052101 | 041111 | 046111 |       |                                                   |
|      | 042116 | 052111 | 020131 | 051117 |       |                                                   |
|      | 042124 | 041440 | 040510 | 020116 |       |                                                   |
|      | 042132 | 053101 | 051105 | 043501 |       |                                                   |
|      | 042140 | 020105 | 051105 | 047522 |       |                                                   |
|      | 042146 | 000122 |        |        |       |                                                   |
| 5359 |        |        |        |        |       |                                                   |
| 5360 | 042150 | 042600 | 051122 | 041520 | DM1:  | .ASCIZ <200>#ERRPC ADDRESS#                       |
|      | 042156 | 020040 | 040440 | 042104 |       |                                                   |
|      | 042164 | 042522 | 051523 | 000    |       |                                                   |

5361  
5362 042171 200 051105 050122 DH2: .ASCIZ <200>#ERRPC EXP'ED REC'ED#  
042176 020103 020040 054105  
042204 023520 042105 020040  
042212 051040 041505 042447  
042220 000104  
5363  
5364 042222 042600 051122 041520 DH3: .ASCIZ <200>#ERRPC#  
042230 000  
5365 042231 200 051105 050122 DH17: .ASCIZ <200>#ERRPC TESTNO EXP'ED REC'ED#  
042236 020103 020040 042524  
042244 052123 047516 020040  
042252 054105 023520 042105  
042260 020040 042522 023503  
042266 042105 000  
5366  
5367 042271 200 042524 052123 DH20: .ASCIZ <200>#TEST S/B CHAN HIGH LOW#  
042276 020040 020040 027523  
042304 020102 020040 020040  
042312 041440 040510 020116  
042320 020040 044040 043511  
042326 020110 020040 046040  
042334 053517 000  
5368 042337 200 051105 050122 DH21: .ASCIZ <200>#ERRPC TSTNO USJNO USRDA ALPCO ALPCI ALPSO#  
042344 020103 020040 051524  
042352 047124 020117 020040  
042360 051525 047112 020117  
042366 020040 051525 042122  
042374 020101 020040 046101  
042402 041520 020117 020040  
042410 046101 041520 020111  
042416 020040 046101 051520  
042424 000117  
5369  
5370  
5371 042426 001116 001466 000000 DT1: .EVEN \$ERRPC,KMADO,0  
5372  
5373 042434 001116 001124 001126 DT2: .WORD \$ERRPC,\$GDDAT,\$BDDAT,0  
042442 000000  
5374  
5375 042444 001116 000000 DT3: .WORD \$ERRPC,0  
5376 042450 001116 001200 001124 DT17: .WORD \$ERRPC,\$TESTN,\$GDDAT,\$BDDAT,0  
042456 001126 000000  
5377  
5378 042462 001200 034614 034566 DT20: \$TESTN,AVEXP,CHAN,RHIGH,RLOW,0  
042470 034622 034620 000000  
5379  
5380 042476 001116 001200 001756 DT21: \$ERRPC,\$TESTN,USJNO,ALPADL,ALPCO,ALPCI,ALPSO,0  
042504 001770 001764 001762  
042512 001766 000000  
5381 042516 000000 000000 DF0: .WORD 0.0  
5382  
5383 042522 020040 000 M2SP: .ASCIZ # #  
5384  
5385  
5386 042525 200 040515 042513 M1: .ASCII <200>'MAKE YOUR OWN JOB(S) ROUTINE FOR LPA-11'

|      |        |        |        |        |     |                                                                                  |
|------|--------|--------|--------|--------|-----|----------------------------------------------------------------------------------|
|      | 042532 | 054440 | 052517 | 020122 |     |                                                                                  |
|      | 042540 | 053517 | 020116 | 047512 |     |                                                                                  |
|      | 042546 | 024102 | 024523 | 051040 |     |                                                                                  |
|      | 042554 | 052517 | 044524 | 042516 |     |                                                                                  |
|      | 042562 | 043040 | 051117 | 046040 |     |                                                                                  |
|      | 042570 | 040520 | 030455 | 061    |     |                                                                                  |
| 5387 | 042575 | 200    | 046120 | 040505 |     | .ASCII <200>'PLEASE ANSWER ALL QUESTIONS, IF IN DOUBT, REFERR TO DOCUMENTATION'' |
|      | 042602 | 042523 | 040440 | 051516 |     |                                                                                  |
|      | 042610 | 042527 | 020122 | 046101 |     |                                                                                  |
|      | 042616 | 020114 | 052521 | 051505 |     |                                                                                  |
|      | 042624 | 044524 | 047117 | 026123 |     |                                                                                  |
|      | 042632 | 044440 | 020106 | 047111 |     |                                                                                  |
|      | 042640 | 042040 | 052517 | 052102 |     |                                                                                  |
|      | 042646 | 020054 | 042522 | 042506 |     |                                                                                  |
|      | 042654 | 051122 | 052040 | 020117 |     |                                                                                  |
|      | 042667 | 047504 | 052503 | 042515 |     |                                                                                  |
|      | 042670 | 052116 | 052101 | 047511 |     |                                                                                  |
|      | 042676 | 116    |        |        |     |                                                                                  |
| 5388 | 042677 | 200    | 042200 | 042105 |     | .ASCIZ <200><200>'DEDICTED OR MULTIUSER (D OR M)''                               |
|      | 042704 | 041511 | 042524 | 020104 |     |                                                                                  |
|      | 042712 | 051117 | 046440 | 046125 |     |                                                                                  |
|      | 042720 | 044524 | 051525 | 051105 |     |                                                                                  |
|      | 042726 | 024040 | 020104 | 051117 |     |                                                                                  |
|      | 042734 | 046440 | 000051 |        |     |                                                                                  |
| 5389 |        |        |        |        |     |                                                                                  |
| 5390 | 042740 | 051600 | 052105 | 041440 | M2: | .ASCIZ <200>'SET CLOCK CSR= ''                                                   |
|      | 042746 | 047514 | 045503 | 041440 |     |                                                                                  |
|      | 042754 | 051123 | 020075 | 000    |     |                                                                                  |
| 5391 |        |        |        |        |     |                                                                                  |
| 5392 | 042761 | 200    | 042523 | 020124 | M3: | .ASCIZ <200>'SET PRESENT BUFFER (2'S COMP) TO ''                                 |
|      | 042766 | 051120 | 051505 | 047105 |     |                                                                                  |
|      | 042774 | 020124 | 052502 | 043106 |     |                                                                                  |
|      | 043002 | 051105 | 024040 | 023462 |     |                                                                                  |
|      | 043010 | 020123 | 047503 | 050115 |     |                                                                                  |
|      | 043016 | 020051 | 047524 | 000040 |     |                                                                                  |
| 5393 |        |        |        |        |     |                                                                                  |
| 5394 | 043024 | 044200 | 053517 | 046440 | M4: | .ASCIZ <200>'HOW MANY JOBS ''                                                    |
|      | 043032 | 047101 | 020131 | 047512 |     |                                                                                  |
|      | 043040 | 051502 | 000040 |        |     |                                                                                  |
| 5395 |        |        |        |        |     |                                                                                  |
| 5396 | 043044 | 045200 | 041117 | 021440 | M5: | .ASCIZ <200>'JOB # ''                                                            |
|      | 043052 | 000040 |        |        |     |                                                                                  |
| 5397 | 043054 | 051440 | 040524 | 051524 | M6: | .ASCIZ '' STATS:''                                                               |
|      | 043062 | 000072 |        |        |     |                                                                                  |
| 5398 | 043064 | 044600 | 047457 | 042040 | M7: | .ASCIZ <200>'I/O DEVICE START MODE WORD= ''                                      |
|      | 043072 | 053105 | 041511 | 020105 |     |                                                                                  |
|      | 043100 | 052123 | 051101 | 020124 |     |                                                                                  |
|      | 043106 | 047515 | 042504 | 053440 |     |                                                                                  |
|      | 043114 | 051117 | 036504 | 000040 |     |                                                                                  |
| 5399 | 043122 | 041200 | 043125 | 042506 | M8: | .ASCIZ <200>'BUFFER SIZE (2'S COMP)= ''                                          |
|      | 043130 | 020122 | 044523 | 042532 |     |                                                                                  |
|      | 043136 | 024040 | 023462 | 020123 |     |                                                                                  |
|      | 043144 | 047503 | 050115 | 036451 |     |                                                                                  |
|      | 043152 | 000040 |        |        |     |                                                                                  |
| 5400 | 043154 | 052600 | 042523 | 020122 | M9: | .ASCIZ <200>'USER STATUS WORD ADDR= ''                                           |
|      | 043162 | 052123 | 052101 | 051525 |     |                                                                                  |



|      |        |        |        |        |      |                                                                          |
|------|--------|--------|--------|--------|------|--------------------------------------------------------------------------|
|      | 043170 | 053440 | 051117 | 020104 |      |                                                                          |
|      | 043176 | 042101 | 051104 | 020075 |      |                                                                          |
|      | 043204 | 000    |        |        |      |                                                                          |
| 5401 | 043205 | 200    | 047510 | 020127 | M97: | .ASCII <200>'HOW MANY TIMES TO FILL BUFFERS (NORM =1) ''                 |
|      | 043212 | 040515 | 054516 | 052040 |      |                                                                          |
|      | 043220 | 046511 | 051505 | 052040 |      |                                                                          |
|      | 043226 | 020117 | 044506 | 046114 |      |                                                                          |
|      | 043234 | 041040 | 043125 | 042506 |      |                                                                          |
|      | 043242 | 051522 | 024040 | 047516 |      |                                                                          |
|      | 043250 | 046522 | 036440 | 024461 |      |                                                                          |
|      | 043256 | 040    |        |        |      |                                                                          |
| 5402 | 043257 | 200    | 047510 | 020127 | M10: | .ASCIZ <200>'HOW MANY BUFFERS ? ''                                       |
|      | 043264 | 040515 | 054516 | 041040 |      |                                                                          |
|      | 043272 | 043125 | 042506 | 051522 |      |                                                                          |
|      | 043300 | 037440 | 000040 |        |      |                                                                          |
| 5403 | 043304 | 044600 | 020123 | 042504 | M11: | .ASCIZ <200>'IS DEVICE OVERRUN FATAL?''                                  |
|      | 043312 | 044526 | 042503 | 047440 |      |                                                                          |
|      | 043320 | 042526 | 051122 | 047125 |      |                                                                          |
|      | 043326 | 043040 | 052101 | 046101 |      |                                                                          |
|      | 043334 | 000077 |        |        |      |                                                                          |
| 5404 | 043336 | 044600 | 020123 | 052502 | M12: | .ASCIZ <200>'IS BUFFER OVERRUN FATAL?''                                  |
|      | 043344 | 043106 | 051105 | 047440 |      |                                                                          |
|      | 043352 | 042526 | 051122 | 047125 |      |                                                                          |
|      | 043360 | 043040 | 052101 | 046101 |      |                                                                          |
|      | 043366 | 000077 |        |        |      |                                                                          |
| 5405 | 043370 | 044600 | 051516 | 043125 | M13: | .ASCIZ <200>'INSUFFICIENT CORE FOR BUFFERS-RETYPE THIS JOBS PRAMETERS?'' |
|      | 043376 | 044506 | 047105 | 020124 |      |                                                                          |
|      | 043404 | 047503 | 042522 | 043040 |      |                                                                          |
|      | 043412 | 051117 | 041040 | 043125 |      |                                                                          |
|      | 043420 | 042506 | 051522 | 051055 |      |                                                                          |
|      | 043426 | 052105 | 050131 | 020105 |      |                                                                          |
|      | 043434 | 044124 | 051511 | 045040 |      |                                                                          |
|      | 043442 | 041117 | 020123 | 051120 |      |                                                                          |
|      | 043450 | 046501 | 052105 | 051105 |      |                                                                          |
|      | 043456 | 037523 | 000    |        |      |                                                                          |
| 5406 | 043461 | 200    | 042504 | 040514 | M14: | .ASCIZ <200>'DELAY BEFORE START''                                        |
|      | 043466 | 020131 | 042502 | 047506 |      |                                                                          |
|      | 043474 | 042522 | 051440 | 040524 |      |                                                                          |
|      | 043502 | 052122 | 000    |        |      |                                                                          |
| 5407 | 043505 | 200    | 052516 | 041115 | M15: | .ASCIZ <200>'NUMBER OF CHNNELS?''                                        |
|      | 043512 | 051105 | 047440 | 020106 |      |                                                                          |
|      | 043520 | 044103 | 047116 | 046105 |      |                                                                          |
|      | 043526 | 037523 | 000    |        |      |                                                                          |
| 5408 | 043531 | 200    | 044514 | 052123 | M16: | .ASCIZ <200>'LIST RANDOM CHANNELS''                                      |
|      | 043536 | 051040 | 047101 | 047504 |      |                                                                          |
|      | 043544 | 020115 | 044103 | 047101 |      |                                                                          |
|      | 043552 | 042516 | 051514 | 000    |      |                                                                          |
| 5409 | 043557 | 200    | 040523 | 050115 | M17: | .ASCIZ <200>'SAMPLE#''                                                   |
|      | 043564 | 042514 | 000043 |        |      |                                                                          |
| 5410 | 043570 | 046600 | 045501 | 020105 | M18: | .ASCIZ <200>'MAKE BUFFER NOW?''                                          |
|      | 043576 | 052502 | 043106 | 051105 |      |                                                                          |
|      | 043604 | 047040 | 053517 | 000077 |      |                                                                          |
| 5411 | 043612 | 047200 | 053505 | 047440 | M0:  | .ASCIZ <200>'NEW OR OLD DATA?''                                          |
|      | 043620 | 020122 | 046117 | 020104 |      |                                                                          |
|      | 043626 | 040504 | 040524 | 000077 |      |                                                                          |
| 5412 | 043634 | 050200 | 042522 | 051523 | M19: | .ASCIZ <200>'PRESS CONTINUE WHEN READY''                                 |

|      |        |        |        |        |      |        |                                 |
|------|--------|--------|--------|--------|------|--------|---------------------------------|
|      | 043642 | 041440 | 047117 | 044524 |      |        |                                 |
|      | 043650 | 052516 | 020105 | 044127 |      |        |                                 |
|      | 043656 | 047105 | 051040 | 040505 |      |        |                                 |
|      | 043664 | 054504 | 000    |        |      |        |                                 |
| 5413 | 043667 | 200    | 044103 | 047101 | M20: | .ASCIZ | <200>'CHAN ADDRESS WORD?'       |
|      | 043674 | 040440 | 042104 | 042522 |      |        |                                 |
|      | 043702 | 051523 | 053440 | 051117 |      |        |                                 |
|      | 043710 | 037504 | 000    |        |      |        |                                 |
| 5414 | 043713 | 200    | 046103 | 041517 | M21: | .ASCIZ | <200>'CLOCK RATE?'              |
|      | 043720 | 020113 | 040522 | 042524 |      |        |                                 |
|      | 043726 | 000077 |        |        |      |        |                                 |
| 5415 | 043730 | 051600 | 040524 | 052122 | M22: | .ASCIZ | <200>'START/EVENT MARK WORD? '' |
|      | 043736 | 042457 | 042526 | 052116 |      |        |                                 |
|      | 043744 | 046440 | 051101 | 020113 |      |        |                                 |
|      | 043752 | 047527 | 042122 | 020077 |      |        |                                 |
|      | 043760 | 000    |        |        |      |        |                                 |
| 5416 | 043761 | 200    | 052123 | 051101 | M23: | .ASCIZ | <200>'START MASK? ''            |
|      | 043766 | 020124 | 040515 | 045523 |      |        |                                 |
|      | 043774 | 020077 | 000    |        |      |        |                                 |
| 5417 | 043777 | 200    | 053105 | 047105 | M24: | .ASCIZ | <200>'EVENT MARK MASK? ''       |
|      | 044004 | 020124 | 040515 | 045522 |      |        |                                 |
|      | 044012 | 046440 | 051501 | 037513 |      |        |                                 |
|      | 044020 | 000040 |        |        |      |        |                                 |
| 5418 |        |        |        |        |      |        |                                 |
| 5419 |        |        |        |        |      | .EVEN  |                                 |

```
5478 ;**
5479 ;** REQUEST DECIPTOR ARRAY FOR THE KW-11K CLOCK
5480 ;** THIS TABLE WILL BE ALTERED BY THIS PROGRAM
5481 ;** AND BY THE LPA-11 OPTION.
5482 044022 000000 KWT: .WORD 0 ;CLOCK START MODE WORD.
5483 044024 000000 .WORD 0 ;CLOCK PRESET REGISTER.
5484 044026 000000 .WORD 0 ;CLOCK BUFFER REGISTER.
5485
5486 ;***** HERE LIES THE MULTI-USER MICRO-CODE IN OCTAL
5487 ;***** FOLLOWED BY THE DEDICATED A TO D MICRO-CODE
5488
5489 044030 MMAST: ;MULTI-USER MICRO-CODE
5490
5495 050034 DDMAST: ;DEDICATED A TO D MICRO-CODE
5500
5501 ;***** UNTIL HERE, ITS LONG AND ONLY OCTAL NUMBERS
5502
5503 054040 000000 DEVLST: .WORD 0 ;MODE WORD FOR START.
(1) 054042 000012 .BLKW 10. ;TEN ADDRESSES ON START.
(1) 054066 DMDT:
(1) .TITLE DMDT
(1) .SBTTL DMDT -- DEDICATED MODE DISPATCH TABLE
(1) .IDENT /LPA.03/
(1)
(1) :
(1) : COPYRIGHT 1976, 1977 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.
(1) :
(1) : THIS SOFTWARE IS FURNISHED TO PURCHASER UNDER A LICENSE FOR USE
(1) : ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION
(1) : OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM,
(1) : EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.
(1) :
(1) : THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT
(1) : NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
(1) : EQUIPMENT CORPORATION.
(1) :
(1) : DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF
(1) : ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
(1) :
(1) : CHARLES A. SAMUELSON
(1) : FEBRUARY 4, 1977
(1) :
(1) : +
(1) : DMDT -- DEDICATED MODE DISPATCH TABLE
(1) : TABLE FOR LPA11 MICRO PROCESSOR DEDICATED MODE SAMPLING
(1) : -
(1) :
(1) : DEFINED VALUES
(1) :
(1) 000022 DMDSIZ=18. ;LENGTH OF DMDT BUFFER IN BYTES
(1) 000100 SDT=100 ;SLAVE DISPATCH TABLE START ADDRESS
(1) 000000 CAINC=0 ;CHANNEL ADDRESS INCREMENT VALUE
(1) 000000 AD1SRL=0 ;ADC #1 STATUS REGISTER ADDRESS LOW BYTE
(1) 000120 SEX=120 ;SELECT EXTERNAL CLOCK START AND INTERRUPT ENABLE
(1) 000035 RONPR=35 ;REQUEST OUTPUT NPR IN MICRO-PROCESSOR
```

```

(1) 000000 CLR=0 ;CLEAR AD STATUS REGISTER
(1) 000235 RONPRL=235 ;REQUEST OUTPUT NPR LOW BYTE IN MICRO-PROCESSOR
(1) 000002 AD1DRL=2 ;ADC #1 DATA REGISTER ADDRESS LOW BYTE
(1) 000015 RINPR=15 ;REQUEST INPUT NPR
(1) 000040 SCS=40 ;SELECT CLOCK OVERFLOW START FOR ADC'S
(1) 000040 AD2SRL=40 ;ADC #2 STATUS REGISTER ADDRESS LOW BYTE
(1) 000042 AD2DRL=42 ;ADC #2 DATA REGISTER ADDRESS LOW BYTE
(1) 000001 AD1SRH=1 ;ADC #1 STATUS REGISTER HIGH BYTE ADDRESS
(1) 000041 AD2SRH=41 ;ADC #2 STATUS REGISTER HIGH BYTE ADDRESS
(1) 000020 SEN=20 ;SELECT EXTERNAL START, NO INTERRUPT ENABLE
(1)
(1)
(1) 054066 DMDT::
(1) 054066 D.OES:: ;ONE ADC, EXTERNAL TRIGGER, SINGLE CHANNEL
(1) 054066 162 000 120 .BYTE DMDSIZ+<3*40>,AD1SRL,SEX,RONPR,AD1DRL,RINPR,CLR,RONPRL,SDT
(1) 054071 035 002 015
(1) 054074 000 235 100
(1) 054111 .=D.OES+23
(1) 054111 D.OEQ:: ;ONE ADC, EXTERNAL TRIGGER, SEQUENTIAL CHANNEL
(1) 054111 162 000 000 .BYTE DMDSIZ+<3*40>,CAINC,AD1SRL,SEX,RONPR,AD1DRL,RINPR,CLR,RONPRL,SDT
(1) 054114 120 035 002
(1) 054117 015 000 235
(1) 054122 100
(1) 054123 000 .BYTE CAINC
(1) 054134 .=D.OEQ+23
(1) 054134 D.OCS:: ;ONE ADC, CLOCK TRIGGER, SINGLE CHANNEL
(1) 054134 162 000 040 .BYTE DMDSIZ+<3*40>,AD1SRL,SCS,RONPR,AD1DRL,RINPR,CLR,RONPRL,SDT
(1) 054137 035 002 015
(1) 054142 000 235 100
(1) 054157 .=D.OCS+23
(1) 054157 D.OCQ:: ;ONE ADC, CLOCK TRIGGER, SEQUENTIAL CHANNEL
(1) 054157 162 000 000 .BYTE DMDSIZ+<3*40>,CAINC,AD1SRL,SCS,RONPR,AD1DRL,RINPR,CLR,RONPRL,SDT
(1) 054162 040 035 002
(1) 054165 015 000 235
(1) 054170 100
(1) 054171 000 .BYTE CAINC
(1) 054202 .=D.OCQ+23
(1) 054202 D.TES:: ;TWO ADC, EXTERNAL TRIGGER, SINGLE CHANNEL
(1) 054202 162 000 120 .BYTE DMDSIZ+<3*40>,AD1SRL,SEX,RONPR,AD1DRL,RINPR,CLR,RONPRL,SDT+10
(1) 054205 035 002 015
(1) 054210 000 235 110
(1) 054213 040 120 035 .BYTE AD2SRL,SEX,RONPR,AD2DRL,RINPR,CLR,RONPRL,SDT
(1) 054216 042 015 000
(1) 054221 235 100
(1) 054225 .=D.TES+23
(1) 054225 D.TEQ:: ;TWO ADC, EXTERNAL TRIGGER, SEQUENTIAL CHANNEL
(1) 054225 162 000 000 .BYTE DMDSIZ+<3*40>,CAINC,AD1SRL,SEX,RONPR,AD1DRL,RINPR,CLR,RONPRL
(1) 054230 120 035 002
(1) 054233 015 000 235
(1) 054236 111 000 040 .BYTE SDT+11,CAINC,AD2SRL,SEX,RONPR,AD2DRL,RINPR,CLR,RONPRL,SDT
(1) 054241 120 035 042
(1) 054244 015 000 235
(1) 054247 100
(1) 054250 .=D.TEQ+23
(1) 054250 D.TCS:: ;TWO ADC, CLOCK TRIGGER, SINGLE CHANNEL
(1) 054250 262 000 040 .BYTE DMDSIZ+<5*40>,AD1SRL,SCS,RONPR,AD1DRL,RINPR,CLR,RONPRL

```

|      |        |        |     |     |                                                                    |
|------|--------|--------|-----|-----|--------------------------------------------------------------------|
| (1)  | 054253 | 035    | 002 | 015 |                                                                    |
| (1)  | 054256 | 000    | 235 |     |                                                                    |
| (1)  | 054260 | 110    | 040 | 040 | .BYTE SDT+10,AD2SRL,SCS,RONPR,AD2DRL,RINPR,CLR,RONPRL,SDT          |
| (1)  | 054263 | 035    | 042 | 015 |                                                                    |
| (1)  | 054266 | 000    | 235 | 100 |                                                                    |
| (1)  | 054273 | 054273 |     |     | .=D.TCS+23                                                         |
| (1)  | 054273 | 262    | 000 | 000 | D.TCQ:: ;TWO ADC, CLOCK TRIGGER, SEQUENTIAL CHANNEL                |
| (1)  | 054276 | 040    | 035 | 002 | .BYTE DMDSIZ+<5*40>,CAINC,AD1SRL,SCS,RONPR,AD1DRL,RINPR,CLR,RONPRL |
| (1)  | 054301 | 015    | 000 | 235 |                                                                    |
| (1)  | 054304 | 111    | 000 | 040 | .BYTE SDT+11,CAINC,AD2SRL,SCS,RONPR,AD2DRL,RINPR,CLR,RONPRL,SDT    |
| (1)  | 054307 | 040    | 035 | 042 |                                                                    |
| (1)  | 054312 | 015    | 000 | 235 |                                                                    |
| (1)  | 054315 | 100    |     |     |                                                                    |
| (1)  | 054316 | 054316 |     |     | .=D.TCQ+23                                                         |
| (1)  |        |        |     |     | : PARALLEL MODE TABLE                                              |
| (1)  | 054316 | 022    | 120 | 000 | D.TESP:: ;TWO ADC, EXTERNAL TRIGGER, SINGLE, PARALLE               |
| (1)  | 054321 | 035    | 040 | 020 | .BYTE DMDSIZ+<0*40>,SEX,AD1SRL,RONPR,AD2SRL,SEN,RONPR,AD1DRL,RINPR |
| (1)  | 054324 | 035    | 002 | 015 |                                                                    |
| (1)  | 054327 | 042    | 015 | 001 | .BYTE AD2DRL,RINPR,AD1SRH,RONPRL,AD2SRH,RONPRL,SDT+6               |
| (1)  | 054332 | 235    | 041 | 235 |                                                                    |
| (1)  | 054335 | 106    |     |     |                                                                    |
| (1)  | 054341 | 054341 |     |     | .=D.TESP+23                                                        |
| (1)  | 054341 | 022    | 120 | 000 | D.TEQP:: ;TWO ADC, EXTERNAL TRIGGER, SEQUENTIAL, PARALLEL          |
| (1)  | 054344 | 035    | 040 | 020 | .BYTE DMDSIZ+<0*40>,SEX,AD1SRL,RONPR,AD2SRL,SEN,RONPR,CAINC,AD1DRL |
| (1)  | 054347 | 035    | 000 | 002 |                                                                    |
| (1)  | 054352 | 015    | 042 | 015 | .BYTE RINPR,AD2DRL,RINPR,AD1SRH,RONPRL,AD2SRH,RONPRL,SDT+6         |
| (1)  | 054355 | 001    | 235 | 041 |                                                                    |
| (1)  | 054360 | 235    | 106 |     |                                                                    |
| (1)  | 054364 | 054364 |     |     | .=D.TEQP+23                                                        |
| (1)  | 054364 | 222    | 040 | 000 | D.TCSP:: ;TWO ADC, CLOCK TRIGGER, SINGLE, PARALLEL                 |
| (1)  | 054367 | 035    | 040 | 040 | .BYTE DMDSIZ+<4*40>,SCS,AD1SRL,RONPR,AD2SRL,SCS,RONPR,AD1DRL,RINPR |
| (1)  | 054372 | 035    | 002 | 015 |                                                                    |
| (1)  | 054375 | 042    | 015 | 001 | .BYTE AD2DRL,RINPR,AD1SRH,RONPRL,AD2SRH,RONPRL,SDT+6               |
| (1)  | 054400 | 235    | 041 | 235 |                                                                    |
| (1)  | 054403 | 106    |     |     |                                                                    |
| (1)  | 054407 | 054407 |     |     | .=D.TCSP+23                                                        |
| (1)  | 054407 | 222    | 040 | 000 | D.TCQ':: ;TWO ADC, CLOCK TRIGGER, SEQUENTIAL, PARALLEL             |
| (1)  | 054412 | 035    | 040 | 040 | .BYTE DMDSIZ+<4*40>,SCS,AD1SRL,RONPR,AD2SRL,SCS,RONPR,CAINC,AD1DRL |
| (1)  | 054415 | 035    | 000 | 002 |                                                                    |
| (1)  | 054420 | 015    | 042 | 015 | .BYTE RINPR,AD2DRL,RINPR,AD1SRH,RONPRL,AD2SRH,RONPRL,SDT+6         |
| (1)  | 054423 | 001    | 235 | 041 |                                                                    |
| (1)  | 054426 | 235    | 106 |     |                                                                    |
| 5504 | 054430 | 000050 |     |     | TODOQ: .BLKW 40.                                                   |
| 5505 | 054550 | 000000 |     |     | TODOC: .WORD 0                                                     |

5506  
5507  
5508  
5509  
5510

:THE FOLLOWING TABLE IS USED BY FLASH TO INDEX THE JOBS BASED  
:ON THE INFO THE LPA RETURNS AS ASSIGNED JOB NUMBERS.  
:FLASH INDEXES THIS TABLE WITH THE JOB NUMBER TO STORE THE RDA  
:ADDR. WHEN CODE=0,ANY OTHER TIME TABLE IS INDEXED TO GET

5511 ;THE RDA ADDR. OF A PARTICULAR JOB.

|      |        |        |  |  |        |       |   |  |  |
|------|--------|--------|--|--|--------|-------|---|--|--|
| 5512 |        |        |  |  |        |       |   |  |  |
| 5513 | 054552 | 000000 |  |  | TABLE: | .WORD | 0 |  |  |
| 5514 | 054554 | 000000 |  |  |        | .WORD | 0 |  |  |
| 5515 | 054556 | 000000 |  |  |        | .WORD | 0 |  |  |
| 5516 | 054560 | 000000 |  |  |        | .WORD | 0 |  |  |
| 5517 | 054562 | 000000 |  |  |        | .WORD | 0 |  |  |
| 5518 | 054564 | 000000 |  |  |        | .WORD | 0 |  |  |
| 5519 | 054566 | 000000 |  |  |        | .WORD | 0 |  |  |
| 5520 | 054570 | 000000 |  |  |        | .WORD | 0 |  |  |
| 5521 | 054572 | 000000 |  |  |        | .WORD | 0 |  |  |

;NOTE THIS AREA FROM TODOQ: TO DONEC: IS  
;CLEARED EACH TIME FLASH IS ENTERED.

|      |        |        |  |  |        |       |     |  |  |
|------|--------|--------|--|--|--------|-------|-----|--|--|
| 5522 |        |        |  |  |        |       |     |  |  |
| 5523 |        |        |  |  |        |       |     |  |  |
| 5524 | 054574 | 000050 |  |  | DONEQ: | .BLKW | 40. |  |  |
| 5525 | 054714 | 000000 |  |  | DONEC: | .WORD | 0   |  |  |

|      |        |        |        |        |         |           |   |  |                         |
|------|--------|--------|--------|--------|---------|-----------|---|--|-------------------------|
| 5526 |        |        |        |        |         |           |   |  |                         |
| 5527 | 054716 | 000000 | 001000 | 000004 | LIST10: | 0,1000,4  |   |  | :AR11 CHO,EXP GR.,TOL,4 |
| 5528 | 054724 | 000003 | 001754 | 000024 |         | 3,1754,24 | : |  | 3,EXP 1754,TOL 24       |
| 5529 | 054732 | 000002 | 001315 | 000024 |         | 2,1315,24 | : |  | 2,EXP 1315,TOL 24       |
| 5530 | 054740 | 000003 | 001754 | 000024 |         | 3,1754,24 | : |  | 3,EXP 1754,TOL 24       |

|      |        |        |        |        |         |            |   |  |                         |
|------|--------|--------|--------|--------|---------|------------|---|--|-------------------------|
| 5531 |        |        |        |        |         |            |   |  |                         |
| 5532 | 054746 | 000000 | 004000 | 000004 | LIST12: | 0,4000,4   |   |  | :AD11K CHO,EXP GR,TOL 4 |
| 5533 | 054754 | 000002 | 004632 | 000050 |         | 2,4632,50  | : |  | 2                       |
| 5534 | 054762 | 000003 | 006000 | 000144 |         | 3,6000,144 |   |  |                         |
| 5535 | 054770 | 000004 | 002000 | 000240 |         | 4,2000,240 |   |  |                         |

5536  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
: \*\*  
: \*\* REQUEST DESCRIPTOR ARRAY FOR JOB #0  
: \*\*  
: \*\* THIS TABLE WILL BE ALERED BY THIS PROGRAM  
: \*\* AND BY THE LPA-11 OPTION.  
: \*\*

|     |        |        |  |  |       |       |         |                          |                             |
|-----|--------|--------|--|--|-------|-------|---------|--------------------------|-----------------------------|
| (1) | 054776 | 000000 |  |  | JOB0: | .WORD | 0       |                          | ;/MODE INFORMATION.         |
| (1) | 055000 | 000000 |  |  |       | .WORD | 0       |                          | ;/WORD COUNT                |
| (1) | 055002 | 055074 |  |  |       | .WORD | 0+JOB0U | ;/USW (USER STATUS WORD) |                             |
| (1) | 055004 | 000    |  |  |       | .BYTE | 0       |                          | ;/USW (EXTENDED ADDR. BITS) |
| (1) | 055005 | 000    |  |  |       | .BYTE | 0       |                          | ;/VBM (VALID BUFFER MASK)   |
| (3) | 055006 | 000000 |  |  |       | .WORD | 0       |                          | ;/BUFFER ADDRESS #0         |
| (3) | 055010 | 000    |  |  |       | .BYTE | 0       |                          | ;/EXTENDED ADDRESS BITS     |
| (3) | 055011 | 000    |  |  |       | .BYTE | 0       |                          | ;/UNUSED                    |
| (3) | 055012 | 000000 |  |  |       | .WORD | 0       |                          | ;/BUFFER ADDRESS #1         |
| (3) | 055014 | 000    |  |  |       | .BYTE | 0       |                          | ;/EXTENDED ADDRESS BITS     |
| (3) | 055015 | 000    |  |  |       | .BYTE | 0       |                          | ;/UNUSED                    |
| (3) | 055016 | 000000 |  |  |       | .WORD | 0       |                          | ;/BUFFER ADDRESS #1         |
| (3) | 055020 | 000    |  |  |       | .BYTE | 0       |                          | ;/EXTENDED ADDRESS BITS     |
| (3) | 055021 | 000    |  |  |       | .BYTE | 0       |                          | ;/UNUSED                    |
| (3) | 055022 | 000000 |  |  |       | .WORD | 0       |                          | ;/BUFFER ADDRESS #1         |
| (3) | 055024 | 000    |  |  |       | .BYTE | 0       |                          | ;/EXTENDED ADDRESS BITS     |
| (3) | 055025 | 000    |  |  |       | .BYTE | 0       |                          | ;/UNUSED                    |
| (3) | 055026 | 000000 |  |  |       | .WORD | 0       |                          | ;/BUFFER ADDRESS #1         |
| (3) | 055030 | 000    |  |  |       | .BYTE | 0       |                          | ;/EXTENDED ADDRESS BITS     |
| (3) | 055031 | 000    |  |  |       | .BYTE | 0       |                          | ;/UNUSED                    |
| (3) | 055032 | 000000 |  |  |       | .WORD | 0       |                          | ;/BUFFER ADDRESS #1         |
| (3) | 055034 | 000    |  |  |       | .BYTE | 0       |                          | ;/EXTENDED ADDRESS BITS     |
| (3) | 055035 | 000    |  |  |       | .BYTE | 0       |                          | ;/UNUSED                    |
| (3) | 055036 | 000000 |  |  |       | .WORD | 0       |                          | ;/BUFFER ADDRESS #1         |

|      |        |        |              |         |                                                         |
|------|--------|--------|--------------|---------|---------------------------------------------------------|
| (3)  | 055040 | 000    | .BYTE        | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055041 | 000    | .BYTE        | 0       | ;/UNUSED                                                |
| (3)  | 055042 | 000000 | .WORD        | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055044 | 000    | .BYTE        | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055045 | 000    | .BYTE        | 0       | ;/UNUSED                                                |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055046 | 055076 | .WORD        | JOBOR   | ;/RCLR (ADDR OF LIST OF RANDOM CHAN)                    |
| (1)  | 055050 | 000    | .BYTE        | 0       | ;/RCL EXTENDED ADR. BITS                                |
| (1)  | 055051 | 000    | .BYTE        | 0       | ;/UNUSED.                                               |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055052 | 000    | JOBOS: .BYTE | 0       | ;/CHANNEL START ADDRESS                                 |
| (1)  | 055053 | 000    | .BYTE        | 0       | ;/NUMBER OF CHANNELS                                    |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055054 | 000    | .BYTE        | 0       | ;/CHANNEL INCREMENT                                     |
| (1)  | 055055 | 000    | .BYTE        | 0       | ;/FATAL ERROR MASK                                      |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055056 | 000000 | .WORD        | 0       | ;/DELAY                                                 |
| (1)  | 055060 | 000000 | .WORD        | 0       | ;/SAMPLE RATE                                           |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055062 | 000    | .BYTE        | 0       | ;/STWD                                                  |
| (1)  | 055063 | 000    | .BYTE        | 0       | ;/EMWD EVENT MARK DIGITAL INT WD#                       |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055064 | 000000 | .WORD        | 0       | ;/ST MSK                                                |
| (1)  | 055066 | 000000 | .WORD        | 0       | ;/EM MSK EVENT MARK DIGITAL IPUT MASK.                  |
| (1)  |        |        |              |         |                                                         |
| (1)  |        |        |              |         | ;/END REG. TABLE                                        |
| (1)  | 055070 | 000003 | .WORD        | 03      | ;/CONTAINS STOP CODE FOR THIS JOB.                      |
| (1)  | 055072 | 000000 | .WORD        | 0       | ;/CONTAINS LOOP COUNT FOR JOB.(NO OF TIMES THUR BUFFER. |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055074 | 000000 | JOB0U: .WORD | 0       | ;/USW USER STATUS WORD                                  |
| (1)  |        |        |              |         |                                                         |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055076 | 000007 | JOBOR: .BLKW | 7       |                                                         |
| (1)  |        |        |              |         |                                                         |
| 5537 |        |        | ;            |         |                                                         |
| (1)  |        |        | ;            |         | REQUEST DESCRIPTOR ARRAY FOR JOB #1                     |
| (1)  |        |        | ;            |         |                                                         |
| (1)  |        |        | ;            |         | THIS TABLE WILL BE ALERED BY THIS PROGRAM               |
| (1)  |        |        | ;            |         | AND BY THE LPA-11 OPTION.                               |
| (1)  |        |        | ;            |         |                                                         |
| (1)  |        |        | ;            |         |                                                         |
| (1)  |        |        | ;            |         |                                                         |
| (1)  | 055114 | 000000 | JOB1: .WORD  | 0       | ;/MODE INFORMATION.                                     |
| (1)  | 055116 | 000000 | .WORD        | 0       | ;/WORD COUNT                                            |
| (1)  | 055120 | 055212 | .WORD        | 0+JOB1U | ;/USW (USER STATUS WORD)                                |
| (1)  | 055122 | 000    | .BYTE        | 0       | ;/USW (EXTENDED ADDR. BITS)                             |
| (1)  | 055123 | 000    | .BYTE        | 0       | ;/VBM (VALID BUFFER MASK)                               |
| (3)  | 055124 | 000000 | .WORD        | 0       | ;/BUFFER ADDRESS #0                                     |
| (3)  | 055126 | 000    | .BYTE        | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055127 | 000    | .BYTE        | 0       | ;/UNUSED                                                |
| (3)  | 055130 | 000000 | .WORD        | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055132 | 000    | .BYTE        | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055133 | 000    | .BYTE        | 0       | ;/UNUSED                                                |
| (3)  | 055134 | 000000 | .WORD        | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055136 | 000    | .BYTE        | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055137 | 000    | .BYTE        | 0       | ;/UNUSED                                                |

|      |        |        |                  |                                           |                                                         |
|------|--------|--------|------------------|-------------------------------------------|---------------------------------------------------------|
| (3)  | 055140 | 000000 | .WORD            | 0                                         | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055142 | 000    | .BYTE            | 0                                         | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055143 | 000    | .BYTE            | 0                                         | ;/UNUSED                                                |
| (3)  | 055144 | 000000 | .WORD            | 0                                         | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055146 | 000    | .BYTE            | 0                                         | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055147 | 000    | .BYTE            | 0                                         | ;/UNUSED                                                |
| (3)  | 055150 | 000000 | .WORD            | 0                                         | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055152 | 000    | .BYTE            | 0                                         | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055153 | 000    | .BYTE            | 0                                         | ;/UNUSED                                                |
| (3)  | 055154 | 000000 | .WORD            | 0                                         | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055156 | 000    | .BYTE            | 0                                         | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055157 | 000    | .BYTE            | 0                                         | ;/UNUSED                                                |
| (3)  | 055160 | 000000 | .WORD            | 0                                         | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055162 | 000    | .BYTE            | 0                                         | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055163 | 000    | .BYTE            | 0                                         | ;/UNUSED                                                |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  | 055164 | 055214 | .WORD            | JOB1R                                     | ;/RCLR (ADDR OF LIST OF RANDOM CHAN)                    |
| (1)  | 055166 | 000    | .BYTE            | 0                                         | ;/RCL EXTENDED ADR. BITS                                |
| (1)  | 055167 | 000    | .BYTE            | 0                                         | ;/UNUSED.                                               |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  | 055170 | 000    | JOB1S: .BYTE     | 0                                         | ;/CHANNEL START ADDRESS                                 |
| (1)  | 055171 | 000    | .BYTE            | 0                                         | ;/NUMBER OF CHANNELS                                    |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  | 055172 | 000    | .BYTE            | 0                                         | ;/CHANNEL INCREMENT                                     |
| (1)  | 055173 | 000    | .BYTE            | 0                                         | ;/FATAL ERROR MASK                                      |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  | 055174 | 000000 | .WORD            | 0                                         | ;/DELAY                                                 |
| (1)  | 055176 | 000000 | .WORD            | 0                                         | ;/SAMPLE RATE                                           |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  | 055200 | 000    | .BYTE            | 0                                         | ;/STWD                                                  |
| (1)  | 055201 | 000    | .BYTE            | 0                                         | ;/EMWD EVENT MARK DIGITAL INT WD#                       |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  | 055202 | 000000 | .WORD            | 0                                         | ;/ST MSK                                                |
| (1)  | 055204 | 000000 | .WORD            | 0                                         | ;/EM MSK EVENT MARK DIGITAL IPUT MASK.                  |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  |        |        | ;/END REG. TABLE |                                           |                                                         |
| (1)  | 055206 | 000003 | .WORD            | 03                                        | ;/CONTAINS STOP CODE FOR THIS JOB.                      |
| (1)  | 055210 | 000000 | .WORD            | 0                                         | ;/CONTAINS LOOP COUNT FOR JOB.(NO OF TIMES THUR BUFFER. |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  | 055212 | 000000 | JOB1U: .WORD     | 0                                         | ;/USW USER STATUS WORD                                  |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  | 055214 | 000007 | JOB1R: .BLKW     | 7                                         |                                                         |
| (1)  |        |        |                  |                                           |                                                         |
| 5538 |        |        | ***              |                                           |                                                         |
| (1)  |        |        | ***              | REQUEST DESCRIPTOR ARRAY FOR JOB #2       |                                                         |
| (1)  |        |        | ***              |                                           |                                                         |
| (1)  |        |        | ***              | THIS TABLE WILL BE ALERED BY THIS PROGRAM |                                                         |
| (1)  |        |        | ***              | AND BY THE LPA-11 OPTION.                 |                                                         |
| (1)  |        |        | ***              |                                           |                                                         |
| (1)  |        |        |                  |                                           |                                                         |
| (1)  | 055232 | 000000 | JOB2: .WORD      | 0                                         | ;/MODE INFORMATION.                                     |
| (1)  | 055234 | 000000 | .WORD            | 0                                         | ;/WORD COUNT                                            |
| (1)  | 055236 | 055330 | .WORD            | 0+JOB2U                                   | ;/USW (USER STATUS WORD)                                |
| (1)  | 055240 | 000    | .BYTE            | 0                                         | ;/USW (EXTENDED ADDR. BITS)                             |



|     |        |        |                  |       |                                                         |
|-----|--------|--------|------------------|-------|---------------------------------------------------------|
| (1) | 055241 | 000    | .BYTE            | 0     | ;/VBM (VALID BUFFER MASK)                               |
| (3) | 055242 | 000000 | .WORD            | 0     | ;/BUFFER ADDRESS #0                                     |
| (3) | 055244 | 000    | .BYTE            | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3) | 055245 | 000    | .BYTE            | 0     | ;/UNUSED                                                |
| (3) | 055246 | 000000 | .WORD            | 0     | ;/BUFFER ADDRESS #1                                     |
| (3) | 055250 | 000    | .BYTE            | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3) | 055251 | 000    | .BYTE            | 0     | ;/UNUSED                                                |
| (3) | 055252 | 000000 | .WORD            | 0     | ;/BUFFER ADDRESS #1                                     |
| (3) | 055254 | 000    | .BYTE            | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3) | 055255 | 000    | .BYTE            | 0     | ;/UNUSED                                                |
| (3) | 055256 | 000000 | .WORD            | 0     | ;/BUFFER ADDRESS #1                                     |
| (3) | 055260 | 000    | .BYTE            | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3) | 055261 | 000    | .BYTE            | 0     | ;/UNUSED                                                |
| (3) | 055262 | 000000 | .WORD            | 0     | ;/BUFFER ADDRESS #1                                     |
| (3) | 055264 | 000    | .BYTE            | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3) | 055265 | 000    | .BYTE            | 0     | ;/UNUSED                                                |
| (3) | 055266 | 000000 | .WORD            | 0     | ;/BUFFER ADDRESS #1                                     |
| (3) | 055270 | 000    | .BYTE            | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3) | 055271 | 000    | .BYTE            | 0     | ;/UNUSED                                                |
| (3) | 055272 | 000000 | .WORD            | 0     | ;/BUFFER ADDRESS #1                                     |
| (3) | 055274 | 000    | .BYTE            | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3) | 055275 | 000    | .BYTE            | 0     | ;/UNUSED                                                |
| (3) | 055276 | 000000 | .WORD            | 0     | ;/BUFFER ADDRESS #1                                     |
| (3) | 055300 | 000    | .BYTE            | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3) | 055301 | 000    | .BYTE            | 0     | ;/UNUSED                                                |
| (1) | 055302 | 055332 | .WORD            | JOB2R | ;/RCLR (ADDR OF LIST OF RANDOM CHAN)                    |
| (1) | 055304 | 000    | .BYTE            | 0     | ;/RCL EXTENDED ADR. BITS                                |
| (1) | 055305 | 000    | .BYTE            | 0     | ;/UNUSED.                                               |
| (1) | 055306 | 000    | JOB2S: .BYTE     | 0     | ;/CHANNEL START ADDRESS                                 |
| (1) | 055307 | 000    | .BYTE            | 0     | ;/NUMBER OF CHANNELS                                    |
| (1) | 055310 | 000    | .BYTE            | 0     | ;/CHANNEL INCREMENT                                     |
| (1) | 055311 | 000    | .BYTE            | 0     | ;/FATAL ERROR MASK                                      |
| (1) | 055312 | 000000 | .WORD            | 0     | ;/DELAY                                                 |
| (1) | 055314 | 000000 | .WORD            | 0     | ;/SAMPLE RATE                                           |
| (1) | 055316 | 000    | .BYTE            | 0     | ;/STWD                                                  |
| (1) | 055317 | 000    | .BYTE            | 0     | ;/EMWD EVENT MARK DIGITAL INT WD#                       |
| (1) | 055320 | 000000 | .WORD            | 0     | ;/ST MSK                                                |
| (1) | 055322 | 000000 | .WORD            | 0     | ;/EM MSK EVENT MARK DIGITAL IPUT MASK.                  |
| (1) |        |        | ;/END REG. TABLE |       |                                                         |
| (1) | 055324 | 000003 | .WORD            | 03    | ;/CONTAINS STOP CODE FOR THIS JOB.                      |
| (1) | 055326 | 000000 | .WORD            | 0     | ;/CONTAINS LOOP COUNT FOR JOB.(NO OF TIMES THUR BUFFER. |
| (1) | 055330 | 000000 | JOB2U: .WORD     | 0     | ;/USW USER STATUS WORD                                  |
| (1) | 055332 | 000007 | JOB2R: .BLKW     | 7     |                                                         |
| (1) | 5539   |        | ;/**             |       |                                                         |

```

(1) : ** REQUEST DESCRIPTOR ARRAY FOR JOB #3
(1) : **
(1) : ** THIS TABLE WILL BE ALERED BY THIS PROGRAM
(1) : ** AND BY THE LPA-11 OPTION.
(1) : **
(1) : **
(1) 055350 000000 JOB3: .WORD 0 ;/MODE INFORMATION.
(1) 055352 000000 .WORD 0 ;/WORD COUNT
(1) 055354 055446 .WORD 0+JOB3U ;/USW (USER STATUS WORD)
(1) 055356 000 .BYTE 0 ;/USW (EXTENDED ADDR. BITS)
(1) 055357 000 .BYTE 0 ;/VBM (VALID BUFFER MASK)
(3) 055360 000000 .WORD 0 ;/BUFFER ADDRESS #0
(3) 055362 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055363 000 .BYTE 0 ;/UNUSED
(3) 055364 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055366 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055367 000 .BYTE 0 ;/UNUSED
(3) 055370 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055372 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055373 000 .BYTE 0 ;/UNUSED
(3) 055374 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055376 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055377 000 .BYTE 0 ;/UNUSED
(3) 055400 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055402 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055403 000 .BYTE 0 ;/UNUSED
(3) 055404 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055406 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055407 000 .BYTE 0 ;/UNUSED
(3) 055410 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055412 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055413 000 .BYTE 0 ;/UNUSED
(3) 055414 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055416 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055417 000 .BYTE 0 ;/UNUSED
(1)
(1) 055420 055450 .WORD JOB3R ;/RCLR (ADDR OF LIST OF RANDOM CHAN)
(1) 055422 000 .BYTE 0 ;/RCL EXTENDED ADR. BITS
(1) 055423 000 .BYTE 0 ;/UNUSED.
(1)
(1) 055424 000 JOB3S: .BYTE 0 ;/CHANNEL START ADDRESS
(1) 055425 000 .BYTE 0 ;/NUMBER OF CHANNELS
(1)
(1)
(1) 055426 000 .BYTE 0 ;/CHANNEL INCREMENT
(1) 055427 000 .BYTE 0 ;/FATAL ERROR MASK
(1)
(1) 055430 000000 .WORD 0 ;/DELAY
(1) 055432 000000 .WORD 0 ;/SAMPLE RATE
(1)
(1) 055434 000 .BYTE 0 ;/STWD
(1) 055435 000 .BYTE 0 ;/EMWD EVENT MARK DIGITAL INT WD#
(1)
(1) 055436 000000 .WORD 0 ;/ST MSK
(1) 055440 000000 .WORD 0 ;/EM MSK EVENT MARK DIGITAL IPUT MASK.
(1)

```

|      |        |        |                 |         |                                                         |
|------|--------|--------|-----------------|---------|---------------------------------------------------------|
| (1)  |        |        | :END REG. TABLE |         |                                                         |
| (1)  | 055442 | 000003 | .WORD           | 03      | :/CONTAINS STOP CODE FOR THIS JOB.                      |
| (1)  | 055444 | 000000 | .WORD           | 0       | :/CONTAINS LOOP COUNT FOR JOB.(NO OF TIMES THUR BUFFER. |
| (1)  |        |        |                 |         |                                                         |
| (1)  | 055446 | 000000 | JOB3U: .WORD    | 0       | :/USW USER STATUS WORD                                  |
| (1)  |        |        |                 |         |                                                         |
| (1)  |        |        |                 |         |                                                         |
| (1)  | 055450 | 000007 | JOB3R: .BLKW    | 7       |                                                         |
| (1)  |        |        |                 |         |                                                         |
| 5540 |        |        | :**             |         |                                                         |
| (1)  |        |        | :**             |         | REQUEST DESCRIPTOR ARRAY FOR JOB #4                     |
| (1)  |        |        | :**             |         |                                                         |
| (1)  |        |        | :**             |         | THIS TABLE WILL BE ALERED BY THIS PROGRAM               |
| (1)  |        |        | :**             |         | AND BY THE LPA-11 OPTION.                               |
| (1)  |        |        | :**             |         |                                                         |
| (1)  |        |        |                 |         |                                                         |
| (1)  | 055466 | 000000 | JOB4: .WORD     | 0       | :/MODE INFORMATION.                                     |
| (1)  | 055470 | 000000 | .WORD           | 0       | :/WORD COUNT                                            |
| (1)  | 055472 | 055564 | .WORD           | 0+JOB4U | :/USW (USER STATUS WORD)                                |
| (1)  | 055474 | 000    | .BYTE           | 0       | :/USW (EXTENDED ADDR. BITS)                             |
| (1)  | 055475 | 000    | .BYTE           | 0       | :/VBM (VALID BUFFER MASK)                               |
| (3)  | 055476 | 000000 | .WORD           | 0       | :/BUFFER ADDRESS #0                                     |
| (3)  | 055500 | 000    | .BYTE           | 0       | :/EXTENDED ADDRESS BITS                                 |
| (3)  | 055501 | 000    | .BYTE           | 0       | :/UNUSED                                                |
| (3)  | 055502 | 000000 | .WORD           | 0       | :/BUFFER ADDRESS #1                                     |
| (3)  | 055504 | 000    | .BYTE           | 0       | :/EXTENDED ADDRESS BITS                                 |
| (3)  | 055505 | 000    | .BYTE           | 0       | :/UNUSED                                                |
| (3)  | 055506 | 000000 | .WORD           | 0       | :/BUFFER ADDRESS #1                                     |
| (3)  | 055510 | 000    | .BYTE           | 0       | :/EXTENDED ADDRESS BITS                                 |
| (3)  | 055511 | 000    | .BYTE           | 0       | :/UNUSED                                                |
| (3)  | 055512 | 000000 | .WORD           | 0       | :/BUFFER ADDRESS #1                                     |
| (3)  | 055514 | 000    | .BYTE           | 0       | :/EXTENDED ADDRESS BITS                                 |
| (3)  | 055515 | 000    | .BYTE           | 0       | :/UNUSED                                                |
| (3)  | 055516 | 000000 | .WORD           | 0       | :/BUFFER ADDRESS #1                                     |
| (3)  | 055520 | 000    | .BYTE           | 0       | :/EXTENDED ADDRESS BITS                                 |
| (3)  | 055521 | 000    | .BYTE           | 0       | :/UNUSED                                                |
| (3)  | 055522 | 000000 | .WORD           | 0       | :/BUFFER ADDRESS #1                                     |
| (3)  | 055524 | 000    | .BYTE           | 0       | :/EXTENDED ADDRESS BITS                                 |
| (3)  | 055525 | 000    | .BYTE           | 0       | :/UNUSED                                                |
| (3)  | 055526 | 000000 | .WORD           | 0       | :/BUFFER ADDRESS #1                                     |
| (3)  | 055530 | 000    | .BYTE           | 0       | :/EXTENDED ADDRESS BITS                                 |
| (3)  | 055531 | 000    | .BYTE           | 0       | :/UNUSED                                                |
| (3)  | 055532 | 000000 | .WORD           | 0       | :/BUFFER ADDRESS #1                                     |
| (3)  | 055534 | 000    | .BYTE           | 0       | :/EXTENDED ADDRESS BITS                                 |
| (3)  | 055535 | 000    | .BYTE           | 0       | :/UNUSED                                                |
| (1)  |        |        |                 |         |                                                         |
| (1)  | 055536 | 055566 | .WORD           | JOB4R   | :/RCLR (ADDR OF LIST OF RANDOM CHAN)                    |
| (1)  | 055540 | 000    | .BYTE           | 0       | :/RCL EXTENDED ADR. BITS                                |
| (1)  | 055541 | 000    | .BYTE           | 0       | :/UNUSED.                                               |
| (1)  |        |        |                 |         |                                                         |
| (1)  | 055542 | 000    | JOB4S: .BYTE    | 0       | :/CHANNEL START ADDRESS                                 |
| (1)  | 055543 | 000    | .BYTE           | 0       | :/NUMBER OF CHANNELS                                    |
| (1)  |        |        |                 |         |                                                         |
| (1)  |        |        |                 |         |                                                         |
| (1)  | 055544 | 000    | .BYTE           | 0       | :/CHANNEL INCREMFNT                                     |
| (1)  | 055545 | 000    | .BYTE           | 0       | :/FATAL ERROR MASK                                      |

```
(1)
(1) 055546 000000 .WORD 0 ;/DELAY
(1) 055550 000000 .WORD 0 ;/SAMPLE RATE
(1)
(1) 055552 000 .BYTE 0 ;/STWD
(1) 055553 000 .BYTE 0 ;/EMWD EVENT MARK DIGITAL INT WD#
(1)
(1) 055554 000000 .WORD 0 ;/ST MSK
(1) 055556 000000 .WORD 0 ;/EM MSK EVENT MARK DIGITAL IPUT MASK.
(1)
(1) ;END REG. TABLE
(1) 055560 000003 .WORD 03 ;/CONTAINS STOP CODE FOR THIS JOB.
(1) 055562 000000 .WORD 0 ;/CONTAINS LOOP COUNT FOR JOB.(NO OF TIMES THUR BUFFER.
(1)
(1) 055564 000000 JOB4U: .WORD 0 ;/USW USER STATUS WORD
(1)
(1) 055566 000007 JOB4R: .BLKW 7
(1)
5541
(1) ;**
(1) ;** REQUEST DESCRIPTOR ARRAY FOR JOB #5
(1) ;**
(1) ;** THIS TABLE WILL BE ALERED BY THIS PROGRAM
(1) ;** AND BY THE LPA-11 OPTION.
(1) ;**
(1)
(1) 055604 000000 JOB5: .WORD 0 ;/MODE INFORMATION.
(1) 055606 000000 .WORD 0 ;/WORD COUNT
(1) 055610 055702 .WORD 0+JOB5U ;/USW (USER STATUS WORD)
(1) 055612 000 .BYTE 0 ;/USW (EXTENDED ADDR. BITS)
(1) 055613 000 .BYTE 0 ;/VBM (VALID BUFFER MASK)
(3) 055614 000000 .WORD 0 ;/BUFFER ADDRESS #0
(3) 055616 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055617 000 .BYTE 0 ;/UNUSED
(3) 055620 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055622 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055623 000 .BYTE 0 ;/UNUSED
(3) 055624 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055626 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055627 000 .BYTE 0 ;/UNUSED
(3) 055630 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055632 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055633 000 .BYTE 0 ;/UNUSED
(3) 055634 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055636 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055637 000 .BYTE 0 ;/UNUSED
(3) 055640 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055642 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055643 000 .BYTE 0 ;/UNUSED
(3) 055644 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055646 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055647 000 .BYTE 0 ;/UNUSED
(3) 055650 000000 .WORD 0 ;/BUFFER ADDRESS #1
(3) 055652 000 .BYTE 0 ;/EXTENDED ADDRESS BITS
(3) 055653 000 .BYTE 0 ;/UNUSED
(1)
```

|      |        |        |              |         |                                                         |
|------|--------|--------|--------------|---------|---------------------------------------------------------|
| (1)  | 055654 | 055704 | .WORD        | JOB5R   | ;/RCLR (ADDR OF LIST OF RANDOM CHAN)                    |
| (1)  | 055656 | 000    | .BYTE        | 0       | ;/RCL EXTENDED ADR. BITS                                |
| (1)  | 055657 | 000    | .BYTE        | 0       | ;/UNUSED.                                               |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055660 | 000    | JOB5S: .BYTE | 0       | ;/CHANNEL START ADDRESS                                 |
| (1)  | 055661 | 000    | .BYTE        | 0       | ;/NUMBER OF CHANNELS                                    |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055662 | 000    | .BYTE        | 0       | ;/CHANNEL INCREMENT                                     |
| (1)  | 055663 | 000    | .BYTE        | 0       | ;/FATAL ERROR MASK                                      |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055664 | 000000 | .WORD        | 0       | ;/DELAY                                                 |
| (1)  | 055666 | 000000 | .WORD        | 0       | ;/SAMPLE RATE                                           |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055670 | 000    | .BYTE        | 0       | ;/STWD                                                  |
| (1)  | 055671 | 000    | .BYTE        | 0       | ;/EMWD EVENT MARK DIGITAL INT WDM                       |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055672 | 000000 | .WORD        | 0       | ;/ST MSK                                                |
| (1)  | 055674 | 000000 | .WORD        | 0       | ;/EM MSK EVENT MARK DIGITAL IPUT MASK.                  |
| (1)  |        |        |              |         |                                                         |
| (1)  |        |        |              |         | ;/END REG. TABLE                                        |
| (1)  | 055676 | 000003 | .WORD        | 03      | ;/CONTAINS STOP CODE FOR THIS JOB.                      |
| (1)  | 055700 | 000000 | .WORD        | 0       | ;/CONTAINS LOOP COUNT FOR JOB.(NO OF TIMES THUR BUFFER. |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055702 | 000000 | JOB5U: .WORD | 0       | ;/USW USER STATUS WORD                                  |
| (1)  |        |        |              |         |                                                         |
| (1)  | 055704 | 000007 | JOB5R: .BLKW | 7       |                                                         |
| (1)  |        |        |              |         |                                                         |
| 5542 |        |        | ;            |         |                                                         |
| (1)  |        |        | ;            |         | REQUEST DESCRIPTOR ARRAY FOR JOB #6                     |
| (1)  |        |        | ;            |         |                                                         |
| (1)  |        |        | ;            |         | THIS TABLE WILL BE ALERED BY THIS PROGRAM               |
| (1)  |        |        | ;            |         | AND BY THE LPA-11 OPTION.                               |
| (1)  |        |        | ;            |         |                                                         |
| (1)  |        |        | ;            |         |                                                         |
| (1)  | 055722 | 000000 | JOB6: .WORD  | 0       | ;/MODE INFORMATION.                                     |
| (1)  | 055724 | 000000 | .WORD        | 0       | ;/WORD COUNT                                            |
| (1)  | 055726 | 056020 | .WORD        | 0+JOB6U | ;/USW (USER STATUS WORD)                                |
| (1)  | 055730 | 000    | .BYTE        | 0       | ;/USW (EXTENDED ADDR. BITS)                             |
| (1)  | 055731 | 000    | .BYTE        | 0       | ;/VBM (VALID BUFFER MASK)                               |
| (3)  | 055732 | 000000 | .WORD        | 0       | ;/BUFFER ADDRESS #0                                     |
| (3)  | 055734 | 000    | .BYTE        | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055735 | 000    | .BYTE        | 0       | ;/UNUSED                                                |
| (3)  | 055736 | 000000 | .WORD        | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055740 | 000    | .BYTE        | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055741 | 000    | .BYTE        | 0       | ;/UNUSED                                                |
| (3)  | 055742 | 000000 | .WORD        | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055744 | 000    | .BYTE        | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055745 | 000    | .BYTE        | 0       | ;/UNUSED                                                |
| (3)  | 055746 | 000000 | .WORD        | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055750 | 000    | .BYTE        | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055751 | 000    | .BYTE        | 0       | ;/UNUSED                                                |
| (3)  | 055752 | 000000 | .WORD        | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055754 | 000    | .BYTE        | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055755 | 000    | .BYTE        | 0       | ;/UNUSED                                                |

|      |        |        |                                                |         |                                                         |
|------|--------|--------|------------------------------------------------|---------|---------------------------------------------------------|
| (3)  | 055756 | 000000 | .WORD                                          | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055760 | 000    | .BYTE                                          | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055761 | 000    | .BYTE                                          | 0       | ;/UNUSED                                                |
| (3)  | 055762 | 000000 | .WORD                                          | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055764 | 000    | .BYTE                                          | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055765 | 000    | .BYTE                                          | 0       | ;/UNUSED                                                |
| (3)  | 055766 | 000000 | .WORD                                          | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 055770 | 000    | .BYTE                                          | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 055771 | 000    | .BYTE                                          | 0       | ;/UNUSED                                                |
| (1)  | 055772 | 056022 | .WORD                                          | JOB6R   | ;/RCLR (ADDR OF LIST OF RANDOM CHAN)                    |
| (1)  | 055774 | 000    | .BYTE                                          | 0       | ;/RCL EXTENDED ADR. BITS                                |
| (1)  | 055775 | 000    | .BYTE                                          | 0       | ;/UNUSED.                                               |
| (1)  | 055776 | 000    | JOB6S: .BYTE                                   | 0       | ;/CHANNEL START ADDRESS                                 |
| (1)  | 055777 | 000    | .BYTE                                          | 0       | ;/NUMBER OF CHANNELS                                    |
| (1)  | 056000 | 000    | .BYTE                                          | 0       | ;/CHANNEL INCREMENT                                     |
| (1)  | 056001 | 000    | .BYTE                                          | 0       | ;/FATAL ERROR MASK                                      |
| (1)  | 056002 | 000000 | .WORD                                          | 0       | ;/DELAY                                                 |
| (1)  | 056004 | 000000 | .WORD                                          | 0       | ;/SAMPLE RATE                                           |
| (1)  | 056006 | 000    | .BYTE                                          | 0       | ;/STWD                                                  |
| (1)  | 056007 | 000    | .BYTE                                          | 0       | ;/EMWD EVENT MARK DIGITAL INT WD#                       |
| (1)  | 056010 | 000000 | .WORD                                          | 0       | ;/ST MSK                                                |
| (1)  | 056012 | 000000 | .WORD                                          | 0       | ;/EM MSK EVENT MARK DIGITAL IPUT MASK.                  |
| (1)  |        |        | ;END REG. TABLE                                |         |                                                         |
| (1)  | 056014 | 000003 | .WORD                                          | 03      | ;/CONTAINS STOP CODE FOR THIS JOB.                      |
| (1)  | 056016 | 000000 | .WORD                                          | 0       | ;/CONTAINS LOOP COUNT FOR JOB.(NO OF TIMES THUR BUFFER. |
| (1)  | 056020 | 000000 | JOB6U: .WORD                                   | 0       | ;/USW USER STATUS WORD                                  |
| (1)  | 056022 | 000007 | JOB6R: .BLKW                                   | 7       |                                                         |
| 5543 |        |        | ; ** REQUEST DESCRIPTOR ARRAY FOR JOB #7       |         |                                                         |
| (1)  |        |        | ; ** THIS TABLE WILL BE ALERED BY THIS PROGRAM |         |                                                         |
| (1)  |        |        | ; ** AND BY THE LPA-11 OPTION.                 |         |                                                         |
| (1)  |        |        | ; **                                           |         |                                                         |
| (1)  | 056040 | 000000 | JOB7: .WORD                                    | 0       | ;/MODE INFORMATION.                                     |
| (1)  | 056042 | 000000 | .WORD                                          | 0       | ;/WORD COUNT                                            |
| (1)  | 056044 | 056136 | .WORD                                          | 0+JOB7U | ;/USW (USER STATUS WORD)                                |
| (1)  | 056046 | 000    | .BYTE                                          | 0       | ;/USW (EXTENDED ADDR. BITS)                             |
| (1)  | 056047 | 000    | .BYTE                                          | 0       | ;/VBM (VALID BUFFER MASK)                               |
| (3)  | 056050 | 000000 | .WORD                                          | 0       | ;/BUFFER ADDRESS #0                                     |
| (3)  | 056052 | 000    | .BYTE                                          | 0       | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 056053 | 000    | .BYTE                                          | 0       | ;/UNUSED                                                |
| (3)  | 056054 | 000000 | .WORD                                          | 0       | ;/BUFFER ADDRESS #1                                     |
| (3)  | 056055 | 000    | .BYTE                                          | 0       | ;/EXTENDED ADDRESS BITS                                 |

|      |        |        |       |       |                                                         |
|------|--------|--------|-------|-------|---------------------------------------------------------|
| (3)  | 056057 | 000    | .BYTE | 0     | ;/UNUSED                                                |
| (3)  | 056060 | 000000 | .WORD | 0     | ;/BUFFER ADDRESS #1                                     |
| (3)  | 056062 | 000    | .BYTE | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 056063 | 000    | .BYTE | 0     | ;/UNUSED                                                |
| (3)  | 056064 | 000000 | .WORD | 0     | ;/BUFFER ADDRESS #1                                     |
| (3)  | 056066 | 000    | .BYTE | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 056067 | 000    | .BYTE | 0     | ;/UNUSED                                                |
| (3)  | 056070 | 000000 | .WORD | 0     | ;/BUFFER ADDRESS #1                                     |
| (3)  | 056072 | 000    | .BYTE | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 056073 | 000    | .BYTE | 0     | ;/UNUSED                                                |
| (3)  | 056074 | 000000 | .WORD | 0     | ;/BUFFER ADDRESS #1                                     |
| (3)  | 056076 | 000    | .BYTE | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 056077 | 000    | .BYTE | 0     | ;/UNUSED                                                |
| (3)  | 056100 | 000000 | .WORD | 0     | ;/BUFFER ADDRESS #1                                     |
| (3)  | 056102 | 000    | .BYTE | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 056103 | 000    | .BYTE | 0     | ;/UNUSED                                                |
| (3)  | 056104 | 000000 | .WORD | 0     | ;/BUFFER ADDRESS #1                                     |
| (3)  | 056106 | 000    | .BYTE | 0     | ;/EXTENDED ADDRESS BITS                                 |
| (3)  | 056107 | 000    | .BYTE | 0     | ;/UNUSED                                                |
| (1)  |        |        |       |       |                                                         |
| (1)  | 056110 | 056140 | .WORD | JOB7R | ;/RCLR (ADDR OF LIST OF RANDOM CHAN)                    |
| (1)  | 056112 | 000    | .BYTE | 0     | ;/RCL EXTENDED ADR. BITS                                |
| (1)  | 056113 | 000    | .BYTE | 0     | ;/UNUSED.                                               |
| (1)  |        |        |       |       |                                                         |
| (1)  | 056114 | 000    | .BYTE | 0     | ;/CHANNEL START ADDRESS                                 |
| (1)  | 056115 | 000    | .BYTE | 0     | ;/NUMBER OF CHANNFLS                                    |
| (1)  |        |        |       |       |                                                         |
| (1)  |        |        |       |       |                                                         |
| (1)  | 056116 | 000    | .BYTE | 0     | ;/CHANNEL INCREMENT                                     |
| (1)  | 056117 | 000    | .BYTE | 0     | ;/FATAL ERROR MASK                                      |
| (1)  |        |        |       |       |                                                         |
| (1)  | 056120 | 000000 | .WORD | 0     | ;/DELAY                                                 |
| (1)  | 056122 | 000000 | .WORD | 0     | ;/SAMPLE RATE                                           |
| (1)  |        |        |       |       |                                                         |
| (1)  | 056124 | 000    | .BYTE | 0     | ;/STWD                                                  |
| (1)  | 056125 | 000    | .BYTE | 0     | ;/EMWD EVENT MARK DIGITAL INT WDN                       |
| (1)  |        |        |       |       |                                                         |
| (1)  | 056126 | 000000 | .WORD | 0     | ;/ST MSK                                                |
| (1)  | 056130 | 000000 | .WORD | 0     | ;/EM MSK EVENT MARK DIGITAL IPUT MASK.                  |
| (1)  |        |        |       |       |                                                         |
| (1)  |        |        |       |       |                                                         |
| (1)  |        |        |       |       |                                                         |
| (1)  |        |        |       |       |                                                         |
| (1)  | 056132 | 000003 | .WORD | 03    | ;/CONTAINS STOP CODE FOR THIS JOB.                      |
| (1)  | 056134 | 000000 | .WORD | 0     | ;/CONTAINS LOOP COUNT FOR JOB.(NO OF TIMES THUR BUFFER. |
| (1)  |        |        |       |       |                                                         |
| (1)  | 056136 | 000000 | .WORD | 0     | ;/USW USER STATUS WORD                                  |
| (1)  |        |        |       |       |                                                         |
| (1)  |        |        |       |       |                                                         |
| (1)  | 056140 | 000007 | .BLKW | 7     |                                                         |
| (1)  |        |        |       |       |                                                         |
| 5544 | 056156 |        |       |       | FRECOR::                                                |
| 5545 | 056156 |        |       |       | BUFFER:                                                 |
| 5546 | 056156 | 000401 | .BLKW | 257.  | BUFF0:                                                  |
| 5547 | 057160 | 000401 | .BLKW | 257.  | BUFF1:                                                  |
| 5548 | 060162 | 000401 | .BLKW | 257.  | BUFF2:                                                  |
| 5549 | 061164 | 000401 | .BLKW | 257.  | BUFF3:                                                  |
| 5550 | 062166 | 000200 | .BLKW | 128.  | BUFF4:                                                  |

DMDT MACY11 30G(1063) 24-OCT-80 09:27 PAGE 36-13  
CRLPAB.P11 20-OCT-80 13:17

DMDT -- DEDICATED MODE DISPATCH TABLE

SEQ 0198

5551 062566 000200  
5552  
5558  
5560  
5561  
5562  
5563  
5564  
5565  
5566  
5567  
5568  
5569  
5570  
5571  
5572 000001

BUFF5: .BLKW 128.

WARNING ..!!!!!!..

THE FREE CORE AREA BETWEEN HERE AND PROGRAM  
"DRLPX0" MAY BE USED BY CERTAIN PARTS OF THIS  
PROGRAM. YOU MAY NOT PLACE ANY CODE HERE.  
LAST AVAILABLE SPACE TO PUT CODE IS BEFORE  
LOCATION "FRECOR" OR AFTER ADDRESS "65500".

IF YOU HAVE TO VIOLATE THIS, DON'T ALLOW ROUTINE  
"MAKEI" TO BE RUN!

.END





|         |        |       |       |       |       |      |       |      |      |      |      |      |      |      |  |
|---------|--------|-------|-------|-------|-------|------|-------|------|------|------|------|------|------|------|--|
| AMTYP2= | 000000 | 1187  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| AMTYP3= | 000000 | 1187  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| AMTYP4= | 000000 | 1187  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| APASS = | 000000 | 1187  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| APRIOR= | 000000 | 1187  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| APTCSU= | 000040 | 4854  | 4856# |       |       |      |       |      |      |      |      |      |      |      |  |
| APTENV= | 000001 | 4847  | 4854  | 4856# |       |      |       |      |      |      |      |      |      |      |  |
| APTSIZ= | 000200 | 1520  | 4856# |       |       |      |       |      |      |      |      |      |      |      |  |
| APTSP0= | 000100 | 4854  | 4856# |       |       |      |       |      |      |      |      |      |      |      |  |
| ARADB   | 001712 | 1467# | 1592* | 1599* | 3096  |      |       |      |      |      |      |      |      |      |  |
| ARADS   | 001710 | 1466# | 1591* | 3042  | 3056  | 3066 | 3067  | 3077 | 3078 | 3087 | 3088 | 3098 | 3101 |      |  |
| ARCB    | 001716 | 1469# | 1594* | 1601* | 3186  |      |       |      |      |      |      |      |      |      |  |
| ARCC    | 001726 | 1473# | 1598* | 1605* |       |      |       |      |      |      |      |      |      |      |  |
| ARCS    | 001714 | 1468# | 1593* | 1600* | 3148  | 3156 | 3165  | 3166 | 3175 | 3176 | 3187 | 3188 | 3198 | 3199 |  |
| ARDS    | 001720 | 1470# | 1595* | 1602* | 3221  | 3231 | 3239  | 3240 | 3250 | 3251 | 3260 | 3261 | 3337 | 3342 |  |
|         |        | 3347  | 3351  |       |       |      |       |      |      |      |      |      |      |      |  |
| ARXB    | 001722 | 1471# | 1596* | 1603* | 3269  | 3270 | 3286  | 3287 | 3302 | 3303 | 3336 | 3346 |      |      |  |
| ARYB    | 001724 | 1472# | 1597* | 1604* | 3277  | 3278 | 3293  | 3294 | 3310 | 3311 | 3341 | 3350 |      |      |  |
| AR11K   | 001610 | 1409# | 1590  | 4031  | 4053  | 4073 |       |      |      |      |      |      |      |      |  |
| ASWREG= | 000000 | 1187  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| ATESTN= | 000000 | 1187  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| AUNIT = | 000000 | 1187  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| AUSWR = | 000000 | 1187  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| AVECT1= | 000300 | 1181# | 1187  | 1316  | 1317  | 1321 |       |      |      |      |      |      |      |      |  |
| AVECT2= | 000000 | 1187  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| AVERR   | 034246 | 4523  | 4606  | 4908# | 5155  |      |       |      |      |      |      |      |      |      |  |
| AVERRL  | 034304 | 4914# | 4945  |       |       |      |       |      |      |      |      |      |      |      |  |
| AVERRN  | 034470 | 4942# | 4954  | 4963  |       |      |       |      |      |      |      |      |      |      |  |
| AVEXP   | 034614 | 4521* | 4604* | 4804* | 4813* | 4932 | 4985# | 5378 |      |      |      |      |      |      |  |
| AVTKN   | 034616 | 4916* | 4927* | 4959* | 4986# |      |       |      |      |      |      |      |      |      |  |
| BIT0 =  | 000001 | 1179# | 2578  | 2581  | 2697  | 2719 | 2728  | 3097 | 3184 | 3421 | 3434 | 3535 | 4047 | 4121 |  |
|         |        | 4313  | 4321  | 4326  | 4329  | 4473 | 4478  | 4509 | 4547 | 4552 | 4592 | 4646 | 4656 | 4889 |  |
|         |        | 5186  | 5243  |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT00 = | 000001 | 1179# |       |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT01 = | 000002 | 1179# |       |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT02 = | 000004 | 1179# |       |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT03 = | 000010 | 1179# |       |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT04 = | 000020 | 1179# |       |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT05 = | 000040 | 1179# |       |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT06 = | 000100 | 1179# |       |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT07 = | 000200 | 1179# |       |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT08 = | 000400 | 1179# | 4849  |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT09 = | 001000 | 1179# | 4847  | 4849  |       |      |       |      |      |      |      |      |      |      |  |
| BIT1 =  | 000002 | 1179# | 2178  | 2384  | 2624  | 2697 | 2703  | 2719 | 2728 | 3184 | 3190 | 3535 | 3541 | 3938 |  |
|         |        | 4025  | 4311  | 4326  | 4449  | 4451 | 4453  | 4455 | 4457 | 4713 | 4782 | 5307 |      |      |  |
| BIT10 = | 002000 | 1179# | 2578  | 2581  | 3034  | 3075 | 3140  | 3216 | 3328 | 3339 | 3344 | 3399 | 3884 | 4029 |  |
|         |        | 4051  | 4071  | 4311  | 4473  | 4509 | 4515  | 4547 | 4592 | 4598 | 4656 | 4705 | 4847 |      |  |
| BIT11 = | 004000 | 1179# | 2169  | 2578  | 2581  | 2668 | 3064  | 3388 | 3902 | 4782 | 4849 | 5307 |      |      |  |
| BIT12 = | 010000 | 1179# | 2578  | 2581  | 3330  | 3358 | 3399  | 3804 | 3809 | 3870 | 4055 | 4313 | 4473 | 4547 |  |
|         |        | 4656  |       |       |       |      |       |      |      |      |      |      |      |      |  |
| BIT13 = | 020000 | 1179# | 2578  | 2581  | 2649  | 3064 | 3388  | 3491 | 3516 | 4034 | 4311 | 4847 |      |      |  |
| BIT14 = | 040000 | 1179# | 1700  | 2163  | 2630  | 2857 | 2859  | 2861 | 2863 | 2865 | 3146 | 3497 | 3591 | 3673 |  |
|         |        | 3793  | 3795  | 3897  | 4076  | 4589 | 4705  | 4849 | 5089 | 5112 | 5307 |      |      |      |  |
| BIT15 = | 100000 | 1179# | 2169  | 2578  | 2581  | 3381 | 3392  | 3403 | 3442 | 3565 | 3609 | 3902 | 4087 | 4589 |  |
|         |        | 4703  | 5071  | 5075  | 5089  | 5112 | 5307  |      |      |      |      |      |      |      |  |
| BIT2 =  | 000004 | 1179# | 2649  | 2668  | 2857  | 3164 | 3399  | 3516 | 4083 | 4449 | 4703 |      |      |      |  |







|        |        |       |       |       |       |       |       |       |       |       |       |       |       |       |
|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| JOB7S  | 056114 | 5543# |       |       |       |       |       |       |       |       |       |       |       |       |
| JOB7U  | 056136 | 4743  | 5543# |       |       |       |       |       |       |       |       |       |       |       |
| KMADO  | 001466 | 1316# | 1523  | 1662* | 1664* | 1700* | 1729  | 1754* | 1784  | 1805* | 1859  | 1905* | 1907  | 1976* |
|        |        | 1979  | 2046* | 2050  | 2114* | 2127* | 2163* | 2169* | 2174  | 2178  | 2292  | 2307  | 2364  | 2384  |
|        |        | 3897* | 3902* | 5307* | 5371  |       |       |       |       |       |       |       |       |       |
| KMAD1  | 001470 | 1316# | 1705* |       |       |       |       |       |       |       |       |       |       |       |
| KMAD2  | 001472 | 1316# | 1665* | 1736  | 1790  | 1867  | 1915  | 2194* | 2196  | 2279* | 2301* | 2317* | 2376* | 2393* |
|        |        | 4866  | 5307* |       |       |       |       |       |       |       |       |       |       |       |
| KMAD3  | 001474 | 1316# | 5307* |       |       |       |       |       |       |       |       |       |       |       |
| KMAD4  | 001476 | 1316# | 1666* | 1742  | 1795  | 2199  | 2278* | 2299* | 2321  | 2374* | 2397  | 5307* |       |       |
| KMAD5  | 001500 | 1316# |       |       |       |       |       |       |       |       |       |       |       |       |
| KMAD6  | 001502 | 1316# | 1667* | 1748  | 1800  | 5307* |       |       |       |       |       |       |       |       |
| KMAD7  | 001504 | 1316# | 1526  | 5307* |       |       |       |       |       |       |       |       |       |       |
| KMCSR  | 001466 | 1315# |       |       |       |       |       |       |       |       |       |       |       |       |
| KWADRO | 001674 | 1451# | 1546* | 2632  | 2641  | 2650  | 2651  | 2660  | 2661  | 2700  | 2701  | 2711  | 2712  |       |
| KWADR1 | 001676 | 1453# | 1547* | 1552* | 2699  |       |       |       |       |       |       |       |       |       |
| KWADR2 | 001700 | 1455# | 1548* | 1553* |       |       |       |       |       |       |       |       |       |       |
| KWADR4 | 001702 | 1457# | 1549* | 1554* | 2669  | 2670  | 2680  | 2681  | 2689  | 2690  | 2722  | 2727  | 2736  | 2737  |
| KWADR5 | 001704 | 1459# | 1550* | 1555* | 2721  |       |       |       |       |       |       |       |       |       |
| KWADR6 | 001706 | 1461# | 1551* | 1556* |       |       |       |       |       |       |       |       |       |       |
| KWT    | 044022 | 3938* | 3941  | 4321* | 4324* | 4326* | 4327* | 4328  | 4476* | 4477  | 4550* | 4551  | 4644* | 4645  |
|        |        | 4779* | 4780* | 4888  | 5016* | 5019* | 5185  | 5482# |       |       |       |       |       |       |
| KWVECL | 002010 | 1514# | 1642* |       |       |       |       |       |       |       |       |       |       |       |
| KWVECP | 002012 | 1515# | 1643* |       |       |       |       |       |       |       |       |       |       |       |
| KW11K  | 001570 | 1401# | 1545  | 4027  |       |       |       |       |       |       |       |       |       |       |
| KW11L  | 002000 | 1509# | 1631  | 1632  |       |       |       |       |       |       |       |       |       |       |
| KW11P  | 002002 | 1510# | 1635  | 1636  |       |       |       |       |       |       |       |       |       |       |
| LF =   | 000012 | 1179# | 4854  |       |       |       |       |       |       |       |       |       |       |       |
| LIST10 | 054716 | 4514  | 4597  | 5527# |       |       |       |       |       |       |       |       |       |       |
| LIST12 | 054746 | 4517  | 4600  | 4801  | 4810  | 5532# |       |       |       |       |       |       |       |       |
| LOOPJ  | 035616 | 5005  | 5135# |       |       |       |       |       |       |       |       |       |       |       |
| LOPC = | 000074 | 1506# |       |       |       |       |       |       |       |       |       |       |       |       |
| LPADBR | 001732 | 1479# | 1609* | 1618* | 3420  | 3459  |       |       |       |       |       |       |       |       |
| LPADH  | 001500 | 1316# |       |       |       |       |       |       |       |       |       |       |       |       |
| LPADL  | 001476 | 1316# | 3944* | 3982* | 4120* | 4160* | 4198* | 4219* | 4249* | 4259* | 4310* | 4328* | 4477* | 4551* |
|        |        | 4645* | 4888* | 5184* | 5185* | 5237* | 5254  |       |       |       |       |       |       |       |
| LPADSR | 001730 | 1478# | 1608* | 3365  | 3380  | 3390  | 3391  | 3401  | 3402  | 3411  | 3412  | 3422  | 3425  | 3439  |
|        |        | 3441  | 3451  | 3452  |       |       |       |       |       |       |       |       |       |       |
| LPCI   | 001466 | 1316# | 3905  | 3911  | 3945* | 3984* | 4121* | 4124  | 4161* | 4198* | 4200  | 4202  | 4220* | 4249* |
|        |        | 4250* | 4253  | 4254* | 4260* | 4271* | 4310* | 4329* | 4332  | 4334  | 4478* | 4552* | 4646* | 4889* |
|        |        | 5184* | 5186* | 5214* | 5225  | 5241* | 5243* | 5265  |       |       |       |       |       |       |
| LPCKBR | 001736 | 1481# | 1611* | 1620* | 3537  |       |       |       |       |       |       |       |       |       |
| LPCKSR | 001734 | 1480# | 1610* | 1619* | 3499  | 3508  | 3517  | 3518  | 3527  | 3528  | 3538  | 3539  | 3549  | 3550  |
| LPCD   | 001472 | 1316# | 3914  | 3916  | 3947  | 3949  | 3963* | 3987  | 3991  | 4133  | 4164  | 4169  | 4210  | 4223  |
|        |        | 4225  | 4261* | 4266  | 4267* | 4272* | 4339  | 5215* | 5226  | 5255  | 5296* |       |       |       |
| LPDASR | 001746 | 1485# | 1615* | 1624* | 3678  | 3688  | 3696  | 3697  | 3707  | 3708  | 3717  | 3718  | 3802  | 3807  |
|        |        | 3812  | 3816  |       |       |       |       |       |       |       |       |       |       |       |
| LPDAXR | 001750 | 1486# | 1616* | 1625* | 3726  | 3727  | 3744  | 3745  | 3763  | 3764  | 3801  | 3811  |       |       |
| LPDAYR | 001752 | 1487# | 1617* | 1626* | 3734  | 3735  | 3752  | 3753  | 3771  | 3772  | 3806  | 3815  |       |       |
| LPIOIR | 001742 | 1483# | 1613* | 1622* | 3589  | 3627  | 3644  | 3651  | 3654  |       |       |       |       |       |
| LPIOOR | 001744 | 1484# | 1614* | 1623* | 3588  | 3615  | 3641  | 3643  | 3653  |       |       |       |       |       |
| LPIOSR | 001740 | 1482# | 1612* | 1621* | 3570  | 3581  | 3592  | 3593  | 3601  | 3602  | 3616  | 3634  | 3661  |       |
| LPMR   | 001470 | 1316# |       |       |       |       |       |       |       |       |       |       |       |       |
| LPMS1  | 001502 | 1316# | 4143  |       |       |       |       |       |       |       |       |       |       |       |
| LPMS2  | 001504 | 1316# |       |       |       |       |       |       |       |       |       |       |       |       |
| LPSO   | 001474 | 1316# | 3921  | 3924  | 3955  | 3965  | 3968  | 3996  | 4136  | 4173  | 4215  | 4229  | 4341  | 4891  |









|         |         |        |       |       |        |       |       |       |       |       |       |       |       |       |
|---------|---------|--------|-------|-------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| TST45   | 026070  | 4457   | 4471# |       |        |       |       |       |       |       |       |       |       |       |
| TST46   | 026426  | 4510   | 4545# |       |        |       |       |       |       |       |       |       |       |       |
| TST47   | 027024  | 4593   | 4640# |       |        |       |       |       |       |       |       |       |       |       |
| TST5    | 005302  | 2424   | 2578# |       |        |       |       |       |       |       |       |       |       |       |
| TST50   | 030256  | 4761#  |       |       |        |       |       |       |       |       |       |       |       |       |
| TST51   | 030634  | 4822#  |       |       |        |       |       |       |       |       |       |       |       |       |
| TST6    | 005770  | 2578   | 2581# |       |        |       |       |       |       |       |       |       |       |       |
| TST7    | 006456  | 2581   | 2622# |       |        |       |       |       |       |       |       |       |       |       |
| TYPDS = | 104405  | 4837   | 5308# |       |        |       |       |       |       |       |       |       |       |       |
| TYPE =  | 104401  | 1644   | 4837  | 4840  | 4841   | 4847  | 4848  | 4850  | 4854  | 4857  | 4999  | 5006  | 5014  | 5017  |
|         |         | 5023   | 5034  | 5036  | 5037   | 5040  | 5043  | 5054  | 5063  | 5065  | 5077  | 5080  | 5083  | 5086  |
|         |         | 5091   | 5094  | 5097  | 5107   | 5116  | 5121  | 5122  | 5138  | 5166  | 5167  | 5307  | 5308# | 5312  |
|         |         | 5317   | 5323  | 5324  | 5326   |       |       |       |       |       |       |       |       |       |
|         |         | 4848   | 4850  | 5035  | 5055   | 5064  | 5307  | 5308# | 5325  | 5327  |       |       |       |       |
| TYPOC = | 104402  | 5308#  |       |       |        |       |       |       |       |       |       |       |       |       |
| TYPON = | 104404  | 5307   | 5308# |       |        |       |       |       |       |       |       |       |       |       |
| TYPOS = | 104403  | 5307   | 5308# |       |        |       |       |       |       |       |       |       |       |       |
| USJNO   | 001756  | 1492#  | 5258* | 5380  |        |       |       |       |       |       |       |       |       |       |
| USRDA   | 001760  | 1493#  |       |       |        |       |       |       |       |       |       |       |       |       |
| USWH =  | 000007  | 1508#  |       |       |        |       |       |       |       |       |       |       |       |       |
| VBM =   | 000007  | 1507#  | 5288  |       |        |       |       |       |       |       |       |       |       |       |
| VECTOR  | 001506  | 1316#  |       |       |        |       |       |       |       |       |       |       |       |       |
| VECTPS  | 001510  | 1316#  |       |       |        |       |       |       |       |       |       |       |       |       |
| VECT1   | 001556  | 1319#  | 1531* | 1532* | 1533   | 2112* | 2124* | 2125* | 4246  | 5209  |       |       |       |       |
| VECT2   | 001560  | 1321#  | 1533* | 1534* | 2123*  | 2137* | 2138* |       |       |       |       |       |       |       |
| VERSN   | 001512  | 1316#  | 2198  | 3896  | 4117   | 4198  | 4249  | 4310  | 5184  |       |       |       |       |       |
| WIR     | 034162  | 2287   | 2302  | 2319  | 2360   | 2377  | 2395  | 4865# |       |       |       |       |       |       |
| WIRT    | 034210  | 4865*  | 4868* | 4875# |        |       |       |       |       |       |       |       |       |       |
| W1      | 037712  | 5307#* |       |       |        |       |       |       |       |       |       |       |       |       |
| W2      | 037714  | 5307#* |       |       |        |       |       |       |       |       |       |       |       |       |
| W3      | 037716  | 5307#* |       |       |        |       |       |       |       |       |       |       |       |       |
| SAERR   | 037442  | 2578   | 2581  | 2634  | 2857   | 2859  | 2861  | 2863  | 2865  | 2883  | 3051  | 3150  | 3222  | 3374  |
|         |         | 3501   | 3571  | 3679  | 5307#* |       |       |       |       |       |       |       |       |       |
| SAPTHD  | 001000  | 1186#  |       |       |        |       |       |       |       |       |       |       |       |       |
| SASTAT= | ***** U | 4856   |       |       |        |       |       |       |       |       |       |       |       |       |
| SATYC   | 033520  | 4856#  |       |       |        |       |       |       |       |       |       |       |       |       |
| SATY1   | 033474  | 4856#  |       |       |        |       |       |       |       |       |       |       |       |       |
| SATY3   | 033502  | 4854   | 4856# |       |        |       |       |       |       |       |       |       |       |       |
| SATY4   | 033512  | 4847   | 4856# |       |        |       |       |       |       |       |       |       |       |       |
| SAUTOB  | 001134  | 1187#  | 1644* | 4850  |        |       |       |       |       |       |       |       |       |       |
| SBASE   | 001250  | 1187#  | 1522  |       |        |       |       |       |       |       |       |       |       |       |
| SRDADR  | 001122  | 1187#  |       |       |        |       |       |       |       |       |       |       |       |       |
| SBDDAT  | 001126  | 1187#  | 1727* | 1729* | 1730   | 1736* | 1737  | 1742* | 1743  | 1748* | 1749  | 1784* | 1786* | 1790* |
|         |         | 1795*  | 1800* | 1975* | 1986   | 2043* | 2058  | 2199* | 2208* | 2210  | 2291* | 2292* | 2294  | 2321* |
|         |         | 2324   | 2363* | 2364* | 2367   | 2397* | 2400  | 2578* | 2581* | 2641  | 2643  | 2651  | 2653  | 2661  |
|         |         | 2662   | 2670  | 2672  | 2681   | 2682  | 2690  | 2691  | 2698* | 2699  | 2701  | 2705  | 2712  | 2713  |
|         |         | 2720*  | 2721  | 2727  | 2729   | 2737  | 2738  | 2857  | 2859  | 2861  | 2863  | 2865  | 2892  | 2893* |
|         |         | 2901   | 2903* | 2904  | 2912   | 2913* | 2914  | 2922  | 2923* | 2931  | 2932  | 2939  | 2940  | 2947  |
|         |         | 2948   | 2955  | 2956  | 2964   | 2965  | 2972  | 2973  | 2980  | 2981  | 2988  | 2989  | 2998  | 2999  |
|         |         | 3006   | 3007  | 3014  | 3015   | 3022  | 3023  | 3040* | 3042  | 3056  | 3057* | 3059  | 3067  | 3068* |
|         |         | 3069   | 3078  | 3079* | 3080   | 3088  | 3089* | 3090  | 3101  | 3103  | 3156  | 3158  | 3166  | 3168  |
|         |         | 3176   | 3178  | 3185* | 3186   | 3188  | 3192  | 3199  | 3201  | 3231  | 3232* | 3240  | 3242* | 3243  |
|         |         | 3251   | 3252* | 3253  | 3261   | 3262* | 3270  | 3271  | 3278  | 3279  | 3287  | 3288  | 3294  | 3295  |
|         |         | 3303   | 3304  | 3311  | 3312   | 3334* | 3341  | 3343* | 3344  | 3350  | 3352* | 3364* | 3365  | 3380  |
|         |         | 3381*  | 3383  | 3391  | 3392*  | 3393  | 3402  | 3403* | 3404  | 3412  | 3413* | 3414  | 3425  | 3427  |
|         |         | 3441   | 3444  | 3452  | 3453*  | 3454  | 3508  | 3510  | 3518  | 3520  | 3528  | 3529  | 3536* | 3537  |
|         |         | 3539   | 3543  | 3550  | 3551   | 3581  | 3582  | 3593  | 3594  | 3602  | 3603  | 3610  | 3618  | 3627  |

CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0209

|         |        |        |       |       |       |       |       |       |       |       |       |       |       |       |
|---------|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|         |        | 3628   | 3634  | 3635  | 3644  | 3645  | 3654  | 3655  | 3688  | 3689* | 3697  | 3699* | 3700  | 3708  |
|         |        | 3709*  | 3710  | 3718  | 3719* | 3727  | 3728  | 3735  | 3736  | 3745  | 3746  | 3753  | 3754  | 3764  |
|         |        | 3765   | 3772  | 3773  | 3799* | 3806  | 3808* | 3809  | 3815  | 3817* | 3870* | 3911* | 3916* | 3924* |
|         |        | 3949*  | 3954* | 3955* | 3956  | 3968* | 3991* | 3996* | 3997  | 4129* | 4136* | 4138  | 4169* | 4173* |
|         |        | 4174   | 4202* | 4210* | 4212  | 4215* | 4225* | 4229* | 4230  | 4253* | 4266* | 4334* | 4341* | 4449* |
|         |        | 4451*  | 4453* | 4455* | 4457* | 4891* | 5029* | 5035  | 5126  | 5128* | 5206* | 5207* | 5220  | 5226* |
|         |        | 5270*  | 5275* | 5373  | 5376  |       |       |       |       |       |       |       |       |       |
| SBELL.  | 001164 | 1187#  | 4847  |       |       |       |       |       |       |       |       |       |       |       |
| SCDW1   | 001254 | 1187#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SCHARC  | 033366 | 4854#* |       |       |       |       |       |       |       |       |       |       |       |       |
| SCKSWR  | 032334 | 4850#  | 5308  |       |       |       |       |       |       |       |       |       |       |       |
| SCMTAG  | 001100 | 1187#  | 1520  |       |       |       |       |       |       |       |       |       |       |       |
| SCM3 =  | 000000 | 1187#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SCNT    | 040412 | 5307#* |       |       |       |       |       |       |       |       |       |       |       |       |
| SCNTLG  | 033061 | 4850#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SCNTLU  | 033054 | 4850#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SCPUOP  | 001222 | 1187#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SCRFL   | 001171 | 1187#  | 4847  | 4848  | 4850  | 4854  |       |       |       |       |       |       |       |       |
| SDATR   | 040064 | 5307#* |       |       |       |       |       |       |       |       |       |       |       |       |
| SDBLK   | 031512 | 4841#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SDEVCT  | 001204 | 1187#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SDEVM   | 001252 | 1187#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SDOAGN  | 031040 | 4837#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SDTBL   | 031502 | 4841#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SENDAD  | 031030 | 1183   | 4837# |       |       |       |       |       |       |       |       |       |       |       |
| SENDCT  | 030672 | 4837#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SENULL  | 031044 | 4837#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SENV    | 001214 | 1187#  | 1644  | 4847  | 4854  | 4856  |       |       |       |       |       |       |       |       |
| SEVM    | 001215 | 1187#  | 1520  | 4854  | 4856  |       |       |       |       |       |       |       |       |       |
| SEOP    | 030636 | 4837#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SEOPCT  | 030664 | 4837#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SERFLG  | 001103 | 1187#  | 4847* | 4849* |       |       |       |       |       |       |       |       |       |       |
| SERMAX  | 001115 | 1187#  | 1520* | 4849* |       |       |       |       |       |       |       |       |       |       |
| SERROR  | 031522 | 1520   | 4847# |       |       |       |       |       |       |       |       |       |       |       |
| SERRPC  | 001116 | 1187#  | 4847* | 4848  | 5371  | 5373  | 5375  | 5376  | 5380  |       |       |       |       |       |
| SERRTB  | 001256 | 1187#  | 4848  |       |       |       |       |       |       |       |       |       |       |       |
| SERRTY  | 031716 | 4847   | 4848# |       |       |       |       |       |       |       |       |       |       |       |
| SERTTL  | 001112 | 1187#  | 4847* |       |       |       |       |       |       |       |       |       |       |       |
| SESCAP  | 001162 | 1187#  | 1520* | 4847  | 4849* |       |       |       |       |       |       |       |       |       |
| SETABL  | 001214 | 1187#  |       |       |       |       |       |       |       |       |       |       |       |       |
| SETEND  | 001256 | 1186   | 1187# |       |       |       |       |       |       |       |       |       |       |       |
| \$FATAL | 001176 | 1187#  | 4856* |       |       |       |       |       |       |       |       |       |       |       |
| \$FFLG  | 033740 | 4856#* |       |       |       |       |       |       |       |       |       |       |       |       |
| \$FILLC | 001156 | 1187#  | 4854  |       |       |       |       |       |       |       |       |       |       |       |
| \$FILLS | 001155 | 1187#  | 4854  |       |       |       |       |       |       |       |       |       |       |       |
| \$GDADR | 001120 | 1187#  |       |       |       |       |       |       |       |       |       |       |       |       |
| \$GDDAT | 001124 | 1187#  | 1728* | 1730  | 1735* | 1737  | 1743  | 1749  | 1783* | 1974* | 1986  | 2044* | 2058  | 2198* |
|         |        | 2210   | 2290* | 2294  | 2322* | 2324  | 2362* | 2367  | 2398* | 2400  | 2578* | 2581* | 2630* | 2632  |
|         |        | 2643   | 2649* | 2650  | 2653  | 2659* | 2660  | 2668* | 2669  | 2672  | 2678* | 2680  | 2682  | 2688* |
|         |        | 2689   | 2697* | 2700  | 2703* | 2705  | 2710* | 2711  | 2719* | 2722  | 2728* | 2729  | 2735* | 2736  |
|         |        | 2857*  | 2859* | 2861* | 2863* | 2865* | 2881* | 2882  | 2899* | 2900  | 2904  | 2910* | 2911  | 2914  |
|         |        | 2920*  | 2921  | 2929* | 2930  | 2932  | 2938  | 2940  | 2946  | 2948  | 2954  | 2956  | 2962* | 2963  |
|         |        | 2965   | 2971  | 2973  | 2979  | 2981  | 2987  | 2989  | 2996* | 2997  | 2999  | 3005  | 3007  | 3013  |
|         |        | 3015   | 3021  | 3023  | 3041* | 3058* | 3064* | 3066  | 3069  | 3075* | 3077  | 3080  | 3086* | 3087  |
|         |        | 3097*  | 3098  | 3099* | 3103  | 3146* | 3148  | 3158  | 3164* | 3165  | 3168  | 3174* | 3175  | 3177* |
|         |        | 3178   | 3184* | 3187  | 3190* | 3192  | 3197* | 3198  | 3200* | 3201  | 3220* | 3221  | 3238* | 3239  |













|         |       |       |
|---------|-------|-------|
| .KMADR  | 55#   | 1316  |
| .KSIS   | 184#  |       |
| .LOADL  | 459#  | 5307  |
| .LPAIN  | 209#  | 5307  |
| .PUTCS  | 418#  | 5307  |
| .RESET  | 329#  | 5307  |
| .SETTR  | 1154# |       |
| .SETUP  | 1154# | 1518  |
| .SWRHI  | 1155# | 1163  |
| .SWRLO  | 1163# |       |
| .TRMTR  | 1154# |       |
| .UTK    | 699#  | 5307  |
| .SACT1  | 1157# | 1183  |
| .SAPT8  | 1157# | 1187# |
| .SAPTH  | 1157# | 1186  |
| .SAPTY  | 1157# | 4856  |
| .SCATC  | 1155# | 1165  |
| .SCMTA  | 1155# | 1187  |
| .\$DB2D | 1157# |       |
| .\$EOP  | 1156# | 4837  |
| .\$ERRO | 1156# | 4847  |
| .\$ERRT | 1156# | 4848  |
| .\$INLP | 652#  | 5307  |
| .\$MMAC | 141#  |       |
| .\$OUTL | 610#  | 5307  |
| .\$POWE | 1155# | 4857  |
| .\$RDGC | 1154# | 4855  |
| .\$READ | 1156# | 4850  |
| .\$SB2D | 1157# |       |
| .\$SC7P | 1156# | 4849  |
| .\$TLKW | 511#  | 5307  |
| .\$TOUT | 1316# | 5307  |
| .\$TRAP | 1154# | 5308  |
| .\$TYPD | 1156# | 4841  |
| .\$TYPE | 1156# | 4854  |
| .\$TYPO | 1155# | 4840  |

|        |        |     |     |    |     |     |   |
|--------|--------|-----|-----|----|-----|-----|---|
| . ABS. | 063166 | 000 | OVR | RO | REL | GBL | I |
|        | 000000 | 001 | CON | RW | REL | LCL | I |

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

CRLPAB,CRLPAB/CRF=CRLPAB.MAC,CRLPAB.P11  
RUN-TIME: 31 27 2 SECONDS  
RUN-TIME RATIO: 95/61=1.5  
CORE USED: 41K (81 PAGES)

;THIS FILE IS THE SAME AS 'CRLPX2.P11' EXCEPT LOADED INTO LOC. 64000  
;IT IS ALSO THE SAME AS 'DRLPX0.P11' EXCEPT NAME CHANGE TO 'CRLPX0.P11'

.LIST MC,BIN,BEX,MEB  
.NLIST MD,CND,ME

177777

ADDRESS=-1  
MACRO DEFFINITIONS FOR M8200 AND M8204 MICRO-PROCESSOR  
INSTRUCTION SET.  
TO BE USED WITH RSX MACRO-11 ASSEMBLER

26-MAY-1976  
\$BEGIN  
\$LOC 64000  
.GLOBL DRLPX0,DRLPY2  
.ENABL GBL

;\*  
;\*MICRO CODE FOR KMC-11

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
452  
453  
454  
455  
456  
457  
458

000000'  
000000

460  
461  
462  
463  
464  
465 064000  
466 064000  
467 064000  
(2) 064000 100407  
468 064002  
(2) 064002 100420  
469 064004  
(2) 064004 100430  
470 064006  
(2) 064006 100432  
471 064010  
(2) 064010 100434  
472 064012  
(2) 064012 100436  
473 064014  
(2) 064014 100440  
474  
475  
476  
477  
478 064016  
479 064016  
(3) 064016 000400  
480 064020  
(3) 064020 061220  
481 064022  
(3) 064022 061222  
482 064024  
(3) 064024 061223  
483 064026  
(3) 064026 061224  
484 064030  
(3) 064030 061225  
485 064032  
(3) 064032 061226  
486 064034  
(3) 064034 061227  
487 064036  
(3) 064036 063226  
488  
489 064040  
(3) 064040 021240  
490 064042  
(3) 064042 000777  
491 064044  
(3) 064044 061222  
492  
493 064046  
(3) 064046 021240  
494  
495 064050

;\*THIS CODE WILL BE DOWN LOADED INTO BOTH  
;\*KMC-11'S. THE CODE RUNS ASYNCHRONOUS TO THE PDP-11 CODE  
;\*WE SYNC THROUGH COMMANDS PASSED VIA THE OUT\*/IBUS\* REGS.  
;\*

DRLPX2:  
DRLPX0: ;JUMP TABLE USED FOR COMMANDS  
BR STARTU ;GOTO START  
.WORD .\$\$\$.  
BR CMNOP ;NOP=1  
.WORD .\$\$\$.  
BR RDSILO ;=2 READ SILO PUT IN BSEL4  
.WORD .\$\$\$.  
BR WRSILO ;=3 READ BSEL4 PUT IN SILO.  
.WORD .\$\$\$.  
BR RDCMND ;=4 READ FAST PATH PUT IN BSEL4  
.WORD .\$\$\$.  
BR WRCMND ;=5 READ BSEL4, PUT IN FAST PATH.  
.WORD .\$\$\$.  
BR SAMP ;=6 TAKE AN A/D SAMPLE  
.WORD .\$\$\$.  
;START OF U CODED

STARTU:  
MOVE # 0,BREG  
.WORD .\$\$\$.  
MOVE BREG,OUT1 <0> ;CLEAR UNIBUS CSRS  
.WORD .\$\$\$.  
MOVE BREG,OUT1 <2>  
.WORD .\$\$\$.  
MOVE BREG,OUT1 <3>  
.WORD .\$\$\$.  
MOVE BREG,OUT1 <4>  
.WORD .\$\$\$.  
MOVE BREG,OUT1 <5>  
.WORD .\$\$\$.  
MOVE BREG,OUT1 <6>  
.WORD .\$\$\$.  
MOVE BREG,OUT1 <7>  
.WORD .\$\$\$.  
MOVE BREG,SPAD <6>  
.WORD .\$\$\$.  
CMNOP: MOVE INP0 <12>,OUT1 <0> ;READ STATUS  
.WORD .\$\$\$.  
MOVE # 377,BREG  
.WORD .\$\$\$.  
MOVE BREG,OUT1 <2> ;INDICATE READY FOR COMMAND.  
.WORD .\$\$\$.  
LOOP: MOVE INP0 <12>,OUT1 <0> ;READ STATUS  
.WORD .\$\$\$.  
MOVE INP1 <2>,SPAD <0> ;READ COMMAND REG.

```
(3) 064050 123040 .WORD .$$$.
496 064052 BZ LOOP ;NO COMMAND THEN LOOP
(2) 064052 101423 .WORD .$$$.
497
498 064054 MOVE INP1 <2> SPAD <0> ;RE-READ COMMAND.
(3) 064054 123040 .WORD .$$$.
499
500 064056 BR SPAD <0> ;BR BASED ON CMND.
(2) 064056 160600 .WORD .$$$.
501 ;NO-USER PROTECTION OFFERED.
502 ;IF YOU ENTER WRONG CODE -
503 ;YOU LOSE.
504
505
506 ;ROUTINE TO READ THE SILO, PUT IN
507 ;*BUS REG 4
508 ;CMD=2
509
510 RDSILO: MOVE INP0 <10>,OUT1 <4> ;READ SILO.
(3) 064060 021204 .WORD .$$$.
511 ;WRITE *BUS
512 064062 BR CMNOP ;RETURN.
(2) 064062 100420 .WORD .$$$.
513
514 ;ROUTINE TO WRITE SILO, READ DATA FROM
515 ;*BUS REG 4
516 ;CMD=3
517
518
519
520 WRSILO: MOVE INP1 <4>,OUT0 <10> ;READ DATA IN *BUS
(3) 064064 122110 .WORD .$$$.
521 ;WRITE SILO.
522 064066 BR CMNOP
(2) 064066 100420 .WORD .$$$.
523
524 ;ROUTINE TO READ FAST PATH (CMND) REG.
525 ;PUT IN *BUS REG 4
526 ;CMD=4
527
528
529
530 RDCMND: MOVE INP0 <11>,OUT1 <4> ;READ FAST PATH
(3) 064070 021224 .WORD .$$$.
531 ;WRITE *BUS.
532 064072 BR CMNOP ;RETURN
(2) 064072 100420 .WORD .$$$.
533
534 ;ROUTINE TO WRITE FAST PATH (CMND) REG.
535 ;TAKE DATA FROM *BUS REG 4.
536 ;CMD=5
537
538
539
540 WRCMND: MOVE INP1 <4>,OUT0 <11> ;READ DATA IN *BUS
(3) 064074 122111 .WORD .$$$.
```

```
541
542 064076 BR CMNOP ;WRITE INTO FAST PATH.
(2) 064076 100420 .WORD .$$$.;RETURN.
543
544
545
546 ;
547 ;THIS ROUTINE TAKES AN A/D SAMPLE.
548 ;CALL= CMND 6 IN BSEL2
549 ;THESE REGS. MUST BE SET UP IN ADVANCE.
550 ;BSEL 3 MUST CONTAIN READ CODE FOR A/D BUFFER.
551 ;BSEL 4,5 MUST CONTAIN A/D CSR SETTING.
552 ;BSEL 6 MUST CONTAIN WRITE CODE FOR A/D CSR
553 ;BSEL 7 MUST CONTAIN READ CODE FOR A/D CSR
554 ; BSEL 3,6,7 WILL REMAIN UNEFFECTIVE.
555 ; BSEL 4,5 WILL CONTAIN A/D SAMPLE.
556 ;BSEL2 WILL CONTAIN CODE 377 WHEN DONE.
567 064100 WPMC SAMP
(4) 064100 020640 .WORD .$$$.
(3) 064102 103040 .WORD .$$$.
568 064104 MOVE INP1 <6>,OUT0 <11> ;SEND A/D WRITE CODE.
(3) 064104 122151 .WORD .$$$.
569 064106 WPMC SAMP1
(4) 064106 020640 .WORD .$$$.
(3) 064110 103043 .WORD .$$$.
570 064112 MOVE INP1 <4>,OUT0 <11> ;SEND LOW BYTE CSR INFO.
(3) 064112 122111 .WORD .$$$.
571 064114 WPMC SAMP2
(4) 064114 020640 .WORD .$$$.
(3) 064116 103046 .WORD .$$$.
572 064120 MOVE INP1 <5> OUT0 <11> ;SEND HIGH BYTE CSR INFO.
(3) 064120 122131 .WORD .$$$.
573 064122 WPMC SLOOP
(4) 064122 020640 .WORD .$$$.
(3) 064124 103051 .WORD .$$$.
574 064126 MOVE INP1 <7>,OUT0 <11> ;SEND READ CODE TO GET A/D CSR.
(3) 064126 122171 .WORD .$$$.
575 064130 WPMC SAMP3
(4) 064130 020640 .WORD .$$$.
(3) 064132 103054 .WORD .$$$.
576 064134 WPMC SLOOP1
(4) 064134 020640 .WORD .$$$.
(4) 064136 061620 .WORD .$$$.
(3) 064140 103056 .WORD .$$$.
577 064142 MOVE INP0 <11>,BREG
(3) 064142 020620 .WORD .$$$.
578 064144 MOVE BREG,SPAD <0>
(3) 064144 063220 .WORD .$$$.
579 064146 WPMC SLOOP2
(4) 064146 020640 .WORD .$$$.
(4) 064150 061620 .WORD .$$$.
(3) 064152 103063 .WORD .$$$.
580 064154 MOVE INP0 <11>,BREG
(3) 064154 020620 .WORD .$$$.
581 064156 BB7 CMNOP ;ABORT IF A/D BIT 15=1
```

|     |        |        |       |                    |                                            |
|-----|--------|--------|-------|--------------------|--------------------------------------------|
| (2) | 064156 | 103420 | .WORD | .SSS.              |                                            |
| 582 | 064160 |        | MOVE  | SPAD <0>,BREG      |                                            |
| (3) | 064160 | 060600 | .WORD | .SSS.              |                                            |
| 583 | 064162 |        | BB7   | LOPE               |                                            |
| (2) | 064162 | 103473 | .WORD | .SSS.              |                                            |
| 584 | 064164 |        | BR    | SLOOP              | :IF A/D NOT DONE,EXIT.                     |
| (2) | 064164 | 100451 | .WORD | .SSS.              |                                            |
| 585 | 064166 |        | LOPE: | MOVE               | INP1 <3>,OUT0 <11> :ISSUE READ A/B BUFFER. |
| (3) | 064166 | 122071 | .WORD | .SSS.              |                                            |
| 586 | 064170 |        | WTMM  | SLOOP3             |                                            |
| (4) | 064170 | 020640 | .WORD | .SSS.              |                                            |
| (4) | 064172 | 061620 | .WORD | .SSS.              |                                            |
| (3) | 064174 | 103074 | .WORD | .SSS.              |                                            |
| 587 | 064176 |        | MOVE  | INPO <11>,OUT1 <4> |                                            |
| (3) | 064176 | 021224 | .WORD | .SSS.              |                                            |
| 588 | 064200 |        | WTMM  | SLOOP4             |                                            |
| (4) | 064200 | 020640 | .WORD | .SSS.              |                                            |
| (4) | 064202 | 061620 | .WORD | .SSS.              |                                            |
| (3) | 064204 | 103100 | .WORD | .SSS.              |                                            |
| 589 | 064206 |        | MOVE  | INPO <11>,OUT1 <5> |                                            |
| (3) | 064206 | 021225 | .WORD | .SSS.              |                                            |
| 590 | 064210 |        | BR    | CMNOP              |                                            |
| (2) | 064210 | 100420 | .WORD | .SSS.              |                                            |
| 591 | 064212 | 177777 | .WORD | -1                 |                                            |
| 592 |        | 000001 | .END  |                    |                                            |

|                 |                   |                |                |                   |
|-----------------|-------------------|----------------|----------------|-------------------|
| ADDRES= 177777  | RDSILO 064060     | .ADD = 000000  | .DEC = 000160  | .SELA = 000200    |
| CLK = 000020    | SAMP 064100       | .ADDWC= 000020 | .DMEM = 002400 | .SELB = 000220    |
| CMNOP 064040    | SAMP1 064106      | .AND = 000260  | .DNOP = 000000 | .SIMM = 000000    |
| DRLPX0 064000 G | SAMP2 064114      | .BB0 = 002000  | .DOUT0= 002000 | .SINO = 020000    |
| DRLPX2 064000 G | SAMP3 064130      | .BB1 = 002400  | .DOUT1= 001000 | .SIN1 = 120000    |
| LOOP 064046     | SLOOP 064122      | .BB4 = 003000  | .DSPAD= 003000 | .SMEM = 040000    |
| LOPE 064166     | SLOOP1 064134     | .BB7 = 003400  | .DSPBR= 003400 | .SUB = 000340     |
| MARHLD= 000000  | SLOOP2 064146     | .BC = 001000   | .DO = 000400   | .SUBWC= 000040    |
| MARINC= 014000  | SLOOP3 064170     | .BR = 000400   | .FO = 000020   | .SUB2C= 000360    |
| MARLD = 010000  | SLOOP4 064200     | .BSBRG= 160000 | .INC = 000060  | .SO = 020000      |
| MARLDX= 004000  | STARTU 064016     | .BSIMM= 100000 | .LORN = 000240 | .XOR = 000320     |
| PAGE0 = 000000  | WRCMND 064074     | .BSMEM= 140000 | .MINUS= 000360 | ..\$\$\$ = 100420 |
| PAGE1 = 001000  | WRSILO 064064     | .BZ = 001400   | .MO = 004000   | ..LOC = 064040    |
| PAGE2 = 002000  | \$\$\$SER= 000001 | .CO = 000400   | .OR = 000300   | .2A = 000120      |
| PAGE3 = 003000  | . = 064214        | .DBR = 000400  | .PLUS = 000000 | .2AWC = 000140    |
| RDCMND 064070   | .ADC = 000100     | .DBRSH= 001400 | .SBREG= 060000 |                   |

|        |        |     |     |    |     |     |   |
|--------|--------|-----|-----|----|-----|-----|---|
| . ABS. | 064214 | 000 | OVR | RW | ABS | LCL | D |
|        | 000000 | 001 | CON | RW | ABS | LCL | I |
| ABCODE | 00400C | 002 | CUN | RW | REL | LCL | I |

ERRORS DETECTED: 0  
 DEFAULT GLOBALS GENERATED: 0  
 CRLPX0,CRLPX0=CRLPX0  
 RUN-TIME: 3 3 0 SECONDS  
 RUN-TIME RATIO: 13/7=1.8  
 CORE USED: 41K (81 PAGES)

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
479  
480  
481  
482  
483  
484  
485  
486

:SAME AS 'DRLPX1.P11' EXCEPT NAME CHANGE TO 'CRLPX1.P11'

.LIST MC,BIN,BEX,MEB  
.NLIST MD,CND,ME

.REM %

COPYRIGHT (C) 1975, 1976, 1977, 1980  
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

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

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

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

%

.REM %

THIS CODE WAS DEVELOPED FOR USE WITH THE  
LPA-11 DIAGNOSTIC, BY EDWARD C. BADGER.

%

177777

ADDRESS=-1  
: MACRO DEFFINITIONS FOR M800 AND M8204 MICR-PROCESSOR  
: INSTRUCTION SET.  
: TO BE USED WITH RSX MACRO-11 ASSEMBLER

26-MAY-1976

: THIS IS A MODIFIED VERSION OF THE MACROS WRITTEN BY W C BROWN  
: MODIFIED FOR USE WITH MACY11 AND LNKX11.  
: WARNING: 'BAB' BITS TAKEN OUT OF BRANCH INSTRUCTIONS.

\$BEGIN  
\$LOC 65000  
.GLOBL DRLPX1  
.ENABL GBL

;\*  
;\*MICRO CODE FOR KMC-11

DRLPX1:

000000'  
000000  
  
  
  
  
065000



```
488
489
490
491
492
493
494
495
496
497
498 065000 TST1: MOVE # 377,BREG ;GET ALL ONES
 (3) 065000 000777 .WORD .$$$
499 065002 MOVE BREG,OUT1 <0> ;PUT ALL ONES INTO CRAM
 (3) 065002 061220 .WORD .$$$
500 065004 MOVE BREG,OUT1 <2> ;
 (3) 065004 061222 .WORD .$$$
501 065006 MOVE BREG,OUT1 <3> ;
 (3) 065006 061223 .WORD .$$$
502 065010 MOVE BREG,OUT1 <4> ;
 (3) 065010 061224 .WORD .$$$
503 065012 MOVE BREG,OUT1 <5> ;
 (3) 065012 061225 .WORD .$$$
504 065014 MOVE BREG,OUT1 <6> ;
 (3) 065014 061226 .WORD .$$$
505 065016 MOVE BREG,OUT1 <7> ;
 (3) 065016 061227 .WORD .$$$
506
507 065020 A1: MOVE INP1 <0>,BREG ;WHEN PDP-11 DONE VERIFICATION
 (3) 065020 120400 .WORD .$$$
508 065022 BZ A1 ;IT WILL CLEAR CARM ADDR. #0.
 (2) 065022 101410 .WORD .$$$
509
510
511
512
513
514
515
516 065024 TST2: MOVE # 0,BREG ;GET ZERO
 (3) 065024 000400 .WORD .$$$
517 065026 MOVE BREG,OUT1 <0> ;PUT INTO ALL CRAM
 (3) 065026 061220 .WORD .$$$
518 065030 MOVE BREG,OUT1 <2> ;
 (3) 065030 061222 .WORD .$$$
519 065032 MOVE BREG,OUT1 <3> ;
 (3) 065032 061223 .WORD .$$$
520 065034 MOVE BREG,OUT1 <4> ;
 (3) 065034 061224 .WORD .$$$
521 065036 MOVE BREG,OUT1 <5> ;
 (3) 065036 061225 .WORD .$$$
522 065040 MOVE BREG,OUT1 <6> ;
 (3) 065040 061226 .WORD .$$$
523 065042 MOVE BREG,OUT1 <7> ;
 (3) 065042 061227 .WORD .$$$
524
525 065044 A2: MOVE INP1 <0>,BREG ;GET ZERO
```

```
(3) 065044 120400 .WORD .$$$.
526 065046 BZ TST3 ;WHEN =377, PDP-11 READY FOR NEXT TEST.
(2) 065046 101425 .WORD .$$$.
527 065050 BR A2
(2) 065050 100422 .WORD .$$$.
528
529
530 :TEST #3
531 : BRANCH TEST
532
533 : ENTERED WHEN PDP-11 RETURNS 377 TO ADDR. 0
534
535 :NOTE WE HAVE TO ASSUME BR AND BZ WORK
536 :OR WE COULDN'T HAVE GOT THIS FAR.
537 :UPON SUCCESSFUL COMPLETION OF THIS TEST WE
538 :PUT CODE 377 INTO CRAM ADDR. 2
539 :AND 0 INTO CRAM ADDR. 0. ON FAILURE
540 :WE ONLY ZERO CRAM ADDR. 0.
541
542 065052 TST3: MOVE # 376,BREG ;GET ZERO BIT 0
(3) 065052 000776 .WORD .$$$.
543 065054 BBO T3E ;ERROR IF BR.
(2) 065054 102062 .WORD .$$$.
544 065056 MOVE # 375,BREG ;BIT 1 = 0
(3) 065056 000775 .WORD .$$$.
545 065060 BB1 T3E ;ERROR IF BR
(2) 065060 102462 .WORD .$$$.
546 065062 MOVE # 357,BREG ;BIT 4 = 0
(3) 065062 000757 .WORD .$$$.
547 065064 BB4 T3E ;ERROR IF BR
(2) 065064 103062 .WORD .$$$.
548 065066 MOVE # 177,BREG ;BIT 7 = 0
(3) 065066 000577 .WORD .$$$.
549 065070 BB7 T3E ;ERROR IF BR
(2) 065070 103462 .WORD .$$$.
550 065072 MOVE # 1,BREG ;POSITIVE BR TEST
(3) 065072 000401 .WORD .$$$.
551 065074 BBO A3 ;SHOULD BRANCH
(2) 065074 102040 .WORD .$$$.
552 065076 BR T3E ;IF NOT, ERROR
(2) 065076 100462 .WORD .$$$.
553 065100 A3: MOVE # 2,BREG ;BR ON BIT 1
(3) 065100 000402 .WORD .$$$.
554 065102 BB1 B3 ;SHOULD BR.
(2) 065102 102443 .WORD .$$$.
555 065104 BR T3E ;IF NOT, ERROR.
(2) 065104 100462 .WORD .$$$.
556 065106 B3: MOVE # 20,BREG ;BR ON BIT 4
(3) 065106 000420 .WORD .$$$.
557 065110 BB4 C3 ;SHOULD BR.
(2) 065110 103046 .WORD .$$$.
558 065112 BR T3E ;IF NOT, ERROR.
(2) 065112 100462 .WORD .$$$.
559 065114 C3: MOVE # 200,BREG ;BR IF BIT 7
(3) 065114 000600 .WORD .$$$.
560 065116 BB7 D3 ;SHOULD BR.
```

```

(2) 065116 103451 .WORD .$$$.

561 065120 BR T3E ;IF NOT, ERROR.

(2) 065120 100462 .WORD .$$$.

562 065122 D3: MOVE # 0,BREG ;TEST BZ NEGATIVE

(3) 065122 000400 .WORD .$$$.

563 065124 BZ T3E ;ERROR IF BR.

(2) 065124 101462 .WORD .$$$.

564 065126 MOVE # 377,BREG ;POS BR

(3) 065126 000777 .WORD .$$$.

565 065130 BZ E3 ;SHOULD BR

(2) 065130 101456 .WORD .$$$.

566 065132 BR T3E ;ERROR IF NOT

(2) 065132 100462 .WORD .$$$.

567 065134 E3:

568 065134 MOVE BREG,OUT1 <2> ;PUT 377 IN OUTPUT REG#2

(3) 065134 061222 .WORD .$$$.

569 065136 BR F3 ;IF BR INSTR FAILS TO WORK,

(2) 065136 100462 .WORD .$$$.

570 065140 MOVE # 0,BREG ;THEN A ZERO GETS PUT IN #2

(3) 065140 000400 .WORD .$$$.

571 065142 MOVE BREG,OUT1 <2> ;A SIGN OF AN ERROR.

(3) 065142 061222 .WORD .$$$.

572 065144 F3:

573 065144 T3E: MOVE # 0,BREG ;SIGNAL PDP-11 THAT WE ARE

(3) 065144 000400 .WORD .$$$.

574 065146 MOVE BREG,OUT1 <0> ;THROUGH BR-TEST.

(3) 065146 061220 .WORD .$$$.

575

576

577 :TEST #4

578 :

579 :ALU TEST

580 :

581 :ENTERED WHEN PDP-11 PUTS A CODE 377 INTO

582 :CRAM ADDR 0

583 :ADDR 2 = 0 IF BAD

584 :

585 065150 TST4: MOVE INP1 <0>,BREG ;GET CRAM ADDR 0

(3) 065150 120400 .WORD .$$$.

586 065152 BZ A4 ;WHEN = 377 DO TEST.

(2) 065152 101467 .WORD .$$$.

587 065154 BR TST4

(2) 065154 100464 .WORD .$$$.

588

589 065156 A4: MOVE # 0,BREG ;GET = 0

(3) 065156 000400 .WORD .$$$.

590 065160 MOVE BREG,OUT1 <2>

(3) 065160 061222 .WORD .$$$.

591 065162 MOVE BREG,SPAD <4>

(3) 065162 063224 .WORD .$$$.

592 065164 DEC SPAD <4> ;MAKE = 377

(2) 065164 063164 .WORD .$$$!.DSPAD

593 065166 BZ B4 ;BR IF GOOD SEC.

(2) 065166 101475 .WORD .$$$.

594 065170 BR T4E

(2) 065170 100503 .WORD .$$$.
```

```

595 065172 B4: DEC SPAD <4> ;SEC = 376
(2) 065172 063164 .WORD .$$$!.DSPAD
596 065174 INC SPAD <4> ;INC = 377
(2) 065174 063064 .WORD .$$$!.DSPAD
597 065176 BZ C4 ;BR ;F = 377
(2) 065176 101501 .WORD .$$$
598 065200 BR T4E
(2) 065200 100503 .WORD .$$$
599 065202 C4: MOVE # 377,BREG
(3) 065202 000777 .WORD .$$$
600 065204 MOVE BREG,OUT1 <2> ;RETURN GOOD CODE
(3) 065204 061222 .WORD .$$$
601
602 065206 T4E: MOVE # 0,BREG ;RETURN CODE 0
(3) 065206 000400 .WORD .$$$
603 065210 MOVE BREG,OUT1 <0>
(3) 065210 061220 .WORD .$$$
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618 065212 TST5: MOVE INP1 <0>,BREG ;WAIT FOR PDP-11
(3) 065212 120400 .WORD .$$$
619 065214 BZ A5
(2) 065214 101510 .WORD .$$$
620 065216 BR TST5
(2) 065216 100505 .WORD .$$$
621
622 065220 A5: MOVE # 2,BREG ;SET TO DO NPR IN
(3) 065220 000402 .WORD .$$$
623 065222 MOVE BREG,OUT0 <5> ;SET HIGH ADDR.
(3) 065222 062225 .WORD .$$$
624 065224 MOVE # 124,BREG ;GET LOW ADDR. OF ADDR. 1124
(3) 065224 000524 .WORD .$$$
625 065226 MOVE BREG,OUT0 <4> ;SET LOW ADDR.
(3) 065226 062224 .WORD .$$$
626 065230 MOVE # 1,BREG ;SET TO NPR IN
(3) 065230 000401 .WORD .$$$
627 065232 MOVE BREG,OUT1 <10> ;CAUSE NPR IN
(3) 065232 061230 .WORD .$$$
628 065234 B5: MOVE INP1 <10>,BREG ;WAIT FOR NPR DONE.
(3) 065234 120600 .WORD .$$$
629 065236 BBO B5
(2) 065236 102116 .WORD .$$$
630
631 065240 MOVE INP0 <0>,BREG ;GET LOW BYTE DATA.

```

```

:TEST 5
:
: NPR TEST
:
: ENTERED IN LINE WHEN PDP-11 PUTS CODE 377 INTO
: CRAM ADDR. 0.
: DOES AN INPUT DATA XFER FROM LOC 1124
: IN PDP-11 MEM TO LOC 1126
: DOES NO DATA CHECKING RETURNS
: CODE 0 TO CRAM ADDR. 0 WHEN DONE.
:

```

```

(3) 065240 020400 .WORD .$$$
632 065242 MOVE BREG,OUT0 <2> ;OUTPUT NPR DATA REG.
(3) 065242 062222 .WORD .$$$
633 065244 MOVE INPO <1>,BREG ;GET HIGH BYTE.
(3) 065244 020420 .WORD .$$$
634 065246 MOVE BREG,OUT0 <3> ;SAVE FOR OUTPUT.
(3) 065246 062223 .WORD .$$$
635 065250 MOVE # 2,BREG ;GET OUTPUT ADDR. = 1126
(3) 065250 000402 .WORD .$$$
636 065252 MOVE BREG,OUT0 <7>
(3) 065252 062227 .WORD .$$$
637 065254 MOVE # 126,BREG ;LOW PART
(3) 065254 000526 .WORD .$$$
638 065256 MOVE BREG,OUT0 <6>
(3) 065256 062276 .WORD .$$$
639 065260 MOVE INP1 <11>,SPAD <4> ;/-STN-
(3) 065260 123224 .WORD .$$$
640 065262 AND SPAD <4>,BREG,SPAD
(4) 065262 063264 .WORD .$$$
641 065264 MOVE # 0,BREG
(3) 065264 000400 .WORD .$$$
642 065266 OR SPAD <4>,BREG,BREG
(4) 065266 060704 .WORD .$$$
643 065270 MOVE BREG,OUT1 <11>
(3) 065270 061231 .WORD .$$$
644 065272 MOVE # 21,BREG
(3) 065272 000421 .WORD .$$$
645 065274 MOVE BREG,OUT1 <10> ;SET NPR OUT.
(3) 065274 061230 .WORD .$$$
646
647 065276 C5: MOVE INP1 <10>,BREG ;WAIT TILL DONE
(3) 065276 120600 .WORD .$$$
648 065300 BBO C5
(2) 065300 102137 .WORD .$$$
649
650 065302 MOVE # 0,BREG ;TELL PDP-11 WE'RE DONE.
(3) 065302 000400 .WORD .$$$
651 065304 MOVE BREG,OUT1 <0>
(3) 065304 061220 .WORD .$$$
652
653 .:
654 .:TEST 6
655 .:
656 .: NPR TEST
657 .:
658 .: ENTERED IN LINE WHEN PDP-11 PUTS CODE 377 INTO
659 .: CRAM ADDR. 0.
660 .: DOES AN INPUT DATA XFER FROM LOC 1124
661 .: IN PDP-11 MEM TO LOC 1126
662 .: DOES NO DATA CHECKING RETURNS
663 .: CODE 0 TO CRAM ADDR. 0 WHEN DONE.
664 .:
665 065306 TST6: MOVE INP1 <0>,BREG ;WAIT FOR PDP-11
(3) 065306 120400 .WORD .$$$
666 065310 BZ A6
(2) 065310 101546 .WORD .$$$

```

```

667 065312 BR TST6
(2) 065312 100543 .WORD .$$$.
668
669 065314 A6: MOVE # 2,BREG ;SET TO DO NPR IN
(3) 065314 000402 .WORD .$$$.
670 065316 MOVE BREG,OUT0 <5> ;SET HIGH ADDR.
(3) 065316 062225 .WORD .$$$.
671 065320 MOVE # 124,BREG ;GET LOW ADDR. OF ADDR. 1124
(3) 065320 000524 .WORD .$$$.
672 065322 MOVE BREG,OUT0 <4> ;SET LOW ADDR.
(3) 065322 062224 .WORD .$$$.
673 065324 MOVE # 1,BREG ;SET TO NPR IN
(3) 065324 000401 .WORD .$$$.
674 065326 MOVE BREG,OUT1 <10> ;CAUSE NPR IN
(3) 065326 061230 .WORD .$$$.
675 065330 B6: MOVE INP1 <10>,BREG ;WAIT FOR NPR DONE.
(3) 065330 120600 .WORD .$$$.
676 065332 BBO B6
(2) 065332 102154 .WORD .$$$.
677
678 065334 MOVE INP0 <0>,BREG ;GET LOW BYTE DATA.
(3) 065334 020400 .WORD .$$$.
679 065336 MOVE BREG,OUT0 <2> ;OUTPUT NPR DATA REG.
(3) 065336 062222 .WORD .$$$.
680 065340 MOVE INP0 <1>,BREG ;GET HIGH BYTE.
(3) 065340 020420 .WORD .$$$.
681 065342 MOVE BREG,OUT0 <3> ;SAVE FOR OUTPUT.
(3) 065342 062223 .WORD .$$$.
682 065344 MOVE # 2,BREG ;GET OUTPUT ADDR. = 1126
(3) 065344 000402 .WORD .$$$.
683 065346 MOVE BREG,OUT0 <7>
(3) 065346 062227 .WORD .$$$.
684 065350 MOVE # 126,BREG ;LOW PART
(3) 065350 000526 .WORD .$$$.
685 065352 MOVE BREG,OUT0 <6>
(3) 065352 062226 .WORD .$$$.
686 065354 MOVE INP1 <11>,SPAD <4> ;/-STN-
(3) 065354 123224 .WORD .$$$.
687 065356 AND SPAD <4>,BREG,SPAD
(4) 065356 063264 .WORD .$$$.
688 065360 MOVE # 0,BREG
(3) 065360 000400 .WORD .$$$.
689 065362 OR SPAD <4>,BREG,BREG
(4) 065362 060704 .WORD .$$$.
690 065364 MOVE BREG,OUT1 <11> ;SET NPR OUT.
(3) 065364 061231 .WORD .$$$.
691 065366 MOVE # 21,BREG
(3) 065366 000421 .WORD .$$$.
692 065370 MOVE BREG,OUT1 <10>
(2) 065370 061230 .WORD .$$$.
693
694 065372 C6: MOVE INP1 <10>,BREG ;WAIT TILL DONE
(3) 065372 120600 .WORD .$$$.
695 065374 BBO C6
(2) 065374 102175 .WORD .$$$.
696

```

```

697 065376 MOVE # 0,BREG ;TELL PDP-11 WE'RE DONE.
(3) 065376 000400 .WORD .$$$
698 065400 MOVE BREG,OUT1 <0>
(3) 065400 061220 .WORD .$$$
699
700 ;TEST 7
701 :
702 INTERRUPT TEST
703 :
704 THIS TEST GENERATES TWO INTERRUPTS
705 :
706 THE 1ST INTERRUPT TO VECTOR XX0 WHEN
707 CRAM ADDR. 0 = 377 BY PDP-11 CONTROL.
708 THE 2ND INTERRUPT TO VECTOR XX4 WHEN
709 CRAM ADDR. 0 = 377 (1ST MODE = 0 AGAIN).
710 :
711 065402 TST7: MOVE INP1 <0>,BREG ;WAIT TILL PDP-11 READY!
(3) 065402 120400 .WORD .$$$
712 065404 BZ A7
(2) 065404 101604 .WORD .$$$
713 065406 BR TST7
(2) 065406 100601 .WORD .$$$
714
715 065410 A7: MOVE INP1 <11>,SPAD <4>
(3) 065410 123224 .WORD .$$$
716 065412 AND SPAD <4>,BREG,SPAD
(4) 065412 063264 .WORD .$$$
717 065414 MOVE # 200,BREG
(3) 065414 000600 .WORD .$$$
718 065416 OR SPAD <4>,BREG,BREG
(4) 065416 060704 .WORD .$$$
719 065420 MOVE BREG,OUT1 <11> ;SET INTR TO ADDR. XX0
(3) 065420 061231 .WORD .$$$
720 065422 B7: MOVE INP1 <11>,BREG ;WAIT TILL DONE
(3) 065422 120620 .WORD .$$$
721 065424 BB7 B7
(2) 065424 103611 .WORD .$$$
722
723 065426 MOVE # 0,BREG ;TELL PDP-11 WE INTERRUPTED
(3) 065426 000400 .WORD .$$$
724 065430 MOVE BREG,OUT1 <0> ;IN CASE WE DIDN'T
(3) 065430 061220 .WORD .$$$
725
726 065432 C7: MOVE INP1 <0>,BREG ;NOW WAIT FOR PDP-11 TO TELL
(3) 065432 120400 .WORD .$$$
727 065434 BZ D7 ;US TO INTR. TO VECTOR XX4
(2) 065434 101620 .WORD .$$$
728 065436 BR C7
(2) 065436 100615 .WORD .$$$
729
730 065440 D7: MOVE INP1 <11>,SPAD <4>
(3) 065440 123224 .WORD .$$$
731 065442 AND SPAD <4>,BREG,SPAD
(4) 065442 063264 .WORD .$$$
732 065444 MOVE # 300,BREG
(3) 065444 000700 .WORD .$$$

```

|     |        |        |          |                                       |
|-----|--------|--------|----------|---------------------------------------|
| 733 | 065446 |        | OR       | SPAD <4>,BREG,BREG                    |
| (4) | 065446 | 060704 | .WORD    | .\$\$\$.                              |
| 734 | 065450 |        | MOVE     | BREG,OUT1 <11> ;SET INTR TO ADDR. XX4 |
| (3) | 065450 | 061231 | .WORD    | .\$\$\$.                              |
| 735 |        |        |          |                                       |
| 736 | 065452 |        | E7: MOVE | INP1 <11>,BREG ;WAIT TILL DONE.       |
| (3) | 065452 | 120620 | .WORD    | .\$\$\$.                              |
| 737 | 065454 |        | BB7      | E7                                    |
| (2) | 065454 | 103625 | .WORD    | .\$\$\$.                              |
| 738 |        |        |          |                                       |
| 739 | 065456 |        | MOVE     | # 0,BREG ;TELL PDP-11 WE THOUGHT      |
| (3) | 065456 | 000400 | .WORD    | .\$\$\$.                              |
| 740 | 065460 |        | MOVE     | BREG,OUT1 <0> ;WE HAD INTERRUPTED!    |
| (3) | 065460 | 061220 | .WORD    | .\$\$\$.                              |
| 741 |        |        |          |                                       |
| 742 | 065462 |        | BR       | .                                     |
| (2) | 065462 | 100631 | .WORD    | .\$\$\$.                              |
| 743 |        |        |          |                                       |
| 744 |        |        |          |                                       |
| 745 | 065464 | 177777 | .WORD    | -1                                    |
| 746 |        | 000001 |          | .END                                  |



|                |                 |                   |                |                   |
|----------------|-----------------|-------------------|----------------|-------------------|
| ADDRES= 177777 | DRLPX1 065000 G | TST6 065306       | .BZ = 001400   | .SBREG= 060000    |
| A1 065020      | D3 065122       | TST7 065402       | .CO = 000400   | .SELA = 000200    |
| A2 065044      | D7 065440       | T3E 065144        | .DBR = 000400  | .SELB = 000220    |
| A3 065100      | E3 065134       | T4E 065206        | .DBRSH= 001400 | .SIMM = 000000    |
| A4 065156      | E7 065452       | \$\$\$SER= 000001 | .DEC = 000160  | .SINO = 020000    |
| A5 065220      | F3 065144       | . = 065466        | .DMEM = 002400 | .SIN1 = 120000    |
| A6 065314      | MARHLD= 000000  | .ADC = 000100     | .DNOP = 000000 | .SMEM = 040000    |
| A7 065410      | MARINC= 014000  | .ADD = 000000     | .DOUT0= 002000 | .SUB = 000340     |
| B3 065106      | MARLD = 010000  | .ADDWC= 000020    | .DOUT1= 001000 | .SUBWC= 000040    |
| B4 065172      | MARLDX= 004000  | .AND = 000260     | .DSPAD= 003000 | .SUB2C= 000360    |
| B5 065234      | PAGE0 = 000000  | .BB0 = 002000     | .DSPBR= 003400 | .SO = 020000      |
| B6 065330      | PAGE1 = 001000  | .BB1 = 002400     | .DO = 000400   | .XOR = 000320     |
| B7 065422      | PAGE2 = 002000  | .BB4 = 003000     | .FO = 000020   | ..\$\$\$ = 100631 |
| CLK = 000020   | PAGE3 = 003000  | .BB7 = 003400     | .INC = 000060  | ..LOC = 065462    |
| C3 065114      | TST1 065000     | .BC = 001000      | .LORN = 000240 | .2A = 000120      |
| C4 065202      | TST2 065024     | .BR = 000400      | .MINUS= 000360 | .2AWC = 000140    |
| C5 065276      | TST3 065052     | .BSBRG= 160000    | .MO = 004000   |                   |
| C6 065372      | TST4 065150     | .BSIMM= 100000    | .OR = 000300   |                   |
| C7 065432      | TST5 065212     | .BSMEM= 140000    | .PLUS = 000000 |                   |

|        |        |     |     |    |     |     |   |
|--------|--------|-----|-----|----|-----|-----|---|
| . ABS. | 065466 | 000 | OVR | RO | REL | LCL | I |
|        | 000000 | 001 | CON | RW | ABS | LCL | I |
| ABCODE | 004000 | 002 | CON | RW | REL | LCL | I |

ERRORS DETECTED: 0  
 DEFAULT GLOBALS GENERATED: 0

CRLPX1,CRLPX1=CRLPX1  
 RUN-TIME: 7 7 0 SECONDS  
 RUN-TIME RATIO: 25/15=1.6  
 CORE USED: 41K (81 PAGES)