

LPA11,DR11K

LPA/DR11-K I/O TEST
CRLPFBO

AH-E435B-MC

COPYRIGHT ' 77-80
FICHE 1 OF 1

JAN 1980

digital

MADE IN USA

IDENTIFICATION

SEG 0001

PRODUCT CODE: AC-E434B-MC
PRODUCT NAME: CRLEPBO LPA/DR11-K DIGITAL I/O TEST
DATE: JANUARY 1978
DATE REVISED: JULY 1979
MAINTAINED BY: DIAGNOSTIC GROUP

COPYRIGHT (C) 1977, 1978, 1979
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 IN EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

TABLE OF CONTENTS

1. ABSTRACT
2. EQUIPMENT REQUIREMENTS
3. LOADING PROCEDURE
4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESSES
5. OPERATING PROCEDURE
6. ERRORS
7. RESTRICTIONS
8. MISCELLANEOUS
 - 8.1 EXECUTION TIMES
 - 8.2 DEVICE ADDRESSES
9. PROGRAM DESCRIPTION
 - 9.1 LOGIC TEST
 - 9.2 CONTROL LINE LOOP
10. LPA11 (SYSTEM) DIAGNOSTIC SUMMARY

ABSTRACT

THIS PROGRAM IS A LOGIC TEST OF THE DR11-K DIGITAL INPUT OUTPUT CONTROL OPTION. MOST OPTION FUNCTIONS CAN BE TESTED.

DUE TO THE FLEXIBILITY OF THE OPTION, THE OPERATOR WILL BE REQUIRED TO SUPPLY OPTION CHARACTERISTICS. THE PROGRAM WILL HANDLE ALL CONFIGURATIONS OF INPUT INTERRUPT SWITCHES AND INPUT DATA LATCHING JUMPERS. FOR SYSTEMS WITH CONSECUTIVE MULTIPLE DR11-K'S, THESE CONFIGURATIONS MUST BE THE SAME. THE FOLLOWING JUMPERS MUST BE INSERTED TO EXECUTE THE LOGIC TEST: W21A, W22A AND W23A.

THIS PROGRAM WILL TEST SEQUENTIAL DR11-K'S STARTING AT THE BUS ADDRESS AND VECTOR IN LOCATIONS '\$BASE' AND '\$VECT1'. FOR NORMAL FACTORY CONFIG., ALL QUESTIONS SHOULD BE ANSWERED WITH A VALUE OF 0.

THIS PROGRAM IS A MODIFIED VERSION OF 'MD-11-DZDRG-E' IT WAS MODIFIED TO ENABLE THE OPERATOR TO CHECK OUT THE DR11-K OPTION WHEN IT IS ON THE LPA11-KX I/O BUS. NO RE-CABLING IS NEEDED. SOME TESTS DONE IN THE ORIGINAL DIAGNOSTIC SUCH AS ARBITRATION TEST, WERE DELETED AS THEY COULD NOT BE CHECKED. IF THIS DIAGNOSTIC DOESN'T FIND A SUSPECTED PROBLEM, YOU MAY HAVE TO RUN 'MD-DZDRG-E' YOU SHOULD RUN 'MD-11-DRLPA' BEFORE RUNNING THIS DIAGNOSTIC. PLEASE READ SECTION 10.

** AFTER INITIAL LOADING OF THE PROGRAM, THE LOGIC TEST MUST BE RUN. **

2. EQUIPMENT REQUIREMENTS

PDP-11 FAMILY COMPUTER WITH CONSOLE I/O TERMINAL AND 16K OF MEMORY
 DR11-K OPTION INSTALLED IN THE LPA-11
 BC08R-1 ONE FOOT OUTPUT TO INPUT WRAPAROUND CABLE
 LPA11-KX OPTION

3. LOADING PROCEDURE

THE PROCEDURE FOR LOADING A BINARY FILE SHOULD BE FOLLOWED.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS (LOGIC TEST)

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

CONTROL :

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G <^G>; THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: SWR=XXXXXX NEW- (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE "'NEW='" HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
 - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED) IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
 - B) IF A CONTROL U <^U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

WILL BE USED AS THE SOFTWARE DISPLAY REGISTER.

SW 15 = 1	100000	HALT ON ERROR
SW 14 = 1	040000	LOOP ON CURRENT SUB-TEST
SW 13 = 1	020000	INHIBIT ERROR TYPINGS
SW 12 = 1	010000	LOOP ON CURRENTLY SELECTED DR11-K
SW 11 = 1	004000	INHIBIT SUB-TEST INTERATIONS
SW 10 = 1	002000	NO OUTPUT TO INPUT WRAPAROUND CABLE
SW 09 = 1	001000	LOOP ON ERROR
SW 08 = 1	000400	LOOP ON TEST IN SWR <7:C>

4.2 STARTING ADDRESSES

- 200 IS THE STARTING ADDRESS OF THE LOGIC TEST.
- 204 IS THE RESTART ADDRESS OF THE LOGIC TEST.
- 210 IS THE STARTING ADDRESS OF THE CONTROL LINE LOOP.
- 214 IS THE STARTING ADDRESS OF THE USER LINK LOOP.

5. OPERATING PROCEDURE

THE FOLLOWING JUMPERS MUST BE INSTALLED TO EXECUTE THE LOGIC TEST: W21A, W22A AND W23A
 IF THE CUSTOMER HAS SELECTED THE 'B' SECTION OF THESE JUMPERS IT MUST BE RETURNED TO THE 'FACTORY' POSITION BEFORE RUNNING THE LOGIC TEST. ** WORST CASE WILL ONLY BE CHANGING THREE JUMPERS. **
 THE OPERATOR MUST INSERT THE CORRECT INFORMATION IN THE SWITCH REGISTER WHEN REQUESTED BY THE PROGRAM OR AN ERROR WILL OCCUR. WITH THE INPUT INTERRUPT SWITCHES AND DATA LATCHING JUMPERS IN THE FACTORY POSITION, ALL SWITCH REGISTER BITS SHOULD BE RESET. ONCE STARTED THE TEST WILL RUN IN IT'S NORMAL MANNER WITHOUT OPERATOR INTERVENTION OR SWITCH SELECTION.
 *** NOTE: OPERATOR MUST INSERT INFORMATION WHEN REQUESTED BY PROGRAM. THE MACHINE WILL TYPE: NEW = AFTER NEW - THE INFORMATION IS INSERTED. REFER TO SECTION 4.1 2) ***

6. ERRORS

THIS PROGRAM USES THE DIAGNOSTIC 'SYSMAC' PACKAGE FOR ERROR REPORTING AND TYPEOUT. REFER TO THE 'ERROR POINTER TABLE' FOR TYPE AND DESCRIPTION OF ERRORS.

BYTE	\$STNM: (LOC. 1102)	CURRENT TEST NUMBER
BYTE	\$ITEMB: (LOC. 1114)	ITEM #N ERROR TABLE INDEX
WORD	\$ERRPC: (LOC. 1116)	ERRORING P.C.
WORD	\$PASS: (LOC. 1176)	CURRENT PASS COUNT

7. RESTRICTIONS

-
1. IF SEQUENTIAL DR11-K'S, ALL DR11-K'S MUST BE IN THE SAME INTERRUPT SWITCHES AND DATA PATH JUMPER CONFIGERATION.
 2. THE FOLLOWING JUMPERS MUST BE IN THE 'FACTORY' POSITION:
W21A, W22A AND W23A
 3. THE OPERATOR MUST SUPPLY THE CORRECT INTERRUPT SWITCHES AND DATA PATH JUMPER AND SWITCH CONFIGURATION INFORMATION TO THE INITIALIZATION QUESTIONS OR AN ERROR WILL OCCUR.
 4. FOR MULTIPLE GROUPS OF CONSECUTIVE DR11-K'S:
THIS DIAGNOSTIC MUST BE RUN FOR EACH GROUP.
 5. AFTER INITIAL LOADING OF THE PROGRAM, THE LOGIC MUST BE RUN FIRST

8. MISCELANEOUS

8.1 EXECUTION TIME

THE LOGIC TEST WILL TAKE APPROXIMATELY 90 SECONDS FOR COMPLETION ON A PDP11/05 TYPE AND WILL TYPE 'END PASS NNNN.'. THE CONTROL LINE LOOP WILL NEVER EXIT.

8.2 DEVICE ADDRESS PROGRAM LOCATIONS

'SBASE' (LOC. 1244) CONTAINS THE DR11-K BASE DEVICE ADDRESS <767770>
'SVECT1' (LOC. 1240) THE LOW 9 BITS CONTAIN THE DR11-K BASE INTERRUPT VECTOR <300>
'SVECT1' (LOC. 1240) THE HIGH 3 BITS CONTAIN THE DR11-K BR LEVEL #4 <200>

*NOTE: IF THESE LOCATIONS ARE CHANGED, THE OPERATOR MUST START THE TEST AGAIN AT LOC. 200. THE PROGRAM WILL USE THE BASE ADDRESS AND VECTOR AND UPDATE THE ACTUAL PROGRAM VALUES.

8.3 USER LINK TO I/O DEVICE

A SPECIAL USER LINK HAS BEEN PROVIDED IN ORDER FOR THE OPERATOR TO EXAMINE OR MODIFY LOCATIONS ON THE LPA11-KX I/O BUS. (NOTE: THIS CANNOT BE DONE DIRECTLY.)

PROCEDURE:

- 1) START THE PROCESSOR AT LOCATION 214
- 2) THE DIALOG TO EXAMINE A LOCATION IS AS FOLLOWS:

```

E OR D      'E'
DEVICE ADDRS= 'OCTAL ADDRS'
XXXXXX
  
```

WHERE XXXXXX IS THE CONTENTS OF THE SPECIFIED LOC.

- 3) THE DIALOG TO MODIFY A LOCATION IS AS FOLLOWS:

```

E OR D      'D'
DATA-      'DATA TO BE DEPOSITED'
  
```

- 4) THE PROGRAM WILL STAY IN THIS LOOP UNTIL THE OPERATOR IS FINISHED. AT THIS TIME THE PROCESSOR SHOULD BE HALTED.

NOTE: THE OPERATORS RESPONSE IS ENCLOSED IN QUOTES.

9. PROGRAM DESCRIPTION THE LOGIC TEST MUST BE RUN FIRST AFTER INITIAL PROGRAM LOADING

9.1 LOGIC TEST <SA 200 AND 204>

THE LOGIC TEST IS A TEST OF THE CONTROL AND INPUT/OUTPUT REGISTERS. ALL JUMPERS AND SWITCHES COMBINATIONS EXCEPT.

W21A, W22A AND W23A CAN BE DIAGNOSED.

THE PROGRAM CHECKS THAT THE DR11-K 'RESET' WILL WORK CORRECTLY.

9.2 CONTROL LINE LOOP <SA 210>

THIS TEST LOOP PROVIDES THE OPERATOR WITH A SCOPE LOOP FOR CHECKING W21, W22 AND W23 IN THE 'B' POSITION.

9.3 USER LINK LOOP <SA 214>

THIS LOOP ENABLES THE OPERATOR TO EXAMINE OR MODIFY LOCATIONS ON THE LPA-11K I/O BUS (REFER TO 8.3).

10. 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-DRLPA. WHEN THE USER RUNS DRLPA HE CAN GENERALLY TELL WHICH OPTION DIAGNOSTIC (LEVEL 3) TO RUN NEXT. M8254 AND BMC-11 ERRORS MAY 'LOOK' ALIKE AND DRLPA MAY NOT BE ABLE TO DISTINGUISH BETWEEN THEM. ARBITRATION ERRORS WILL NOT BE DETECTED BY THIS DIAGNOSTIC.

LEVEL THREE DIAGNOSTICS AID IN DETERMINING IF THE ERROR WAS IN FACT ON THE OPTION THE DRLPA SPECIFIED. THE USER MAY 'LOGP' ON THE ERROR. WITHIN LEVEL THREE, 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:

<u>OPTION</u>	<u>GROUP</u>	<u>DIAG. #</u>	<u>DIAG. TITLE</u>
LPA11-KX	LEVEL 2	MD-11-DRLPA	LPA11-K SYSTEM DIAG.
M8254	'B'	MD-11-DRLPN	M8254 (IPBM) DIAG.
AA11-K	A	MD-11-DRLPB	AA11-K DIAG.
	B	MD-11-DZAAC	AA11-K DIAG.
AR11	A	MD-11-DRLPC	LPA/AR11 DIAG. #1
	A	MD-11-DRLPD	LPA/AR11 DIAG. #2
	A	MD-11-DRLPE	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-DRLPH	LPA/LPS11 DIAG. #1
	A	MD-11-DRLPI	LPA/LPS11 DIAG. #2
	A	MD-11-DRLPJ	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-DZLPL	LPA/DMC BASIC MICRO-CPU R/W TEST
	B	MD-11-DZLPM	LPA/DMC JMP+ROM READ TEST

PRODUCT CODE: MAINDEC-11-DZDRG-B
PRODUCT NAME: DR11-K DIGITAL I/O TEST
DATE: APRIL 1976
MAINTAINER: DIANOSTIC GROUP

PRODUCT CODE: MAINDEC-11-DRLPF-A
PRODUCT NAME: LPA/DR11-K DIGITAL I/O TEST
DATE: JANUARY 1978
MAINTAINER: DIAGNOSTIC GROUP

REASON FOR DEVELOPMENT:

- 1) TO ENABLE THE OPERATOR TO CHECK OUT THE DR11-K OPTION WHEN IT IS ON THE LPA11-KX I/O BUS.

CHANGES MADE:

- 1) TOOK OUT CERTAIN TESTS FROM ORIGINAL DIAGNOSTIC (I.E. INTERRUPTS, TIME DEPENDENT CODE).
- 2) REPLACED DIRECT LINKS TO DEVICE WITH MACRO CALLS TO THE KMC-11 MICRO CODE. KMC-11 MICRO CODE (FILE:DRLPX2) HANDLES DIRECT COMMUNICATIONS WITH THE DEVICE.

FILE: DRLPA.MAC
CONTAINS MACRO LINKS BETWEEN PDP-11 CODE AND KMC-11 MICRO CODE. FILE: DRLPX2 NEEDS TO BE ASSEMBLED WITH DRLPF (SEE .CTL FILE).

FILE: DRLPX2
MICRO CODE FILE THAT GETS LOADED INTO THE KMC-11 VIA ROUTINES IN DRLPA.MAC.

DRLPX2.P11 IS ASSEMBLED WITH MACY11 (ONLY) AS ANY OTHER .P11 FILE. THE RESULTS OF ITS ASSEMBLY IS A .OBJ MODULE AS WAS THE RESULT OF THE ASSEMBLY OF THE DIAGNOSTIC .P11 FILE. BOTH .OBJ FILES GET LINKED WITH LNKX11 (ONLY).

FILE: DRLPF.CTL
THIS FILE EXPLAINS SEQUENCE OF ASSEMBLES AND LINKS. IT IS IN TOPS-20 FORMAT.

PRODUCT CODE: AC-E434B-MC
 DIAGNOSTIC CODE: MD-11-CRLPF-B
 PRODUCT NAME: CRLPF8 LPA/DR11-K DIGITAL I/O TEST
 DATE: JAN. 1978
 DATE REVISED: JULY 1979
 MAINTAINER: DIAGNOSTIC GROUP

////////////////////////////////////

PROBLEMS

1. IF THE 'WRAP-AROUND' CABLE IS CONNECTED, THE PROGRAM GETS A 'BUS TRAP' IN 'TST34'. THE PROGRAM TRYS TO ADDRESS THE DR-11K ON THE 11 BUS AND NOT THE LPA BUS
2. USER LINK SECTION HAS A NON STANDARD STARTING ADDRESS
3. SECTIONS OF THE ORIGINAL PROGRAM STILL REMAIN EVEN WHEN THE LPA-11 CANNOT SUPPORT THESE MDES

SOLUTIONS

1. 4 8. LOCATION PATCH SOLVES THE PROBLEM
2. ASSIGN 214 AS THE STARTING ADDRESS FOR THE 'USER LINK' LOOP.
3. EDIT OUT THOSE SECTIONS THAT CAN NEVER BE USED.

////////////////////////////////////

2958	BASIC DEFINITIONS
2959	OPERATIONAL SWITCH SETTINGS
2961	TRAP CATCHER
(1)	STARTING ADDRESS(ES)
2966	ACT11 HOOKS
2968	APT PARAMETER BLOCK
2969	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
3076	INITIALIZE THE COMMON TAGS
3115	TYPE PROGRAM NAME
(2)	GET VALUE FOR SOFTWARE SWITCH REGISTER
3250	T1 TEST FOR NO BUS ERRORS
3258	T2 TEST THAT OUTPUT REG. CAN HOLD #-1
3266	T3 TEST THAT RESET CLEARS OUTPUT REG.
3276	T4 TEST THAT OUTPUT REG. CAN HOLD #52525
3285	T5 TEST THAT OUTPUT REG. CAN HOLD #125252
3294	T6 TEST THAT OUTPUT REG. CAN HOLD A COUNT PATTERN
3304	T7 FLOAT A 1 ACROSS THE OUTPUT REGISTER
3319	T10 FLOAT A 0 ACROSS THE OUTPUT REGISTER
3336	T11 TEST FOR SLOW OUTPUT GATES WITH #125252
3348	T12 TEST FOR SLOW OUTPUT GATES WITH #52525
3361	T13 TEST OUTPUT DATA ACCEPT FLAG
3373	T14 TEST OUTPUT INTERRUPT ENABLE
3383	T15 TEST INPUT DATA READY FLAG
3391	T16 TEST INPUT INTERRUPT ENABLE
3411	T17 TEST EXTERNAL TRANSFERS - CABLE MUST BE CONNECTED
3426	T20 TEST INPUT WITH #-1
3438	T21 TEST INPUT WITH #52525
3450	T22 TEST INPUT WITH #125252
3461	T23 TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
3496	T24 FLOAT A 1 ACROSS NON-LATCHING INPUT BITS
3532	T25 FLOAT A 1 ACROSS LATCHING INPUT BITS
3559	T26 FLOAT A 0 ACROSS LATCHING INPUT BITS
3586	T27 TEST FOR SLOW INPUT GATES WITH #125252
3614	T30 TEST FOR SLOW INPUT GATES WITH #52525
3640	T31 TEST THAT RESET CLEARS INPUT REGISTER BITS
3653	T32 TEST THAT WHEN OUTPUTTING THE INPUT DATA READY FLAG SETS
3666	T33 TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
3680	T34 TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
3730	T35 TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
3767	T36 DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
3783	END OF PASS ROUTINE
3784	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3837	SCOPE HANDLER ROUTINE
3838	ERROR HANDLER ROUTINE
3840	ERROR MESSAGE TYPEOUT ROUTINE
3842	BINARY TO OCTAL (ASCII) AND TYPE
3844	POWER DOWN AND UP ROUTINES
3848	TYPE ROUTINE
3849	READ AN OCTAL NUMBER FROM THE TTY
3850	TTY INPUT ROUTINE
3853	APT COMMUNICATIONS ROUTINE
3854	TRAP DECODER
(3)	TRAP TABLE

1
2
3
4
5
6
7
8
9
10
11
12
13
52
53
54
140
156
169
182
183
415
416
457
509
608
650
697
746

.REM [

LPA.MAC

WELCOME, THIS DIAGNOSTIC IS ONE IN A SERIES OF DIAGNOSTIC
DESIGNED IN ORDER TO AID YOU IN TESTING THE LPA-11XX OPTION.
I 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


```
(1) 000004 R4= 24 ;;GENERAL REGISTER
(1) 000005 R5= 25 ;;GENERAL REGISTER
(1) 000006 R6= 26 ;;GENERAL REGISTER
(1) 000007 R7= 27 ;;GENERAL REGISTER
(1) 000006 SP= 26 ;;STACK POINTER
(1) 000007 PC= 27 ;;PROGRAM COUNTER
```

```
(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) ;;*'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) ;;*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
2959          .SBTTL OPERATIONAL SWITCH SETTINGS
(1)          ;*
(1)          ;*          SWITCH          USE
(1)          ;*          -----
(1)          ;*          15          HALT ON ERROR
(1)          ;*          14          LOOP ON TEST
(1)          ;*          13          INHIBIT ERROR TYPEOUTS
(1)          ;*          12          LOOP ON CURRENTLY SELECTED DR11-K
(1)          ;*          11          INHIBIT ITERATIONS
(1)          ;*          10          OUTPUT TO INPUT WRAPAROUND CABLE NOT CONNECTED
(1)          ;*          9          LOOP ON ERROR
(1)          ;*          8          LOOP ON TEST IN SWR<7:0>
2960          .SBTTL TRAP CATCHER
2961          ;*
(1)          ;*          =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
(1)          .SBTTL STARTING ADDRESS(ES)
(1)          000200          000137          001546          JMP @#BEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
    
```

2963 000204 000137 001556
 2964 000210 000137 013572
 2965 000214 000137 022552

JMP @#BEGIN1 ;JUMP TO THE RESTART ADDRESS
 JMP @#EXITST ;JUMP TO THE CONTROL LINES LOOP
 JMP @#SUITE ;JUMP TO THE USER LINK LOOP

2966

.SBTTL ACT11 HOOKS

(1)

(2)

;HOOKS REQUIRED BY ACT11

(1)

\$SVPC-. ;SAVE PC

(1) 000220

.=46

(1) 000046

\$ENDAD ;:1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .\$EOP

(1) 000046 013312

.-52

(1) 000052

.WORD 0 ;:2)SET LOC.52 TO ZERO

(1) 000052 000000

.\$SVPC ;: RESTORE PC

(1) 000220

.-1000

2967 001000

.SBTTL APT PARAMETER BLOCK

2968

(1)

(2)

;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT

(1)

(2)

.\$X= ;:SAVE CURRENT LOCATION

(1) 001000

.-24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM

(1) 000024

200 ;:FOR APT START UP

(1) 000200

.-44 ;:POINT TO APT INDIRECT ADDRESS PNTR.

(1) 000044

\$APTHDR ;:POINT TO APT HEADER BLOCK

(1) 000044 001000

.\$X ;:RESET LOCATION COUNTER

(1) 001000

(2)

;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC

(1)

;INTERFACE SPEC.

(1)

\$APTHD:

(1) 001000

\$HIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.

(1) 001000 000000

\$MBADR: .WORD \$MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)

(1) 001002 001170

\$TSTM: .WORD 30 ;:RUN TIME OF LONGEST TEST

(1) 001004 000030

\$PASTM: .WORD 10 ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)

(1) 001006 000010

\$UNITM: .WORD 30 ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT

(1) 001010 000030

.WORD \$ETEND-\$MAIL/2 ;:LENGTH MAILBOX-ETABLE (WORDS)

(1) 001012 000031

2969
(1)
(2)
(1)
(1)
(1)
(1)
(1) 001100
(1) 001100 000000
(1) 001102 000
(1) 001103 000
(1) 001104 000000
(1) 001106 000000
(1) 001110 000000
(1) 001112 000000
(1) 001114 000
(1) 001115 001
(1) 001116 000000
(1) 001120 000000
(1) 001122 000000
(1) 001124 000000
(1) 001126 000000
(1) 001130 000000
(1) 001132 000000
(1) 001134 000
(1) 001135 000
(1) 001136 000000
(1) 001140 177570
(1) 001142 177570
(1) 001144 177560
(1) 001146 177562
(1) 001150 177564
(1) 001152 177566
(1) 001154 000
(1) 001155 002
(1) 001156 012
(1) 001157 000
(1) 001160 000000
(1) 001162 000000
(1) 001164 077
(1) 001165 015
(1) 001166 000012
(2)
(2)
(2)
(3)
(2)
(2) 001170
(2) 001170 000000
(2) 001172 000000
(2) 001174 000000
(2) 001176 000000
(2) 001200 000000
(2) 001202 000000
(2) 001204 000000
(2) 001206 000000

.SBTTL COMMON TAGS

: *THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
: *USED IN THE PROGRAM.
.=1100
\$CMTAG: .WORD 0 :: START OF COMMON TAGS
\$TSTNM: .BYTE 0 :: CONTAINS THE TEST NUMBER
\$ERFLG: .BYTE 0 :: CONTAINS ERROR FLAG
\$ICNT: .WORD 0 :: CONTAINS SUBTEST ITERATION COUNT
\$LPADR: .WORD 0 :: CONTAINS SCOPE LOOP ADDRESS
\$LPERR: .WORD 0 :: CONTAINS SCOPE RETURN FOR ERRORS
\$ERTTL: .WORD 0 :: CONTAINS TOTAL ERRORS DETECTED
\$ITEMB: .BYTE 0 :: CONTAINS ITEM CONTROL BYTE
\$ERMAX: .BYTE 1 :: CONTAINS MAX. ERRORS PER TEST
\$ERRPC: .WORD 0 :: CONTAINS PC OF LAST ERROR INSTRUCTION
\$GDADR: .WORD 0 :: CONTAINS ADDRESS OF 'GOOD' DATA
\$BDADR: .WORD 0 :: CONTAINS ADDRESS OF 'BAD' DATA
\$GDDAT: .WORD 0 :: CONTAINS 'GOOD' DATA
\$BDDAT: .WORD 0 :: CONTAINS 'BAD' DATA
:: RESERVED--NOT TO BE USED
\$AUTOB: .BYTE 0 :: AUTOMATIC MODE INDICATOR
\$INTAG: .BYTE 0 :: INTERRUPT MODE INDICATOR
\$SWR: .WORD DSWR :: ADDRESS OF SWITCH REGISTER
\$DISPLAY: .WORD DDISP :: ADDRESS OF DISPLAY REGISTER
\$TKS: 177560 :: TTY KBD STATUS
\$TKB: 177562 :: TTY KBD BUFFER
\$TPS: 177564 :: TTY PRINTER STATUS REG. ADDRESS
\$TPB: 177566 :: TTY PRINTER BUFFER REG. ADDRESS
\$NULL: .BYTE 0 :: CONTAINS NULL CHARACTER FOR FILLS
\$FILLS: .BYTE 2 :: CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLC: .BYTE 12 :: INSERT FILL CHARS. AFTER A 'LINE FEED'
\$TPFLG: .BYTE 0 :: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
\$TIMES: 0 :: MAX. NUMBER OF ITERATIONS
\$ESCAPE: 0 :: ESCAPE ON ERROR ADDRESS
\$QUES: .ASCII /?/ :: QUESTION MARK
\$CRLF: .ASCII <15> :: CARRIAGE RETURN
\$LF: .ASCII <12> :: LINE FEED

.SBTTL APT MAILBOX-ETABLE

.EVEN
\$MAIL: :: APT MAILBOX
\$MSGTY: .WORD AMSGTY :: MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL :: FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN :: TEST NUMBER
\$PASS: .WORD APASS :: PASS COUNT
\$DEVCT: .WORD ADEVCT :: DEVICE COUNT
\$UNIT: .WORD AUNIT :: I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD :: MESSAGE ADDRESS
\$MSGLG: .WORD AMSGLG :: MESSAGE LENGTH

```
(2) 001210          $ETABLE:          ;; APT ENVIRONMENT TABLE
(2) 001210          $ENV: .BYTE   AENV          ;; ENVIRONMENT BYTE
(2) 001211          $ENVM: .BYTE   AENVM         ;; ENVIRONMENT MODE BITS
(2) 001212          $$WREG: .WORD   ASWREG        ;; APT SWITCH REGISTER
(2) 001214          $USWR: .WORD   AUSWR         ;; USER SWITCHES
(2) 001216          $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) 001220          $MAMS1: .BYTE   AMAMS1        ;; HIGH ADDRESS,M.S. BYTE
(2) 001221          $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) 001222          $MADR1: .WORD   AMADR1        ;; HIGH ADDRESS,BLK#1
(2)                : *                          MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2) 001224          $MAMS2: .BYTE   AMAMS2        ;; HIGH ADDRESS,M.S. BYTE
(2) 001225          $MTYP2: .BYTE   AMTYP2        ;; MEM. TYPE,BLK#2
(2) 001226          $MADR2: .WORD   AMADR2        ;; MEM.LAST ADDRESS,BLK#2
(2) 001230          $MAMS3: .BYTE   AMAMS3        ;; HIGH ADDRESS,M.S.BYTE
(2) 001231          $MTYP3: .BYTE   AMTYP3        ;; MEM. TYPE,BLK#3
(2) 001232          $MADR3: .WORD   AMADR3        ;; MEM.LAST ADDRESS,BLK#3
(2) 001234          $MAMS4: .BYTE   AMAMS4        ;; HIGH ADDRESS,M.S.BYTE
(2) 001235          $MTYP4: .BYTE   AMTYP4        ;; MEM. TYPE,BLK#4
(2) 001236          $MADR4: .WORD   AMADR4        ;; MEM.LAST ADDRESS,BLK#4
(2) 001240          $VECT1: .WORD   AVECT1        ;; INTERRUPT VECTOR#1,BUS PRIORITY#1
(2) 001242          $VECT2: .WORD   AVECT2        ;; INTERRUPT VECTOR#2BUS PRIORITY#2
(2) 001244          $BASE: .WORD   ABASE         ;; BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 001246          $DEVN: .WORD   ADEVN         ;; DEVICE MAP
(2) 001250          $CDW1: .WORD   ACDW1         ;; CONTROLLER DESCRIPTION WORD#1
(2) 001252          $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 FORM

(1) $ERRTB:
2970
2971 ;ITEM 1
2972 001252 014512 EM1 ;STATUS REGISTER IN ERROR
2973 001254 015132 DH1 ;ERRPC DRADD STATUS EXPECTED
2974 001256 015422 DT1 ;$ERRPC DRADD $BDDAT $GDDAT
2975 001260 015464 DF0
2976
2977 ;ITEM 2
2978 001262 014543 EM2 ;INPUT REGISTER IN ERROR
2979 001264 015202 DH2 ;ERRPC DRADD INPUT EXPECTED
2980 001266 015422 DT1 ;$ERRPC DRADD $BDDAT $GDDAT
2981 001270 015464 DF0
2982
2983 ;ITEM 3
2984 001272 014573 EM3 ;OUTPUT REGISTER IN ERROR
2985 001274 015250 DH3 ;ERRPC DRADD OUTPUT EXPECTED
2986 001276 015422 DT1 ;$ERRPC DRADD $BDDAT $GDDAT
2987 001300 015464 DF0
2988
2989 ;ITEM 4
2990 001302 014624 EM4 ;INPUT FAILED TO INTERRUPT
2991 001304 015316 DH4 ;ERRPC DRADD
2992 001306 015436 DT4 ;$ERRPC DRADD
2993 001310 015464 DF0
2994
2995 ;ITEM 5
2996 001312 014656 EM5 ;OUTPUT FAILED TO INTERRUPT
2997 001314 015316 DH4 ;ERRPC DRADD
2998 001316 015436 DT4 ;$ERRPC DRADD
2999 001320 015464 DF0
3000
3001 ;ITEM 6
3002 001322 014711 EM6 ;UNEXPECTED INTERRUPT
3003 001324 015316 DH4 ;ERRPC DRADD
3004 001326 015436 DT4 ;$ERRPC DRADD
3005 001330 015464 DF0
3006
3008 ;ITEM 7
3009 001332 014736 EM7 ;OPERATOR INTERVENTION ERROR
3010 001334 015316 DH4 ;ERRPC DRADD
3011 001336 015436 DT4 ;$ERRPC DRADD
```



```

3012 001340 015464          DFO
3013
3014          ;ITEM 10
3015 001342 014772          EM10          ;INTERRUPT INPUT BIT FAILED TO SET INPUT READY
3016 001344 015343          DH10          ;ERRPC DRADD STATUS EXPECTED INPUT BIT
3017 001346 015446          DT10          ;$ERRPC DRADD $BDDAT $GDDAT BRLEV3
3018 001350 015464          DFO
3019
3020          ;ITEM 11
3021 001352 015055          EM11          ;NON-INTERRUPTING INPUT BIT SET INPUT READY
3022 001354 015343          DH10          ;ERRPC DRADD STATUS EXPECTED INPUT BIT
3023 001356 015446          DT10          ;$ERRPC DRADD $BDDAT $GDDAT BRLEV3
3024 001360 015464          DFO
3025
(1)          ;ADDRESS OF KMC-11 OF LPA-11 THE ADDR FOR KMADO MAY BE
(1)          ; CHANGED BY THE USER TO REFLECT
( )          ; A DIFFERENT KMC-11 ADDR. THE
(1)          ; REST OF THE ADDRESSES WILL
(1)          ; BE CHANGED BY THE PROGRAM.
(1)          ;
(1)          ;
(1)          ;
(1)          ;
(1)          ;
(1)          ;
(1)          ;
(1)          ;
(1) 001362          LPCI:
(1) 001362 170460          KMADO: .WORD 170460          ;BASE KMC ADDR. MAY BE PATCHED BY USER.
(1)          ;
(1) 001364          LPMR:
(1) 001364 170461          KMAD1: .WORD 170460+1          ;>DO NOT <;KMC-CSR ADDR
(1) 001366          LPCO:
(1) 001366 170462          KMAD2: .WORD 170460+2          ;>PATCH <;
(1) 001370          LPSO:
(1) 001370 170463          KMAD3: .WORD 170460+3          ;>THIS AREA <
(1) 001372          LPADL:
(1) 001372 170464          KMAD4: .WORD 170460+4          ;
(1) 001374          LPADH:
(1) 001374 170465          KMAD5: .WORD 170460+5          ;>DO NOT <
(1) 001376          LPMS1:
(1) 001376 170466          KMAD6: .WORD 170460+6          ;>PATCH <
(1) 001400          LPMS2:
(1) 001400 170467          KMAD7: .WORD 170460+7          ;>THIS AREA <
(1)          ;
(1) 001402 000300          VECTOR: .WORD AVECT18777          ;BASE VECTOR OF KMC
(1) 001404 000304          VECTPS: .WORD 4+AVECT18777          ;VECTR ADDR.+2
(1)          ;
(1) 001406 000004          VERSN: .WORD 4          ;CURRENT VERSION NUMBER OF MICROCODE.
(1)          ;
(1) 001410 000000          .DVLS: .WORD 0          ;/DEVICE LIST OF I/O ADDR. DEFINED
(1) 001412 000020          .BLKW 16.          ;/BY INIT.
(1)          ;
3026          ;
3027 001452 167770          BASEBA: 167770          ;STARTING BASE BUS ADDRESS
3028 001454 000300          BASEIV: 300          ;STARTING BASE INTERRUPT ADDRESS
3029 001456 000200          BASEBR: 200
3030 001460 000240          CPU: 240          ;1 MS. CPU DELAY FACTOR 240 FOR 11/05
3031          ; 620 FOR 11/40
3032 001462 000000          NMBEXT: 0          ;ADDITIONAL DR-11-K
3033 001464 000000          NBEXT: 0
3034

```

3035	001466	167770	DRADD:	167770	:CURRENT DR11-K STARTING ADDRESS
3036	001470	000300	DRIV:	300	:CURRENT DR11-K STARTING INTERRUPT VECTOR
3037					
3038					
3039	001472	167770	LOC1:	167770	:LOC BOX #1 ADDR
3040	001474	167760	LOC2:	167760	:LOC BOX #2 ADDR
3041	001476	167750	LOC3:	167750	:LOC BOX #3 ADDR
3042	001500	167770	GRSTAT:	167770	:DR STATUS
3043	001502	167772	GRDAI:	167772	:INPUT REG.
3044	001504	167774	GRDIO:	167774	:OUTPUT REG.
3045	001506	167775	GRBHIO:	167775	:OUTPUT REG HIGH BYTE
3046	001510	000000	NOTLCH:	0	
3047	001512	000000	INTBIT:	0	
3048	001514	000300	GRIVA:	300	
3049	001516	000302	GRIVSA:	302	
3050	001520	000304	GRIVB:	304	
3051	001522	000306	GRIVSB:	306	
3052	001524	000200	DIOBRL:	200	
3053	001526	000000	BRLEV1:	0	
3054	001530	000000	BRLEV2:	0	
3055	001532	000000	BRLEV3:	0	
3056	001534	000000	ODDJMP:	0	
3057	001536	000000	MINSIN:	0	
3058	001540	000000	TRANST:	0	
3059	001542	000000	JUMPER:	0	:1 IF RUNNING H322 JUMPER CONFIG ;2 IF COULTER JUMPER CONFIG
3060	001544	000000	\$TMDAT:	0	

```
3061
3068
3069
3070
3071 001546 005000
3072 001550 005037 001542
3073 001554 000402
3074 001556 012700 177777
3075 001562 000240
3076
(1)
(1) 001564 012706 001100
(1) 001570 005026
(1) 001572 022706 001140
(1) 001576 001374
(1) 001600 012706 001100
(1) 001604 012737 015504 000020
(1) 001612 012737 000340 000022
(1) 001620 012737 015764 000030
(1) 001626 012737 000340 000032
(1) 001634 012737 020514 000034
(1) 001642 012737 000340 000036
(1) 001650 012737 016540 000024
(1) 001656 012737 000340 000026
(1) 001664 013737 013260 013252
(1) 001672 005037 001160
(1) 001676 005037 001162
(1) 001702 112737 000001 001115
(1) 001710 012737 001710 001106
```

```
*****
: DIGITAL I-O LOGIC TEST
*****
BEGIN: CLR R0 ;CLEAR R0
        CLR JUMPER ;INDICATE NORMAL FACTORY BUILD
        BR RBEG
BEGIN1: MOV #-1,R0 ;LOAD R0
RBEG: NOP
.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV # $CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SWR,R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #STACK,SP ;;SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
MOV # $SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@IOTVEC+2 ;;LEVEL 7
MOV # $ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340,@EMTVEC+2 ;;LEVEL 7
MOV # $TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340,@TRAPVEC+2;LEVEL 7
MOV # $PWRDN,@PWRVEC ;;POWER FAILURE VECTOR
MOV #340,@PWRVEC+2 ;;LEVEL 7
MOV SENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOV #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
MOV #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
```

```
(1) 001716 012737 001716 001110      MOV      #,$LPERR      ;;SETUP THE ERROR LOOP ADDRESS
(2)                                     ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2)                                     ;;EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 001724 013746 000004                MOV      @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 001730 012737 001764 000004        MOV      #64$,@#ERRVEC ;;SET UP ERROR VECTOR
(2) 001736 012737 177570 001140        MOV      #DSWR,SWR     ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 001744 012737 177570 001142        MOV      #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 001752 022777 177777 177160        CMP      #-1,@SWR     ;;TRY TO REFERENCE HARDWARE SWR
(2) 001760 001012                        BNE      66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2)                                     ;;AND THE HARDWARE SWR IS NOT -1
(2) 001762 000403                        BR       65$          ;;BRANCH IF NO TIMEOUT
(2) 001764 012716 001772                64$:     MOV      #65$,(SP)    ;;SET UP FOR TRAP RETURN
(2) 001770 000002                        RTI
(2) 001772 012737 000176 001140        65$:     MOV      #SWREG,SWR   ;;POINT TO SOFTWARE SWR
(2) 002000 012737 000174 001142        MOV      #DISPREG,DISPLAY
(2) 002006 012637 000004                66$:     MOV      (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 002012 005037 001176                CLR      $PASS       ;;CLEAR PASS COUNT
(2) 002016 132737 000200 001211        BITB    #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 002024 001403                        BEQ     67$          ;;YES,USE NON-APT SWITCH
(2) 002026 012737 001212 001140        MOV     #$$SWREG,SWR ;;NO,USE APT SWITCH REGISTER
(2) 002034                                67$:
3077                                     ;
(1)                                     ; THIS SECTION OF CODE HANDLES INITIALIZING LPA-11 FUNCTIONS
(1)                                     ;
(1)
(1) 002034 010046                        MOV     R0,-(SP)
(1) 002036 010146                        MOV     R1,-(SP)
(1) 002040 013700 001362                MOV     KMADO,R0      ;GET KMC-11 ADDRESS.
(1) 002044 012701 001364                MOV     #KMAD1,R1    ;GET ADDR. OF ADDR. LIST.
(1)
(1) 002050                                68$:     INC     R0            ;UPDATE ADDR.
(1) 002052 010021                        MOV     R0,(1)+      ;WRITE ADDR.
(1) 002054 020127 001402                CMP     R1,#KMAD7+2  ;DONE ALL ADDRESSES?
(1) 002060 001373                        BNE     68$          ;NO - DO NEXT ADDR.
(1) 002062 005037 001410                CLR     .DVLS        ;CLR ADDR. LIST.
(1) 002066 012601                        MOV     (SP)+,R1
(1) 002070 012600                        MOV     (SP)+,R0
3078 002072 013737 001244 001452        MOV     $BASE,BASEBA ;GET APT DEFINED ADDRESS
3079 002100 013737 001240 001454        MOV     $VECT1,BASEIV ;GET VECTOR
3080 002106 042737 160000 001454        BIC     #160000,BASEIV
3081 002114 113737 001241 001456        MOV     $VECT1+1,BASEBR ;GET PRIORITY
3082 002122 042737 177437 001456        BIC     #177437,BASEBR
3083 002130 012737 002176 000004        MOV     #1$,@#ERRVEC ;LOAD BUS ERROR
3084 002136 013702 001452                MOV     BASEBA,R2    ;LOAD STARTING ADDRESS
3085 002142 005003                        CLR     R3           ;CLEAR COUNT
3086 002144 010237 001544                2$:     MOV     R2,$TMDAT   ;TEST IF EXISTENT
3087
(1)
3088 002160 005737 021456                ;*     MOV     @$TMDAT,$GDDAT ;/READ DEVICE REG $TMDAT,PUT DATA IN $GDDAT.
3089 002164 001004                        TST     $AERR
3090 002166 162702 000010                BNE     1$
3091 002172 005203                        SUB     #10,R2       ;EXIST, UPDATE TEST ADDRESS
3092 002174 000763                        INC     R3           ;UPDATE # OF DR11'S
3093 002176 005703                        1$:     BR      2$
3094 002200 001014                        BNE     3$           ;TEST IF FIRST DOES EXIST
                                     ;BR
```

```

3095 002202 032777 020000 176730 BIT #SW13,@SWR ;TEST IF INHIBIT TYPEOUT
3096 002210 001002 BNE 4$ ;BR IF YES
3097 002212 104401 013754 TYPE, BUSTRP ;TELL OPERATOR BASE DR11K DOESN'T EXIST
3098 002216 032777 100000 176714 4$: BIT #SW15,@SWR ;TEST HALT ON ERROR
3099 002224 001401 BEQ 5$ ;BR IF NO HALT
3100 002226 000000 HALT ;FIRST DR11-K DOES NOT EXIST
3101 ;CHECK THE PROGRAM DEVICE ADDRESS
3102 002230 000745 5$: BR 2$ ;TRY AGAIN
3103
3104 002232 005303 3$: DEC R3 ;ADJUST R3
3105 002234 010337 001462 MOV R3,NMBEXT ;SAVE THE NUMBER OF ADDITIONAL DR11-K
3106 002240 012737 003010 000004 MOV #VECTRP,@ERRVEC ;RESET BUS ERROR
3107 002246 012737 000340 000006 MOV #340,@ERRVEC+2
3108 002254 004737 022430 RBEG1: JSR PC,$RESET
3109 002260 005737 000042 TST @#42 ;TEST IF UNDER A MONITOR
3110 002264 001002 BNE 4$ ;BR IF
3111 002266 005700 TST R0 ;TEST R0
3112 002270 001402 BEQ 2$ ;BR IF CLEARED
3113 002272 000137 003320 4$: JMP IOTEST ;JUMP IF SET
3114 002276 2$:
(1) 002276 104401 002304 TYPE ,65$ ;;TYPE ASCIZ STRING
(1) 002302 000433 BR 64$ ;;GET OVER THE ASCIZ
(1) ;;65$: .ASCIZ <15><12><15><12>/CRLPFB LPA-DR11-K DIGITAL INPUT OUTPUT LOGIC TEST/
(1) 002372 64$:
3115 .SBTTL TYPE PROGRAM NAME
(1) ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002372 005227 177777 INC #-1 ;;FIRST TIME?
(1) 002376 001044 BNE 66$ ;;BRANCH IF NO
(1) 002400 022737 013312 000042 CMP #SENDAD,@#42 ;;ACT-11?
(1) 002406 001440 BEQ 66$ ;;BRANCH IF YES
(1) 002410 104401 002456 TYPE ,67$ ;;TYPE ASCIZ STRING
(2) .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002414 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 002420 001012 BNE 68$ ;;BRANCH IF YES
(2) 002422 123727 001210 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
(2) 002430 001406 BEQ 68$ ;;BRANCH IF YES
(2) 002432 023727 001140 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
(2) 002440 001005 BNE 69$ ;;BRANCH IF NO
(2) 002442 104406 GTSWR ;;GET SOFT-SWR SETTINGS
(2) 002444 000403 BR 69$
(2) 002446 112737 000001 001134 68$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
(2) 002454 69$:
(1) 002454 000415 BR 66$ ;;GET OVER THE ASCIZ
(1) ;;67$: .ASCIZ <CRLF><15><12>/MAINDEC-11-CRLPF-B/<15><12><CRLF>
(1) 66$:
3116 002510 013746 001462 MOV NMBEXT,-(SP)
3117 002514 104403 TYPOS
3118 002516 000002 .WORD 2
3119 002520 104401 002526 TYPE ,71$ ;;TYPE ASCIZ STRING
(1) 002524 000422 BR 70$ ;;GET OVER THE ASCIZ
(1) ;;71$: .ASCIZ /(8) ADDITIONAL DR11-K'S CONNECTED/<15><12>
(1) 70$:
3120 002572 022737 000001 001542 CMP #1,JUMPER ;TEST IF LOC-BOX CONFIG.
3121 002600 001013 BNE 1$ ;BR IF NOT
3122 002602 005037 001510 CLR NOTLCH ;LOAD LOC-BOX JUMPER CONFIG
3123 002606 012737 177777 001512 MOV #-1,INTBIT ;
    
```

```

LPA DR11-K LOGIC TEST MD-11-CRLPF-B MACY11 30G(1063) 08-AUG-79 L 2 10:09 PAGE 7-8
CRLPFB.P11 08-AUG-79 10:08 GET VALUE FOR SOFTWARE SWITCH REGISTER SEQ 0024

3124 002614 005037 001536 CLR MINSIN
3125 002620 005037 001540 CLR TRANST
3126 002624 000137 003320 JMP IOTEST ;RUN THE LOGIC TEST WITH LOC-BOX JUMPERS
3127 002630 022737 000002 001542 1$: CMP #2,JUMPER ;TEST IF COULTER CONFIG.
3128 002636 001015 BNE 3$ ;BR IF NOT
3129 002640 012737 177777 001510 MOV #-1,NOTLCH ;LOAD COULTER JUMPER CONFIG.
3130 002646 012737 170000 001512 MOV #170000,INTBIT
3131 002654 005037 001536 CLR MINSIN
3132 002660 012737 170000 001540 MOV #170000,TRANST
3133 002666 000137 003320 JMP IOTEST ;RUN THE LOGIC TEST WITH COULTER JUMPERS
3134 002672 004537 003234 3$: JSR R5,TALK ;TALK TO THE ANIMALS
3135 002676 014015 SWNLB
3136 002700 001510 NOTLCH ;ABOUT NON LATCH INPUT BITS
3137 002702 004537 003234 JSR R5,TALK ;TALK TO THE ANIMAL AGAIN
3138 002706 014113 SWINTB
3139 002710 001512 INTBIT ;ABOUT THE INTERRUPTING BITS
3140 002712 004537 003234 JSR R5,TALK ;ISN'T THIS BOARING
3141 002716 014211 SWPOSB
3142 002720 001536 MINSIN ;AGAIN-ABOUT THE MINUS INPUT BITS
3143 002722 004537 003234 JSR R5,TALK ;HO-HUM
3144 002726 014305 SWTRAB
3145 002730 001540 TRANST ;AGAIN-ABOUT THE TRANSITION BITS
3146
3147 ; DETERMINE THAT THE OPERATOR DID NOT MAKE A LOGICAL SWITCH ERROR
3148 002732 012737 010000 001124 MOV #BIT12,$GDDAT ;LOAD TEST BIT
3149 002740 033737 001124 001536 10$: BIT $GDDAT,MINSIN ;TEST IF NEG INPUT BIT
3150 002746 001407 BEQ 11$ ;BR IF NOT
3151 002750 033737 001124 001540 BIT $GDDAT,TRANST ;TEST IF TRANS. INPUT
3152 002756 001403 BEQ 11$ ;BR IF NOT
3153 002760 104007 ERROR 7 ;LOGICAL OPEF- OR ERROR
3154 ;INPUT BIT CAN'T BE NEG. AND TRANS. AT THE SAME
3155 002762 000137 001546 JMP BEGIN ;
3156
3157 002766 006137 001124 11$: ROL $GDDAT ;MOVE LEFT
3158 002772 103362 BCC 10$ ;BR UNTIL DONE
3159
3160 002774 004537 003234 JSR R5,TALK ;ASK THE OPERATOR
3161 003000 014403 SWDPOB
3162 003002 001532 BRLEV3 ;ABOUT PROGRAM OPTIONS
3163 003004 000137 003320 JMP IOTEST
3164
3165 ; INTERRUPT TO UNEXPECTED VECTOR
3166 003010 021627 001000 VECTR: CMP (SP),#1000 ;TEST IF IN PROGRAM CODE
3167 003014 103402 BLO 1$ ;BR IF NOT
3168 003016 000000 70$: HALT ;FATAL BUS TRAP IN PROGRAM AREA
3169 003020 000776 BR 70$
3170 003022 021627 000200 1$: CMP (SP),#200 ;TEST IF IN SACRED VECTOR AREA
3171 003026 101002 BHI 2$ ;BR IF NOT
3172 003030 000000 71$: HALT ;FATAL VECTOR TRAP
3173 003032 000776 BR 71$
3174 003034 012637 003232 2$: MOV (SP)+,72$ ;GET ADDR THE TRAP OCCURRED TO
3175 003040 162737 000004 003232 SUB #4,72$ ;MAKE REAL ADDRESS
3176 003046 032777 020000 176064 BIT #SW13,@SWR ;TEST IF TYPE ERROR INHIBIT
3177 003054 001050 BNE 3$ ;BR IF INHIBIT
3178 003056 104401 003064 TYPE ,65$ ;:TYPE ASCIZ STRING
(1) 003062 000417 BR 64$ ;:GET OVER THE ASCIZ

```

```

(1)
(1) 003122
3179 003122 013746 003232
3180 003126 104402
3181 003130 104401 003136
(1) 003134 000420
(1)
(1) 003176
3182 003176 042737 000007 003232
3183 003204 032777 100000 175726
3184 003212 001401
3185 003214 000000
3186 003216 022626
3187 003220 013737 003232 001470
3188 003226 000137 003346
3189 003232 000000
3190
3191
3192
3193 003234 012537 003246
3194 003240 012537 003312
3195 003244 104401
3196 003246 014015
3197 003250 023727 001140 000176
3198 003256 001006
3199 003260 104401 020223
3200 003264 104412
3201 003266 012677 175646
3202 003272 000403
3203 003274 104401
3204 003276 014470
3205 003300 000000
3206 003302 017777 175632 000002
3207 003310 000205
3208 003312 001510
3209
3210
3211 003314 104006
3212 003316 000002
3213
3214 003320 004737 022430
3215 003324 013737 001452 001466
3216 003332 013737 001454 001470
3217 003340 013737 001462 001464
3218 003346 004737 022430
3219 003352 012701 000240
3220 003356 012702 000242
3221 003362 010221
3222 003364 012721 004700
3223 003370 022222
3224 003372 020227 001076
3225 003376 001371
3226
3227 003400 004737 022430
3228 003404 004737 003412
3229 003410 000453

::65$: .ASCIZ <15><12>/DR11K INTERRUPTED TO LOC. /
64$:
MOV 72$,-(SP) ;LOAD VALUE TO BE TYPED
TYPOC
TYPE ,67$ ;:TYPE ASCIZ STRING
BR 66$ ;:GET OVER THE ASCIZ
::67$: .ASCIZ / AND WILL NOW USE THAT VECTOR/<15><12>
66$:
3$: BIC #7,72$ ;:MASK OFF LOWER 3 BITS
BIT #SW15,@SWR ;:TEST IF HALT ON ERROR
BEQ 4$ ;:BR IF NOT
HALT ;:DR11K INTERRUPTED TO WRONG VECTOR-PROG WILL NOW USE THAT VECTOR
4$: CMP (SP)+,(SP)+ ;:CLEAN THE STACK
MOV 72$,DRIV ;:LOAD NEW VECTOR ADDRESS
JMP RBEG2 ;:TEST THE DR11K AT THE NEW VECTOR
72$: 0

;OPERATOR QUESTIONS AND ANSWER ROUTINE
TALK: MOV (R5)+,10$ ;:GET ASCII POINTER
MOV (R5)+,11$ ;:GET POINTER TO DATA FROM OPERATOR
TYPE
10$: SWNLB ;:TELL OPERATOR TO LOAD SWITCHES
CMP SWR,#SWREG ;:TEST IF SWITCH REG. EXISTS
BNE 1$ ;:BR IF NO
TYPE ,SMSWR ;:TYPE 'SWR'='
RDOCT ;:READ OCTAL
MOV (SP)+,@SWR ;:SAVE THE SWITCHES
BR 2$
1$: TYPE
DEPCNT ;:TELL OPERATOR TO DEPRESS CONT
HALT ;:WAIT FOR OPERATOR
2$: MOV @SWR,@11$ ;:LOAD SWITCH VALUE
RTS R5 ;:EXIT
11$: NOTLCH ;:POINTER TO DATA TO BE LOADED

;UNEXPECTED INTERRUPT
UNEXPT: ERROR 6 ;:UNEXPECTED INTERRUPT DURING A SUB-TEST
RTI ;:EXIT

IOTEST: JSR PC,$RESET
MOV BASEBA,DRADD ;:LOAD INITIAL STARTING ADDRESS
MOV BASEIV,DRIV ;:LOAD INITIAL STARTING VECTOR
MOV NMBEXT,NBEXT ;:RELOAD # AVAILABLE
RBEG2: JSR PC,$RESET
MOV #240,R1 ;:LOAD R1
MOV #242,R2
5$: MOV R2,(R1)+ ;:SAVE THE ADDRESS
MOV #4700,(R1)+ ;:LOAD 'JSR PC,R0'
CMP (R2)+,(R2)+ ;:BUMP R2
CMP R2,#$CMTAG-2 ;:TEST FOR LAST
BNE 5$ ;:BR UNTIL DONE

IOTST1: JSR PC,$RESET
JSR PC,SETADD ;:SET UP BUS ADDRESS AND VECTOR
BR IOTTS1

```



```
3230 003412 013737 001466 001500 SETADD: MOV DRADD,GRSTAT ;LOAD 1ST ADDRESS
3231 003420 013737 001466 001502 MOV DRADD,GRDAI ;LOAD 2ND ADDRESS
3232 003426 062737 000002 001502 ADD #2,GRDAI
3233 003434 013737 001466 001504 MOV DRADD,GRDIO ;LOAD 3RD ADDRESS
3234 003442 062737 000004 001504 ADD #4,GRDIO
3235 003450 013737 001466 001506 MOV DRADD,GRBHIO ;LOAD 4TH ADDRESS
3236 003456 062737 000005 001506 ADD #5,GRBHIO
3237 003464 013737 001470 001514 MOV DRIV,GRIVA ;LOAD FIRST VECTOR
3238 003472 013737 001470 001516 MOV DRIV,GRIVSA
3239 003500 062737 000002 001516 ADD #2,GRIVSA
3240 003506 013737 001470 001520 MOV DRIV,GRIVB ;LOAD 2ND VECTOR
3241 003514 062737 000004 001520 ADD #4,GRIVB
3242 003522 013737 001470 001522 MOV DRIV,GRIVSB
3243 003530 062737 000006 001522 ADD #6,GRIVSB
3244 003536 000207 RTS PC
3245 003540 013737 001456 001524 IOTTS1: MOV BASEBR,DIOBRL ;LOAD BR LEVEL
3246 003546 005037 001534 CLR ODDJMP
3247 003552 053737 001536 001534 BIS MINSIN,ODDJMP ;SET MINUS INPUT BITS
3248 003560 053737 001540 001534 BIS TRANST,ODDJMP ;SET TRANSITION INPUT BITS
3249 003566 012777 003314 175720 MOV #UNEXPT,@GRIVA ;RESET INPUT VECTOR
(1) 003574 012777 000340 175714 MOV #340,@GRIVSA
(1) 003602 012777 003314 175710 MOV #UNEXPT,@GRIVB ;RESET OUTPUT VECTOR
(1) 003610 012777 000340 175704 MOV #340,@GRIVSB
3250 *****
(3) : *TEST 1 TEST FOR NO BUS ERRORS
(3) *****
(2) 003616 000004 TST1: SCOPE
3251 003620 012737 000001 001174 MOV #1,$TESTN
3252 003626 012737 000001 001102 MOV #1,$STNM
3253 003634 012737 000000 001544 MOV #0,$TMDAT
3254 (1)
3255 : * MOV $TMDAT,@GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
(1) : * MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
3256 (1) : * MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3257
3258 *****
(3) : *TFST 2 TEST THAT OUTPUT REG. CAN HOLD #-1
(3) *****
(2) 003672 000004 TST2: SCOPE
3259 003674 012737 177777 001124 MOV #-1,$GDDAT ;LOAD EXPECTED
3260 003702 012737 177777 001544 MOV #-1,$TMDAT ;ALL ONES TO REGISTER
3261 (1)
3262 : * MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(1) : * MOV @GRDIO,$BDDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
3263 003730 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
3264 003736 001401 BEQ TST3 ;:BR IF EQUAL
3265 003740 104003 ERROR 3 ;REG WILL NOT HOLD ONES
3266 *****
(3) : *TEST 3 TEST THAT RESET CLEARS OUTPUT REG.
(3) *****
(2) 003742 000004 TST3: SCOPE
(1) 003744 012737 000002 001160 MOV #2,$TIMES ;:DO 2 ITERATIONS
3267 003752 005037 001124 CLR $GDDAT
```

3268 003756 012737 177777 001544
3269
(1)
3270 003774 004737 022430
3271
(1)
3272 004010 005737 001126
3273 004014 001401
3274 004016 104003
3275
3276
(3)
(3)
(2) 004020 000004
3277 004022 012737 052525 001124
3278 004030 012737 052525 001544
3279
(1)
3280
(1)
3281 004056 023737 001124 001126
3282 004064 001401
3283 004066 104003
3284
3285
(3)
(3)
(2) 004070 000004
3286 004072 012737 125252 001124
3287 004100 012737 125252 001544
3288
(1)
3289
(1)
3290 004126 023737 001124 001126
3291 004134 001401
3292 004136 104003
3293
3294
(3)
(3)
(2) 004140 000004
(1) 004142 012737 000001 001160
3295 004150 012737 004162 001110
3296 004156 005037 001124
3297 004162
(1)
(1)
3298
(1)
3299 004202 023737 001124 001126
3300 004210 001401
3301 004212 104003
3302 004214 005237 001124
3303 004220 001360
3304

```
MOV #1,$TMDAT
;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
JSR PC,$RESET ;SET DATA TO ALL ONES
;* MOV @GRDIO,$BDDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
TST $BDDAT
BEQ TST4 ;:BR IF EQUAL
ERROR 3 ;REG FAILED TO CLEAR

*****
;*TEST 4 TEST THAT OUTPUT REG. CAN HOLD #52525
*****
TST4: SCOPE
MOV #52525,$GDDAT ;LOAD EXPECTED VALUE
MOV #52525,$TMDAT
;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
;* MOV @GRDIO,$BDDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST5 ;:BR IF EQUAL
ERROR 3 ;DATA NOT=52525

*****
;*TEST 5 TEST THAT OUTPUT REG. CAN HOLD #125252
*****
TST5: SCOPE
MOV #125252,$GDDAT ;LOAD EXPECTED VALUE
MOV #125252,$TMDAT
;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
;* MOV @GRDIO,$BDDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST6 ;:BR IF EQUAL
ERROR 3 ;DATA NOT=125252

*****
;*TEST 6 TEST THAT OUTPUT REG. CAN HOLD A COUNT PATTERN
*****
TST6: SCOPE
MOV #1,$TIMES ;:DO 1 ITERATION
MOV #2,$SLPERR ;LOAD SCOPE ERROR RETURN
CLR $GDDAT ;CLEAR PATTERN
2$:
;* MOV $GDDAT,@GRDIO ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
;* MOV @GRDIO,$BDDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 1$ ;:BR IF EQUAL
ERROR 3 ;OUTPUT REG.FAILED TO HOLD COUNT PATTERN
1$: INC $GDDAT ;UPDATE THE PATTERN
BNE 2$ ;TRY AGAIN
*****
```

```
(3) ;*TEST 7 FLOAT A 1 ACROSS THE OUTPUT REGISTER
(3) ;*****
(2) 004222 000004 TST7: SCOPE
3305 004224 012737 004240 001110 MOV #1$, $LPERR ;LOAD SCOPE ERROR RETURN
3306 004232 012737 000001 001124 MOV #BIT0, $GDDAT ;LOAD EXPECTED VALUE
3307
3308 004240 012737 000000 001544 1$: MOV #0, $TMDAT ;CLEAR OUTPUT
3309
(1) ;* MOV $TMDAT, @GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3310 004256 053737 001124 001544 BIS $GDDAT, $TMDAT ;SET THAT BIT
3311
(1) ;* MOV $TMDAT, @GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3312
(1) ;* MOV @GRDIO, $BDDAT ;/READ DEVICE REG GRDIO, PUT DATA IN $BDDAT.
3313 004304 023737 001124 001126 CMP $GDDAT, $BDDAT ;TEST RESULTS
3314 004312 001401 BEQ 2$ ;BR IF EQUAL ?
3315 004314 104003 ERROR 3
3316
3317 004316 006337 001124 2$: ASL $GDDAT ;SHIFT EXPECTED DATA
3318 004322 001346 BNE 1$ ;BR UNTIL DONE
3319
;*****
(3) ;*TEST 10 FLOAT A 0 ACROSS THE OUTPUT REGISTER
(3) ;*****
(2) 004324 000004 TST10: SCOPE
3320 004326 012737 004342 001110 MOV #1$, $LPERR ;LOAD SCOPE ERROR RETURN
3321 004334 012737 000001 001124 MOV #BIT0, $GDDAT ;LOAD EXPECTED VALUE
3322
3323 004342 012737 177777 001544 1$: MOV #-1, $TMDAT ;LOAD OUTPUT TO A ONE
3324
(1) ;* MOV $TMDAT, @GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3325 004360 043737 001124 001544 BIC $GDDAT, $TMDAT ;CLEAR A BIT
3326
(1) ;* MOV $TMDAT, @GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3327
(1) ;* MOV @GRDIO, $BDDAT ;/READ DEVICE REG GRDIO, PUT DATA IN $BDDAT.
3328 004406 005137 001126 COM $BDDAT ;COMPLEMENT IT
3329 004412 023737 001124 001126 CMP $GDDAT, $BDDAT ;EQUAL ?
3330 004420 001401 BEQ 2$ ;BR IF EQUAL
3331 004422 104003 ERROR 3
3332
3333 004424 006337 001124 2$: ASL $GDDAT ;SHIFT LEFT
3334 004430 001344 BNE 1$ ;BRANCH UNTIL DONE
3335
;*****
(3) ;*TEST 11 TEST FOR SLOW OUTPUT GATES WITH #125252
(3) ;*****
(2) 004432 000004 TST11: SCOPE
3337 004434 012737 125252 001124 MOV #125252, $GDDAT ;LOAD EXPECTED VALUE
3338
(1) ;* MOV $GDDAT, @GRDIO ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
3343
(2) ;* MOV @GRDIO, $TMDAT ;/READ DEVICE REG GRDIO, PUT DATA IN $TMDAT.
(1) 004462 005137 001544 COM $TMDAT
(2)
(2) ;* MOV $TMDAT, @GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
```

D 3

```
(2) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.  
(1) 004506 005137 001544 COM $TMDAT  
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
(2) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.  
(1) 004532 005137 001544 COM $TMDAT  
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
(2) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.  
(1) 004556 005137 001544 COM $TMDAT  
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
(2) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.  
(1) 004602 005137 001544 COM $TMDAT  
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
(2) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.  
(1) 004626 005137 001544 COM $TMDAT  
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
(2) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.  
(1) 004652 005137 001544 COM $TMDAT  
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
(2) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.  
(1) 004676 005137 001544 COM $TMDAT  
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
3344 (1) ;* MOV @GRDIO,$BDDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.  
3345 004722 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST REGISTER  
3346 004730 001401 BEQ TST12 ;:BR IF CORRECT  
3347 004732 104003 ERROR 3  
3348 ;*****  
(3) ;*TEST 12 TEST FOR SLOW OUTPUT GATES WITH #52525  
(3) ;*****  
TST12: SCOPE  
3349 004736 000004 052525 001124 MOV #52525,$GDDAT ;LOAD EXPECTED VALUE  
3350 (1) ;* MOV $GDDAT,@GRDIO ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO  
3355 (2) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.  
(1) 004764 005137 001544 COM $TMDAT  
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
(2) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.  
(1) 005010 005137 001544 COM $TMDAT  
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
```

```

(2)
(2)
(1) 005034 005137 001544      ;*   MOV   @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
(2)                               COM   $TMDAT
(2)                               ;*   MOV   $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)                               ;*   MOV   @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
(1) 005060 005137 001544      COM   $TMDAT
(2)                               ;*   MCV   $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)                               ;*   MOV   @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
(1) 005104 005137 001544      COM   $TMDAT
(2)                               ;*   MOV   $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)                               ;*   MOV   @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
(1) 005130 005137 001544      COM   $TMDAT
(2)                               ;*   MOV   $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)                               ;*   MOV   @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
(1) 005154 005137 001544      COM   $TMDAT
(2)                               ;*   MOV   $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)                               ;*   MOV   @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
(1) 005200 005137 001544      COM   $TMDAT
(2)                               ;*   MOV   $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)                               ;*   MOV   @GRDIO,$BDDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $BDDAT.
3356                               CMP   $GDDAT,$BDDAT  ;TEST PATTERN
(1)                               BEQ   TST13           ;;BR IF EQUAL
3357 005224 023737 001124 001126 ;*   ERROR 3           ;OUTPUT REGISTER IN ERROR
3358 005232 001401
3359 005234 104003
3360
3361 ;:*****
(3) ;*TEST 13 TEST OUTPUT DATA ACCEPT FLAG
(3) ;:*****
(2) 005236 000004 TST13: SCOPE
3362 005240 012737 000000 001544 MOV   #0,$TMDAT
3363
(1) ;*   MOV   $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3364 005256 012737 177777 001544 ;*   MOV   #-1,$TMDAT
3365
(1) ;*   MOV   $TMDAT,@GRDAI  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
3366 005274 012737 100000 001124 ;*   MOV   #BIT15,$GDDAT ;LOAD EXPECTED
3367
(1) ;*   MOV   $GDDAT,@GRSTAT  ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
3368
(1) ;*   MOV   @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
3369 005322 033737 001124 001126 ;*   BIT   $GDDAT,$BDDAT
3370 005330 001001 ;*   BNE   TST14           ;;BR IF SET
3371 005332 104001 ;*   ERROR 1           ;ERROR, BIT 15 FAILED TO SET
3372
3373 ;:*****

```

```
(3) : *TEST 14 TEST OUTPUT INTERRUPT ENABLE
(3) : *****
(2) 005334 000004 TST14: SCOPE
3374 005336 012737 000000 001544 MOV #0,$TMDAT ;CLEAR STATUS
3375
(1) : * MOV $TMDAT,@GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
3376 005354 012737 040000 001124 MOV #BIT14,$GDDAT ;LOAD EXPECTED
3377
(1) : * MOV $GDDAT,@GRSTAT ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
3378
(1) : * MOV @GRSTAT,$BDDAT ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
3379 005402 033737 001124 001126 BIT $GDDAT,$BDDAT
3380 005410 001001 BNE TST15 ;;BR IF SET
3381 005412 104001 ERROR 1 ;ERROR BIT 14 FAILED TO SET
3382
3383 : *****
(3) : *TEST 15 TEST INPUT DATA READY FLAG
(3) : *****
(2) 005414 000004 TST15: SCOPE
3384 005416 012737 000200 001124 MOV #BIT7,$GDDAT ;LOAD EXPECTED
3385
(1) : * MOV $GDDAT,@GRSTAT ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
3386
(1) : * MOV @GRSTAT,$BDDAT ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
3387 005444 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
3388 005452 001401 BEQ TST16 ;;BR IF EQUAL
3389 005454 104001 ERROR 1 ;ERROR, BIT 7 FAILED TO SET
3390
3391 : *****
(3) : *TEST 16 TEST INPUT INTERRUPT ENABLE
(3) : *****
(2) 005456 000004 TST16: SCOPE
3392 005460 012737 000000 001544 MOV #0,$TMDAT ;CLEAR STATUS
3393
(1) : * MOV $TMDAT,@GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
3394 005476 012737 000100 001124 MOV #BIT6,$GDDAT ;LOAD EXPECTED
3395
(1) : * MOV $GDDAT,@GRSTAT ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRSTAT
3396
(1) : * MOV @GRSTAT,$BDDAT ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
3397 005524 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE
3398 005532 001401 BEQ TST17 ;;BR IF EQUAL
3399 005534 104001 ERROR 1 ;ERROR, BIT 6 FAILED TO SET
3400
3405
3410
3411 : *****
(3) : *TEST 17 TEST EXTERNAL TRANSFERS - CABLE MUST BE CONNECTED
(3) : *****
(2) 005536 000004 TST17: SCOPE
3412 005540 032777 002000 173372 BIT #BIT10,@SWR ;TEST SWITCH BIT
3413 005546 001402 BEQ 1$ ;BRANCH IF DOWN
3414 005550 000137 012230 JMP DRT21 ;BYPASS SOME TEST USING THE EXTERNAL CABLE
3415 005554
(1) 005554 012737 000000 001544 1$: MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
(2)
```



```

3449
3450
(3)
(3)
(2) 006106 000004
3451 006110 012737 000000 001544
(2)
(2)
3452 006126 012737 177777 001544
(2)
(2)
3453 006144 012737 125252 001124
3454 006152 043737 001534 001124
3455
(1)
3456
(1)
3457 006200 043737 001534 001126
3458 006206 023737 001124 001126
3459 006214 001401
3460 006216 104002
3461
(3)
(3)
(2) 006220 000004
(1) 006222 012737 000001 001160
3462 006230 005737 001534
3463 006234 001514
3464 006236 012737 010000 001124
3465 006244 033737 001534 001124
3466 006252 001502
3467 006254 033737 001510 001124
3468 006262 001076
3469
(1)
3470 006274 012737 177777 001544
3471
(1)
3472
(1)
3473 006322 043737 001124 001544
3474
(1)
3475
(1)
3476 006350 023737 001124 001126
3477 006356 001401
3478 006360 104002
3479
3480
3481 006362 033737 001124 001536
3482 006370 001033
3483 006372 012737 177777 001544
3484
(1)
3485

```

```

*****
*TEST 22 TEST INPUT WITH #125252
*****
TST22: SCOPE
MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
MOV #-1,$TMDAT
;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
MOV #125252,$GDDAT ;LOAD EXPECTED
BIC ODDJMP,$GDDAT ;MASK
;* MOV $GDDAT,@GRDIO ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
;* MOV @GRDAI,$BDDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
BIC ODDJMP,$BDDAT ;MASK
CMP $GDDAT,$BDDAT ;COMPARE
BEQ TST23 ;:BR IF EQUAL
ERROR 2 ;ERROR, INPUT DID NOT EQUAL OUTPUT
*****
*TEST 23 TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS
*****
TST23: SCOPE
MOV #1,$TIMES ;:DO 1 ITERATION
ODDJMP ;TEST IF ANY ODD JUMPERS
TST TST24 ;:BR IF NONE
BEQ TST24 ;:BR IF NONE
MOV #BIT12,$GDDAT ;LOAD TEST BIT
1$: BIT ODDJMP,$GDDAT ;TEST IF ODD JUMPER BIT
BEQ 2$ ;BR IF NOT
BIT NOTLCH,$GDDAT ;TEST IF LATCHING INPUT BIT
BNE 2$ ;BR IF NOT
;* MOV $GDDAT,@GRDIO ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
MOV #-1,$TMDAT ;CLEAR INPUT
;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
BIC $GDDAT,$TMDAT
;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
;* MOV @GRDAI,$BDDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
CMP $GDDAT,$BDDAT ;COMPARE
BEQ 3$ ;BR IF EQUAL
ERROR 2 ;ERROR, NEG. INPUT OR NEG. TRANSITION
; INPUT BIT FAILED TO SET INPUT REGISTER
3$: BIT $GDDAT,MINSIN ;TEST IF NEG. INPUT BIT
BNE 2$ ;BR IF
MOV #-1,$TMDAT ;CLEAR INPUT
;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI

```

```

LPA DR11-K LOGIC TEST MD-11-CRLPF-B MACY11 30G(1063) 08-AUG-79 10:09 PAGE 7-18
CRLPFB.P11 08-AUG-79 10:08 T23 TEST THE NEG. AND TRANSITION LATCHING INPUT DATA BITS SEQ 0034

(1)
3486 006420 053737 001124 001544 :* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
3487 BIS $GDDAT,$TMDAT
(1)
3488 :* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(1)
3489 006446 023737 001124 001126 :* MOV @GRDAI,$BDDAT ;/READ DFVICE REG GRDAI,PUT DATA IN $BDDAT.
3490 006454 001401 CMP $GDDAT,$BDDAT ;COMPARE
3491 006456 104002 BEQ 2$ ;BR IF EQUAL
3492 ERROR 2 ;ERROR, POSITIVE INPUT TRANSITION
3493 ;LOGIC FAILED TO SET INPUT REGISTER BIT
3494 006460 006137 001124 2$: ROL $GDDAT ;CHANGE DATA PATTERN
3495 006464 103267 BCC 1$ ;BR IF MORE DATA
3496
(3)
(3)
(2) 006466 000004
3497 006470 012737 006516 001110 TST24: SCOPE
MOV #2$,$LPERR ;LOAD ERROR SCOPE RETURN
3498
3499 006476 012737 000001 001530 MOV #BIT0,BRLEV2 ;LOAD EXPECTED
3500 006504 013737 001510 001532 MOV NOTLCH,BRLEV3 ;GET NON-LATCH
3501 006512 005137 001532 COM BRLEV3 ;COMPLEMENT
3502 006516 013737 001530 001124 2$: MOV BRLEV2,$GDDAT ;LOAD GOOD
3503 006524 033737 001124 001534 BIT $GDDAT,ODDJMP ;TEST IF ODD JUMPER
3504 006532 001055 BNE 1$ ;BYPASS IF ODD JUMPER
3505 006534 033737 001124 001510 BIT $GDDAT,NOTLCH ;TEST FOR NON-LATCH
3506 006542 00145 BEQ 1$ ;BR IF LATCHING
3507
3508
(1)
3509 :* MOV $GDDAT,@GRDIO ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO
(1)
3510 006564 043737 001532 001126 :* MOV @GRDAI,$BDDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
3511 006572 023737 001124 001126 BIC BRLEV3,$BDDAT ;MASK TO LATCH BITS
3512 006600 001401 CMP $GDDAT,$BDDAT ;COMPARE
3513 006602 104002 BEQ 3$ ;:BR IF EQUAL
3514 ERROR 2 ;INPUT REGISTER IN ERROR
3515 ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
3516 ;SUB-TEST CLEAR THE OUTPUT BIT AND TEST THE INPUT DOES NOT LATCH
3517
3518 006604 3$:
(1)
(1)
3519 006614 043737 001124 001544 :* MOV @GRDIO,$TMDAT ;/READ DFVICE REG GRDIO,PUT DATA IN $TMDAT.
3520 BIC $GDDAT,$TMDAT
(1)
3521 :* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(1)
3522 006642 043737 001532 001126 :* MOV @GRDAI,$BDDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
3523 006650 005037 001124 BIC BRLEV3,$BDDAT ;MASK TO LATCH BITS
3524 006654 033737 001530 001126 CLR $GDDAT ;CLEAR EXPECTED
3525 006662 001401 BIT BRLEV2,$BDDAT ;TEST FOR BIT
3526 006664 104002 BEQ 1$ ;:BR IF CLEARED
3527 ERROR 2 ;INPUT BIT LATCHED IN ERROR
3528 ;WAS CORRECT LATCH/NON-LATCH SUPPLIED ?
3529 006666 006337 001530 1$: ASL BRLEV2 ;CHANGE PATTERN

```



```
3546  
(1) 3547 006776 012737 000000 001544 ;* MOV $GDDAT,@GRDIO ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO  
MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER  
(2) 3548 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
(2) 3549 007024 023737 001124 001126 ;* MOV @GRDAI,$BDDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.  
CMP $GDDAT,$BDDAT ;COMPARE  
3550 007032 001401 BEQ 1$ ;:BR IF EQUAL  
3551 007034 104002 ERROR 2 ;INPUT REGISTER FAILED TO LATCH DATA  
3552  
3553 007036 1$:  
(1)  
(1) 3554 007046 053737 001124 001544 ;* MOV @GRDAI,$TMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.  
BIS $GDDAT,$TMDAT  
3555  
(1) 3556 007064 006337 001124 ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI  
ASL $GDDAT ;CHANGE PATTERN  
3557 007070 001326 BNE 2$ ;BR UNTIL DONE  
3558  
3559 ;:*****  
(3) ;*TEST 26 FLOAT A 0 ACROSS LATCHING INPUT BITS  
(3) ;:*****  
(2) 007072 000004 TST26: SCOPE  
3560 007074 012737 007144 001110 MOV #2$,$LPERR ;LOAD ERROR SCOPE RETURN  
3561 007102 012737 000001 001532 MOV #BIT0,BRLEV3 ;LOAD EXPECTED  
3562 007110 012737 000000 001544 MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER  
(2)  
(2) 3563 007126 012737 177777 001544 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
MOV #-1,$TMDAT  
3564 ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI  
3565 007144 033737 001532 001510 2$: BIT BRLEV3,NOTLCH ;TEST FOR LATCHING  
3566 007152 001041 BNE 1$ ;BR IF NOT  
3567 007154 033737 001124 001534 BIT $GDDAT,ODDJMP ;TEST IF ODD JUMPER  
3568 007162 001035 BNE 1$ ;BYPASS IF ODD JUMPER BIT  
3569  
3570 007164 012737 177777 001124 MOV #-1,$GDDAT ;LOAD  
3571 007172 043737 001510 001124 BIC NOTLCH,$GDDAT  
3572 007200 043737 001532 001124 BIC BRLEV3,$GDDAT ;MAKE BRLEV3  
3573  
(1) 3574 007216 012737 000000 001544 ;* MOV $GDDAT,@GRDIO ;/ PUT DATA FROM $GDDAT TO DEVICE REG GRDIO  
MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER  
(2)  
(2) 3575 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO  
3576  
(1) 3577 007244 023737 001124 001126 ;* MOV @GRDAI,$BDDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.  
CMP $GDDAT,$BDDAT ;COMPARE  
3578 007252 001401 BEQ 1$ ;:BR IF EQUAL  
3579 007254 104002 ERROR 2 ;INPUT REGISTER FAILED TO LATCH DATA  
3580  
3581 007256 1$:  
(1)  
(1) ;* MOV @GRDAI,$TMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
```

```

LPA DR11-K LOGIC TEST MD-11-CRLPF-B MACY11 30G(1063) 08-AUG-79L 3 10:09 PAGE 8-1
CRLPFB.P11 08-AUG-79 10:08 T26 FLOAT A 0 ACROSS LATCHING INPUT BITS SEQ 0037

3582 007266 053737 001532 001544 BIS BRLEV3,$TMDAT
3583
(1)
3584 007304 006337 001532 ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
3585 007310 001315 ASL BRLEV3 ;CHANGE PATTERN
3586 ;* BNE 2$ ;BR UNTIL DONE
;*****
;*TEST 27 TEST FOR SLOW INPUT GATES WITH #125252
;*****
(2) 007312 000004 TST27: SCOPE
3587
3588 007314 012737 125252 001124 MOV #125252,$GDDAT ;LOAD EXPECTED
3589 007322 043737 001510 001124 BIC NOTLCH,$GDDAT ;CONVERT
3590 007330 043737 001534 001124 BIC ODDJMP,$GDDAT ;MASK
3591 007336 013700 001124 MOV $GDDAT,R0 ;LOAD PATTERN
3592 007342 012737 000000 001544 MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
(2)
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3593 007360 012737 177777 001544 MOV #-1,$TMDAT
(2)
(2) ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
3594
3604
(1) ;* MOV R0,@GRDIO ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(2) MOV R0,$TMDAT
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 007422 012737 000000 001544 MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
(3)
(3) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2) ;* MOV @GRDAI,$TMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 007450 013701 001544 MOV $TMDAT,R1
(2)
(2) ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 007464 005100 COM R0
(2)
(2) ;* MOV R0,@GRDIO ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(1) 007476 010037 001544 MOV R0,$TMDAT
(2)
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 007512 012737 000000 001544 MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
(3)
(3) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2) ;* MOV @GRDAI,$TMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 007540 013701 001544 MOV $TMDAT,R1
(2)
(2) ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 007554 005100 COM R0
(2)
(2) ;* MOV R0,@GRDIO ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(1) 007566 010037 001544 MOV R0,$TMDAT
(2)
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 007602 012737 000000 001544 MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
(3)

```



```
(3)          ;*      MOV      $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)          ;*      MOV      @GRDAI,$TMDAT    ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 010170 013701 001544      MOV      $TMDAT,R1
(2)
(2)          ;*      MOV      $TMDAT,@GRDAI    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 010204 005100      COM      R0
(2)
(2)          ;*      MOV      R0,@GRDIO        ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(1) 010216 010037 001544      MOV      R0,$TMDAT
(2)
(2)          ;*      MOV      $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 010232 012737 000000 001544      MOV      #0,$TMDAT    ;CLEAR OUTPUT REGISTER
(3)
(3)          ;*      MOV      $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)          ;*      MOV      @GRDAI,$TMDAT    ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 010260 013701 001544      MOV      $TMDAT,R1
(2)
(2)          ;*      MOV      $TMDAT,@GRDAI    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 010274 005100      COM      R0
(2)
(2)          ;*      MOV      R0,@GRDIO        ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(1) 010306 010037 001544      MOV      R0,$TMDAT
(2)
(2)          ;*      MOV      $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 010322 012737 000000 001544      MOV      #0,$TMDAT    ;CLEAR OUTPUT REGISTER
(3)
(3)          ;*      MOV      $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)          ;*      MOV      @GRDAI,$TMDAT    ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 010350 013701 001544      MOV      $TMDAT,R1
(2)
(2)          ;*      MOV      $TMDAT,@GRDAI    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 010364 005100      COM      R0
(2)
(2)          ;*      MOV      R0,@GRDIO        ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(1) 010376 010037 001544      MOV      R0,$TMDAT
(2)
(2)          ;*      MOV      $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 010412 012737 000000 001544      MOV      #0,$TMDAT    ;CLEAR OUTPUT REGISTER
(3)
(3)          ;*      MOV      $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)          ;*      MOV      @GRDAI,$TMDAT    ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 010440 013701 001544      MOV      $TMDAT,R1
(2)
(2)          ;*      MOV      $TMDAT,@GRDAI    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 010454 005100      COM      R0
(2)
(2)          ;*      MOV      R0,@GRDIO        ;/ PUT DATA FROM R0 TO DEVICE REG GRDIO
(1) 010466 010037 001544      MOV      R0,$TMDAT
(2)
(2)          ;*      MOV      $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 010502 012737 000000 001544      MOV      #0,$TMDAT    ;CLEAR OUTPUT REGISTER
(3)
```



```

(3)          ;      ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)          ;*      MOV      @GRDAI,$TMDAT      ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 010530 013701 001544      MOV      $TMDAT,R1
(2)
(2)          ;*      MOV      $TMDAT,@GRDAI      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 010544 005100      COM      R0
3605
3606 010546 010137 001126      MOV      R1,$BDDAT      ;LOAD READ
3607 010552 000240      NOP
3608 010554 000240      NOP
3609 010556 000240      NOP
3610 010560 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE
3611 010566 001401      BEQ     TST30      ;;BR IF EQUAL
3612 010570 104002      ERROR   2      ;INPUT GATE SLOW
3613
3614          ;*****
(3)          ;*TEST 30      TEST FOR SLOW INPUT GATES WITH #52525
(3)          ;*****
(2) 010572 000004      TST30:  SCOPE
3615
3616 010574 012737 052525 001124      MOV      #52525,$GDDAT      ;SETUP EXPECTED
3617 010602 043737 001534 001124      BIC     ODDJMP,$GDDAT      ;MASK ODD JUMPER BITS
3618 010610 043737 001510 001124      BIC     NOTLCH,$GDDAT      ;CONVERT
3619 010616 013700 001124      MOV      $GDDAT,R0
3620 010622 012737 000000 001544      MOV      #0,$TMDAT      ;CLEAR OUTPUT REGISTER
(2)
(2)          ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3621 010640 012737 177777 001544      MOV      #-1,$TMDAT
(2)
(2)          ;*      MOV      $TMDAT,@GRDAI      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
3622
3631 010656 010037 001544      MOV      R0,$TMDAT
(2)
(2)          ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 010672 012737 000000 001544      MOV      #0,$TMDAT      ;CLEAR OUTPUT REGISTER
(3)
(3)          ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)          ;*      MOV      @GRDAI,$TMDAT      ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 010720 013701 001544      MOV      $TMDAT,R1
(2)
(2)          ;*      MOV      $TMDAT,@GRDAI      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 010734 005100      COM      R0
(1) 010736 010037 001544      MOV      R0,$TMDAT
(2)
(2)          ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 010752 012737 000000 001544      MOV      #0,$TMDAT      ;CLEAR OUTPUT REGISTER
(3)
(3)          ;*      MOV      $TMDAT,@GRDIO      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)          ;*      MOV      @GRDAI,$TMDAT      ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 011000 013701 001544      MOV      $TMDAT,R1
(2)
(2)          ;*      MOV      $TMDAT,@GRDAI      ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 011014 005100      COM      R0

```

```
(1) 011016 010037 001544          MOV    R0,$TMDAT
(2)
(2)                                ;*    MOV    $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 011032 012737 000000 001544  ;*    MOV    #0,$TMDAT        ;CLEAR OUTPUT REGISTER
(3)
(3)                                ;*    MOV    $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)                                ;*    MOV    @GRDAI,$TMDAT    ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 011060 013701 001544          MOV    $TMDAT,R1
(2)
(2)                                ;*    MOV    $TMDAT,@GRDAI    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 011074 005100                  COM    R0
(1) 011076 010037 001544          MOV    R0,$TMDAT
(2)
(2)                                ;*    MOV    $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 011112 012737 000000 001544  ;*    MOV    #0,$TMDAT        ;CLEAR OUTPUT REGISTER
(3)
(3)                                ;*    MOV    $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)                                ;*    MOV    @GRDAI,$TMDAT    ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 011140 013701 001544          MOV    $TMDAT,R1
(2)
(2)                                ;*    MOV    $TMDAT,@GRDAI    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 011154 005100                  COM    R0
(1) 011156 010037 001544          MOV    R0,$TMDAT
(2)
(2)                                ;*    MOV    $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 011172 012737 000000 001544  ;*    MOV    #0,$TMDAT        ;CLEAR OUTPUT REGISTER
(3)
(3)                                ;*    MOV    $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)                                ;*    MOV    @GRDAI,$TMDAT    ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 011220 013701 001544          MOV    $TMDAT,R1
(2)
(2)                                ;*    MOV    $TMDAT,@GRDAI    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 011234 005100                  COM    R0
(1) 011236 010037 001544          MOV    R0,$TMDAT
(2)
(2)                                ;*    MOV    $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 011252 012737 000000 001544  ;*    MOV    #0,$TMDAT        ;CLEAR OUTPUT REGISTER
(3)
(3)                                ;*    MOV    $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)                                ;*    MOV    @GRDAI,$TMDAT    ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(1) 011300 013701 001544          MOV    $TMDAT,R1
(2)
(2)                                ;*    MOV    $TMDAT,@GRDAI    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 011314 005100                  COM    R0
(1) 011316 010037 001544          MOV    R0,$TMDAT
(2)
(2)                                ;*    MOV    $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) 011332 012737 000000 001544  ;*    MOV    #0,$TMDAT        ;CLEAR OUTPUT REGISTER
(3)
(3)                                ;*    MOV    $TMDAT,@GRDIO    ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2)                                ;*    MOV    @GRDAI,$TMDAT    ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
```

D 4

LPA DR11-K LOGIC TEST MD-11-CRLPF-B MACY11 30G(1063) 08-AUG-79 10:09 PAGE 8-6
 CRLPFB.P1' 08-AUG-79 10 08 T30 TEST FOR SLOW INPUT GATES WITH #52525 SEQ 0042

```

(1) 011360 013701 001544      MOV      $TMDAT,R1
(2)
(2)
(1) 011374 005100      ;*      MOV      $TMDAT,@GRDAI  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 011376 010037 001544      COM      RO
(2)
(2) 011412 012737 000000 001544 ;*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2) 011412 012737 000000 001544 ;*      MOV      #0,$TMDAT      ;CLEAR OUTPUT REGISTER
(3)
(3)
(2)
(2) 011440 013701 001544      ;*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(1) 011440 013701 001544      ;*      MOV      @GRDAI,$TMDAT  ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(2)
(2) 011440 013701 001544      ;*      MOV      $TMDAT,R1
(2)
(2) 011454 005100      ;*      MOV      $TMDAT,@GRDAI  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 011454 005100      COM      RO
(1) 011456 010037 001544      MOV      RO,$TMDAT
(2)
(2) 011472 012737 000000 001544 ;*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2) 011472 012737 000000 001544 ;*      MOV      #0,$TMDAT      ;CLEAR OUTPUT REGISTER
(3)
(3)
(2)
(2) 011520 013701 001544      ;*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(1) 011520 013701 001544      ;*      MOV      @GRDAI,$TMDAT  ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(2)
(2) 011520 013701 001544      ;*      MOV      $TMDAT,R1
(2)
(2) 011534 005100      ;*      MOV      $TMDAT,@GRDAI  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 011534 005100      COM      RO
(1) 011536 010037 001544      MOV      RO,$TMDAT
(2)
(2) 011552 012737 000000 001544 ;*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2) 011552 012737 000000 001544 ;*      MOV      #0,$TMDAT      ;CLEAR OUTPUT REGISTER
(3)
(3)
(2)
(2) 011600 013701 001544      ;*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(1) 011600 013701 001544      ;*      MOV      @GRDAI,$TMDAT  ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(2)
(2) 011600 013701 001544      ;*      MOV      $TMDAT,R1
(2)
(2) 011614 005100      ;*      MOV      $TMDAT,@GRDAI  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 011614 005100      COM      RO
(1) 011616 010037 001544      MOV      RO,$TMDAT
(2)
(2) 011632 012737 000000 001544 ;*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2)
(2) 011632 012737 000000 001544 ;*      MOV      #0,$TMDAT      ;CLEAR OUTPUT REGISTER
(3)
(3)
(2)
(2) 011660 013701 001544      ;*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(1) 011660 013701 001544      ;*      MOV      @GRDAI,$TMDAT  ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
(2)
(2) 011660 013701 001544      ;*      MOV      $TMDAT,R1
(2)
(2) 011674 005100      ;*      MOV      $TMDAT,@GRDAI  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
(1) 011674 005100      COM      RO
3632
3633 011676 010137 001126      MOV      R1,$BDDAT      ;LOAD VALUE READ
3634 011702 000240      NOP
3635 011704 000240      NOP

```

```

3636 011706 000240      NOP
3637 011710 023737 001124 001126    CMP    $GDDAT,$BDDAT      ;COMPARE
3638 011716 001401      BEQ    TST31              ;;BR IF EQUAL
3639 011720 104002      ERROR  2
3640                                     ;*****
(3)                                     ;*TEST 31      TEST THAT RESET CLEARS INPUT REGISTER BITS
(3)                                     ;*****
(2) 011722 000004      TST31: SCOPE
(1) 011724 012737 000002 001160      MOV    #2,$TIMES          ;;DO 2 ITERATIONS
3641 011732 012737 000000 001544      MOV    #0,$TMDAT         ;CLEAR OUTPUT REGISTER
(2)                                     ;*
(2) 011750 012737 177777 001544      MOV    $TMDAT,@GRDIO     ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3642 011750 012737 177777 001544      MOV    #-1,$TMDAT
(2)                                     ;*
(2) 011766 012737 177777 001544      MOV    $TMDAT,@GRDAI     ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
3643 011766 012737 177777 001544      MOV    #-1,$TMDAT         ;LOAD OUTPUT
3644                                     ;*
(1) 012004 005037 001124 001124      MOV    $TMDAT,@GRDIO     ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3645 012004 005037 001124 001124      CLR    $GDDAT            ;LOAD EXPECTED
3646 012010 004737 022430      JSR    PC,$RESET
3647                                     ;*
(1) 012024 043737 001534 001126      MOV    @GRDAI,$BDDAT     ;/READ DEVICE REG GRDAI,PUT DATA IN $BDDAT.
3648 012024 043737 001534 001126      BIC    ODDJMP,$BDDAT     ;MASK ODD JUMPERS
3649 012032 005737 001126      TST    $BDDAT
3650 012036 001401      BEQ    TST32              ;;BR IF ALL BITS CLEARED
3651 012040 104002      ERROR  2                  ;INPUT REG. FAILED TO CLEAR UPON RESET INST.
3652                                     ;*****
3653                                     ;*TEST 32      TEST THAT WHEN OUTPUTTING THE INPUT DATA READY FLAG SETS
(3)                                     ;*****
(3)                                     ;*****
(2) 012042 000004      TST32: SCOPE
3654 012044 012737 000200 001124      MOV    #BIT7,$GDDAT      ;LOAD EXPECTED
3655 012052 012737 000000 001544      MOV    #0,$TMDAT
3656                                     ;*
(1) 012100 022727 000000 000000      MOV    $TMDAT,@GRSTAT   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
3657                                     ;*
(1) 012100 022727 000000 000000      MOV    $TMDAT,@GRDIO     ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3658 012100 022727 000000 000000      CMP    #0,#0             ;DELAY
3659                                     ;*
(1) 012116 023737 001124 001126      MOV    @GRSTAT,$BDDAT   ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
3660 012116 023737 001124 001126      CMP    $GDDAT,$BDDAT     ;COMPARE
3661 012124 001401      BEQ    TST33              ;;BR IF EQUAL
3662 012126 104001      ERROR  1                  ;INPUT DATA READY FLAG FAILED TO SET
3663                                     ;TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
3664                                     ;*****
3665                                     ;*TEST 33      TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET
(3)                                     ;*****
(3)                                     ;*****
(2) 012130 000004      TST33: SCOPE
3667 012132 012737 100000 001124      MOV    #BIT15,$GDDAT    ;LOAD EXPECTED
3668 012140 012737 000000 001544      MOV    #0,$TMDAT
3669                                     ;*
(1) 012130 000004      MOV    $TMDAT,@GRSTAT   ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
3670 (1) 012140 012737 000000 001544      MOV    $TMDAT,@GRDIO     ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO

```

F 4

IPA DR11-K LOGIC TEST MD-11-CRLPF-B MACY11 30G(1063) 08-AUG-79 10:09 PAGE 8-8
CRLPFB.P11 08-AUG-79 10:08 T33 TEST THAT WHEN THE INPUT BUFFER IS READ THE OUTPUT FLAG IS SET SEQ 0044

```

3671 012166 022727 000000 000000      CMP      #0,#0
3672 (1)
3673 012204 013700 001544      :*      MOV      @GRDAI,$TMDAT  ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
3674 (1)      MOV      $TMDAT,R0
3675 012220 005737 001126      :*      MOV      @GRSTAT,$BDDAT  ;/READ DEVICE REG GRSTAT,PUT DATA IN $BDDAT.
3676 012224 100401      TST      $BDDAT
3677 012226 104001      BMI      TST34      ;:BR IF SET
3678      ERROR      1      ;INPUT DATA READY FLAG FAILED TO SET
3679
3680 012230      DRT21:
3681 (4)      ;:*****
3682 (3)      ;:*TEST 34      TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG
3683 (3)      ;:*****
3684 (2) 012230 000004      TST34: SCOPE
3685 012232 032777 002000 166700      BIT      #BIT10,@SWR      ;:TEST CABLE SWITCH
3686 012240 001172      BNE      TST35      ;:BYPASS IF NO I/O CABLE
3687 012242 012737 012256 001110      MOV      #1$,$LPERR      ;:LOAD ERROR SCOPE RETURN
3688 012250 012737 000001 001532      MOV      #BIT0,BRLEV3      ;:LOAD INTERRUPT BIT
3689 012256 005037 001126      1$: CLR      $BDDAT      ;:CLEAR BAD DATA
3690 012262 012737 000200 001124      MOV      #BIT7,$GDDAT      ;:LOAD GOOD DATA
3691 012270 033737 001532 001512      BIT      BRLEV3,INTBIT      ;:TEST IF THIS BIT WILL INTERRUPT
3692 012276 001550      BEQ      3$      ;:NO TRY NEXT BIT
3693 012300 012737 000000 001544      MOV      #0,$TMDAT      ;:CLEAR OUTPUT REGISTER
3694 (2)
3695 (2)
3696 012316 012737 177777 001544      :*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3697 (2)      MOV      #-1,$TMDAT
3698 (2)
3699 012334 012737 000000 001544      :*      MOV      $TMDAT,@GRDAI  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
3700 (1)      MOV      #0,$TMDAT      ;:CLEAR STATUS
3701 (1)      :*      MOV      $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
3702 012362 053737 001532 001544      :*      MOV      @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
3703 (1)      BIS      BRLEV3,$TMDAT
3704 (1)      :*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3705 012410 043737 001532 001544      :*      MOV      @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
3706 (1)      BIC      BRLEV3,$TMDAT
3707 (1)      :*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3708 (1)      :*      MOV      @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
3709 012436 053737 001532 001544      :*      MOV      $TMDAT,@GRDIO  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3710 (1)      :*      MOV      @GRDIO,$TMDAT  ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
3711 012454 012737 000000 001544      MOV      #0,$TMDAT      ;:CLEAR STATUS
3712 (1)      :*      MOV      $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
3713 (1)      :*      MOV      $TMDAT,@GRSTAT  ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT

```

```

LPA DR11-K LOGIC TEST MD-11-CRLPF-B MACY11 30G(1063) 08-AUG-79 10:09 PAGE 8-9
CRLPFB.P11 08-AUG-79 10:08 T34 TEST THAT INTERRUPT INPUT BITS SET INPUT READY FLAG SEQ 0045

3708 ;IF INTERRUPT INPUT SWITCH IS ON
3709
(1) ;* MOV @GRSTAT,$TMDAT ;/READ DEVICE REG GRSTAT,PUT DATA IN $TMDAT.
3710 012512 105737 001544 ;* TSTB $TMDAT
3711 012516 100401 BMI 2$ ;:BR IF SET
3712 012520 104010 ERROR 10 ;INPUT INTERRUPT BIT FAILED TO SET INPUT READY
3713 ;?? DID OPERATOR GIVE CORRECT
3714 ;INPUT INTERRUPT BITS ??
3715
3716 012522 2$:
(1)
(1) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
3717 012532 043737 001532 001544 ;* BIC BRLEV3,$TMDAT
3718
(1) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3719
(1) ;* MOV @GRDAI,$TMDAT ;/READ DEVICE REG GRDAI,PUT DATA IN $TMDAT.
3720 012560 053737 001532 001544 ;* BIS BRLEV3,$TMDAT
3721
(1) ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
3722 012576 005037 001124 CLR $GDDAT ;CLEAR EXPECTED
3723 012602 005077 166672 CLR @GRSTAT ;CLEAR STATUS
3724 012606 117737 166666 001126 MOVB @GRSTAT,$BDDAT ;READ STATUS
3725 012614 100001 BPL 3$ ;:BR IF CLEARED
3726 012616 104001 ERROR 1 ;INPUT READY FAILED TO CLEAR
3727
3728 012620 006337 001532 3$: ASL BRLEV3 ;CHANGE BIT
3729 012624 001214 BNE 1$ ;BR IF NOT DONE
3730
(3) ;*****
(3) ;*TEST 35 TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG
(2) ;*****
3731 012626 000004 TST35: SCOPE
3732 012630 032777 002000 166302 BIT #BIT10,@SWR ;TEST CABLE SWITCH
3733 012636 001127 BNE TST36 ;:BYPASS IF NO I/O CABLE
3734 012640 012737 012654 001110 MOV #1$,$LPERR ;LOAD ERROR SCOPE RETURN
3735 012646 012737 000001 001532 MOV #BIT0,BRLEV3 ;LOAD NON-INTERRUPT BIT
3736 012654 012737 000200 001126 1$: MOV #200,$BDDAT ;LOAD BAD DATA
3737 012662 005037 001124 CLR $GDDAT ;CLEAR GOOD DATA
3738
3739 012666 033737 001532 001512 BIT BRLEV3,INTBIT ;TEST IF THIS BIT WILL INTERRUPT
3740 012674 001105 BNE 3$ ;:YES SKIP AND TRY NEXT BIT
3741 012676 012737 000000 001544 MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
(2)
(2) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3742 012714 012737 177777 001544 ;* MOV #-1,$TMDAT
(2)
(2) ;* MOV $TMDAT,@GRDAI ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDAI
3743 012732 012737 000000 001544 ;* MOV #0,$TMDAT ;CLEAR STATUS
3744
(1) ;* MOV $TMDAT,@GRSTAT ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRSTAT
3745
(1) ;* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
3746 012760 053737 001532 001544 ;* BIS BRLEV3,$TMDAT
3747
(1) ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDA TO DEVICE REG GRDIO

```

```

LPA DR11-K LOGIC TEST MD-11-CRLPF-B MACY11 30G(1063) 08-AUG-79 10:09 PAGE 8-10
CRLPFB.P11 08-AUG-79 10:08 T35 TEST THAT NON-INTERRUPT INPUT BITS DO NOT SET INPUT READY FLAG SEQ 0046
3748 012776 043737 001532 001544 BIC BRLEV3,$TMDAT ;CLEAR OUTPUT
3749 (1) :* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3750 (1) :* MOV @GRDIO,$TMDAT ;/READ DEVICE REG GRDIO,PUT DATA IN $TMDAT.
3751 013024 053737 001532 001544 BIS BRLEV3,$TMDAT
3752 (1) :* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3753 (1) :* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3754 013052 012737 000000 001544 MOV #0,$TMDAT ;CLEAR FLAG FROM DATA READY
3755 (1) :* MOV $TMDAT,@ ;/ PUT DATA FROM $TMDAT TO DEVICE REG
3756 ;SHOULD REMAIN SET VIA DIRECT SET SIDE
3757 (1) :* MOV @GRSTAT,$TMDAT ;/READ DEVICE REG GRSTAT,PUT DATA IN $TMDAT.
3758 013100 105737 001544 TSTB $TMDAT
3759 013104 100001 BPL 3$ ;;BR IF CLEAR
3760 013106 104011 ERROR 11 ;INPUT NON-INTERRUPT BIT SET INPUT READY
3761 ;?? DID OPERATOR GIVE CORRECT
3762 ;INPUT INTERRUPT BITS ??
3763
3764
3765 013110 006337 001532 3$: ASL BRLEV3 ;CHANGE BIT
3766 013114 001257 BNE 1$ ;BR IF NOT DONE
3767 (3) ;*****
3768 (3) ;*TEST 36 DETERMINE IF MORE DR11-K'S ARE TO BE TESTED
3769 (2) ;*****
3770 (1) 013116 000004 TST36: SCOPE
3771 (1) 013120 012737 000001 001160 MOV #1,$TIMES ;;DO 1 ITERATION
3772 013126 005737 001464 BYPASS: TST NBEXT ;TEST IF ANY
3773 013132 001415 BEQ 1$ ;BR IF NONE
3774 013134 032777 010000 165776 BIT #SW12,@SWR ;TEST BIT12 OF SWR
3775 013142 001024 BNE BYPAS1 ;INHIBIT TESTING NEXT DR11-K
3776 013144 162737 000010 001466 SUB #10,DRADD ;UPDATE DEVICE ADDRESS
3777 013152 062737 000010 001470 ADD #10,DRIV ;UPDATE DEVICE VECTOR
3778 013160 005337 001464 DEC NBEXT ;ANOTHER ONE ?
3779 013164 000413 BR BYPAS1 ;BR IF ANOTHER
3780 013166 013737 001452 001466 1$: MOV BASEBA,DRADD ;RELOAD ADDRESS
3781 013174 013737 001454 001470 MOV BASEIV,DRIV ;RELOAD VECTOR
3782 013202 013737 001462 001464 MOV NMBEXT,NBEXT ;RELOAD NUMBER
3783 (1) JMP $EOP ;DONE
3784 (1) 013210 000137 013224 BYPAS1: MOV #-1,R0
3785 (1) 013214 012700 177777 JMP RBEG2 ;TEST ANOTHER UNIT
3786 (1) 013220 000137 003346
3787 (1)
3788 (1) .SBTTL END OF PASS ROUTINE
3789 (1) ;*****
3790 (1) ;*INCREMENT THE PASS NUMBER ($PASS)
3791 (1) ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
3792 (1) ;*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
3793 (1) ;*IF THERES A MONITOR GO TO IT
3794 (1) ;*IF THERE ISN'T JUMP TO BYPAS1
3795 (1)
3796 (1) $EOP:
3797 (1) 013224 SCOPE
3798 (1) 013224 000004

```

(1)	013226	005037	001102		CLR	\$STNM	::ZERO THE TEST NUMBER
(1)	013232	005037	001160		CLR	\$TIMES	::ZERO THE NUMBER OF ITERATIONS
(1)	013236	005237	001176		INC	\$PASS	::INCREMENT THE PASS NUMBER
(*)	013242	042737	100000	001176	BIC	#100000,\$PASS	::DON'T ALLOW A NEG. NUMBER
(1)	013250	005327			DEC	(PC)+	::LOOP?
(1)	013252	000001			\$EOPCT: .WORD	1	
(1)	013254	003022			BGT	\$DOAGN	::YES
(1)	013256	012737			MOV	(PC)+,@(PC)+	::RESTORE COUNTER
(1)	013260	000001			\$ENDCT: .WORD	1	
(1)	013262	013252			\$EOPCT		
(1)	013264	104401	013331		TYPE	\$ENDMG	::TYPE 'END PASS #'
(2)	013270	013746	001176		MOV	\$PASS,-(SP)	::SAVE \$PASS FOR TYPEOUT
(2)	013274	104405			TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN
(1)	013276	104401	013326		TYPE	\$ENULL	::TYPE A NULL CHARACTER
(1)	013302	013700	000042		\$GET42: MOV	@42,R0	::GET MONITOR ADDRESS
(1)	013306	001405			BEQ	\$DOAGN	::BRANCH IF NO MONITOR
(1)	013310	000005			RESET		::CLEAR THE WORLD
(1)	013312	004710			\$ENDAD: JSR	PC,(R0)	::GO TO MONITOR
(1)	013314	000240			NOP		::SAVE ROOM
(1)	013316	000240			NOP		::FOR
(1)	013320	000240			NOP		::ACT11
(1)	013322				\$DOAGN:		
(1)	013322	000137			JMP	@(PC)+	::RETURN
(1)	013324	013214			\$RTNAD: .WORD	BYPAS1	
(1)	013326	377	377	000	\$ENULL: .BYTE	-1,-1,0	::NULL CHARACTER STRING
(1)	013331	015	042412	042116	\$ENDMG: .ASCIIZ	<15><12>/END PASS #/	
(1)	013336	050040	051501	020123			
(1)	013344	000043					

3784 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

(1)							
(2)							
(1)							::*****
(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)	013346				\$TYPDS:		
(3)	013346	010046			MOV	R0,-(SP)	::PUSH R0 ON STACK
(3)	013350	010146			MOV	R1,-(SP)	::PUSH R1 ON STACK
(3)	013352	010246			MOV	R2,-(SP)	::PUSH R2 ON STACK
(3)	013354	010346			MOV	R3,-(SP)	::PUSH R3 ON STACK
(3)	013356	010546			MOV	R5,-(SP)	::PUSH R5 ON STACK
(1)	013360	012746	020200		MOV	#20200,-(SP)	::SET BLANK SWITCH AND SIGN
(1)	013364	016605	000020		MOV	20(SP),R5	::GET THE INPUT NUMBER
(1)	013370	100004			BPL	1\$::BR IF INPUT IS POS.
(1)	013372	005405			NEG	R5	::MAKE THE BINARY NUMBER POS.
(1)	013374	112766	000055	000001	MOVB	#'-,1(SP)	::MAKE THE ASCII NUMBER NEG.
(1)	013402	005000			1\$: CLR	R0	::ZERO THE CONSTANTS INDEX
(1)	013404	012703	013562		MOV	#\$DBLK,R3	::SETUP THE OUTPUT POINTER
(1)	013410	112723	000040		MOVB	#',(R3)+	::SET THE FIRST CHARACTER TO A BLANK
(1)	013414	005002			2\$: CLR	R2	::CLEAR THE BCD NUMBER
(1)	013416	016001	013552		MOV	\$DTBL(R0),R1	::GET THE CONSTANT


```

(1) 013422 160105      3$: SUB R1,R5      ;;FORM THIS BCD DIGIT
(1) 013424 002402      BLT 4$          ;;BR IF DONE
(1) 013426 005202      INC R2          ;;INCREASE THE BCD DIGIT BY 1
(1) 013430 000774      BR 3$
(1) 013432 060105      4$: ADD R1,R5      ;;ADD BACK THE CONSTANT
(1) 013434 005702      TST R2          ;;CHECK IF BCD DIGIT=0
(1) 013436 001002      BNE 5$          ;;FALL THROUGH IF 0
(1) 013440 105716      TSTB (SP)       ;;STILL DOING LEADING 0'S?
(1) 013442 100407      BMI 7$          ;;BR IF YES
(1) 013444 106316      5$: ASLB (SP)       ;;MSD?
(1) 013446 103003      BCC 6$          ;;BR IF NO
(1) 013450 116663 000001 177777  MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
(1) 013456 052702 000060 6$: BIS #'0,R2      ;;MAKE THE BCD DIGIT ASCII
(1) 013462 052702 00004C 7$: BIS #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 013466 110223      MOVB R2,(R3)+   ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 013470 005720      TST (R0)+      ;;JUST INCREMENTING
(1) 013472 020027 000010  CMP R0,#10      ;;CHECK THE TABLE INDEX
(1) 013476 002746      BLT 2$          ;;GO DO THE NEXT DIGIT
(1) 013500 003002      BGT 8$          ;;GO TO EXIT
(1) 013502 010502      MOV R5,R2      ;;GET THE LSD
(1) 013504 000764      BR 6$          ;;GO CHANGE TO ASCII
(1) 013506 105726      8$: TSTB (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 013510 100003      BPL 9$          ;;BR IF NO
(1) 013512 116663 177777 177776  MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 013520 105013      9$: CLRB (R3)     ;;SET THE TERMINATOR
(3) 013522 012605      MOV (SP)+,R5   ;;POP STACK INTO R5
(3) 013524 012603      MOV (SP)+,R3   ;;POP STACK INTO R3
(3) 013526 012602      MOV (SP)+,R2   ;;POP STACK INTO R2
(3) 013530 012601      MOV (SP)+,R1   ;;POP STACK INTO R1
(3) 013532 012600      MOV (SP)+,R0   ;;POP STACK INTO R0
(1) 013534 104401 013562  TYPE $DBLK      ;;NOW TYPE THE NUMBER
(1) 013540 016666 000002 000004  MOV 2(SP),4(SP) ;;ADJUST THE STACK
(1) 013546 012616      MOV (SP)+,(SP)
(1) 013550 000002      RTI            ;;RETURN TO USER
(1) 013552 023420      $DTBL: 10000.
(1) 013554 001750      1000.
(1) 013556 000144      100.
(1) 013560 000012      10.
(1) 013562 000004      $DBLK: .BLKW 4

3785
3786 ; MISC. EXTERNAL LOGIC TEST
3787 013572 012706 001100  EXTST: MOV #STACK,SP
3788 013576 004737 003412  JSR PC,SETADD ;SET UP ADDRESS AND VECTOR
3789 013602 012737 040000 013752 2$: MOV #BIT14,EXTCNT ;LOAD COUNT
3790 013610 012737 000000 001544 1$: MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
(1) 013610 012737 000000 001544 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(2) ;* MOV #125252,$TMDAT ;LOAD OUTPUT
3791 013626 012737 125252 001544 ;*
3792 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(1) ;*
3793 ;* MOV @GRDAI,EXTTMP ;/READ DEVICE REG GRDAI,PUT DATA IN EXTTMP.
(1) ;*
3794 ;* MOV EXTTMP,@GRDAI ;/ PUT DATA FROM EXTTMP TO DEVICE REG GRDAI
(1) ;* MOV #0,$TMDAT ;CLEAR OUTPUT REGISTER
3795 013664 012737 000000 001544

```

```

(2)
(2)
3796 013702 012737 052525 001544 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
3797 MOV #52525,$TMDAT ;LOAD OUTPUT
(1)
3798 ;* MOV $TMDAT,@GRDIO ;/ PUT DATA FROM $TMDAT TO DEVICE REG GRDIO
(1)
3799 ;* MOV @GRDAI,EXTTMP ;/READ DEVICE REG GRDAI,PUT DATA IN EXTMP.
(1)
3800 013740 005337 013752 ;* MOV EXTTMP,@GRDAI ;/ PUT DATA FROM EXTTMP TO DEVICE REG GRDAI
3801 013744 001321 DEC EXTCNT ;FINISHED COUNT
3802 013746 000715 BNE 1$
3803 BR 2$ ;LOOP
3804 013750 000000 EXTTMP: 0
3805 013752 000000 EXTCNT: 0
3806
3807
3808 013754 005015 052502 020123 BUSTRP: .ASCIZ <15><12>/BUS TIME-OUT ON SELECTED DR11K/
013762 044524 042515 047455
013770 052125 047440 020116
013776 042523 042514 052103
014004 042105 042040 030522
014012 045461 000
3809 014015 123 052105 051440 SWNLB: .ASCIZ /SET SWITCH REGISTER BITS EQUAL TO THE NON-LATCHING INPUT BITS/
014022 044527 041524 020110
014030 042522 044507 052123
014036 051105 041040 052111
014044 020123 050505 040525
014052 020114 047524 052040
014060 042510 047040 047117
014066 046055 052101 044103
014074 047111 020107 047111
014102 052520 020124 044502
014110 051524 000
3810 014113 123 052105 051440 SWINTB: .ASCIZ /SET SWITCH REGISTER BITS EQUAL TO THE INTERRUPTING INPUT BITS/
014120 044527 041524 020110
014126 042522 044507 052123
014134 051105 041040 052111
014142 020123 050505 040525
014150 020114 047524 052040
014156 042510 044440 052116
014164 051105 052522 052120
014172 047111 020107 047111
014200 052520 020124 044502
014206 051524 000
3811 014211 123 052105 051440 SWPOSB: .ASCIZ /SET SWITCH REGISTER BITS 15-12 EQUAL TO POSITIVE INPUT BITS/
014216 044527 041524 020110
014224 042522 044507 052123
014232 051105 041040 052111
014240 020123 032461 030455
014246 020062 050505 040525
014254 020114 047524 050040
014262 051517 052111 053111
014270 020105 047111 052520
014276 020124 044502 051524
014304 000
  
```

3812	014305	123	052105	051440	SWTRAB: .ASCIZ /SET SWITCH REGISTER BITS 15-12 EQUAL TO TRANSITION INPUT BITS/
	014312	044527	041524	020110	
	014320	042522	044507	052123	
	014326	051105	041040	052111	
	014334	020123	032461	030455	
	014342	020062	050505	040525	
	014350	020114	047524	052040	
	014356	040522	051516	052111	
	014364	047511	020116	047111	
	014372	052520	020124	044502	
	014400	051524	000		
3813	014403	123	052105	051440	SWDPOB: .ASCIZ /SET SWITCH REGISTER WITH THE DESIRED PROGRAM OPTIONS/
	014410	044527	041524	020110	
	014416	042522	044507	052123	
	014424	051105	053440	052111	
	014432	020110	044124	020105	
	014440	042504	044523	042522	
	014446	020104	051120	043517	
	014454	040522	020115	050117	
	014462	044524	047117	000123	
3814	014470	005015	042504	051120	DEPCNT: .ASCIZ <15><12>/DEPRESS CONT./<15><12>
	014476	051505	020123	047503	
	014504	052116	006456	000012	
3815	014512	052123	052101	051525	EM1: .ASCIZ /STATUS REGISTER IN ERROR/
	014520	051040	043505	051511	
	014526	042524	020122	047111	
	014534	042440	051122	051117	
	014542	000			
3816	014543	111	050116	052125	EM2: .ASCIZ /INPUT REGISTER IN ERROR/
	014550	051040	043505	051511	
	014556	042524	020122	047111	
	014564	042440	051122	051117	
	014572	000			
3817	014573	117	052125	052520	EM3: .ASCIZ /OUTPUT REGISTER IN ERROR/
	014600	020124	042522	044507	
	014606	052123	051105	044440	
	014614	020116	051105	047522	
	014622	000122			
3818	014624	047111	052520	020124	EM4: .ASCIZ /INPUT FAILED TO INTERRUPT/
	014632	040506	046111	042105	
	014640	052040	020117	047111	
	014646	042524	051122	050125	
	014654	000124			
3819	014656	052517	050124	052125	EM5: .ASCIZ /OUTPUT FAILED TO INTERRUPT/
	014664	043040	044501	042514	
	014672	020104	047524	044440	
	014700	052116	051105	052522	
	014706	052120	000		
3820	014711	125	042516	050130	EM6: .ASCIZ /UNEXPECTED INTERRUPT/
	014716	041505	042524	020104	
	014724	047111	042524	051122	
	014732	050125	000124		
3821	014736	050117	051105	052101	EM7: .ASCIZ /OPERATOR INTERVENTION ERROR/
	014744	051117	044440	052116	
	014752	051105	042526	052116	
	014760	047511	020116	051105	

3822	014766	047522	000122	051122	EM10:	.ASCIZ	/INTERRUPT INPUT BIT FAILED TO SET INPUT READY FLAG/
	014772	047111	042524	047111			
	015000	050125	020124	044502			
	015006	052520	020124	046111			
	015014	020124	040506	020117			
	015022	042105	052040	047111			
	015030	042523	020124	042522			
	015036	052520	020124	046106			
	015044	042101	020131	000			
	015052	043501	000				
3823	015055	116	047117	044455	EM11:	.ASCIZ	/NON-INTERRUPT INPUT BIT SET INPUT READY FLAG/
	015062	052116	051105	052522			
	015070	052120	044440	050116			
	015076	052125	041040	052111			
	015104	051440	052105	044440			
	015112	050116	052125	051040			
	015120	040505	054504	043040			
	015126	040514	000107				
3824							
3825	015132	051105	050122	020103	DH1:	.ASCIZ	/ERRPC DRADD TSTNUM STATUS EXPECTED/
	015140	020040	051104	042101			
	015146	004504	051524	047124			
	015154	046525	020040	051440			
	015162	040524	052524	020123			
	015170	042440	050130	041505			
	015176	042524	000104				
3826	015202	051105	050122	020103	DH2:	.ASCIZ	/ERRPC DRADD TSTNUM INPUT EXPECTED/
	015210	020040	051104	042101			
	015216	004504	051524	047124			
	015224	046525	044411	050116			
	015232	052125	020040	042440			
	015240	050130	041505	042524			
	015246	000104					
3827	015250	051105	050122	020103	DH3:	.ASCIZ	/ERRPC DRADD TSTNUM OUTPUT EXPECTED/
	015256	020040	051104	042101			
	015264	004504	051524	047124			
	015272	046525	047411	052125			
	015300	052520	020124	042440			
	015306	050130	041505	042524			
	015314	000104					
3828	015316	051105	050122	020103	DH4:	.ASCIZ	/ERRPC DRADD TSTNUM/
	015324	020040	051104	042101			
	015332	004504	051524	047124			
	015340	046525	000				
3829	015343	105	051122	041520	DH10:	.ASCIZ	/ERRPC DRADD TSTNUM STATUS EXPECT INPUT BIT/
	015350	020040	042040	040522			
	015356	042104	052011	052123			
	015364	052516	004515	052123			
	015372	052101	051525	020040			
	015400	054105	042520	052103			
	015406	020040	047111	052520			
	015414	020124	044502	000124			
3830							
3831	015422	001116	001466	015502	DT1:	.EVEN	\$ERRPC,DRADD,TSTNUM,\$BDDAT,\$GDDAT,0
	015430	001126	001124	000000			
3832	015436	001116	001466	015502	DT4:	\$ERRPC,DRADD,TSTNUM,0	


```

(1) 016114 032777 001000 163016 3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?
(1) 016122 001402 BEQ 4$ ;;BR IF NO
(1) 016124 013716 001110 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
(1) 016130 005737 001162 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
(1) 016134 001402 BEQ 5$ ;;BR IF NONE
(1) 016136 013716 001162 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 016142 5$: CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
(1) 016142 022737 013312 000042 BNE 6$ ;;BRANCH IF NO
(1) 016150 001001 HALT ;;YES
(1) 016152 000000 6$: RTI ;;RETURN
(1) 016154 000002
    
```

3839
3840

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

(1) *****
(2) *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) 016156 $ERRTYP:
(1) 016156 104401 001165 TYPE $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 016162 010046 MOV RO,-(SP) ;;SAVE RO
(1) 016164 005000 CLR RO ;;PICKUP THE ITEM INDEX
(1) 016166 153700 001114 BISB @#$ITEMB,RO
(1) 016172 001004 BNE 1$ ;;IF ITEM NUMBER IS ZERO, JUST
(1) 016174 013746 001116 MOV $ERRPC,-(SP) ;;TYPE THE PC OF THE ERROR
(2) 016200 104402 TYPOC ;;SAVE $ERRPC FOR TYPEOUT
(1) 016202 000426 BR 6$ ;;ERROR ADDRESS
(1) 016204 005300 1$: DEC RO ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 016206 006300 ASL RO ;;GFT OUT
(1) 016210 006300 ASL RO ;;ADJUST THE INDEX SO THAT IT WILL
(1) 016212 006300 ASL RO ;; WORK FOR THE ERROR TABLE
(1) 016214 062700 001252 ADD #ERRTB,RO ;;FORM TABLE POINTER
(1) 016220 012037 016230 MOV (RO)+,2$ ;;PICKUP 'ERROR MESSAGE' POINTER
(1) 016224 001404 BEQ 3$ ;;SKIP TYPEOUT IF NO POINTER
(1) 016226 104401 TYPE ;;TYPE THE 'ERROR MESSAGE'
(1) 016230 000000 2$: .WORD 0 ;;'ERROR MESSAGE' POINTER GOES HERE
(1) 016232 104401 001165 TYPE $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 016236 012037 016246 3$: MOV (RO)+,4$ ;;PICKUP 'DATA HEADER' POINTER
(1) 016242 001404 BEQ 5$ ;;SKIP TYPEOUT IF 0
(1) 016244 104401 TYPE ;;TYPE THE 'DATA HEADER'
(1) 016246 000000 4$: .WORD 0 ;;'DATA HEADER' POINTER GOES HERE
(1) 016250 104401 001165 TYPE $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 016254 011000 5$: MOV (RO),RO ;;PICKUP 'DATA TABLE' POINTER
(1) 016256 001004 BNE 7$ ;;GO TYPE THE DATA
(1) 016260 012600 6$: MOV (SP)+,RO ;;RESTORE RO
(1) 016262 104401 001165 TYPE $CRLF ;;'CARRIAGE RETURN' & 'LINE FEED'
(1) 016266 000207 7$: RTS PC ;;RETURN
(2) 016270 013046 MOV @ (RO)+,-(SP) ;;SAVE @ (RO)+ FOR TYPEOUT
(2) 016272 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 016274 005710 TS+ (RO) ;;IS THERE ANOTHER NUMBER?
(1) 016276 00177C BEQ 6$ ;;BR IF NO
    
```



```

(1) 016440 100016          BPL      7$          ;;BR IF NO
(1) 016442 042703 177770  BIC      #177770,R3  ;;GET RID OF JUNK
(1) 016446 001002          BNE      4$          ;;TEST FOR 0
(1) 016450 005704          TST      R4          ;;SUPPRESS THIS 0?
(1) 016452 001403          BEQ      5$          ;;BR IF YES
(1) 016454 005204          4$: INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
(1) 016456 052703 000060  BIS      #'0,R3     ;;MAKE THIS DIGIT ASCII
(1) 016462 052703 000040  5$: BIS      #' ,R3   ;;MAKE ASCII IF NOT ALREADY
(1) 016466 110337 016532  MOVB     R3,8$      ;;SAVE FOR TYPING
(1) 016472 104401 016532  TYPE     ,8$       ;;GO TYPE THIS DIGIT
(1) 016476 105337 016534  7$: DECB    $OCNT    ;;COUNT BY 1
(1) 016502 003347          BGT      2$          ;;BR IF MORE TO DO
(1) 016504 002402          BLT      6$          ;;BR IF DONE
(1) 016506 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 016510 000744          BR       2$          ;;GO DO THE LAST DIGIT
(1) 016512 012605          6$: MOV     (SP)+,R5  ;;RESTORE R5
(1) 016514 012604          MOV     (SP)+,R4  ;;RESTORE R4
(1) 016516 012603          MOV     (SP)+,R3  ;;RESTORE R3
(1) 016520 016666 000002 000004 MOV     2(SP),4(SP) ;;SET THE STACK FOR RETURNING
(1) 016526 012616          MOV     (SP)+,(SP)
(1) 016530 000002          RTI          ;;RETURN
(1) 016532 000          8$: .BYTE   0      ;;STORAGE FOR ASCII DIGIT
(1) 016533 000          .BYTE   0      ;;TERMINATOR FOR TYPE ROUTINE
(1) 016534 000          $OCNT: .BYTE   0  ;;OCTAL DIGIT COUNTER
(1) 016535 000          $OFILL: .BYTE  0  ;;ZERO FILL SWITCH
(1) 016536 000000          $OMODE: .WORD  0  ;;NUMBER OF DIGITS TO TYPE
    
```

3843
3844

.SBITL POWER DOWN AND UP ROUTINES

```

(1)
(2)
(1)
(1) 016540 012737 016704 000024 $PWRDN: MOV     # $ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1) 016546 012737 000340 000026 MOV     #340,@#PWRVEC+2 ;;PRIO:7
(3) 016554 010046          MOV     R0,-(SP)    ;;PUSH R0 ON STACK
(3) 016556 010146          MOV     R1,-(SP)    ;;PUSH R1 ON STACK
(3) 016560 010246          MOV     R2,-(SP)    ;;PUSH R2 ON STACK
(3) 016562 010346          MOV     R3,-(SP)    ;;PUSH R3 ON STACK
(3) 016564 010446          MOV     R4,-(SP)    ;;PUSH R4 ON STACK
(3) 016566 010546          MOV     R5,-(SP)    ;;PUSH R5 ON STACK
(3) 016570 017746 162344  MOV     @SWR,-(SP)   ;;PUSH @SWR ON STACK
(1) 016574 010637 016710  MOV     SP,$SAVR6   ;;SAVE SP
(1) 016600 012737 016612 000024 MOV     # $PWRUP,@#PWRVEC ;;SET UP VECTOR
(1) 016606 000000          HALT
(1) 016610 000776          BR      .-2        ;;HANG UP
(1)
(2)
(1)
(1) 016612 012737 016704 000024 $PWRUP: MOV     # $ILLUP,@#PWRVEC ;;SET FOR FAST DOWN!
(1) 016620 013706 016710  MOV     $SAVR6,SP   ;;GET SP
(1) 016624 005037 016710  CLR     $SAVR6      ;;WAIT LOOP FOR THE TTY
(1) 016630 005237 016710  1$: INC     $SAVR6   ;;WAIT FOR THE INC
(1) 016634 001375          BNE     1$         ;;OF WORD
(3) 016636 012677 162276  MOV     (SP)+,@SWR  ;;POP STACK INTO @SWR
(3) 016642 012605          MOV     (SP)+,R5   ;;POP STACK INTO R5
(3) 016644 012604          MOV     (SP)+,R4   ;;POP STACK INTO R4
(3) 016646 012603          MOV     (SP)+,R3   ;;POP STACK INTO R3
    
```

```

(3) 016650 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
(3) 016652 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
(3) 016654 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
(1) 016656 012737 016540 000024  MOV      #SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 016664 012737 000340 000026  MOV      #340,@#PWRVEC+2 ;;PRIO:7
(1) 016672 104401      TYPE     ;;REPORT THE POWER FAILURE
(1) 016674 016712  $PWRMG: .WORD  PWRMSG      ;;POWER FAIL MESSAGE POINTER
(1) 016676 012716      MOV      (PC)+,(SP)    ;;RESTART AT IOTEST
(1) 016700 003320  $PWRAD: .WORD  IOTEST    ;;RESTART ADDRESS
(1) 016702 000002      RTI
(1) 016704 000000  $ILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
(1) 016706 000776      BR      .-2           ;; BEFORE THE POWER DOWN WAS COMPLETE
(1) 016710 000000  $SAVR6: 0            ;;PUT THE SP HERE
3845 016712 005015 042522 052123  PWRMSG: .ASCIZ <15><12>/RESTARTING AFTER A POWER FAILURE/<15><12>
      016720 051101 044524 043516
      016726 040440 052106 051105
      016734 040440 050040 053517
      016742 051105 043040 044501
      016750 052514 042522 005015
      016756      000
  
```

```

3846 016760      .EVEN
3847
3848 .SBTTL TYPE ROUTINE
  
```

```

(1) *****
(2) *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) 016760 105737 001157  $TYPE: TSTB  $TPFLG      ;;IS THERE A TERMINAL?
(1) 016764 100002      BPL      1$          ;;BR IF YES
(1) 016766 000000      HALT     ;;HALT HERE IF NO TERMINAL
(1) 016770 000430      BR      3$          ;;LEAVE
(1) 016772 010046 1$:      MOV      R0,-(SP)    ;;SAVE R0
(1) 016774 017600 000000  MOV      @2(SP),R0    ;;GET ADDRESS OF ASCIZ STRING
(1) 017000 122737 000000 001210  CMPB    #APTENV,$ENV  ;;RUNNING IN APT MODE
(1) 017006 001011      BNE     62$         ;;NO,GO CHECK FOR APT CONSOLE
(1) 017010 132737 000100 001211  BITB    #APTPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1) 017016 001405      BEQ     62$         ;;NO,GO CHECK FOR CONSOLE
(1) 017020 010037 017030  MOV      R0,61$     ;;SETUP MESSAGE ADDRESS FOR APT
(1) 017024 004737 020254  JSR     PC,$ATY3    ;;SPOOL MESSAGE TO APT
(1) 017030 000000 61$:    .WORD  0          ;;MESSAGE ADDRESS
(1) 017032 132737 000040 001211 62$:    BITB    #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(1) 017040 001003      BNE     60$         ;;YES,SKIP TYPE OUT
(1) 017042 112046 2$:      MOVB    (R0)+,-(SP)  ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 017044 001005      BNE     4$          ;;BR IF IT ISN'T THE TERMINATOR
  
```

```
(1) 017046 005726          TST      (SP)+          ;; IF TERMINATOR POP IT OFF THE STACK
(1) 017050 012600          60$: MOV      (SP)+,R0      ;; RESTORE R0
(1) 017052 062716 000002    3$:  ADD      #2,(SP)      ;; ADJUST RETURN PC
(1) 017056 000002          RTI          ;; RETURN
(1) 017060 122716 000011    4$:  CMPB     #HT,(SP)      ;; BRANCH IF <HT>
(1) 017064 001430          BEQ      8$              ;;
(1) 017066 122716 000200    CMPB     #CRLF,(SP)      ;; BRANCH IF NOT <CRLF>
(1) 017072 001006          BNE      5$              ;;
(1) 017074 005726          TST      (SP)+          ;; POP <CR><LF> EQUIV
(1) 017076 104401          TYPE     $CRLF          ;; TYPE A CR AND LF
(1) 017100 001165          CLR      $CHARCNT       ;; CLEAR CHARACTER COUNT
(1) 017102 105037 017236    CLRB     2$              ;; GET NEXT CHARACTER
(1) 017106 000755          BR       2$              ;;
(1) 017110 004737 017172    5$:  JSR      PC,$TYPEPC     ;; GO TYPE THIS CHARACTER
(1) 017114 123726 001156    6$:  CMPB     $FILLC,(SP)+  ;; IS IT TIME FOR FILLER CHARS.?
(1) 017120 001350          BNE      2$              ;; IF NO GO GET NEXT CHAR.
(1) 017122 013746 001154    MOV      $NULL,-(SP)    ;; GET # OF FILLER CHARS. NEEDED
(1)                                ;; AND THE NULL CHAR.
(1) 017126 105366 000001    7$:  DECB     1(SP)         ;; DOES A NULL NEED TO BE TYPED?
(1) 017132 002770          BLT      6$              ;; BR IF NO--GO POP THE NULL OFF OF STACK
(1) 017134 004737 017172    JSR      PC,$TYPEPC     ;; GO TYPE A NULL
(1) 017140 105337 017236    DECB     $CHARCNT       ;; DO NOT COUNT AS A COUNT
(1) 017144 000770          BR       7$              ;; LOOP
(1)                                ;HORIZONTAL TAB PROCESSOR
(1)
(1) 017146 112716 000040    8$:  MOVB     #' ,(SP)      ;; REPLACE TAB WITH SPACE
(1) 017152 004737 017172    9$:  JSR      PC,$TYPEPC     ;; TYPE A SPACE
(1) 017156 132737 000007 017236  BITB     #7,$CHARCNT     ;; BRANCH IF NOT AT
(1) 017164 001372          BNE      9$              ;; TAB STOP
(1) 017166 005726          TST      (SP)+          ;; POP SPACE OFF STACK
(1) 017170 000724          BR       2$              ;; GET NEXT CHARACTER
(1) 017172 105777 161752    $TYPEPC: TSTB     @STPS     ;; WAIT UNTIL PRINTER IS READY
(1) 017176 100375          BPL      $TYPEPC        ;;
(1) 017200 116677 000002 161744  MOVB     2(SP),@STPB     ;; LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 017206 122766 000015 000002  CMPB     #CR,2(SP)      ;; IS CHARACTER A CARRIAGE RETURN?
(1) 017214 001003          BNE      1$              ;; BRANCH IF NO
(1) 017216 105037 017236    CLRB     $CHARCNT       ;; YES--CLEAR CHARACTER COUNT
(1) 017222 000406          BR       $TYPEPC        ;; EXIT
(1) 017224 122766 000012 000002  1$:  CMPB     #LF,2(SP)     ;; IS CHARACTER A LINE FEED?
(1) 017232 001402          BEQ      $TYPEPC        ;; BRANCH IF YES
(1) 017234 105227          INCB     (PC)+          ;; COUNT THE CHARACTER
(1) 017236 000000          $CHARCNT: .WORD 0      ;; CHARACTER COUNT STORAGE
(1) 017240 000207          $TYPEPC: RTS      PC
(1)
(1) 3849 .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
(1)
(1)  (2)  ;*****
(1)  (1)  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)  (1)  ;*CHANGE IT TO BINARY.
(1)  (1)  ;*CALL:
(1)  (1)  ;*   RDOCT          ;; READ AN OCTAL NUMBER
(1)  (1)  ;*   RETURN HERE    ;; LOW ORDER BITS ARE ON TOP OF THE STACK
(1)  (1)  ;*                   ;; HIGH ORDER BITS ARE IN $HI OCT
(1)  (1)
(1) 017242 011646          $RDOCT: MOV      (SP),-(SP)  ;; PROVIDE SPACE FOR THE
```



```
(1) 020106 104401 001165      TYPE      ,SCLRF      ;;TYPE A 'CR' & 'LF'  
(1) 020112 104401 020202      TYPE      $TTYIN     ;;TYPE THE INPUT STRING  
(1) 020116 000717      BR        2$         ;;GO PICKUP ANOTHER CHACTER  
(1) 020120 104401 001164      4$:      TYPE      $QUES     ;;TYPE A '?'  
(1) 020124 000712      BR        1$         ;;CLEAR THE BUFFER AND LOOP  
(1) 020126 111337 020200      3$:      MOVVB     (R3),9$    ;;ECHO THE CHARACTER  
(1) 020132 104401 020200      TYPE      ,9$         ;;  
(1) 020136 122723 000015      CMPB     #15,(R3)+   ;;CHECK FOR RETURN  
(1) 020142 001305      BNE      2$         ;;LOOP IF NOT RETURN  
(1) 020144 105063 177777      CLRB     -1(R3)     ;;CLEAR RETURN (THE 15)  
(1) 020150 104401 001166      TYPE      ,SLF      ;;TYPE A LINE FEED  
(1) 020154 005726      TST      (SP)+      ;;CLEAN RUBOUT KEY FROM THE STACK  
(1) 020156 012603      MOV      (SP)+,R3    ;;RESTORE R3  
(1) 020160 011646      MOV      (SP),-(SP)  ;;ADJUST THE STACK AND PUT ADDRESS OF THE  
(1) 020162 016666 000004 000002      MOV      4(SP),2(SP) ;;FIRST ASCII CHARACTER ON IT  
(1) 020170 012766 020202 000004      MOV      #TTYIN,4(SP)  
(1) 020176 000002      RTI              ;;RETURN  
(1) 020200 000      9$:      .BYTE     0         ;;STORAGE FOR ASCII CHAR. TO TYPE  
(1) 020201 000      .BYTE     0         ;;TERMINATOR  
(1) 020202 000007      $TTYIN: .BLKB     7         ;;RESERVE 7 BYTES FOR TTY INPUT  
(1) 020211 136 006525 000012      $CNTLU: .ASCIZ  /^U/<15><12> ;;CONTROL 'U'  
(1) 020216 043536 005015 000      $CNTLG: .ASCIZ  /^G/<15><12> ;;CONTROL 'G'  
(1) 020223 015 051412 051127      $MSWR:  .ASCIZ  <15><12>/SWR - /  
(1) 020230 036440 000040      $MNEW:  .ASCIZ  / NEW - /  
(1) 020234 020040 042516 020127  
(1) 020242 020075 000  
(1) 020246 .EVEN
```

3851
3852
3853

.SBTTL APT COMMUNICATIONS ROUTINE

```
(1) (2) *****  
(1) 020246 112737 000001 020512 $ATY1: MOVVB #1,$FFLG ;;TO REPORT FATAL ERROR  
(1) 020254 112737 000001 020510 $ATY3: MOVVB #1,$MFLG ;;TO TYPE A MESSAGE  
(1) 020262 000403      BR        $ATYC  
(1) 020264 112737 000001 020512 $ATY4: MOVVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR  
(1) 020272 $ATYC:  
(3) 020272 010046      MOV      R0,-(SP)   ;;PUSH R0 ON STACK  
(3) 020274 010146      MOV      R1,-(SP)   ;;PUSH R1 ON STACK  
(1) 020276 105737 020510      TSTB     $MFLG     ;;SHOULD TYPE A MESSAGE?  
(1) 020302 001450      BEQ      5$         ;;IF NOT: BR  
(1) 020304 122737 000001 001210      CMPB     #APTENV,$ENV ;;OPERATING UNDER APT?  
(1) 020312 001031      BNE      3$         ;;IF NOT: BR  
(1) 020314 132737 000100 001211      BITB     #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?  
(1) 020322 001425      BEQ      3$         ;;IF NOT: BR  
(1) 020324 017600 000004      MOV      @4(SP),R0  ;;GET MESSAGE ADDR.  
(1) 020330 062766 000002 000004      ADD      #2,4(SP)   ;;BUMP RETURN ADDR.  
(1) 020336 005737 001170      1$:      TST      $MSGTYPE  ;;SEE IF DONE W/ LAST XMISSION?  
(1) 020342 001375      BNF      1$         ;;IF NOT: WAIT  
(1) 020344 010637 001204      MOV      R0,$MSGAD ;;PUT ADDR IN MAILBOX  
(1) 020350 105720      2$:      TSTB     (R0)+     ;;FIND END OF MESSAGE  
(1) 020352 001376      BNE      2$         ;;  
(1) 020354 163700 001204      SUB      $MSGAD,R0  ;;SUB START OF MESSAGE  
(1) 020360 006200      ASR      R0         ;;GET MESSAGE LNGTH IN WORDS  
(1) 020362 010037 001206      MOV      R0,$MSGGLT ;;PUT LENGTH IN MAILBOX  
(1) 020366 012737 000004 001170      MOV      #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
```



```

(3) ;*BY THE 'TRAP' INSTRUCTION.
(3)
(3) : ROUTINE
(3) :-----
(3) $TRPAD: .WORD $TRAP2
(3) 020550 020536 $TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
(3) 020552 016760 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3) 020554 016336 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3) 020556 016312 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
(3) 020560 016352 $TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
(3) 020562 013346
(1)
(3) 020564 017414 $GTSWR ;;CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING
(1)
(3) 020566 017344 $CKSWR ;;CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
(3) 020570 017626 $RDCHR ;;CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
(3) 020572 017746 $RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
(3) 020574 017242 $RDOCT ;;CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
3855
3856 020576 000000 BUFFER: 0 ;10 WORD BUFFER FOR THE COULTER INTERFACE TEST
3857
(2)
(2) ;*
(2) ;*THIS SUB CODE IS USED TO INITIALIZE THE LPA-11
(2) ;*FIRST WE WILL LOAD MICROCODE INTO KMC-11
(2) ;*NEXT WE WILL INIT BOTH UPROCESSORS
(2) ;*THEN WE WILL LOAD DEVICE TABLE IN SLAVE UP.
(2) ;*THE ORDER OF LOAD IS DETERMINED BY THE USER.
(2)
(2) ;*
(2) ;* CALL= JSR R5,$LPAI
(2) ;* .WORD 0 ;ADDR. OF DEVICE ADDRESS.
(2) ;* ROUTINES REQUIRED: .LOADLP
(2) ;* PROGRAMS REQUIRED: DRLPX2
(2)
(2) ;*
(2) ;* ;RETURNS WITH $AERR=1 IF SLAVE
(2) ;* ;MICRO SAYS AN ADDR. DOES NOT EXSIST. IN THE LIST.
(2) ;*
(2) $LPAI:
(2) 020600 MOV 4,-(SP)
(2) 020600 013746 000004
(2) 020604 000413 BR 31$
(2) ;FIELD DOES NOT HAVE A BUS SWITCH TO
(2) ;WORRY ABOUT,SO WE WILL UNCONDITIONALLY
(2) ;BRANCH AROUND THE NEXT CODE THAT
(2) ;WORKS BASED ON A BUS SWITCH.
(2) ;CODE LEFT IN HERE FOR IN HOUSE
(2) ;PERSONAL WHO MAY PATCH THIS BRANCH
(2) ;INSTRUCTION TO A <NOP> OCTAL <240>
(2) ;IN ORDER TO RUN PROGRAM WITH A SWITCH.
(2)
(2) ;NOTE THIS 'SWITCH' IS A PIECE OF INHOUSE
(2) ;TEST EQUIPMENT ONLY IT CONNECTS
(2) ;THE UNIBUS TO THE I/O BUS FOR
(2) ;CERTAIN TESTING.
(2)
(2) 020606 012737 020632 000004 MOV #30$,4
(2) 020614 005237 170000 INC 170000
(3) 020620 104401 020626 TYPE ,65$ ;;TYPE ASCIZ STRING
(3) 020624 000401 BR 64$ ;;GET OVER THE ASCIZ

```

```

(3)          ;;65$: .ASCIZ <7>##
(3) 020630   64$:
(2) 020630   000401   BR      31$
(2) 020632   022626   30$:   CMP      (SP)+,(SP)+
(2) 020634   012637   000004   31$:   MOV      (SP)+,4      ;ALL THIS JUNK MUST BE REMOVED..
(2) 020640   005037   021456   CLR      $AERR
(2) 020644   004537   021460   JSR      R5,$LOAD      ;LOAD MICRO-CODE.
(2) 020650   000000G   .WORD   DRLPX2         ;FILE 'DRLPX2.OBJ'
(2) 020652   052777   040000   160502   BIS      #BIT14,@KMADO ;ISSUE KMC+DMC INIT.
(2) 020660   1$:
(2) 020660   010146   MOV      R1,-(SP)      ;'HANGS' HERE THEN KMC-11 ERROR.
(2) 020662   005001   CLR      R1
(2) 020664   005201   2$:   INC      R1              ;STALL FOR DMC-UP
(2) 020666   001376   BNE     2$
(2) 020670   012777   104000   160464   MOV      #BIT15.BIT11,@KMADO ;SET RUN, AND ENABLE ARBITRATION.
(2) 020676   105201   25$:   INCB    R1
(2) 020700   001376   BNE     25$
(2) 020702   032777   000040   160452   BIT      #BIT5,@KMADO  ;SLAVE READY? (READING IPBM SR)
(2) 020710   001401   BEQ     3$
(2) 020712   104000   ERROR   ;FATAL LPA-11 ERROR SLAVE NOT READY.
(2) 020714   012777   000004   160444   3$:   MOV      #4,@KMAD2      ;RFAD FAST PATH
(2) 020722   004537   022370   4$:   JSR      R5,$TOUT       ;-TOUT-CHECK FOR TIME OUT
(3) 020726   104000   ERROR   ;/TIME-OUT ERROR
(3)          ;/WE FAILED TO COMPLETE
(3)          ;/CURRENT OPERATION.
(3)          ;/CONTINUES IN THIS LOOP
(3)          ;/WOULD MAKE US 'HANG' HERE
(3) 020730   000774   BR      4$
(3)          ;/RETURNS HERE-FROM-TIMED OUT.
(2) 020732   122777   000377   160426   CMPB    #377,@KMAD2     ;WAIT TILL KMC DONE COMMAND.
(2) 020740   001370   BNE     4$
(2) 020742   122777   000377   160422   CMPB    #377,@KMAD4     ;IF FAST PATH=377 THEN ERROR.
(2) 020750   001001   BNE     35$
(2) 020752   104000   ERROR   ;IPBM ERROR (SLAVE SIDE)
(2)          ;YOU MUST RUN IPBM DIAGNOSTIC.
(2) 020754   122777   000004   160410   35$:   CMPB    #4,@KMAD4      ;IS THIS THE CORRECT VERSION OF MICRO-CODE?
(2) 020762   001543   BEQ     5$              ;YES-CONTINUE.
(2) 020764   005227   177777   INC     #-1
(2) 020770   001140   BNE     5$
(2) 020772   005227   177777   INC     #-1
(2) 020776   001135   BNE     5$
(3) 021000   104401   021006   TYPE    ,67$          ;;TYPE ASCIZ STRING
(3) 021004   000440   BR      66$          ;;GET OVER THE ASCIZ
(3)          ;;67$: .ASCIZ <200>'W A R N I N G THIS PROGRAM WAS DESIGNED TO RUN WITH VERSION 4'
(3) 021106   66$:
    
```

Line	Address	Code	Label	Op	Opnd	Comment
(3)	021106	104401	021114	TYPE	69\$::TYPE ASCIZ STRING
(3)	021112	000430		BR	68\$::GET OVER THE ASCIZ
(3)				69\$: .ASCIZ	<200>'MICRO-CODE. ANOTHER VERSION CODE WAS DETECTED.'	
(3)	021174			68\$:		
(3)	021174	104401	021202	TYPE	71\$::TYPE ASCIZ STRING
(3)	021200	000434		BR	70\$::GET OVER THE ASCIZ
(3)				71\$: .ASCIZ	<200>'THIS MAY OR MAYNOT CAUSE FALSE ERROR TO BE REPORTED.'	<200><200>
(3)	021272			70\$:		
(2)	021272	112737	177777	021424	5\$: MOVB	#0-1,11\$:DAC CODE FOR SLAVE.
(2)	021300	012501				
(2)	021302	021127	000000	6\$: MOV	(5)+,R1 :GET NEXT DEVICE ADDR.	
(2)	021306	001444				
(2)	021310	105237	021424			
(2)	021314	113777	021424	160050		
(2)	021322	004737	021426			
(2)	021326	112177	160040			
(2)	021332	004737	021426			
(2)	021336	112177	160030			
(2)	021342	004737	021426			
(2)						
(2)	021346	032777	000002	160006	7\$: BIT	#BIT1,@KMAD0 :WAIT FOR FIFO DATA
(2)	021354	001374				
(2)	021356	112777	000002	160002		
(2)						
(2)	021364					
(3)	021364	004537	022370			
(3)						
(3)	021370	104000				
(3)						
(3)						
(3)						
(3)						
(3)						
(3)	021372	000774				
(3)						
(2)	021374	122777	000377	157764		
(2)	021402	001370				
(2)	021404	105777	157762			
(2)	021410	001734				
(2)						
(2)	021412	005237	021456			
(2)						
(2)	021416	005041				
(2)	021420	012601				
(2)	021422	000205				
(2)						
(2)	021424	000000				
(2)						
(2)	021426	112777	000003	157732	20\$: MOVB	#3,@KMAD2 :ISSUE FIFO WRITE
(2)	021434					
(3)	021434	004537	022370			
(3)						
(3)	021440	104000				
(3)						

(2) 021616 000722 BR 1\$:TO PRESS CONTINUE TO GIVE IT
 (2) :ANOTHER CHANCE, BUT I DOUBT
 (2) :THAT THAT WOULD WORK. SINCE I'VE
 (2) :ALREADY GIVEN I' 177 (JCTAL) CHANCES.
 (2) :TRY RUNNING THE KMC-11 DIAGNOSTIC.

(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) 021620 010046 \$TLKW: MOV R0,-(SP) :SAVE R0
 (2) 021622 012500 MOV (5)+,R0 :GET DEVICE OFFSET
 (2) 021624 052700 000340 BIS #340,R0 :ADD WRITE CODE.
 (2) 021630 004737 022102 JSR PC,\$LPW :WAIT FOR FAST PATH READY
 (2) 021634 010037 021726 MOV R0,W1
 (2) 021640 010077 157526 MOV R0,@KMAD4
 (2) 021644 112777 000005 157514 MOVB #5,@KMAD2 :ISSUE FAST PATH WRITE
 (2) 021652 004737 022102 JSR PC,\$LPW :WAIT FOR RDY
 (2) 021656 011537 021730 MOV (5),W2
 (2) 021662 112577 157504 MOVB (5)+,@KMAD4 :WRITE LOW BYTE DATA.
 (2) 021666 112777 000005 157472 MOVB #5,@KMAD2 :FP WRITE
 (2) 021674 004737 022102 JSR PC,\$LPW
 (2) 021700 111537 021732 MOVB (5),W3
 (2) 021704 112577 157462 MOVB (5)+,@KMAD4 :WRITE HIGH BYTE
 (2) 021710 112777 000005 157450 MOVB #5,@KMAD2
 (2) 021716 004737 022102 JSR PC,\$LPW
 (2) 021722 012600 MOV (SP)+,R0
 (2) 021724 000205 RTS R5 :EXIT DONE.
 (2) 021726 000000 W1: 0
 (2) 021730 000000 W2: 0
 (2) 021732 000000 W3: 0

(2) :*THIS ROUTINE ISSUES A READ COMMAND TO THE LPA-11
 (2) :*
 (2) :* CALL = JSR R5,\$TLKR
 (2) :* .WORD 0 :OFFSET OF DEVICE
 (2) :* :RETURNS HERE
 (2) :* :DATA IN WORD \$DATR
 (2) :*

(2) 021734 010046 \$TLKR: MOV R0,-(SP) :SAVE R0
 (2) 021736 012500 MOV (5)+,R0 :GET OFFSET
 (2) 021740 052700 000300 BIS #300,R0 :ADD READ CODE
 (2) 021744 004737 022102 JSR PC,\$LPW :WAIT TILL READY
 (2) 021750 110077 157416 MOV R0,@KMAD4
 (2) 021754 112777 000005 157404 MOVB #5,@KMAD2 :ISSUE WRITE FP
 (2) 021762 004737 022102 JSR PC,\$LPW
 (2) 021766 010037 022076 MOV R0,RD1
 (2) 021772 1\$: JSR R5,\$TOUT
 (3) 021772 004537 022370 : -TOUT-CHECK FOR TIMEOUT

```

(3) 021776 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) 022000 000774 BR 1$
(3) ;/RETURNS HERE-FROM-TIMED OUT.
(2) 022002 032777 000040 157352 BIT #BIT5,@KMADO ;FAST PATH GOT DATA?
(2) 022010 001370 BNE 1$
(2) 022012 112777 000004 157346 MOVB #4,@KMAD2 ;ISSUE FAST PATH READ
(2) 022020 004737 022102 JSR PC,$LPW
(2) 022024 117737 157342 022100 MOVB @KMAD4,$DATR ;GET LOW BYTE
(2) 022032 2$: JSR R5,$TOUT ;-TOUT-CHECK FOR TIMEOUT
(3) 022032 004537 022370
(3) 022036 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) 022040 000774 BR 2$
(3) ;/RETURNS HERE-FROM-TIMED OUT.
(2) 022042 032777 000040 157312 BIT #BIT5,@KMADO ;FAST PATH READY?
(2) 022050 001370 BNE 2$
(2) 022052 112777 000004 157306 MOVB #4,@KMAD2 ;ISSUE FAST PATH READ
(2) 022060 004737 022102 JSR PC,$LPW
(2) 022064 117737 157302 022101 MOVB @KMAD4,$DATR+1 ;SAVE HIGH BYTE
(2) 022072 012600 MOV (SP)+,R0
(2) 022074 000205 RTS R5
(2) 022076 000000 RD1: 0
(2) 022100 000000 $DATR: .WORD 0
(2) ;THIS ROUTINE WAITS FOR KMC-CODE TO BECOME READY AS WELL
(2) ;AS FAST PATH TO BE READ.
(2) ;
(2) ; CALL = JSR PC,$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) 022102 010146 $LPW: MOV R1,-(SP) ;SAVE R1
(2) 022104 005001 CLR R1
(2) 022106 122777 000377 157252 1$: CMPB #377,@KMAD2 ;FINISHED INSTRUCTION?
(2) 022114 001403 BEQ 2$
(2) 022116 005201 INC R1 ;TIME OUT?
(2) 022120 001372 BNE 1$
(2) 022122 000411 BR 10$
(2) 022124 032777 000020 157230 2$: BIT #BIT4,@KMADO ;FAST PATH READ?
(2) 022132 001403 BEQ 3$
    
```


3861 042300 . =42300
3862
3863 000001 .END

ABASE = 167770	2956#	2969			
ACDW1 = 000000	2969				
ACDW2 = 000000	2969				
ACPUOP= 000000	2969				
ADDW0 = 000000	2969				
ADDW1 = 000000	2969				
ADDW10= 000000	2969				
ADDW11= 000000	2969				
ADDW12= 000000	2969				
ADDW13= 000000	2969				
ADDW14= 000000	2969				
ADDW15= 000000	2969				
ADDW2 = 000000	2969				
ADDW3 = 000000	2969				
ADDW4 = 000000	2969				
ADDW5 = 000000	2969				
ADDW6 = 000000	2969				
ADDW7 = 000000	2969				
ADDW8 = 000000	2969				
ADDW9 = 000000	2969				
ADEVCT= 000000	2969				
ADEVN = 000000	2969				
AENV = 000000	2969				
AENVN 000000	2969				
AFATAL- 000000	2969				
AMADR1 000000	2969				
AMADR2- 000000	2969				
AMADR3= 000000	2969				
AMADR4= 000000	2969				
AMAMS1= 000000	2969				
AMAMS2= 000000	2969				
AMAMS3= 000000	2969				
AMAMS4= 000000	2969				
AMSGAD= 000000	2969				
AMSGLG= 000000	2969				
AMSGTY= 000000	2969				
AMTYP1= 000000	2969				
AMTYP2= 000000	2969				
AMTYP3= 000000	2969				
AMTYP4= 000000	2969				
APASS - 000000	2969				
APRIOR= 000000	2969				
APTCSU= 000040	3848	3853#			
APTENV= 000001	3838	3848	3853#		
APTSIZ= 000200	3076	3853#			
APTSPO= 000100	3848	3853#			
ASWREG- 000000	2969				
AESTN- 000000	2969				
AUNIT = 000000	2969				
AUSWR - 000000	2969				
AVECT1= 100300	2957#	2969	3025		
AVECT2= 000000	2969				
BASEBA 001452	3027#	3078*	3084	3215	3776
BASEBR 001456	3029#	3081*	3082*	3245	
BASEI' 001454	3028#	3079*	3080*	3216	3777
BEGIN 001546	2961	3071#	3155		

SW06	=	000100	2958#						
SW07	=	000200	2958#						
SW08	=	000400	2958#						
SW09	=	001000	2958#						
SW1	=	000002	2958#						
SW10	=	002000	2958#						
SW11	=	004000	2958#						
SW12	=	010000	2958#	3770					
SW13	=	020000	2958#	3095	3176				
SW14	=	040000	2958#						
SW15	=	100000	2958#	3098	3183				
SW2	=	000004	2958#						
SW3	=	000010	2958#						
SW4	=	000020	2958#						
SW5	=	000040	2958#						
SW6	=	000100	2958#						
SW7	=	000200	2958#						
SW8	=	000400	2958#						
SW9	=	001000	2958#						
TALK	=	003234	3134	3137	3140	3143	3160	3193#	
TBITVE	=	000014	2958#						
TIME	=	022540	3857#*						
TKVEC	=	000060	2958#						
TPVEC	=	000064	2958#						
TRANST	=	001540	3058#	3125*	3132*	3145	3151	3248	
TRAPVE	=	000034	2958#	3076*					
TRTVEC	=	000014	2958#						
TSTNUM	=	015502	3831	3832	3833	3835#	3838*		
TST1	=	003616	3250#						
TST10	=	004324	3319#						
TST11	=	004432	3336#						
TST12	=	004734	3346	3348#					
TST13	=	005236	3358	3361#					
TST14	=	005334	3370	3373#					
TST15	=	005414	3380	3383#					
TST16	=	005456	3388	3391#					
TST17	=	005536	3398	3411#					
TST2	=	003672	3258#						
TST20	=	005662	3423	3426#					
TST21	=	005774	3435	3438#					
TST22	=	006106	3447	3450#					
TST23	=	006220	3459	3461#					
TST24	=	006466	3463	3496#					
TST25	=	006674	3532#						
TST26	=	007072	3559#						
TST27	=	007312	3586#						
TST3	=	003742	3264	3266#					
TST30	=	010572	3611	3614#					
TST31	=	011722	3638	3640#					
TST32	=	012042	3650	3653#					
TST33	=	012130	3661	3666#					
TST34	=	012230	3676	3680#					
TST35	=	012626	3682	3730#					
TST36	=	013116	3732	3767#					
TST4	=	004020	3273	3276#					
TST5	=	004070	3282	3285#					

SDMAST	1792#														
SDMDT	2821#														
SMMAST	761#														
SSCMRE	2969#														
SSCMTM	2969#														
SSESCA	2958#														
SSNEWT	2958#	3250	3258	3266	3276	3285	3294	3304	3319	3336	3348	3361	3373	3383	3391
	3411	3426	3438	3450	3461	3496	3532	3559	3586	3614	3640	3653	3666	3680	3730
	3767														
SSSET	3854#														
SSSETM	3076#														
SSSKIP	2958#	3264	3273	3282	3291	3346	3358	3370	3380	3388	3398	3423	3435	3447	3459
	3463	3611	3638	3650	3667	3676	3682	3732							
.EQUAT	2929#	2958													
.HEADE	2929#	2954													
.KMADR	55#	3025													
.KSIS	184#	3077													
.LOADL	458#	3857													
.LPAIN	209#	3857													
.PUTCS	417#	3857													
.RESET	328#	3857													
.SETUP	2930#	3007													
.SWRHI	2931#	2959													
.SWRLO	2959#														
.UTK	698#	3857													
.SACT1	2932#	2966													
.SAPT8	2932#	2969#													
.SAPTH	2932#	2968													
.SAPTY	2932#	3853													
.SCATC	2929#	2961													
.SCMTA	2929#	2969													
.SEOP	2929#	3783													
.SERRO	2929#	3838													
.SERR1	2931#	3840													
.\$INLP	651#	3857													
.\$MMAC	141#														
.\$OUTL	609#	3857													
.\$PARM	2930#														
.\$POWE	2930#	3844													
.\$RDOC	2931#	3849													
.\$READ	2930#	3850													
.\$SAVE	2930#														
.\$SCOP	2930#	3837													
.\$SPAC	2930#														
.\$SWDO	2930#														
.\$TLKW	510#	3857													
.\$TOUT	3025#	3857													
.\$TRAP	2930#	3854													
.\$TYPD	2931#	3784													
.\$TYPE	2929#	2930#	3848												
.\$TYPO	2929#	3842													

. ABS. 042300 000 CON RW ABS GBL D
 000000 001 CON RW REL LCL I

LPA DR11-K LOGIC TEST MD-11-CRLPF-B
CRLPFB.P11 08-AUG-79 10:08

MACY11 30G(1063) 08-AUG-79^{J 7} 10:09 PAGE 10-2
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0087

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

CRLPFB,CRLPFB/CRF=DRLPA.MAC,CRLPFB
RUN-TIME: 32 20 1 SECONDS
RUN-TIME RATIO: 103/53=1.9
CORE USED: 39K (77 PAGES)