

KDF11-B,
DLV11-J

KDF11-B SLU/DLV11-J
CJDLAA0

AH-S971A-MC
FICHE 1 OF 1

APR 1982
COPYRIGHT © 1982
MADE IN USA

A large grid of microfilm frames, each containing a small, illegible image or document snippet. The grid is organized into rows and columns, with some frames appearing blank or containing faint markings.

.REM %

IDENTIFICATION

PRODUCT CODE: AC-S970A-MC

PRODUCT NAME: CJD LAA0 KDF11B SLU/DLV11J TST

PRODUCT DATE: JULY 1981

MAINTAINER: DIAGNOSTIC ENGINEERING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C): 1982 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	DECX/11

TABLE OF CONTENTS

1.0	GENERAL PROGRAM INFORMATION.
1.1	PROGRAM PURPOSE (ABSTRACT).
1.2	SYSTEM REQUIREMENTS.
1.3	RELATED DOCUMENTS, STANDARDS AND DIAGNOSTIC HIERARCHY PREREQUISITES.
2.0	OPERATING INSTRUCTIONS.
2.1	LOADING AND STARTING PROCEDURES.
2.2	SPECIAL ENVIRONMENTS.
2.3	OPERATIONAL SWITCH SETTINGS
2.4	PROGRAM OPTIONS.
2.5	EXECUTION TIMES.
2.6	POWER FAIL.

3.0	ERROR INFORMATION.
3.1	ERROR REPORTING PROCEDURE.
3.2	ERROR HALTS.
4.0	PERFORMANCE REPORTS.
5.0	DEVICE INFORMATION TABLES.
6.0	SUMMARY OF TESTS.

1.0 GENERAL PROGRAM INFORMATION.

1.1 PROGRAM PURPOSE (ABSTRACT).

THIS DIAGNOSTIC IS A LOGIC TEST TO VERIFY THE OPERATION OF THE
SLU'S ON THE KDF11-B AND UP TO 2 DLV11-J MODULES.

1. ALL SELECTED CHANNELS ARE TESTED INDIVIDUALLY
2. THE DLV11-J MODULES ARE TESTED AS A WHOLE FOR CHANNEL INTERACTION
PROBLEMS. THIS DIAGNOSTIC IS DESIGNED TO TEST AND DETECT
ERRORS TO THE LOGIC LEVEL (NOT TO THE CHIP LEVEL).

THIS DIAGNOSTIC OPERATES UP TO 2 DLV11-J SERIAL LINE INTERFACES
CONFIGURED AT CONSECUTIVE BASE ADDRESSES.
BITS 9 AND 10 OF THE USER SWITCH REGISTER, \$USWR (1220), ARE USED TO
DETERMINE WHAT DEVICES ARE PRESENT.

THE PROGRAM WILL PRINT OUT ALL MODULES & CHANNELS
TO BE TESTED BEFORE PROCEEDING.

THE OPERATOR MUST INSTALL DATA WRAP AROUND CONNECTORS TO DO
DATA TESTING. TO BYPASS DATA TESTS, THE OPERATOR MUST MODIFY
'\$USWR (1220)' (USER SWITCH REGISTER) , SEE PROGRAM OPTIONS SEC. 2.4)

THE DEFAULT ADDRESSES & VECTORS ARE AS FOLLOWS:

177560 -CONSOLE INTERFACE DEVICE ADDRESS
176500 -FIRST SERIAL CHANNEL ADDRESS OF UP TO 8 CONSECUTIVE
SERIAL LINE DEVICES.

60 - VECTOR FOR CONSOLF DEVICE INTERFACE.
300 - VECTOR FOR FIRST OF 8 DEVICES.

THIS PROGRAM IS DESIGNED TO RUN ON A PDP-11/23B WITH 8K OF
MEMORY. IT CAN RUN UNDER XXDP &
APT MONITORS, AND ON PROCESSORS WITH NO HARDWARE
SWITCH REGISTER.

1.2 SYSTEM REQUIREMENTS.

HARDWARE REQUIREMENTS:

ANY PDP 11/23B PROCESSOR
8K MEMORY - MINIMUM
A SPECIAL DATA WRAP AROUND CONNECTOR OR EQUIVALENT
(REQ'D IF DATA WRAP AROUND TESTS DESIRED)

IF DATA WRAP AROUND TESTS ARE BYPASSED:
TESTS 7-12, 14-17, 21, 22 ARE BYPASSED.

SOFTWARE REQUIREMENTS:

THIS DIAGNOSTIC IS DESIGNED TO RUN IN ANY OF THE
FOLLOWING WAYS:
STAND ALONE
WITH APT MONITOR
WITH XXDP MONITOR (CHAINABLE IF RENAMED TO .BIC EXTENSION)

THIS DIAGNOSTIC IS NOT DESIGNED TO RUN WITH THE
DIAGNOSTIC SUPERVISOR.

1.3 RELATED DOCUMENTS AND STANDARDS.

DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS
APT
SYSMAC
SPMACJ USERS GUIDE

175-003-009-02
MD-11-DZZMA
MD-11-DZQAC
RAS - 81 015 - 0

DIAGNOSTIC HIERARCHY PREREQUISITES

NO SPECIAL DIAGNOSTICS ARE REQUIRED TO RUN BEFORE THIS, BUT
THE PROCESSOR, MEMORY, AND BUS ARE ASSUMED TO BE FULLY
OPERATIONAL.

THIS DIAGNOSTIC ASSUMES THAT THE OPERATOR WILL INITIALIZE
LOCATION '\$USWR (1220)' TO THE PROPER VALUE. (SEE SEC. 2.4)

2.0 OPERATING INSTRUCTIONS.

2.1 LOADING AND STARTING PROCEDURES.

USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMAT
TED MEDIA.

THERE IS ONE STARTING LOCATION FOR THIS DIAGNOSTIC.
IT IS LOCATION 200.

AS SOON AS TESTING STARTS, THE OPERATOR CAN CHANGE THE SWITCH

REGISTER ONLY BY A 'BREAK' & MANUALLY LOADING LOCATION 176 (SWREG) WITH THE DESIRED CONTENTS (SEE SEC. 2.3) THEN TYPING A 'P' TO PROCEED.

THE NORMAL STARTING PROCEDURE WOULD BE TO USE THE DEFAULT VALUE ALREADY STORED IN LOCATION 1220(\$USWR). THEREFORE, THE OPERATOR SHOULD JUST TYPE IN 'R CJDLAA'. HOWEVER, IF A VALUE OTHER THAN THE DEFAULT VALUE (31) IS DESIRED IN \$USWR(1220) THE OPERATOR SHOULD TYPE IN 'L CJDLAA'. AFTER THE PROGRAM HAS BEEN LOADED INTO MEMORY THE MONITOR WILL STILL HAVE CONTROL. THE OPERATOR SHOULD HIT THE BREAK KEY. IN RESPONSE TO THE 'AT' SIGN, LOCATION 1220(\$USWR) CAN BE ALTERED BY TYPING IN '1220' FOLLOWED BY A SLASH. THE CONTENTS OF THE LOCATION WILL BE TYPED OUT. THE NEW VALUE SHOULD BE TYPED IN FOLLOWED BY A CARRIAGE-RETURN. ANOTHER 'AT' SIGN WILL BE TYPED ON THE DEVICE. THE OPERATOR MUST THEN TYPE IN 200G, THE PROGRAM SHOULD THEN RUN BY ITSELF.

2.2 SPECIAL ENVIRONMENTS.

THIS DIAGNOSTIC FOLLOWS THE STANDARD PROCEDURE FOR RUNNING UDER APT, XXDP MONITORS, AS DESCRIBED IN THEIR RESPECTIVE PROCEDURES MANUAL AND SYSMAC PACKAGE.

2.3 OPERATIONAL SWITCH SETTINGS

THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED FOR ALL OPERATIONAL SWITCH SETTINGS. THIS CAN BE ACCOMPLISHED IN THE FOLLOWING WAY:

- 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). LEADING ZEROS NEED NOT BE TYPED, AND IF MORE THAN 6 DIGITS ARE TYPED THE LAST 6 WILL BE USED. 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 3.
 - C) IF THE INPUT CHARACTER IS NOT ONE OF THE CHARACTERS MENTIONED ABOVE THEN A QUESTION MARK (?) WILL BE TYPED FOLLOWED BY A CARRIAGE RETURN AND A LINE FEED SEQUENCE THEN PROCEED FROM STEP 3 (ERASING ALL PREVIOUS INPUT).

4) THE DIAGNOSTIC WILL CONTINUE ON TYPING <CR>.

NOTE: BECAUSE OF FREQUENT BUS RESETS IN THE PROGRAM, IT MAY BE NECESSARY TO DO 'CONTROL-G' SEVERAL TIMES. ALTERNATELY, A 'BREAK' & MANUALLY LOADING LOC. 176, FOLLOWED BY A 'P' TO PROCEED WILL ALSO WORK.

SOFTWARE SWITCH REGISTER OPTIONS (SWREG)

BIT 15 SET = 100000 = HALT ON ERROR
14 SET = 40000 = LOOP ON TEST (TO BE USED ONLY WHILE TESTING IN PR
13 SET = 20000 = INHIBIT ERROR TYPEOUTS
12 SET = 10000 = ENABLE PERFORMANCE REPORTS
11 SET = 4000 = INHIBIT ITERATIONS
10 SET = 2000 = BELL ON ERROR
9 SET = 1000 = LOOP ON ERROR
8 SET = 400 = LOOP ON TEST IN SWR<7:0>
7:0 = NUMBER OF TEST TO LOOP ON (USED WITH BIT 8)
(ALL TESTS PREVIOUS TO THE SELECTED TEST
ARE EXECUTED FIRST WITH 1 ITERATION ONLY)

2.4 PROGRAM OPTIONS.

LOCATION '\$DEVN' IS USED AS A BIT MAP TO INDICATE WHICH UNIT NUMBERS ARE PRESENT AND WILL BE TESTED.

\$DEVN (BITS 15 THRU 8 NOT USED)

								!----MODULE 2-----!				!----MODULE 1----		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

:	:	:	:	:	:	:	:	3	2	1	0	3	2	1

NOTE: SIZING IS PERFORMED ONLY ONCE AT THE BEGINNING OF THE PROGRAM. IF \$DEVN IS TO BE CHANGED, THE PROGRAM MUST BE RESTARTED AT 200 FOR IT TO BE EFFECTIVE.

OPTIONS

LOCATION \$USWR (1220) CONTAINS ALL THE USER SELECTABLE OPTIONS. THE VALUE IN THIS WORD MUST CONFORM TO THE ACTUAL BOARD CONFIGURATION.

BIT POSITION	DEFINITION	\$USWR DEFAULT VALUE
0	# OF DATA BITS TRANSMITTED 0 = 7 BITS, 1 = 8 BITS	1 = 8 BITS
1	PARITY ENABLED	0 = NO
2	EVEN ODD PARITY	0 = ODD
3	BREAK DETECTION ENABLED	1 = YES
4	RUN DATA WRAP AROUND TESTS	1 = YES

<10:9> DEVICES PRESENT

<10:9> = 0 = ONLY TESTING KDF11-B
 <10:9> = 1 = A KDF11-B AND ONE DLV11-J
 <10:9> = 2 = A KDF11-B AND 2 DLV11-J'S

IMPORTANT:*****

1. FOR DIAGNOSTIC PURPOSES, ALL CHANNELS MUST BE CONFIGURED TO THE SAME BIT/WORD LENGTH.
2. WHEN BITS 10:9 ARE SET TO ONE, THE DLV11 ADDRESS MUST BE SET AT 176500 AND BE CONTIGUOUS. THE SECOND SLU ADDRESS OF THE KDF11-B MUST BE 176540. IF BITS 10:9 ARE SET TO TWO THEN THE SECOND SLU ON THE KDF11-B MUST BE DESELECTED. THE DLV11 MODULES MUST START AT ADDRESS 176500 AND BE CONTIGUOUS.

SUMMARY OF DEVICE ADDRESSES & VECTORS.

			VECTORS		
			REC	XMIT	
			---	---	
DEV ADDRESS (RCSR)					

MODULE #1	CHANNEL	0	\$BASE+00	\$VECT1+00	\$VECT1+4
		1	+10	+10	+14
		2	+20	+20	+24
		3	+30	+30	+34
MODULE #2	CHANNEL	0	+40	+40	+44
		1	+50	+50	+54
		2	+60	+60	+64
		3	+70	+70	+74
CONSOLE DEVICE			\$TKS	TKVEC=60	TPVEC=64

SUMMARY OF USER LOCATIONS & DEFAULTS.

	LOC	DEFAULT	
\$BASE	1250	176500	
\$VECT1	1244	300	
\$TKS	1144	177560	
TKVEC		60	
TPVEC		64	
\$USWR	1220	0031	SEE SEC. 2.4
\$DEVM	1252	0	SEE SEC. 2.4
SWREG	176	0	SEE SEC. 2.3

2.5

EXECUTION TIMES.
 THE TEST TIME IS BAUD RATE DEPENDENT; HIGHER BAUD RATES RESULT IN SHORTER PASS TIMES.

2.6 POWER FAIL.

AUTO START FROM POWER FAIL IS NOT IMPLEMENTED IN THIS PROGRAM.

3.0 ERROR INFORMATION.

3.1 ERROR REPORTING PROCEDURE.

SINCE THIS DIAGNOSTIC WAS DESIGNED TO FIT IN 8K OF MEMORY THE
ERROR TYPEOUT IS VERY BRIEF. THE FORMAT OF THE ERROR TYPEOUT
IS AS FOLLOWS:

TEST#____,ERROR#____,PC=____,ADDRESS=____,VECTOR=____

WHERE ALL VALUES TYPED ARE OCTAL.
THE ADDRESS AND VECTOR REFER TO THE FAILING CHANNEL.
FOR FURTHER INFORMATION THE LISTING MUST BE CONSULTED.
BITS 15,13,10 AND 9 OF THE SWITCH REGISTER (SWREG) CONTROL THE
SEQUENCE OF EVENTS AFTER AN ERROR IS CAUGHT.

BIT 15 SET: CAUSES THE PROGRAM TO HALT IN THE ERROR ROUTIN
IF THE PROGRAM IS CONTINUED, IT WILL PROCEED
FROM WHERE IT HALTED.

BIT 13 SET: DISABLES THE PRINTING OF THE ERROR MESSAGE.

BIT 10 SET: CAUSES THE BELL TO RING ON ERROR.

BIT 9 SET: CAUSES THE DIAGNOSTIC TO LOOP FROM BEGINNING
OF TEST TO ERROR.

THE ERROR ROUTINE SUPPORTS THE CONTROL G <^G> FUNCTION.
REFER TO SECTION 2.3 FOR DETAILS.

3.2 ERROR HALTS.

ONLY IF BIT 15 OF THE SWITCH REGISTER (SWREG) IS SET
WHEN AN ERROR OCCURS.

4.0 PERFORMANCE REPORTS.

PERFORMANCE REPORTS. (BIT 12 SET IN THE SWITCH REGISTER 'SWREG')

'ERRORS:--' : THE TOTAL NUMBER OF ERRORS
ON THIS PASS.

AFTER ALL MODULES & CHANNELS TO BE TESTED HAVE BEEN EXERCISED,
AN END PASS STATEMENT IS TYPED:

'END PASS#-----.'

5.0 DEVICE REGISTERS.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
RCSR	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****
									RCVR	RCVR					
									DONE	IE					
RBUF	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****
	ERRO	OR	FR	P											
	R	ERR	ERR	ERR											
TCSR	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****
									XMIT	XMIT					
									RDY	IE					
TBUF	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****	*****

NOTES:

1. RCSR AT BASE ADDRESS (\$BASE)
RBUF AT \$BASE+2
TCSR AT \$BASE+4
TBUF AT \$BASE+6
2. BLANK BOXES INDICATE UNUSED AND RESERVED BIT POSITIONS. SEE THE LISTING FOR AN EXPLANATION OF THE BITS.
3. ORERR = OVERRUN ERROR
FRERR = FRAMING ERROR
PERR = PARITY ERROR

6.0 SUMMARY OF TESTS AND SPECIAL SUBROUTINES.

PHASE 1 TESTS

TEST 1 ADDRESSABILITY

THIS TEST VERIFIES THAT ALL 4 REGISTERS OF THE CHANNEL UNDER TEST RESPOND TO THEIR ADDRESSES.

THE FOLLOWING 3 TESTS TEST ALL 'READ WRITE' BITS

TEST 2 BREAK - TCSR 0 SET, CLEAR, RESET

449
450 TEST 3 XMITIE - TCSR 6 SET, CLEAR, RESET
451 ---- -
452
453
454 TEST 4 RCVRIE - RCSR 6 SET, CLEAR, RESET
455 ---- -
456
457
458 TEST 5 XMITRDY - TCSR 7 - IS SET BY INIT
459 ---- --
460
461
462 TEST 6 XMIT RDY - TCSR 7 - CLEARS WHEN TBUF IS LOADED
463 ---- -- WITH A CHARACTER AND THAT IT SETS WITHIN A
464 REASONABLE AMOUNT OF TIME.
465
466
467 TEST 7 OUTPUTTING A CHAR FROM TBUF (WITH WRAP AROUND CONNECTED)
468 ---- -- RESULTS IN RCVRDONE SETTING WITHIN A
469 REASONABLE AMOUNT OF TIME AND THAT RESET
470 CLEARS THE BIT.
471
472 TEST 10 RCVRDONE IS CLEARED BY SETTING READER ENABLE
473 ---- --
474
475
476 TEST 11 RCVRDONE IS CLEARED BY READING RBUF
477 ---- --
478
479
480 TEST 12 OVERRUN & ERROR BIT - RBUF 14
481 ---- --
482 TEST 13 TRANSMITTER INTERRUPT LOGIC TEST
483 ---- --
484 LOGICALLY THIS IS 4 SEPARATE TESTS
485 A) DOES TRANSMITTER INTERRUPT LOGIC WORK
486 B) AT PRIORITY OF 0
487 C) AND ONLY ONCE
488 D) BUT NOT WITH INTERRUPT ENABLE CLEAR
489
490
491 TEST 14 RECEIVER INTERRUPT LOGIC TEST THIS TEST COVERS ALL
492 ---- -- OF THE RECEIVER SIDE OF THE INTERRUPT LOGIC IN
493 CHARACTER MODE.
494
495
496 TEST 15 TEST DATA WRAP AROUND: FLAG MODE.
497 ---- --
498
499 TEST 16 TEST DATA WRAP AROUND: INTERRUPT MODE.
500 ---- --
501
502
503 TEST 17 TEST BREAK DETECTION LOGIC TRANSMIT KNOWN CHAR
504 ---- --

- A) TRANSMIT KNOWN CHAR WITH BREAK SET
AND COMPARE RECEIVED WITH 0
- B) TEST FOR FRAMING ERROR ON BREAK
- C) IF PARITY IS ENABLED AND ODD PARITY IS SELECTED,
CHECK TO BE SURE PARITY ERROR WAS GENERATED
- D) IF PARITY IS ENABLED AND EVEN PARITY IS SELECTED,
CHECK TO BE SURE NO PARITY ERROR OCCURRED

TEST 20 NOT A TEST - PREPARE FOR TESTS 21 & 22

TEST 21 TEST THAT CHANNELS INTERRUPT AT ASSIGNED PRIORITY

TEST 22 TEST DATA TRANSFERS WITH ALL ACTIVE LINES INTERRUPTING.

%

.TITLE KDF11-B / DLV11-J TEST
.*COPYRIGHT (C) 1980
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY GREGORY GLEZMAN & GARY PAPAIZIAN
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
.*

: THIS PROGRAM WAS ASSEMBLED USING SPMACJ VERSION Y03.3
: *****

: THIS PROGRAM WAS FIRST COMPLETED BY G GLEZMAN AND G PAPAIZIAN
: IT WAS THEN MODIFIED BY RICHARD KIMBALL
: AND ONCE AGAIN ALTERED BY E S REDNER
: THERE IS ROOM FOR MORE
: *****
: *****
: SBTTL OPERATIONAL SWITCH SETTINGS

```
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
```

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<7:0>

```
.SBTTL BASIC DEFINITIONS

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

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

;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0                ;;GENERAL REGISTER
R1= %1                ;;GENERAL REGISTER
R2= %2                ;;GENERAL REGISTER
R3= %3                ;;GENERAL REGISTER
R4= %4                ;;GENERAL REGISTER
R5= %5                ;;GENERAL REGISTER
R6= %6                ;;GENERAL REGISTER
R7= %7                ;;GENERAL REGISTER
SP= %6                ;;STACK POINTER
PC= %7                ;;PROGRAM COUNTER

;*PRIORITY LEVEL DEFINITIONS
PR0= 0                ;;PRIORITY LEVEL 0
PR1= 40               ;;PRIORITY LEVEL 1
PR2= 100              ;;PRIORITY LEVEL 2
PR3= 140              ;;PRIORITY LEVEL 3
PR4= 200              ;;PRIORITY LEVEL 4
PR5= 240              ;;PRIORITY LEVEL 5
PR6= 300              ;;PRIORITY LEVEL 6
PR7= 340              ;;PRIORITY LEVEL 7

;*SWITCH REGISTER SWITCH DEFINITIONS
SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
```

001100

000011

000012

000015

000200

177776

177774

177772

177570

177570

000000

000001

000002

000003

000004

000005

000006

000007

000006

000007

000040

000100

000140

000200

000240

000300

000340

100000

040000

020000

010000

617	004000	SW11=	4000
618	002000	SW10=	2000
619	001000	SW09=	1000
620	000400	SW08=	400
621	000200	SW07=	200
622	000100	SW06=	100
623	000040	SW05=	40
624	000020	SW04=	20
625	000010	SW03=	10
626	000004	SW02=	4
627	000002	SW01=	2
628	000001	SW00=	1
629		.EQUIV	SW09,SW9
630		.EQUIV	SW08,SW8
631		.EQUIV	SW07,SW7
632		.EQUIV	SW06,SW6
633		.EQUIV	SW05,SW5
634		.EQUIV	SW04,SW4
635		.EQUIV	SW03,SW3
636		.EQUIV	SW02,SW2
637		.EQUIV	SW01,SW1
638		.EQUIV	SW00,SW0
639			
640		;*DATA BIT DEFINITIONS (BIT00 TO BIT15)	
641	100000	BIT15=	100000
642	040000	BIT14=	40000
643	020000	BIT13=	20000
644	010000	BIT12=	10000
645	004000	BIT11=	4000
646	002000	BIT10=	2000
647	001000	BIT09=	1000
648	000400	BIT08=	400
649	000200	BIT07=	200
650	000100	BIT06=	100
651	000040	BIT05=	40
652	000020	BIT04=	20
653	000010	BIT03=	10
654	000004	BIT02=	4
655	000002	BIT01=	2
656	000001	BIT00=	1
657		.EQUIV	BIT09,BIT9
658		.EQUIV	BIT08,BIT8
659		.EQUIV	BIT07,BIT7
660		.EQUIV	BIT06,BIT6
661		.EQUIV	BIT05,BIT5
662		.EQUIV	BIT04,BIT4
663		.EQUIV	BIT03,BIT3
664		.EQUIV	BIT02,BIT2
665		.EQUIV	BIT01,BIT1
666		.EQUIV	BIT00,BIT0
667			
668		;*BASIC "CPU" TRAP VECTOR ADDRESSES	
669	000004	ERRVEC=	4
670	000010	RESVEC=	10
671	000014	TBITVEC=	14
672	000014	TRTVEC=	14

;;TIME OUT AND OTHER ERRORS
 ;;RESERVED AND ILLEGAL INSTRUCTIONS
 ;;T BIT
 ;;TRACE TRAP

673	000014	BPTVEC= 14	::BREAKPOINT TRAP (BPT)
674	000020	IOTVEC= 20	::INPUT/OUTPUT TRAP (IOT) **SCOPE**
675	000024	PWRVEC= 24	::POWER FAIL
676	000030	EMTVEC= 30	::EMULATOR TRAP (EMT) **ERROR**
677	000034	TRAPVEC=34	::"TRAP" TRAP
678	000060	TKVEC= 60	::TTY KEYBOARD VECTOR
679	000064	TPVEC= 64	::TTY PRINTER VECTOR
680	000240	PIRQVEC=240	::PROGRAM INTERRUPT REQUEST VECTOR
681	000001	ADRS= R1	
682			
683			
684		:*****	
685		:SBTTL TRAP CATCHER	
686			
687	000000	. =0	
688		:*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"	
689		:*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS	
690		:*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS	
691	000174	. =174	
692	000174 000000	DISPREG: .WORD 0	::SOFTWARE DISPLAY REGISTER
693	000176 000000	SWREG: .WORD 0	::SOFTWARE SWITCH REGISTER
694		.SBTTL	STARTING ADDRESS(ES)
695	000200 000137 001330	JMP	@#START ::JUMP TO STARTING ADDRESS OF PROGRAM
696			
697			
698			
699			
700		: OFFLINE TESTING FOR INTERRUPT VECTOR PROBLEMS	
701			
702	000400	. =400	
703			
704	000400 005000	CLR R0	
705	000402 012720 017632	1\$: MOV #INTSRV,(R0)+	::SET INTR HANDLER PTR
706	000406 012720 000340	MOV #PR7,(R0)+	::SET PRIORITY
707	000412 020027 000400	CMP R0,#400	::ALL DONE?
708	000416 001371	BNE 1\$	::BR IF NO
709	000420 005037 000176	CLR 176	::CLEAN UP SWREG
710	000424 000137 001330	JMP START	::GO DO IT


```

711
712      .SBTTL  ACT11 HOOKS
713
714      ;*****
715      ;HOOKS REQUIRED BY ACT11
716          $SVPC=.          ;SAVE PC
717          .=46
718      000046 021712      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
719          .=52
720      000052 000000      .WORD 0      ;;2)SET LOC.52 TO ZERO
721          .=$SVPC          ;; RESTORE PC
722          .=1000
723      .SBTTL  APT PARAMETER BLOCK
724
725      ;*****
726      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
727      ;*****
728          .SX=.      ;;SAVE CURRENT LOCATION
729          .=24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
730      000024 000200      200      ;;FOR APT START UP
731          .=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
732      000044 001000      $APTHDR    ;;POINT TO APT HEADER BLOCK
733          .=$X      ;;RESET LOCATION COUNTER
734      ;*****
735      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
736      ;INTERFACE SPEC.
737
738      $APTHD:
739      001000 000000      $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
740      001002 001174      $MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
741      001004 000031      $TSTM: .WORD 25.      ;;RUN TIM OF LONGEST TEST
742      001006 000144      $PASTM: .WORD 100.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
743      001010 000144      $UNITM: .WORD 100.      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
744      001012 000030      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
  
```

```
745
746
747
748
749
750
751      001100
752 001100      000000
753 001100      000000
754 001102      000
755 001103      000
756 001104      000000
757 001106      000000
758 001110      000000
759 001112      000000
760 001114      000
761 001115      001
762 001116      000000
763 001120      000000
764 001122      000000
765 001124      000000
766 001126      000000
767 001130      000000
768 001132      000000
769 001134      000
770 001135      000
771 001136      000000
772 001140      177570
773 001142      177570
774 001144      177560
775 001146      177562
776 001150      177564
777 001152      177566
778 001154      000
779 001155      002
780 001156      012
781 001157      000
782 001160      000000
783 001162      000000
784 001164      177607 000377
785 001170      077
786 001171      015
787 001172      000012
788
789
790
791
792
793 001174
794 001174      000000
795 001176      000000
796 001200      000000
797 001202      000000
798 001204      000000
799 001206      000000
800 001210      000000

.SBTTL COMMON TAGS

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

      . =1100
$CMTAG:      . =1100      :: START OF COMMON TAGS
      .WORD      0
$TSTNM:      .BYTE      0      :: CONTAINS THE TEST NUMBER
$ERRFLG:      .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
      .WORD      0      :: RESERVED--NOT TO BE USED
      .WORD      0
$AUTOB:      .BYTE      0      :: AUTOMATIC MODE INDICATOR
$INTAG:      .BYTE      0      :: INTERRUPT MODE INDICATOR
      .WORD      0
$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
$BELL:      .ASCIZ      <207><377><377>      :: CODE FOR BELL
$QUES:      .ASCII      /?/      :: QUESTION MARK
$CRLF:      .ASCII      <15>      :: CARRIAGE RETURN
$LF:      .ASCIZ      <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
```


801	001212	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
802	001214		\$ETABLE:		::APT ENVIRONMENT TABLE
803	001214	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
804	001215	000	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
805	001216	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
806	001220	000031	\$USWR: .WORD	AUSWR	::USER SWITCHES
807	001222	000000	\$CPUOP: .WORD	ACPUOP	::CPU TYPE,OPTIONS
808			::*		BITS 15-11=CPU TYPE
809			::*		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
810			::*		11/70=06,PDQ=07,Q=10
811			::*		BIT 10=REAL TIME CLOCK
812			::*		BIT 9=FLOATING POINT PROCESSOR
813			::*		BIT 8=MEMORY MANAGEMENT
814	001224	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
815	001225	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
816			::*		MEM.TYPE BYTE -- (HIGH BYTE)
817			::*		900 NSEC CORE=001
818			::*		300 NSEC BIPOLAR=002
819			::*		500 NSEC MOS=003
820	001226	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
821			::*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
822	001230	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
823	001231	000	\$MTYP2: .BYTE	AMTYP2	::MEM.TYPE,BLK#2
824	001232	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
825	001234	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
826	001235	000	\$MTYP3: .BYTE	AMTYP3	::MEM.TYPE,BLK#3
827	001236	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
828	001240	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
829	001241	000	\$MTYP4: .BYTE	AMTYP4	::MEM.TYPE,BLK#4
830	001242	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
831	001244	000300	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
832	001246	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
833	001250	176500	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
834	001252	000000	\$DEVN: .WORD	ADEVN	::DEVICE MAP
835	001254		\$ETEND:		
836			.MEXIT		

837
 838
 839
 840
 841
 842
 843
 844
 845
 846
 847
 848
 849
 850
 851 001254
 852
 853 001254 000000
 854 001256 176500
 855 001260 000300
 856 001262 001256
 857 001264 001260
 858 001266 001262
 859 001270 001264
 860
 861
 862
 863 001272 000000
 864 001274 177560
 865 001276 000000
 866 001300 000000
 867 001302 000000
 868 001304 000000
 869 001306 000000
 870 001310 000000
 871 001312 000000
 872 001314 000000
 873 001316 000000
 874 001320 000000
 875 001322 000000
 876 001324 000000
 877 001326 000000
 878 001330
 879
 880
 881 001330 012706 001100
 882 001334 005026
 883 001336 022706 001140
 884 001342 001374
 885 001344 012706 001100
 886
 887 001350 012737 023534 000020
 888 001356 012737 000340 000022
 889 001364 012737 023334 000030
 890 001372 012737 000340 000032
 891 001400 012737 024466 000034
 892 001406 012737 000340 000036

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

;; GLOBAL DATA
 I: .WORD 0
 DLADD: .WORD 176500
 DLVEC: .WORD 300
 RCSR: DLADD + 0
 RBUF: DLADD + 2
 TCSR: DLADD + 4
 TBUF: DLADD + 6

CONTST: .WORD 0
 CONADR: .WORD 177560
 PHASE2: .WORD 0
 P1CNT: .WORD 0
 SECSLU: .WORD 0
 NODLV: .WORD 0
 KDDLVL: .WORD 0
 DLV211: .WORD 0
 DLBASE: .WORD 0
 KDFTST: .WORD 0
 SFTREG: .WORD 0
 MASK: .WORD 0
 KDF11B: .WORD 0
 CONSOL: .WORD 0
 TMP9: .WORD 0

START:

.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


```
893 001414 013737 021660 021652      MOV    $ENDCT,$EOPCT      ;;SETUP END-OF-PROGRAM COUNTER
894 001422 005037 001160              CLR    $TIMES           ;;INITIALIZE NUMBER OF ITERATIONS
895 001426 005037 001162              CLR    $ESCAPE          ;;CLEAR THE ESCAPE ON ERROR ADDRESS
896 001432 112737 000001 001115      MOVB   #1,$SERMAX        ;;ALLOW ONE ERROR PER TEST
897 001440 012737 001440 001106      MOV    #,$SLP/DR         ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
898 001446 012737 001446 001110      MOV    #,$SLP/RR         ;;SETUP THE ERROR LOOP ADDRESS
899                                     ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
900                                     ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
901 001454 013746 000004              MOV    @#ERRVEC,-(SP)      ;;SAVE ERROR VECTOR
902 001460 012737 001514 000004      MOV    #64$,@#ERRVEC    ;;SET UP ERROR VECTOR
903 001466 012737 177570 001140      MOV    #DSWR,SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
904 001474 012737 177570 001142      MOV    #DDISP,DISPLAY    ;;AND A HARDWARE DISPLAY REGISTER
905 001502 022777 177777 177430      CMP    #-1,@SWR         ;;TRY TO REFERENCE HARDWARE SWR
906 001510 001012                    BNE     66$                ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
907                                     ;;AND THE HARDWARE SWR IS NOT = -1
908 001512 000403                    BR      65$                ;;BRANCH IF NO TIMEOUT
909 001514 012716 001522 64$:         MOV    #65$,(SP)          ;;SET UP FOR TRAP RETURN
910 001520 000002                    RTI
911 001522 012737 000176 001140 65$:  MOV    #SWREG,SWR         ;;POINT TO SOFTWARE SWR
912 001530 012737 000174 001142      MOV    #DISPREG,DISPLAY
913 001536 012637 000004 66$:        MOV    (SP)+,@#ERRVEC      ;;RESTORE ERROR VECTOR
914
915 001542 005037 001202              CLR    $PASS             ;;CLEAR PASS COUNT
916 001546 132737 000200 001215      BITB   #APTSIZE,$ENVM    ;;TEST USER SIZE UNDER APT
917 001554 001403                    BEQ     67$                ;;YES,USE NON-APT SWITCH
918 001556 012737 001216 001140      MOV    #$$SWREG,SWR        ;;NO,USE APT SWITCH REGISTER
919 001564
920                                     .SBTTL  TYPE PROGRAM NAME
921                                     ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
922 001564 005227 177777              INC     #-1                ;;FIRST TIME?
923 001570 001045                    BNE     68$                ;;BRANCH IF NO
924 001572 022737 021712 000042      CMP     #SENDAD,@#42      ;;ACT-11?
925 001600 001441                    BEQ     68$                ;;BRANCH IF YES
926 001602 104401 001650              TYPE    ,69$              ;;TYPE ASCIZ STRING
927                                     .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
928 001606 005737 000042              TST     @#42              ;;ARE WE RUNNING UNDER XXDP/ACT?
929 001612 001012                    BNE     70$                ;;BRANCH IF YES
930 001614 123727 001214 000001      CMPB    $ENV,#1           ;;ARE WE RUNNING UNDER APT?
931 001622 001406                    BEQ     70$                ;;BRANCH IF YES
932 001624 023727 001140 000176      CMP     SWR,#SWREG        ;;SOFTWARE SWITCH REG SELECTED?
933 001632 001005                    BNE     71$                ;;BRANCH IF NO
934 001634 104406                    GTSWR                      ;;GET SOFT-SWR SETTINGS
935 001636 000403                    BR      71$
936 001640 112737 000001 001134 70$:  MOVB   #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
937 001646 71$:
938 001646 000416                    BR      68$                ;;GET OVER THE ASCIZ
939                                     ;;69$: .ASCIZ <CRLF>*KDF11-B AND DLV11-J TEST*<CRLF>
940 001704 68$:
941
942
943 001704                    LET $MXCNT := #1                    ;:ITERATION CNT FOR SCOPE ROUTINE
944 001704 012737 000001 024012      MOV     #1,$MXCNT
945
946 001712                    LET P1CNT := #0
947 001712 005037 001300                    CLR     P1CNT
948 001716                    LET NODLV := #0
```

```

949 001716 005037 001304
950 001722
951 001722 005037 001306
952 001726
953 001726 005037 001310
954 001732
955 001732 005037 001320
956 001736
957 001736 004737 016614
958 001742
959 001742 005037 001256
960 001746
961 001746 005037 001276
962 001752
963 001752 005037 001322
964 001756
965 001756 005037 001272
966 001762
967 001762 005037 001314
968 001766
969 001766 012737 176500 001312
970 001774
971 001774 005037 001324
972
973
974 002000 012706 001100
975 002004 004737 020540
976
977 002010
978 002010 013737 001256 001262
979 002016
980 002016 013737 001256 001264
981 002024 062737 000002 001264
982 002032
983 002032 013737 001256 001266
984 002040 062737 000004 001266
985 002046
986 002046 013737 001256 001270
987 002054 062737 000006 001270
988 002062
989 002062 005037 001102

      LET KDDL V := #0
      LET DLV211 := #0
      LET MASK := #0
      INLINE <JSR      PC,SIZE>
      LET DLADD := #0
      LET PHASE2 := #0
      LET KDF11B := #0
      LET CONST := #0
      LET KDFTST := #0
      LET DLBASE := #176500
      LET CONSOL := #0

      CLR      NODLV
      CLR      KDDL V
      CLR      DLV211
      CLR      MASK
      ;SIZE THE BUS
      JSR      PC,SIZE
      CLR      DLADD
      CLR      PHASE2
      CLR      KDF11B
      CLR      CONST
      CLR      KDFTST
      MOV      #176500,DLBASE
      ;SET UP INIT COND
      CLR      CONSOL

      LOOP:  MOV      #STACK,SP
             JSR      PC,CYCLE
             ;RESET STACK POINTER
             ;SETUP MODULE
             ;RCSR = DLADD + 0
      LET      RCSR := DLADD
             MOV      DLADD,RCSR
      LET      RBUF := DLADD + #2
             MOV      DLADD,RBUF
             ADD      #2,RBUF
      LET      TCSR := DLADD + #4
             MOV      DLADD,TCSR
             ADD      #4,TCSR
      LET      TBUF := DLADD + #6
             MOV      DLADD,TBUF
             ADD      #6,TBUF
      LET $TSTNM := #0
      CLR      $TSTNM

```



```

990 *****
991 *TEST 1 ADDRESSABILITY
992 *
993 * THIS TEST VERIFIES THAT ALL 4 REGISTERS OF THE CHANNEL UNDER TEST
994 * RESPOND TO THEIR ADDRESSES.
995 *
996 *****
997 TST1: SCOPE
998 MOV #1,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
999 LET ADRS := DLADD
1000 MOV DLADD,ADRS
1001 ; SET UP INTERRUPT
1002 MOV #INTSRV,ERRVEC
1003 MOV #PR7,ERRVEC+2
1004 LET I := #0
1005 CLR I
1006 REPEAT
1007 $50000:
1008 BGNSUB
1009 MOV #64,$,SLPERR
1010 ;CLEAR FLAG
1011 LET INTFLG := #0
1012 CLR INTFLG
1013 ;READ FLAG
1014 IF INTFLG NE #0 THEN
1015 TST INTFLG
1016 BNE +6
1017 JMP $50001
1018 ; FATAL ERROR
1019 ERRDF 1
1020 ENDIF
1021 $50001:
1022 ENDSUB
1023 LET I := I + #2
1024 ADD #2,I
1025 LET ADRS := DLADD + I
1026 MOV DLADD,ADRS
1027 ADD I,ADRS
1028 UNTIL I EQ #8.
1029 CMP I,#8.
1030 BNE $50000
1031
1032 BIT #177476,@RCSR ;CHECK THAT ALL UNUSED BITS ARE 0
1033 BEQ 1$
1034 ERROR 27 ;RCSR HAS UNUSED BITS SET
1035
1036 BIT #7400,@RBUF
1037 BEQ 2$
1038 ERROR 30 ;RBUF HAS UNUSED BITS SET
1039
1040 BIT #177476,@TCSR
1041 BEQ 3$
1042 ERROR 31 ;TCSR HAS UNUSED BITS SET
1043
1044
1045

```

```

1046
1047 002236 032777 177400 177024 3$: BIT #177400,@TBUF
1048 002244 001401 BEQ 4$
1049 002246 104032 ERROR 32 ;TBUF HAS ;UNUSED BITS SET
1050
1051 002250 012737 000006 000004 4$: MOV #6,ERRVEC ;RESTORE
1052 002256 005037 000006 CLR ERRVEC+2
1053
1054 ;*****
1055 ;* THE FOLLOWING 3 TESTS TEST ALL 'READ WRITE' BITS
1056 ;*****
1057
1058
1059 ;*****
1060 ;*TEST 2 BREAK - TCSR 0 SET, CLEAR, RESET
1061 ;* THIS BIT IS THE ONLY ONE IN THIS POSITION
1062 ;* THAT IS READ AND WRITE.
1063 ;*****
1064 002262 000004 TST2: SCOPE
1065 002264 012737 000002 001200 MOV #2,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1066
1067 IF #BIT03 NOTSETIN $USWR THEN
1068 002272 032737 000010 001220 BIT #BIT03,$USWR
1069 002300 001402 BEQ .+6
1070 002302 000137 002310 JMP $50002
1071 002306 ; BREAK NOT INSTALLED
1072 002306 000503 BR TST3 ;;EXIT THIS TEST
1073 002310 EXIT
1074 002310 ENDF
1075 002310 $50002:
1076 002310 IF #BIT00 SETIN CONSOL THEN ;IS THIS THE CONSOL
1077 002316 032737 000001 001324 BIT #BIT00,CONSOL
1078 002320 000137 002326 BNE .+6
1079 ; KDF11-B CONSOL DOES..
1080 002324 000474 BR TST3 ;;EXIT THIS TEST
1081 002324 EXIT
1082 002326 ENDF
1083 002326 $50003:
1084 ; SEE IF IT IS CLEAR
1085 BGNSUB
1086 002326 012737 002334 001110 MOV #64,$SLPERR
1087 002326 IF #BIT00 SETIN @TCSR THEN
1088 002334 032777 000001 176724 BIT #BIT00,@TCSR
1089 002342 001002 BNE .+6
1090 002344 000137 002352 JMP $50004
1091 ; BREAK DID NOT RESET IN TCSR
1092 ERRHRD 2
1093
1094 002350 ERROR 2
1095 002350 104002
1096 002352 ENDF
1097 002352 $50004:
1098 002352 ENDSUB
1099
1100 ; TRY TO SET BREAK BIT
1101 002352 BGNSUB
  
```


1102	002352	012737	002360	001110
1103	002360			
1104	002360	052777	000001	176700
1105				
1106	002366			
1107	002366	032777	000001	176672
1108	002374	001402		
1109	002376	000137	002404	
1110				
1111	002402			
1112	002402	104003		
1113	002404			
1114	002404			
1115	002404			
1116				
1117				
1118	002404			
1119	002404	012737	002412	001110
1120				
1121	002412			
1122	002412	042777	000001	176646
1123				
1124	002420			
1125	002420	032777	000001	176640
1126	002426	001C92		
1127	002430	000137	002436	
1128				
1129	002434			
1130	002434	104004		
1131	002436			
1132	002436			
1133	002436			
1134				
1135				
1136	002436			
1137	002436	012737	002444	001110
1138	002444			
1139	002444	032737	000002	001214
1140	002452	001406		
1141	002454	023727	001202	000001
1142	002462	002402		
1143	002464	000137	002516	
1144	002470			
1145				
1146	002470			
1147	002470	052777	000001	176570
1148				
1149	002476			
1150	002476	000005		
1151	002500			
1152	002500	032777	000001	176560
1153	002506	001002		
1154	002510	000137	002516	
1155				
1156	002514			
1157	002514	104005		

ERROR 3

ERROR 4

RESET

ERROR 5

```

MOV #64$, $LPERR
LET @TCSR := @TCSR SET.BY #BIT00
BIS #BIT00, @TCSR
; STUCK TO 0
IF #BIT00 NOTSETIN @TCSR THEN
  BIT #BIT00, @TCSR
  BEQ +6
  JMP $50005
; BREAK DID NOT SET IN TCSR
ERRHRD 3
ENDIF
$50005:
ENDSUB
; TRY TO CLEAR A SET BIT
BGNSUB
MOV #64$, $LPERR
LET @TCSR := @TCSR CLR.BY #BIT00
BIC #BIT00, @TCSR
; SHOULD HAVE CLEARED
IF #BIT00 SETIN @TCSR THEN
  BIT #BIT00, @TCSR
  BNE +6
  JMP $50006
; BREAK DID NOT CLEAR IN TCSR
ERRHRD 4
ENDIF
$50006:
ENDSUB
; NOW SEE IF RESET CLEARS IT
BGNSUB
MOV #64$, $LPERR
BIT #BIT01, $ENV
BEQ $50007
CMP $PASS, #1
BLT +6
JMP $50010
$50007:
LET @TCSR := @TCSR SET.BY #BIT00
BIS #BIT00, @TCSR
; ISSUE BUS RESET
BRESET
IF #BIT00 SETIN @TCSR THEN
  BIT #BIT00, @TCSR
  BNE +6
  JMP $50011
; BREAK DID NOT RESET IN TCSR
ERRHRD 5

```

KDF11-B / DLV11-J TEST MACY11 30(1046) 05-OCT-81 16:37 PAGE 24 K 2
CJDLAA.P11 29-SEP-81 11:42 T2 BREAK - TCSR 0 SET, CLEAR, RESET

SEQ 0023

1158 002516
1159 002516
1160 002516
1161 002516
1162 002516

ENDIF

ENDIF

\$50011:

\$50010:

ENDSUB


```

1163
1164
1165 *****
1166 :*TEST 3 XMITIE - TCSR 6 SET, CLEAR, RESET
1167 :*****
1168 TST3: SCOPE
1169 MOV #3,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1170 SETPRI #PR7 ; USE PRIORITY OF 7
1171
1172 ; SEE IF IT IS CLEAR
1173 BGNSUB
1174 MOV #65$,$LPERR
1175
1176 IF #BIT06 SETIN @TCSR THEN
1177 BIT #BIT06,@TCSR
1178 BNE .+6
1179 JMP $50012
1180 ; XMITIE DID NOT RESET IN TCSR
1181 ERRHRD 12
1182 ERROR 12
1183 ENDIF
1184 $50012:
1185 ENDSUB
1186
1187 ; TRY TO SET XMITIE BIT
1188 BGNSUB
1189 MOV #64$,$LPERR
1190 LET @TCSR := @TCSR SET.BY #BIT06
1191 BIS #BIT06,@TCSR
1192 ; STUCK TO 0
1193 IF #BIT06 NOTSETIN @TCSR THEN
1194 BIT #BIT06,@TCSR
1195 BEQ .+6
1196 JMP $50013
1197 ; XMIT DID NOT SET IN TCSR
1198 ERRHRD 13
1199 ERROR 13
1200 ENDIF
1201 $50013:
1202 ENDSUB
1203
1204 ; TRY TO CLEAR A SET BIT
1205 BGNSUB
1206 MOV #64$,$LPERR
1207
1208 LET @TCSR := @TCSR CLR.BY #BIT06
1209 BIC #BIT06,@TCSR
1210 ; SHOULD HAVE CLEARED
1211 IF #BIT06 SETIN @TCSR THEN
1212 BIT #BIT06,@TCSR
1213 BNE .+6
1214 JMP $50014
1215 ; XMIT DID NOT CLEAR IN TCSR
1216 ERRHRD 14
1217 ERROR 14
1218 ENDIF
  
```

1219	002650			
1220	002650			
1221				
1222				
1223	002650			
1224	002650	012737	002656	001110
1225	002656			
1226	002656	032737	000002	001214
1227	002664	001406		
1228	002666	023727	001202	000001
1229	002674	002402		
1230	002676	000137	002730	
1231	002702			
1232				
1233	002702			
1234	002702	052777	000100	176356
1235				
1236	002710			
1237	002710	000005		
1238	002712			
1239	002712	032777	000100	176346
1240	002720	001002		
1241	002722	000137	002730	
1242				
1243	002726			
1244	002726	104015		
1245	002730			
1246	002730			
1247	002730			
1248	002730			
1249	002730			

```

; NOW SEE IF RESET CLEARS IT
BGNSUB

```

```
IF #BIT01 NOTSETIN $ENV OR $PASS LT #1 THEN
```

RESET

ERROR 15

ENDIF

ENDSUB

\$50014:

MOV #64\$,\$LPERR

```

BIT      #BIT01,$ENVV
BEQ      $50015
CMP      $PASS,#1
BLT      .+6
JMP      $50016
$50015:

```

```

LET      @TCSR := @TCSR SET.BY #BIT06
          BIS      #BIT06,@TCSR
: ISSUE BUS RESET
BRESSET

```

```
IF      #BIT06 SET IN @TCSR THEN
        BIT      #BIT06,@TCSR
        BNE      .+6
        JMP      $50017
; XMIT DID NOT RESET IN TCSR
ERRHRD 15
```

ENDIF

\$50C17:

\$50016:

ENDSUB


```

1250
1251
1252
1253
1254
1255
1256 002730 000004
1257 002732 012737 000004 001200
1258
1259 002740
1260 002740 012737 002746 001110
1261
1262 002746
1263 002746 032777 000100 176306
1264 002754 001002
1265 002756 000137 002764
1266
1267 002762
1268 002762 104035
1269 002764
1270 002764
1271 002764
1272
1273
1274 002764
1275 002764 012737 002772 001110
1276 002772
1277 002772 052777 000100 176262
1278
1279 003000
1280 003000 032777 000100 176254
1281 003006 001402
1282 003010 000137 003016
1283
1284 003014
1285 003014 104036
1286 003016
1287 003016
1288 003016
1289
1290
1291 003016
1292 003016 012737 003024 001110
1293
1294 003024
1295 003024 042777 000100 176230
1296
1297 003032
1298 003032 032777 000100 176222
1299 003040 001002
1300 003042 000137 003050
1301
1302 003046
1303 003046 104037
1304 003050
1305 003050
  
```

```

*****
:TEST 4      RCVRIE - RCSR 6      SET, CLEAR, RESET
:          THIS BIT IS THE ONLY ONE IN THIS POSITION
:          THAT IS READ AND WRITE.
*****
  
```

```

TST4:  SCOPE
      MOV  #4,$TESTN      ;:SET TEST NUMBER IN APT MAIL BOX
                        ; SEE IF IT IS CLEAR
                        BGNSUB
                        MOV  #64$, $LPERR
                        IF  #BIT06 SETIN @RCSR THEN
                        BIT  #BIT06,@RCSR
                        BNE  +6
                        JMP  $50020
                        ; RCVRIE DID NOT RESET IN RCSR
                        ERRHRD 35
                        ENDIF
                        $50020:
                        ENDSUB
                        ; TRY TO SET RCVRIE BIT
                        BGNSUB
                        MOV  #64$, $LPERR
                        LET  @RCSR := @RCSR SET.BY #BIT06
                        BIS  #BIT06,@RCSR
                        ; STUCK TO 0
                        IF  #BIT06 NOTSETIN @RCSR THEN
                        BIT  #BIT06,@RCSR
                        BEQ  +6
                        JMP  $50021
                        ; RCVRIE DID NOT SET IN RCSR
                        ERRHRD 36
                        ENDIF
                        $50021:
                        ENDSUB
                        ; TRY TO CLEAR A SET BIT
                        BGNSUB
                        MOV  #64$, $LPERR
                        LET  @RCSR := @RCSR CLR.BY #BIT06
                        BIC  #BIT06,@RCSR
                        ; SHOULD HAVE CLEARED
                        IF  #BIT06 SETIN @RCSR THEN
                        BIT  #BIT06,@RCSR
                        BNE  +6
                        JMP  $50022
                        ; RCVRIE DID NOT CLEAR IN RCSR
                        ERRHRD 37
                        ENDIF
                        $50022:
  
```

ERROR 35

ERROR 36

ERROR 37

1306	003050				ENDSUB
1307					
1308					; NOW SEE IF RESET CLEARS IT
1309	003050				BGNSUB
1310	003050	012737	003056	001110	MOV #64\$, \$LPERR
1311	003056				IF #BIT01 NOTSETIN \$ENV OR \$PASS LT #1 THEN
1312	003056	032737	000002	001214	BIT #BIT01, \$ENV
1313	003064	001406			BEQ \$50023
1314	003066	023727	001202	000001	CMP \$PASS, #1
1315	003074	002402			BLT +6
1316	003076	000137	003130		JMP \$50024
1317	003102				\$50023:
1318					
1319	003102				LET @RCSR := @RCSR SET BY #BIT06
1320	003102	052777	000100	176152	BIS #BIT06, @RCSR
1321					; ISSUE BUS RESET
1322	003110				BRESET
1323	003110	000005			RESET
1324	003112				
1325	003112	032777	000100	176142	IF #BIT06 SETIN @RCSR THEN
1326	003120	001002			BIT #BIT06, @RCSR
1327	003122	000137	003130		BNE +6
1328					JMP \$50025
1329	003126				; RCVRIE DID NOT RESET IN RCSR
1330	003126	104040			ERRHRD 40
1331	003130				ENDIF
1332	003130				\$50025:
1333	003130				\$50024:
1334	003130				
1335	003130				ENDSUB


```

1336
1337
1338 *****
1339 :*TEST 5 TEST THAT XMITRDY - TCSR 7 - IS SET BY INIT
1340 :*****
1341 TST5: SCOPE #5,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
1342 MOV
1343 IF #BIT01 NOTSETIN $ENV OR $PASS LT #1 THEN
1344 003130 000004 000005 001200
1345 003132 012737 000002 001214 BIT #BIT01,$ENV
1346 003140 032737 001202 000001 BEQ $50026
1347 003146 001406 CMP $PASS,#1
1348 003150 023727 BLT .+6
1349 003156 002402 JMP $50027
1350 003160 000137 003230 $50026:
1351 003164 000005 INLINE <RESET> ;THIS SHOULD SET THE BIT
1352 003166 000005 RESET
1353 003166 032777 000200 176072 IF #BIT07 NOTSETIN @TCSR THEN
1354 003174 001402 BIT #BIT07,@TCSR
1355 003176 000137 003204 BEQ .+6
1356 003202 104042 ERRHRD 42
1357 003202 104042 ERROR 42
1358 003204 $50030:
1359 003204 LET @TCSR := @TCSR CLR.BY #BIT07 ;TRY TO CLR IT
1360 003204 042777 000200 176054 BIC #BIT07,@TCSR
1361 003204 032777 000200 176046 IF #BIT07 NOTSETIN @TCSR THEN
1362 003212 001402 BIT #BIT07,@TCSR
1363 003220 000137 003230 BEQ .+6
1364 003222 104011 ERRHRD 11
1365 003226 104011 ERROR 11
1366 003226 104011
1367 003226 104011
1368 003230
1369 003230
1370 003230
1371 003230
  
```

```

1372
1373
1374
1375
1376
1377
1378 003230 000004
1379 003232 012737 000006 001200
1380
1381 003240
1382 003240 023727 001324 000001
1383 003246 001402
1384 003250 000137 003256
1385 003254
1386 003254 000555
1387 003256
1388 003256
1389
1390 003256
1391 003256 005037 003602
1392 003262
1393 003262
1394
1395 003262
1396 003262 005037 003604
1397 003266
1398 003266 005037 003606
1399
1400
1401
1402
1403
1404
1405 003272
1406 003272 105077 175772
1407
1408
1409
1410 003276
1411 003276 010546
1412 003300 010605
1413 003302 162706 000010
1414 003306 010546
1415 003310 012745 177777
1416 003314 013745 001266
1417 003320 012745 000200
1418 003324 012745 000500
1419 003330 004737 016272
1420 003334 012605
1421 003336 010506
1422 003340 012605
1423
1424
1425 003342
1426 003342 103402
1427 003344 000137 003356

*****
*TEST 6      TEST THAT XMIT RDY - TCSR 7 - CLEARS
*            WHEN TBUF IS LOADED WITH A CHARACTER
*            AND THAT IT SETS WITHIN A REASONABLE AMOUNT OF TIME.
*****
TST6:  SCOPE
      MOV  #6,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX

                        IF CONSOL EQ #BIT00 THEN
                                CMP  CONSOL,#BIT00
                                BEQ  +6
                                JMP  $50032
                        EXIT
                        ;;EXIT THIS TEST
                        ENDIF

                        $50032:
                        LET PASS := #0      ;INIT COUNT OF TIMES THRU
                        CLR  PASS
                        LOOP
                                ; START OF LOOP
                                $50033:
                                ; MAX OF 2 TIMES THRU
                                LET ERRORFLAG := #0
                                CLR  ERRORFLAG
                                LET EXITFLAG := #0
                                CLR  EXITFLAG

                                ; LOAD TBUF WITH ONE CHARACTER
                                ; WAIT FOR READY TO SET
                                ; (SHOULD BE VERY SHORT WAIT
                                ; SINCE UART DOUBLE BUFFERS ITS INPUT)

                                ;SEND A CHARACTER
                                LET @TBUF :B= #0
                                CLRB  @TBUF
                                ;WAIT A MAXIMUM
                                ;OF 50 MSEC FOR
                                ;XMIT RDY TO SET IN TCSR
                                CALL TIMER IN <#500,#BIT07,TCSR,#-1>
                                MOV  R5,-(SP)
                                MOV  SP,R5
                                SUB  #10,SP
                                MOV  R5,-(SP)
                                MOV  #-1,-(R5)
                                MOV  TCSR,-(R5)
                                MOV  #BIT07,-(R5)
                                MOV  #500,-(R5)
                                JSR  PC,TIMER
                                MOV  (SP)+,R5
                                MOV  R5,SP
                                MOV  (SP)+,R5
                                ;TIMER RETURNS AN ERROR IF BIT DID
                                ;NOT MEET CONDITION WITHIN TIME LIMIT
                                IF.ERROR THEN
                                        BCS  +6
                                        JMP  $50035

```



```

1428                                     ;XMIT RDY DID NOT SET IN TCSR
1429 003350                               ERRHRD 66
1430 003350 104066                       ERROR 66
1431 003352 000137 003610               JMP TST7
1432 003356                               ENDIF
1433 003356                               $50035:
1434 003356 105077 175706               CLRB @TBUF ;SHIP 1'ST CHAR
1435 003362 105777 175700               TSTB @TCSR ;WAIT FOR RDY
1436 003366 100375                       BPL 1$
1437                                     ; LOAD TBUF WITH A SECOND CHARACTER (TO DOUBLE BUFFER)
1438                                     ; CHECK IMMEDIATELY THAT XMITRDY IS CLEAR
1439                                     ; AND THEN WAIT FOR IT TO SET
1440
1441                                     ;SEND SECOND CHARACTER
1442 003370                               LET @TBUF :B= #0
1443 003370 105077 175674               CLRB @TBUF
1444 003374 000240                       NOP ; GIVE IT TIME TO CLEAR
1445                                     ; XMITRDY SHOULD HAVE CLEARED UPON
1446                                     ; RECEIPT OF A CHARACTER
1447 003376                               IF #BIT07 SETIN @TCSR THEN
1448 003376 032777 000200 175662         BIT #BIT07,@TCSR
1449 003404 001002                       BNE +6
1450 003406 000137 003424               JMP $50036
1451                                     ; XMITRDY DID NOT CLEAR IN TCSR
1452                                     ; WILL RESULT IN ERR 67 IF FAILS 2X
1453 003412                               LET ERRORFLAG := #-1
1454 003412 012737 177777 003604         MOV #-1,ERRORFLAG
1455                                     ; DEFER ERROR TYPEOUT
1456
1457 003420                               ELSE
1458 003420 000137 003500               JMP $50037
1459 003424                               $50036:
1460                                     ;WAIT A MAXIMUM
1461                                     ;OF 100 MSEC FOR
1462                                     ;XMIT RDY TO SET IN TCSR
1463 003424 010546                       CALL TIMER IN <#1000,#BIT07,TCSR,#-1>
1464 003424 010605                       MOV R5,-(SP)
1465 003426 010605                       MOV SP,R5
1466 003430 162706 000010               SUB #10,SP
1467 003434 010546                       MOV R5,-(SP)
1468 003436 012745 177777               MOV #-1,-(R5)
1469 003442 013745 001266               MOV TCSR,-(R5)
1470 003446 012745 000200               MOV #BIT07,-(R5)
1471 003452 012745 001000               MOV #1000,-(R5)
1472 003456 004737 016272               JSR PC,TIMER
1473 003462 012605                       MOV (SP)+,R5
1474 003464 010506                       MOV R5,SP
1475 003466 012605                       MOV (SP)+,R5
1476 003470                               IF.ERROR THEN
1477 003470 103402                       BCS +6
1478 003472 000137 003500               JMP $50040
1479                                     ;XMIT RDY DID NOT SET IN TCSR
1480                                     ERRHRD 70
1481 003476 104070                       ERROR 70
1482 003500                               ENDIF
1483 003500                               $50040:

```

```

1484 003500
1485 003500
1486 003500
1487 003500 023727 003604 177777
1488 003506 001402
1489 003510 000137 003550
1490 003514
1491 003514 005237 003602
1492 003520
1493 003520 023727 003602 000001
1494 003526 003002
1495 003530 000137 003544
1496
1497
1498 003534
1499 003534 104067
1500 003536
1501 003536 012737 177777 003606
1502 003544
1503 003544
1504 003544
1505 003544 000137 003556
1506 003550
1507 003550
1508 003550 012737 177777 003606
1509 003556
1510 003556
1511 003556
1512 003556 023727 003606 177777
1513 003564 001002
1514 003566 000137 003574
1515 003572
1516 003572
1517 003572 000633
1518 003574
1519 003574
1520 003574 117700 175464
1521 003600
1522 003600 000403
1523 003602 000000
1524 003604 000000
1525 003606 000000
  
```

ERROR 67

LET RO :B= @RBUF

```

      BR      TST7
PASS: 0
ERRORFLAG: 0
EXITFLAG: 0
  
```

```

      ENDIF ; OF DEFERED ERROR CALL
      $50037:
      IF ERRORFLAG EQ #-1 THEN
        CMP      ERRORFLAG, #-1
        BEQ      .+6
        JMP      $50041
      LET PASS := PASS + #1
        INC      PASS
      IF PASS GT #1 THEN
        CMP      PASS, #1
        BGT      .+6
        JMP      $50042
      ; CALL ERROR IF 2ND TRY
      ; ON XMIT RDY NOT CLEARING
      ERRHRD 67
      LET EXITFLAG := #-1
        MOV      #-1, EXITFLAG
      ENDIF
    ELSE
      $50042:
      ; NO ERROR
      JMP      $50043
    $50041:
      LET EXITFLAG := #-1
        MOV      #-1, EXITFLAG
      ENDIF
    EXIF
      $50043:
      EXITFLAG EQ #-1
        CMP      EXITFLAG, #-1
        BNE      $50044
        JMP      $50034
      $50044:
      ENDLOOP
        BR      $50033
      $50034:
      ; CLR RBUF BY READING IT
        MOVB     @RBUF, RO
      EXIT ; SKIP AROUND FLAG WORDS
    ;;EXIT THIS TEST
  
```



```

1526
1527
1528
1529
1530
1531
1532 003610 000004
1533 003612 012737 000007 001200
1534
1535 003620
1536 003620 032737 000020 001220
1537 003626 001406
1538 003630 023727 001324 000001
1539 003636 001402
1540 003640 000137 003646
1541 003644
1542 003644
1543 003644 000462
1544 003646
1545 003646
1546
1547 003646
1548 003646 012737 003654 001110
1549
1550
1551 003654
1552 003654 105077 175410
1553
1554
1555
1556
1557 003660
1558 003660 010546
1559 003662 010605
1560 003664 162706 000010
1561 003670 010546
1562 003672 012745 177777
1563 003676 013745 001262
1564 003702 012745 000200
1565 003706 012745 000500
1566 003712 004737 016272
1567 003716 012605
1568 003720 010506
1569 003722 012605
1570
1571
1572 003724
1573 003724 103402
1574 003726 000137 003734
1575
1576 003732
1577 003732 104071
1578 003734
1579 003734
1580
1581 003734

*****
*TEST 7      TEST THAT OUTPUTTING A CHAR FROM TBUF (WITH WRAP CONNECTED)
*            RESULTS IN RCVRDONE SETTING WITHIN A REASONABLE AMOUNT OF TIME
*            AND THAT RESET CLEARS THE BIT.
*****
TST7:  SCOPE
      MOV  #7,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX

      IF #BIT04 NOTSET IN $USWR OR CONSOL EQ #BIT00 THE
      BIT  #BIT04,$USWR
      BEQ  $50045
      CMP  CONSOL,#BIT00
      BEQ  .+6
      JMP  $50046
      $50045:

      BR   TST10          EXIT
                        ;;EXIT THIS TEST
                        ENDIF

      $50046:

      BGNSUB
      MOV  #64,$LPERR
      ; SEND A CHARACTER AND LET IT WRAP AROUND

      LET @TBUF :B= #0
      CLRB @TBUF

      ; WAIT A MAXIMUM OF 50 MSEC
      ; FOR RCVR DONE TO SET IN
      ; RCSR
      CALL TIMER IN <#500,#BIT07,RCSR,#-1>
      MOV  R5,-(SP)
      MOV  SP,R5
      SUB  #10,SP
      MOV  R5,-(SP)
      MOV  #-1,-(R5)
      MOV  RCSR,-(R5)
      MOV  #BIT07,-(R5)
      MOV  #500,-(R5)
      JSR  PC,TIMER
      MOV  (SP)+,R5
      MOV  R5,SP
      MOV  (SP)+,R5

      ;DIDN'T SET IN TIME
      IF .ERROR THEN
      BCS  .+6
      JMP  $50047
      ; RCVRDONE DID NOT SET IN RCSR
      ERRHRD 71

      ENDIF

      $50047:

      ENDSUB
  
```

```

1582
1583 003734                                BGNSUB
1584 003734 012737 003742 001110          MOV    #64$, $LPERR
1585                                     ; NOW THAT IT IS SET SEE IF IT CAN BE RESET
1586 003742                                IF #BIT01 NOTSETIN $ENV OR $PASS LT #1 THEN
1587 003742 032737 000002 001214          BIT     #BIT01, $ENV
1588 003750 001406                          BEQ     $50050
1589 003752 023727 001202 000001          CMP     $PASS, #1
1590 003760 002402                          BLT     .+6
1591 003762 000137 003770                          JMP     $50051
1592 003766                                $50050:
1593 003766                                BRESET
1594 003766 000005                                RESET
1595 003770                                ENDIF
1596 003770
1597
1598 003770                                IF #BIT07 SETIN @RCSR THEN
1599 003770 032777 000200 175264          BIT     #BIT07, @RCSR
1600 003776 001002                          BNE     .+6
1601 004000 000137 004006                          JMP     $50052
1602                                     ; RCVRDONE DID NOT RESET IN RCSR.
1603 004004                                ERRHRD 72
1604 004004 104072                                ERROR 72
1605 004006                                ENDIF
1606 004006                                $50052:
1607 004006                                ;READ RBUF TO CLR IT
1608 004006 117700 175252          LET R0 :B= @RBUF          MOVB    @RBUF, R0
1609 004012                                ENDSUB
  
```



```

1610
1611
1612
1613
1614 004012 000004
1615 004014 012737 000010 001200
1616
1617 004022
1618 004022 032737 000020 001220
1619 004030 001406
1620 004032 023727 001324 000001
1621 004040 001402
1622 004042 000137 004050
1623 004046
1624 004046
1625 004046 000456
1626 004050
1627 004050
1628
1629 004050
1630 004050 012737 004056 001110
1631
1632
1633
1634 004056
1635 004056 105077 175206
1636
1637
1638
1639 004062
1640 004062 010546
1641 004064 010605
1642 004066 162706 000010
1643 004072 010546
1644 004074 012745 177777
1645 004100 013745 001262
1646 004104 012745 000200
1647 004110 012745 000500
1648 004114 004737 016272
1649 004120 012605
1650 004122 010506
1651 004124 012605
1652
1653 004126
1654 004126 103402
1655 004130 000137 004136
1656
1657 004134
1658 004134 104025
1659
1660 004136
1661 004136
1662 004136
1663
1664
1665

```

 :TEST 10 TEST THAT RCVRDONE IS CLEARED BY SETTING READER ENABLE

 TST10: SCOPE
 MOV #10,\$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

 IF #BIT04 NOTSETIN \$USWR OR CONSOL EQ #BIT00 THE
 BIT #BIT04,\$USWR
 BEQ \$50053
 CMP CONSOL,#BIT00
 BEQ +6
 JMP \$50054
 \$50053:
 EXIT
 BR TST11 ;;EXIT THIS TEST
 ENDIF
 \$50054:
 BGNSUB
 MOV #64,\$SLPERR
 ; OUTPUT A CHARACTER AND WAIT FOR XMITRDY TO SET.
 ; OUTPUT A CHARACTER
 LET @TBUF :B= #0
 CLRB @TBUF
 ; WAIT MAXIMUM OF 500 MSEC
 ; FOR RCVRDONE TO SET IN
 ; RCSR
 CALL TIMER IN <#500,#BIT07,RCSR,#-1>
 MOV R5,-(SP)
 MOV SP,R5
 SUB #10,SP
 MOV R5,-(SP)
 MOV #-1,-(R5)
 MOV RCSR,-(R5)
 MOV #BIT07,-(R5)
 MOV #500,-(R5)
 JSR PC,TIMER
 MOV (SP)+,R5
 MOV R5,SP
 MOV (SP)+,R5
 ; DID IT BECOME READY?
 IF.ERROR THEN
 BCS +6
 JMP \$50055
 ;RCVRDONE DID NOT SET IN RCSR
 ERRHRD 25
 ; SET IT BACK TO CONTINUE
 ENDIF
 \$50055:
 ENDSUB
 ;NOW THAT IT IS SET LETS SEE IF SETTING
 ;READER ENABLE CLEARS RCVRDONE

```

1666
1667 004136
1668 004136 023727 001322 000001
1669 004144 001402
1670 004146 000137 004154
1671 004152
1672 004152 000414
1673
1674 004154
1675 004154
1676 004154
1677 004154 052777 000001 175100
1678 004162
1679 004162 032777 000200 175072
1680 004170 001002
1681 004172 000137 004200
1682
1683 004176
1684 004176 104026
1685
1686 004200
1687 004200
1688 004200
1689 004200 117700 175060
  
```

LET R0 :B= @RBUF

ERROR 26

BR TST11

```

IF KDF11B EQ #BIT00 THEN
  CMP KDF11B,#BIT00
  BEQ +6
  JMP $50056
  ; KDF11-B SLU'S DON'T..
  ; ..HAVE 'READER RUN'
EXIT
::EXIT THIS TEST
ENDIF
$50056:
LET @RCSR := @RCSR SET.BY #BIT00
BIS #BIT00,@RCSR
IF #BIT07 SET IN @RCSR THEN
  BIT #BIT07,@RCSR
  BNE +6
  JMP $50057
;RCVRDONE DID NOT CLEAR IN RCSR
ERRHRD 26
; SET IT BACK TO CONTINUE
ENDIF
$50057:
;READ RBUF TO CLR IT
MOVB @RBUF,R0
  
```



```

1690
1691
1692
1693
1694 004204 000004
1695 004206 012737 000011 001200
1696
1697 004214
1698 004214 032737 000020 001220
1699 004222 001406
1700 004224 023727 001324 000001
1701 004232 001402
1702 004234 000137 004242
1703 004240
1704 004240
1705 004240 000444
1706 004242
1707 004242
1708
1709 004242
1710 004242 012737 004250 001110
1711
1712
1713
1714 004250
1715 004250 105077 175014
1716
1717
1718
1719 004254
1720 004254 010546
1721 004256 010605
1722 004260 162706 000010
1723 004264 010546
1724 004266 012745 177777
1725 004272 013745 001262
1726 004276 012745 000200
1727 004302 012745 000500
1728 004306 004737 016272
1729 004312 012605
1730 004314 010506
1731 004316 012605
1732
1733 004320
1734 004320 103402
1735 004322 000137 004330
1736
1737 004326
1738 004326 104073
1739
1740 004330
1741 004330
1742 004330
1743
1744
1745

;*****
;TEST 11 TEST THAT RCVRDONE IS CLEARED BY READING RBUF
;*****
TST11: SCOPE
MOV #11,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX

IF #BIT04 NOTSETIN $USWR OR CONSOL EQ #BIT00 THE
BIT #BIT04,$USWR
BEQ $50060
CMP CONSOL,#BIT00
BEQ +6
JMP $50061
$50060:

EXIT
BR TST12 ;;EXIT THIS TEST
ENDIF

$50061:

BGNSUB
MOV #64,$SLPERR
; OUTPUT A CHARACTER AND WAIT FOR RCVRDONE TO SET.

; OUTPUT A CHARACTER
LET @TBUF :B= #0
CLRB @TBUF
; WAIT MAXIMUM OF 500 MSEC
; FOR RCVRDONE TO SET IN
; RCSR
CALL TIMER IN <#500,#BIT07,RCSR,#-1>
MOV R5,-(SP)
MOV SP,R5
SUB #10,SP
MOV R5,-(SP)
MOV #-1,-(R5)
MOV RCSR,-(R5)
MOV #BIT07,-(R5)
MOV #500,-(R5)
JSR PC,TIMER
MOV (SP)+,R5
MOV R5,SP
MOV (SP)+,R5
; DID IT BECOME READY?
IF.ERROR THEN
BCS +6
JMP $50062
;RCVRDONE DID NOT SET IN RCSR
ERRHRD 73

; SET IT BACK TO CONTINUE
ENDIF
$50062:

ENDSUB

; NOW THAT IT IS SET LETS SEE IF READING THE
; BUFFER CLEARS RCVRDONE.

```

1746
 1747
 1748 004330
 1749 004330 117700 174730
 1750
 1751 004334
 1752 004334 032777 000200 174720
 1753 004342 001002
 1754 004344 000137 004352
 1755
 1756 004350
 1757 004350 104074
 1758
 1759 004352
 1760 004352

ERROR 74

```

;READ BUFFER
LET R0 :B= @RBUF
      MOVB @RBUF,R0

IF #BIT07 SETIN @RCSR THEN
      BIT #BIT07,@RCSR
      BNE +6
      JMP $50063
;RCVRDONE DID NOT CLEAR IN RCSR
ERRHRD 74

; SET IT BACK TO CONTINUE
ENDIF
$50063:
  
```



```
1761
1762
1763 *****
1764 *TEST 12      TEST THE OVERRUN & ERROR BITS - RBUF 14
1765 *****
1766 TST12: SCOPE 000004 000012 001200
1767      MOV      #12,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
1768
1769      IF CONSOL EQ #0 THEN
1770          TST      CONSOL
1771          BEQ      .+6
1772          JMP      $50064
1773      IF #BIT04 SET IN $USWR THEN
1774          BIT      #BIT04,$USWR
1775          BNE      .+6
1776          JMP      $50065
1777      ;; NULL ---EXECUTE TEST
1778      ELSE
1779          JMP      $50066
1780          $50065:
1781          JMP      TST13
1782      ENDIF
1783      $50066:
1784      ELSE
1785          JMP      $50067
1786          $50064:
1787          JMP TST13
1788      ENDIF
1789      $50067:
1790
1791
1792      BGNSUB
1793      MOV      #64,$LPERR
1794      :OUTPUT 2 CHARACTERS
1795      :THIS SHOULD AN CAUSE OVERRUN ERROR.
1796
1797      :OUTPUT 1 CHARACTER
1798      LET @TBUF :B= #0
1799      CLRB      @TBUF
1800
1801      CALL TIMER IN <#500,#BIT07,RCSR,#-1>
1802      MOV      R5,-(SP)
1803      MOV      SP,R5
1804      SUB      #10,SP
1805      MOV      R5,-(SP)
1806      MOV      #-1,-(R5)
1807      MOV      RCSR,-(R5)
1808      MOV      #BIT07,-(R5)
1809      MOV      #500,-(R5)
1810      JSR      PC,TIMER
1811      MOV      (SP)+,R5
1812      MOV      R5,SP
1813      MOV      (SP)+,R5
1814
1815      :DID IT SET IN TIME?
1816      IF.ERROR THEN
```

1817 004506 103402
 1818 004510 000137 004516
 1819
 1820 004514
 1821 004514 104016
 1822 004516
 1823 004516
 1824
 1825
 1826 004516
 1827 004516 105077 174546
 1828
 1829 004522
 1830 004522 010546
 1831 004524 010605
 1832 004526 162706 000002
 1833 004532 010546
 1834 004534 012745 000454
 1835 004540 004737 016524
 1836 004544 012605
 1837 004546 010506
 1838 004550 012605
 1839
 1840
 1841 004552
 1842 004552 017704 174506
 1843
 1844
 1845 004556
 1846 004556 032704 040000
 1847 004562 001402
 1848 004564 000137 004574
 1849
 1850 004570
 1851 004570 104101
 1852
 1853
 1854 004572
 1855 004572 000502
 1856 004574
 1857 004574
 1858 004574
 1859
 1860
 1861 004574
 1862 004574 012737 004602 001110
 1863 004602
 1864 004602 032704 104000
 1865 004606 001402
 1866 004610 000137 004620
 1867
 1868
 1869 004614
 1870 004614 104102
 1871
 1872

ERROR 16

ENDIF

BCS +6
 JMP \$50070
 ;RCVRDONE DID NOT SET IN RCSR
 ERRHRD 16

\$50070:

;OUTPUT 2ND CHARACTER
 LET @TBUF :B= #0
 CLRB @TBUF
 ;LET OVERRUN HAPPEN
 WAITMS 300.
 MOV R5, -(SP)
 MOV SP, R5
 SUB #2, SP
 MOV R5, -(SP)
 MOV #300, -(R5)
 JSR PC, WAIT
 MOV (SP)+, R5
 MOV R5, SP
 MOV (SP)+, R5

;READ BUFFER AND ERROR BITS
 LET R4 := @RBUF
 MOV @RBUF, R4

;IT DIDN'T SET
 IF #BIT14 NOTSETIN R4 THEN
 BIT #BIT14, R4
 BEQ +6
 JMP \$50071
 ;ORERR DID NOT SET IN RBUF
 ERRHRD 101

ERROR 101

;NO USE COMPOUNDING ERRORS
 EXIT

BR TST13 ;:EXIT THIS TEST
 ENDIF

\$50071:

ENDSUB

;NOW SEE IF ERROR BIT SET WITH OVERRUN ERROR:
 BGNSUB

MOV #64\$, \$LPERR
 IF #ERROR NOTSETIN R4 THEN
 BIT #ERROR, R4
 BEQ +6
 JMP \$50072

;ERROR DID NOT SET IN RBUF
 ERRHRD 102

ERROR 102

;--WHEN ORERR SET.


```

1873
1874 004616
1875 004616 000470
1876 004620
1877 004620
1878 004620
1879
1880 004620
1881 004620 012737 004626 001110
1882
1883
1884 004626
1885 004626 032777 040000 174430
1886 004634 001402
1887 004636 000137 004646
1888
1889
1890 004642
1891 004642 104103
1892
1893 004644
1894 004644 000455
1895 004646
1896 004646
1897 004646
1898
1899 004646
1900 004646 012737 004654 001110
1901
1902
1903
1904
1905
1906 004654
1907 004654 105077 174410
1908 004660
1909 004660 010546
1910 004662 010605
1911 004664 162706 000010
1912 004670 010546
1913 004672 012745 177777
1914 004676 013745 001262
1915 004702 012745 000200
1916 004706 012745 000500
1917 004712 004737 016272
1918 004716 012605
1919 004720 010506
1920 004722 012605
1921
1922
1923 004724
1924 004724 103402
1925 004726 000137 004734
1926
1927 004732
1928 004732 104020
  
```

```

;GET OUT NOW.
EXIT
;:EXIT THIS TEST
ENDIF
$50072:
ENDSUB
BGNSUB
MOV #64,$LPERR
;CHECK REAL RBUF TO SEE IF ORERR IS STILL SET.
IF #BIT14 NOTSETIN @RBUF THEN
  BIT #BIT14,@RBUF
  BEQ +6
  JMP $50073
;READING RBUF CLEARED ORERR.
ERRHRD 103
;SKIP REST OF TEST
EXIT
;:EXIT THIS TEST
ENDIF
$50073:
ENDSUB
BGNSUB
MOV #64,$LPERR
;READING RBUF ABOVE SHOULD ENABLE ERROR TO BE CLEARED
;BY NEXT TRANSFER.
;NOW SEE IF THEY CLEAR WHEN ANOTHER CHAR. IS RECEIVED
;SEND A CHARACTER AROUND.
LET @TBUF :B= #0
CLRB @TBUF
CALL TIMER IN <#500,#BIT07,RCSR,#-1>
MOV R5,-(SP)
MOV SP,R5
SUB #10,SP
MOV R5,-(SP)
MOV #-1,-(R5)
MOV RCSR,-(R5)
MOV #BIT07,-(R5)
MOV #500,-(R5)
JSR PC,TIMER
MOV (SP)+,R5
MOV R5,SP
MOV (SP)+,R5
;DID IT SET IN TIME?
IF .ERROR THEN
  BCS +6
  JMP $50074
;RCVRDONE DID NOT SET IN RCSR
ERRHRD 20
  
```

ERROR 20

```

1929 004734
1930 004734
1931
1932 004734
1933 004734 032777 040000 174322
1934 004742 001002
1935 004744 000137 004754
1936
1937 004750
1938 004750 104104 ERROR 104
1939
1940
1941
1942 004752
1943 004752 000412 BR TST13
1944 004754
1945 004754
1946
1947 004754
1948 004754 032777 104000 174302
1949 004762 001002
1950 004764 000137 004772
1951
1952 004770
1953 004770 104105 ERROR 105
1954
1955 004772
1956 004772
1957 004772 LET R0 :B= @RBUF
1958 004772 117700 174266
1959 004776
1960 004776
1961 004776 000400 BR TST13
  
```

```

ENDIF
$50074:
IF #BIT14 SETIN @RBUF THEN
  BIT #BIT14,@RBUF
  BNE +6
  JMP $50075
;ORERR DID NOT CLEAR IN RBUF
ERRHRD 104

;-AFTER RECEIVING ANOTHER CHAR
;SKIP AROUND REST
EXIT
;;EXIT THIS TEST
ENDIF
$50075:
IF #ERROR SETIN @RBUF THEN
  BIT #ERROR,@RBUF
  BNE +6
  JMP $50076
;ERROR DID NOT CLEAR IN RBUF
ERRHRD 105

ENDIF
$50076:
;READ RBUF TO CLR IT
MOVB @RBUF,R0
ENDSUB
EXIT
;;EXIT THIS TEST
  
```



```

1962
1963
1964
1965
1966
1967
1968
1969
1970
1971 005000 000004
1972 005002 012737 000013 001200
1973
1974 005010
1975 005010 005037 017640
1976
1977
1978 005014
1979 005014 013703 001260
1980
1981 005020
1982 005020 062703 000004
1983
1984 005024 012723 017632
1985 005030 012713 000340
1986 005034
1987 005034 012737 005042 001110
1988
1989 005042
1990 005042 010546
1991 005044 010605
1992 005046 162706 000010
1993 005052 010546
1994 005054 012745 177777
1995 005060 013745 001266
1996 005064 012745 000200
1997 005070 012745 000500
1998 005074 004737 016272
1999 005100 012605
2000 005102 010506
2001 005104 012605
2002
2003
2004 005106
2005 005106 042777 000100 174152
2006
2007
2008 005114
2009
2010
2011 005126
2012 005126 052777 000100 174132
2013
2014 005134
2015 005134 010546
2016 005136 010605
2017 005140 162706 000010

*****
*TEST 13 TRANSMITTER INTERRUPT LOGIC TEST
* LOGICALLY THIS IS 4 SEPARATE TESTS
* A) DOES TRANSMITTER INTERRUPT LOGIC WORK
* B) AT PRIORITY OF 0
* C) AND ONLY ONCE
* D) BUT NOT WITH INTERRUPT ENABLE CLEAR
*****
TST13: SCOPE
MOV #13,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
;;CLEAR 'INTERRUPT OCCURED' FLAG
LET INTFLG := #0
CLR INTFLG

;;GET VECTOR ADDRESS
LET R3 := DLVEC
MOV DLVEC,R3
;;FOR THE TRANSMITTER
LET R3 := R3 + #4
ADD #4,R3
;;SET VECTOR TO POINT TO TRANS.SRV AT PRI

BGNSUB
MOV #64,$SLPERR
;; MAKE SURE THAT TRANSMITTER READY IS SET
CALL TIMER IN <#500,#BIT07,TCSR,#-1>
MOV R5,-(SP)
MOV SP,R5
SUB #10,SP
MOV R5,-(SP)
MOV #-1,-(R5)
MOV TCSR,-(R5)
MOV #BIT07,-(R5)
MOV #500,-(R5)
JSR PC,TIMER
MOV (SP)+,R5
MOV R5,SP
MOV (SP)+,R5

;;CLEAR INTERRUPT ENABLE
LET @TCSR := @TCSR CLR.BY #BIT06
BIC #BIT06,@TCSR

;;SET IT TO 0

SETPRI #PRO

;;NOW SET I.E. BIT
LET @TCSR := @TCSR SET.BY #BIT06
BIS #BIT06,@TCSR

CALL TIMER IN <#500,#1,INTFLG,#-1>
MOV R5,-(SP)
MOV SP,R5
SUB #10,SP

```

2018 005144 010546
 2019 005146 012745 177777
 2020 005152 012745 017640
 2021 005156 012745 000001
 2022 005162 012745 000500
 2023 005166 004737 016272
 2024 005172 012605
 2025 005174 010506
 2026 005176 012605
 2027
 2028
 2029 005200
 2030 005200 103402
 2031 005202 000137 005210
 2032
 2033 005206
 2034 005206 104106
 2035 005210
 2036 005210
 2037
 2038
 2039 005210
 2040 005210 010546
 2041 005212 010605
 2042 005214 162706 000002
 2043 005220 010546
 2044 005222 012745 000764
 2045 005226 004737 016524
 2046 005232 012605
 2047 005234 010506
 2048 005236 012605
 2049
 2050
 2051 005240
 2052 005240 023727 017640 000001
 2053 005246 003002
 2054 005250 000137 005256
 2055
 2056 005254
 2057 005254 104107
 2058 005256
 2059 005256
 2060 005256
 2061
 2062 005256
 2063 005256 012737 005264 001110
 2064
 2065 005264
 2066 005264 042777 000100 173774
 2067
 2068 005272
 2069 005272 005037 017640
 2070
 2071
 2072 005276
 2073 005276 005077 173766

ERROR 106

ERROR 107

```

MOV R5, -(SP)
MOV #-1, -(R5)
MOV #INTFLG, -(R5)
MOV #1, -(R5)
MOV #500, -(R5)
JSR PC, TIMER
MOV (SP)+, R5
MOV R5, SP
MOV (SP)+, R5

;DID IT SET IN TIME?
IF.ERROR THEN
  BCS .+6
  JMP $50077
; INTERRUPT DID NOT OCCUR
ERRHRD 106
ENDIF

$50077:
;LET POSSIBLE 2'ND INTERR OCCUR
WAITMS 500.
MOV R5, -(SP)
MOV SP, R5
SUB #2, SP
MOV R5, -(SP)
MOV #500, -(R5)
JSR PC, WAIT
MOV (SP)+, R5
MOV R5, SP
MOV (SP)+, R5

;DID EXACTLY 1 INTERRUPT OCCUR
IF INTFLG GT #1 THEN
  CMP INTFLG, #1
  BGT .+6
  JMP $50100
; TRANSMITTER INTERRUPTED TWICE
ERRHRD 107
ENDIF

$50100:
ENDSUB
; INTERRUPT WITHOUT INTERRUPT ENABLE SET
BGNSUB
MOV #64$, $LPERR
; CLEAR INTERRUPT ENABLE
LET @TCSR := @TCSR CLR.BY #BIT06
BIC #BIT06, @TCSR
; CLEAR 'INTERRUPT OCCURED' FLAG
LET INTFLG := #0
CLR INTFLG
; NO INTERRUPTS SHOULD OCCUR, PSW STILL AT 0.
; DARE IT TO HAPPEN
LET @TBUF := #0
CLR @TBUF
  
```



```

2074                                     ;SEE IF INT FLAG EVER SETS
2075 005302                                     CALL TIMER IN <#1000,#1,#INTFLG,#-1>
2076 005302 010546                                     MOV      R5,-(SP)
2077 005304 010605                                     MOV      SP,R5
2078 005306 162706 000010                                     SUB      #10,SP
2079 005312 010546                                     MOV      R5,-(SP)
2080 005314 012745 177777                                     MOV      #-1,-(R5)
2081 005320 012745 017640                                     MOV      #INTFLG,-(R5)
2082 005324 012745 000001                                     MOV      #1,-(R5)
2083 005330 012745 001000                                     MOV      #1000,-(R5)
2084 005334 004737 016272                                     JSR      PC,TIMER
2085 005340 012605                                     MOV      (SP)+,R5
2086 005342 010506                                     MOV      R5,SP
2087 005344 012605                                     MOV      (SP)+,R5
2088
2089 005346 103403                                     ;C BIT SHOULD BE SET
2090                                     ;INTERR STILL OCCURED WITH IE DISABLED
2091 005350                                     ERRHRD 110
2092 005350 104110                                     ERROR 110
2093 005352                                     LET R0 :B= @RBUF
2094 005352 117700 173706                                     ;READ RBUF TO CLR IT
2095 005356                                     MOV      @RBUF,R0
2096
2097                                     1$:
2098                                     ;*****
2099                                     ;*TEST 14 RECEIVER INTERRUPT LOGIC TEST
2100                                     ;* THIS TEST COVERS ALL OF THE RECEIVER
2101                                     ;* SIDE OF THE INTERRUPT LOGIC IN
2102                                     ;* CHARACTER MODE.
2103                                     ;*****
2104 005356 000004 TST14: SCOPE
2105 005360 012737 000014 001200 MOV      #14,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
2106 005366 032737 000020 001220 IF #BIT04 NOTSETIN $USWR OR CONSOL EQ #BIT00 THEN
2107 005374 001406 BIT      #BIT04,$USWR
2108 005376 023727 001324 000001 BEQ      $50101
2109 005404 001402 CMP      CONSOL,#BIT00
2110 005406 000137 005416 BEQ      +6
2111 005412 000137 006034 JMP      $50102
2112 005412 000137 006034 JMP      TST15 ; EXIT TEST
2113 005416 ENDIF
2114 005416
2115                                     $50101:
2116                                     ;CLEAR INTERRUPT OCCURED FLAG
2117 005416 SETVEC DLVEC,#INTSRV,#PR7 ;SET UP RECEIVER INTER.VECTOR
2118 005416 010146 MOV      R1,-(SP)
2119 005420 013701 001260 MOV      DLVEC,R1
2120 005424 012721 017632 MOV      #INTSRV,(R1)+
2121 005430 012711 000340 MOV      #PR7,(R1)
2122 005434 012601 MOV      (SP)+,R1
2123                                     ;PRIORITY 0 AND MULTIPLE INTERRUPT TEST.-RCVRIE
2124 005436 BGNSUB
2125 005436 012737 005444 001110 MOV      #64$,$LPERR
2126 005444 LET INTFLG := #0
2127 005444 005037 017640 CLR      INTFLG
2128                                     ;CLEAR INTERRUPTS
2129 005450 LET @RCSR := @RCSR CLR.BY #BIT06

```

2130	005450	042777	000100	173604	
2131					
2132					
2133	005456				SETPRI #PRO
2134					
2135					
2136	005470				
2137	005470	105077		173574	
2138					
2139					
2140					
2141	005474				
2142	005474	010546			
2143	005476	010605			
2144	005500	162706		000010	
2145	005504	010546			
2146	005506	012745		177777	
2147	005512	013745		001266	
2148	005516	012745		000200	
2149	005522	012745		000500	
2150	005526	004737		016272	
2151	005532	012605			
2152	005534	010506			
2153	005536	012605			
2154					
2155	005540				
2156	005540	052777	000100	173514	
2157					
2158					
2159	005546				
2160	005546	010546			
2161	005550	010605			
2162	005552	162706		000010	
2163	005556	010546			
2164	005560	012745		177777	
2165	005564	012745		017640	
2166	005570	012745		000001	
2167	005574	012745		000500	
2168	005600	004737		016272	
2169	005604	012605			
2170	005606	010506			
2171	005610	012605			
2172					
2173					
2174	005612				
2175	005612	103402			
2176	005614	000137		005622	
2177					
2178	005620				
2179	005620	104111			ERROR 111
2180	005622				
2181	005622				
2182					
2183	005622				
2184	005622	010546			
2185	005624	010605			

```

      BIC      #BIT06,@RCSR
;CHANGE PRIORITY
;...TO 0

;SEND A CHARACTER
LET @TBUF := #0
      CLRB    @TBUF
;WAIT A MAXIMUM
;OF 50 MSEC FOR
;XMIT RDY TO SET IN TCSR
CALL TIMER IN <#500,#BIT07,TCSR,#-1>
      MOV     R5,-(SP)
      MOV     SP,R5
      SUB     #10,SP
      MOV     R5,-(SP)
      MOV     #-1,-(R5)
      MOV     TCSR,-(R5)
      MOV     #BIT07,-(R5)
      MOV     #500,-(R5)
      JSR     PC,TIMER
      MOV     (SP)+,R5
      MOV     R5,SP
      MOV     (SP)+,R5
;SET INTERRUPT ENABLE
LET @RCSR := @RCSR SET.BY #BIT06
      BIS     #BIT06,@RCSR
;LET IT COME IN.

CALL TIMER IN <#500,#1,#INTFLG,#-1>
      MOV     R5,-(SP)
      MOV     SP,R5
      SUB     #10,SP
      MOV     R5,-(SP)
      MOV     #-1,-(R5)
      MOV     #INTFLG,-(R5)
      MOV     #1,-(R5)
      MOV     #500,-(R5)
      JSR     PC,TIMER
      MOV     (SP)+,R5
      MOV     R5,SP
      MOV     (SP)+,R5

;DID IT SET IN TIME?
IF.ERROR THEN
      BCS     +6
      JMP     $50103
;INTERRUPT DID NOT OCCUR
ERRHRD 111

ENDIF

$50103:
;LET POSSIBLE 2'ND INTERR OCCUR
WAITMS 500
      MOV     R5,-(SP)
      MOV     SP,R5

```


2186	005626	162706	000002	
2187	005632	010546		
2188	005634	012745	000500	
2189	005640	004737	016524	
2190	005644	012605		
2191	005646	010506		
2192	005650	012605		
2193				
2194				
2195	005652			
2196	005652	023727	017640	000001
2197	005660	003002		
2198	005662	000137	005670	
2199				
2200	005666			
2201	005666	104112		
2202	005670			
2203	005670			
2204	005670			
2205				
2206				
2207				
2208				
2209				
2210				
2211				
2212	005670			
2213	005670	012737	005676	001110
2214				
2215				
2216	005676			
2217	005676	042777	000100	173356
2218				
2219	005704			
2220	005704	005037	017640	
2221				
2222	005710			
2223	005710	105077	173354	
2224				
2225				
2226	005714			
2227	005714	010546		
2228	005716	010605		
2229	005720	162706	000010	
2230	005724	010546		
2231	005726	012745	005000	
2232	005732	012745	017640	
2233	005736	012745	000001	
2234	005742	012745	000500	
2235	005746	004737	016272	
2236	005752	012605		
2237	005754	010506		
2238	005756	012605		
2239				
2240	005760			
2241	005760	103402		

ERROR 112

; INTERRUPT WITHOUT IE SET.

```

SUB      #2,SP
MOV      R5,-(SP)
MOV      #500,-(R5)
JSR      PC,WAIT
MOV      (SP)+,R5
MOV      R5,SP
MOV      (SP)+,R5

;EXACTLY 1 INTERRUPT?
IF INTFLG GT #1 THEN
    CMP    INTFLG,#1
    BGT    +6
    JMP    $50104
;RECEIVER INTERRUPTED TWICE
ERRHRD 112

ENDIF
$50104:
ENDSUB

; INTERRUPT WITHOUT IE SET.
BGNSUB
    MOV    #64$,SLPERR

;CLEAR INTERRUPT
LET @RCSR := @RCSR CLR.BY #BIT06
    BIC    #BIT06,@RCSR
;CLEAR INTERRUPT FLAG
LET INTFLG := #0
    CLR    INTFLG
; SEND A CHARACTER
LET @TBUF := #0
    CLRB   @TBUF
; DARE IT

CALL TIMER IN <#500,#1,#INTFLG,#CLR>
    MOV    R5,-(SP)
    MOV    SP,R5
    SUB    #10,SP
    MOV    R5,-(SP)
    MOV    #CLR,-(R5)
    MOV    #INTFLG,-(R5)
    MOV    #1,-(R5)
    MOV    #500,-(R5)
    JSR    PC,TIMER
    MOV    (SP)+,R5
    MOV    R5,SP
    MOV    (SP)+,R5

;DID IT CLEAR IN TIME?
IF.ERROR THEN
    BCS    +6
  
```

```

2242 005762 000137 005770
2243
2244 005766
2245 005766 104113
2246 005770
2247 005770
2248 005770
2249 005770 117700 173270
2250 005774
2251 005774
2252 006006
2253 006006 032737 000002 001214
2254 006014 001406
2255 006016 023727 001202 000001
2256 006024 002402
2257 006026 000137 006034
2258 006032
2259
2260 006032
2261 006032 000005
2262 006034
2263 006034
2264
2265
2266
2267
2268
2269
2270 006034 000004
2271 006036 012737 000015 001200
2272 006044
2273 006044 032737 000020 001220
2274 006052 001406
2275 006054 023727 001324 000001
2276 006062 001402
2277 006064 000137 006072
2278 006070
2279
2280 006070
2281 006070 000560
2282 006072
2283 006072
2284 006072
2285 006072 032737 000002 001214
2286 006100 001406
2287 006102 023727 001202 000001
2288 006110 002402
2289 006112 000137 006120
2290 006116
2291 006116
2292 006116 000005
2293 006120
2294 006120
2295
2296 006120
2297 006120 005002

                                JMP      $50105
                                :INTERR STILL OCCURED WITH IE DISABLED
                                ERRHRD 113
                                ENDIF
                                $50105:
                                :READ RBUF TO CLR IT
                                MOV      @RBUF,R0
                                ENDSUB
                                SETPRI  #PR7 ;RAISE CPU PRIORITY
                                IF #BIT01 NOTSETIN $ENV OR $PASS LT #1 THEN
                                BIT      #BIT01,$ENV
                                BEQ      $50106
                                CMP      $PASS,#1
                                BLT      .+6
                                JMP      $50107
                                $50106:
                                :CLEAR THE WORLD
                                BRESET
                                ENDIF
                                RESET
                                $50107:

                                :*****
                                :*TEST 15 TEST DATA WRAP AROUND: FLAG MODE
                                :*****
                                TST15: SCOPE
                                MOV      #15,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
                                IF #BIT04 NOTSETIN $USWR OR CONSOL EQ #BIT00 THE
                                BIT      #BIT04,$USWR
                                BEQ      $50110
                                CMP      CONSOL,#BIT00
                                BEQ      .+6
                                JMP      $50111
                                $50110:
                                :CAN'T TEST WITHOUT A WRAP
                                EXIT
                                BR      TST16 ;:EXIT THIS TEST
                                ENDIF
                                $50111:
                                IF #BIT01 NOTSETIN $ENV OR $PASS LT #1 THEN
                                BIT      #BIT01,$ENV
                                BEQ      $50112
                                CMP      $PASS,#1
                                BLT      .+6
                                JMP      $50113
                                $50112:
                                BRESET
                                $50113:
                                :BINARY COUNT PATTERN
                                INCR R2 FROM #0 TO #377 BY #1
                                CLR      R2
  
```


2298 006122 000401
 2299 006124
 2300 006124 005202
 2301 006126
 2302 006126 020227 000377
 2303 006132 003402
 2304 006134 000137 006432
 2305 006140
 2306
 2307
 2308
 2309
 2310 006140
 2311 006140 010546
 2312 006142 010605
 2313 006144 162706 000010
 2314 006150 010546
 2315 006152 012745 177777
 2316 006156 013745 001266
 2317 006162 012745 000200
 2318 006166 012745 000500
 2319 006172 004737 016272
 2320 006176 012605
 2321 006200 010506
 2322 006202 012605
 2323 006204
 2324 006204 103402
 2325 006206 000137 006216
 2326
 2327 006212
 2328 006212 104123
 2329 006214
 2330 006214 000506
 2331 006216
 2332 006216
 2333
 2334
 2335 006216
 2336 006216 110277 173046
 2337
 2338 006222
 2339 006222 010546
 2340 006224 010605
 2341 006226 162706 000010
 2342 006232 010546
 2343 006234 012745 177777
 2344 006240 013745 001262
 2345 006244 012745 000200
 2346 006250 012745 000500
 2347 006254 004737 016272
 2348 006260 012605
 2349 006262 010506
 2350 006264 012605
 2351 006266
 2352 006266 103402
 2353 006270 000137 006300

ERROR 123
 BR TST16

EXIT
 ;;EXIT THIS TEST
 ENDIF

BR \$50115
 \$50114:
 INC R2
 \$50115:
 CMP R2,#377
 BLE \$50116
 JMP \$50117
 \$50116:

; MAKE SURE IT'S READY
 CALL TIMER IN <#500,#BIT07,TCSR,#-1>
 MOV R5,-(SP)
 MOV SP,R5
 SUB #10,SP
 MOV R5,-(SP)
 MOV #-1,-(R5)
 MOV TCSR,-(R5)
 MOV #BIT07,-(R5)
 MOV #500,-(R5)
 JSR PC,TIMER
 MOV (SP)+,R5
 MOV R5,SP
 MOV (SP)+,R5
 IF.ERROR THEN
 BCS .+6
 JMP \$50120
 ; TRANSMITTER NEVER BECAME READY
 ERRHRD 123

\$50120:

; START IT ON ITS WAY
 LET @TBUF :B= R2
 MOV R2,@TBUF
 ; NOW WAIT FOR RECIEVER DONE
 CALL TIMER IN <#500,#BIT07,RCSR,#-1>
 MOV R5,-(SP)
 MOV SP,R5
 SUB #10,SP
 MOV R5,-(SP)
 MOV #-1,-(R5)
 MOV RCSR,-(R5)
 MOV #BIT07,-(R5)
 MOV #500,-(R5)
 JSR PC,TIMER
 MOV (SP)+,R5
 MOV R5,SP
 MOV (SP)+,R5
 IF.ERROR THEN
 BCS .+6
 JMP \$50121

2354	006274				ERRHRD 124
2355	006274	104124		ERROR 124	
2356					: RECIEVER NEVER BECAME READY
2357	006276				EXIT
2358	006276	000455		BR TST16	::EXIT THIS TEST
2359	006300				ENDIF
2360	006300				\$50121:
2361					:RETRIEVE
2362					LET R3 := @RBUF
2363	006300				MOV @RBUF,R3
2364	006300	017703	172760		:CHECK FOR ERROR DURING TRANSFER
2365					IF #ERROR SETIN R3 THEN
2366	006304				BIT #ERROR,R3
2367	006304	032703	104000		BNE .+6
2368	006310	001002			JMP \$50122
2369	006312	000137	006320		:ERROR BIT SET IN HIGH BYTE OF RBUF
2370					ERRHRD 200
2371	006316				ENDIF
2372	006316	104200		ERROR 200	\$50122:
2373	006320				
2374	006320				
2375					
2376					
2377					:COMPARE DATA
2378	006320				IF R3 NE R2 THEN
2379	006320	020302			CMP R3,R2
2380	006322	001002			BNE .+6
2381	006324	000137	006376		JMP \$50123
2382	006330				IF #BIT0 NOTSETIN \$USWR THEN
2383	006330	032737	000001 001220		BIT #BIT0,\$USWR
2384	006336	001402			BEQ .+6
2385	006340	000137	006366		JMP \$50124
2386	006344				IF R2 LT #200 THEN
2387	006344	020227	000200		CMP R2,#200
2388	006350	002402			BLT .+6
2389	006352	000137	006362		JMP \$50125
2390					:DATA COMPARE ERR IN 7 BIT WORD
2391	006356				ERRHRD 117
2392	006356	104117		ERROR 117	EXIT
2393	006360				::EXIT THIS TEST
2394	006360	000424		BR TST16	ENDIF
2395	006362				\$50125:
2396	006362				ELSE
2397	006362				JMP \$50126
2398	006362	000137	006372		\$50124:
2399	006366				:DATA COMP ERR IN 8 BIT WORD
2400					ERRHRD 17
2401	006366				EXIT
2402	006366	104017		ERROR 17	::EXIT THIS TEST
2403	006370				ENDIF
2404	006370	000420		BR TST16	
2405	006372				\$50126:
2406	006372				ELSE IF #BIT0 NOTSETIN \$USWR AND R2 GT #177 THEN
2407	006372				JMP \$50127
2408	006372	000137	006430		\$50123:
2409	006376				

2410 006376 032737 000001 001220
 2411 006404 001402
 2412 006406 000137 006430
 2413 006412 020227 000177
 2414 006416 003002
 2415 006420 000137 006430

BIT #BIT0,\$USWR
 BEQ .+6
 JMP \$50130
 CMP R2,#177
 BGT .+6
 JMP \$50130

;GETTING 8 BITS ON 7 BIT XMIT
 ;MAKE SURE \$USWR SETUP CORRECTLY.
 ERRHRD 22

2416
 2417
 2418 006424
 2419 006424 104022
 2420 006426
 2421 006426 000401

ERROR 22

BR TST16

EXIT
 ;;EXIT THIS TEST
 ENDIF

\$50130:
 \$50127:

ENDINC ;R2

BR \$50114
 \$50117:

2422 006430
 2423 006430
 2424 006430
 2425 006430
 2426 006430 000635
 2427 006432
 2428
 2429

 ;*TEST 16 TEST DATA WRAP AROUND: INTERRUPT MODE
 ;*****

TST16: SCOPE

MOV #16,\$TESTN

;;SET TEST NUMBER IN APT MAIL BOX

IF #BIT04 NOTSETIN \$USWR OR CONSOL EQ #BIT00 THE

2430
 2431
 2432 006432 000004
 2433 006434 012737 000016 001200
 2434 006442
 2435 006442 032737 000020 001220
 2436 006450 001406
 2437 006452 023727 001324 000001
 2438 006460 001402
 2439 006462 000137 006472
 2440 006466
 2441 006466 000137 007166
 2442 006472
 2443 006472

BIT #BIT04,\$USWR
 BEQ \$50131
 CMP CONSOL,#BIT00
 BEQ .+6
 JMP \$50132
 \$50131:

JMP TST17

;EXIT TEST
 ENDIF

\$50132:

; THIS TEST WILL RUN BOTH TRANSMITTER AND
 ; RECIEVER AT FULL SPEED TESTING
 ; THE ABILITY OF THE MODULE
 ; TO HANDLE INTERRUPTS FROM BOTH SIDES AT ONCE.
 ;
 ;DOUBLE BUFFERING IS NOT FULLY TESTED BECAUSE OF
 ;APT CONSIDERATIONS. I.E. 'BREAK' FROM APT
 ;CAUSES OVERRUN ERRORS. THEREFORE TRANSMIT INTR IS
 ;ENABLED ONLY AFTER THE RECVR HAS OBTAINED THE LAST WORD

; THIS TEST WILL TRANSFER A MAXIMUM OF 400(8)
 ; CHARACTERS THROUGH THE MODULE, BUT IF AN ERROR
 ; IS DETECTED BY THE TEST A PREMATURE SHUTDOWN OCCURS.

;CHANGE PRIORITY
 ;...TO 0

SETPRI #PRO

;GET VECTOR ADDRESS
 LET R1 := DLVEC

MOV DLVEC,R1
 ;RCVR VECTOR

2460
 2461 006472
 2462
 2463 006504
 2464 006504 013701 001260
 2465

2466	006510			
2467	006510	012721	007044	
2468	006514			
2469	006514	012721	000340	
2470				
2471				
2472	006520			
2473	006520	012721	006776	
2474	006524			
2475	006524	012711	000340	
2476				
2477				
2478	006530			
2479	006530	005037	006772	
2480	006534			
2481	006534	005037	007042	
2482	006540			
2483	006540	032737	000001	001220
2484	006546	001402		
2485	006550	000137	006566	
2486	006554			
2487	006554	012737	000200	006774
2488	006562			
2489	006562	000137	006574	
2490	006566			
2491	006566			
2492	006566	012737	000400	006774
2493	006574			
2494	006574			
2495				
2496	006574			
2497	006574	000005		
2498				
2499				
2500	006576			
2501	006576	052777	000100	172462
2502				
2503	006604			
2504	006604	052777	000100	172450
2505				
2506				
2507				
2508	006612			
2509	006612			
2510	006612			
2511	006612	023737	007042	006774
2512	006620	001403		
2513	006622	005737	006772	
2514	006626	003771		
2515	006630			
2516				
2517				
2518				
2519	006630			
2520	006630	042777	000100	172430
2521	006636			

RESET

```

LET (R1)+ := #REC
MOV #REC,(R1)+
LET (R1)+ := #PR7
MOV #PR7,(R1)+
;POINT TO TRANSMITTER VECTOR
;AND SET IT UP ALSO
LET (R1)+ := #TRAN
MOV #TRAN,(R1)+
LET (R1) := #PR7
MOV #PR7,(R1)

; CLEAR ERROR COUNTER
LET ERRCNT := #0
CLR ERRCNT
LET DATA := #0
;XMIT DATA
CLR DATA
IF #BIT0 NOTSET IN $USWR THEN
  BIT #BIT0,$USWR
  BEQ +6
  JMP $50133
  LET NUMBER := #200
  MOV #200,NUMBER
ELSE
  JMP $50134
  LET NUMBER := #400
  MOV #400,NUMBER
ENDIF
$50134:
BRESET ;SET UP ALL REGISTERS

;SET I.E. IN TRANSMITTER
LET @TCSR := @TCSR SET.BY #BIT06
BIS #BIT06,@TCSR
;AND RECEIVER
LET @RCSR := @RCSR SET.BY #BIT06
BIS #BIT06,@RCSR

;NOW WE WAIT
REPEAT
$50135:
UNTIL DATA EQ NUMBER OR ERRCNT GT #0
CMP DATA,NUMBER
BEQ $50136
TST ERRCNT
BLE $50135
$50136:

;NOW LETS CHECK.
;TURN OFF ALL INTR ENABLE
LET @TCSR := @TCSR CLR.BY #BIT06
BIC #BIT06,@TCSR
LET @RCSR := @RCSR CLR.BY #BIT06

```


2522	006636	042777	000100	172416	
2523	006644				
2524	006644	005737	006772		
2525	006650	001002			
2526	006652	000137	006770		
2527	006656				
2528	006656	032737	104000	007164	
2529	006664	001002			
2530	006666	000137	006766		
2531	006672				
2532	006672	032737	040000	007164	
2533	006700	001002			
2534	006702	000137	006714		
2535					
2536	006706				
2537	006706	104220			ERROR 220
2538	006710				
2539	006710	000137	006762		
2540	006714				
2541	006714	032737	020000	007164	
2542	006722	001002			
2543	006724	000137	006736		
2544					
2545	006730				
2546	006730	104221			ERROR 221
2547	006732				
2548	006732	000137	006762		
2549	006736				
2550	006736	032737	010000	007164	
2551	006744	001002			
2552	006746	000137	006760		
2553					
2554	006752				
2555	006752	104222			ERROR 222
2556	006754				
2557	006754	000137	006762		
2558	006760				
2559					
2560	006760				
2561	006760	104024			ERROR 24
2562	006762				
2563	006762				
2564	006762				
2565	006762				
2566	006762				
2567	006762	000137	006770		
2568	006766				
2569					
2570	006766				
2571	006766	104120			ERROR 120
2572	006770				
2573	006770				
2574	006770				
2575	006770				
2576	006770				
2577	006770	000476			BR TST17

```

      BIC      #BIT06, @RCSR
IF ERRCNT NE #0 THEN
      TST      ERRCNT
      BNE      .+6
      JMP      $50137
      IF #ERROR SETIN RHL D THEN
          BIT      #ERROR, RHL D
          BNE      .+6
          JMP      $50140
      IF #BIT14 SETIN RHL D THEN
          BIT      #BIT14, RHL D
          BNE      .+6
          JMP      $50141
      :OVERRUN ERROR
      ERRHRD 220
      ELSE IF #BIT13 SETIN RHL D THEN
          JMP      $50142
          $50141:
          BIT      #BIT13, RHL D
          BNE      .+6
          JMP      $50143
      :FRAMING ERROR
      ERRHRD 221
      ELSE IF #BIT12 SETIN RHL D THEN
          JMP      $50144
          $50143:
          BIT      #BIT12, RHL D
          BNE      .+6
          JMP      $50145
      :PARITY ERROR
      ERRHRD 222
      ELSE
          JMP      $50146
          $50145:
      :UNKNOWN ERROR
      ERRHRD 24
      ENDIF
      $50146:
      $50144:
      $50142:
      ELSE
          JMP      $50147
          $50140:
      :DATA COMPARE ERROR
      ERRHRD 120
      ENDIF
      $50147:
      ENDIF
      $50137:
      EXIT      :SKIP OVER SUPPORT ROUTINES & STORAGE
      ;;EXIT THIS TEST
  
```

KDF11-B / DLV11-J TEST MACY11 30(1046) 05-OCT-81 16:37 PAGE 54 B 5
CJDLAA.P11 29-SEP-81 11:42 T16 TEST DATA WRAP AROUND: INTERRUPT MODE

SEQ 0053

2578
2579
2580 006772 000000
2581 006774 000000

ERRCNT: 0
NUMBER: 0


```

2582
2583
2584
2585
2586 006776
2587 006776
2588 006776
2589 006776 023737 007042 006774
2590 007004 001002
2591 007006 000137 007032
2592 007012 005737 006772
2593 007016 001402
2594 007020 000137 007032
2595
2596 007024
2597 007024 013777 007042 172236
2598 007032
2599 007032
2600
2601 007032
2602 007032 042777 000100 172226
2603 007040
2604 007040 000002
2605
2606 007042 000000
2607
2608
2609
2610
2611 007044
2612 007044
2613
2614 007044
2615 007044 017737 172214 007164
2616
2617 007052
2618 007052 032737 104000 007164
2619 007060 001006
2620 007062 023737 007164 007042
2621 007070 001002
2622 007072 000137 007122
2623 007076
2624
2625 007076
2626 007076 013737 006774 007042
2627 007104
2628 007104 042777 000100 172150
2629 007112
2630 007112 005237 006772
2631 007116
2632 007116 000137 007162
2633 007122
2634 007122
2635 007122 005237 007042
2636 007126
2637 007126 023737 007042 006774

```

```

;*****
;TRANSMIT INTERRUPT HANDLER
;*****
BGNSRV TRAN
TRAN:
IF DATA NE NUMBER AND ERRCNT EQ #0 THEN
  CMP DATA,NUMBER
  BNE .+6
  JMP $50150
  TST ERRCNT
  BEQ .+6
  JMP $50150
;SHIP OUT WORD
LET @TBUF := DATA
MOV DATA,@TBUF
ENDIF
$50150:
;STOP INTERR, NOT EXER DOUBL BUFFER
LET @TCSR := @TCSR CLR.BY #BIT06
BIC #BIT06,@TCSR
ENDSRV
RTI
DATA: 0
;*****
;RECEIVER INTERRUPT HANDLER
;*****
BGNSRV REC
REC:
;GET CHAR
LET RHL := @RBUF
MOV @RBUF,RHL
;CHECK ERROR
IF #ERROR SETIN RHL OR RHL NE DATA THEN
  BIT #ERROR,RHL
  BNE $50151
  CMP RHL,DATA
  BNE .+6
  JMP $50152
$50151:
;STOP ALL INTERR PROC & GET OUT
LET DATA := NUMBER
MOV NUMBER,DATA
LET @RCSR := @RCSR CLR.BY #BIT06
BIC #BIT06,@RCSR
LET ERRCNT := ERRCNT + #1
INC ERRCNT
ELSE
  JMP $50153
$50152:
LET DATA := DATA + #1
INC DATA
IF DATA EQ NUMBER THEN
  CMP DATA,NUMBER

```

RHLD: 0

```

:*****
:*TEST 17      TEST BREAK DETECTION LOGIC
:*             TRANSMIT KNOWN CHAR WITH BREAK SET
:*             AND COMPARE RECEIVED WITH 0.
:*             FRAMING ERROR WILL ALSO BE CHECKED
:*             IF ERROR BITS ARE ENABLED.
:*****

```

```

TST17:  SCOPE
        MOV     #17,$TESTN

```

```

;;SET TEST NUMBER IN APT MAIL BOX
;;DONT DO THIS TEST IF 'BREAK' GENERATION
;;ENABLED, ELSE WILL HALT TO CONSOL ODT.
;;DO IF BREAK 'DETECTION' IS ENABLED.
IF #BIT04 NOTSETIN $USWR OR #BIT03 NOTSETIN $USWR THEN

```

```

        BIT      #BIT04,$USWR
        BEQ      $50156
        BIT      #BIT03,$USWR
        BEQ      .+6
        JMP      $50157
$50156:
        :EXIT TEST

```

ELSE

```

                JMP          $50160
                $50157:
IF CONSOL EQ #BIT00 THEN
                CMP          CONSOL,#BIT00
                BEQ          +6
                JMP          $50161
                ;CAN'T TEST CONSOL
                ;EXIT TEST
ENDIF

```

ENDIF

BGNSUB

\$50161:
\$50160:
MOV #64\$,\$LPERR

2694	007260			
2695	007260	005037	010100	
2696				
2697	007264			
2698	007264	052777	000001	171774
2699				
2700	007272			
2701	007272	012777	000125	171770
2702				
2703	007300			
2704	007300	010546		
2705	007302	010605		
2706	007304	162706	000010	
2707	007310	010546		
2708	007312	012745	177777	
2709	007316	013745	001262	
2710	007322	012745	000200	
2711	007326	012745	000500	
2712	007332	004737	016272	
2713	007336	012605		
2714	007340	010506		
2715	007342	012605		
2716	007344			
2717	007344	103402		
2718	007346	000137	007354	
2719				
2720	007352			
2721	007352	104115		
2722	007354			
2723	007354			
2724	007354			
2725	007354	017700	171704	
2726	007360			
2727	007360	105700		
2728	007362	001002		
2729	007364	000137	007376	
2730				
2731	007370			
2732	007370	052737	000001	010100
2733	007376			
2734	007376			
2735	007376			
2736	007376	032700	020000	
2737	007402	001402		
2738	007404	000137	007416	
2739	007410			
2740	007410	052737	000002	010100
2741	007416			
2742	007416			
2743	007416			
2744	007416	032737	004000	001220
2745	007424	001002		
2746	007426	000137	007512	
2747				
2748	007432			
2749	007432	032737	000004	001220

ERROR 115

```

LET ERRCHK := #0          ; CLEAR ERROR WORD
                           CLR      ERRCHK
                           ; SET BREAK BIT
LET @TCSR := @TCSR SET.BY #BIT00
                           BIS      #BIT00,@TCSR
                           ; NON-ZERO CHAR. '*'
LET @TBUF := #125
                           MOV      #125,@TBUF
                           ; WAIT FOR DONE
CALL TIMER IN <#500,#BIT07,RCSR,#-1>
                           MOV      R5,-(SP)
                           MOV      SP,R5
                           SUB      #10,SP
                           MOV      R5,-(SP)
                           MOV      #-1,-(R5)
                           MOV      RCSR,-(R5)
                           MOV      #BIT07,-(R5)
                           MOV      #500,-(R5)
                           JSR      PC,TIMER
                           MOV      (SP)+,R5
                           MOV      R5,SP
                           MOV      (SP)+,R5

IF.ERROR THEN
                           BCS      +6
                           JMP      $50162
                           ; RECIEVER DONE DID NOT SET
                           ERRHRD 115

ENDIF

$50162:
LET R0 := @RBUF
                           MOV      @RBUF,R0
IFB R0 NE #0 THEN
                           TSTB     R0
                           BNE      +6
                           JMP      $50163
                           ; BREAK DID NOT EQUAL 0
LET ERRCHK := ERRCHK SET.BY #BIT0
                           BIS      #BIT0,ERRCHK

ENDIF

$50163:
IF #BIT13 NOTSETIN R0 THEN
                           BIT      #BIT13,R0
                           BEQ      +6
                           JMP      $50164
                           LET ERRCHK := ERRCHK SET.BY #BIT1
                           BIS      #BIT1,ERRCHK

ENDIF

$50164:
IF #BIT11 SETIN $USWR THEN
                           BIT      #BIT11,$USWR
                           BNE      +6
                           JMP      $50165
                           ; ODD PARITY ENABLED
IF #BIT02 NOTSETIN $USWR THEN
                           BIT      #BIT02,$USWR

```

2750 007440 001402
 2751 007442 000137 007472
 2752
 2753 007446
 2754 007446 032700 010000
 2755 007452 001402
 2756 007454 000137 007466
 2757
 2758 007460
 2759 007460 052737 000004 010100
 2760 007466
 2761 007466
 2762 007466
 2763 007466 000137 007512
 2764 007472
 2765 007472
 2766 007472 032700 010000
 2767 007476 001002
 2768 007500 000137 007512
 2769 007504
 2770 007504 052737 000010 006772
 2771 007512
 2772 007512
 2773 007512
 2774 007512
 2775 007512
 2776 007512
 2777
 2778 007512
 2779 007512 032737 000002 001214
 2780 007520 001406
 2781 007522 023727 001202 000001
 2782 007530 002402
 2783 007532 000137 007552
 2784 007536
 2785 007536
 2786 007536 000005
 2787 007540 032777 170000 171516
 2788 007546 001401
 2789 007550 104033
 2790 007552
 2791 007552
 2792 007552
 2793
 2794 007552
 2795 007552 032737 000001 010100
 2796 007560 001002
 2797 007562 000137 007570
 2798 007566
 2799 007566 104121
 2800 007570
 2801 007570
 2802 007570
 2803 007570 032737 000002 010100
 2804 007576 001002
 2805 007600 000137 007606

IF #BIT01 NOTSETIN \$ENV OR \$PASS LT #1 THEN

RESET
 BIT #170000,@RBUF
 BEQ ESR1
 ERROR 33

ENDIF ;RESET DID NOT CLEAR ERROR,FR ERR,OR PERR IN RBUF

ESR1:

ERROR 121

BEQ .+6
 JMP \$50166
 ;BREAK SHOULD GENERATE A PARITY ERRO
 IF #BIT12 NOTSETIN R0 THEN
 BIT #BIT12,R0
 BEQ .+6
 JMP \$50167
 ;NO PAR ERROR WHEN THERE SHOULD
 LET ERRCHK := ERRCHK SET.BY #BIT2
 BIS #BIT2,ERRCHK
 ENDIF
 \$50167:
 ELSE
 JMP \$50170
 \$50166:
 IF #BIT12 SETIN R0 THEN
 BIT #BIT12,RC
 BNE .+6
 JMP \$50171
 LET ERRCNT := ERRCNT SET.BY #BIT3
 BIS #BIT3,ERRCNT
 ENDIF
 \$50171:
 ENDIF
 \$50170:
 ENDIF
 \$50165:
 BIT #BIT01,\$ENV
 BEQ \$50172
 CMP \$PASS,#1
 BLT .+6
 JMP \$50173
 \$50172:
 BRESET ;CLEAN UP
 \$50173:
 IF #BIT0 SETIN ERRCHK THEN
 BIT #BIT0,ERRCHK
 BNE .+6
 JMP \$50174
 ERRHRD 121 ;BREAK ERROR
 ENDIF
 \$50174:
 IF #BIT1 SETIN ERRCHK THEN
 BIT #BIT1,ERRCHK
 BNE .+6
 JMP \$50175

2806	007604				
2807	007604	104122		ERROR	122
2808	007606				
2809	007606				
2810	007606				
2811	007606	032737	000004	010100	
2812	007614	001002			
2813	007616	000137	007624		
2814	007622				
2815	007622	104235		ERROR	235
2816					
2817					
2818	007624				
2819	007624				
2820	007624				
2821	007624	032737	000010	010100	
2822	007632	001002			
2823	007634	000137	007642		
2824	007640				
2825	007640	104236		ERROR	236
2826					
2827					
2828					
2829	007642				
2830	007642				
2831	007642				
2832	007642				
2833	007642	012737	007650	001110	
2834					
2835	007650				
2836	007650	052777	000001	171410	
2837					
2838	007656				
2839	007656	010546			
2840	007660	010605			
2841	007662	162706	000010		
2842	007666	010546			
2843	007670	012745	177777		
2844	007674	013745	001266		
2845	007700	012745	000001		
2846	007704	012745	000500		
2847	007710	004737	016272		
2848	007714	012605			
2849	007716	010506			
2850	007720	012605			
2851					
2852					
2853	007722				
2854	007722	103402			
2855	007724	000137	007732		
2856					
2857	007730				
2858	007730	104021		ERROR	21
2859	007732				
2860	007732				
2861					

```

ERRHRD 122 ; FRAMING ERROR
ENDIF
$50175:
IF #BIT2 SETIN ERRCHK THEN
  BIT #BIT2,ERRCHK
  BNE +6
  JMP $50176
ERRHRD 235
;NO PARITY ERROR WHEN
;THERE SHOULD BE
ENDIF
$50176:
IF #BIT3 SETIN ERRCHK THEN
  BIT #BIT3,ERRCHK
  BNE +6
  JMP $50177
ERRHRD 236
;PARITY ERROR SHOULD NOT HAVE
;OCCURED WITH EVEN PARITY
;ENABLED AND BREAK SET
ENDIF
$50177:
ENDSUB
BGNSUB
MOV #64$, $LPERR
;SET BREAK BIT
LET @TCSR := @TCSR SET.BY #BIT00
BIS #BIT00,@TCSR
CALL TIMER IN <#500,#BIT00,TCSR,#-1>
MOV R5,-(SP)
MOV SP,R5
SUB #10,SP
MOV R5,-(SP)
MOV #-1,-(R5)
MOV TCSR,-(R5)
MOV #BIT00,-(R5)
MOV #500,-(R5)
JSR PC,TIMER
MOV (SP)+,R5
MOV R5,SP
MOV (SP)+,R5
;DID IT SET?
IF.ERROR THEN
  BCS +6
  JMP $50200
;BREAK DID NOT SET
ERRHRD 21
ENDIF
$50200:

```

```

2862
2863 007732
2864 007732 042777 000001 171326
2865 007740
2866 007740 010546
2867 007742 010605
2868 007744 162706 000002
2869 007750 010546
2870 007752 012745 000144
2871 007756 004737 016524
2872 007762 012605
2873 007764 010506
2874 007766 012605
2875
2876
2877
2878 007770
2879 007770 017700 171270
2880
2881 007774
2882 007774 012777 000125 171266
2883
2884 010002
2885 010002 010546
2886 010004 010605
2887 010006 162706 000010
2888 010012 010546
2889 010014 012745 177777
2890 010020 013745 001262
2891 010024 012745 000200
2892 010030 012745 000500
2893 010034 004737 016272
2894 010040 012605
2895 010042 010506
2896 010044 012605
2897 010046
2898 010046 103402
2899 010050 000137 010056
2900
2901 010054
2902 010054 104230 ERROR 230
2903 010056
2904 010056
2905
2906
2907 010056
2908 010056 127727 171202 000125
2909 010064 001002
2910 010066 000137 010074
2911
2912 010072
2913 010072 104231 ERROR 231
2914 010074
2915 010074
2916 010074
2917 010074
  
```

```

;CLEAR BREAK BIT
LET @TCSR := @TCSR CLR.BY #BIT00
      BIC #BIT00,@TCSR
      WAITMS 100.
      MOV R5,-(SP)
      MOV SP,R5
      SUB #2,SP
      MOV R5,-(SP)
      MOV #100,-(R5)
      JSR PC,WAIT
      MOV (SP)+,R5
      MOV R5,SP
      MOV (SP)+,R5

;READ RBUF TO CLEAR ERRORS & REC DONE
LET R0 := @RBUF
      MOV @RBUF,R0
;SEND CHAR
LET @TBUF := #125
      MOV #125,@TBUF
;WAIT FOR DONE BIT
CALL TIMER IN <#500,#BIT07,RCSR,#-1>
      MOV R5,-(SP)
      MOV SP,R5
      SUB #10,SP
      MOV R5,-(SP)
      MOV #-1,-(R5)
      MOV RCSR,-(R5)
      MOV #BIT07,-(R5)
      MOV #500,-(R5)
      JSR PC,TIMER
      MOV (SP)+,R5
      MOV R5,SP
      MOV (SP)+,R5

IF.ERROR THEN
      BCS +6
      JMP $50201
;RECEIVER NEVER CAME READY
ERRHRD 230

ENDIF

$50201:
;WAS CHAR AFTER BREAK RECEIVED
IFB @RBUF NE #125 THEN
      CMPB @RBUF,#125
      BNE +6
      JMP $50202
;CHAR AFTER BREAK NOT RECEIVED CORRECTLY
ERRHRD 231

ENDIF

$50202:
ENDSUB
INLINE <JMP LOOP>
  
```



```

2918 010074 000137 002000                                JMP    LOOP
2919 010100 000000                                ERRCHK: .WORD 0
2920
2921 010102                                TEST20:
2922 *****
2923 :*TEST 20          NOT A TEST -
2924 *****
2925 010102 000004                                TST20:  SCOPE
2926                                           ;DLVEC IS 340, DLADD IS 176540
2927                                           ;OR
2928                                           ;DLVEC IS 400, DLADD IS 176600
2929
2930
2931 010104                                LET $TESTN := #20
2932 010104 012737 000020 001200                                MOV    #20,$TESTN
2933
2934 010112                                IF SECSLU EQ #176500 THEN
2935 010112 023727 001302 176500                                CMP    SECSLU,#176500
2936 010120 001402                                BEQ    +6
2937 010122 000137 010146                                JMP    $50203
2938 010126                                LET DLVEC := DLVEC - #40
2939 010126 162737 000040 001260                                SUB    #40,DLVEC
2940 010134                                LET DLADD := DLADD - #40
2941 010134 162737 000040 001256                                SUB    #40,DLADD
2942 010142                                ELSE                                ;EITHER ZERO OR 176540
2943 010142 000137 010162                                JMP    $50204
2944 010146                                $50203:
2945 010146                                LET DLVEC := $VECT1                                ;300
2946 010146 013737 001244 001260                                MOV    $VECT1,DLVEC
2947 010154                                LET DLADD := $BASE                                ;176500
2948 010154 013737 001250 001256                                MOV    $BASE,DLADD
2949 010162                                ENDIF
2950 010162                                $50204:
2951
2952 010162                                MODTST:
2953 *****
2954 :*TEST 21          TEST THAT CHANNELS INTR AT ASSIGNED PRIORITY
2955 :*                INTERRUPTS WILL BE ENABLED ON ALL ACTIVE CHANNELS.
2956 :*                RECEIVER AND TRANSMITTER. THEN WE'LL CHECK TO
2957 :*                SEE IF THEY INTERRUPTED IN THE ASSIGNED SEQUENCE.
2958 :*                THIS TEST WILL BE BYPASSED IF THERE ARE NO DLV11-J'S.
2959 *****
2960 010162 000004                                TST21:  SCOPE
2961 010164 012737 000021 001200                                MOV    #21,$TESTN                                ;;SET TEST NUMBER IN APT MAIL BOX
2962
2963
2964                                ;CLEAR OUT INTERRUPT TABLE
2965 010172                                LET R0 := #INTRTABLE
2966 010172 012700 011452                                MOV    #INTRTABLE,R0
2967 010176                                REPEAT
2968 010176                                $50205:
2969 010176                                LET (R0)+ := #0
2970 010176 005020                                CLR    (R0)+
2971 010200                                UNTIL R0 EQ #TABEND
2972 010200 020027 011512                                CMP    R0,#TABEND
2973 010204 001374                                BNE    $50205
  
```

```

2974
2975
2976 010206
2977
2978
2979 010220
2980 010220 013700 001260
2981 010224
2982 010224 012701 010612
2983 010230
2984 010230
2985 010230
2986 010230 010120
2987 010232
2988 010232 012720 000340
2989 010236
2990 010236 062701 000030
2991 010242
2992 010242 020127 011412
2993 010246 001370
2994
2995
2996
2997
2998
2999 010250
3000 010250 013737 001252 011512
3001 010256
3002 010256 005037 011514
3003 010262
3004 010262 013700 001250
3005 010266
3006 010266
3007 010266
3008 010266 032737 000001 011512
3009 010274 001002
3010 010276 000137 010406
3011
3012 010302
3013 010302 052710 000100
3014 010306
3015 010306 062700 000004
3016
3017 010312
3018 010312 052710 000100
3019
3020 010316
3021 010316 012760 000252 000002
3022
3023 010324
3024 010324 162700 000004
3025
3026 010330
3027 010330 010546
3028 010332 010605
3029 010334 162706 000010
  
```

SETPRI #PR7

;SET PRIORITY TO 7

```

;SET UP ALL INTERRUPT VECTORS
LET R0 := DLVEC
LET R1 := #RCVROSRV
REPEAT
  LET (R0)+ := R1
  LET (R0)+ := #PR7
  LET R1 := R1 + #30
UNTIL R1 EQ #SRVEND
MOV DLVEC,R0
MOV #RCVROSRV,R1
$50206:
MOV R1,(R0)+
MOV #PR7,(R0)+
ADD #30,R1
CMP R1,#SRVEND
BNE $50206
  
```

;ENABLE INTERRUPTS ON ALL ACTIVE LINES,
 ;ALSO, XMIT CHAR'S TO PRIME RECEIVERS

```

;COPY DEVICES
LET CHMASK := $DEV
LET CHCNT := #0
LET R0 := $BASE
REPEAT
  IF #BIT0 SETIN CHMASK THEN
    BIT #BIT0,CHMASK
    BNE +6
    JMP $50210
;SET RCSR IE
LET (R0) := (R0) SET.BY #BIT6
BIS #BIT6,(R0)
LET R0 := R0 + #4
ADD #4,R0
;SET XCSR IE
LET (R0) := (R0) SET.BY #BIT6
BIS #BIT6,(R0)
;LOAD XBUF
LET 2(R0) := #252
MOV #252,2(R0)
;GO BACK TO RCSR
LET R0 := R0 - #4
SUB #4,R0
;LOOK FOR RCVR DONE
CALL TIMER IN <#500,#BIT07,R0,#-1>
MOV R5,-(SP)
MOV SP,R5
SUB #10,SP
  
```


3030 010340 010546
 3031 010342 012745 177777
 3032 010346 010045
 3033 010350 012745 000200
 3034 010354 012745 000500
 3035 010360 004737 016272
 3036 010364 012605
 3037 010366 010506
 3038 010370 012605
 3039
 3040 010372
 3041 010372 103402
 3042 010374 000137 010402
 3043
 3044 010400
 3045 010400 104023
 3046 010402
 3047 010402
 3048
 3049 010402
 3050 010402 005237 011514
 3051 010406
 3052 010406
 3053 010406
 3054 010406 062700 000010
 3055
 3056 010412
 3057 010412 006237 011512
 3058 010416
 3059 010416 032737 000001 001306
 3060 010424 001002
 3061 010426 000137 010440
 3062 010432
 3063 010432 012737 000004 010444
 3064 010440
 3065 010440
 3066 010440
 3067 010440 023727 011514 000010
 3068 010446 001307
 3069
 3070
 3071
 3072
 3073
 3074
 3075
 3076
 3077
 3078
 3079
 3080 010450
 3081 010450 012704 011452
 3082 010454
 3083 010454 013737 001256 001262
 3084
 3085

ERROR 23

IF #BIT00 SET IN KDDL V THEN

LET .+12 := #4

ENDIF

LET RCSR := DLADD

MOV R5, -(SP)
 MOV #-1, -(R5)
 MOV R0, -(R5)
 MOV #BIT07, -(R5)
 MOV #500, -(R5)
 JSR PC, TIMER
 MOV (SP)+, R5
 MOV R5, SP
 MOV (SP)+, R5
 ;DID IT SET IN TIME?
 IF .ERROR THEN
 BCS .+6
 JMP \$50211
 ;RCVR DONE DID NOT SET IN RCSR
 ERRHRD 23

ENDIF

\$50211:

LET CHCNT := CHCNT + #1
 INC CHCNT
 ENDIF

\$50210:

LET R0 := R0 + #10
 ADD #10, R0
 ;SETUP FOR NEXT CHANNEL
 LET CHMASK := CHMASK SHIFT -1
 ASR CHMASK

CHMASK

BIT #BIT00, KDDL V

BNE .+6

JMP \$50212

;TEST 4 MODULES

MOV #4, .+12

\$50212:

UNTIL CHCNT EQ #10

CMP CHCNT, #10

BNE \$50207

;ALL XMIT & REC INTERRUPTS SHOULD BE
 ;IN CONTENTION.
 ;RECEIVE INTERRUPTS HAVE PRIORITY
 ;OVER XMIT INTERRUPTS.
 ;LOW CHANNELS HAVE PRIORITY OVER HIGH CHANNELS.
 ;THEREFORE, ONCE CPU PRIORITY IS LOWERED,
 ;INTERRUPTS S/B IN THE ORDER SHOWN IN THE
 ;CHANNEL IDENTIFIER TABLE (RCHO:)

;SET UP POINTER FOR SERVICE ROUTINE
 LET R4 := #INTRTABLE

MOV #INTRTABLE, R4

;GET BACK TO ORG ADRS

MOV DLADD, RCSR

;LET EM GO

```
3086 010462          SETPRI #PRO
3087
3088
3089 010474          BRESET          ;DISABLE ALL INTERRUPTS
3090 010474 000005          RESET
3091
3092 010476          LET TCNT := #0
3093 010476 005037 011520          CLR          TCNT
3094
3095 010502          ;CLEAR ERR FLAG
3096 010502 005037 011516          LET PRIERR := #0          CLR          PRIERR
3097 010506          LET R0 := #INTRTABLE
3098 010506 012700 011452          MOV          #INTRTABLE,R0
3099
3100 010512          ;SETUP EXPECTED VALUE
3101 010512 012701 011454          LET R1 := #INTRTABLE+2
3102 010516 013737 001252 011512          MOV          $DEVM,CHMASK          ;GET # OF CHANNELS IN TCNT
3103
3104 010524 006237 011512          1$: ASR          CHMASK
3105 010530 103002          BCC          2$
3106 010532 005237 011520          INC          TCNT          ;GET TOTAL CHANNEL CT
3107 010536 005737 011512          2$: TST          CHMASK
3108 010542 001370          BNE          1$
3109
3110 010544 005337 011520          4$: DEC          TCNT
3111 010550 001410          BEQ          5$
3112 010552 023727 011520 177777          CMP          TCNT,#-1          ;ALL 1'S ?
3113 010560 001404          BEQ          5$          ;BR IF YES
3114 010562 022120          CMP          (R1)+,(R0)+          ;R1-R0
3115 010564 003367          BGT          4$          ;BR IF R1>R0
3116 010566 005237 011516          INC          PRIERR          ;ELSE ERROR
3117
3118 010572          5$:
3119 010572
3120 010572 005737 011516          IF PRIERR NE #0 THEN
3121 010576 001002          TST          PRIERR
3122 010600 000137 010606          BNE          +6
3123 010604          JMP          $50213
3124 010604 104250          ERRHRD 250
3125
3126          ;CHANNELS DID NOT INTR ACCORDING TO
3127          ;ASSIGNED PRIORITY. THE BYTE ENTRIES
3128          ;IN INTRTABLE: SHOULD BE IN THE ORDER
3129          ;THAT THEY APPEAR IN THE CHANNEL #
3130          ;TABLE .(EXCLUDING CHANNELS NOT ACTIVE)
3131 010606          ENDIF
3132 010606          $50213:
3133 010606 000137 011524          JMP          TST22
3134
3135          *****
3136          ;*          START OF SERVICE ROUTINES
3137          ;*          *****
3138
3139 010612          BGNSRV RCVROSRV
3140 010612          RCVROSRV:
3141          ;PUT INTR IDENTIFIER IN INTRTABLE
```



```

3142 010612
3143 010612 013724 011412
3144
3145 010616
3146 010616 013737 001262 011522
3147 010624
3148 010624 062737 000000 011522
3149
3150 010632
3151 010632 042777 000100 000662
3152 010640
3153 010640 000002
3154
3155
3156 010642
3157 010642
3158
3159 010642
3160 010642 013724 011432
3161
3162 010646
3163 010646 013737 001262 011522
3164 010654
3165 010654 062737 000004 011522
3166
3167 010662
3168 010662 042777 000100 000632
3169 010670
3170 010670 000002
3171
3172 010672
3173 010672
3174
3175 010672
3176 010672 013724 011414
3177
3178 010676
3179 010676 013737 001262 011522
3180 010704
3181 010704 062737 000010 011522
3182
3183 010712
3184 010712 042777 000100 000602
3185 010720
3186 010720 000002
3187
3188 010722
3189 010722
3190
3191 010722
3192 010722 013724 011434
3193
3194 010726
3195 010726 013737 001262 011522
3196 010734
3197 010734 062737 000014 011522
  
```

RTI

XMIT0SRV:

RTI

RCVR1SRV:

RTI

XMIT1SRV:

```

      LET (R4)+ := RCH0
      MOV      RCH0,(R4)+
      ;GENERATE CSR ADDRESS
      LET TEMP := RCSR
      MOV      RCSR,TEMP
      LET TEMP := TEMP + #0
      ADD      #0,TEMP
      ;ONE INTR IS ALL WE WANT FROM HERE
      LET @TEMP := @TEMP CLR.BY #BIT6
      BIC      #BIT6,@TEMP
ENDSRV

BGNSRV XMIT0SRV
      ;PUT INTR IDENTIFIER IN INTRTABLE
      LET (R4)+ := TCH0
      MOV      TCH0,(R4)+
      ;GENERATE CSR ADDRESS
      LET TEMP := RCSR
      MOV      RCSR,TEMP
      LET TEMP := TEMP + #4
      ADD      #4,TEMP
      ;ONE INTR IS ALL WE WANT FROM HERE
      LET @TEMP := @TEMP CLR.BY #BIT6
      BIC      #BIT6,@TEMP
ENDSRV

BGNSRV RCVR1SRV
      ;PUT INTR IDENTIFIER IN INTRTABLE
      LET (R4)+ := RCH1
      MOV      RCH1,(R4)+
      ;GENERATE CSR ADDRESS
      LET TEMP := RCSR
      MOV      RCSR,TEMP
      LET TEMP := TEMP + #10
      ADD      #10,TEMP
      ;ONE INTR IS ALL WE WANT FROM HERE
      LET @TEMP := @TEMP CLR.BY #BIT6
      BIC      #BIT6,@TEMP
ENDSRV

BGNSRV XMIT1SRV
      ;PUT INTR IDENTIFIER IN INTRTABLE
      LET (R4)+ := TCH1
      MOV      TCH1,(R4)+
      ;GENERATE CSR ADDRESS
      LET TEMP := RCSR
      MOV      RCSR,TEMP
      LET TEMP := TEMP + #14
      ADD      #14,TEMP
  
```

3198
 3199 010742
 3200 010742 042777 000100 000552
 3201 010750
 3202 010750 000002 RTI
 3203
 3204 010752
 3205 010752 RCVR2SRV:
 3206
 3207 010752
 3208 010752 013724 011416
 3209
 3210 010756
 3211 010756 013737 001262 011522
 3212 010764
 3213 010764 062737 000020 011522
 3214
 3215 010772
 3216 010772 042777 000100 000522
 3217 011000
 3218 011000 000002 RTI
 3219
 3220 011002
 3221 011002 XMIT2SRV:
 3222
 3223 011002
 3224 011002 013724 011436
 3225
 3226 011006
 3227 011006 013737 001262 011522
 3228 011014
 3229 011014 062737 000024 011522
 3230
 3231 011022
 3232 011022 042777 000100 000472
 3233 011030
 3234 011030 000002 RTI
 3235
 3236 011032
 3237 011032 RCVR3SRV:
 3238
 3239 011032
 3240 011032 013724 011420
 3241
 3242 011036
 3243 011036 013737 001262 011522
 3244 011044
 3245 011044 062737 000030 011522
 3246
 3247 011052
 3248 011052 042777 000100 000442
 3249 011060
 3250 011060 000002 RTI
 3251
 3252 011062
 3253 011062 XMIT3SRV:

```

;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
BIC #BIT6,@TEMP
ENDSRV

BGNSRV RCVR2SRV

;PUT INTR IDENTIFIER IN INTRTABLE
LET (R4)+ := RCH2
MOV RCH2,(R4)+
;GENERATE CSR ADDRESS
LET TEMP := RCSR
MOV RCSR,TEMP
LET TEMP := TEMP + #20
ADD #20,TEMP
;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
BIC #BIT6,@TEMP
ENDSRV

BGNSRV XMIT2SRV

;PUT INTR IDENTIFIER IN INTRTABLE
LET (R4)+ := TCH2
MOV TCH2,(R4)+
;GENERATE CSR ADDRESS
LET TEMP := RCSR
MOV RCSR,TEMP
LET TEMP := TEMP + #24
ADD #24,TEMP
;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
BIC #BIT6,@TEMP
ENDSRV

BGNSRV RCVR3SRV

;PUT INTR IDENTIFIER IN INTRTABLE
LET (R4)+ := RCH3
MOV RCH3,(R4)+
;GENERATE CSR ADDRESS
LET TEMP := RCSR
MOV RCSR,TEMP
LET TEMP := TEMP + #30
ADD #30,TEMP
;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
BIC #BIT6,@TEMP
ENDSRV

BGNSRV XMIT3SRV

```



```

3254
3255 011062
3256 011062 013724 011440
3257
3258 011066
3259 011066 013737 001262 011522
3260 011074
3261 011074 062737 000034 011522
3262
3263 011102
3264 011102 042777 000100 000412
3265 011110
3266 011110 000002
3267 011112
3268 011112
3269
3270 011112
3271 011112 013724 011422
3272
3273 011116
3274 011116 013737 001262 011522
3275 011124
3276 011124 062737 000040 011522
3277
3278 011132
3279 011132 042777 000100 000362
3280 011140
3281 011140 000002
3282 011142
3283 011142
3284
3285 011142
3286 011142 013724 011442
3287
3288 011146
3289 011146 013737 001262 011522
3290 011154
3291 011154 062737 000044 011522
3292
3293 011162
3294 011162 042777 000100 000332
3295 011170
3296 011170 000002
3297 011172
3298 011172
3299
3300 011172
3301 011172 013724 011424
3302
3303 011176
3304 011176 013737 001262 011522
3305 011204
3306 011204 062737 000050 011522
3307
3308 011212
3309 011212 042777 000100 000302
  
```

RTI

RCVR4SRV:

RTI

XMIT4SRV:

RTI

RCVR5SRV:

ENDSRV

BGNSRV RCVR4SRV

ENDSRV

BGNSRV XMIT4SRV

ENDSRV

BGNSRV RCVR5SRV

```

;PUT INTR IDENTIFIER IN INTRTABLE
LET (R4)+ := TCH3
      MOV      TCH3,(R4)+
;GENERATE CSR ADDRESS
LET TEMP := RCSR
      MOV      RCSR,TEMP
LET TEMP := TEMP + #34
      ADD      #34,TEMP
;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
      BIC      #BIT6,@TEMP

ENDSRV
BGNSRV RCVR4SRV

;PUT INTR IDENTIFIER IN INTRTABLE
LET (R4)+ := RCH4
      MOV      RCH4,(R4)+
;GENERATE CSR ADDRESS
LET TEMP := RCSR
      MOV      RCSR,TEMP
LET TEMP := TEMP + #40
      ADD      #40,TEMP
;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
      BIC      #BIT6,@TEMP

ENDSRV
BGNSRV XMIT4SRV

;PUT INTR IDENTIFIER IN INTRTABLE
LET (R4)+ := TCH4
      MOV      TCH4,(R4)+
;GENERATE CSR ADDRESS
LET TEMP := RCSR
      MOV      RCSR,TEMP
LET TEMP := TEMP + #44
      ADD      #44,TEMP
;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
      BIC      #BIT6,@TEMP

ENDSRV
BGNSRV RCVR5SRV

;PUT INTR IDENTIFIER IN INTRTABLE
LET (R4)+ := RCH5
      MOV      RCH5,(R4)+
;GENERATE CSR ADDRESS
LET TEMP := RCSR
      MOV      RCSR,TEMP
LET TEMP := TEMP + #50
      ADD      #50,TEMP
;ONE INTR IS ALL WE WANT FROM HERE
LET @TEMP := @TEMP CLR.BY #BIT6
      BIC      #BIT6,@TEMP
  
```

```

3310 011220
3311 011220 000002 RTI
3312 011222
3313 011222 XMIT5SRV:
3314
3315 011222
3316 011222 013724 011444
3317
3318 011226
3319 011226 013737 001262 011522
3320 011234
3321 011234 062737 000054 011522
3322
3323 011242
3324 011242 042777 000100 000252
3325 011250
3326 011250 000002 RTI
3327 011252
3328 011252 RCVR6SRV:
3329
3330 011252
3331 011252 013724 011426
3332
3333 011256
3334 011256 013737 001262 011522
3335 011264
3336 011264 062737 000060 011522
3337
3338 011272
3339 011272 042777 000100 000222
3340 011300
3341 011300 000002 RTI
3342 011302
3343 011302 XMIT6SRV:
3344
3345 011302
3346 011302 013724 011446
3347
3348 011306
3349 011306 013737 001262 011522
3350 011314
3351 011314 062737 000064 011522
3352
3353 011322
3354 011322 042777 000100 000172
3355 011330
3356 011330 000002 RTI
3357 011332
3358 011332 RCVR7SRV:
3359
3360 011332
3361 011332 013724 011430
3362
3363 011336
3364 011336 013737 001262 011522
3365 011344
  
```

```

ENDSRV
BGNSRV XMIT5SRV
      :PUT INTR IDENTIFIER IN INTRTABLE
      LET (R4)+ := TCH5
      MOV TCH5,(R4)+
      :GENERATE CSR ADDRESS
      LET TEMP := RCSR
      MOV RCSR,TEMP
      LET TEMP := TEMP + #54
      ADD #54,TEMP
      :ONE INTR IS ALL WE WANT FROM HERE
      LET @TEMP := @TEMP CLR.BY #BIT6
      BIC #BIT6,@TEMP
ENDSRV
BGNSRV RCVR6SRV
      :PUT INTR IDENTIFIER IN INTRTABLE
      LET (R4)+ := RCH6
      MOV RCH6,(R4)+
      :GENERATE CSR ADDRESS
      LET TEMP := RCSR
      MOV RCSR,TEMP
      LET TEMP := TEMP + #60
      ADD #60,TEMP
      :ONE INTR IS ALL WE WANT FROM HERE
      LET @TEMP := @TEMP CLR.BY #BIT6
      BIC #BIT6,@TEMP
ENDSRV
BGNSRV XMIT6SRV
      :PUT INTR IDENTIFIER IN INTRTABLE
      LET (R4)+ := TCH6
      MOV TCH6,(R4)+
      :GENERATE CSR ADDRESS
      LET TEMP := RCSR
      MOV RCSR,TEMP
      LET TEMP := TEMP + #64
      ADD #64,TEMP
      :ONE INTR IS ALL WE WANT FROM HERE
      LET @TEMP := @TEMP CLR.BY #BIT6
      BIC #BIT6,@TEMP
ENDSRV
BGNSRV RCVR7SRV
      :PUT INTR IDENTIFIER IN INTRTABLE
      LET (R4)+ := RCH7
      MOV RCH7,(R4)+
      :GENERATE CSR ADDRESS
      LET TEMP := RCSR
      MOV RCSR,TEMP
      LET TEMP := TEMP + #70
  
```


3366 011344 062737 000070 011522
 3367
 3368 011352
 3369 011352 042777 000100 000142
 3370 011360
 3371 011360 000002
 3372 011362
 3373 011362
 3374
 3375 011362
 3376 011362 013724 011450
 3377
 3378 011366
 3379 011366 013737 001262 011522
 3380 011374
 3381 011374 062737 000074 011522
 3382
 3383 011402
 3384 011402 042777 000100 000112
 3385 011410
 3386 011410 000002
 3387 011412
 3388
 3389
 3390
 3391
 3392
 3393
 3394
 3395
 3396 011412 000000
 3397 011414 000001
 3398 011416 000002
 3399 011420 000003
 3400 011422 000004
 3401 011424 000005
 3402 011426 000006
 3403 011430 000007
 3404 011432 000010
 3405 011434 000011
 3406 011436 000012
 3407 011440 000013
 3408 011442 000014
 3409 011444 000015
 3410 011446 000016
 3411 011450 000017
 3412
 3413
 3414
 3415
 3416
 3417
 3418 011452 000020
 3419 011512
 3420
 3421 011512 000000

RTI

XMIT7SRV:

SRVEND: RTI

***CHANNEL IDENTIFIER TABLE**
 ;THE ENTRIES FROM THIS
 ;TABLE ARE PLACED IN THE INTRTABLE
 ;IN THE ORDER THAT THE INTR'S
 ;OCCUR. (EXCLUDING NON-ACTIVE CH'S)

RCH0: .WORD 0
 RCH1: .WORD 1
 RCH2: .WORD 2
 RCH3: .WORD 3
 RCH4: .WORD 4
 RCH5: .WORD 5
 RCH6: .WORD 6
 RCH7: .WORD 7
 TCH0: .WORD 10
 TCH1: .WORD 11
 TCH2: .WORD 12
 TCH3: .WORD 13
 TCH4: .WORD 14
 TCH5: .WORD 15
 TCH6: .WORD 16
 TCH7: .WORD 17

;THIS TABLE WILL CONTAIN ENTRIES
 ;REPRESENTING EACH INTR IN THE ORDER
 ;THAT IT OCCURED
 ; S/B IN THE ABOVE ORDER

INTRTABLE: .BLKW 16.
 TABEND:

CHMASK: .WORD 0 ; INDICATE WHICH CH'S ARE ACTIVE FOR THIS TEST

ADD #70,TEMP
 ;ONE INTR IS ALL WE WANT FROM HERE
 LET @TEMP := @TEMP CLR.BY #BIT6
 BIC #BIT6,@TEMP

ENDSRV

BGNSRV XMIT7SRV

;PUT INTR IDENTIFIER IN INTRTABLE
 LET (R4)+ := TCH7

MOV TCH7,(R4)+
 ;GENERATE CSR ADDRESS
 LET TEMP := RCSR

MOV RCSR,TEMP
 LET TEMP := TEMP + #74

ADD #74,TEMP
 ;ONE INTR IS ALL WE WANT FROM HERE
 LET @TEMP := @TEMP CLR.BY #BIT6
 BIC #BIT6,@TEMP

ENDSRV

3422 011514 000000
 3423 011516 000000
 3424 011520 000000
 3425 011522 000000

CHCNT: .WORD 0 ;CHANNEL COUNTER
 PRIERR: .WORD 0 ;ERROR WORD, CONTAINS NO. OF INTR OUT OF ORDER
 TCNT: .WORD 0 ;TABLE PASS CTR
 TEMP: .WORD 0 ;FOR SERVICE ROUTINES

 :TEST 22 TEST DATA TRANSFERS WITH ALL ACTIVE LINES INTERRUPTING
 : IN THIS WE'LL ENABLE INTERRUPTS ON ALL CHANNELS
 : FOR THIS DLV11-J. THEN WE'LL XMIT AN INCREMENTING
 : DATA PATTERN VIA INTERRUPTS AND RECORD THE RECEIVER
 : INTR. IN THE RECEIVER STATUS TABLE.
 : NOTE: DOUBLE BUFFERING CANNOT BE TESTED AT ITS MAX SPEED
 : BECAUSE OF APT CONSIDERATIONS. I.E. APT SENDS
 : 'BREAKS' WHICH CAUSE OVERRUN ERRORS. THEREFORE
 : THE XMIT IE IS NOT ENABLED AGAIN UNTIL THE RECV
 : HAS OBTAINED THE PREVIOUS WORD. THIS TEST WILL
 : NOT BE EXECUTED IF THERE ARE NO DLV11-J'S.
 :*****

3440
 3441 011524 000004
 3442 011526 012737 000022 001200
 3443
 3444 011534
 3445 011534 005037 016122
 3446
 3447
 3448
 3449 011540
 3450 011540 012700 014114
 3451 011544
 3452 011544
 3453 011544
 3454 011544 005020
 3455 011546
 3456 011546 020027 016114
 3457 011552 001374
 3458
 3459
 3460 011554
 3461 011554 012701 012534
 3462 011560
 3463 011560 013700 001260
 3464 011564
 3465 011564
 3466 011564
 3467 011564 010120
 3468 011566
 3469 011566 012720 000340
 3470 011572
 3471 011572 062701 000072
 3472 011576
 3473 011576 010120
 3474 011600
 3475 011600 012720 000340
 3476 011604
 3477 011604 062701 000044

TST22: SCOPE
 MOV #22,\$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
 ;CLR ERROR WORD,
 LET ERWRD := #0
 CLR ERWRD
 ;CLEAR OUT RECEIVER TABLES
 LET R0 := #CHOTAB
 MOV #CHOTAB,R0
 REPEAT
 \$50214:
 LET (R0)+ := #0
 CLR (R0)+
 UNTIL R0 EQ #STATEND
 CMP R0,#STATEND
 BNE \$50214
 ;SET UP ALL VECTORS
 LET R1 := #ROSRV
 MOV #ROSRV,R1
 LET R0 := DLVEC
 MOV DLVEC,R0
 REPEAT
 \$50215:
 LET (R0)+ := R1
 MOV R1,(R0)+
 LET (R0)+ := #PR7
 MOV #PR7,(R0)+
 LET R1 := R1 + #72
 ADD #72,R1
 LET (R0)+ := R1
 MOV R1,(R0)+
 LET (R0)+ := #PR7
 MOV #PR7,(R0)+
 LET R1 := R1 + #44
 ADD #44,R1

3478	011610				UNTIL R1 EQ #SEREND	
3479	011610	020127	014114			CMP R1,#SEREND
3480	011614	001363				BNE \$50215
3481					:SET PRIORITY TO 7	
3482	011616			SETPRI #PR7		
3483						
3484					:ENABLE INTR'S ON ALL ACTIVE LINES	
3485					:THIS MODULE	
3486	011630				LET RO := \$BASE	
3487	011630	013700	001250			MOV \$BASE,RO
3488					:INITIALIZE	
3489	011634				LET CHMSK := \$DEVM	
3490	011634	013737	001252	016114		MOV \$DEVM,CHMSK
3491	011642				LET CHCTR := #0	
3492	011642	005037	016116			CLR CHCTR
3493					:ACTIVE CHANNEL CTR	
3494	011646				LET ACTCH := #0	
3495	011646	005037	016120			CLR ACTCH
3496	011652				REPEAT	
3497	011652					\$50216:
3498	011652				IF #BIT0 SETIN CHMSK THEN	
3499	011652	032737	000001	016114		BIT #BIT0,CHMSK
3500	011660	001002				BNE .+6
3501	011662	000137	011716			JMP \$50217
3502					:# OF ACTIVE CH	
3503	011666				LET ACTCH := ACTCH + #1	
3504	011666	005237	016120			INC ACTCH
3505					:SET RCSR IE	
3506	011672				LET (R0) := (R0) SET.BY #BIT6	
3507	011672	052710	000100			BIS #BIT6,(R0)
3508	011676				LET RO := RO + #4	
3509	011676	062700	000004			ADD #4,RO
3510					:SET XCSR IE	
3511	011702				LET (R0) := (R0) SET.BY #BIT6	
3512	011702	052710	000100			BIS #BIT6,(R0)
3513	011706				LET RO := RO - #4	
3514	011706	162700	000004			SUB #4,RO
3515	011712				LET CHCTR := CHCTR + #1	
3516	011712	005237	016116			INC CHCTR
3517	011716				ENDIF	
3518	011716					\$50217:
3519	011716				LET RO := RO + #10	
3520	011716	062700	000010			ADD #10,RO
3521	011722				LET CHMSK := CHMSK SHIFT -1	
3522	011722	006237	016114			ASR CHMSK
3523						
3524						
3525	011726				:4 OR 10 MODULES	
3526	011726	032737	000001	001306	IF #BIT00 SETIN KDDL V THEN	
3527	011734	001002				BIT #BIT00,KDDL V
3528	011736	000137	011750			BNE .+6
3529	011742				LET .+12 := #4	
3530	011742	012737	000004	011754		JMP \$50220
3531	011750				ENDIF	
3532	011750					MOV #4,+.12
3533	011750					\$50220:
					UNTIL CHCTR EQ #10	

3534	011750	023727	016116	000010
3535	011756	001335		
3536				
3537				
3538	011760			
3539	011760	012737	014114	016252
3540	011766			
3541	011766	012737	014314	016254
3542	011774			
3543	011774	012737	014514	016256
3544	012002			
3545	012002	012737	014714	016260
3546	012010			
3547	012010	012737	015114	016262
3548	012016			
3549	012016	012737	015314	016264
3550	012024			
3551	012024	012737	015514	016266
3552	012032			
3553	012032	012737	015714	016270
3554				
3555				
3556				
3557				
3558	012040			
3559	012040	005037	016126	
3560				
3561	012044			
3562	012044	005004		
3563				
3564	012046			
3565	012046	012705	016130	
3566	012052			
3567	012052			
3568	012052			
3569	012052	032737	000001	001220
3570	012060	001402		
3571	012062	000137	012102	
3572	012066			
3573	012066	016425	016212	
3574	012072			
3575	012072	016425	016212	
3576	012076			
3577	012076	000137	012112	
3578	012102			
3579	012102			
3580	012102	016425	016232	
3581	012106			
3582	012106	016425	016232	
3583	012112			
3584	012112			
3585	012112			
3586	012112	062704	000002	
3587	012116			
3588	012116	032737	000001	001306
3589	012124	001002		

IF #BIT00 SETIN KDDL V THEN

```

CMP      CHCTR,#10
BNE      $50216

;INIT BUFFER POINTERS FOR SERVICE ROUTINES
LET X0 := #CH0TAB
MOV      #CH0TAB,X0
LET X1 := #CH1TAB
MOV      #CH1TAB,X1
LET X2 := #CH2TAB
MOV      #CH2TAB,X2
LET X3 := #CH3TAB
MOV      #CH3TAB,X3
LET X4 := #CH4TAB
MOV      #CH4TAB,X4
LET X5 := #CH5TAB
MOV      #CH5TAB,X5
LET X6 := #CH6TAB
MOV      #CH6TAB,X6
LET X7 := #CH7TAB
MOV      #CH7TAB,X7

;FILL XBUF WORDS
;FROM TMP0 THRU TMP7E
;START WORD CH 0 ALWAYS 0
LET TMP0 := #0
CLR      TMP0

;OFFSET TO INDEX
LET R4 := #0
CLR      R4

;PTR TO TABLE TO BE FILLED
LET R5 := #TMP0E
MOV      #TMP0E,R5

REPEAT
$50221:
IF #BIT0 NOTSETIN $USWR THEN
  BIT     #BIT0,$USWR
  BEQ     .+6
  JMP     $50222
  LET (R5)+ := TABL7(R4)
  MOV     TABL7(R4),(R5)+
  LET (R5)+ := TABL7(R4)
  MOV     TABL7(R4),(R5)+
ELSE
  JMP     $50223
$50222:
  LET (R5)+ := TABL8(R4)
  MOV     TABL8(R4),(R5)+
  LET (R5)+ := TABL8(R4)
  MOV     TABL8(R4),(R5)+
ENDIF
$50223:
LET R4 := R4 + #2
ADD      #2,R4
;4 OR 10 MODULES
BIT     #BIT00,KDDL V
BNE     .+6
  
```



```

                                JMP      $50224
                                ;TEST 4 MODULES
                                MOV      #10,..+10
                                $50224:
UNTIL R4 EQ #20
                                CMP      R4,#20
                                BNE      $50221
;INIT TMP10
;WILL BE DECR BY RECVR SERVICE ROUTINES TILL 0
LET TMP10 := ACTCH
                                MOV      ACTCH,TMP10
                                ;SET PRIORITY TO 0

REPEAT
                                $50225:
UNTIL TMP10 EQ #0
                                TST      TMP10
                                BNE      $50225
;WAIT FOR RECV'S TO CATCH UP
LET TMP10 := #100000
                                MOV      #100000C,TMP10
REPEAT
                                $50226:
LET TMP10 := TMP10 - #1
                                DEC      TMP10
UNTIL TMP10 EQ #0
                                TST      TMP10
                                BNE      $50226

                                ;DISABLE ALL INTR'S
BRES#T

                                ;CHECK RECEIVER TABLE
                                ;RESET TMP0-7
                                LET TMP0 := #0
                                CLR      TMP0
                                LET TMP1 := TMP0E
                                MOV      TMP0E,TMP1
                                LET TMP2 := TMP1E
                                MOV      TMP1E,TMP2
                                LET TMP3 := TMP2E
                                MOV      TMP2E,TMP3
                                MOV      TMP3E,TMP4
                                MOV      TMP4E,TMP5
                                MOV      TMP5E,TMP6
                                MOV      TMP6E,TMP7

                                ;INITIALIZE
                                LET CHCTR := #0

```

3646	012274	005037	016116	
3647	012300			
3648	012300	013737	001252	016114
3649	012306			
3650	012306			
3651	012306			
3652	012306	032737	000001	016114
3653	012314	001002		
3654	012316	000137	012432	
3655	012322			
3656	012322	013700	016116	
3657				
3658	012326			
3659	012326	006300		
3660				
3661	012330			
3662	012330	016002	016172	
3663				
3664	012334			
3665	012334	006300		
3666				
3667	012336			
3668	012336	016001	016126	
3669	012342			
3670	012342	062700	000002	
3671				
3672	012346			
3673	012346	016003	016126	
3674				
3675	012352			
3676	012352			
3677	012352			
3678	012352	032712	100000	
3679	012356	001002		
3680	012360	000137	012372	
3681	012364			
3682	012364	052737	000400	016122
3683	012372			
3684	012372			
3685				
3686	012372			
3687	012372	121201		
3688	012374	001002		
3689	012376	000137	012406	
3690	012402			
3691	012402	005237	016122	
3692	012406			
3693	012406			
3694				
3695	012406			
3696	012406	005201		
3697				
3698	012410			
3699	012410	062702	000002	
3700				
3701	012414			

```

      CLR      CHCTR
      LET CHMSK := $DEV
      MOV      $DEV,CHMSK
REPEAT
      $50227:
      IF #BIT0 SET IN CHMSK THEN
        BIT     #BIT0,CHMSK
        BNE     +6
        JMP     $50230
      LET R0 := CHCTR
      MOV      CHCTR,R0
      ;OFFSET TO INDEX (2X)
      LET R0 := R0 SHIFT +1
      ASL      R0
      ;GET TABLE LOC FOR THAT CH.
      LET R2 := TABL6(R0)
      MOV      TABL6(R0),R2
      ;OFFSET TO INDEX (4X)
      LET R0 := R0 SHIFT +1
      ASL      R0
      ;GET EXPECTED STARTING WORD
      LET R1 := TMP0(R0)
      MOV      TMP0(R0),R1
      LET R0 := R0 + #2
      ADD      #2,R0
      ;GET MAX WORD FOR THAT CH.
      LET R3 := TMP0(R0)
      MOV      TMP0(R0),R3
REPEAT
      $50231:
      IF #BIT15 SET IN (R2) THEN
        BIT     #BIT15,(R2)
        BNE     +6
        JMP     $50232
      LET ERWRD := ERWRD SET BY #BIT8
      BIS      #BIT8,ERWRD
      ENDIF
      $50232:
      ;CHECK FOR DATA COMPARE ERROR
      IFB (R2) NE R1 THEN
        CMPB    (R2),R1
        BNE     +6
        JMP     $50233
      LET ERWRD := ERWRD + #1
      INC      ERWRD
      ENDIF
      $50233:
      ;BUMP EXPECTED DATA WORD
      LET R1 := R1 + #1
      INC      R1
      ;BUMP TABLE PTR
      LET R2 := R2 + #2
      ADD      #2,R2
      UNTIL R1 EQ R3 OR ERWRD NE #0
  
```


3702	012414	020103		
3703	012416	001403		
3704	012420	005737	016122	
3705	012424	001752		
3706	012426			
3707	012426			
3708	012426	005237	016116	
3709	012432			
3710	012432			
3711	012432			
3712	012432	006237	016114	
3713	012436			
3714	012436	032737	000001	001306
3715	012444	001002		
3716	012446	000137	012460	
3717	012452			
3718	012452	012737	000004	012464
3719	012460			
3720	012460			
3721	012460			
3722	012460	023727	016116	000010
3723	012466	001403		
3724	012470	005737	016122	
3725	012474	001704		
3726	012476			
3727				
3728	012476			
3729	012476	032737	000400	016122
3730	012504	001002		
3731	012506	000137	012514	
3732				
3733	012512			
3734	012512	104270		
3735	012514			
3736	012514			
3737	012514			
3738	012514	105737	016122	
3739	012520	001002		
3740	012522	000137	012530	
3741				
3742	012526			
3743	012526	104271		
3744	012530			
3745	012530			
3746	012530	000137	021324	
3747				
3748				
3749				
3750				
3751				
3752	012534			
3753	012534			
3754	012534			
3755	012534	013737	001262	016124
3756	012542			
3757	012542	062737	000002	016124

```

CMP      R1,R3      $50234
BEQ      $50234
TST      ERWRD      $50231
BEQ      $50234:
LET CHCTR := CHCTR + #1
INC      CHCTR
ENDIF
$50230:
LET CHMSK := CHMSK SHIFT -1
ASR      CHMSK
IF #BIT00 SETIN KDDL V THEN
BIT      #BIT00,KDDL V
BNE      .+6
JMP      $50235
LET .+12 := #4
;TEST 4 MODULES
MOV      #4, .+12
ENDIF
$50235:
UNTIL CHCTR EQ #10 OR ERWRD NE #0
CMP      CHCTR,#10
BEQ      $50236
TST      ERWRD
BEQ      $50227
$50236:
IF #BIT8 SETIN ERWRD THEN
BIT      #BIT8,ERWRD
BNE      .+6
JMP      $50237
;ERROR FLAG UP AFTER TRANSFER
ERRHRD 270
ERROR    270
ENDIF
$50237:
IFB ERWRD NE #0 THEN
TSTB     ERWRD
BNE      .+6
JMP      $50240
;DATA COMPARE ERROR
ERRHRD 271
ERROR    271
ENDIF
$50240:
JMP      ENDDER
;EXIT TEST
*****
*          START OF SERVICE ROUTINES
*****
BGNSRV ROSRV
ROSRV:
LET TMP := RCSR
MOV      RCSR,TMP
LET TMP := TMP + #2
ADD      #2,TMP

```

```

3758 012550
3759 012550 017777 003350 003474
3760 012556
3761 012556 062737 000002 016252
3762 012564
3763 012564 023737 016126 016130
3764 012572 001002
3765 012574 000137 012620
3766
3767 012600
3768 012600 062737 000002 016124
3769
3770 012606
3771 012606 052777 000100 003310
3772 012614
3773 012614 000137 012624
3774 012620
3775
3776 012620
3777 012620 005337 016170
3778 012624
3779 012624
3780 012624
3781 012624 000002
3782
3783 012626
3784 012626
3785 012626
3786 012626 013737 001262 016124
3787 012634
3788 012634 062737 000006 016124
3789 012642
3790 012642 013777 016126 003254
3791 012650
3792 012650 005237 016126
3793
3794 012654
3795 012654 162737 000002 016124
3796
3797
3798 012662
3799 012662 042777 000100 003234
3800 012670
3801 012670 000002
3802
3803 012672
3804 012672
3805 012672
3806 012672 013737 001262 016124
3807 012700
3808 012700 062737 000012 016124
3809 012706
3810 012706 017777 003212 003340
3811 012714
3812 012714 062737 000002 016254
3813 012722
  
```

RTI

XOSRV:

RTI

R1SRV:

```

      LET @X0 := @TMP
      MOV      @TMP,@X0
      LET X0 := X0 + #2
      ADD      #2,X0
      IF TMPO NE TMPOE THEN
        CMP      TMPO, TMPOE
        BNE      +6
        JMP      $50241
      ;GO TO XCSR
      LET TMP := TMP + #2
      ADD      #2,TMP
      ;ENABLE XMIT INTERRU
      LET @TMP := @TMP SET.BY #BIT6
      BIS      #BIT6,@TMP
      ELSE
        JMP      $50242
      $50241:
      ;ALL DONE
      LET TMP10 := TMP10 - #1
      DEC      TMP10
      ENDIF
      $50242:
ENDSRV

BGNSRV XOSRV
      LET TMP := RCSR
      MOV      RCSR,TMP
      LET TMP := TMP + #6
      ADD      #6,TMP
      LET @TMP := TMPO
      MOV      TMPO,@TMP
      LET TMPO := TMPO + #1
      INC      TMPO
      ;GO BACK TO XCSR
      LET TMP := TMP - #2
      SUB      #2,TMP
      ;DISABLE XMIT INTERRUPTS,
      ;NOT EXER DOUBLE BUFFERING
      LET @TMP := @TMP CLR.BY #BIT6
      BIC      #BIT6,@TMP
ENDSRV

BGNSRV R1SRV
      LET TMP := RCSR
      MOV      RCSR,TMP
      LET TMP := TMP + #12
      ADD      #12,TMP
      LET @X1 := @TMP
      MOV      @TMP,@X1
      LET X1 := X1 + #2
      ADD      #2,X1
      IF TMP1 NE TMP1E THEN
  
```


3814 012722 023737 016132 016134
 3815 012730 001002
 3816 012732 000137 012756
 3817
 3818 012736
 3819 012736 062737 000002 016124
 3820
 3821 012744
 3822 012744 052777 000100 003152
 3823 012752
 3824 012752 000137 012762
 3825 012756
 3826
 3827 012756
 3828 012756 005337 016170
 3829 012762
 3830 012762
 3831 012762
 3832 012762 000002
 3833
 3834 012764
 3835 012764
 3836 012764
 3837 012764 013737 001262 016124
 3838 012772
 3839 012772 062737 000016 016124
 3840 013000
 3841 013000 013777 016132 003116
 3842 013006
 3843 013006 005237 016132
 3844
 3845 013012
 3846 013012 162737 000002 016124
 3847
 3848
 3849 013020
 3850 013020 042777 000100 003076
 3851 013026
 3852 013026 000002
 3853
 3854 013030
 3855 013030
 3856 013030
 3857 013030 013737 001262 016124
 3858 013036
 3859 013036 062737 000022 016124
 3860 013044
 3861 013044 017777 003054 003204
 3862 013052
 3863 013052 062737 000002 016256
 3864 013060
 3865 013060 023737 016136 016140
 3866 013066 001002
 3867 013070 000137 013114
 3868
 3869 013074

RTI

X1SRV:

RTI

R2SRV:

CMP TMP1,TMP1E
 BNE +6
 JMP \$50243
 ;GO TO XCSR
 LET TMP := TMP + #2
 ADD #2,TMP
 ;ENABLE XMIT INTERRU
 LET @TMP := @TMP SET.BY #BIT6
 BIS #BIT6,@TMP
 ELSE
 JMP \$50244
 \$50243:
 ;ALL DONE
 LET TMP10 := TMP10 - #1
 DEC TMP10
 ENDIF
 \$50244:
 ENDSRV
 BGNSRV X1SRV
 LET TMP := RCSR
 MOV RCSR,TMP
 LET TMP := TMP + #16
 ADD #16,TMP
 LET @TMP := TMP1
 MOV TMP1,@TMP
 LET TMP1 := TMP1 + #1
 INC TMP1
 ;GO BACK TO XCSR
 LET TMP := TMP - #2
 SUB #2,TMP
 ;DISABLE XMIT INTERRUPTS,
 ;NOT EXER DOUBLE BUFF
 LET @TMP := @TMP CLR.BY #BIT6
 BIC #BIT6,@TMP
 ENDSRV
 BGNSRV R2SRV
 LET TMP := RCSR
 MOV RCSR,TMP
 LET TMP := TMP + #22
 ADD #22,TMP
 LET @X2 := @TMP
 MOV @TMP,@X2
 LET X2 := X2 + #2
 ADD #2,X2
 IF TMP2 NE TMP2E THEN
 CMP TMP2,TMP2E
 BNE +6
 JMP \$50245
 ;GO TO XCSR
 LET TMP := TMP + #2

3870	013074	062737	000002	016124
3871				
3872	013102			
3873	013102	052777	000100	003014
3874	013110			
3875	013110	000137	013120	
3876	013114			
3877				
3878	013114			
3879	013114	005337	016170	
3880	013120			
3881	013120			
3882	013120			
3883	013120	000002		
3884				
3885	013122			
3886	013122			
3887	013122			
3888	013122	013737	001262	016124
3889	013130			
3890	013130	062737	000026	016124
3891	013136			
3892	013136	013777	016136	002760
3893	013144			
3894	013144	005237	016136	
3895				
3896	013150			
3897	013150	162737	000002	016124
3898				
3899				
3900	013156			
3901	013156	042777	000100	002740
3902	013164			
3903	013164	000002		
3904				
3905	013166			
3906	013166			
3907	013166			
3908	013166	013737	001262	016124
3909	013174			
3910	013174	062737	000032	016124
3911	013202			
3912	013202	017777	002716	003050
3913	013210			
3914	013210	062737	000002	016260
3915	013216			
3916	013216	023737	016142	016144
3917	013224	001002		
3918	013226	000137	013252	
3919				
3920	013232			
3921	013232	062737	000002	016124
3922				
3923	013240			
3924	013240	052777	000100	002656
3925	013246			

RTI

X2SRV:

RTI

R3SRV:

```

      ADD      #2,TMP
      ;ENABLE XMIT INTERR
      LET @TMP := @TMP SET.BY #BIT6
      BIS      #BIT6,@TMP
ELSE
      JMP      $50246
      $50245:
      ;ALL DONE
      LET TMP10 := TMP10 - #1
      DEC      TMP10
ENDIF
      $50246:
ENDSRV

BGNSRV X2SRV
      LET TMP := RCSR
      MOV      RCSR,TMP
      LET TMP := TMP + #26
      MOV      RCSR,TMP
      LET @TMP := TMP2
      MOV      TMP2,@TMP
      LET TMP2 := TMP2 + #1
      INC      TMP2
      ;GO BACK TO XCSR
      LET TMP := TMP - #2
      SUB      #2,TMP
      ;DISABLE XMIT INTERRUPTS,
      ;NOT EXER DOUBLE BUFF.
      LET @TMP := @TMP CLR.BY #BIT6
      BIC      #BIT6,@TMP
ENDSRV

BGNSRV R3SRV
      LET TMP := RCSR
      MOV      RCSR,TMP
      LET TMP := TMP + #32
      ADD      #32,TMP
      LET @X3 := @TMP
      MOV      @TMP,@X3
      LET X3 := X3 + #2
      ADD      #2,X3
      IF TMP3 NE TMP3E THEN
      CMP      TMP3,TMP3E
      BNE      +6
      JMP      $50247
      ;GO TO XCSR
      LET TMP := TMP + #2
      ADD      #2,TMP
      ;ENABLE XMIT INTERR
      LET @TMP := @TMP SET.BY #BIT6
      BIS      #BIT6,@TMP
ELSE

```


3926 013246 000137 013256
 3927 013252
 3928
 3929 013252
 3930 013252 005337 016170
 3931 013256
 3932 013256
 3933 013256
 3934 013256 000002
 3935
 3936 013260
 3937 013260
 3938 013260
 3939 013260 013737 001262 016124
 3940 013266
 3941 013266 062737 000036 016124
 3942 013274
 3943 013274 013777 016142 002622
 3944 013302
 3945 013302 005237 016142
 3946
 3947 013306
 3948 013306 162737 000002 016124
 3949
 3950
 3951 013314
 3952 013314 042777 000100 002602
 3953 013322
 3954 013322 000002
 3955
 3956 013324
 3957 013324
 3958 013324
 3959 013324 013737 001262 016124
 3960 013332
 3961 013332 062737 000042 016124
 3962 013340
 3963 013340 017777 002560 002714
 3964 013346
 3965 013346 062737 000002 016262
 3966 013354
 3967 013354 023737 016146 016150
 3968 013362 001002
 3969 013364 000137 013410
 3970
 3971 013370
 3972 013370 062737 000002 016124
 3973
 3974 013376
 3975 013376 052777 000100 002520
 3976 013404
 3977 013404 000137 013414
 3978 013410
 3979
 3980 013410
 3981 013410 005337 016170

X3SRV:

RTI

RTI

R4SRV:

JMP \$50250
 \$50247:
 ;ALL DONE
 LET TMP10 := TMP10 - #1
 DEC TMP10
 ENDIF
 \$50250:
 ENDSRV
 BGNSRV X3SRV
 LET TMP := RCSR
 MOV RCSR, TMP
 LET TMP := TMP + #36
 ADD #36, TMP
 LET @TMP := TMP3
 MOV TMP3, @TMP
 LET TMP3 := TMP3 + #1
 INC TMP3
 ;GO BACK TO XCSR
 LET TMP := TMP - #2
 SUB #2, TMP
 ;DISABLE XMIT INTERRUPTS,
 ;NOT EXER DOUBLE BUFF
 LET @TMP := @TMP CLR.BY #BIT6
 BIC #BIT6, @TMP
 ENDSRV
 BGNSRV R4SRV
 LET TMP := RCSR
 MOV RCSR, TMP
 LET TMP := TMP + #42
 ADD #42, TMP
 LET @X4 := @TMP
 MOV @TMP, @X4
 LET X4 := X4 + #2
 ADD #2, X4
 IF TMP4 NE TMP4E THEN
 CMP TMP4, TMP4E
 BNE +6
 JMP \$50251
 ;GO TO XCSR
 LET TMP := TMP + #2
 ADD #2, TMP
 ;ENABLE XMIT INTERRUPT
 LET @TMP := @TMP SET.BY #BIT6
 BIS #BIT6, @TMP
 ELSE
 JMP \$50252
 \$50251:
 ;ALL DONE
 LET TMP10 := TMP10 - #1
 DEC TMP10

```

3982 013414
3983 013414
3984 013414
3985 013414 000002 RTI
3986
3987 013416
3988 013416 X4SRV:
3989 013416
3990 013416 013737 001262 016124
3991 013424
3992 013424 062737 000046 016124
3993 013432
3994 013432 013777 016146 002464
3995 013440
3996 013440 005237 016146
3997
3998 013444
3999 013444 162737 000002 016124
4000
4001
4002 013452
4003 013452 042777 000100 002444
4004 013460
4005 013460 000002 RTI
4006
4007 013462
4008 013462 R5SRV:
4009 013462
4010 013462 013737 001262 016124
4011 013470
4012 013470 062737 000052 016124
4013 013476
4014 013476 017777 002422 002560
4015 013504
4016 013504 062737 000002 016264
4017 013512
4018 013512 023737 016152 016154
4019 013520 001002
4020 013522 000137 013546
4021
4022 013526
4023 013526 062737 000002 016124
4024
4025 013534
4026 013534 052777 000100 002362
4027 013542
4028 013542 000137 013552
4029 013546
4030
4031 013546
4032 013546 005337 016170
4033 013552
4034 013552
4035 013552
4036 013552 000002 RTI
4037
  
```

```

ENDIF
$50252:
ENDSRV
BGNSRV X4SRV
  LET TMP := RCSR
  MOV RCSR,TMP
  LET TMP := TMP + #46
  ADD #46,TMP
  LET @TMP := TMP4
  MOV TMP4,@TMP
  LET TMP4 := TMP4 + #1
  INC TMP4
  :GO BACK TO XCSR
  LET TMP := TMP - #2
  SUB #2,TMP
  :DISABLE XMIT INTERRUPTS,
  :NOT EXER DOUBLE BUFF.
  LET @TMP := @TMP CLR.BY #BIT6
  BIC #BIT6,@TMP
ENDSRV
BGNSRV R5SRV
  LET TMP := RCSR
  MOV RCSR,TMP
  LET TMP := TMP + #52
  ADD #52,TMP
  LET @X5 := @TMP
  MOV @TMP,@X5
  LET X5 := X5 + #2
  ADD #2,X5
  IF TMP5 NE TMP5E THEN
    CMP TMP5,TMP5E
    BNE +6
    JMP $50253
  :GO TO XCSR
  LET TMP := TMP + #2
  ADD #2,TMP
  :ENABLE XMIT INTERR
  LET @TMP := @TMP SET.BY #BIT6
  BIS #BIT6,@TMP
ELSE
  JMP $50254
  $50253:
  :ALL DONE
  LET TMP10 := TMP10 - #1
  DEC TMP10
ENDIF
$50254:
ENDSRV
  
```


4038 013554
4039 013554
4040 013554
4041 013554 013737 001262 016124
4042 013562
4043 013562 062737 000056 016124
4044 013570
4045 013570 013777 016152 002326
4046 013576
4047 013576 005237 016152
4048
4049 013602
4050 013602 162737 000002 016124
4051
4052
4053 013610
4054 013610 042777 000100 002306
4055 013616
4056 013616 000002
4057
4058 013620
4059 013620
4060 013620
4061 013620 013737 001262 016124
4062 013626
4063 013626 062737 000062 016124
4064 013634
4065 013634 017777 002264 002424
4066 013642
4067 013642 062737 000002 016266
4068 013650
4069 013650 023737 016156 016160
4070 013656 001002
4071 013660 000137 013704
4072
4073 013664
4074 013664 062737 000002 016124
4075
4076 013672
4077 013672 052777 000100 002224
4078 013700
4079 013700 000137 013710
4080 013704
4081
4082 013704
4083 013704 005337 016170
4084 013710
4085 013710
4086 013710
4087 013710 000002
4088
4089 013712
4090 013712
4091 013712
4092 013712 013737 001262 016124
4093 013720

X5SRV:

RTI

R6SRV:

RTI

X6SRV:

BGNSRV X5SRV

LET TMP := RCSR
MOV RCSR,TMP
LET TMP := TMP + #56
ADD #56,TMP
LET @TMP := TMP5
MOV TMP5,@TMP
LET TMP5 := TMP5 + #1
INC TMP5
;GO BACK TO XCSR
LET TMP := TMP - #2
SUB #2,TMP
;DISABLE XMIT INTERRUPTS,
;NOT EXER DOUBLE BUFF.
LET @TMP := @TMP CLR.BY #BIT6
BIC #BIT6,@TMP

ENDSRV

BGNSRV R6SRV

LET TMP := RCSR
MOV RCSR,TMP
LET TMP := TMP + #62
ADD #62,TMP
LET @X6 := @TMP
MOV @TMP,@X6
LET X6 := X6 + #2
ADD #2,X6
IF TMP6 NE TMP6E THEN
CMP TMP6,TMP6E
BNE +6
JMP \$50255
;GO TO XCSR
LET TMP := TMP + #2
ADD #2,TMP
;ENABLE XMIT INTERRUPT
LET @TMP := @TMP SET.BY #BIT6
BIS #BIT6,@TMP
ELSE
JMP \$50256
\$50255:
;ALL DONE
LET TMP10 := TMP10 - #1
DEC TMP10
ENDIF
\$50256:

ENDSRV

BGNSRV X6SRV

LET TMP := RCSR
MOV RCSR,TMP
LET TMP := TMP + #66

4094	013720	062737	000066	016124	
4095	013726				
4096	013726	013777	016156	002170	
4097	013734				
4098	013734	005237	016156		
4099					
4100	013740				
4101	013740	162737	000002	016124	
4102					
4103					
4104	013746				
4105	013746	042777	000100	002150	
4106	013754				
4107	013754	000002			RTI
4108					
4109	013756				
4110	013756				R7SRV:
4111	013756				
4112	013756	013737	001262	016124	
4113	013764				
4114	013764	062737	000072	016124	
4115	013772				
4116	013772	017777	002126	002270	
4117	014000				
4118	014000	062737	000002	016270	
4119	014006				
4120	014006	023737	016162	016164	
4121	014014	001002			
4122	014016	000137	014042		
4123					
4124	014022				
4125	014022	062737	000002	016124	
4126					
4127	014030				
4128	014030	052777	000100	002066	
4129	014036				
4130	014036	000137	014046		
4131	014042				
4132					
4133	014042				
4134	014042	005337	016170		
4135	014046				
4136	014046				
4137	014046				
4138	014046	000002			RTI
4139					
4140	014050				
4141	014050				X7SRV:
4142	014050				
4143	014050	013737	001262	016124	
4144	014056				
4145	014056	062737	000076	016124	
4146	014064				
4147	014064	013777	016162	002032	
4148	014072				
4149	014072	005237	016162		

```

      ADD      #66,TMP
      LET @TMP := TMP6
      MOV      TMP6,@TMP
      LET TMP6 := TMP6 + #1
      INC      TMP6
      ;GO BACK TO XCSR
      LET TMP := TMP - #2
      SUB      #2,TMP
      ;DISABLE XMIT INTERRUPTS,
      ;NOT EXER DOUBLE BUFF.
      LET @TMP := @TMP CLR.BY #BIT6
      BIC      #BIT6,@TMP
ENDSRV

BGNSRV R7SRV
      LET TMP := RCSR
      MOV      RCSR,TMP
      LET TMP := TMP + #72
      ADD      #72,TMP
      LET @X7 := @TMP
      MOV      @TMP,@X7
      LET X7 := X7 + #2
      ADD      #2,X7
      IF TMP7 NE TMP7E THEN
        CMP      TMP7,TMP7E
        BNE      +6
        JMP      $50257
      ;GO TO XCSR
      LET TMP := TMP + #2
      ADD      #2,TMP
      ;ENABLE XMIT INTERR
      LET @TMP := @TMP SET.BY #BIT6
      BIS      #BIT6,@TMP
      ELSE
        JMP      $50260
      ;ALL DONE
      LET TMP10 := TMP10 - #1
      DEC      TMP10
      ENDIF
      $50260:
ENDSRV

BGNSRV X7SRV
      LET TMP := RCSR
      MOV      RCSR,TMP
      LET TMP := TMP + #76
      ADD      #76,TMP
      LET @TMP := TMP7
      MOV      TMP7,@TMP
      LET TMP7 := TMP7 + #1
      INC      TMP7
  
```



```

4150
4151 014076
4152 014076 162737 000002 016124
4153
4154
4155 014104
4156 014104 042777 000100 002012
4157 014112
4158 014112 000002
4159 014114
4160
4161
4162
4163 014114 000100
4164 014314 000100
4165 014514 000100
4166 014714 000100
4167 015114 000100
4168 015314 000100
4169 015514 000100
4170 015714 000100
4171 016114
4172 016114 000000
4173 016116 000000
4174 016120 000000
4175 016122 000000
4176
4177
4178 016124 000000
4179
4180 016126 000000
4181 016130 000000
4182 016132 000000
4183 016134 000000
4184 016136 000000
4185 016140 000000
4186 016142 000000
4187 016144 000000
4188 016146 000000
4189 016150 000000
4190 016152 000000
4191 016154 000000
4192 016156 000000
4193 016160 000000
4194 016162 000000
4195 016164 000002
4196
4197 016170 000000
4198
4199 016172 014114
4200 016174 014314
4201 016176 014514
4202 016200 014714
4203 016202 015114
4204 016204 015314
4205 016206 015514
  
```

RTI
 SEREND:

;** RECEIVER STATUS TABLES **

CH0TAB: .BLKW 100
 CH1TAB: .BLKW 100
 CH2TAB: .BLKW 100
 CH3TAB: .BLKW 100
 CH4TAB: .BLKW 100
 CH5TAB: .BLKW 100
 CH6TAB: .BLKW 100
 CH7TAB: .BLKW 100

STATEND:

CHMSK: .WORD 0
 CHCTR: .WORD 0
 ACTCH: .WORD 0
 ERWRD: .WORD 0

TMP: .WORD 0

TMP0: 0
 TMP0E: 0
 TMP1: 0
 TMP1E: 0
 TMP2: 0
 TMP2E: 0
 TMP3: 0
 TMP3E: 0
 TMP4: 0
 TMP4E: 0
 TMP5: 0
 TMP5E: 0
 TMP6: 0
 TMP6E: 0
 TMP7: 0
 TMP7E: .BLKW 2

TMP10: .WORD

TABL6: CH0TAB
 CH1TAB
 CH2TAB
 CH3TAB
 CH4TAB
 CH5TAB
 CH6TAB

:GO BACK TO XCSR
 LET TMP := TMP - #2
 SUB #2,TMP
 :DISABLE XMIT INTERRUPTS,
 :NOT EXER DOUBLE BUFF.
 LET @TMP := @TMP CLR.BY #BIT6
 BIC #BIT6,@TMP

ENDSRV

:7 BIT WDS 8 BIT WDS
 : 0- 37 0- 77
 : 40- 77 100-177
 :100-137 200-277
 :140-177 300-377

:MASK OF ACTIVE CHANNELS
 :CHANNEL CTR
 :ACTIVE CHANNEL CTR
 :ERROR WORD, LOW BYTE IS NO. OF DATA ERRORS
 :HIGH BYTE INDICATES ERROR FLAG DETECTED

:TEMP STORAGE FOR SERVICE ROUTINES

:START WORD FOR CH 0
 :END WORD FOR CH 0
 :START 1
 :END 1
 :START 2
 :END 2
 :START 3
 :END 3

4206 016210 015714
4207
4208 016212 000040
4209 016214 000100
4210 016216 000140
4211 016220 000200
4212 016222 000240
4213 016224 000300
4214 016226 000340
4215 016230 000400
4216
4217 016232 000100
4218 016234 000200
4219 016236 000300
4220 016240 000400
4221 016242 000500
4222 016244 000600
4223 016246 000700
4224 016250 001000
4225 016252 000000
4226 016254 000000
4227 016256 000000
4228 016260 000000
4229 016262 000000
4230 016264 000000
4231 016266 000000
4232 016270 000000
4233
4234

CH7TAB

TABL7: 40 ;7 BIT WORD TABLE

100
140
200
240
300
340
400

TABL8: 100 ;8 BIT WORD TABLE

200
300
400
500
600
700
1000

X0: .WORD 0
X1: .WORD 0
X2: .WORD 0
X3: .WORD 0
X4: .WORD 0
X5: .WORD 0
X6: .WORD 0
X7: .WORD 0

4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290016272
016272
016272 016537 000004 016512
016300 016537 000000 016514
016306 105037 016516

016312
016312

016312 036577 000002 000172
016320 001402
016322 000137 016336
016326 105037 016517
016332
016332 000137 016344
016336
016336 112737 177777 016517
016344
016344
016344
016344 123765 016517 000006
016352 001402*****
* ROUTINE:TIMER
* THIS ROUTINE IS USED TO TEST THE STATUS OF ANY BIT
* IN ANY REGISTER.
* INPUTS:
* HOWLONG THE MAXIMUM AMOUNT OF TIME TO SPEND IN
* THIS ROUTINE.
* WHICHBIT A MASK WITH THE BIT(S) SET THAT ARE
* TO BE CHECKED.
* REG A POINTER TO THE REGISTER TO BE CHECKED
* SETCLR THE DESIRED RESULTS
* EITHER #-1 OR #CLEAR
* OUTPUT:
* THE 'C' BIT IS SET TO INDICATE AN ERROR
* BUT IT IS TESTED BY THE IF.ERROR STATEMENT
*
* NOTE:: THE USE OF (R5) IS PART OF THE LINKAGE
* MECHANISM BETWEEN THE CALLER AND THE CALLED
*****ROUTINE TIMER <HOWLONG,WHICHBIT,REG,SETCLR>
TIMER:
LET REGSAV := REG(R5) ; GET POINTER TO REGISTER
LET TIMSAV := HOWLONG(R5) ; SAVE HOWLONG FOR QUICKER RETRIEVAL
MOV REG(R5),REGSAV
MOV HOWLONG(R5),TIMSAV
LET FLAG :B= #0 ; INITIALIZE THE EXIT FLAG
CLRB FLAG

; START OF AN INFINITE LOOP
LOOP
\$50263:
; TEST TO SEE IF WHICHBIT IS SET
IF WHICHBIT(R5) NOTSETIN @REGSAV THEN
BIT WHICHBIT(R5),@REGSAV
BEQ +6
JMP \$50265
LET HOLDSC :B= #0
CLRB HOLDSC
ELSE
JMP \$50266
\$50265:
LET HOLDSC :B= #-1 ; REMEMBER THIS
MOVB #-1,HOLDSC
ENDIF
\$50266:
; NOW SEE IF THAT WAS WHAT WE WANTED
IFB HOLDSC EQ SETCLR(R5) THEN
CMPB HOLDSC,SETCLR(R5)
BEQ +6

4291 016354 000137 016366
 4292
 4293 016360
 4294 016360 112737 000001 016516
 4295 016366
 4296 016366
 4297
 4298 016366
 4299 016366 123727 016516 000001
 4300 016374 001002
 4301 016376 000137 016452
 4302 016402
 4303 016402
 4304 016402 005737 016514
 4305 016406 003002
 4306 016410 000137 016452
 4307 016414
 4308
 4309
 4310
 4311 016414
 4312 016414 010546
 4313 016416 010605
 4314 016420 162706 000002
 4315 016424 010546
 4316 016426 012745 000001
 4317 016432 004737 016524
 4318 016436 012605
 4319 016440 010506
 4320 016442 012605
 4321 016444
 4322 016444 005337 016514
 4323 016450
 4324 016450 000720
 4325 016452
 4326
 4327
 4328
 4329
 4330 016452
 4331 016452 032737 000001 016516
 4332 016460 001002
 4333 016462 000137 016504
 4334 016466 005737 016514
 4335 016472 003002
 4336 016474 000137 016504
 4337 016500
 4338 016500 000137 016520
 4339 016504
 4340 016504
 4341
 4342 016504
 4343 016504 000261
 4344 016506 000137 016522
 4345
 4346

: ONLY 2 WAYS TO GET HERE
 : 1). WE RAN OUT OF TIME---ERROR !!
 : 2). THE BIT IS IN THE CORRECT CONDITION--GOOD !!
 IF #BIT00 SET IN FLAG AND TIMSAV GT #0 THEN

RETURN NO.ERROR
 ENDIF

RETURN ERROR

JMP \$50267
 ; JUST THE THING WE NEEDED
 LET FLAG := #BIT00
 MOV #BIT00, FLAG
 ENDIF
 \$50267:
 EXIFB FLAG EQ #BIT00
 CMPB FLAG, #BIT00
 BNE \$50270
 JMP \$50264
 \$50270:
 EXIF TIMSAV LE #0
 TST TIMSAV
 BGT \$50271
 JMP \$50264
 \$50271:
 ; ONE WAY OR THE OTHER, WE ARE DONE
 ; IF WE ARE STILL HERE THEN HANG AROUND A WHILE
 CALL WAIT IN <#1>
 MOV R5, -(SP)
 MOV SP, R5
 SUB #2, SP
 MOV R5, -(SP)
 MOV #1, -(R5)
 JSR PC, WAIT
 MOV (SP)+, R5
 MOV R5, SP
 MOV (SP)+, R5
 LET TIMSAV := TIMSAV - #1 ; COUNTING DOWN
 DEC TIMSAV
 ; CONTINUED AT THE TOP
 BR \$50263
 \$50264:
 BIT #BIT00, FLAG
 BNE +6
 JMP \$50272
 TST TIMSAV
 BGT +6
 JMP \$50272
 JMP \$50261
 \$50272:
 SEC
 JMP \$50262

4347 016512 000000
4348 016514 000000
4349 016516 000
4350 016517 000

REGSAV: .WORD 0
TIMSAV: .WORD 0
FLAG: .BYTE 0
HOLDSC: .BYTE 0

; WE ARE DONE GO BACK HOME
ENDRTN

4351
4352 016520
4353 016520
4354 016520 000241
4355 016522
4356 016522 000207

\$50261:
CLC
\$50262:
RTS PC

4357
4358
4359
4360
4361
4362
4363
4364
4365
4366 016524
4367 016524
4368 016524
4369 016532
4370 016532 016501 000000
4371 016536
4372 016536 012702 000001
4373 016542 000402
4374 016544
4375 016544 062702 000001
4376 016550
4377 016550 020201
4378 016552 101402
4379 016554 000137 016604
4380 016560
4381 016560
4382 016560 005003
4383 016562 000401
4384 016564
4385 016564 005203
4386 016566
4387 016566 020327 000100
4388 016572 003402
4389 016574 000137 016602
4390 016600
4391 016600
4392 016600 000771
4393 016602
4394 016602
4395 016602 000760
4396 016604
4397 016604
4398 016612
4399 016612
4400 016612
4401 016612 000207

* ROUTINE:WAIT
* THIS ROUTINE IS USED TO DELAY EXECUTION OF THE
* MAIN PROGRAM FOR A SPECIFIED AMOUNT OF TIME.
* THIS IS ACCOMPLISHED BY INCREMENTING A
* REGISTER UP TO A LIMIT. THE INNER LOOP IS SET
* TO APPROXIMATE 1 MILLI SEC.

ROUTINE WAIT <TIME>

WAIT:

PUSH <R1,R2,R3>
LET R1 := TIME(R5)INCRU R2 FROM #1 TO R1 BY #1
MOV TIME(R5),R1MOV #1,R2
BR \$50276\$50275:
ADD #01,R2\$50276:
CMP R2,R1BLOS \$50277
JMP \$50300INCR R3 FROM #0 TO #100 BY #1
CLR R3

BR \$50302

\$50301:
INC R3\$50302:
CMP R3,#100BLE \$50303
JMP \$50304

\$50303:

ENDINC

BR \$50301
\$50304:

ENDINC

BR \$50275
\$50300:

ENDRTN POP <R3,R2,R1>

\$50273:
\$50274:

RTS PC

.SBTTL ROUTINE TO SIZE THE BUS

```

*****
THIS ROUTINE WILL LOOK AT THE BUS AND DETERMINE
WHAT DEVICES ARE THERE BY LOOKING AT $USWR BITS
10 AND 9.
*****
  
```

ROUTINE SIZE

SIZE:

4402							
4403							
4404							
4405							
4406							
4407							
4408							
4409							
4410							
4411							
4412							
4413	016614						
4414	016614						
4415							
4416	016614						
4417	016614	012737	017632	000004			
4418	016622						
4419	016622	012737	000340	000006			
4420	016630						
4421	016630	013737	001220	001300			
4422	016636						
4423	016636	006237	001300				
4424	016642	006237	001300				
4425	016646	006237	001300				
4426	016652	006237	001300				
4427	016656	006237	001300				
4428	016662	006237	001300				
4429	016666	006237	001300				
4430	016672	006237	001300				
4431	016676	006237	001300				
4432							
4433	016702						
4434	016702	013746	001300				
4435	016706	002002					
4436	016710	000137	017110				
4437	016714	023727	001300	000004			
4438	016722	003402					
4439	016724	000137	017110				
4440	016730	006316					
4441	016732	062716	016740				
4442	016736	013607					
4443	016740						
4444	016740	016752					
4445	016742	016770					
4446	016744	017006					
4447	016746	017024					
4448	016750	017024					
4449	016752						
4450	016752						
4451	016752						
4452	016752	012737	000001	001304			
4453	016760						
4454	016760	004737	017764				
4455	016764						
4456	016764	000137	017114				
4457	016770						

CASE 0

LET NODLV := #1

INLINE <JSR PC,ONKDF> ;NO DLV'S

CASE 1

```

;MY INTR SERVICE ROUTINE
MOV #INTSRV,ERRVEC
MOV #340,ERRVEC+2
;A TEMP STORE
MOV $USWR,P1CNT
;SHIFT OVER TO ZERO
ASR P1CNT
ASR P1CNT
ASR P1CNT
ASR P1CNT
ASR P1CNT
ASR P1CNT
ASR P1CNT
ASR P1CNT
ASR P1CNT
ASR P1CNT
MOV P1CNT,-(SP)
BGE .+6
JMP $50315
CMP P1CNT,#4
BLE .+6
JMP $50315
ASL (SP)
ADD #50307,(SP)
MOV @ (SP)+,PC
$50307:
.WORD $50314
.WORD $50313
.WORD $50312
.WORD $50311
.WORD $50310
$50314:
MOV #1,NODLV
JSR PC,ONKDF
JMP $50316
$50313:
  
```

4458	016770				LET KDDL V := #1		
4459	016770	012737	000001	001306			MOV #1,KDDL V
4460	016776				INLINE <JSR PC,KDDL VJ>	:ONE DLV	JSR PC,KDDL VJ
4461	016776	004737	020276		CASE 2		
4462	017002						JMP \$50316
4463	017002	000137	017114				\$50312:
4464	017006				LET DLV211 := #1		MOV #1,DLV211
4465	017006	012737	000001	001310			JSR PC,KD2DLV
4466	017014				INLINE <JSR PC,KD2DLV>	:TWO DLV'S	
4467	017014	004737	017642		DEFAULT		JMP \$50316
4468	017020						\$50311:
4469	017020	000137	017114				\$50310:
4470	017024				IFB \$AUTOB EQ #0 THEN		TSTB \$AUTOB
4471	017024						BEQ +6
4472	017024						JMP \$50317
4473	017024	105737	001134				
4474	017030	001402			TYPTXT <<CRLF>!CANNOT TEST THIS CONFIGURATION!>		
4475	017032	000137	017104		ENDIF		\$50317:
4476	017036				INLINE <BR .>		BR .
4477	017104						
4478	017104	000777			ENDSELECT		BR \$50316
4479	017104						\$50315:
4480	017106	000402					ADD #2,SP
4481	017110						\$50316:
4482	017110	062706	000002				
4483	017114						
4484					IF DLV211 EQ #1 THEN		CMP DLV211,#1
4485	017114	023727	001310	000001			BEQ +6
4486	017122	001402					JMP \$50320
4487	017124	000137	017230		IFB \$AUTOB EQ #0 THEN		
4488	017130						TSTB \$AUTOB
4489	017130	105737	001134				BEQ +6
4490	017134	001402			TYPTXT <<CRLF>!WILL TEST KDF11-B!>		JMP \$50321
4491	017136	000137	017230		TYPTXT <<CRLF>!CONSOL ADRS !>		
4492	017142				LET -(SP) := CONADR		
4493	017174				TYPOS		MOV CONADR,-(SP)
4494	017220	013746	001274		.BYTE 6		
4495	017224	104403			.BYTE 0		
4496	017226	006			ENDIF		\$50321:
4497	017227	000			ENDIF		\$50320:
4498	017230						
4499	017230				IF KDDL V EQ #1 THEN		CMP KDDL V,#1
4500	017230	023727	001306	000001			BEQ +6
4501	017236	001402			LET DLV211 := #0		JMP \$50322
4502	017240	000137	017316				
4503	017244						

4514	017244	005037	001310				CLR	DLV211
4515	017250				IFB \$AUTOB EQ #0 THEN		TSTB	\$AUTOB
4516	017250	105737	001134				BEQ	+6
4517	017254	001402					JMP	\$50323
4518	017256	000137	017316					
4519	017262				TYPTXT <<CRLF>!SLU 2 ADRS !>			
4520	017306				LET -(SP) := SECSLU		MOV	SECSLU, -(SP)
4521	017306	013746	001302					
4522	017312	104403			TYPOS			
4523	017314	006			.BYTE 6			
4524	017315	000			.BYTE 0			
4525	017316				ENDIF			
4526	017316						\$50323:	
4527	017316				ENDIF		\$50322:	
4528	017316							
4529								
4530	017316				IF NODLV EQ #1 THEN			
4531	017316	023727	001304	000001			CMP	NODLV, #1
4532	017324	001402					BEQ	+6
4533	017326	000137	017342				JMP	\$50324
4534	017332				LET DLV211 := #0			
4535	017332	005037	001310				CLR	DLV211
4536	017336				LET KDDL V := #0			
4537	017336	005037	001306				CLR	KDDL V
4538	017342				ENDIF			
4539	017342						\$50324:	
4540								
4541	017342				LET ERRVEC := #6	:RESTORE		
4542	017342	012737	000006	000004			MOV	#6, ERRVEC
4543	017350				LET ERRVEC+2 := #0	:VECTORS		
4544	017350	005037	000006				CLR	ERRVEC+2
4545					*****			
4546								
4547	017354				IF \$DEV M EQ #0 THEN	:ANY DLV'S		
4548	017354	005737	001252				TST	\$DEV M
4549	017360	001402					BEQ	+6
4550	017362	000137	017442				JMP	\$50325
4551	017366				IFB \$AUTOB EQ #0 THEN			
4552	017366	105737	001134				TSTB	\$AUTOB
4553	017372	001402					BEQ	+6
4554	017374	000137	017436				JMP	\$50326
4555	017400				TYPTXT <<CRLF>!THERE ARE NO DLV11'S !>			
4556	017436				ENDIF		\$50326:	
4557	017436							
4558	017436				ELSE		JMP	\$50327
4559	017436	000137	017630				\$50325:	
4560	017442							
4561	017442				DLBASE := \$BASE		MOV	\$BASE, DLBASE
4562	017442	013737	001250	001312				
4563	017450				LET MASK := MASK SET.BY #BIT00		BIS	#BIT00, MASK
4564	017450	052737	000001	001320				
4565	017456				WHILE DLBASE NE #176700 DO			
4566	017456						\$50330:	
4567	017456	023727	001312	176700			CMP	DLBASE, #176700
4568	017464	001002					BNE	+6
4569	017466	000137	017600				JMP	\$50331

```

4570 017472          IF MASK SETIN $DEVN THEN
4571 017472 033737 001320 001252
4572 017500 001002
4573 017502 000137 017564
4574 017506
4575 017506 105737 001134
4576 017512 001402
4577 017514 000137 017554
4578 017520          TYPTXT      <<CRLF>>!DLV11-J MODULE ADRS !>
4579 017554          ENDIF
4580 017554          LET -(SP) := DLBASE
4581 017554
4582 017554 013746 001312
4583 017560 104403
4584 017562 006
4585 017563 000
4586 017564          ENDIF
4587 017564
4588 017564          LET DLBASE := DLBASE + #10
4589 017564 062737 000010 001312
4590 017572          LET MASK := MASK SHIFT 1
4591 017572 006337 001320
4592 017576          ENDDO
4593 017576 000727
4594 017600
4595 017600          LET -(SP) := R0
4596 017600 010046
4597 017602          INCRU R0 FROM #0 TO #10000 BY #1
4598 017602 005000
4599 017604 000402
4600 017606
4601 017606 062700 000001
4602 017612
4603 017612 020027 010000
4604 017616 101402
4605 017620 000137 017626
4606 017624
4607 017624          ENDINC
4608 017624 000770
4609 017626
4610 017626          LET R0 := (SP)+
4611 017626 012600
4612 017630          ENDIF
4613 017630
4614
4615 017630          ENDRTN
4616 017630
4617 017630
4618 017630 000207
4619
4620
4621
4622          .SBTTL  INTSRV  INTERRUPT SERVICE ROUTINE
4623          :*****
4624          :*      THIS GLOBAL ROUTINE DOES NOTHING BUT INCREMENT
4625

```

```

BIT      MASK,$DEVN
BNE      .+6
JMP      $50332

TSTB     $AUTOB
BEQ      .+6
JMP      $50333

$50333:
MOV      DLBASE,-(SP)

$50332:
ADD      #10,DLBASE
ASL      MASK
BR        $50330
$50331:
;SAVE R0
MOV      R0,-(SP)

CLR      R0
BR        $50335
$50334:
ADD      #01,R0
$50335:
CMP      R0,#10000
BLOS     $50336
JMP      $50337
$50336:
;WAIT FOR TERMINAL TO STOP
BR        $50334
$50337:
;RESTORE R0
MOV      (SP)+,R0

$50327:

$50305:
$50306:
RTS      PC

```



```

4626      ;*      'INTFLG' EACH TIME IT IS CALLED. IT ASSUMES
4627      ;*      THAT THE MAIN CALLING ROUTINE WILL KNOW WHAT
4628      ;*      TO LOOK FOR.
4629      ;*****
4630      BGNSRV INTSRV      ;GLOBAL INTERRUPT SERVICE ROUTINE
4631      INTSRV:
4632      ;ADD 1 TO 'INTERRUPT OCCURED' FLAG
4633      ;          LET INTFLG := INTFLG + #1
4634      ;          INC      INTFLG
4635      ;THAT'S ALL
4636      ENDSRV
4637      RTI
4638      INTFLG: 0
4639      .SBTTL ROUTINE FOR KDF11 AND 2 DLV11'S
4640
4641      ;*****
4642      ;*      THIS ROUTINE WILL MAKE SURE THAT THE
4643      ;*      PROPER DEVICES ARE THERE.
4644      ;*****
4645
4646      ROUTINE KD2DLV      ;KDF11-B AS CONSOLE AND 2
4647      ;          KD2DLV:
4648      ;DLV11-J'S AT 176500
4649      LET SFTREG := #1
4650      LET TMP9 := $BASE      ;BASE ADRES      MOV      #1,SFTREG
4651      ;          MOV      $BASE,TMP9
4652      ;LOOK FOR DLV'S
4653      ;          JSR      PC,REPEAT
4654      ;          PC,REPEAT
4655      IF $DEVN NE #377 THEN
4656      ;          CMP      $DEVN,#377
4657      ;          BNE      .+6
4658      ;          JMP      $50342
4659      IFB $AUTOB EQ #0 THEN
4660      ;          TSTB     $AUTOB
4661      ;          BEQ      .+6
4662      ;          JMP      $50343
4663      TYPTXT      <<CRLF>>! DEVICE MAP DOES NOT AGREE!>
4664      ENDIF
4665      ;          $50343:
4666      ;          BR      .
4667      ;          $50342:
4668      ;          MOV      #177560,CONADR
4669      ;          $50340:
4670      ;          $50341:
4671      ;          RTS      PC
4672      ;          $50340:
4673      ;          $50341:
4674      ;          RTS      PC
4675      ;          $50340:
4676      ;          $50341:
4677      ;          RTS      PC
4678      ;          $50340:
4679      ;          $50341:
4680      ;          RTS      PC
4681      ;          $50340:
4682      ;          $50341:
4683      ;          RTS      PC
4684      ;          $50340:
4685      ;          $50341:
4686      ;          RTS      PC
4687      ;          $50340:
4688      ;          $50341:
4689      ;          RTS      PC
4690      ;          $50340:
4691      ;          $50341:
4692      ;          RTS      PC
4693      ;          $50340:
4694      ;          $50341:
4695      ;          RTS      PC
4696      ;          $50340:
4697      ;          $50341:
4698      ;          RTS      PC
4699      ;          $50340:
4700      ;          $50341:
4701      ;          RTS      PC
4702      ;          $50340:
4703      ;          $50341:
4704      ;          RTS      PC
4705      ;          $50340:
4706      ;          $50341:
4707      ;          RTS      PC
4708      ;          $50340:
4709      ;          $50341:
4710      ;          RTS      PC
4711      ;          $50340:
4712      ;          $50341:
4713      ;          RTS      PC
4714      ;          $50340:
4715      ;          $50341:
4716      ;          RTS      PC
4717      ;          $50340:
4718      ;          $50341:
4719      ;          RTS      PC
4720      ;          $50340:
4721      ;          $50341:
4722      ;          RTS      PC
4723      ;          $50340:
4724      ;          $50341:
4725      ;          RTS      PC
4726      ;          $50340:
4727      ;          $50341:
4728      ;          RTS      PC
4729      ;          $50340:
4730      ;          $50341:
4731      ;          RTS      PC
4732      ;          $50340:
4733      ;          $50341:
4734      ;          RTS      PC
4735      ;          $50340:
4736      ;          $50341:
4737      ;          RTS      PC
4738      ;          $50340:
4739      ;          $50341:
4740      ;          RTS      PC
4741      ;          $50340:
4742      ;          $50341:
4743      ;          RTS      PC
4744      ;          $50340:
4745      ;          $50341:
4746      ;          RTS      PC
4747      ;          $50340:
4748      ;          $50341:
4749      ;          RTS      PC
4750      ;          $50340:
4751      ;          $50341:
4752      ;          RTS      PC
4753      ;          $50340:
4754      ;          $50341:
4755      ;          RTS      PC
4756      ;          $50340:
4757      ;          $50341:
4758      ;          RTS      PC
4759      ;          $50340:
4760      ;          $50341:
4761      ;          RTS      PC
4762      ;          $50340:
4763      ;          $50341:
4764      ;          RTS      PC
4765      ;          $50340:
4766      ;          $50341:
4767      ;          RTS      PC
4768      ;          $50340:
4769      ;          $50341:
4770      ;          RTS      PC
4771      ;          $50340:
4772      ;          $50341:
4773      ;          RTS      PC
4774      ;          $50340:
4775      ;          $50341:
4776      ;          RTS      PC
4777      ;          $50340:
4778      ;          $50341:
4779      ;          RTS      PC
4780      ;          $50340:
4781      ;          $50341:
4782      ;          RTS      PC
4783      ;          $50340:
4784      ;          $50341:
4785      ;          RTS      PC
4786      ;          $50340:
4787      ;          $50341:
4788      ;          RTS      PC
4789      ;          $50340:
4790      ;          $50341:
4791      ;          RTS      PC
4792      ;          $50340:
4793      ;          $50341:
4794      ;          RTS      PC
4795      ;          $50340:
4796      ;          $50341:
4797      ;          RTS      PC
4798      ;          $50340:
4799      ;          $50341:
4800      ;          RTS      PC
4801      ;          $50340:
4802      ;          $50341:
4803      ;          RTS      PC
4804      ;          $50340:
4805      ;          $50341:
4806      ;          RTS      PC
4807      ;          $50340:
4808      ;          $50341:
4809      ;          RTS      PC
4810      ;          $50340:
4811      ;          $50341:
4812      ;          RTS      PC
4813      ;          $50340:
4814      ;          $50341:
4815      ;          RTS      PC
4816      ;          $50340:
4817      ;          $50341:
4818      ;          RTS      PC
4819      ;          $50340:
4820      ;          $50341:
4821      ;          RTS      PC
4822      ;          $50340:
4823      ;          $50341:
4824      ;          RTS      PC
4825      ;          $50340:
4826      ;          $50341:
4827      ;          RTS      PC
4828      ;          $50340:
4829      ;          $50341:
4830      ;          RTS      PC
4831      ;          $50340:
4832      ;          $50341:
4833      ;          RTS      PC
4834      ;          $50340:
4835      ;          $50341:
4836      ;          RTS      PC
4837      ;          $50340:
4838      ;          $50341:
4839      ;          RTS      PC
4840      ;          $50340:
4841      ;          $50341:
4842      ;          RTS      PC
4843      ;          $50340:
4844      ;          $50341:
4845      ;          RTS      PC
4846      ;          $50340:
4847      ;          $50341:
4848      ;          RTS      PC
4849      ;          $50340:
4850      ;          $50341:
4851      ;          RTS      PC
4852      ;          $50340:
4853      ;          $50341:
4854      ;          RTS      PC
4855      ;          $50340:
4856      ;          $50341:
4857      ;          RTS      PC
4858      ;          $50340:
4859      ;          $50341:
4860      ;          RTS      PC
4861      ;          $50340:
4862      ;          $50341:
4863      ;          RTS      PC
4864      ;          $50340:
4865      ;          $50341:
4866      ;          RTS      PC
4867      ;          $50340:
4868      ;          $50341:
4869      ;          RTS      PC
4870      ;          $50340:
4871      ;          $50341:
4872      ;          RTS      PC
4873      ;          $50340:
4874      ;          $50341:
4875      ;          RTS      PC
4876      ;          $50340:
4877      ;          $50341:
4878      ;          RTS      PC
4879      ;          $50340:
4880      ;          $50341:
4881      ;          RTS      PC
4882      ;          $50340:
4883      ;          $50341:
4884      ;          RTS      PC
4885      ;          $50340:
4886      ;          $50341:
4887      ;          RTS      PC
4888      ;          $50340:
4889      ;          $50341:
4890      ;          RTS      PC
4891      ;          $50340:
4892      ;          $50341:
4893      ;          RTS      PC
4894      ;          $50340:
4895      ;          $50341:
4896      ;          RTS      PC
4897      ;          $50340:
4898      ;          $50341:
4899      ;          RTS      PC
4900      ;          $50340:
4901      ;          $50341:
4902      ;          RTS      PC
4903      ;          $50340:
4904      ;          $50341:
4905      ;          RTS      PC
4906      ;          $50340:
4907      ;          $50341:
4908      ;          RTS      PC
4909      ;          $50340:
4910      ;          $50341:
4911      ;          RTS      PC
4912      ;          $50340:
4913      ;          $50341:
4914      ;          RTS      PC
4915      ;          $50340:
4916      ;          $50341:
4917      ;          RTS      PC
4918      ;          $50340:
4919      ;          $50341:
4920      ;          RTS      PC
4921      ;          $50340:
4922      ;          $50341:
4923      ;          RTS      PC
4924      ;          $50340:
4925      ;          $50341:
4926      ;          RTS      PC
4927      ;          $50340:
4928      ;          $50341:
4929      ;          RTS      PC
4930      ;          $50340:
4931      ;          $50341:
4932      ;          RTS      PC
4933      ;          $50340:
4934      ;          $50341:
4935      ;          RTS      PC
4936      ;          $50340:
4937      ;          $50341:
4938      ;          RTS      PC
4939      ;          $50340:
4940      ;          $50341:
4941      ;          RTS      PC
4942      ;          $50340:
4943      ;          $50341:
4944      ;          RTS      PC
4945      ;          $50340:
4946      ;          $50341:
4947      ;          RTS      PC
4948      ;          $50340:
4949      ;          $50341:
4950      ;          RTS      PC
4951      ;          $50340:
4952      ;          $50341:
4953      ;          RTS      PC
4954      ;          $50340:
4955      ;          $50341:
4956      ;          RTS      PC
4957      ;          $50340:
4958      ;          $50341:
4959      ;          RTS      PC
4960      ;          $50340:
4961      ;          $50341:
4962      ;          RTS      PC
4963      ;          $50340:
4964      ;          $50341:
4965      ;          RTS      PC
4966      ;          $50340:
4967      ;          $50341:
4968      ;          RTS      PC
4969      ;          $50340:
4970      ;          $50341:
4971      ;          RTS      PC
4972      ;          $50340:
4973      ;          $50341:
4974      ;          RTS      PC
4975      ;          $50340:
4976      ;          $50341:
4977      ;          RTS      PC
4978      ;          $50340:
4979      ;          $50341:
4980      ;          RTS      PC
4981      ;          $50340:
4982      ;          $50341:
4983      ;          RTS      PC
4984      ;          $50340:
4985      ;          $50341:
4986      ;          RTS      PC
4987      ;          $50340:
4988      ;          $50341:
4989      ;          RTS      PC
4990      ;          $50340:
4991      ;          $50341:
4992      ;          RTS      PC
4993      ;          $50340:
4994      ;          $50341:
4995      ;          RTS      PC
4996      ;          $50340:
4997      ;          $50341:
4998      ;          RTS      PC
4999      ;          $50340:
5000      ;          $50341:
5001      ;          RTS      PC
5002      ;          $50340:
5003      ;          $50341:
5004      ;          RTS      PC
5005      ;          $50340:
5006      ;          $50341:
5007      ;          RTS      PC
5008      ;          $50340:
5009      ;          $50341:
5010      ;          RTS      PC
5011      ;          $50340:
5012      ;          $50341:
5013      ;          RTS      PC
5014      ;          $50340:
5015      ;          $50341:
5016      ;          RTS      PC
5017      ;          $50340:
5018      ;          $50341:
5019      ;          RTS      PC
5020      ;          $50340:
5021      ;          $50341:
5022      ;          RTS      PC
5023      ;          $50340:
5024      ;          $50341:
5025      ;          RTS      PC
5026      ;          $50340:
5027      ;          $50341:
5028      ;          RTS      PC
5029      ;          $50340:
5030      ;          $50341:
5031      ;          RTS      PC
5032      ;          $50340:
5033      ;          $50341:
5034      ;          RTS      PC
5035      ;          $50340:
5036      ;          $50341:
5037      ;          RTS      PC
5038      ;          $50340:
5039      ;          $50341:
5040      ;          RTS      PC
5041      ;          $50340:
5042      ;          $50341:
5043      ;          RTS      PC
5044      ;          $50340:
5045      ;          $50341:
5046      ;          RTS      PC
5047      ;          $50340:
5048      ;          $50341:
5049      ;          RTS      PC
5050      ;          $50340:
5051      ;          $50341:
5052      ;          RTS      PC
5053      ;          $50340:
5054      ;          $50341:
5055      ;          RTS      PC
5056      ;          $50340:
5057      ;          $50341:
5058      ;          RTS      PC
5059      ;          $50340:
5060      ;          $50341:
5061      ;          RTS      PC
5062      ;          $50340:
5063      ;          $50341:
5064      ;          RTS      PC
5065      ;          $50340:
5066      ;          $50341:
5067      ;          RTS      PC
5068      ;          $50340:
5069      ;          $50341:
5070      ;          RTS      PC
5071      ;          $50340:
5072      ;          $50341:
5073      ;          RTS      PC
5074      ;          $50340:
5075      ;          $50341:
5076      ;          RTS      PC
5077      ;          $50340:
5078      ;          $50341:
5079      ;          RTS      PC
5080      ;          $50340:
5081      ;          $50341:
5082      ;          RTS      PC
5083      ;          $50340:
5084      ;          $50341:
5085      ;          RTS      PC
5086      ;          $50340:
5087      ;          $50341:
5088      ;          RTS      PC
5089      ;          $50340:
5090      ;          $50341:
5091      ;          RTS      PC
5092      ;          $50340:
5093      ;          $50341:
5094      ;          RTS      PC
5095      ;          $50340:
5096      ;          $50341:
5097      ;          RTS      PC
5098      ;          $50340:
5099      ;          $50341:
5100      ;          RTS      PC
5101      ;          $50340:
5102      ;          $50341:
5103      ;          RTS      PC
5104      ;          $50340:
5105      ;          $50341:
5106      ;          RTS      PC
5107      ;          $50340:
5108      ;          $50341:
5109      ;          RTS      PC
5110      ;          $50340:
5111      ;          $50341:
5112      ;          RTS      PC
5113      ;          $50340:
5114      ;          $50341:
5115      ;          RTS      PC
5116      ;          $50340:
5117      ;          $50341:
5118      ;          RTS      PC
5119      ;          $50340:
5120      ;          $50341:
5121      ;          RTS      PC
5122      ;          $50340:
5123      ;          $50341:
5124      ;          RTS      PC
5125      ;          $50340:
5126      ;          $50341:
5127      ;          RTS      PC
5128      ;          $50340:
5129      ;          $50341:
5130      ;          RTS      PC
5131      ;          $50340:
5132      ;          $50341:
5133      ;          RTS      PC
5134      ;          $50340:
5135      ;          $50341:
5136      ;          RTS      PC
5137      ;          $50340:
5138      ;          $50341:
5139      ;          RTS      PC
5140      ;          $50340:
5141      ;          $50341:
5142      ;          RTS      PC
5143      ;          $50340:
5144      ;          $50341:
5145      ;          RTS      PC
5146      ;          $50340:
5147      ;          $50341:
5148      ;          RTS      PC
5149      ;          $50340:
5150      ;          $50341:
5151      ;          RTS      PC
5152      ;          $50340:
5153      ;          $50341:
5154      ;          RTS      PC
5155      ;          $50340:
5156      ;          $50341:
5157      ;          RTS      PC
5158      ;          $50340:
5159      ;          $50341:
5160      ;          RTS      PC
5161      ;          $50340:
5162      ;          $50341:
5163      ;          RTS      PC
5164      ;          $50340:
5165      ;          $50341:
5166      ;          RTS      PC
5167      ;          $50340:
5168      ;          $50341:
5169      ;          RTS      PC
5170      ;          $50340:
5171      ;          $50341:
5172      ;          RTS      PC
5173      ;          $50340:
5174      ;          $50341:
5175      ;          RTS      PC
5176      ;          $50340:
5177      ;          $50341:
5178      ;          RTS      PC
5179      ;          $50340:
5180      ;          $50341:
5181      ;          RTS      PC
5182      ;          $50340:
5183      ;          $50341:
5184      ;          RTS      PC
5185      ;          $50340:
5186      ;          $50341:
5187      ;          RTS      PC
5188      ;          $50340:
5189      ;          $50341:
5190      ;          RTS      PC
5191      ;          $50340:
5192      ;          $50341:
5193      ;          RTS      PC
5194      ;          $50340:
5195      ;          $50341:
5196      ;          RTS      PC
5197      ;          $50340:
5198      ;          $50341:
5199      ;          RTS      PC
5200      ;          $50340:
5201      ;          $50341:
5202      ;          RTS      PC
5203      ;          $50340:
5204      ;          $50341:
5205      ;          RTS      PC
5206      ;          $50340:
5207      ;          $50341:
5208      ;          RTS      PC
5209      ;          $50340:
5210      ;          $50341:
5211      ;          RTS      PC
5212      ;          $50340:
5213      ;          $50341:
5214      ;          RTS      PC
5215      ;          $50340:
5216      ;          $50341:
5217      ;          RTS      PC
5218      ;          $50340:
5219      ;          $50341:
5220      ;          RTS      PC
5221      ;          $50340:
5222      ;          $50341:
5223      ;          RTS      PC
5224      ;          $50340:
5225      ;          $50341:
5226      ;          RTS      PC
5227      ;          $50340:
5228      ;          $50341:
5229      ;          RTS      PC
5230      ;          $50340:
5231      ;          $50341:
5232      ;          RTS      PC
5233      ;          $50340:
5234      ;          $50341:
5235      ;          RTS      PC
5236      ;          $50340:
5237      ;          $50341:
5238      ;          RTS      PC
5239      ;          $50340:
5240      ;          $50341:
5241      ;          RTS      PC
5242      ;          $50340:
5243      ;          $50341:
5244      ;          RTS      PC
5245      ;          $50340:
5246      ;          $50341:
5247      ;          RTS      PC
5248      ;          $50340:
5249      ;          $50341:
5250      ;          RTS      PC
5251      ;          $50340:
5252      ;          $50341:
5253      ;          RTS      PC
5254      ;          $50340:
5255      ;          $50341:
5256      ;          RTS      PC
5257      ;          $50340:
5258      ;          $50341:
5259      ;          RTS      PC
5260      ;          $50340:
5261      ;          $50341:
5262      ;          RTS      PC
5263      ;          $50340:
5264      ;          $50341:
5265      ;          RTS      PC
5266      ;          $50340:
5267      ;          $50341:
5268      ;          RTS      PC
5269      ;          $50340:
5270      ;          $50341:
5271      ;          RTS      PC
5272      ;          $50340:
5273      ;          $50341:
5274      ;          RTS      PC
5275      ;          $50340:
5276      ;          $50341:
5277      ;          RTS      PC
5278      ;          $50340:
5279      ;          $50341:
5280      ;          RTS      PC
5281      ;          $50340:
5282      ;          $50341:
5283      ;          RTS      PC
5284      ;          $50340:
5285      ;          $50341:
5286      ;          RTS      PC
5287      ;          $50340:
5288      ;          $50341:
5289      ;          RTS      PC
5290      ;          $50340:
5291      ;          $50341:
5292      ;          RTS      PC
5293      ;          $50340:
5294      ;          $50341:
5295      ;          RTS      PC
5296      ;          $50340:
5297      ;          $50341:
5298      ;          RTS      PC
5299      ;          $50340:
5300      ;          $50341:
5301      ;          RTS      PC
5302      ;          $50340:
5303      ;          $50341:
5304      ;          RTS      PC
5305      ;          $50340:
5306      ;          $50341:
5307      ;          RTS      PC
5308      ;          $50340:
5309      ;          $50341:
5310      ;          RTS      PC
5311      ;          $50340:
5312      ;          $50341:
5313      ;          RTS      PC
5314      ;          $50340:
5315      ;          $50341:
5316      ;          RTS      PC
5317      ;          $50340:
5318      ;          $50341:
5319      ;          RTS      PC
5320      ;          $50340:
5321      ;          $50341:
5322      ;          RTS      PC
5323      ;          $50340:
5324      ;          $50341:
5325      ;          RTS      PC
5326      ;          $50340:
5327      ;          $50341:
5328      ;          RTS      PC
5329      ;          $50340:
5330      ;          $50341:
5331      ;          RTS      PC
5332      ;          $50340:
5333      ;          $50341:
5334      ;          RTS      PC
5335      ;          $50340:
5336      ;          $50341:
5337      ;          RTS      PC
5338      ;          $50340:
5339      ;          $50341:
5340      ;          RTS      PC
5341      ;          $50340:
5342      ;          $50341:
5343      ;          RTS      PC
5344      ;          $50340:
5345      ;          $50341:
5346      ;          RTS      PC
5347      ;          $50340:
5348      ;          $50341:
5349      ;          RTS      PC
5350      ;          $50340:
5351      ;          $50341:
5352      ;          RTS      PC
5353      ;          $50340:
5354      ;          $50341:
5355      ;          RTS      PC
5356      ;          $50340:
5357      ;          $50341:
5358      ;          RTS      PC
5359      ;          $50340:
5360      ;          $50341:
5361      ;          RTS      PC
5362      ;          $50340:
5363      ;          $50341:
5364      ;          RTS      PC
5365      ;          $50340:
5366      ;          $50341:
5367      ;          RTS      PC
5368      ;          $50340:
5369      ;          $50341:
5370      ;          RTS      PC
5371      ;          $50340:
5372      ;          $50341:
5373      ;          RTS      PC
5374      ;          $50340:
5375      ;          $50341:
5376      ;          RTS      PC
5377      ;          $50340:
5378      ;          $50341:
5379      ;          RTS      PC
5380      ;          $50340:
5381      ;          $50341:
5382      ;          RTS      PC
5383      ;          $50340:
5384      ;          $50341:
5385      ;          RTS      PC
5386      ;          $50340:
5387      ;          $50341:
5388      ;          RTS      PC
5389      ;          $50340:
5390      ;          $50341:
5391      ;          RTS      PC
5392      ;          $50340:
5393      ;          $50341:
5394      ;          RTS      PC
5395      ;          $50340:
5396      ;          $50341:
5397      ;          RTS      PC
5398      ;          $50340:
5399      ;          $50341:
5400      ;          RTS      PC
5401      ;          $50340:
5402      ;          $50341:
5403      ;          RTS      PC
5404      ;          $50340:
5405      ;          $50341:
5406      ;          RTS      PC
5407      ;          $50340:
5408      ;          $50341:
5409      ;          RTS      PC
5410      ;          $50340:
5411      ;          $50341:
5412      ;          RTS      PC
5413      ;          $50340:
5414      ;          $50341:
5415      ;          RTS      PC
5416      ;          $50340:
5417      ;          $50341:
5418      ;          RTS      PC
5419      ;          $50340:
5420      ;          $50341:
5421      ;          RTS      PC
5422      ;          $50340:
5423      ;          $50341:
5424      ;          RTS      PC
5425      ;          $50340:
5426      ;          $50341:
5427      ;          RTS      PC
5428      ;          $50340:
5429      ;          $50341:
5430      ;          RTS      PC
5431      ;          $50340:
5432      ;          $50341:
5433      ;          RTS      PC
5434      ;          $50340:
5435      ;          $50341:
5436      ;          RTS      PC
5437      ;          $50340:
5438      ;          $50341:
5439      ;          RTS      PC
5440      ;          $50340:
5441      ;          $50341:
5442      ;          RTS      PC
5443      ;          $50340:
5444      ;          $50341:
5445      ;          RTS      PC
5446      ;          $50340:
5447      ;          $50341:
5448      ;          RTS      PC
5449      ;          $50340:
5450      ;          $50341:
5451      ;          RTS      PC
5452      ;          $50340:
5453      ;          $50341:
5454      ;          RTS      PC
5455      ;          $50340:
5456      ;          $50341:
5457      ;          RTS      PC
5458      ;          $50340:
5459      ;          $50341:
5460      ;          RTS      PC
5461      ;          $50340:
5462      ;          $50341:
5463      ;          RTS      PC
5464      ;          $50340:
5465      ;          $50341:
5466      ;          RTS      PC
5467      ;          $50340:
5468      ;          $50341:
5469      ;          RTS      PC
5470      ;          $50340:
5471      ;          $50341:
5472      ;          RTS      PC
5473      ;          $50340:
5474      ;          $50341:
5475      ;          RTS      PC
5476      ;          $50340:
5477      ;          $50341:
5478      ;          RTS      PC
5479      ;          $50340:
5480      ;          $50341:
5481      ;          RTS      PC
5482      ;          $50340:
5483      ;          $50341:
5484      ;          RTS      PC
5485      ;          $50340:
5486      ;          $50341:
5487      ;          RTS      PC
5488      ;          $50340:
5489      ;          $50341:
5490      ;          RTS      PC
5491      ;          $50340:
5492      ;          $50341:
5493      ;          RTS      PC
5494      ;          $50340:
5495      ;          $50341:
5496      ;          RTS      PC
5497      ;          $50340:
5498      ;          $50341:
5499      ;          RTS      PC
5500      ;          $50340:
5501      ;          $50341:
5502      ;          RTS      PC
5503      ;          $50340:
5504      ;          $50341:
5505      ;          RTS      PC
5506      ;          $50340:
5507      ;          $50341:
5508      ;          RTS      PC
5509      ;          $50340:
5510      ;          $50341:
5511      ;          RTS      PC
5512      ;          $50340:
5513      ;          $50341:
5514
```

```
4682 *****
4683 * THIS ROUTINE WILL CHECK THE PROPER ADDRESSES
4684 * FOR THE 2 SLU'S ON THE KDF11-B. NO DLV11-J'S
4685 *****
4686
4687 017764 ROUTINE ONKDF
4688 017764
4689
4690 017764 ;CONSOLE ADRS
4691 017764 012737 177560 001326 LET TMP9 := #177560
4692 017772 013737 001326 001274 LET CONADR := TMP9
4693 017772 013737 001326 001274 LET INTFLG := #0
4694 020000 005037 017640 CLR INTFLG
4695 020000 005037 017640 LET R5 := @TMP9
4696 020004 017705 161316 MOV @TMP9,R5
4697 020004 017705 161316 IF INTFLG NE #0 THEN
4698 020010
4699 020010 005737 017640 TST INTFLG
4700 020014 001002 BNE +6
4701 020016 000137 020062 JMP $50346
4702 020022 IFB $AUTOB EQ #0 THEN
4703 020022 105737 001134 TSTB $AUTOB
4704 020026 001402 BEQ +6
4705 020030 000137 020060 JMP $50347
4706 020034
4707 020060 TYPTXT <<CRLF>& NO CONSOLE &
4708 020060 ENDIF
4709 020060
4710 020060 000777 INLINE <BR .>
4711 020062 ENDIF
4712 020062
4713 020062 LET TMP9 := $BASE ;176500
4714 020062 013737 001250 001326 LET INTFLG := #0
4715 020070 005037 017640 LET R5 := @TMP9
4716 020070 005037 017640 IF INTFLG EQ #0 THEN
4717 020074 017705 161226
4718 020074 017705 161226
4719 020100
4720 020100 005737 017640 TST INTFLG
4721 020104 001402 BEQ +6
4722 020106 000137 020124 JMP $50350
4723 020112 LET SECSLU := $BASE ;NO INTR, THE DEVICE IS THERE
4724 020112 013737 001250 001302 MOV $BASE,SECSLU
4725 020120 ELSE
4726 020120 000137 020200 JMP $50351
4727 020124 $50350:
4728 020124 IFB $AUTOB EQ #0 THEN
4729 020124 105737 001134 TSTB $AUTOB
4730 020130 001402 BEQ +6
4731 020132 000137 020176 JMP $50352
4732 020136 TYPTXT <<CRLF>& SECOND SLU WRONG ADRS&
4733 020176 ENDIF
4734 020176
4735 020176 000777 INLINE <BR .>
4736 020176 000777
4737 020200 ENDIF
```



```

4738 020200
4739 020200
4740 020200 005037 001252
4741 020204
4742 020204 012737 000001 001310
4743 020212
4744 020212 012737 000001 001306
4745
4746 020220
4747 020220
4748 020220
4749 020220 000207
4750
4751 .SBTTL ROUTINE TO FIND DLV11-J'S
4752
4753 ::*****
4754 ::* THIS ROUTINE WILL LOOK FOR THE DLV11-J BEING
4755 ::* IN THE PROPER ADRS LOCATIONS.
4756 ::*****
4757
4758 020222 ROUTINE REPEAT
4759 020222
4760
4761 020222 REPEAT
4762 020222
4763 020222 LET INTFLG := #0
4764 020222 005037 017640 CLR INTFLG
4765 020226 LET R5 := @TMP9
4766 020226 017705 161074 MOV @TMP9,R5
4767 020232 IF INTFLG EQ #0 THEN
4768 020232 005737 017640 TST INTFLG
4769 020236 001402 BEQ +6
4770 020240 000137 020252 JMP $50356
4771 020244 LET $DEV := $DEV SET.BY SFTREG ;SET BIT IN DEV
4772 020244 053737 001316 001252 BIS SFTREG,$DEV
4773 020252 ENDIF
4774 020252 LET SFTREG := SFTREG SHIFT 1 ;ELSE A ZERO
4775 020252 006337 001316 ASL SFTREG
4776 020256 LET TMP9 := TMP9 + #10
4777 020256 062737 000010 001326 ADD #10,TMP9
4778 020256 UNTIL TMP9 EQ #176600
4779 020264 023727 001326 176600 CMP TMP9,#176600
4780 020264 001353 BNE $50355
4781
4782
4783 020274 ENDRTN
4784 020274
4785 020274
4786 020274 000207
4787
4788 .SBTTL ROUTINE FOR KDF11 AND ONE DLV11-J
4789
4790 ::*****
4791 ::* THIS ROUTINE IS USED WHEN THE CONSOLE IS ON
4792 ::* THE KDF11-B AND WE HAVE ONE DLV11-J MODULE.
4793 ::*****

```

\$50351:

```

;SET THESE FOR PROPER
;PRINT OUTS
CLR $DEV
MOV #1,DLV211
MOV #1,KDDL

```

\$50344:

\$50345:

RTS PC

REPEAT:

\$50355:

```

CLR INTFLG
MOV @TMP9,R5
TST INTFLG
BEQ +6
JMP $50356
;SET BIT IN DEV
BIS SFTREG,$DEV

```

\$50356:

```

;ELSE A ZERO
ASL SFTREG
ADD #10,TMP9
CMP TMP9,#176600
BNE $50355

```

\$50353:

\$50354:

RTS PC

4794					ROUTINE KDDLJV		
4795	020276					KDDLJV:	
4796	020276						
4797							
4798	020276				LET TMP9 := \$BASE		
4799	020276	013737	001250	001326		MOV	\$BASE,TMP9
4800	020304				LET TMP9 := TMP9 + #40	;176540	
4801	020304	062737	000040	001326		ADD	#40,TMP9
4802	020312				LET SECSLU := TMP9		
4803	020312	013737	001326	001302		MOV	TMP9,SECSLU
4804	020320				LET INTFLG := #0	;MAKE SURE CH1 IS THERE	
4805	020320	005037	017640			CLR	INTFLG
4806	020324				LET R5 := @TMP9		
4807	020324	017705	160776			MOV	@TMP9,R5
4808							
4809	020330				IF INTFLG NE #0 THEN	;IF 0 THEN NO INTERRUPT	
4810	020330	005737	017640			TST	INTFLG
4811	020334	001002				BNE	.+6
4812	020336	000137	020402			JMP	\$50361
4813	020342				IFB \$AUTOB EQ #0 THEN		
4814	020342	105737	001134			TSTB	\$AUTOB
4815	020346	001402				BEQ	.+6
4816	020350	000137	020400			JMP	\$50362
4817	020354				TYPTXT <<CRLF>& NO KDF11-B&		
4818	020400				ENDIF		
4819	020400					\$50362:	
4820	020400				INLINE <BR .>		
4821	020400	000777				BR	
4822	020402				ENDIF	;SHOULD BE SOMETHING THERE	
4823	020402					\$50361:	
4824							
4825	020402				LET SFTREG := #1		
4826	020402	012737	000001	001316		MOV	#1,SFTREG
4827	020410				LET TMP9 := \$BASE	;SET UP TMP9 TO 176500	
4828	020410	013737	001250	001326		MOV	\$BASE,TMP9
4829	020416				INLINE <JSR PC,REPEAT>	;LOOK FOR OTHERS	
4830	020416	004737	020222			JSR	PC,REPEAT
4831							
4832	020422				LET \$DEVM := \$DEVM CLR.BY #BIT04		
4833	020422	042737	000020	001252		BIC	#BIT04,\$DEVM
4834	020430				IF \$DEVM NE #17 THEN	;SDEVM MUST BE 17	
4835	020430	023727	001252	000017		CMP	\$DEVM,#17
4836	020436	001002				BNE	.+6
4837	020440	000137	020522			JMP	\$50363
4838	020444				IFB \$AUTOB EQ #0 THEN		
4839	020444	105737	001134			TSTB	\$AUTOB
4840	020450	001402				BEQ	.+6
4841	020452	000137	020520			JMP	\$50364
4842	020456				TYPTXT <<CRLF>!DEVICE MAP DOES NOT AGREE!>		
4843	020520				ENDIF		
4844	020520					\$50364:	
4845	020520				INLINE <BR .>		
4846	020520	000777				BR	.
4847	020522				ENDIF		
4848	020522					\$50363:	
4849							

4850	020522				LET CONADR := #177560	
4851	020522	012737	177560	001274		
4852	020530				LET DLV211 := #1	
4853	020530	012737	000001	001310		
4854						
4855	020536				ENDRTN	
4856	020536					
4857	020536					
4858	020536	000207				
4859						
4860						
4861						
4862						
4863						
4864						
4865						
4866						
4867	020540					
4868	020540					
4869						
4870	020540					
4871	020540	005737	001256			
4872	020544	001402				
4873	020546	000137	020606			
4874	020552					
4875	020552	012737	000001	001324		
4876	020560					
4877	020560	012737	000060	001260		
4878	020566					
4879	020566	013737	001144	001256		
4880	020574					
4881	020574	012737	000001	001272		
4882	020602					
4883	020602	000137	020740			
4884	020606					
4885	020606					
4886	020606	005037	001324			
4887	020612					
4888	020612	023737	001256	001274		
4889	020620	001402				
4890	020622	000137	020732			
4891	020626					
4892	020626	013737	001244	001260		
4893	020634					
4894	020634	023727	001300	000002		
4895	020642	001402				
4896	020644	000137	020670			
4897	020650					
4898	020650	013737	001250	001256		
4899	020656					
4900	020656	012737	000001	001314		
4901	020664					
4902	020664	000137	020676			
4903	020670					
4904	020670					
4905	020670	013737	001302	001256		

```

;SET FOR PROPER PRINTING
MOV #177560,CONADR
MOV #1,DLV211

$50357:
$50360:
RTS PC

.SBTTL ROUTINE TO SET UP ADDRESS FOR TESTING

*****
* THIS ROUTINE FINDS THE NEXT CORRECT
* ADDRESS AND VECTOR OF A DEVICE TO BE TESTED.
*****

ROUTINE CYCLE

IF DLADD EQ #0 THEN

    LET CONSOL := #1
    LET DLVEC := #TKVEC
    LET DLADD := $TKS
    LET CONTST := #1
ELSE
    LET CONSOL := #0
    IF DLADD EQ CONADR THEN

        LET DLVEC := $VECT1
        IF P1CNT EQ #2 THEN

            LET DLADD := $BASE
            LET KDFTST := #1
        ELSE
            LET DLADD := SECSLU
    ENDIF
ENDIF

CYCLE:

TST DLADD
BEQ +6
JMP $50367
;CONSOLE TESTING BIT
MOV #1,CONSOL
;BEFORE TESTING SET IT
MOV #TKVEC,DLVEC
;DLADD EQ CONSOLE
MOV $TKS,DLADD
;TESTED THE CONSOLE
MOV #1,CONTST

JMP $50370
$50367:
;CLR CONSOLE TESTING BIT
CLR CONSOL

CMP DLADD,CONADR
BEQ +6
JMP $50371
;DLVEC EQ 300
MOV $VECT1,DLVEC
;SECSLU IS DESELECTED
CMP P1CNT,#2
BEQ +6
JMP $50372
;176500
MOV $BASE,DLADD
MOV #1,KDFTST

JMP $50373
$50372:
MOV SECSLU,DLADD

```

4906 020676
 4907 020676
 4908 020676
 4909 020676 023727 001302 176540
 4910 020704 001402
 4911 020706 000137 020720
 4912 020712
 4913 020712 062737 000040 001260
 4914 020720
 4915 020720
 4916 020720
 4917 020720 012737 000001 001322
 4918 020726
 4919 020726 000137 020740
 4920 020732
 4921 020732
 4922 020732 012737 000001 001314
 4923 020740
 4924 020740
 4925
 4926 020740
 4927 020740
 4928
 4929 020740
 4930 020740 023727 001272 000001
 4931 020746 001402
 4932 020750 000137 021322
 4933 020754 023727 001314 000001
 4934 020762 001402
 4935 020764 000137 021322
 4936 020770
 4937 020770 005737 001252
 4938 020774 001402
 4939 020776 000137 021004
 4940 021002
 4941 021002 000550
 4942 021004
 4943 021004
 4944 021004
 4945 021004 023727 001302 176500
 4946 021012 001402
 4947 021014 000137 021140
 4948 021020
 4949 021020 005737 001320
 4950 021024 001402
 4951 021026 000137 021074
 4952 021032
 4953 021032 052737 000020 001320
 4954 021040
 4955 021040 013737 001244 001260
 4956 021046 062737 000040 001260
 4957 021054
 4958 021054 013737 001302 001256
 4959 021062 062737 000040 001256
 4960 021070
 4961 021070 000137 021134

ENDIF
 IF SECSLU EQ #176540 THEN
 LET DLVEC := DLVEC + #40
 ENDIF
 LET KDF11B := #1
 ELSE
 LET KDFTST := #1
 ENDIF
 ;THIS BIT IS
 ;SET AFTER TESTING
 ENDIF
 IF CONTST EQ #1 AND KDFTST EQ #1 THEN
 IF \$DEVN EQ #0 THEN
 INLINE <BR ENDDER>
 ENDIF
 IF SECSLU EQ #176500 THEN
 IF MASK EQ #0 THEN
 LET MASK := MASK SET.BY #BIT04
 LET DLVEC := \$VECT1 + #40
 LET DLADD := SECSLU + #40
 ELSE

\$50373:
 CMP SECSLU,#176540
 BEQ .+6
 JMP \$50374
 ADD #40,DLVEC
 \$50374:
 ;SECOND SLU BIT
 MOV #1,KDF11B
 JMP \$50375
 \$50371:
 MOV #1,KDFTST
 \$50375:
 \$50370:
 ;BOTH SLU'S ON KDF11B
 CMP CONTST,#1
 BEQ .+6
 JMP \$50376
 CMP KDFTST,#1
 BEQ .+6
 JMP \$50376
 ;HAVE BEEN TESTED
 TST \$DEVN
 BEQ .+6
 JMP \$50377
 BR ENDDER
 \$50377:
 CMP SECSLU,#176500
 BEQ .+6
 JMP \$50400
 TST MASK
 BEQ .+6
 JMP \$50401
 BIS #BIT04,MASK
 MOV \$VECT1,DLVEC
 ADD #40,DLVEC
 MOV SECSLU,DLADD
 ADD #40,DLADD
 JMP \$50402

4962	021074			
4963	021074			
4964	021074	006337	001320	
4965	021100			
4966	021100	062737	000010	001260
4967	021106			
4968	021106	062737	000010	001256
4969	021114			
4970	021114	033737	001320	001252
4971	021122	001402		
4972	021124	000137	021134	
4973	021130			
4974	021130	000137	010102	
4975	021134			
4976	021134			
4977	021134			
4978	021134			
4979	021134			
4980	021134	000137	021322	
4981	021140			
4982	021140			
4983	021140	005737	001320	
4984	021144	001402		
4985	021146	000137	021200	
4986	021152			
4987	021152	052737	000001	001320
4988	021160			
4989	021160	013737	001244	001260
4990	021166			
4991	021166	013737	001250	001256
4992	021174			
4993	021174	000137	021322	
4994	021200			
4995	021200			
4996	021200	006337	001320	
4997	021204			
4998	021204	062737	000010	001260
4999	021212			
5000	021212	062737	000010	001256
5001	021220			
5002	021220	023727	001320	000020
5003	021226	001402		
5004	021230	000137	021302	
5005	021234	005737	001302	
5006	021240	001002		
5007	021242	000137	021302	
5008	021246			
5009	021246	006337	001320	
5010	021252	006337	001320	
5011	021256	006337	001320	
5012	021262	006337	001320	
5013	021266			
5014	021266	062737	000040	001260
5015	021274			
5016	021274	062737	000040	001256
5017	021302			

```

      LET MASK := MASK SHIFT 1
      LET DLVEC := DLVEC + #10
      LET DLADD := DLADD + #10
      IF MASK NOTSETIN $DEVN THEN
        INLINE <JMP      TEST20>
      ENDIF
    ENDIF
  ELSE
    IF MASK EQ #0 THEN
      LET MASK := MASK SET.BY #BIT00
      LET DLVEC := $VECT1
      LET DLADD := $BASE
    ELSE
      LET MASK := MASK SHIFT 1
      LET DLVEC := DLVEC + #10
      LET DLADD := DLADD + #10
      IF MASK EQ #20 AND SECSLU NE #0 THEN
        LET MASK := MASK SHIFT 4
        LET DLVEC := DLVEC + #40
        LET DLADD := DLADD + #40
      ENDIF
    ENDIF
  ENDIF

```

```

$50401:
;NEXT DEVN AND ADDRESS
ASL      MASK
ADD      #10,DLVEC
ADD      #10,DLADD
BIT      MASK,$DEVN
BEQ      .+6
JMP      $50403
JMP      TEST20

$50403:
$50402:
176540 OR ZERO
JMP      $50404
$50400:
TST      MASK
BEQ      .+6
JMP      $50405
BIS      #BIT00,MASK
;START AT 300
MOV      $VECT1,DLVEC
;START AT 176500
MOV      $BASE,DLADD
JMP      $50406
$50405:
ASL      MASK
ADD      #10,DLVEC
;NXT DEVN AND ADDRESS
ADD      #10,DLADD
CMP      MASK,#20
BEQ      .+6
JMP      $50407
TST      SECSLU
BNE      .+6
JMP      $50407
;BYPASS CONSOLE
ASL      MASK
ASL      MASK
ASL      MASK
ASL      MASK
ADD      #40,DLVEC
ADD      #40,DLADD

```

```

5018 021302
5019 021302
5020 021302 033737 001320 001252
5021 021310 001402
5022 021312 000137 021322
5023 021316
5024 021316 000137 010102
5025 021322
5026 021322
5027 021322
5028 021322
5029 021322
5030 021322
5031 021322
5032 021322
5033
5034 021322
5035 021322
5036 021322
5037 021322 000207
5038
5039
5040
5041
5042
5043 021324
5044 021324
5045
5046 021324
5047 021324 005037 001314
5048 021330
5049 021330 005037 001322
5050 021334
5051 021334 013737 001274 001256
5052 021342
5053 021342 032777 010000 157570
5054 021350 001002
5055 021352 000137 021416
5056 021356
5057 021356 105737 001134
5058 021362 001402
5059 021364 000137 021416
5060 021370
5061 021410
5062 021416
5063 021416
5064 021416
5065 021416
5066
5067 021416
5068 021416 005037 001320
5069 021422
5070 021422 005037 001112
5071 021426
5072 021426 000476
5073

      IF MASK NOTSETIN $DEVN THEN
      ;ANY MORE DEVICES ?
      BIT    MASK,$DEVN
      BEQ    .+6
      JMP    $50410
      JMP    TEST20
      JMP    TEST20
      $50410:
      $50406:
      $50404:
      $50376:
      $50365:
      $50366:
      RTS    PC

      *****
      :      THIS IS THE ROUTINE CALLED ENDDER FROM ABOVE
      :      *****
      ROUTINE ENDDER
      ENDDER:
      LET KDFTST := #0
      CLR    KDFTST
      LET KDF11B := #0
      CLR    KDF11B
      LET DLADD := CONADR
      MOV    CONADR,DLADD
      IF #BIT12 SETIN @SWR THEN
      BIT    #BIT12,@SWR
      BNE    .+6
      JMP    $50413
      IFB $AUTOB EQ #0 THEN
      TSTB    $AUTOB
      BEQ    .+6
      JMP    $50414
      TYPTXT    <<CRLF>*ERRORS: *>
      TYPDEC    $ERTTL
      ENDDER
      ENDDER
      LET MASK := #0
      CLR    MASK
      LET $ERTTL := #0
      CLR    $ERTTL
      INLINE <BR    $EOP>
      ;SET UP FOR MORE PASSES
      BR    $EOP
      ;GO TO END OF PASS ROUTINE
  
```



```
5074  
5075  
5076  
5077 021430      INLINE <HALT>  
5078 021430 000000  
5079 021432      ENDRTN  
5080 021432  
5081 021432  
5082 021432 000207
```

;AND THEN TO LOOP. WE WILL ONLY
;TEST THE CONSOLE ON THE FIRST PASS

HALT
\$50411:
\$50412:
RTS PC

					ROUTINE MYTYPE	MYTYPE:
5083						
5084	021434					
5085	021434					
5086	021434					
5087	021454					
5088	021462					
5089	021502	113737	001114	001176		
5090	021510					
5091	021520					
5092	021536					
5093	021544					
5094	021562					
5095	021570					
5096	021610					
5097	021616	104401	001171			
5098	021622					
5099	021622					
5100	021622					
5101	021622	000207				
5102						

TYPTXT	<<CRLF>*TEST # *>	
TYPOCT	\$TESTN	
TYPTXT	<*,ERROR # *>	
MOVB	\$ITEMB,\$FATAL	; APT FATAL ERROR NUMBER
TYPOCS	\$FATAL	
TYPTXT	<*,PC = *>	
TYPOCT	\$ERRPC	
TYPTXT	<*,CSR: *>	
TYPOCT	DLADD	
TYPTXT	<*,VECTOR: *>	
TYPOCT	DLVEC	
TYPE	,\$CRLF	
	ENDRTN	
		\$50415:
		\$50416:
		RTS PC

.SBTTL END OF PASS ROUTINE

 *INCREMENT THE PASS NUMBER (\$PASS)
 *INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
 *TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
 *IF THERES A MONITOR GO TO IT
 *IF THERE ISN'T JUMP TO LOOP

\$EOP:

SCOPE			
CLR	\$TSTNM		::ZERO THE TEST NUMBER
CLR	\$TIMES		::ZERO THE NUMBER OF ITERATIONS
INC	\$PASS		::INCREMENT THE PASS NUMBER
BIC	#100000,\$PASS		::DON'T ALLOW A NEG. NUMBER
DEC	(PC)+		::LOOP?
\$EOPCT: .WORD	1		
BGT	\$DOAGN		::YES
MOV	(PC)+,@(PC)+		::RESTORE COUNTER
\$ENDCT: .WORD	1		
\$EOPCT			
TYPE	,\$ENDMG		::TYPE 'END PASS #'
MOV	\$PASS,-(SP)		::SAVE \$PASS FOR TYPEOUT
TYPDS			::GO TYPE--DECIMAL ASCII WITH SIGN
TYPE	,\$ENULL		::TYPE A NULL CHARACTER
\$GET42: MOV	@42,R0		::GET MONITOR ADDRESS
BEQ	\$DOAGN		::BRANCH IF NO MONITOR
RESET			::CLEAR THE WORLD
\$ENDAD: JSR	PC,(R0)		::GO TO MONITOR
NOP			::SAVE ROOM
NOP			::FOR
NOP			::ACT11
\$DOAGN:			
JMP	@(PC)+		::RETURN
\$RTNAD: .WORD	LOOP		
\$ENULL: .BYTE	-1,-1,0		::NULL CHARACTER STRING
\$ENDMG: .ASCIIZ	<15><12>/END PASS #/		

5103				
5104				
5105				
5106				
5107				
5108				
5109				
5110				
5111				
5112	021624			
5113	021624	000004		
5114	021626	005037	001102	
5115	021632	005037	001160	
5116	021636	005237	001202	
5117	021642	042737	100000	001202
5118	021650	005327		
5119	021652	000001		
5120	021654	003022		
5121	021656	012737		
5122	021660	000001		
5123	021662	021652		
5124	021664	104401	021731	
5125	021670	013746	001202	
5126	021674	104405		
5127	021676	104401	021726	
5128	021702	013700	000042	
5129	021706	001405		
5130	021710	000005		
5131	021712	004710		
5132	021714	000240		
5133	021716	000240		
5134	021720	000240		
5135	021722			
5136	021722	000137		
5137	021724	002000		
5138	021726	377	377	000
5139	021731	015	042412	042116
5140	021736	050040	051501	020123
5141	021744	000043		

.SBTTL TYPE ROUTINE

 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
 *NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
 *NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
 *NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.
 *

*CALL:
 *1) USING A TRAP INSTRUCTION
 * TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
 *OR
 * TYPE
 * MESADR
 *

5142										
5143										
5144										
5145										
5146										
5147										
5148										
5149										
5150										
5151										
5152										
5153										
5154										
5155										
5156										
5157										
5158										
5159	021746	105737	001157			\$TYPE:	TSTB	\$TPFLG		::IS THERE A TERMINAL?
5160	021752	100002					BPL	1\$		::BR IF YES
5161	021754	000000					HALT			::HALT HERE IF NO TERMINAL
5162	021756	000430					BR	3\$		::LEAVE
5163	021760	010046				1\$:	MOV	R0,-(SP)		::SAVE R0
5164	021762	017600	000002				MOV	@2(SP),R0		::GET ADDRESS OF ASCIZ STRING
5165	021766	122737	000001	001214			CMPB	#APTENV,\$ENV		::RUNNING IN APT MODE
5166	021774	001011					BNE	62\$		::NO,GO CHECK FOR APT CONSOLE
5167	021776	132737	000100	001215			BITB	#APTSPOOL,\$ENV		::SPOOL MESSAGE TO APT
5168	022004	001405					BEQ	62\$		::NO,GO CHECK FOR CONSOLE
5169	022006	010037	022016				MOV	R0,61\$		::SETUP MESSAGE ADDRESS FOR APT
5170	022012	004737	023074				JSR	PC,\$ATY3		::SPOOL MESSAGE TO APT
5171	022016	000000				61\$:	.WORD	0		::MESSAGE ADDRESS
5172	022020	132737	000040	001215		62\$:	BITB	#APTCSUP,\$ENV		::APT CONSOLE SUPPRESSED
5173	022026	001003					BNE	60\$		::YES,SKIP TYPE OUT
5174	022030	112046				2\$:	MOVB	(R0)+,-(SP)		::PUSH CHARACTER TO BE TYPED ONTO STACK
5175	022032	001005					BNE	4\$		::BR IF IT ISN'T THE TERMINATOR
5176	022034	005726					TST	(SP)+		::IF TERMINATOR POP IT OFF THE STACK
5177	022036	012600				60\$:	MOV	(SP)+,R0		::RESTORE R0
5178	022040	062716	000002			3\$:	ADD	#2,(SP)		::ADJUST RETURN PC
5179	022044	000002					RTI			::RETURN
5180	022046	122716	000011			4\$:	CMPB	#HT,(SP)		::BRANCH IF <HT>
5181	022052	001430					BEQ	8\$		
5182	022054	122716	000200				CMPB	#CRLF,(SP)		::BRANCH IF NOT <CRLF>
5183	022060	001006					BNE	5\$		
5184	022062	005726					TST	(SP)+		::POP <CR><LF> EQUIV
5185	022064	104401					TYPE			::TYPE A CR AND LF
5186	022066	001171					\$CRLF			
5187	022070	105037	022276				CLRB	\$CHARCNT		::CLEAR CHARACTER COUNT
5188	022074	000755					BR	2\$		::GET NEXT CHARACTER
5189	022076	004737	022160			5\$:	JSR	PC,\$TYPE		::GO TYPE THIS CHARACTER
5190	022102	123726	001156			6\$:	CMPB	\$FILLC,(SP)+		::IS IT TIME FOR FILLER CHARS.?
5191	022106	001350					BNE	2\$		::IF NO GO GET NEXT CHAR.
5192	022110	013746	001154				MOV	\$NULL,-(SP)		::GET # OF FILLER CHARS. NEEDED
5193										::AND THE NULL CHAR.
5194	022114	105366	000001			7\$:	DECB	1(SP)		::DOES A NULL NEED TO BE TYPED?
5195	022120	002770					BLT	6\$		::BR IF NO--GO POP THE NULL OFF OF STACK
5196	022122	004737	022160				JSR	PC,\$TYPE		::GO TYPE A NULL
5197	022126	105337	022276				DECB	\$CHARCNT		::DO NOT COUNT AS A COUNT


```

5198 022132 000770          BR      7$          ::LOOP
5199
5200          :HORIZONTAL TAB PROCESSOR
5201
5202 022134 112716 000040      8$:  MOVB  #' (SP)          ::REPLACE TAB WITH SPACE
5203 022140 004737 022160      9$:  JSR   PC,$TYPEC          ::TYPE A SPACE
5204 022144 132737 000007 022276  BITB  #7,$CHARCNT          ::BRANCH IF NOT AT
5205 022152 001372          BNE   9$          ::TAB STOP
5206 022154 005726          TST   (SP)+          ::POP SPACE OFF STACK
5207 022156 000724          BR     2$          ::GET NEXT CHARACTER
5208 022160
5209 022160 105777 156760      $TYPEC: TSTB  @STKS          ::CHAR IN KYBD BUFFER?
5210 022164 100022          BPL   10$          ::BR IF NOT
5211 022166 017746 156754          MOV  @STKB,-(SP)          ::GET CHAR
5212 022172 042716 177600          BIC  #177600,(SP)          ::STRIP EXTRANEIOUS BITS
5213 022176 122716 000023          CMPB #XOFF,(SP)          ::WAS CHAR XOFF
5214 022202 001012          BNE   102$          ::BR IF NOT
5215 022204
5216 022204 105777 156734      101$: TSTB  @STKS          ::WAIT FOR CHAR
5217 022210 100375          BPL   101$          ::BR IF NOT
5218 022212 117716 156730          MOVB @STKB,(SP)          ::GET CHAR
5219 022216 042716 177600          BIC  #177600,(SP)          ::STRIP IT
5220 022222 122716 000021          CMPB #XON,(SP)          ::WAS IT XON?
5221 022226 001366          BNE   101$          ::BR IF NOT
5222 022230
5223 022230 005726          102$: TST   (SP)+          ::FIX STACK
5224 022232
5225 022232 105777 156712      10$: TSTB  @STPS          ::WAIT UNTIL PRINTER IS READY
5226 022236 100375          BPL   10$          ::BR IF NOT
5227 022240 116677 000002 156704          MOVB 2(SP),@STPB          ::LOAD CHAR TO BE TYPED INTO DATA REG.
5228 022246 122766 000015 000002          CMPB #CR,2(SP)          ::IS CHARACTER A CARRIAGE RETURN?
5229 022254 001003          BNE   1$          ::BRANCH IF NO
5230 022256 105037 022276          CLRB  $CHARCNT          ::YES--CLEAR CHARACTER COUNT
5231 022262 000406          BR     $TYPEX          ::EXIT
5232 022264 122766 000012 000002 1$:  CMPB #LF,2(SP)          ::IS CHARACTER A LINE FEED?
5233 022272 001402          BEQ   $TYPEX          ::BRANCH IF YES
5234 022274 105227          INCB  (PC)+          ::COUNT THE CHARACTER
5235 022276 000000          $CHARCNT: .WORD 0          ::CHARACTER COUNT STORAGE
5236 022300 000207          $TYPEX: RTS   PC
5237

```

```
5238 .SBTTL TTY INPUT ROUTINE
5239
5240 ::*****
5241 .ENABL LSB
5242
5243 ::*****
5244 ::SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
5245 ::ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
5246 ::SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
5247 ::WHEN OPERATING IN TTY FLAG MODE.
5248 022302 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ::IS THE SOFT-SWR SELECTED?
5249 022310 001074 BNE 15$ ::BRANCH IF NO
5250 022312 105777 156626 TSTB @STKS ::CHAR THERE?
5251 022316 100071 BPL 15$ ::IF NO, DON'T WAIT AROUND
5252 022320 117746 156622 MOVB @STKB,-(SP) ::SAVE THE CHAR
5253 022324 042716 177600 BIC #^C177,(SP) ::STRIP-OFF THE ASCII
5254 022330 022726 000007 CMP #7,(SP)+ ::IS IT A CONTROL G?
5255 022334 001062 BNE 15$ ::NO, RETURN TO USER
5256 022336 123727 001134 000001 CMPB $AUTOB,#1 ::ARE WE RUNNING IN AUTO-MODE?
5257 022344 001456 BEQ 15$ ::BRANCH IF YES
5258
5259 022346 104401 023037 $GTSWR: TYPE ,SCNTLG ::ECHO THE CONTROL-G (^G)
5260 022352 104401 023044 TYPE ,SMSWR ::TYPE CURRENT CONTENTS
5261 022356 013746 000176 MOV SWREG,-(SP) ::SAVE SWREG FOR TYPEOUT
5262 022362 104402 TYPEPOC ::GO TYPE--OCTAL ASCII(ALL DIGITS)
5263 022364 104401 023055 TYPE ,SMNEW ::PROMPT FOR NEW SWR
5264 022370 005046 19$: CLR -(SP) ::CLEAR COUNTER
5265 022372 005046 CLR -(SP) ::THE NEW SWR
5266 022374 105777 156544 7$: TSTB @STKS ::CHAR THERE?
5267 022400 100375 BPL 7$ ::IF NOT TRY AGAIN
5268
5269 022402 117746 156540 MOVB @STKB,-(SP) ::PICK UP CHAR
5270 022406 042716 177600 BIC #^C177,(SP) ::MAKE IT 7-BIT ASCII
5271
5272
5273
5274 022412 021627 000025 9$: CMP (SP),#25 ::IS IT A CONTROL-U?
5275 022416 001005 BNE 10$ ::BRANCH IF NOT
5276 022420 104401 023032 TYPE ,SCNTLU ::YES, ECHO CONTROL-U (^U)
5277 022424 062706 000006 20$: ADD #6,SP ::IGNORE PREVIOUS INPUT
5278 022430 000757 BR 19$ ::LET'S TRY IT AGAIN
5279
5280
5281 022432 021627 000015 10$: CMP (SP),#15 ::IS IT A <CR>?
5282 022436 001022 BNE 16$ ::BRANCH IF NO
5283 022440 005766 000004 TST 4(SP) ::YES, IS IT THE FIRST CHAR?
5284 022444 001403 BEQ 11$ ::BRANCH IF YES
5285 022446 016677 000002 156464 MOV 2(SP),@SWR ::SAVE NEW SWR
5286 022454 062706 000006 11$: ADD #6,SP ::CLEAR UP STACK
5287 022460 104401 001171 14$: TYPE ,SCRLF ::ECHO <CR> AND <LF>
5288 022464 123727 001135 000001 CMPB $INTAG,#1 ::RE-ENABLE TTY KBD INTERRUPTS?
5289 022472 001003 BNE 15$ ::BRANCH IF NOT
5290 022474 012777 000100 156442 MOV #100,@STKS ::RE-ENABLE TTY KBD INTERRUPTS
5291 022502 000002 15$: RTI ::RETURN
5292 022504 004737 022160 16$: JSR PC,$TYPEC ::ECHO CHAR
5293 022510 021627 000060 CMP (SP),#60 ::CHAR < 0?
```



```

5294 022514 002420          BLT      18$          ::BRANCH IF YES
5295 022516 021627 000067    CMP      (SP),#67      ::CHAR > 7?
5296 022522 003015          BGT      18$          ::BRANCH IF YES
5297 022524 042726 000060    BIC      #60,(SP)+      ::STRIP-OFF ASCII
5298 022530 005766 000002    TST      2(SP)         ::IS THIS THE FIRST CHAR
5299 022534 001403          BEQ      17$          ::BRANCH IF YES
5300 022536 006316          ASL      (SP)         ::NO, SHIFT PRESENT
5301 022540 006316          ASL      (SP)         ::CHAR OVER TO MAKE
5302 022542 006316          ASL      (SP)         ::ROOM FOR NEW ONE.
5303 022544 005266 000002    17$: INC      2(SP)         ::KEEP COUNT OF CHAR
5304 022550 056616 177776    BIS      -2(SP),(SP)    ::SET IN NEW CHAR
5305 022554 000707          BR       7$          ::GET THE NEXT ONE
5306 022556 104401 001170    18$: TYPE    $QUES      ::TYPE ?<CR><LF>
5307 022562 000720          BR       20$          ::SIMULATE CONTROL-U
5308          .DSABL   LSB
5309
5310
5311          ::*****
5312          ::THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
5313          ::CALL:
5314          ::      RDCHR          ::INPUT A SINGLE CHARACTER FROM THE TTY
5315          ::      RETURN HERE   ::CHARACTER IS ON THE STACK
5316          ::                  ::WITH PARITY BIT STRIPPED OFF
5317          ::
5318
5319 022564 011646          $RDCHR: MOV      (SP),-(SP)    ::PUSH DOWN THE PC
5320 022566 016666 000004 000002    MOV      4(SP),2(SP)  ::SAVE THE PS
5321 022574 105777 156344    1$:  TSTB     @ $IKS      ::WAIT FOR
5322 022600 100375          BPL      1$          ::A CHARACTER
5323 022602 117766 156340 000004    MOVB     @ $TKB,4(SP)  ::READ THE TTY
5324 022610 042766 177600 000004    BIC      #^C<177>,4(SP)  ::GET RID OF JUNK IF ANY
5325 022616 026627 000004 000023    CMP      4(SP),#23    ::IS IT A CONTROL-S?
5326 022624 001013          BNE      3$          ::BRANCH IF NO
5327 022626 105777 156312    2$:  TSTB     @ $TKS      ::WAIT FOR A CHARACTER
5328 022632 100375          BPL      2$          ::LOOP UNTIL ITS THERE
5329 022634 117746 156306    MOVB     @ $TKB,-(SP)  ::GET CHARACTER
5330 022640 042716 177600          BIC      #^C177,(SP)  ::MAKE IT 7-BIT ASCII
5331 022644 022627 000021    CMP      (SP)+,#21    ::IS IT A CONTROL-Q?
5332 022650 001366          BNE      2$          ::IF NOT DISCARD IT
5333 022652 000750          BR       1$          ::YES, RESUME
5334 022654 026627 000004 000021    3$:  CMP      4(SP),#$XON  ::IS IT A RANDOM XON?
5335 022662 001744          BEQ      1$          ::BRANCH IF YES
5336 022664 026627 000004 000140    CMP      4(SP),#140  ::IS IT UPPER CASE?
5337 022672 002407          BLT      4$          ::BRANCH IF YES
5338 022674 026627 000004 000175    CMP      4(SP),#175  ::IS IT A SPECIAL CHAR?
5339 022702 003003          BGT      4$          ::BRANCH IF YES
5340 022704 042766 000040 000004    BIC      #40,4(SP)  ::MAKE IT UPPER CASE
5341 022712 000002    4$:  RTI          ::GO BACK TO USER
5342          ::*****
5343          ::THIS ROUTINE WILL INPUT A STRING FROM THE TTY
5344          ::CALL:
5345          ::      RDLIN          ::INPUT A STRING FROM THE TTY
5346          ::      RETURN HERE   ::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
5347          ::                  ::TERMINATOR WILL BE A BYTE OF ALL 0'S
5348
5349 022714 010346          $RDLIN: MOV      R3,-(SP)    ::SAVE R3

```

5350	022716	012703	023022	1\$:	MOV	#STTYIN,R3	::GET ADDRESS
5351	022722	022703	023032	2\$:	CMP	#STTYIN+8.,R3	::BUFFER FULL?
5352	022726	101405			BLOS	4\$	::BR IF YES
5353	022730	104410			RDCHR		::GO READ ONE CHARACTER FROM THE TTY
5354	022732	112613			MOVB	(SP)+,(R3)	::GET CHARACTER
5355	022734	122713	000177	10\$:	CMPB	#177,(R3)	::IS IT A RUBOUT
5356	022740	001003			BNE	3\$	::SKIP IF NOT
5357	022742	104401	001170	4\$:	TYPE	,SQUES	::TYPE A '?'
5358	022746	000763			BR	1\$	::CLEAR THE BUFFER AND LOOP
5359	022750	111337	023020	3\$:	MOVB	(R3),9\$	::ECHO THE CHARACTER
5360	022754	104401	023020		TYPE	,9\$	
5361	022760	122723	000015		CMPB	#15,(R3)+	::CHECK FOR RETURN
5362	022764	001356			BNE	2\$	::LOOP IF NOT RETURN
5363	022766	105063	177777		CLRB	-1(R3)	::CLEAR RETURN (THE 15)
5364	022772	104401	001172		TYPE	,SLF	::TYPE A LINE FEED
5365	022776	012603			MOV	(SP)+,R3	::RESTORE R3
5366	023000	011646			MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
5367	023002	016666	000004 000002		MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT
5368	023010	012766	023022 000004		MOV	#STTYIN,4(SP)	
5369	023016	000002			RTI		::RETURN
5370	023020	000		9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE
5371	023021	000			.BYTE	0	::TERMINATOR
5372	023022	000010		\$TTYIN:	.BLKB	8.	::RESERVE 8 BYTES FOR TTY INPUT
5373	023032	052536	005015 000	\$CNTLU:	.ASCIIZ	/^U/<15><12>	::CONTROL 'U'
5374	023037	136	006507 000012	\$CNTLG:	.ASCIIZ	/^G/<15><12>	::CONTROL 'G'
5375	023044	005015	053523 020122	\$MSWR:	.ASCIIZ	<15><12>/SWR = /	
5376	023052	020075	000				
5377	023055	040	047040 053505	\$MNEW:	.ASCIIZ	/ NEW = /	
5378	023062	036440	000040				


```

5379 .SBTTL APT COMMUNICATIONS ROUTINE
5380
5381 *****
5382 023066 112737 000001 023332 $ATY1: MOVB #1,$FFLG ::TO REPORT FATAL ERROR
5383 023074 112737 000001 023330 $ATY3: MOVB #1,$MFLG ::TO TYPE A MESSAGE
5384 023102 000403 BR $ATYC
5385 023104 112737 000001 023332 $ATY4: MOVB #1,$FFLG ::TO ONLY REPORT FATAL ERROR
5386 023112 $ATYC:
5387 023112 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
5388 023114 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
5389 023116 105737 023330 TSTB $MFLG ::SHOULD TYPE A MESSAGE?
5390 023122 001450 BEQ 5$ ::IF NOT: BR
5391 023124 122737 000001 001214 CMPB #APTENV,$ENV ::OPERATING UNDER APT?
5392 023132 001031 BNE 3$ ::IF NOT: BR
5393 023134 132737 000100 001215 BITB #APTPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
5394 023142 001425 BEQ 3$ ::IF NOT: BR
5395 023144 017600 000004 MOV @4(SP),R0 ::GET MESSAGE ADDR.
5396 023150 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
5397 023156 005737 001174 1$: TST $MSGTYPE ::SEE IF DONE W/ LAST XMISSION?
5398 023162 001375 BNE 1$ ::IF NOT: WAIT
5399 023164 010037 001210 MOV R0,$MSGAD ::PUT ADDR IN MAILBOX
5400 023170 105720 2$: TSTB (R0)+ ::FIND END OF MESSAGE
5401 023172 001376 BNE 2$
5402 023174 163700 001210 SUB $MSGAD,R0 ::SUB START OF MESSAGE
5403 023200 006200 ASR R0 ::GET MESSAGE LGTH IN WORDS
5404 023202 010037 001212 MOV R0,$MSGGLT ::PUT LENGTH IN MAILBOX
5405 023206 012737 000004 001174 MOV #4,$MSGTYPE ::TELL APT TO TAKE MSG.
5406 023214 000413 BR 5$
5407 023216 017637 000004 023242 3$: MOV @4(SP),4$ ::PUT MSG ADDR IN JSR LINKAGE
5408 023224 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDRESS
5409 023232 013746 177776 MOV 177776,-(SP) ::PUSH 177776 ON STACK
5410 023236 004737 021746 JSR PC,$TYPE ::CALL TYPE MACRO
5411 023242 000000 4$: .WORD 0
5412 023244 5$:
5413 023244 105737 023332 10$: TSTB $FFLG ::SHOULD REPORT FATAL ERROR?
5414 023250 001416 BEQ 12$ ::IF NOT: BR
5415 023252 005737 001214 TST $ENV ::RUNNING UNDER APT?
5416 023256 001413 BEQ 12$ ::IF NOT: BR
5417 023260 005737 001174 11$: TST $MSGTYPE ::FINISHED LAST MESSAGE?
5418 023264 001375 BNE 11$ ::IF NOT: WAIT
5419 023266 017637 000004 001176 MOV @4(SP),$FATAL ::GET ERROR #
5420 023274 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
5421 023302 005237 001174 INC $MSGTYPE ::TELL APT TO TAKE ERROR
5422 023306 105037 023332 12$: CLRB $FFLG ::CLEAR FATAL FLAG
5423 023312 105037 023331 CLRB $LFLG ::CLEAR LOG FLAG
5424 023316 105037 023330 CLRB $MFLG ::CLEAR MESSAGE FLAG
5425 023322 012601 MOV (SP)+,R1 ::POP STACK INTO R1
5426 023324 012600 MOV (SP)+,R0 ::POP STACK INTO R0
5427 023326 000207 RTS PC ::RETURN
5428 023330 000 $MFLG: .BYTE 0 ::MESSG. FLAG
5429 023331 000 $LFLG: .BYTE 0 ::LOG FLAG
5430 023332 000 $FFLG: .BYTE 0 ::FATAL FLAG
5431 023334 .EVEN
5432 000200 APTSIZE=200
5433 000001 APTENV=001
5434 000100 APTPOOL=100

```

KDF11-B / DLV11-J TEST MACY11 30(1046) 05-OCT-81 16:37 PAGE 110 F 9
CJDLAA.P11 29-SEP-81 11:42 APT COMMUNICATIONS ROUTINE

SEQ 0109

5435

000040

APTC SUP=040

.SBTTL ERROR HANDLER ROUTINE

```
*****  
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,  
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL  
*AND GO TO MYTYPE ON ERROR  
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:  
*SW15=1      HALT ON ERROR  
*SW13=1      INHIBIT ERROR TYPEOUTS  
*SW10=1      BELL ON ERROR  
*SW09=1      LOOP ON ERROR  
*CALL  
*          ERROR      N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
```

\$ERROR:

```
7$:      CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR  
        INCB          $ERRFLG      ;;SET THE ERROR FLAG  
        BEQ           7$          ;;DON'T LET THE FLAG GO TO ZERO  
        MOV          $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG  
        BIT          #BIT10,@SWR    ;;BELL ON ERROR?  
        BEQ           1$          ;;NO - SKIP  
        TYPE         ,SBELL        ;;RING BELL  
1$:      INC          $ERTTL        ;;COUNT THE NUMBER OF ERRORS  
        MOV          (SP),$ERRPC    ;;GET ADDRESS OF ERROR INSTRUCTION  
        SUB          #2,$ERRPC  
        MOVB         @ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE  
        BIT          #BIT13,@SWR    ;;SKIP TYPEOUT IF SET  
        BNE          20$          ;;SKIP TYPEOUTS  
        JSR          PC,MYTYPE      ;;GO TO USER ERROR ROUTINE  
        TYPE         ,SCRLF  
20$:     CMPB         #APTENV,$ENV   ;;RUNNING IN APT MODE  
        BNE          2$          ;;NO,SKIP APT ERROR REPORT  
        MOVB         $ITEMB,21$    ;;SET ITEM NUMBER AS ERROR NUMBER  
        JSR          PC,$ATY4      ;;REPORT FATAL ERROR TO APT  
21$:     .BYTE        0  
        .BYTE        0  
22$:     BR          22$          ;;APT ERROR LOOP  
2$:      TST          @SWR          ;;HALT ON ERROR  
        BPL          3$          ;;SKIP IF CONTINUE  
        HALT         ;;HALT ON ERROR!  
3$:      BIT          #BIT09,@SWR   ;;TEST FOR CHANGE IN SOFT-SWR  
        BEQ          4$          ;;LOOP ON ERROR SWITCH SET?  
        MOV          $LPERR,(SP)    ;;FUDGE RETURN FOR LOOPING  
        TST          $ESCAPE       ;;CHECK FOR AN ESCAPE ADDRESS  
        BEQ          5$          ;;BR IF NONE  
        MOV          $ESCAPE,(SP)  ;;FUDGE RETURN ADDRESS FOR ESCAPE  
4$:      TST          $ESCAPE  
5$:      CMP          #SENDAD,@#42  ;;ACT-11 AUTO-ACCEPT?  
        BNE          6$          ;;BRANCH IF NO  
        HALT         ;;YES  
6$:      RTI                  ;;RETURN
```

5490
 5491
 5492
 5493
 5494
 5495
 5496
 5497
 5498
 5499
 5500
 5501
 5502
 5503
 5504
 5505
 5506
 5507
 5508
 5509
 5510
 5511
 5512
 5513
 5514
 5515
 5516
 5517
 5518
 5519
 5520
 5521
 5522
 5523
 5524
 5525
 5526
 5527
 5528
 5529
 5530
 5531
 5532
 5533
 5534
 5535
 5536
 5537
 5538
 5539
 5540
 5541
 5542
 5543
 5544
 5545

023534
 023534 104407
 023536 032777 040000 155374
 023544 001114
 023546 000416
 023550 013746 000004
 023554 012737 023574 000004
 023562 005737 177060
 023566 012637 000004
 023572 000463
 023574 022626
 023576 012637 000004
 023602 000423
 023604
 023604 032777 000400 155326
 023612 001404
 023614 127737 155320 001102
 023622 001465
 023624 105737 001103
 023630 001421
 023632 123737 001115 001103
 023640 101015
 023642 032777 001000 155270
 023650 001404
 023652 013737 001110 001106
 023660 000446
 023662 105037 001103
 023666 005037 001160
 023672 000415
 023674 032777 004000 155236
 023702 001011
 023704 005737 001202
 023710 001406
 023712 005237 001104
 023716 023737 001160 001104
 023724 002024
 023726 012737 000001 001104
 023734 013737 024012 001160
 023742 105237 001102
 023746 113737 001102 001200

.SBTTL SCOPE HANDLER ROUTINE

```

*****
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERRFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1      LOOP ON TEST
*SW11=1      INHIBIT ITERATIONS
*SW09=1      LOOP ON ERROR
*SW08=1      LOOP ON TEST IN SWR<7:0>
*CALL
*          SCOPE          ;;SCOPE=IOT

$SCOPE:
      CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
      BIT            #BIT14,@SWR          ;;LOOP ON PRESENT TEST?
      BNE            $OVER          ;;YES IF SW14=1
      #####START OF CODE FOR THE XOR TESTER#####
      $XTSTR: BR      6$          ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
                                   ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
      MOV            @#ERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
      MOV            #5$,@#ERRVEC          ;;SET FOR TIMEOUT
      TST            @#177060          ;;TIME OUT ON XOR?
      MOV            (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
      BR             $SVLAD          ;;GO TO THE NEXT TEST
      5$: CMP        (SP)+,(SP)+          ;;CLEAR THE STACK AFTER A TIME OUT
      MOV            (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
      BR             7$          ;;LOOP ON THE PRESENT TEST
      6$:#####END OF CODE FOR THE XOR TESTER#####
      BIT            #BIT08,@SWR          ;;LOOP ON SPEC. TEST?
      BEQ            2$          ;;BR IF NO
      CMPB           @SWR,$TSTNM          ;;ON THE RIGHT TEST? SWR<7:0>
      BEQ            $OVER          ;;BR IF YES
      2$: TSTB       $ERRFLG          ;;HAS AN ERROR OCCURRED?
      BEQ            3$          ;;BR IF NO
      CMPB           $ERMAX,$ERRFLG          ;;MAX. ERRORS FOR THIS TEST OCCURRED?
      BHI            3$          ;;BR IF NO
      BIT            #BIT09,@SWR          ;;LOOP ON ERROR?
      BEQ            4$          ;;BR IF NO
      7$: MOV        $LPERR,$LPADR          ;;SET LOOP ADDRESS TO LAST SCOPE
      BR             $OVER
      4$: CLRB       $ERRFLG          ;;ZERO THE ERROR FLAG
      CLR            $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
      BR             1$          ;;ESCAPE TO THE NEXT TEST
      3$: BIT        #BIT11,@SWR          ;;INHIBIT ITERATIONS?
      BNE            1$          ;;BR IF YES
      TST            $PASS          ;;IF FIRST PASS OF PROGRAM
      BEQ            1$          ;;INHIBIT ITERATIONS
      INC            $ICNT          ;;INCREMENT ITERATION COUNT
      CMP            $TIMES,$ICNT          ;;CHECK THE NUMBER OF ITERATIONS MADE
      BGE            $OVER          ;;BR IF MORE ITERATION REQUIRED
      1$: MOV        #1,$ICNT          ;;REINITIALIZE THE ITERATION COUNTER
      MOV            $MXCNT,$TIMES          ;;SET NUMBER OF ITERATIONS TO DO
      $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
      MOVB           $TSTNM,$TESTN          ;;SET TEST NUMBER IN APT MAILBOX

```


5546	023754	011637	001106		MOV	(SP), \$LPADR	::SAVE SCOPE LOOP ADDRESS
5547	023760	011637	001110		MOV	(SP), \$LPERR	::SAVE ERROR LOOP ADDRESS
5548	023764	005037	001162		CLR	\$ESCAPE	::CLEAR THE ESCAPE FROM ERROR ADDRESS
5549	023770	112737	000001	001115	MOVB	#1, \$ERMAX	::ONLY ALLOW ONE(1) ERROR ON NEXT TEST
5550	023776	013777	001102	155136	\$OVER: MOV	\$TSTNM, @DISPLAY	::DISPLAY TEST NUMBER
5551	024004	013716	001106		MOV	\$LPADR, (SP)	::FUDGE RETURN ADDRESS
5552	024010	000002			RTI		::FIXES PS
5553	024012	003720			\$MXCNT: 2000.		::MAX. NUMBER OF ITERATIONS

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```
*****  
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT  
*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE  
*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED  
*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE  
*REPLACED WITH SPACES.  
*CALL:
```

```
*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK  
*      TYPDS      ;;GO TO THE ROUTINE
```

\$TYPDS:

```
MOV      R0,-(SP)      ;;PUSH R0 ON STACK  
MOV      R1,-(SP)      ;;PUSH R1 ON STACK  
MOV      R2,-(SP)      ;;PUSH R2 ON STACK  
MOV      R3,-(SP)      ;;PUSH R3 ON STACK  
MOV      R5,-(SP)      ;;PUSH R5 ON STACK  
MOV      #20200,-(SP)   ;;SET BLANK SWITCH AND SIGN  
MOV      20(SP),R5      ;;GET THE INPUT NUMBER  
BPL      1$            ;;BR IF INPUT IS POS.  
NEG      R5            ;;MAKE THE BINARY NUMBER POS.  
MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.  
1$:      CLR      R0      ;;ZERO THE CONSTANTS INDEX  
MOV      #SDBLK,R3      ;;SETUP THE OUTPUT POINTER  
MOVB     #'',(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK  
2$:      CLR      R2      ;;CLEAR THE BCD NUMBER  
MOV      $DTBL(R0),R1    ;;GET THE CONSTANT  
3$:      SUB      R1,R5    ;;FORM THIS BCD DIGIT  
BLT      4$            ;;BR IF DONE  
INC      R2            ;;INCREASE THE BCD DIGIT BY 1  
BR       3$  
4$:      ADD      R1,R5    ;;ADD BACK THE CONSTANT  
TST      R2            ;;CHECK IF BCD DIGIT=0  
BNE      5$            ;;FALL THROUGH IF 0  
TSTB     (SP)          ;;STILL DOING LEADING 0'S?  
BMI      7$            ;;BR IF YES  
5$:      ASLB     (SP)      ;;MSD?  
BCC      6$            ;;BR IF NO  
MOVB     1(SP),-1(R3)    ;;YES--SET THE SIGN  
6$:      BIS      #'0,R2    ;;MAKE THE BCD DIGIT ASCII  
7$:      BIS      #' ,R2    ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT  
MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER  
TST      (R0)+          ;;JUST INCREMENTING  
CMP      R0,#10        ;;CHECK THE TABLE INDEX  
BLT      2$            ;;GO DO THE NEXT DIGIT  
BGT      8$            ;;GO TO EXIT  
MOV      R5,R2          ;;GET THE LSD  
BR       6$            ;;GO CHANGE TO ASCII  
8$:      TSTB     (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?  
BPL      9$            ;;BR IF NO  
MOVB     -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING  
9$:      CLRB     (R3)      ;;SET THE TERMINATOR  
MOV      (SP)+,R5      ;;POP STACK INTO R5  
MOV      (SP)+,R3      ;;POP STACK INTO R3  
MOV      (SP)+,R2      ;;POP STACK INTO R2
```

```
5554  
5555  
5556  
5557  
5558  
5559  
5560  
5561  
5562  
5563  
5564  
5565  
5566 024014  
5567 024014 010046  
5568 024016 010146  
5569 024020 010246  
5570 024022 010346  
5571 024024 010546  
5572 024026 012746 020200  
5573 024032 016605 000020  
5574 024036 100004  
5575 024040 005405  
5576 024042 112766 000055 000001  
5577 024050 005000 1$:  
5578 024052 012703 024230  
5579 024056 112723 000040  
5580 024062 005002 2$:  
5581 024064 016001 024220  
5582 024070 160105 3$:  
5583 024072 002402  
5584 024074 005202  
5585 024076 000774  
5586 024100 060105 4$:  
5587 024102 005702  
5588 024104 001002  
5589 024106 105716  
5590 024110 100407  
5591 024112 106316 5$:  
5592 024114 103003  
5593 024116 116663 000001 177777  
5594 024124 052702 000060  
5595 024130 052702 000040 7$:  
5596 024134 110223  
5597 024136 005720  
5598 024140 020027 000010  
5599 024144 002746  
5600 024146 003002  
5601 024150 010502  
5602 024152 000764  
5603 024154 105726 8$:  
5604 024156 100003  
5605 024160 116663 177777 177776  
5606 024166 105013 9$:  
5607 024170 012605  
5608 024172 012603  
5609 024174 012602
```


5610	024176	012601			MOV	(SP)+,R1	::POP STACK INTO R1
5611	024200	012600			MOV	(SP)+,R0	::POP STACK INTO R0
5612	024202	104401	024230		TYPE	\$DBLK	::NOW TYPE THE NUMBER
5613	024206	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
5614	024214	012616			MOV	(SP)+,(SP)	
5615	024216	000002			RTI		::RETURN TO USER
5616	024220	023420			\$DTBL:	10000.	
5617	024222	001750				1000.	
5618	024224	000144				100.	
5619	024226	000012				10.	
5620	024230	000004			\$DBLK:	.BLKW 4	

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOS      ;;CALL FOR TYPEOUT
*      .BYTE     N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE     M      ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON      ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC      ;;CALL FOR TYPEOUT
$TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
        MOV      1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
        MOV      (SP)+, $OMODE+1      ;;NUMBER OF DIGITS TO TYPE
        ADD      #2, (SP)      ;;ADJUST RETURN ADDRESS
        BR       $TYPON
$TYPOC: MOV      #1, $OFILL      ;;SET THE ZERO FILL SWITCH
        MOV      #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
        MOV      #5, $OCNT      ;;SET THE ITERATION COUNT
        MOV      R3, -(SP)      ;;SAVE R3
        MOV      R4, -(SP)      ;;SAVE R4
        MOV      R5, -(SP)      ;;SAVE R5
        MOV      $OMODE+1, R4      ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6, R4      ;;SUBTRACT IT FOR MAX. ALLOWED
        MOV      R4, $OMODE      ;;SAVE IT FOR USE
        MOV      $OFILL, R4      ;;GET THE ZERO FILL SWITCH
        MOV      12(SP), R5      ;;PICKUP THE INPUT NUMBER
        CLR      R3      ;;CLEAR THE OUTPUT WORD
        ROL      R5      ;;ROTATE MSB INTO 'C'
        BR       3$      ;;GO DO MSB
        ROL      R5      ;;FORM THIS DIGIT
        ROL      R5
        ROL      R5
        MOV      R5, R3
        ROL      R3      ;;GET LSB OF THIS DIGIT
        DECB     $OMODE      ;;TYPE THIS DIGIT?
        BPL      7$      ;;BR IF NO
        BIC      #177770, R3      ;;GET RID OF JUNK
        BNE      4$      ;;TEST FOR 0
        TST      R4      ;;SUPPRESS THIS 0?
        BEQ      5$      ;;BR IF YES
1$:
2$:
3$:
4$:
5$:

```

5621				
5622				
5623				
5624				
5625				
5626				
5627				
5628				
5629				
5630				
5631				
5632				
5633				
5634				
5635				
5636				
5637				
5638				
5639				
5640				
5641				
5642				
5643				
5644				
5645				
5646	024240	017646	000000	
5647	024244	116637	000001	024463
5648	024252	112637	024465	
5649	024256	062716	000002	
5650	024262	000406		
5651	024264	112737	000001	024463
5652	024272	112737	000006	024465
5653	024300	112737	000005	024462
5654	024306	010346		
5655	024310	010446		
5656	024312	010546		
5657	024314	113704	024465	
5658	024320	005404		
5659	024322	062704	000006	
5660	024326	110437	024464	
5661	024332	113704	024463	
5662	024336	016605	000012	
5663	024342	005003		
5664	024344	006105		
5665	024346	000404		
5666	024350	006105		
5667	024352	006105		
5668	024354	006105		
5669	024356	010503		
5670	024360	006103		
5671	024362	105337	024464	
5672	024366	100016		
5673	024370	042703	177770	
5674	024374	001002		
5675	024376	005704		
5676	024400	001403		

5677	024402	005204		4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S
5678	024404	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
5679	024410	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
5680	024414	110337	024460		MOVB	R3,8\$	::SAVE FOR TYPING
5681	024420	104401	024460		TYPE	8\$	::GO TYPE THIS DIGIT
5682	024424	105337	024462	7\$:	DECB	\$OCNT	::COUNT BY 1
5683	024430	003347			BGT	2\$	::BR IF MORE TO DO
5684	024432	002402			BLT	6\$	::BR IF DONE
5685	024434	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
5686	024436	000744			BR	2\$	::GO DO THE LAST DIGIT
5687	024440	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
5688	024442	012604			MOV	(SP)+,R4	::RESTORE R4
5689	024444	012603			MOV	(SP)+,R3	::RESTORE R3
5690	024446	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
5691	024454	012616			MOV	(SP)+,(SP)	
5692	024456	000002			RTI		::RETURN
5693	024460	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
5694	024461	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
5695	024462	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
5696	024463	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
5697	024464	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

5698
 5699
 5700
 5701
 5702
 5703
 5704
 5705
 5706
 5707
 5708
 5709
 5710
 5711
 5712
 5713
 5714
 5715
 5716
 5717
 5718
 5719
 5720
 5721
 5722
 5723
 5724
 5725
 5726
 5727
 5728
 5729
 5730
 5731
 5732
 5733
 5734
 5735
 5736
 5737
 5738
 5739
 5740
 5741
 5742
 5743

024466 010046
 024470 016600 000002
 024474 005740
 024476 111000
 024500 006300
 024502 016000 024522
 024506 000200

024510 011646
 024512 016666 000004 000002
 024520 000002

024522 024510
 024524 021746
 024526 024264
 024530 024240
 024532 024300
 024534 024014
 024536 022352
 024540 022302
 024542 022564
 024544 022714

000001

.SBTTL TRAP DECODER

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

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

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

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

.SBTTL TRAP TABLE

;;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
 ;;BY THE "TRAP" INSTRUCTION.

	ROUTINE	
\$TRPAD:	.WORD \$TRAP2	
\$TYPE	::CALL=TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
\$TYPOC	::CALL=TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOS	::CALL=TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPON	::CALL=TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPDS	::CALL=TYPDS	TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
\$GTSWR	::CALL=GTSWR	TRAP+6(104406) GET SOFT-SWR SETTING
\$CKSWR	::CALL=CKSWR	TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
\$RDCHR	::CALL=RDCHR	TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
\$RDLIN	::CALL=RDLIN	TRAP+11(104411) TTY TYPEIN STRING ROUTINE

;;MAXMEM FOR APT = 17400
 ;;MAXMEM NON APT = 17500 (ABS LOADER)

.END

ABASE = 176500
 ACDW1 = 000000
 ACDW2 = 000000
 ACPUOP = 000000
 ACTCH = 016120
 ADDW0 = 000000
 ADDW1 = 000000
 ADDW10 = 000000
 ADDW11 = 000000
 ADDW12 = 000000
 ADDW13 = 000000
 ADDW14 = 000000
 ADDW15 = 000000
 ADDW2 = 000000
 ADDW3 = 000000
 ADDW4 = 000000
 ADDW5 = 000000
 ADDW6 = 000000
 ADDW7 = 000000
 ADDW8 = 000000
 ADDW9 = 000000
 ADEVCT = 000000
 ADEVN = 000000
 ADRS = 0000001
 AENV = 000000
 AENVN = 000000
 AFATAL = 000000
 AMADR1 = 000000
 AMADR2 = 000000
 AMADR3 = 000000
 AMADR4 = 000000
 AMAMS1 = 000000
 AMAMS2 = 000000
 AMAMS3 = 000000
 AMAMS4 = 000000
 AMSGAD = 000000
 AMSGLG = 000000
 AMSGTY = 000000
 AMTYP1 = 000000
 AMTYP2 = 000000
 AMTYP3 = 000000
 AMTYP4 = 000000
 APASS = 000000
 APRIOR = 000000
 APTCSU = 000040
 APTENV = 000001
 APTSIZ = 000200
 APTSPO = 000100
 ASSEMB = 000010
 ASWREG = 000000
 ATESTN = 000000
 AUNIT = 000000
 AUSWR = 000031

AVECT1 = 000300
 AVECT2 = 000000
 BIT0 = 000001
 BIT00 = 000001
 BIT01 = 000002
 BIT02 = 000004
 BIT03 = 000010
 BIT04 = 000020
 BIT05 = 000040
 BIT06 = 000100
 BIT07 = 000200
 BIT08 = 000400
 BIT09 = 001000
 BIT1 = 000002
 BIT10 = 002000
 BIT11 = 004000
 BIT12 = 010000
 BIT13 = 020000
 BIT14 = 040000
 BIT15 = 100000
 BIT2 = 000004
 BIT3 = 000010
 BIT4 = 000020
 BIT5 = 000040
 BIT6 = 000100
 BIT7 = 000200
 BIT8 = 000400
 BIT9 = 001000
 BPTVEC = 000014
 CHCNT = 011514
 CHCTR = 016116
 CHMASK = 011512
 CHMSK = 016114
 CHOTAB = 014114
 CH1TAB = 014314
 CH2TAB = 014514
 CH3TAB = 014714
 CH4TAB = 015114
 CH5TAB = 015314
 CH6TAB = 015514
 CH7TAB = 015714
 CKSWR = 104407
 CONADR = 001274
 CONSOL = 001324
 CONST = 001272
 CR = 000015
 CRLF = 000200
 CYCLE = 020540
 DATA = 007042
 DDISP = 177570
 DIAGMC = 000000
 DISPLA = 001142
 DISPRE = 000174

DLADD = 001256
 DLBASE = 001312
 DLVEC = 001260
 DLV211 = 001310
 DSWR = 177570
 EMTVEC = 000030
 ENDDER = 021324
 ERFLG = 000400
 ERRCHK = 010100
 ERRCNT = 006772
 ERRORF = 003604
 ERRVEC = 000004
 ERWRD = 016122
 FSR = 005010
 ESR1 = 007552
 EXITFL = 003606
 FLAG = 016516
 GTSWR = 104406
 HOLDSC = 016517
 HOWLON = 000000
 HT = 000011
 I = 001254
 INTFLG = 017640
 INTRTA = 011452
 INTSRV = 017632
 IOTVEC = 000020
 KDDLJ = 001306
 KDDLJV = 020276
 KDF11B = 001322
 KD2DLV = 017642
 LF = 000012
 LOOP = 002000
 MASK = 001320
 MODTST = 010162
 MYTYPE = 021434
 NODLV = 001304
 NUMBER = 006774
 ONKDF = 017764
 PASS = 003602
 PHASE2 = 001276
 PIRQ = 177772
 PIRQVE = 000240
 PRIERR = 011516
 PR0 = 000000
 PR1 = 000040
 PR2 = 000100
 PR3 = 000140
 PR4 = 000200
 PR5 = 000240
 PR6 = 000300
 PR7 = 000340
 PS = 177776

PSW = 177776
 PWRVEC = 000024
 P1CNT = 001300
 RBUF = 001264
 RCH0 = 011412
 RCH1 = 011414
 RCH2 = 011416
 RCH3 = 011420
 RCH4 = 011422
 RCH5 = 011424
 RCH6 = 011426
 RCH7 = 011430
 RCSR = 001262
 RCVR0S = 010612
 RCVR1S = 010672
 RCVR2S = 010752
 RCVR3S = 011032
 RCVR4S = 011112
 RCVR5S = 011172
 RCVR6S = 011252
 RCVR7S = 011332
 RDCHR = 104410
 RDLIN = 104411
 REC = 007044
 REG = 000004
 REGSAV = 016512
 REPEAT = 020222
 RESVEC = 000010
 RHLA = 007164
 ROSRV = 012534
 R1SRV = 012672
 R2SRV = 013030
 R3SRV = 013166
 R4SRV = 013324
 R5SRV = 013462
 R6 = 0000006
 R6SRV = 013620
 R7 = 0000007
 R7SRV = 013756
 SECSLU = 001302
 SEREND = 014114
 SETCLR = 000006
 SFTREG = 001316
 SIZE = 016614
 SRVEND = 011412
 STACK = 001100
 START = 001330
 STATEN = 016114
 STKLMT = 177774
 SWR = 001140
 SWREG = 000176
 SW0 = 000001
 SW00 = 000001

SW01 = 000002
 SW02 = 000004
 SW03 = 000010
 SW04 = 000020
 SW05 = 000040
 SW06 = 000100
 SW07 = 000200
 SW08 = 000400
 SW09 = 001000
 SW1 = 000002
 SW10 = 002000
 SW11 = 004000
 SW12 = 010000
 SW13 = 020000
 SW14 = 040000
 SW15 = 100000
 SW2 = 000004
 SW3 = 000010
 SW4 = 000020
 SW5 = 000040
 SW6 = 000100
 SW7 = 000200
 SW8 = 000400
 SW9 = 001000
 SYM = 000037
 SYMD = 000007
 SYMS = 000007
 TABEND = 011512
 TABL6 = 016172
 TABL7 = 016212
 TABL8 = 016232
 TBITVE = 000014
 TBUF = 001270
 TCH0 = 011432
 TCH1 = 011434
 TCH2 = 011436
 TCH3 = 011440
 TCH4 = 011442
 TCH5 = 011444
 TCH6 = 011446
 TCH7 = 011450
 TCNT = 011520
 TCSR = 001266
 TEMP = 011522
 TEST20 = 010102
 TIME = 000000
 TIMER = 016272
 TIMSAV = 016514
 TKVEC = 000060
 TMP = 016124
 TMP0 = 016126
 TMP0E = 016130
 TMP1 = 016132

TMP1E	016134	XOSRV	012626	SERRTB	001254	SLSTTA=	000001	SSWREG	001216
TMP10	016170	X1	016254	SERTTL	001112	SMADR1	001226	SSWRMK=	000000
TMP2	016136	X1SRV	012764	SESCAP	001162	SMADR2	001232	STAGLE=	177777
TMP2E	016140	X2	016256	SETABL	001214	SMADR3	001236	STAGNU=	050417
TMP3	016142	X2SRV	013122	SETEND	001254	SMADR4	001242	STEMP =	000300
TMP3E	016144	X3	016260	SFATAL	001176	SMAIL	001174	STESTN	001200
TMP4	016146	X3SRV	013260	SFFLG	023332	SMAMS1	001224	STIMES	001160
TMP4E	016150	X4	016262	SFILLC	001156	SMAMS2	001230	STKB	001146
TMP5	016152	X4SRV	013416	SFILLS	001155	SMAMS3	001234	STKS	001144
TMP5E	016154	X5	016264	SF\$AND=	000310	SMAMS4	001240	STN =	000023
TMP6	016156	X5SRV	013554	SF\$BAD=	000401	SMADR	001002	STPB	001152
TMP6E	016160	X6	016266	SF\$BLA=	000170	SMFLG	023330	STPFLG	001157
TMP7	016162	X6SRV	013712	SF\$CAS=	000150	SMNEW	023055	STPS	001150
TMP7E	016164	X7	016270	SF\$DEC=	000220	SMSGAD	001210	STRAP	024466
TMP9	001326	X7SRV	014050	SF\$DO =	000340	SMSGLG	001212	STRAP2	024510
TPVEC =	000064	\$APTHD	001000	SF\$FAL=	000405	SMSGTY	001174	STRP =	000012
TRAN	006776	\$ATYC	023112	SF\$G00=	000400	SMSWR	023044	STRPAD	024522
TRAPVE=	000034	\$ATY1	023066	SF\$IF =	000110	SMTYP1	001225	STSK0 =	050413
TRTVEC=	000014	\$ATY3	023074	SF\$INC=	000210	SMTYP2	001231	STSK1 =	050414
TST1	002066	\$ATY4	023104	SF\$LOO=	000200	SMTYP3	001235	STSK2 =	050406
TST10	004012	\$AUTOB	001134	SF\$NAM=	000160	SMTYP4	001241	STSK3 =	050410
TST11	004204	\$BASE	001250	SF\$NO =	000403	SMXCNT	024012	STSK4 =	050333
TST12	004352	\$BDADR	001122	SF\$OR =	000320	\$NESTL=	177777	STSK5 =	177777
TST13	005000	\$BDDAT	001126	SF\$RTI=	000350	\$NSK0 =	000300	STSK6 =	050316
TST14	005356	\$BELL	001164	SF\$RTN=	000300	\$NSK1 =	000110	STSK7 =	050317
TST15	006034	\$BGNLE=	177777	SF\$SEL=	000140	\$NSK2 =	000110	STSTM	001004
TST16	006432	\$BRJMP=	000000	SF\$THE=	000330	\$NSK3 =	000110	STSTNM	001102
TST17	007166	\$CHARC	022276	SF\$TRU=	000404	\$NSK4 =	000110	STTYIN	023022
TST2	002262	\$CKSWR	022302	SF\$UNT=	000130	\$NULL	001154	STYPDS	024014
TST20	010102	\$CMTAG	001100	SF\$WHI=	000120	\$NWTST=	000001	STYPE	021746
TST21	010162	\$CM3 =	000000	SF\$YES=	000402	\$OCNT	024462	STYPEC	022160
TST22	011524	\$CNTLG	023037	\$GDADR	001120	\$OMODE	024464	STYPEX	022300
TST3	002516	\$CNTLU	023032	\$GDDAT	001124	\$OVER	023776	STYPOC	024264
TST4	002730	\$CPUOP	001222	\$GET42	021702	\$PASS	001202	STYPON	024300
TST5	003130	\$CRLF	001171	\$GTSWR	022352	\$PASTM	001006	STYPOS	024240
TST6	003230	\$DBLK	024230	\$HD =	000000	\$QUES	001170	\$U =	000402
TST7	003610	\$DEVCT	001204	\$HIBTS	001000	\$RDCHR	022564	\$UNIT	001206
TYPDS =	104405	\$DEVN	001252	\$ICNT	001104	\$RDLIN	022714	\$UNITM	001010
TYPE =	104401	\$DOAGN	021722	\$IFLEV=	177777	\$RDSZ =	000010	\$USWR	001220
TYPOC =	104402	\$DTBL	024220	\$INTAG	001135	\$RTNAD	021724	\$VECT1	001244
TYPON =	104404	\$ENDAD	021712	\$ISK0 =	000001	\$SAVE =	050314	\$VECT2	001246
TYPOS =	104403	\$ENDCT	021660	\$ISK1 =	000001	\$SAVE2=	050315	\$XOFF =	000023
WAIT	016524	\$ENDMG	021731	\$ISK2 =	000001	\$SAVLE=	177777	\$XON =	000021
WHICHB=	000002	\$ENULL	021726	\$ISK3 =	000001	\$SCOPE	023534	\$XTSTR	023546
XMIT0S	010642	\$ENV	001214	\$ITEMB	001114	\$SELLE=	000000	\$ARGC=	000000
XMIT1S	010722	\$ENVN	001215	\$LF	001172	\$SETUP=	000127	\$BYTE=	000402
XMIT2S	011002	\$EOP	021624	\$LFLG	023331	\$SSK0 =	050337	\$CASE=	000404
XMIT3S	011062	\$EOPCT	021652	\$LO =	177777	\$SSK1 =	000402	\$DST =	000000
XMIT4S	011142	\$ERFLG=	000400	\$LOCTA=	000001	\$SSK2 =	050315	\$ELOC=	000402
XMIT5S	011222	\$ERMAX	001115	\$LPADR	001106	\$STUP =	177777	\$ERFL=	000000
XMIT6S	011302	\$ERRFL	001103	\$LPERR	001110	\$SVLAD	023742	\$FLAG=	000001
XMIT7S	011362	\$ERROR	023334	\$LSK0 =	000000	\$SVPC =	000430	\$FRMB=	000000
XO	016252	\$ERRPC	001116	\$LSTIN=	000001	\$SWR =	167400	\$FROM=	000000

\$\$GET4= 000000	\$50042 003544	\$50127 006430	\$50214 011544	\$50301 016564
\$\$IN = 000002	\$50043 003556	\$50130 006430	\$50215 011564	\$50302 016566
\$\$INH = 000402	\$50044 003572	\$50131 006466	\$50216 011652	\$50303 016600
\$\$LOC = 020272	\$50045 003644	\$50132 006472	\$50217 011716	\$50304 016602
\$\$LOCN= 000000	\$50046 003646	\$50133 006566	\$50220 011750	\$50305 017630
\$\$OUT = 000000	\$50047 003734	\$50134 006574	\$50221 012052	\$50306 017630
\$\$REG = 177777	\$50050 003766	\$50135 006612	\$50222 012102	\$50307 016740
\$\$RETU= 000000	\$50051 003770	\$50136 006630	\$50223 012112	\$50310 017024
\$\$RTN1= 050415	\$50052 004006	\$50137 006770	\$50224 012140	\$50311 017024
\$\$RTN2= 050416	\$50053 004046	\$50140 006766	\$50225 012166	\$50312 017006
\$\$SRC = 000000	\$50054 004050	\$50141 006714	\$50226 012202	\$50313 016770
\$\$TGSV= 050316	\$50055 004136	\$50142 006762	\$50227 012306	\$50314 016752
\$\$TGS1= 050315	\$50056 004154	\$50143 006736	\$50230 012432	\$50315 017110
\$\$TGS2= 050316	\$50057 004200	\$50144 006762	\$50231 012352	\$50316 017114
\$\$TO = 000001	\$50060 004240	\$50145 006760	\$50232 012372	\$50317 017104
\$\$TOB = 000000	\$50061 004242	\$50146 006762	\$50233 012406	\$50320 017230
\$\$TOTL= 000002	\$50062 004330	\$50147 006770	\$50234 012426	\$50321 017230
\$\$TAG= 050000	\$50063 004352	\$50150 007032	\$50235 012460	\$50322 017316
\$OFILL 024463	\$50064 004424	\$50151 007076	\$50236 012476	\$50323 017316
\$50000 002122	\$50065 004414	\$50152 007122	\$50237 012514	\$50324 017342
\$50001 002152	\$50066 004420	\$50153 007162	\$50240 012530	\$50325 017442
\$50002 002310	\$50067 004430	\$50154 007154	\$50241 012620	\$50326 017436
\$50003 002326	\$50070 004516	\$50155 007162	\$50242 012624	\$50327 017630
\$50004 002352	\$50071 004574	\$50156 007222	\$50243 012756	\$50330 017456
\$50005 002404	\$50072 004620	\$50157 007232	\$50244 012762	\$50331 017600
\$50006 002436	\$50073 004646	\$50160 007252	\$50245 013114	\$50332 017564
\$50007 002470	\$50074 004734	\$50161 007252	\$50246 013120	\$50333 017554
\$50010 002516	\$50075 004754	\$50162 007354	\$50247 013252	\$50334 017606
\$50011 002516	\$50076 004772	\$50163 007376	\$50250 013256	\$50335 017612
\$50012 002564	\$50077 005210	\$50164 007416	\$50251 013410	\$50336 017624
\$50013 002616	\$50100 005256	\$50165 007512	\$50252 013414	\$50337 017626
\$50014 002650	\$50101 005412	\$50166 007472	\$50253 013546	\$50340 017762
\$50015 002702	\$50102 005416	\$50167 007466	\$50254 013552	\$50341 017762
\$50016 002730	\$50103 005622	\$50170 007512	\$50255 013704	\$50342 017754
\$50017 002730	\$50104 005670	\$50171 007512	\$50256 013710	\$50343 017752
\$50020 002764	\$50105 005770	\$50172 007536	\$50257 014042	\$50344 020220
\$50021 003016	\$50106 006032	\$50173 007552	\$50260 014046	\$50345 020220
\$50022 003050	\$50107 006034	\$50174 007570	\$50261 016520	\$50346 020062
\$50023 003102	\$50110 006070	\$50175 007606	\$50262 016522	\$50347 020060
\$50024 003130	\$50111 006072	\$50176 007624	\$50263 016312	\$50350 020124
\$50025 003130	\$50112 006116	\$50177 007642	\$50264 016452	\$50351 020200
\$50026 003164	\$50113 006120	\$50200 007732	\$50265 016336	\$50352 020176
\$50027 003230	\$50114 006124	\$50201 010056	\$50266 016344	\$50353 020274
\$50030 003204	\$50115 006126	\$50202 010074	\$50267 016366	\$50354 020274
\$50031 003230	\$50116 006140	\$50203 010146	\$50270 016402	\$50355 020222
\$50032 003256	\$50117 006432	\$50204 010162	\$50271 016414	\$50356 020252
\$50033 003262	\$50120 006216	\$50205 010176	\$50272 016504	\$50357 020536
\$50034 003574	\$50121 006300	\$50206 010230	\$50273 016612	\$50360 020536
\$50035 003356	\$50122 006320	\$50207 010266	\$50274 016612	\$50361 020402
\$50036 003424	\$50123 006376	\$50210 010406	\$50275 016544	\$50362 020400
\$50037 003500	\$50124 006366	\$50211 010402	\$50276 016550	\$50363 020522
\$50040 003500	\$50125 006362	\$50212 010440	\$50277 016560	\$50364 020520
\$50041 003550	\$50126 006372	\$50213 010606	\$50300 016604	\$50365 021322

\$50366	021322	\$50374	020720	\$50402	021134	\$50410	021322	\$50416	021622
\$50367	020606	\$50375	020740	\$50403	021134	\$50411	021432	.	= 024546
\$50370	020740	\$50376	021322	\$50404	021322	\$50412	021432	.\$x	= 001000
\$50371	020732	\$50377	021004	\$50405	021200	\$50413	021416		
\$50372	020670	\$50400	021140	\$50406	021322	\$50414	021416		
\$50373	020676	\$50401	021074	\$50407	021302	\$50415	021622		

. ABS. 024546 000

ERRORS DETECTED: 0

CJDLAA,CJDLAA.PRT/SOL=SPMACJ/ML,CJDLAA.SML,CJDLAA.P11
RUN-TIME: 121 120 .8 SECONDS
RUN-TIME RATIO: 605/242=2.5
CORE USED: 51. (102 PAGES)