

DJ11

EXERCISER & ONLINE TEST
CZDJBF0

AH-8509F-MC

COPYRIGHT 72-78

FICHE 1 OF 1

JAN 1979

digital

MADE IN USA

.REM @

IDENTIFICATION

PRODUCT CODE: AC-8507F-MC
PRODUCT NAME: CZDJBFO DJ11 EXER & ONLINE
PROGRAM DATE: JUNE 1978
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) 1972, 1978 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

CONTENTS

1.	ABSTRACT
2.	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
2.3	PRELIMINARY PROGRAMS
3.	LOADING PROCEDURE
4.	STARTING PROCEDURE
4.1	CONTROL SWITCH SETTINGS
4.2	STARTING ADDRESS
4.3	PROGRAM AND OPERATOR ACTION
5.	OPERATING PROCEDURE
5.1	OPERATIONAL SWITCH SETTINGS
5.2	SUBROUTINE ABSTRACTS
5.3	PROGRAM AND OPERATOR ACTION
6.	ERRORS
6.1	ERROR PRINTOUT
6.2	ERROR RECOVERY
6.3	ERROR COUNTER
7.	RESTRICTIONS
8.	MISCELLANEOUS
8.1	EXECUTION TIME
8.2	STACK POINTER
8.3	PASS COUNTER
8.4	POWER FAIL
9.	PROGRAM DESCRIPTION

1. ABSTRACT

THIS PROGRAM CONSISTS OF THREE SUB-PROGRAMS WHICH EXERCISE THE DJ11 ASYNCHRONOUS MULTIPLEXER. PROGRAM 1 IS AN OFF-LINE EXERCISER. PROGRAM 2 IS AN ON-LINE EXERCISER WHICH CONTINUOUSLY TRANSMITS THE LAST CHARACTER RECEIVED. PROGRAM 3 IS AN ECHO TEST.

NOTE: PROGRAM 1 WILL RUN ANY SILO ALARM LEVEL SETTING.
(FOR PROGRAM 2 AND 3 SEE SECTION 9.)

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 STANDARD COMPUTER WITH CONSOLE TELETYPE
UP TO 16 DJ11 ASYNCHRONOUS MULTIPLEXERS.

2.2 STORAGE

THIS PROGRAM USES ALL OF 8K, EXCEPT ABSOLUTE LOADER.

2.3 PRELIMINARY PROGRAMS

CZDJA DJ11 LOGIC TESTS

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR ABS TAPES.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE 5.1 (ALL DOWN FOR WORST CASE TESTING)

4.2 STARTING ADDRESS

THE PROGRAM SHOULD ALWAYS BE STARTED AT 200. IT MAY BE
RESTARTED AT 1000 AFTER ALL PARAMETERS HAVE BEEN SELECTED.

4.3 PROGRAM AND OPERATOR ACTION

- 1) LOAD PROGRAM INTO MEMORY USING ABS LOADER.
- 2) LOAD ADDRESS 200.
- 3) IF HARDWARE SWITCH REGISTER IS AVAILABLE, SET SWITCHES (SEE SEC. 5.1), ALL DOWN FOR WORST CASE, PRESS START.
- 4) IF SWITCH-LESS PROCESSOR SIMPLY PRESS START.
- 5) ENTER THE PROGRAM NUMBER (1, 2, OR 3).
- 6) SELECT LINES IF SW<8> IS ON A 1.
- 7) PROGRAM 1 WILL LOOP AND BELL WILL RING ONCE EVERY PASS. 'EOP' IS ALSO PRINTED ON EACH PASS.

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

AT SA 200, ALL SWITCHES DOWN IS WORST CASE TESTING. FOR PROGRAM 1 ONLY, THE BELL WILL RING AND EOP IS PRINTED UPON COMPLETION OF A PASS OF THE ENTIRE PROGRAM.

THE SWITCH SETTINGS ARE:

SW<15> - 1 HALT ON ERROR
SW<13> = 1 INHIBIT PRINTOUT
SW<12> - 1 PRINT SILO ALARM LEVEL (PROG1 ONLY)
SW<10> - 1 BELL ON ERROR
 0 BELL ON PASS COMPLETE (PROG1 ONLY)
SW<9> - 1 INHIBIT MAINTENANCE (PROG1 ON-LINE)
SW<8> - 1 SELECT LINES FOR TEST (SEE 5.3)
PROG1 ONLY:
SW<2:0> 0 BINARY COUNT PATTERN
 1 'THE QUICK SILVER GRAY FOX ...'
 2 ALPHA-NUMERIC (40-177)
 3-7 ... NOT USED

THIS PROGRAM HAS BEEN MODIFIED TO RUN ON A PROCESSOR WITH OR WITHOUT A HARDWARE SWITCH REGISTER. WHEN FIRST EXECUTED THE PROGRAM TESTS THE EXISTENCE OF A HARDWARE SWITCH REGISTER. IF NOT FOUND A SOFTWARE SWITCH REGISTER LOCATION (SWREG=LOC. 176) IS DEFAULTED TO. IF THIS IS THE CASE, UPON EXECUTION THE CONTENTS OF THE SWREG ARE DUMPED IN OCTAL ON THE CONSOLE TTY AND ANY CHANGES ARE REQUESTED

(I.E.) SWR=XXXXXX NEW=

POSSIBLE RESPONSES ARE:

1. <CR> IF NO CHANGES ARE TO BE MADE.
2. 6 DIGITS 0-7 TO REPRESENT IN OCTAL THE NEW SWITCH REGISTER VALUE; LAST DIGIT FOLLOWED BY <CR>.
3. ^J TO ALLOW REENTERING VALUE IF ERROR IS COMMITTED KEYING IN SWREG VALUE.

BUILT INTO THE PROGRAM IS THE ABILITY TO DYNAMICALLY CHANGE THE CONTENTS OF SWREG DURING PROGRAM EXECUTION. BY STRIKING ^G (CNTRL G) ON CONSOLE TTY THE OPERATOR SETS A REQUEST FLAG TO CHANGE THE CONTENTS OF SWREG, WHICH IS PROCESSED IN KEY AREAS OF THE PROGRAM CODE (IE) ERROR ROUTINES, AFTER HALTS END OF PASS, AND OTHER APPLICABLE AREAS.

5.2 SUBROUTINE ABSTRACTS

5.2.1 HLT

THIS ROUTINE (CALLED BY AN EMT INSTRUCTION) PRINTS OUT AN ERROR MESSAGE (SEE 6.1). TO INHIBIT TYPEOUTS, PUT SW<13> ON A 1. TO RING THE BELL ON AN ERROR, PUT SW<10> ON A 1.

5.2.2 ALMCK (PROG1 ONLY)

IN THE NORMAL OPERATION THE 'DONE' BIT IS SET AS EACH CHARACTER IS READ INTO THE FI/FO BUFFER(SILO). BUT THIS 'DONE' CONDITION CAN BE DELAYED TO CAUSE DONE ON THE 5, 9, OR 17 CHARACTER. THIS IS DONE BY CUTTING ONE OF THE JUMPERS (W1,W2,W3) ON THE M7285 CONTROL BOARD. THE PROGRAM TESTS FOR THIS 'SILO ALARM LEVEL' AND IF SW12 IS SET (1) IT WILL PRINT OUT THE LEVEL (IN OCTAL) AT WHICH EACH DJ11 IS WILL SET 'DONE'. THE SUBROUTINE ALSO ADJUSTS THE CHARACTER COUNTERS TO ENSURE THAT THE MAXIMUM NUMBER OF CHARACTERS TO BE TRANSFERED IS A MULTIPLE OF THE SILO ALARM LEVEL. THIS ENSURES THAT ALL DATA WILL BE READ OUT OF THE SILO. DONE WILL NOT SET IF THE NUMBER OF CHARACTERS IN THE FI/FO BUFFER IS LESS THAN THE SILO ALARM LEVEL. (NOTE CHARACTER PRESENT IS SET ON EACH CHARACTER IN THE BUFFER, REGARDLESS OF THE SILO ALARM LEVEL.)

5.2.3 TRAPCATCHER

A '..+2' - 'HALT' SEQUENCE IS REPEATED FROM 0 - 56 TO DETECT ANY UNEXPECTED TRAPS AND A '..+2' - 'IOT' SEQUENCE IS REPEATED FROM 60 - 776 TO DETECT ANY UNEXPECTED INTERRUPTS. THUS ANY UNEXPECTED TRAPS WILL HALT AT THE VECTOR + 2. ANY UNEXPECTED INTERRUPTS WILL RESULT IN AN ERROR MESSAGE AND 'HALT' IN 'IOTRAP'.

5.3 PROGRAM AND OPERATOR ACTION

AFTER THE DEVICE PARAMETERS ARE REPORTED, THE PROGRAM TYPES 'PROGRAM #: ' AT WHICH TIME THE OPERATOR ENTERS '1', '2', OR '3' DEPENDING ON THE SUB-PROGRAM HE WISHES TO RUN.

IF SW<8> IS ON A 1, THE PROGRAM WILL TYPE OUT ' N SELECT

LINE\$ ='' THE OPERATOR RESPONDS BY TYPING IN AN OCTAL NUMBER REPRESENTING THE LINE(S) WHICH ARE TO BE TESTED FOR THAT DJ11.(INPUT A 1 FOR LINE 0,A 7 FOR LINES 0,1,AND 2, ETC. THE SAME AS IF YOU WERE DIRECTLY SETTING THE TCR OF THE DJ11.) IF MORE DJ11'S ARE ON THE SYSTEM THE N WILL INDICATE THE NEXT DJ11 AND THE PROMPT IS REISSUED. WHEN ALL LINES ARE SELECTED THE PROGRAM WILL RUN THE SELECTED SUBPROGRAM.

6. ERRORS

6.1 ERROR PRINTOUT

THE FORMAT IS AS FOLLOWS:

ADR (R1) (R2) (R3) (R4)

WHERE:

ADR - ADDRESS OF ERROR HLT
(RN) - CONTENTS OF GENERAL REGISTER 'N'. FROM NONE TO FOUR OF THESE MAY BE TYPED DEPENDING ON THE NUMBER FOLLOWING THE HLT: E.G., HLT+3 WOULD TYPE (R1) THRU (R3); HLT (BY ITSELF) WOULD STOP AFTER TYPING ADR AND DJADR.

TO FIND THE FAILING TEST, LOOK AT THE LISTING ABOVE THE ADDRESS TYPED. IN MOST CASES THE COMMENT BESIDE THE HLT TELLS WHAT WAS BEING CHECKED AND WHAT WAS EXPECTED.

6.2 ERROR RECOVERY

RESTART AT 200 OR 1000.

6.3 ERROR COUNTER

AN ERROR COUNT IS KEPT IN 'ERRORS'. IT CAN BE CLEARED FROM THE CONSOLE, BY RESTARTING AT 200, OR BY RELOADING THE PROGRAM.

7. RESTRICTIONS

THIS PROGRAM REQUIRES THAT THE DEVICE ADDRESSES FOLLOW THE FLOATING ADDRESS CONVENTION (DJ11'S WILL BE FIRST, STARTING AT 160010, THEN THE DH11'S) AND THAT THE VECTOR ADDRESSES ALL BE CONTIGUOUS. IF THE FIRST DJ11 ADDRESS IS NONSTANDARD (OTHER THAN 160010) THEN LOC. 1270 MUST BE CHANGED TO CONTAIN THIS NONSTANDARD ADDRESS.

IF THIS PROGRAM IS RUN WITH A MONITOR, I.E. ACT11 OR DDP,

ONLY PROGRAM 1 IS RUN.

8. MISCELLANEOUS

8.1 EXECUTION TIME (PROG1 ONLY)

DUE TO THE VARIOUS BAUD RATES AVAILABLE AND THE ABILITY TO CHECK UP TO 16 DJ11'S AT ONCE, THE EXECUTION TIME CAN VARY ANYWHERE FROM 3 SECONDS TO NEARLY AN HOUR. THE FOLLOWING TYPICAL TIMES ARE FOR ONE DJ11 WITH ALL LINES AT THE SAME SPEED, 8 LEVEL CODE, 2 STOP BITS, AND NO PARITY ON A PDP-11/20. FOR MULTIPLE DJ11'S, MULTIPLY THESE TIMES BY THE NUMBER OF UNITS SELECTED FOR TEST.

APPROX BAUD RUN TIME

75 00:10:00
110 00:07:00
134.5 00:05:40
150 00:05:00
300 00:02:30
600 00:01:15
1200 00:00:40
1800 00:00:30
2400 00:00:20
4800 00:00:10
9600 00:00:05

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1200

8.3 PASS COUNT (PROG1 ONLY)

A 32 BIT (2 WORDS) PASS COUNT IS KEPT IN 'PCNT'. IT CAN BE CLEARED FROM THE CONSOLE, BY RESTARTING AT 200, OR BY RELOADING THE PROGRAM.

8.4 POWER FAIL

EACH PROGRAM CAN BE POWER FAILED WITH NO ERRORS. TO USE, START THE PROGRAM AS USUAL AND POWER DOWN THEN UP AT ANY TIME. THE ROUTINE SHOULD TYPE 'POWER' AND RESTART THE PROGRAM WITH NO OTHER ERROR TYPEOUTS.

9. PROGRAM DESCRIPTION

THIS PROGRAM CONSISTS OF THREE SUB-PROGRAMS WHICH EXERCISE THE LOGIC OF UP TO 16 DJ11 ASYNCHRONOUS DATA MULTIPLEXERS.

PROGRAM 1: EXERCISER (OFF-LINE)

THIS PROGRAM EXERCISES UP TO 256 LINES (16 DJ11'S) SIMULTANEOUSLY IN MAINTENANCE MODE. THREE DIFFERENT DATA PATTERNS MAY BE SELECTED FROM THE SWITCH REGISTER. THE DATA PATTERN IS REPEATED A MINIMUM OF 16 TIMES FOR EACH PASS. THE PROGRAM SHOULD BE RUN FOR AT LEAST 2 PASSES WITH ALL SWITCHES DOWN. SW<9> ON A ONE DISABLES THE MAINTENANCE MODE, REQUIRING TURN-AROUND CARDS AT THE TERMINATION OF EACH LINE BEING TESTED. (NOTE: THE RECEIVER AND TRANSMIT LINES MUST BE JUMPED FOR THE SAME SPEED.)

PROGRAM 2: CONTINUOUS ECHO EXERCISER (ON-LINE)

THIS PROGRAM CONTINUOUSLY TRANSMITS THE LAST CHARACTER IT RECEIVED ON THE RESPECTIVE LINE. A NULL (000) WILL 'ECHO' 72 TIMES AND THEN TURN OFF THE TRANSMITTER.

PROGRAM 3: ECHO TEST (ON-LINE)

THIS PROGRAM TRANSMITS A HEADING (*ECHO TEST*) ON EACH LINE AND THEN ECHOS EVERYTHING THAT IT RECEIVES.
CAUTION: IF CHARACTERS ARE RECEIVED FASTER THAN THEY CAN BE TRANSMITTED, THE SOFTWARE BUFFERS MAY OVERFLOW.

NOTE: THE ON-LINE EXERCISERS (PROG2 AND PROG3) ARE OPERATOR DEPENDENT, AND THEREFORE DO NOT LOOP. I.E. NO PASSES. ACT11 AND DDP MONITORS WILL ONLY RUN PROC1.

SW	SWITCH	USE
SW15-	100000	:HALT ON ERROR
SW14=	40000	:NOT USED
SW13-	20000	:INHIBIT ERROR TYPEOUTS
SW12=	10000	:PRINT SILO ALARM LEVEL
SW11-	4000	:NOT USED
SW10=	2000	:0 - BELL ON PASS COMPLETE
		:1 - BELL ON ERROR
SW9=	1000	:ON-LINE (PROG1)
SW8-	400	:SELECT LINES (INITIALIZATION *IME ONLY)
:SW<0:2>		SELECT MESSAGE (PROG1 ONLY)
.REM.		

DJ11 REGISTER BIT ASSIGNMENTS:

CONTROL STATUS REGISTER (CSR) XXXXX0

BIT0	RECEIVER ENABLE (READ/WRITE)
BIT1	HALF DUPLEX SELECT (READ/WRITE)
BIT2	MAINTENANCE (READ/WRITE)
BIT3	CLEAR MOS (WRITE ONLY)
BIT4	CLEAR MOS FLAG (READ ONLY)
BIT5	NOT USED
BIT6	RECEIVER INTERRUPT ENABLE (READ/WRITE)
BIT7	DONE (READ ONLY)
BIT8	MASTER TRANSMITTER SCAN ENABLE (READ/WRITE)
BIT9	NOT USED
BIT10	READ/WRITE BREAK REGISTER (READ/WRITE)
BIT11	NOT USED
BIT12	STATUS ENABLE (READ/WRITE)
BIT13	FI/FO OVERRUN (READ ONLY)
BIT14	MASTER TRANSMITTER INTERRUPT ENABLE (READ/WRITE)
BIT15	TRANSMITTER READY (READ ONLY)

RECEIVER BUFFER REGISTER (RBUF) XXXXX2 (READ ONLY)

BIT0-7	RECEIVED CHARACTER
BIT8-11	LINE NUMBER
BIT12	PARITY ERROR
BIT13	FRAMING ERROR
BIT14	UART OVERRUN ERROR
BIT15	CHARACTER PRESENT

TRANSMITTER CONTROL REGISTER (TCR) XXXXX4 (READ/WRITE)

BIT0-15 STOP THE SCANNER ON CORRESPONDING LINE

TRANSMITTER BUFFER (TBUF) XXXXX6

BIT0-7	TRANSMITTED CHARACTER (WRITE ONLY)
BIT8-11	LINE NUMBER (READ ONLY)

BREAK CONTROL STATUS REGISTER (BCSR) XXXXX4 (BIT10 OF CSR SET) (READ/WRITE)

BIT0-15 TRANSMIT A BREAK ON CORRESPONDING LINE.

SCOPE= TRAP
 HLT= EMT
 TYPE= IOT
 PS= 177776
 R0= %0
 R1= %1
 R2= %2
 R3= %3
 R4= %4
 R5= %5
 TTY= %5
 SP= %6
 PC= %7
 BELL= 7
 BIT0= 1
 BIT1= 2
 BIT2= 4
 BIT3= 10
 BIT4= 20
 BIT5= 40
 BIT6= 100
 BIT7= 200
 BIT8= 400
 BIT9= 1000
 BIT10= 2000
 BIT11= 4000
 BIT12= 10000
 BIT13= 20000
 BIT14= 40000
 BIT15= 100000
 LEVEL7 = 340
 OPEN = 0
 STACK 1200

481
 482 000000 000000 000000
 483
 484
 485
 486
 487
 488 000046 000046
 489 000046 014614
 490
 491 000174 000174
 492 000174 000000
 493 000176 000000
 494
 495 000200 000200
 496 000200 000137 006312
 497 000200 001000
 498 001000 000137 007310
 499
 500 001200

. = 0 ; TRAP CATCHER IN LOCATIONS 0 THRU 776
 0,0 ; LOCATIONS 0 AND 2 CONTAIN 'HALT' INSTRUCTIONS
 ; LOCATIONS 4 THRU 56 CONTAIN '+2' AND 'HALT' IN EVERY VECTOR
 ; LOCATIONS 60 THRU 776 CONTAIN '+2' AND 'IOT' IN EVERY VECTOR

. 46
 SENDAD

. 174
 DISPREG 0
 SWREG: 0

. 200
 JMP BEGIN ; 200 ALWAYS IS THE STARTING ADDRESS
 . 1000
 JMP RESTAR ; RESTART ADDRESS

. 1200

501	001200	000000	ICNT: 0	; ITERATION COUNT-LH, TEST NO.-RH
502	001202	000000	ERRORS: 0	; ERROR COUNT REGISTER
503	001204	000000 000000	PCNT: 0,0	; PASS COUNT REGISTER
504				
505	001210	177570	SWR: 177570	
506	001212	177570	DISPLAY: 177570	
507				
508	001214	000000	SAVIT: 0	
509	001216	000020	TIMES: 20	; MINIMUM NUMBER OF MESSAGES (PROG1)
510	001220	000000	SVSW0: OPEN	; MAP OF LINES SELECTED, DJ11 #0
511	001222	000000	SVSW1: OPEN	; MAP OF LINES SELECTED, DJ11 #1
512	001224	000000	SVSW2: OPEN	; MAP OF LINES SELECTED, DJ11 #2
513	001226	000000	SVSW3: OPEN	; MAP OF LINES SELECTED, DJ11 #3
514	001230	000000	SVSW4: OPEN	; MAP OF LINES SELECTED, DJ11 #4
515	001232	000000	SVSW5: OPEN	; MAP OF LINES SELECTED, DJ11 #5
516	001234	000000	SVSW6: OPEN	; MAP OF LINES SELECTED, DJ11 #6
517	001236	000000	SVSW7: OPEN	; MAP OF LINES SELECTED, DJ11 #7
518	001240	000000	SVSW10: OPEN	; MAP OF LINES SELECTED, DJ11 #10
519	001242	000000	SVSW11: OPEN	; MAP OF LINES SELECTED, DJ11 #11
520	001244	000000	SVSW12: OPEN	; MAP OF LINES SELECTED, DJ11 #12
521	001246	000000	SVSW13: OPEN	; MAP OF LINES SELECTED, DJ11 #13
522	001250	000000	SVSW14: OPEN	; MAP OF LINES SELECTED, DJ11 #14
523	001252	000000	SVSW15: OPEN	; MAP OF LINES SELECTED, DJ11 #15
524	001254	000000	SVSW16: OPEN	; MAP OF LINES SELECTED, DJ11 #16
525	001256	000000	SVSW17: OPEN	; MAP OF LINES SELECTED, DJ11 #17
526	001260	000000	MARK: 0	
527	001262	000030	BUFSIZ: 30	; RECEIVE BUFFER SIZE (PROG3)
528	001264	000001	UNITS: 1	; NUMBER OF UNITS ON THE SYSTEM
529	001266	160010	DEVADR: 160010	; FIRST DEVICE ADDRESS
530	001270	000300	VECADR: 300	; FIRST VECTOR ADDRESS
531	001272	000240	RCVLVL: 240	; RECEIVER BR LEVEL = 5
532	001274	000240	XMTLVL: 240	; TRANSMITTER BR LEVEL = 5
533	001276	000000	ISRFLG: 0	; INTR SVC RTN FLAG
534	001300	000000	ALMFLG: 0	; SILO ALARM LEVEL FLAG
535	001302	000000	TIMERA: 0	; TIME COUNTERS
536	001304	000000	TIMERB: 0	
537	001306	000000	COUNT: 0	; VALUE OF THE SILO ALARM LEVEL
538	001310	000000	SUM: 0	
539				
540				
541				
542				
543				
544	001312	000400	XMTTAB: .BLKW 400	; TRANSMIT DATA POINTER TABLE
545				
546	002312	000400	RCVTAB: .BLKW 400	; RECEIVE DATA POINTER TABLE (PROG1)
547				; RECEIVE DATA TABLE (PROG2 AND PROG3)
548				
549	003312	000400	MAXTAB: .BLKW 400	; POINTER FOR XMIT/RCV BFRS.
550				
551	004312	000400	XMTCNT: .BLKW 400	; TRANSMIT DATA COUNTER
552		004313	RCVCNT=XMTCNT+1	; RECEIVE DATA COUNTER
553				
554	005312	000400	MASK: .BLKW 400	; CHARACTER MASK TABLE
555		005313	CNTTAB=MASK+1	; ITERATION COUNT AND FLAGS

::++F

::++F

```
556
557 006312 012706 001200      BEGIN: MOV    #STACK, SP      ;SET UP STACK POINTER
558 006316 004737 016230      JSR     PC, SUSWRR
559 006322 012700 000020      MOV     #20, R0
560 006326 012720 015712      MOV     #IOTRAP, (R0)+ ;IOT VECTOR (20)
561 006332 012720 000340      MOV     #340, (R0)+
562 006336 012720 015550      MOV     #PDOWNS, (R0)+ ;POWER FAIL VECTOR (24)
563 006342 012720 000340      MOV     #340, (R0)+
564 006346 012720 014630      MOV     #EMTS, (R0)+ ;EMT VECTOR (30)
565 006352 012720 000340      MOV     #340, (R0)+
566 006356 005037 001202      CLR     ERRORS      ;CLEAR ERROR COUNTER
567 006362 005037 001204      CLR     PCNT        ;CLEAR PASS COUNTER
568 006366 005037 001206      CLR     PCNT+2
569 006372 012700 000300      MOV     #300, R0    ;START OF FLOATING VECTOR AREA
570 006376 005720      2$: TST     (R0)+      ;UPDATE POINTER
571 006400 010060 177776      MOV     R0, -2(R0)  ;PUT '+2' IN EACH VECTOR
572 006404 012720 000004      MOV     #IOT, (R0)+ ;AND 'IOT'
573 006410 022700 001000      CMP     #1000, R0  ;CHECK FOR END OF FLOATING VECTOR AREA
574 006414 003370      BGT     2$      ;BRANCH IF MORE
575 006416 005037 001300      CLR     ALMFLG     ;CLEAR THE ALARM FLAG
576 006422 012737 000400 011232 MOV     #256., CNTNIT ;SET MAXIMUM SIZE VALUES
577 006430 012737 000106 011236 MOV     #70., CNTNIT+4 ;FOR PROG #1
578 006436 012737 000106 011242 MOV     #70., CNTNIT+10
579 006444 012737 000001 001216 MOV     #1, TIMES    ;SET FOR QV ON FIRST PASS
580
581 ;*****
582 ;ROUTINE TO MAP ALL THE DJ11'S ON THE SYSTEM
583 ;*****
584
585 006452 013700 001266      DJMAP: MOV     DEVADR, R0    ;GET FIRST FLOATING ADDRESS
586 006456 012702 000001      MOV     #1, R2      ;COUNTER FOR DJ11'S
587 006462 012737 000002 000006 MOV     #RTI, @#6    ;RTI WHEN TIME-OUT
588 006470 005001      5$: CLR     R1      ;SET UP COUNTER
589 006472 000261      1$: SEC     ;SET CARRY
590 006474 005710      TST     (R0)      ;CHECK FOR A DEVICE
591 006476 103404      BCS     7$      ;BRANCH IF NONE
592 006500 062700 000010      6$: ADD     #10, R0    ;GO TO NEXT DEVICE ADDRESS
593 006504 005201      INC     R1      ;COUNT DJ11'S
594 006506 000771      BR      1$      ;LOOK FOR MORE
595
596 006510 005037 000006      7$: CLR     @#6      ;RESTORE TIMEOUT VECTOR
597 006514 010137 001264      MOV     R1, UNITS ;SAVE COUNT
598 006520 001005      BNE     GETVEC
599 006522 000004 016624      TYPE, MSG01 ;TYPE 'NO DJ11'S.'
600 006526 000000      HALT    ;FATAL ERROR
601 006530 000137 014542      JMP     @#DONE ;RESTART
602
603 ;*****
604 ;ROUTINE TO DETERMINE VECTOR ADDRESSES OF DJ11'S
605 ;*****
606
607 006534 013746 000020      GETVEC: MOV     @#20, -(SP) ;SAVE IOT VECTOR
608 006540 012737 006570 000020 MOV     #1$, @#20 ;RESET IOT VECTOR
609 006546 013701 001266      MOV     DEVADR, R1 ;FIRST DJ ADDRESS
610 006552 012711 040400      MOV     #40400, (R1) ;SET CSR
611 ;BIT8= TRANS SCAN ENABLE
```

```
612
613 006556 012761 000001 000004      MOV    #1,      4(R1)      ;BIT14= TRANS INTERRUPT ENABLE
614 006564 000001                      WAIT                      ;TCR, LINE 0
615 006566 000407                      BR      3$           ;WAIT FOR AN INTERRUPT
616                                     ;CONTINUE AFTER INTERUPT
617 006570 011602                      1$:  MOV    (SP),    R2      ;SAVE VECTOR ADR. (+4)
618 006572 162716 000010                SUB    #10,    (SP)      ;REPOSITION ADR TO RCV. VEC.
619 006576 011637 001270                MOV    (SP),    VECADR    ;SAVE FIRST VECTOR
620 006602 022626                      CMP    (SP)+,   (SP)+    ;RESET STACK FROM IOT
621 006604 000002                      RTI                      ;RETURN FROM INITIAL INTERUPT
622
623 006606 012637 000020                3$:  MOV    (SP)+,   @#20    ;RESTORE IOT VECTOR
624 006612 005742                      TST    -(R2)      ;POINT TO XMT. VEC. +2
625 006614 013703 001264                MOV    UNITS,   R3      ;SET UP UNIT COUNTER
626
627                                     ;CHECK THAT VECTORS ARE CONTIGUOUS
628
629 006620 005061 000004                2$:  CLR    4(R1)      ;CLEAR TCR
630 006624 005011                      CLR    (R1)        ;CLEAR CSR
631 006626 012712 000004                MOV    #IOT,   (R2)    ;RESTORE IOT TO XMT. VEC.+2
632 006632 005303                      DEC    R3          ;CHECK FOR MORE DJ11'S
633 006634 001415                      BEQ    REPORT    ;BRANCH IF DONE
634 006636 062701 000010                ADD    #10,    R1      ;UPDATE DJ ADR. POINTER
635 006642 062702 000010                ADD    #10,    R2      ;UPDATE VECTOR POINTER
636 006646 012712 000002                MOV    #RTI,   (R2)    ;RTI ON INTERRUPT
637 006652 012711 040400                MOV    #40400, (R1)   ;SET CSR
638 006656 012761 000001 000004                MOV    #1,      4(R1)   ;TCR LINE 0
639 006664 000001                      WAIT                      ;WAIT FOR AN INTERRUPT
640 006666 000754                      BR      2$
641
642                                     ;REPORT CONFIGURATION
643
644 006670 032777 020000 172312 REPORT: BIT    #BIT13, @SWR    ;CHECK FOR INHIBIT TYP0UT
645 006676 001026                      BNE                      ;SKIP REPORT IF SET
646 006700 000004 016445                TYPE,   MSGMDN
647 006704 000004 016434                TYPE,   RETURN
648 006710 000004 016506                TYPE,   MSGADR
649 006714 013705 001266                MOV    DEVADR,TTY ;TYPE DEVADR IN OCTAL
650 006720 004737 015366                JSR    PC,PRINTR  ;TYPE LEADING ZERO'S
651 006724 000004 016536                TYPE,   MSGVEC
652 006730 013705 001270                MOV    VECADR,TTY ;TYPE VECADR IN OCTAL
653 006734 004737 015376                JSR    PC,PRINTS  ;AND SUPRESS LEADING ZERO'S
654 006740 000004 016562                TYPE,   MSGNUM
655 006744 013705 001264                MOV    UNITS,TTY  ;TYPE UNITS IN OCTAL
656 006750 004737 015376                JSR    PC,PRINTS  ;AND SUPRESS LEADING ZERO'S
```

```
657
658
659
660
661
662 006754 022737 000176 001210 GETLEN: CMP #SWREG,SWR
663 006762 001002 BNE 6$
664 006764 004737 016126 JSR PC,CNTLU
665 006770 013700 001264 6$: MOV UNITS, R0 ;SET UP UNIT COUNTER
666 006774 013701 001266 MOV DEVADR, R1 ;SET UP DEVICE ADDRESS POINTER
667 007000 012702 000001 1$: MOV #1, R2 ;SET UP LINE MARKER
668 007004 012711 000415 MOV #415, (R1) ;RCV ENB, CMOS, MAINT., TRANS SCAN ENB
669 007010 032711 000020 10$: BIT #BIT4, (R1) ;WAIT FOR MOS TO CLEAR
670 007014 001375 BNE 10$
671 007016 010261 000004 2$: MOV R2, 4(R1) ;TRANS CONTROL, ONE LINE AT A TIME
672 007022 005711 3$: TST (R1) ;WAIT FOR TRANS READY
673 007024 100376 BPL 3$
674 007026 012761 000377 000006 MOV #377, 6(R1) ;SEND A RUBOUT
675 007034 006302 ASL R2 ;SKIP 4 LINES
676 007036 006302 ASL R2
677 007040 006302 ASL R2
678 007042 006302 ASL R2
679 007044 103364 BCC 2$ ;BRANCH BACK IF MORE LINES
680 007046 005061 000004 CLR 4(R1) ;CLEAR TCR
681 007052 062701 000010 ADD #10, R1 ;UPDATE POINTER TO NEXT UNIT
682 007056 005300 DEC R0 ;CHECK FOR MORE UNITS
683 007060 001347 BNE 1$
684 007062 013700 001264 MOV UNITS, R0 ;SET UP UNIT COUNTER
685 007066 013701 001266 MOV DEVADR, R1 ;SET UP DEVICE ADDRESS POINTER
686 007072 012702 005312 MOV #MASK, R2 ;SET UP CHAR LEN TABLE POINTER
687 007076 012703 000004 4$: MOV #4, R3 ;SET UP CHAR COUNTER
688 007102 016104 000002 5$: MOV 2(R1), R4 ;SAVE AND CHECK CHAR PRESENT
689 007106 100375 BPL 5$
690 007110 010405 MOV R4, R5 ;DUP DATA
691 007112 000305 SWAB R5 ;LINE # IN LOW BYTE
692 007114 042705 177760 BIC #177760,R5 ;CLEAR ALL BUT LINE #
693 007120 006305 ASL R5 ;*2
694 007122 060205 ADD R2, R5 ;MAKE POINTER TO CHAR TABLE
695 007124 105104 COMB R4 ;MAKE DATA INTO MASK
696 007126 042704 177400 BIC #177400,R4 ;CLEAR UPPER BYTE
697 007132 010425 MOV R4, (R5)+ ;SAVE THE MASK
698 007134 010425 MOV R4, (R5)+ ;SAVE THE MASK
699 007136 010425 MOV R4, (R5)+ ;SAVE THE MASK
700 007140 010425 MOV R4, (R5)+ ;SAVE THE MASK
701 007142 005303 DEC R3 ;COUNT TO 4
702 007144 001356 BNE 5$
703 007146 062701 000010 ADD #10, R1 ;ADDRESS POINTER TO NEXT DJ
704 007152 062702 000040 ADD #40, R2 ;CHAR LEN TABLE POINTER
705 007156 005300 DEC R0 ;COUNT UNITS
706 007160 001346 BNE 4$ ;BRANCH BACK IF MORE
```

```
707
708
709
710
711
712
713
714
715 007162 005737 000042
716 007166 001040
717 007170 000004 016605
718 007174 004537 015022
719 007200 007332
720 007202 001367
721 007204 032777 000400 171776
722 007212 001426
723 007214 005000
724 007216 012701 001220
725 007222 000004 016434
726 007226 000004 016411
727 007232 010005
728 007234 004737 015376
729 007240 000004 016414
730 007244 004537 015022
731 007250 001214
732 007252 013721 001214
733 007256 005200
734 007260 020037 001264
735 007264 001356
736 007266 000410
737
738 007270 013700 001264
739 007274 012701 001220
740 007300 012721 177777
741 007304 005300
742 007306 001374
743
744
745 007310 013700 007332
746 007314 001722
747 007316 022700 000003
748 007322 103717
749 007324 006300
750 007326 000170 007332
751
752 007332
753 007332 000001
754 007334 007342
755 007336 012004
756 007340 013210

:*****
:SELECT THE PROGRAM TO BE RUN
:PROGRAM 1: OFF-LINE EXERCISER
:PROGRAM 2: ON-LINE EXERCISER (TRANSMIT LAST CHARACTER RECEIVED)
:PROGRAM 3: ON-LINE ECHO EXERCISER
:*****

SELPRO: TST @#42 ;CHECK FOR ACT 11 OR DDP
        BNE ALL ;BRANCH IF MONITOR
        TYPE, MSGPRG
        JSR R5, READIN ;READ A NUMBER FROM THE CTY
        .WORD PROGNO
        BNE SELPRO
        BIT #BIT8, @SWR ;CHECK FOR SW<8>, SELECT LINES
        BEQ ALL ;BRANCH IF NOT
        CLR R0 ;SET UP UNIT COUNTER, DISPLAY
        MOV #SVSW0,R1 ;SET UP SWITCH TABLE POINTER
SWITCH: TYPE, RETURN
        TYPE, MNUM
        MOV R0,TTY ;PRINT THE NUMBER OF THE DR11C
        JSR PC,PRINTS
        TYPE, MSGSEL ;ASL FOR THE SELECTED LINES
        JSR R5,READIN
        .WORD SAVIT
        MOV SAVIT,(R1)+
        INC R0 ;COUNT UNITS
        CMP R0, UNITS ;CHECK FOR MORE UNITS
        BNE SWITCH ;BRANCH IF MORE
        BR RESTAR ;GO DO IT

ALL: MOV UNITS, R0 ;SET UP UNIT COUNTER
     MOV #SVSW0,R1 ;SET UP SWITCH TABLE POINTER
1$: MOV #177777,(R1)+ ;SET ALL LINES
    DEC R0 ;COUNT UNITS
    BNE 1$ ;BRANCH IF MORE

.SBTTL RESTAR POINT

RESTAR: MOV PROGNO, R0
        BEQ SELPRO
        CMP #3, R0
        BLO SELPRO
        ASL R0
        JMP @PROGAD (R0)

PROGNO.
PROGAD: 1 ;DEFAULT TO PROGRAM 1
        PROG1
        PROG2
        PROG3
```

```
757
758
759
760
761
762
763
764
765
766
767 007342 000005
768 007344 005037 001276
769 007350 012706 001200
770 007354 052737 000340 177776
771 007362 012701 001312
772 007366 012702 002000
773 007372 005021
774 007374 005302
775 007376 001375
776 007400 012702 000400
777 007404 005201
778 007406 105021
779 007410 005302
780 007412 001374
781
782
783
784
785
786
787
788 007414 005000
789 007416 013701 001266
790 007422 013702 001270
791 007426 012703 010372
792 007432 010322
793 007434 013722 001272
794 007440 022323
795 007442 010113
796 007444 062723 000002
797 007450 005723
798 007452 010322
799 007454 013722 001274
800 007460 022323
801 007462 010123
802 007464 005011
803 007466 052711 050510
804
805
806
807
808
809 007472 032711 000020
810 007476 001375
811 007500 005737 001300
812 007504 001002

;*****
;PROGRAM 1: TRANSMIT AND RECEIVE ALL LINES SIMULTANEOUSLY
;OFF-LINE: SW<9> = 0
;ON-LINE: SW<9> = 1
;USES THE DATA TABLE SELECTED BY SW<2:0>
;EACH LINE REPEATS THE PATTERN AT LEAST 16 TIMES
;PER PASS.
;*****

PROG1: RESET ;CLEAR OUT THE WORLD
CLR ISRFLG ;CLEAR INTR. SVC. RTN. FLAG. ;:++F
MOV #STACK, SP ;RESET THE STACK POINTER
BIS #340, #PS ;PROCESSOR TO LEVEL 7
MOV #XMTTAB, R1 ;FIRST TABLE POINTER
MOV #2000, R2 ;LENGTH OF TABLES (WORDS) ;:++F
1$: CLR (R1)+ ;CLEAR THE TABLE
DEC R2
BNE 1$
MOV #400, R2 ;LENGTH OF MASK/COUNT TABLE
2$: INC R1 ;SKIP MASK
CLRB (R1)+ ;CLEAR COUNT
DEC R2
BNE 2$

;ROUTINE TO INITIALIZE ALL DJ11'S AND THEIR ISR'S:
;SET UP ALL INTERRUPT VECTORS
;SET UP DEVICE ADDRESSES IN LINKER ROUTINES
;SET CSR'S EVERYTHING ENABLED
;SET TCR'S, ALL LINES ENABLED

P1INIT: CLR R0
MOV DEVADR, R1
MOV VECADR, R2
MOV #RISRO, R3
11$: MOV R3, (R2)+ ;SET UP RECEIVER INTERRUPT VECTOR
MOV RCVLVL, (R2)+
CMP (R3)+, (R3)+ ;ADD 4 TO R3
MOV R1, (R3) ;ADDRESS OF CSR
ADD #2, (R3)+ ;ADDRESS OF RBUF
TST (R3)+
MOV R3, (R2)+ ;SET UP TRANSMITTER INTERRUPT VECTOR
MOV XMTLVL, (R2)+
CMP (R3)+, (R3)+
MOV R1, (R3)+ ;ADDRESS OF CSR
CLR (R1) ;CLEAR CSR
BIS #50510, (R1) ;SET UP CSR
;BIT3 = CLEAR MOS
;BIT6 = RECEIVER INTERRUPT ENABLE
;BIT8 = TRANSMITTER SCAN ENABLE
;BIT12 = STATUS ENABLE
;BIT14 = TRANSMITTER INTERRUPT ENABLE
13$: BJT #BIT4, (R1) ;CHECK FOR MOS TO CLEAR
BNE 13$
TST ALMFLG ;HAS THE SILO ALARM LEVEL BEEN CHECKED?
BNF 10$ ;YES
```

```
813 007506 004737 010024      JSR    PC,ALMCK      ;NO,GO DO IT
814 007512 006300      ASL    R0          ;UNIT # * 2
815 007514 016061 001220 000004 10$:  MOV    SVSW0(R0),4(R1) ;SET TCR BITS (CSR + 4)
816 007522 006200      ASR    R0          ;RESTORE UNIT COUNTER
817 007524 012737 000001 001260  MOV    #1, MARK      ;SET UP MARKER
818 007532 017705 171452      MOV    @SWR, P5      ;GET SWITCH SETTINGS
819 007536 042705 177770      BIC    #177770,R5      ;MASK MESSAGE #
820 007542 006305      ASL    R5
821 007544 006305      ASL    R5
822 007546 012304      MOV    (R3)+, R4      ;SET UP OFFSET TO TABLES
823 007550 033761 001260 000004 14$:  BIT    MARK, 4(R1) ;CHECK FOR LINE SELECTED IN TCR
824 007556 001414      BEQ    15$
825 007560 016564 011230 001312      MOV    ADRNIT(5),XMTTAB(4)
826 007566 016564 011230 002312      MOV    ADRNIT(5),RCVTAB(4)
827 007574 116564 011232 004312      MOV    CNTNIT(5),XMTCNT(4)
828 007602 116564 011232 004313      MOV    CNTNIT(5),RCVCNT(4)
829 007610 005724      5$:  TST    (R4)+      ;INC OFFSET TO NEXT LINE
830 007612 006337 001260      ASL    MARK
831 007616 103354      BCC    14$
832 007620 032777 001000 171362      BIT    #BIT9, @SWR      ;CHECK FOR ON-LINE
833 007626 001024      BNE    21$      ;BRANCH IF ON-LINE
834 007630 052711 000014      BIS    #14, (R1)      ;SET THE MAINTENANCE BIT AND CLR MOS
835 007634 032711 000020      20$:  BIT    #BIT4,(R1)      ;WAIT FOR MOS TO CLEAR
836 007640 001375      BNE    20$
837 007642 052711 000001      BIS    #1,(R1)      ;TURN ON RCV ENABLE
838 007646 062701 000010      12$:  ADD    #10,R1      ;POINT TO THE NEXT CSR
839 007652 005200      INC    R0
840 007654 020037 001264      CMP    R0, UNITS
841 007660 001264      BNE    11$
842 007662 012737 000001 001300      MOV    #1,ALMFLG      ;SET THE ALARM LEVEL FLAG
843 007670 042737 000140 177776      BIC    #140, @#PS      ;LOWER PROCESSOR PRIORITY
844 007676 000406      BR     FORGND      ;GO DO IT
845
846 007700 052711 000001      21$:  BIS    #1,(R1)      ;TURN ON RCV EN
847 007704 005761 000002      22$:  TST    2(R1)      ;CLEAR JUNK OUT OF THE RBUF
848 007710 100775      BMI    22$
849 007712 000755      BR     12$
850
851
852
853
854
855
856
```

```
*****
;PROG1 BACKGROUND PROGRAM TO MONITOR TABLES
*****
; NOTE - PROGRAM MAY HANG IN A LOOP.
; IF THIS HAPPENS, RUN DZDJA.
```

```
857 007714 012701 004312      FORGND: MOV    #XMTCNT,R1
858 007720 012702 000400      MOV    #400,R2
859 007724 105711      21$:  TSTB    (R1)      ;CHECK FOR COUNT TABLE CLR
860 007726 001376      BNE    21$      ;BRANCH IF NOT
861 007730 062701 000002      ADD    #2,R1      ;GO TO NEXT LINE ENTRY
862 007734 005302      DEC    R2      ;COUNT LINES
863 007736 001372      BNE    21$      ;BRANCH IF MORE LINES
864 007740 012701 004313      MOV    #RCVCNT,R1
865 007744 012702 000400      MOV    #400,R2
866 007750 121137 001306      22$:  CMPB    (R1),COUNT ;IS # OF CHAR LEFT IN RBUF LESS THAN
867                                     ;THE SILO ALARM LEVEL
868 007754 003375      BGT    22$      ;IF NO WAIT FOR THEM
```

869	007756	062701	000002		ADD	#2,P1	;IF YES IGNORE THEM	
870	007762	005302			DEC	R2	;COUNT LINES	
871	007764	001371			BNE	22\$	;BRANCH IF MORE LINES	
872	007766	005337	001216		DEC	TIMES	;DO THIS AGAIN?	
873	007772	001402			BEQ	23\$	;NO, GET OUT	
874	007774	000137	007342		JMP	PROG1		
875	010000	005737	001276	23\$:	TST	ISRFLG	;SEE IF FLAG IS ZERO	::++F
876	010004	001002			BNE	1\$	;IF NOT, BR	::++F
877	010006	000004	016360		TYPE,	TRNERR	;IF YES, RCV DATA ERROR	::++F
878	010012	012737	000020	001216	MOV	#20,TIMES	;DO IT 16 TIMES THE NEXT TIME	
879	010020	000137	014542		JMP	@DONE	;SKIP ISR'S	
880	010024	012761	000001	000004	MOV	#1,4(R1)	;SET LINE 0 IN THE TCR	
881	010032	052711	000004		BIS	#BIT2,(R1)	;SET THE MAINT BIT	
882	010036	052711	000001		BIS	#1,(R1)	;SET RCV EN AFTER MAINTENANCE BIT	
883	010042	005037	001306		CLR	COUNT		
884	010046	005037	001310		CLR	SUM		
885	010052	005037	001302		CLR	TIMERA		
886	010056	012737	000200	001304	MOV	#200,TIMERB	;SET UP TIME CONSTANTS	
887	010064	005711			TST	(R1)	;WAIT FOR TRANSFER READY BIT	
888	010066	100373			BPL	2\$		
889	010070	112761	000377	000006	MOV	#377,6(R1)	;OUTPUT A CHAR TO TBUF	
890	010076	005237	001306		INC	COUNT	;COUNT EACH CHAR	
891	010102	105711			TST	(R1)	;CHECK FOR DONE IN THE CSR	
892	010104	100405			BMI	3\$	;IF SET GET OUT OF THE LOOP	
893	010106	004537	010336		JSR	R5,TIME	;GIVE DONE TIME TO SET	
894	010112	000773			BR	1\$	;RETURN TO TEST FOR DONE AGAIN	
895	010114	000760			BR	2\$	;RETURN TO OUTPUT ANOTHER CHAR	
896	010116	000742			BR	ALMCK	;ERROR RETURN TRY AGAIN	
897	010120	042711	000001		BIC	#BIT0,(R1)	;TURN OFF RCV ENABLE	
898	010124	052711	000010		BIS	#BIT3,(R1)	;CLEAR MOS	
899	010130	022737	000001	001306	CMP	#1,COUNT	;IF SILO LEVEL SET FOR 1 THEN GET OUT	
900	010136	001437			BEQ	4\$		
901	010140	063737	001306	001310	ADD	COUNT,SUM	;GET THE LARGEST MULTIPLE OF THE	
902	010146	023727	001310	000106	CMP	SUM,#70.	;SILO ALARM LEVEL AND USE IT IN THE	
903	010154	002771			BLT	5\$	;THREE SIZE LOCATIONS	
904	010156	001403			BEQ	6\$	;IF EQUAL USE IT	
905	010160	163737	001306	001310	SUB	COUNT,SUM	;IF GREATER SUBTRACT ONE COUNT OFF	
906	010166	013737	001310	011236	MOV	SUM,CNTNIT+4	;AS CLOSE TO 70 AS POSSIBLE	
907	010174	013737	001310	011242	MOV	SUM,CNTNIT+10		
908	010202	063737	001306	001310	ADD	COUNT,SUM	;CONTINUE TO A COUNT OF 256	
909	010210	023727	001310	000377	CMP	SUM,#377		
910	010216	002771			BLT	7\$		
911	010220	163737	001306	001310	SUB	COUNT,SUM		
912	010226	013737	001310	011232	MOV	SUM,CNTNIT	;AS CLOSE TO 256 AS POSSIBLE	
913	010234	000411			BR	11\$	;ALL SET GET OUT	
914	010236	012737	000377	011232	MOV	#377,CNTNIT	;PUT SIZE TO MAX VALUE	
915	010244	012737	000106	011236	MOV	#70.,CNTNIT+4		
916	010252	012737	000106	011242	MOV	#70.,CNTNIT+10		
917	010260	042711	000004		BIC	#BIT2,(R1)	;TURN OFF THE MAINT BIT	
918	010264	032711	000020		BIT	#BIT4,(R1)	;WAIT FOR MOS TO CLEAR	
919	010270	001375			BNE	12\$		
920	010272	004737	016310		JSR	PC,KBDINT	;GET THE SWITCH REGISTER	
921	010276	032777	010000	170704	BIT	#SW12,@SWR	;PRINT ALARM LEVEL?	
922	010304	001413			BEQ	13\$	;NO	
923	010306	000004	016733		TYPE,	M.A. ARM		
924	010312	010105			MOV	R1,TTY	;YES, PRINT CSR FIRST	

```

925 010314 004737 015366 JSR PC,PRINTR
926 010320 000004 016727 TYPE, MSGDAS
927 010324 013705 001306 MOV COUNT,TTY ;PRINT ALARM LEVEL
928 010330 004737 015366 JSR PC,PRINTR
929 010334 000207 13$: RTS PC
930
931
932
933 010336 105237 001302 TIME: INCB TIMERA ;INCREMENT THROUGH ONE WORD
934 010342 001012 BNE 1$ ;GO TEST FOR DONE AGAIN
935 010344 005337 001304 DEC TIMERB ;MAKE TIMERB LARGER IF FAST PROCESSOR
936 010350 001007 BNE 1$
937 010352 023727 001306 000022 CMP COUNT,#22 ;HAVE OUTPUTTED 18 TIMES
938 010360 001002 BNE 2$ ;NO, GO OUTPUT ANOTHER CHAR
939 010362 104001 HLT+1 ;YES,DONE DID NOT SET AFTER 18 OUTPUTS
940 ;R1 - CSR
941 010364 005725 TST (R5)+ ;SET R5 FOR ERROR RETURN
942 010366 005725 2$: TST (R5)+ ;SET R5 FOR NEXT OUTPUT RETURN
943 010370 000205 1$: RTS R5 ;RETURN FROM ABOVE OR RETEST DONE
944
945
946
947
948

```

```

*****
;PROG1 LINKERS TO DJ11 INTERRUPT SERVICE ROUTINES
*****

```

```

949 010372 004037 011106 RISR0: JSR R0,RCVISR
950 010376 160012 000000 .WORD <160012+<0*10>>,<40*0>
951 010402 004037 010772 XISR0: JSR R0,XMTISR
952 010406 160010 000000 .WORD <160010+<0*10>>,<40*0>
953 010412 004037 011106 RISR1: JSR R0,RCVISR
954 010416 160022 000040 .WORD <160012+<1*10>>,<40*1>
955 010422 004037 010772 XISR1: JSR R0,XMTISR
956 010426 160020 000040 .WORD <160010+<1*10>>,<40*1>
957 010432 004037 011106 RISR2: JSR R0,RCVISR
958 010436 160032 000100 .WORD <160012+<2*10>>,<40*2>
959 010442 004037 010772 XISR2: JSR R0,XMTISR
960 010446 160030 000100 .WORD <160010+<2*10>>,<40*2>
961 010452 004037 011106 RISR3: JSR R0,RCVISR
962 010456 160042 000140 .WORD <160012+<3*10>>,<40*3>
963 010462 004037 010772 XISR3: JSR R0,XMTISR
964 010466 160040 000140 .WORD <160010+<3*10>>,<40*3>
965 010472 004037 011106 RISR4: JSR R0,RCVISR
966 010476 160052 000200 .WORD <160012+<4*10>>,<40*4>
967 010502 004037 010772 XISR4: JSR R0,XMTISR
968 010506 160050 000200 .WORD <160010+<4*10>>,<40*4>
969 010512 004037 011106 RISR5: JSR R0,RCVISR
970 010516 160062 000240 .WORD <160012+<5*10>>,<40*5>
971 010522 004037 010772 XISR5: JSR R0,XMTISR
972 010526 160060 000240 .WORD <160010+<5*10>>,<40*5>
973 010532 004037 011106 RISR6: JSR R0,RCVISR
974 010536 160072 000300 .WORD <160012+<6*10>>,<40*6>
975 010542 004037 010772 XISR6: JSR R0,XMTISR
976 010546 160070 000300 .WORD <160010+<6*10>>,<40*6>
977 010552 004037 011106 RISR7: JSR R0,RCVISR
978 010556 160102 000340 .WORD <160012+<7*10>>,<40*7>
979 010562 004037 010772 XISR7: JSR R0,XMTISR
980 010566 160100 000340 .WORD <160010+<7*10>>,<40*7>

```

981 010572 004037 011106
982 010576 160112 000400
983 010602 004037 010772
984 010606 160110 000400
985 010612 004037 011106
986 010616 160122 000440
987 010622 004037 010772
988 010626 160120 000440
989 010632 004037 011106
990 010636 160132 000500
991 010642 004037 010772
992 010646 160130 000500
993 010652 004037 011106
994 010656 160142 000540
995 010662 004037 010772
996 010666 160140 000540
997 010672 004037 011106
998 010676 160152 000600
999 010702 004037 010772
1000 010706 160150 000600
1001 010712 004037 011106
1002 010716 160162 000640
1003 010722 004037 010772
1004 010726 160160 000640
1005 010732 004037 011106
1006 010736 160172 000700
1007 010742 004037 010772
1008 010746 160170 000700
1009 010752 004037 011106
1010 010756 160202 000740
1011 010762 004037 010772
1012 010766 160200 000740
1013
1014
1015
1016
1017
1018 010772
1019 010772 010146
1020 010774 010246
1021 010776 012001
1022 011000 005711
1023 011002 100035
1024 011004 116102 000007
1025 011010 006302
1026 011012 061002
1027 011014 105762 004312
1028 011020 001410
1029 011022 117261 001312 000006
1030 011030 105362 004312
1031 011034 005262 001312
1032 011040 000757
1033 011042 010346
1034 011044 005062 001312
1035 011050 161002
1036 011052 006202

RISR10: JSR R0,RCVISR
.WORD <160012+<10*10>>,<40*10>
XISR10: JSR R0,XMTISR
.WORD <160010+<10*10>>,<40*10>
RISR11: JSR R0,RCVISR
.WORD <160012+<11*10>>,<40*11>
XISR11: JSR R0,XMTISR
.WORD <160010+<11*10>>,<40*11>
RISR12: JSR R0,RCVISR
.WORD <160012+<12*10>>,<40*12>
XISR12: JSR R0,XMTISR
.WORD <160010+<12*10>>,<40*12>
RISR13: JSR R0,RCVISR
.WORD <160012+<13*10>>,<40*13>
XISR13: JSR R0,XMTISR
.WORD <160010+<13*10>>,<40*13>
RISR14: JSR R0,RCVISR
.WORD <160012+<14*10>>,<40*14>
XISR14: JSR R0,XMTISR
.WORD <160010+<14*10>>,<40*14>
RISR15: JSR R0,RCVISR
.WORD <160012+<15*10>>,<40*15>
XISR15: JSR R0,XMTISR
.WORD <160010+<15*10>>,<40*15>
RISR16: JSR R0,RCVISR
.WORD <160012+<16*10>>,<40*16>
XISR16: JSR R0,XMTISR
.WORD <160010+<16*10>>,<40*16>
RISR17: JSR R0,RCVISR
.WORD <160012+<17*10>>,<40*17>
XISR17: JSR R0,XMTISR
.WORD <160010+<17*10>>,<40*17>

;PROG1 TRANSMITTER INTERRUPT SERVICE ROUTINE

XMTISR:
MOV R1,-(6) ;PUSH R1 ON STACK
MOV R2,-(6) ;PUSH R2 ON STACK
MOV (R0)+,R1
'\$: TST (R1) ;CHECK FOR TRANS READY
BPL 4\$
MOVB 7(R1),R2 ;GET LINE NO.
ASL R2
ADD (R0),R2
TSTB XMTCNT(2) ;TST FOR ZERO
BEQ 2\$
MOVB @XMTTAB(2),6(1) ;SEND A CHARACTER
DECB XMTCNT(2) ;COUNT CHARACTERS
INC XMTTAB(2) ;UPDATE TABLE POINTER
BR 1\$
2\$: MOV R3,-(SP)
CLR XMTTAB(2) ;CLEAR TABLE POINTER
SUB (R0),R2
ASR R2

```
1037 011054 005003          CLR      R3
1038 011056 000261          SEC
1039 011060 006103      3$:  ROL      R3
1040 011062 005302          DEC      R2
1041 011064 100375          BPL      3$
1042 011066 040361 000004    BIC      R3,4(R1)      ;CLEAR TCR BIT FOR LINE
1043 011072 012603          MOV      (SP)+,R3      ;RESTORE R3
1044 011074 000741          BR       1$
1045 011076          4$:
1046 011076 012602          MOV      (6)+,R2      ;POP STACK INTO R2
1047 011100 012601          MOV      (6)+,R1      ;POP STACK INTO R1
1048 011102 012600          MOV      (6)+,R0      ;POP STACK INTO R0
1049 011104 000002          RTI
1050
1051          ;*****
1052          ;PROG1 RECEIVER INTERRUPT SERVICE ROUTINE
1053          ;*****
1054
1055 011106          RCVISR:
1056 011106 010146          MOV      R1,-(6)      ;PUSH R1 ON STACK
1057 011110 010246          MOV      R2,-(6)      ;PUSH R2 ON STACK
1058 011112 010346          MOV      R3,-(6)      ;PUSH R3 ON STACK
1059 011114 010446          MOV      R4,-(6)      ;PUSH R4 ON STACK
1060 011116 012001          MOV      (R0)+,R1      ;GET RBUF ADDRESS
1061 011120 011102      1$:  MOV      (R1),R2      ;READ THE DATA
1062 011122 100032          BPL      7$      ;BRANCH IF NO CHAR PRESENT
1063 011124 032702 070000    BIT      #70000,R2      ;CHECK FOR ERRORS
1064 011130 001403          BEQ      2$      ;BRANCH IF OK
1065 011132 104002          HLT+2      ;RECEIVER ERROR
1066          ;R1-RBUF ADDRESS
1067          ;R2=CONTENTS OF RBUF
1068          ;BIT12=PARITY ERROR
1069          ;BIT13=FRAMING ERROR
1070          ;BIT14=UART OVERRUN
1071 011134 042702 070000    BIC      #70000, R2      ;CLEAR ERROR BITS FOR SPEED
1072 011140 010204      2$:  MOV      R2, R4      ;DUP THE RBUF
1073 011142 105004          CLRB     R4      ;CLEAR THE DATA
1074 011144 000304          SWAB     R4      ;LINE # TO LOW BYTE
1075 011146 106304          ASLB     R4      ;LINE # * 2, ALSO CLR CHAR PRESENT
1076 011150 061004          ADD      (R0),R4      ;ADD OFFSET
1077 011152 117403 002312    MOV      @RCVTAB(R4),R3      ;GET EXPECTED DATA
1078 011156 046403 005312    BIC      MASK(4),R3      ;MASK CHARACTER LENGTH
1079 011162 120302          CMPB     R3,R2
1080 011164 001403          BEQ      3$      ;BRANCH IF OK
1081 011166 042703 177400    BIC      #177400,R3      ;MAKE SURE UPPER BYTE CLEAR
1082 011172 104003          HLT+3      ;DATA ERROR
1083          ;R1-RBUF ADDRESS
1084          ;R2=CONTENTS OF RBUF (DATA)
1085          ;R3 EXPECTED DATA
1086 011174 105364 004313      3$:  DECB     RCVcnt(4)
1087 011200 001403          BEQ      7$
1088 011202 005264 002312    INC      RCVTAB(4)      ;UPDATE TABLE POINTER
1089 011206 000744          BR       1$      ;CONTINUE
1090 011210
1091 011210 012604          7$:  MOV      (6)+,R4      ;POP STACK INTO R4
1092 011212 012603          MOV      (6)+,R3      ;POP STACK INTO R3
```

```

1093 011214 012602      MOV      (6)+,R2      ;POP STACK INTO R2
1094 011216 012601      MOV      (6)+,R1      ;POP STACK INTO R1
1095 011220 012600      MOV      (6)+,R0      ;POP STACK INTO R0
1096 011222 005237 001276 INC      ISRFLG      ;STEP INT SVC RTN FLAG      ;:++F
1097 011226 000002      RTI
1098
1099      ;*****
1100      ;PROG1 DATA TABLES
1101      ;*****
1102
1103 011230 011274      ADRNIT: BINARY      ;SW<2:0>=0 BINARY COUNT PATTERN
1104 011232 000377      CNTNIT: 377      ;SIZE=256.
1105 011234 011674      PHRASE      ;SW<2:0>=1 'THE QUICK SILVER GRAY FOX...'
1106 011236 000106      70.      ;SIZE=70.
1107 011240 011266      SIXBIT      ;SW<2:0>=2 040 THRU 137
1108 011242 000106      70.      ;SIZE=70.
1109 011244 016770      END
1110 011246 000001      1
1111 011250 017370      END+400
1112 011252 000001      1
1113 011254 017770      END+1000
1114 011256 000001      1
1115 011260 020370      END+1400
1116 011262 000001      1
1117 011264 020770      END+2000
1118 011266 005015 177777 177777 SIXBIT: .ASCII <15><12><377><377><377><377> ;CR-LF, FILLERS
1119 011274 040      BINARY: .BYTE 40
1120 011275 041      .BYTE 41
1121 011276 042      .BYTE 42
1122 011277 043      .BYTE 43
1123 011300 044      .BYTE 44
1124 011301 045      .BYTE 45
1125 011302 046      .BYTE 46
1126 011303 047      .BYTE 47
1127 011304 050      .BYTE 50
1128 011305 051      .BYTE 51
1129 011306 052      .BYTE 52
1130 011307 053      .BYTE 53
1131 011310 054      .BYTE 54
1132 011311 055      .BYTE 55
1133 011312 056      .BYTE 56
1134 011313 057      .BYTE 57
1135 011314 060      .BYTE 60
1136 011315 061      .BYTE 61
1137 011316 062      .BYTE 62
1138 011317 063      .BYTE 63
1139 011320 064      .BYTE 64
1140 011321 065      .BYTE 65
1141 011322 066      .BYTE 66
1142 011323 067      .BYTE 67
1143 011324 070      .BYTE 70
1144 011325 071      .BYTE 71
1145 011326 072      .BYTE 72
1146 011327 073      .BYTE 73
1147 011330 074      .BYTE 74
1148 011331 075      .BYTE 75

```

1149	011332	076	.BYTE	76
1150	011333	077	.BYTE	77
1151	011334	100	.BYTE	100
1152	011335	101	.BYTE	101
1153	011336	102	.BYTE	102
1154	011337	103	.BYTE	103
1155	011340	104	.BYTE	104
1156	011341	105	.BYTE	105
1157	011342	106	.BYTE	106
1158	011343	107	.BYTE	107
1159	011344	110	.BYTE	110
1160	011345	111	.BYTE	111
1161	011346	112	.BYTE	112
1162	011347	113	.BYTE	113
1163	011350	114	.BYTE	114
1164	011351	115	.BYTE	115
1165	011352	116	.BYTE	116
1166	011353	117	.BYTE	117
1167	011354	120	.BYTE	120
1168	011355	121	.BYTE	121
1169	011356	122	.BYTE	122
1170	011357	123	.BYTE	123
1171	011360	124	.BYTE	124
1172	011361	125	.BYTE	125
1173	011362	126	.BYTE	126
1174	011363	127	.BYTE	127
1175	011364	130	.BYTE	130
1176	011365	131	.BYTE	131
1177	011366	132	.BYTE	132
1178	011367	133	.BYTE	133
1179	011370	134	.BYTE	134
1180	011371	135	.BYTE	135
1181	011372	136	.BYTE	136
1182	011373	137	.BYTE	137
1183	011374	140	.BYTE	140
1184	011375	141	.BYTE	141
1185	011376	142	.BYTE	142
1186	011377	143	.BYTE	143
1187	011400	144	.BYTE	144
1188	011401	145	.BYTE	145
1189	011402	146	.BYTE	146
1190	011403	147	.BYTE	147
1191	011404	150	.BYTE	150
1192	011405	151	.BYTE	151
1193	011406	152	.BYTE	152
1194	011407	153	.BYTE	153
1195	011410	154	.BYTE	154
1196	011411	155	.BYTE	155
1197	011412	156	.BYTE	156
1198	011413	157	.BYTE	157
1199	011414	160	.BYTE	160
1200	011415	161	.BYTE	161
1201	011416	162	.BYTE	162
1202	011417	163	.BYTE	163
1203	011420	164	.BYTE	164
1204	011421	165	.BYTE	165

1205	011422	166	.BYTE	166
1206	011423	167	.BYTE	167
1207	011424	170	.BYTE	170
1208	011425	171	.BYTE	171
1209	011426	172	.BYTE	172
1210	011427	173	.BYTE	173
1211	011430	174	.BYTE	174
1212	011431	175	.BYTE	175
1213	011432	176	.BYTE	176
1214	011433	177	.BYTE	177
1215	011434	200	.BYTE	200
1216	011435	201	.BYTE	201
1217	011436	202	.BYTE	202
1218	011437	203	.BYTE	203
1219	011440	204	.BYTE	204
1220	011441	205	.BYTE	205
1221	011442	206	.BYTE	206
1222	011443	207	.BYTE	207
1223	011444	210	.BYTE	210
1224	011445	211	.BYTE	211
1225	011446	212	.BYTE	212
1226	011447	213	.BYTE	213
1227	011450	214	.BYTE	214
1228	011451	215	.BYTE	215
1229	011452	216	.BYTE	216
1230	011453	217	.BYTE	217
1231	011454	220	.BYTE	220
1232	011455	221	.BYTE	221
1233	011456	222	.BYTE	222
1234	011457	223	.BYTE	223
1235	011460	224	.BYTE	224
1236	011461	225	.BYTE	225
1237	011462	226	.BYTE	226
1238	011463	227	.BYTE	227
1239	011464	230	.BYTE	230
1240	011465	231	.BYTE	231
1241	011466	232	.BYTE	232
1242	011467	233	.BYTE	233
1243	011470	234	.BYTE	234
1244	011471	235	.BYTE	235
1245	011472	236	.BYTE	236
1246	011473	237	.BYTE	237
1247	011474	240	.BYTE	240
1248	011475	241	.BYTE	241
1249	011476	242	.BYTE	242
1250	011477	243	.BYTE	243
1251	011500	244	.BYTE	244
1252	011501	245	.BYTE	245
1253	011502	246	.BYTE	246
1254	011503	247	.BYTE	247
1255	011504	250	.BYTE	250
1256	011505	251	.BYTE	251
1257	011506	252	.BYTE	252
1258	011507	253	.BYTE	253
1259	011510	254	.BYTE	254
1260	011511	255	.BYTE	255

1261	011512	256	.BYTE	256
1262	011513	257	.BYTE	257
1263	011514	260	.BYTE	260
1264	011515	261	.BYTE	261
1265	011516	262	.BYTE	262
1266	011517	263	.BYTE	263
1267	011520	264	.BYTE	264
1268	011521	265	.BYTE	265
1269	011522	266	.BYTE	266
1270	011523	267	.BYTE	267
1271	011524	270	.BYTE	270
1272	011525	271	.BYTE	271
1273	011526	272	.BYTE	272
1274	011527	273	.BYTE	273
1275	011530	274	.BYTE	274
1276	011531	275	.BYTE	275
1277	011532	276	.BYTE	276
1278	011533	277	.BYTE	277
1279	011534	300	.BYTE	300
1280	011535	301	.BYTE	301
1281	011536	302	.BYTE	302
1282	011537	303	.BYTE	303
1283	011540	304	.BYTE	304
1284	011541	305	.BYTE	305
1285	011542	306	.BYTE	306
1286	011543	307	.BYTE	307
1287	011544	310	.BYTE	310
1288	011545	311	.BYTE	311
1289	011546	312	.BYTE	312
1290	011547	313	.BYTE	313
1291	011550	314	.BYTE	314
1292	011551	315	.BYTE	315
1293	011552	316	.BYTE	316
1294	011553	317	.BYTE	317
1295	011554	320	.BYTE	320
1296	011555	321	.BYTE	321
1297	011556	322	.BYTE	322
1298	011557	323	.BYTE	323
1299	011560	324	.BYTE	324
1300	011561	325	.BYTE	325
1301	011562	326	.BYTE	326
1302	011563	327	.BYTE	327
1303	011564	330	.BYTE	330
1304	011565	331	.BYTE	331
1305	011566	332	.BYTE	332
1306	011567	333	.BYTE	333
1307	011570	334	.BYTE	334
1308	011571	335	.BYTE	335
1309	011572	336	.BYTE	336
1310	011573	337	.BYTE	337
1311	011574	340	.BYTE	340
1312	011575	341	.BYTE	341
1313	011576	342	.BYTE	342
1314	011577	343	.BYTE	343
1315	011600	344	.BYTE	344
1316	011601	345	.BYTE	345

1317	011602	346	.BYTE	346
1318	011603	347	.BYTE	347
1319	011604	350	.BYTE	350
1320	011605	351	.BYTE	351
1321	011606	352	.BYTE	352
1322	011607	353	.BYTE	353
1323	011610	354	.BYTE	354
1324	011611	355	.BYTE	355
1325	011612	356	.BYTE	356
1326	011613	357	.BYTE	357
1327	011614	360	.BYTE	360
1328	011615	361	.BYTE	361
1329	011616	362	.BYTE	362
1330	011617	363	.BYTE	363
1331	011620	364	.BYTE	364
1332	011621	365	.BYTE	365
1333	011622	366	.BYTE	366
1334	011623	367	.BYTE	367
1335	011624	370	.BYTE	370
1336	011625	371	.BYTE	371
1337	011626	372	.BYTE	372
1338	011627	373	.BYTE	373
1339	011630	374	.BYTE	374
1340	011631	375	.BYTE	375
1341	011632	376	.BYTE	376
1342	011633	377	.BYTE	377
1343	011634	000	.BYTE	0
1344	011635	001	.BYTE	1
1345	011636	002	.BYTE	2
1346	011637	003	.BYTE	3
1347	011640	004	.BYTE	4
1348	011641	005	.BYTE	5
1349	011642	006	.BYTE	6
1350	011643	007	.BYTE	7
1351	011644	010	.BYTE	10
1352	011645	011	.BYTE	11
1353	011646	012	.BYTE	12
1354	011647	013	.BYTE	13
1355	011650	014	.BYTE	14
1356	011651	015	.BYTE	15
1357	011652	016	.BYTE	16
1358	011653	017	.BYTE	17
1359	011654	020	.BYTE	20
1360	011655	021	.BYTE	21
1361	011656	022	.BYTE	22
1362	011657	023	.BYTE	23
1363	011660	024	.BYTE	24
1364	011661	025	.BYTE	25
1365	011662	026	.BYTE	26
1366	011663	027	.BYTE	27
1367	011664	030	.BYTE	30
1368	011665	031	.BYTE	31
1369	011666	032	.BYTE	32
1370	011667	033	.BYTE	33
1371	011670	034	.BYTE	34
1372	011671	035	.BYTE	35

CZDJBFO DJ11 EXER & ONLINE
CZDJBFO.P11 21-JUN-78 13:56

MACY11 30A(1052) 21-JUN-78 14:01
DATA TABLES

8 3
PAGE 28

SEQ 0027

1373 011672 036
1374 011673 037

.BYTE 36
.BYTE 37

1375
1376 011674 005015 177777 177777
1377 011702 044124 020105 052521
1378 011710 041511 020113 044523
1379 011716 053114 051105 043440
1380 011724 040522 020131 047506
1381 011732 020130 052512 050115
1382 011740 042105 047440 042526
1383 011746 020122 026071 033470
1384 011754 026066 032065 026063
1385 011762 030462 027060 020060
1386 011770 040514 054532 042040
1387 011776 043517 020523 000
1388 012004

PHRASE: .ASCII <15><12><377><377><377><377>
.ASCIIZ 'THE QUICK SILVER GRAY FOX JUMPED OVER 9,876,543,210.0 LAZY DOGS!'

.EVEN

1389

1390

1391

1392

1393

1394

1395

1396

1397 012004 000005

1398 012006 012706 001200

1399 012012 052737 000340 177776

1400 012020 012701 001312

1401 012024 012702 002000

1402 012030 005021

1403 012032 005302

1404 012034 001375

1405 012036 012702 000400

1406 012042 005201

1407 012044 105021

1408 012046 005302

1409 012050 001374

1410

1411

1412

1413

1414

1415

1416

1417 012052 005000

1418 012054 013701 001266

1419 012060 013702 001270

1420 012064 012703 012242

1421 012070 010322

1422 012072 013722 001272

1423 012076 022323

1424 012100 010113

1425 012102 062723 000002

1426 012106 005723

1427 012110 010322

1428 012112 013722 001274

1429 012116 022323

1430 012120 010123

1431 012122 012721 050501

1432

1433

1434

1435

1436

1437 012126 005721

1438 012130 006300

1439 012132 016011 001220

1440 012136 006200

1441 012140 012737 000001 001260

1442 012146 012304

1443 012150 033711 001260

1444 012154 001406

```
*****
PROGRAM 2:  ON-LINE MULTI-ECHO EXERCISER
             TRANSMITS THE LAST CHARACTER RECEIVED ON ITS RESPECTIVE
             LINE.  A CARRIAGE RETURN AND LINE FEED ARE INSERTED
             EVERY 72 CHARACTERS.
*****
```

```
PROG2:  RESET                ;CLEAR OUT THE WORLD
        MOV     #STACK, SP    ;RESET THE STACK POINTER
        BIS     #340, @#PS    ;PROCESSOR TO LEVEL 7
        MOV     #XMTTAB, R1   ;FIRST TABLE POINTER
        MOV     #2000, R2     ;LENGTH OF TABLES (WORDS)
1$:      CLR     (R1)+         ;CLEAR THE TABLE
        DEC     R2
        BNE     1$
        MOV     #400, R2      ;LENGTH OF MASK/COUNT TABLE
2$:      INC     R1            ;SKIP MASK
        CLRB    (R1)+         ;CLEAR COUNT
        DEC     R2
        BNE     2$
```

```
;ROUTINE TO INITIALIZE ALL DJ11'S AND THEIR ISR'S:
;SET UP ALL INTERRUPT VECTORS
;SET UP DEVICE ADDRESSES IN LINKER ROUTINES
;SET CSR'S EVERYTHING ENABLED
;SET TCR'S, ALL LINES ENABLED
```

```
P2INIT: CLR     R0
        MOV     DEVADR, R1
        MOV     VECADR, R2
        MOV     #R2SR0, R3    ;SET UP POINTER TO LINKERS
1$:      MOV     R3, (R2)+     ;SET UP RECEIVER INTERRUPT VECTOR
        MOV     RCVI VL, (R2)+
        CMP     (R3)+, (R3)+  ;ADD 4 TO R3
        MOV     R1, (R3)      ;ADDRESS OF CSR
        ADD     #2, (R3)+     ;ADDRESS OF RBUF
        TST     (R3)+
        MOV     R3, (R2)+     ;SET UP TRANSMITTER INTERRUPT VECTOR
        MOV     XMTLVL, (R2)+
        CMP     (R3)+, (R3)+
        MOV     R1, (R3)+     ;ADDRESS OF CSR
        MOV     #50501, (R1)+ ;SET UP CSR
                                ;BIT0 = RECEIVER ENABLE
                                ;BIT6 = RECEIVER INTERRUPT ENABLE
                                ;BIT8 = TRANSMITTER SCAN ENABLE
                                ;BIT12 = STATUS ENABLE
                                ;BIT14 = TRANSMITTER INTERRUPT ENABLE
        TST     (R1)+
        ASL     R0
        MOV     SVSW0(R0), (R1) ;UNIT # * 2
        ASR     R0            ;SET TCR BITS FOR SELECTED LINES
        MOV     #1, MARK     ;RESET UNIT COUNTER
        MOV     (R3)+, R4    ;SET UP MARKER
4$:      BIT     MARK, (R1)   ;SET UP OFFSET TO TABLES
        BEQ     5$          ;CHECK FOR LINE SELECTED
```

CZDJBF0 DJ11 EXER & ONLINE
CZDJBF.P11 21-JUN-78 13:56

MACY11 30A(1052) 21-JUN-78 14:01 PAGE 30
PROG2: ON-LINE EXERCISER (TRANSMIT LAST CHARACTER RECEIVED)

SEQ 0029

```
1445 012156 012764 016641 001312      MOV    #MSGP2, XMTTAB(4) ;SET UP XMTR TABLE
1446 012164 012764 000045 004312      MOV    #45, XMTCNT(4) ;SET UP COUNT
1447 012172 005724          5$:      TST    (R4)+          ;INC OFFSET TO NEXT LINE
1448 012174 006337 001260          ASL    MARK
1449 012200 103363          BCC    4$
1450 012202 022121          CMP    (R1)+, (R1)+ ;ADD 4
1451 012204 005200          INC    R0
1452 012206 020037 001264          CMP    R0, UNITS
1453 012212 001326          BNE    1$
1454 012214 042737 000140 177776      BIC    #140, @#PS ;LOWER PROCESSOR PRIORITY
1455
```

```
;*****
;PROG2 FOREGROUND PROGRAM TO READ/WRITE MEMORY
;*****
```

```
1460 012222 012700 020000      FORP2: MOV    #20000,R0 ;TOP OF 4K BANK OF MEMORY
1461 012226 000241          CLC
1462 012230 005540          1$:      ADC    -(R0) ;FAST READ/WRITE TO MEMORY
1463 012232 001376          BNE    1$ ;RAPID REPEAT
1464 012234 005700          TST    R0 ;CHECK FOR LOC 0
1465 012236 001374          BNE    1$ ;BRANCH IF MORE MEMORY
1466 012240 000770          BR     FORP2 ;LOOP FOR EVER!
```

```
;*****
;PROG2 LINKERS TO DJ11 INTERRUPT SERVICE ROUTINES
;*****
```

```
1472 012242 004037 013034      R2SR0: JSR    R0,P2RISR
1473 012246 160012 000000          .WORD    <160012+<0*10>>,<0*40>
1474 012252 004037 012642      X2SR0: JSR    R0,P2XISR
1475 012256 160020 000000          .WORD    <160020+<0*10>>,<0*40>
1476 012262 004037 013034      R2SR1: JSR    R0,P2RISR
1477 012266 160022 000040          .WORD    <160012+<1*10>>,<1*40>
1478 012272 004037 012642      X2SR1: JSR    R0,P2XISR
1479 012276 160030 000040          .WORD    <160020+<1*10>>,<1*40>
1480 012302 004037 013034      R2SR2: JSR    R0,P2RISR
1481 012306 160032 000100          .WORD    <160012+<2*10>>,<2*40>
1482 012312 004037 012642      X2SR2: JSR    R0,P2XISR
1483 012316 160040 000100          .WORD    <160020+<2*10>>,<2*40>
1484 012322 004037 013034      R2SR3: JSR    R0,P2RISR
1485 012326 160042 000140          .WORD    <160012+<3*10>>,<3*40>
1486 012332 004037 012642      X2SR3: JSR    R0,P2XISR
1487 012336 160050 000140          .WORD    <160020+<3*10>>,<3*40>
1488 012342 004037 013034      R2SR4: JSR    R0,P2RISR
1489 012346 160052 000200          .WORD    <160012+<4*10>>,<4*40>
1490 012352 004037 012642      X2SR4: JSR    R0,P2XISR
1491 012356 160060 000200          .WORD    <160020+<4*10>>,<4*40>
1492 012362 004037 013034      R2SR5: JSR    R0,P2RISR
1493 012366 160062 000240          .WORD    <160012+<5*10>>,<5*40>
1494 012372 004037 012642      X2SR5: JSR    R0,P2XISR
1495 012376 160070 000240          .WORD    <160020+<5*10>>,<5*40>
1496 012402 004037 013034      R2SR6: JSR    R0,P2RISR
1497 012406 160072 000300          .WORD    <160012+<6*10>>,<6*40>
1498 012412 004037 012642      X2SR6: JSR    R0,P2XISR
1499 012416 160100 000300          .WORD    <160020+<6*10>>,<6*40>
1500 012422 004037 013034      R2SR7: JSR    R0,P2RISR
```

1501 012426 160102 000340
1502 012432 004037 012642
1503 012436 160110 000340
1504 012442 004037 013034
1505 012446 160112 000400
1506 012452 004037 012642
1507 012456 160120 000400
1508 012462 004037 013034
1509 012466 160122 000440
1510 012472 004037 012642
1511 012476 160130 000440
1512 012502 004037 013034
1513 012506 160132 000500
1514 012512 004037 012642
1515 012516 160140 000500
1516 012522 004037 013034
1517 012526 160142 000540
1518 012532 004037 012642
1519 012536 160150 000540
1520 012542 004037 013034
1521 012546 160152 000600
1522 012552 004037 012642
1523 012556 160160 000600
1524 012562 004037 013034
1525 012566 160162 000640
1526 012572 004037 012642
1527 012576 160170 000640
1528 012602 004037 013034
1529 012606 160172 000700
1530 012612 004037 012642
1531 012616 160200 000700
1532 012622 004037 013034
1533 012626 160202 000740
1534 012632 004037 012642
1535 012636 160210 000740

.WORD <160012+<7*10>>,<7*40>
X2SR7: JSR R0,P2XISR
.WORD <160020+<7*10>>,<7*40>
R2SR10: JSR R0,P2RISR
.WORD <160012+<10*10>>,<10*40>
X2SR10: JSR R0,P2XISR
.WORD <160020+<10*10>>,<10*40>
R2SR11: JSR R0,P2RISR
.WORD <160012+<11*10>>,<11*40>
X2SR11: JSR R0,P2XISR
.WORD <160020+<11*10>>,<11*40>
R2SR12: JSR R0,P2RISR
.WORD <160012+<12*10>>,<12*40>
X2SR12: JSR R0,P2XISR
.WORD <160020+<12*10>>,<12*40>
R2SR13: JSR R0,P2RISR
.WORD <160012+<13*10>>,<13*40>
X2SR13: JSR R0,P2XISR
.WORD <160020+<13*10>>,<13*40>
R2SR14: JSR R0,P2RISR
.WORD <160012+<14*10>>,<14*40>
X2SR14: JSR R0,P2XISR
.WORD <160020+<14*10>>,<14*40>
R2SR15: JSR R0,P2RISR
.WORD <160012+<15*10>>,<15*40>
X2SR15: JSR R0,P2XISR
.WORD <160020+<15*10>>,<15*40>
R2SR16: JSR R0,P2RISR
.WORD <160012+<16*10>>,<16*40>
X2SR16: JSR R0,P2XISR
.WORD <160020+<16*10>>,<16*40>
R2SR17: JSR R0,P2RISR
.WORD <160012+<17*10>>,<17*40>
X2SR17: JSR R0,P2XISR
.WORD <160020+<17*10>>,<17*40>

;PROG2 TRANSMITTER INTERRUPT SERVICE ROUTINE

1536
1537
1538
1539
1540
1541 012642
1542 012642 010146
1543 012644 010246
1544 012646 012001
1545 012650 005711
1546 012652 100064
1547 012654 116102 000007
1548 012660 006302
1549 012662 061002
1550 012664 105762 004312
1551 012670 001413
1552 012672 117261 001312 000006
1553 012700 105362 004312
1554 012704 105762 005313
1555 012710 001357
1556 012712 005262 001312

P2XISR:
MOV R1,-(6) ;PUSH R1 ON STACK
MOV R2,-(6) ;PUSH R2 ON STACK
MOV (R0)+,R1
1\$: TST (R1) ;CHECK FOR TRANS READY
BPL 4\$
MOVB 7(R1),R2 ;GET LINE NO.
ASL R2
ADD (R0),R2
TSTB XMT CNT(2) ;TST FOR ZERO
BEQ 2\$;GET OUT
MOVB @XMTTAB(2),6(R1) ;SEND A CHARACTER
DECB XMT CNT(2) ;COUNT CHARACTERS
TSTB CNTTAB(2) ;CHECK FOR MESSAGE OR DATA
BNE 1\$;BRANCH IF DATA
INC XMTTAB(2) ;UPDATE TABLE POINTER

```
1557 012716 000754 BR 1$
1558 012720 105162 005313 2$: COMB CNTTAB(2) ;CHANGE FLAG
1559 012724 001430 BEQ 3$ ;BRANCH IF WAS DATA
1560 012726 012762 002312 001312 MOV #RCVTAB,XMTTAB(2) ;SET UP POINTER TO RECEIVER TABLE
1561 012734 060262 001312 ADD R2,XMTTAB(2) ;ADD OFFSET
1562 012740 112762 000110 004312 MOVB #72,XMTCNT(2) ;COUNT 72. CHARACTERS TO THE LINE
1563 012746 105762 002312 TSTB RCVTAB(2) ;CHECK FOR A BREAK
1564 012752 001336 BNE 1$ ;BRANCH IF REAL DATA
1565 012754 161002 SUB (R0), R2 ;RECOVER LINE NUMBER
1566 012756 006202 ASR R2
1567 012760 005037 001260 CLR MARK ;SET UP MARKER
1568 012764 000261 SEC
1569 012766 006137 001260 5$: ROL MARK ;MOVE MARKER
1570 012772 005302 DEC R2 ;COUNT LINES
1571 012774 100374 BPL 5$ ;BRANCH IF MORE
1572 012776 043761 001260 000004 BIC MARK, 4(R1) ;CLEAR TCR BIT
1573 013004 000721 BR 1$ ;CONTINUE
1574 013006 012762 016434 001312 3$: MOV #RETURN,XMTTAB(2) ;TYPE CARRIAGE RETURN, LINE FEED
1575 013014 112762 000002 004312 MOVB #2,XMTCNT(2) ;COUNTER OF 2 CHARACTERS
1576 013022 000712 BR 1$
1577 013024 4$:
1578 013024 012602 MOV (6)+,R2 ;POP STACK INTO R2
1579 013026 012601 MOV (6)+,R1 ;POP STACK INTO R1
1580 013030 012600 MOV (6)+,R0 ;POP STACK INTO R0
1581 013032 000002 RTI
```

```
1582
1583 ;*****
1584 ;PROG2 RECEIVER INTERRUPT SERVICE ROUTINE
1585 ;*****
1586
```

```
1587 013034 P2RISR:
1588 013034 010146 MOV R1,-(6) ;PUSH R1 ON STACK
1589 013036 010246 MOV R2,-(6) ;PUSH R2 ON STACK
1590 013040 010346 MOV R3,-(6) ;PUSH R3 ON STACK
1591 013042 012001 MOV (R0)+,R1 ;GET RBUF ADDRESS
1592 013044 011102 1$: MOV (R1),R2 ;READ THE DATA
1593 013046 100053 BPL 7$ ;BRANCH IF NO CHAR PRESENT
1594 013050 032702 070000 BIT #70000,R2 ;CHECK FOR ERRORS
1595 013054 001402 BEQ 2$ ;BRANCH IF OK
1596 013056 104002 HLT+2 ;RECEIVER ERROR
1597 ;R1=RBUF ADDRESS
1598 ;R2=CONTENTS OF RBUF
1599 ;BIT12=PARITY ERROR
1600 ;BIT13=FRAMING ERROR
1601 ;BIT14=UART OVERRUN
1602 013060 000771 BR 1$ ;FORGET THE DATA
1603
1604 013062 010203 2$: MOV R2,R3 ;DUP THE RBUF
1605 013064 105003 CLRB R3 ;CLEAR THE DATA
1606 013066 000303 SWAB R3 ;LINE # TO LOW BYTE
1607 013070 106303 ASLB R3 ;LINE # * 2, ALSO CLR CHAR PRESENT
1608 013072 061003 ADD (R0),R3 ;ADD OFFSET
1609 013074 136302 005312 BITB MASK(3),R2 ;CHECK CHARACTER LENGTH
1610 013100 001401 BEQ 3$ ;BRANCH IF OK
1611 013102 104002 HLT+2 ;CHARACTER LENGTH ERROR
1612 ;R1=RBUF ADDRESS
```

```

1613
1614 013104 105763 002312 3$: TSTB RCVTAB(3) ;R2=CONTENTS OF RBUF(DATA)
1615 013110 001017 BNE 5$ ;CHECK FOR BREAK
1616 013112 110263 002312 MOVB R2, RCVTAB(3) ;BRANCH IF REAL DATA
1617 013116 161003 SUB (R0), R3 ;SAVE THE DATA
1618 013120 006203 ASR R3 ;RECOVER LINE NUMBER
1619 013122 005037 001260 CLR MARK ;SET UP MARKER
1620 013126 000261 SEC
1621 013130 006137 001260 4$: ROL MARK ;UPDATE MARKER
1622 013134 005303 DEC R3 ;COUNT LINES
1623 013136 100374 BPL 4$ ;BRANCH IF MORE
1624 013140 053761 001260 000002 BIS MARK, 2(R1) ;SET TCR BIT
1625 013146 000736 BR 1$ ;CONTINUE
1626
1627 013150 110263 002312 5$: MOVB R2, RCVTAB(3) ;SAVE THE DATA
1628 013154 105163 005313 COMB CNTTAB(3) ;SET MESSAGE FLAG
1629 013160 012763 016434 001312 MOV #RETURN, XMTTAB(3) ;TYPE CARRIAGE RETURN, LINE FEED
1630 013166 112763 000002 004312 MOVB #2, XMTCNT(3) ;MESSAGE LENGTH
1631 013174 000723 BR 1$
1632
1633 013176 012603 7$: MOV (6)+, R3 ;POP STACK INTO R3
1634 013200 012602 MOV (6)+, R2 ;POP STACK INTO R2
1635 013202 012601 MOV (6)+, R1 ;POP STACK INTO R1
1636 013204 012600 MOV (6)+, R0 ;POP STACK INTO R0
1637 013206 000002 RTI
  
```

```

1638
1639
1640
1641
1642
1643 013210 000005
1644 013212 012706 001200
1645 013216 052737 000340 177776
1646 013224 012701 001312
1647 013230 012702 002000
1648 013234 005021
1649 013236 005302
1650 013240 001375
1651 013242 012702 000400
1652 013246 005201
1653 013250 105021
1654 013252 005302
1655 013254 001374
1656 013256 012705 016770
1657
1658
1659
1660
1661
1662
1663
1664 013262 005000
1665 013264 013701 001266
1666 013270 013702 001270
1667 013274 012703 013562
1668 013300 010322
1669 013302 013722 001272
1670 013306 022323
1671 013310 010113
1672 013312 062723 000002
1673 013316 005723
1674 013320 010322
1675 013322 013722 001274
1676 013326 022323
1677 013330 010123
1678 013332 012723 050400
1679
1680
1681
1682 013336 005721
1683 013340 006300
1684 013342 016011 001220
1685 013346 006200
1686 013350 012737 000001 001260
1687 013356 012304
1688 013360 010246
1689 013362 010346
1690 013364 033711 001260
1691 013370 001420
1692 013372 010564 001312
1693 013376 010564 002312

;*****
;PROGRAM 3: ECHO EXERCISER
;*****

PROG3: RESET ;CLEAR OUT THE WORLD
MOV #STACK, SP ;RESET THE STACK POINTER
BIS #340, @#PS ;PROCESSOR TO LEVEL 7
MOV #XMTTAB, R1 ;FIRST TABLE POINTER
MOV #2000, R2 ;LENGTH OF TABLES (WORDS)
1$: CLR (R1)+ ;CLEAR THE TABLE
DFC R2
BNE 1$
MOV #400, R2 ;LENGTH OF MASK/COUNT TABLE
2$: INC R1 ;SKIP MASK
CLRB (R1)+ ;CLEAR COUNT
DEC R2
BNE 2$
MOV #END, R5 ;SET UP BUFFER POINTER

;ROUTINE TO INITIALIZE ALL DJ11'S AND THEIR ISR'S:
;SET UP ALL INTERRUPT VECTORS
;SET UP DEVICE ADDRESSES IN LINKER ROUTINES
;SET CSR'S EVERYTHING ENABLED
;SET TCR'S, ALL LINES ENABLED

P3INIT: CLR R0
MOV DEVADR, R1
MOV VECADR, R2
MOV #R3SR0, R3 ;SET UP POINTER TO LINKERS
1$: MOV R3, (R2)+ ;SET UP RECEIVER INTERRUPT VECTOR
MOV RCVLVL, (R2)+
CMP (R3)+, (R3)+ ;ADD 4 TO R3
MOV R1, (R3) ;ADDRESS OF CSR
ADD #2, (R3)+ ;ADDRESS OF RBUF
TST (R3)+
MOV R3, (R2)+ ;SET UP TRANSMITTER INTERRUPT VECTOR
MOV XMTLVL, (R2)+
CMP (R3)+, (R3)+
MOV R1, (R3)+ ;ADDRESS OF CSR
MOV #50400, (R1)+ ;SET UP CSR, TRANSMITTER ONLY
;BIT8 = TRANSMITTER SCAN ENABLE
;BIT12 = STATUS ENABLE
;BIT14 = TRANSMITTER INTERRUPT ENABLE

TST (R1)+
ASL R0 ;UNIT # * 2
MOV SVSWO(R0), (R1) ;SET TCR BITS FOR SELECTED LINES
ASR R0 ;RESET UNIT COUNTER
MOV #1, MARK ;SET UP MARKER
MOV (R3)+, R4 ;SET UP OFFSET TO TABLES
MOV R2, -(6) ;PUSH R2 ON STACK
MOV R3, -(6) ;PUSH R3 ON STACK
2$: BIT MARK, (R1) ;CHECK FOR LINE SELECTED
BEQ 6$
MOV R2, XMTTAB(4) ;SET UP HEADER MESSAGE
MOV R5, RCVTAB(4) ;SET UP RECEIVER TABLE

```

```
1694 013402 013702 001262      MOV    BUFSIZ, R2      ;SET UP COUNTER
1695 013406 012703 016707      MOV    #MSGP3, R3     ;SET UP MESSAGE POINTER
1696 013412 112325      3$: MOV    (R3)+, (R5)+ ;MOVE MESSAGE INTO BUFFER
1697 013414 001404      BEQ     5$      ;BRANCH IF END OF MESSAGE
1698 013416 005302      DEC     R2      ;COUNT BUFFER SIZE
1699 013420 001374      BNE     3$      ;BRANCH IF MORE
1700 013422 000403      BR      6$      ;BRANCH IF DONE
1701 013424 105025      4$: CLRB   (R5)+ ;CLEAR REST OF BUFFER
1702 013426 005302      5$: DEC     R2      ;COUNT BUFFER SIZE
1703 013430 001375      BNE     4$      ;BRANCH IF MORE
1704 013432 010564 003312      6$: MOV    R5, MAXTAB(4) ;SETUP BFR POINTER TABLE.
1705 013436 005724      TST     (R4)+ ;INC OFFSET TO NEXT LINE
1706 013440 006337 001260      ASL     MARK
1707 013444 103347      BCC     2$
1708 013446 012603      MOV    (6)+, R3 ;POP STACK INTO R3
1709 013450 012602      MOV    (6)+, R2 ;POP STACK INTO R2
1710 013452 022121      CMP     (R1)+, (R1)+ ;ADD 4
1711 013454 005200      INC     RC
1712 013456 020037 001264      CMP     RC, UNITS
1713 013462 001306      BNE     1$
1714 013464 042737 000140 177776 BIC     #140, @MPS ;LOWER PROCESSOR PRIORITY
1715
1716
1717
1718
1719
```

```
*****
;PROG3 FOREGROUND PROGRAM TO START RECEIVERS, THEN EXERCISE MEMORY.
*****
```

```
1720 013472 012701 001312      FORP3: MOV    #XMTTAB, R1
1721 013476 012702 000400      MOV    #400, R2
1722 013502 005711      1$: TST     (R1) ;CHECK FOR XMTR TABLE CLR
1723 013504 001376      BNE     1$      ;BRANCH IF NOT
1724 013506 062701 000002      ADD     #2, R1 ;GO TO NEXT LINE ENTRY
1725 013512 005302      DEC     R2      ;COUNT LINES
1726 013514 001372      BNE     1$      ;BRANCH IF MORE LINES
1727 013516 013700 001264      MOV    UNITS, R0 ;SET UP UNIT COUNTER
1728 013522 013701 001266      MOV    DEVADR, R1 ;AND DEVICE ADDRESS POINTER
1729 013526 052711 000101      2$: BIS     #101, (R1) ;SET RECEIVER ENABLES OF CSR
1730 ;BIT0 = RECEIVER ENABLE
1731 ;BIT6 = RCV INTERRUPT ENABLE
1732 013532 062701 000010      ADD     #10, R1 ;UPDATE TO NEXT DJ11
1733 013536 005300      DEC     R0      ;COUNT DJ11'S
1734 013540 001372      BNE     2$
1735 013542 012700 020000      MEMX3: MOV    #20000, R0 ;TOP OF 4K BANK OF MEMORY
1736 013546 000241      CLC
1737 013550 005540      1$: ADC     -(R0) ;FAST READ/WRITE TO MEMORY
1738 013552 001376      BNE     1$      ;RAPID REPEAT
1739 013554 005700      TST     RC ;CHECK FOR LOC 0
1740 013556 001374      BNE     1$      ;BRANCH IF MORE MEMORY
1741 013560 000770      BR      MEMX3 ;LOOP FOR EVER
1742
1743
1744
1745
1746
```

```
*****
;PROG3 LINKERS TO DJ11 INTERRUPT SERVICE ROUTINES
*****
```

```
1747 013562 004037 014312      43SRC: JSR     R0, P3RISR
1748 013566 160012 000000      .WORD   <160012>, <0*10>, <0*40>
1749 013572 004037 014162      43SRC: JSR     R0, P3XISR
```

1750	013576	160020	000000				
1751	013602	004037	014312	R3SR1:	JSR	RO,P3RISR	<160020+<0*10>>,<0*40>
1752	013606	160022	000040				
1753	013612	004037	014162	X3SR1:	JSR	RO,P3XISR	<160012+<1*10>>,<1*40>
1754	013616	160030	000040				
1755	013622	004037	014312	R3SR2:	JSR	RO,P3RISR	<160020+<1*10>>,<1*40>
1756	013626	160032	000100				
1757	013632	004037	014162	X3SR2:	JSR	RO,P3XISR	<160012+<2*10>>,<2*40>
1758	013636	160040	000100				
1759	013642	004037	014312	R3SR3:	JSR	RO,P3RISR	<160020+<2*10>>,<2*40>
1760	013646	160042	000140				
1761	013652	004037	014162	X3SR3:	JSR	RO,P3XISR	<160012+<3*10>>,<3*40>
1762	013656	160050	000140				
1763	013662	004037	014312	R3SR4:	JSR	RO,P3RISR	<160020+<3*10>>,<3*40>
1764	013666	160052	000200				
1765	013672	004037	014162	X3SR4:	JSR	RO,P3XISR	<160012+<4*10>>,<4*40>
1766	013676	160060	000200				
1767	013702	004037	014312	R3SR5:	JSR	RO,P3RISR	<160020+<4*10>>,<4*40>
1768	013706	160062	000240				
1769	013712	004037	014162	X3SR5:	JSR	RO,P3XISR	<160012+<5*10>>,<5*40>
1770	013716	160070	000240				
1771	013722	004037	014312	R3SR6:	JSR	RO,P3RISR	<160020+<5*10>>,<5*40>
1772	013726	160072	000300				
1773	013732	004037	014162	X3SR6:	JSR	RO,P3XISR	<160012+<6*10>>,<6*40>
1774	013736	160100	000300				
1775	013742	004037	014312	R3SR7:	JSR	RO,P3RISR	<160020+<6*10>>,<6*40>
1776	013746	160102	000340				
1777	013752	004037	014162	X3SR7:	JSR	RO,P3XISR	<160012+<7*10>>,<7*40>
1778	013756	160110	000340				
1779	013762	004037	014312	R3SR10:	JSR	RO,P3RISR	<160020+<7*10>>,<7*40>
1780	013766	160112	000400				
1781	013772	004037	014162	X3SR10:	JSR	RO,P3XISR	<160012+<10*10>>,<10*40>
1782	013776	160120	000400				
1783	014002	004037	014312	R3SR11:	JSR	RO,P3RISR	<160020+<10*10>>,<10*40>
1784	014006	160122	000440				
1785	014012	004037	014162	X3SR11:	JSR	RO,P3XISR	<160012+<11*10>>,<11*40>
1786	014016	160130	000440				
1787	014022	004037	014312	R3SR12:	JSR	RO,P3RISR	<160020+<11*10>>,<11*40>
1788	014026	160132	000500				
1789	014032	004037	014162	X3SR12:	JSR	RO,P3XISR	<160012+<12*10>>,<12*40>
1790	014036	160140	000500				
1791	014042	004037	014312	R3SR13:	JSR	RO,P3RISR	<160020+<12*10>>,<12*40>
1792	014046	160142	000540				
1793	014052	004037	014162	X3SR13:	JSR	RO,P3XISR	<160012+<13*10>>,<13*40>
1794	014056	160150	000540				
1795	014062	004037	014312	R3SR14:	JSR	RO,P3RISR	<160020+<13*10>>,<13*40>
1796	014066	160152	000600				
1797	014072	004037	014162	X3SR14:	JSR	RO,P3XISR	<160012+<14*10>>,<14*40>
1798	014076	160160	000600				
1799	014102	004037	014312	R3SR15:	JSR	RO,P3RISR	<160020+<14*10>>,<14*40>
1800	014106	160162	000640				
1801	014112	004037	014162	X3SR15:	JSR	RO,P3XISR	<160012+<15*10>>,<15*40>
1802	014116	160170	000640				
1803	014122	004037	014312	R3SR16:	JSR	RO,P3RISR	<160020+<15*10>>,<15*40>
1804	014126	160172	000700				
1805	014132	004037	014162	X3SR16:	JSR	RO,P3XISR	<160012+<16*10>>,<16*40>

1806 014136 160200 000700
1807 014142 004037 014312
1808 014146 160202 000740
1809 014152 004037 014162
1810 014156 160210 000740
1811
1812
1813
1814
1815

.WORD <160020+<16*10>>,<16*40>
R3SR17: JSR R0,P3RISR
.WORD <160012+<17*10>>,<17*40>
X3SR17: JSR R0,P3XISR
.WORD <160020+<17*10>>,<17*40>

;PROG3 TRANSMITTER INTERRUPT SERVICE ROUTINE

1816 014162
1817 014162 010146
1818 014164 010246
1819 014166 012001
1820 014170 005711
1821 014172 100043
1822 014174 116102 000007
1823 014200 006302
1824 014202 061002
1825 014204 117261 001312 000006
1826 014212 105072 001312
1827 014216 005262 001312
1828 014222 026262 003312 001312
1829 014230 001003
1830 014232 163762 001262 001312
1831 014240 105772 001312
1832 014244 001351
1833 014246 010346
1834 014250 005062 001312
1835 014254 161002
1836 014256 006202
1837 014260 005003
1838 014262 000261
1839 014264 006103
1840 014266 005302
1841 014270 100375
1842 014272 040361 000004
1843 014276 012603
1844 014300 000733
1845 014302
1846 014302 012602
1847 014304 012601
1848 014306 012600
1849 014310 000002
1850

P3XISR:
MOV R1,-(6) ;PUSH R1 ON STACK
MOV R2,-(6) ;PUSH R2 ON STACK
MOV (R0)+,R1
1\$: TST (R1) ;CHECK FOR TRANS READY
BPL 4\$
MOVB 7(R1),R2 ;GET LINE NO.
ASL R2
ADD (R0),R2
MOVB @XMTTAB(2),6(R1);SEND A CHARACTER
CLRB @XMTTAB(2) ;CLR TABLE AFTER USE
INC XMTTAB(2) ;UPDATE TABLE POINTER
CMP MAXTAB(2),XMTTAB(2) ;CHECK FOR END OF BUFFER ;:++F
BNE 5\$;BRANCH IF NOT
SUB BUFSIZ,XMTTAB(2) ;RESET BUFFER POINTER ;:++F
5\$: TSTB @XMTTAB(2) ;CHECK NEXT CHARACTER
BNE 1\$;BRANCH IF MORE DATA
2\$: MOV R3,-(SP)
CLR XMTTAB(2) ;CLEAR TABLE POINTER
SUB (R0),R2
ASR R2
CLR R3
3\$: ROL R3
DEC R2
BPL 3\$
BIC R3,4(R1) ;CLEAR TCR BIT FOR LINE
MOV (SP)+,R3 ;RESTORE R3
BR 1\$
4\$: MOV (6)+,R2 ;POP STACK INTO R2
MOV (6)+,R1 ;POP STACK INTO R1
MOV (6)+,R0 ;POP STACK INTO R0
RTI

;PROG3 RECEIVER INTERRUPT SERVICE ROUTINE

1851
1852
1853
1854
1855 014312
1856 014312 010146
1857 014314 010246
1858 014316 010346
1859 014320 010446
1860 014322 012001
1861 014324 011102

P3RISR:
MOV R1,-(6) ;PUSH R1 ON STACK
MOV R2,-(6) ;PUSH R2 ON STACK
MOV R3,-(6) ;PUSH R3 ON STACK
MOV R4,-(6) ;PUSH R4 ON STACK
MOV (R4)+,R1 ;GET RBUF ADDRESS
1\$: MOV (R1),R2 ;READ THE DATA

1862	014326	100077			BPL	8\$			;BRANCH IF NO CHAR PRESENT
1863	014330	032702	070000		BIT	#70000,R2			;CHECK FOR ERRORS
1864	014334	001402			BEQ	2\$			;BRANCH IF OK
1865	014336	104002			HLT+2				;RECEIVER ERROR
1866									;R1=RBUF ADDRESS
1867									;R2=CONTENTS OF RBUF
1868									;BIT12=PARITY ERROR
1869									;BIT13=FRAMING ERROR
1870									;BIT14=UART OVERRUN
1871	014340	000771			BR	1\$			;SKIP BAD DATA
1872	014342	010204			2\$: MOV	R2, R4			;DUP THE RBUF
1873	014344	105004			CLRB	R4			;CLEAR THE DATA
1874	014346	000304			SWAB	R4			;LINE # TO LOW BYTE
1875	014350	106304			ASLB	R4			;LINE # * 2, ALSO CLR CHAR PRESENT
1876	014352	061004			ADD	(R0),R4			;ADD OFFSET
1877	014354	136402	005312		BITB	MASK(4).R2			;CHECK CHARACTER LENGTH
1878	014360	001401			BEQ	3\$			;BRANCH IF OK
1879	014362	104002			HLT+2				;CHARACTER LENGTH ERROR
1880									;R1=RBUF ADDRESS
1881									;R2=CONTENTS OF RBUF (DATA)
1882	014364	005764	002312		3\$: TST	RCVTAB(4)			;CHECK FOR UNSELECTED LINE
1883	014370	001002			BNE	4\$			;BRANCH IF OK
1884	014372	104002			HLT+2				;RECEIVED DATA ON UNSELECTED LINE
1885									;R1 = RBUF ADDRESS
1886									;R2 = CONTENTS OF RBUF
1887	014374	000753			BR	1\$			;IGNORE THE DATA
1888	014376	105774	002312		4\$: TSTB	@RCVTAB(4)			;CHECK FOR DATA BUFFER FULL
1889	014402	001403			BEQ	5\$			;BRANCH IF OK
1890	014404	104002			HLT+2				;SOFTWARE DATA BUFFER OVERFLOW
1891									;POSSIBLE TRANSMITTER PROBLEM
1892									;R1 = RBUF ADDRESS
1893									;R2 = CONTENTS OF RBUF
1894					;NOTE:	IF THE ABOVE ERROR WAS DUE TO OVERLOAD, INCREASING THE CONTENTS			
1895					:	OF 'BUFSIZ' MAY RECTIFY THE PROBLEM.			
1896					:	'BUFSIZ' MUST BE A MULTIPLE OF 2.			
1897					:	INCREASING IT MAY CAUSE THE BUFFERS TO OVERFLOW 4K.			
1898	014406	000137	013210		JMP	PROG3			;RESTART ON THIS TYPE ERROR
1899									
1900	014412	005764	001312		5\$: TST	XMTTAB(4)			;CHECK FOR TRANSMITTER ACTIVE
1901	014416	001414			BEQ	6\$			;BRANCH IF INACTIVE
1902	014420	110274	002312		MOVB	R2, @RCVTAB(4)			;PUT THE DATA IN THE BUFFER
1903	014424	005264	002312		INC	RCVTAB(4)			;UPDATE POINTER TO NEXT SPACE
1904	014430	026464	003312	002312	CMP	MAXTAB(4),RCVTAB(4)			;CHECK FOR END OF BUFFER
1905	014436	001332			BNE	1\$			;BRANCH IF NOT
1906	014440	163764	001262	002312	SUB	BUFSIZ,RCVTAB(4)			;RESET BUFFER POINTER
1907	014446	000726			BR	1\$			
1908	014450	016464	003312	002312	6\$: MOV	MAXTAB(4),RCVTAB(4)			;RESET TABLE POINTER
1909	014456	163764	001262	002312	SUB	BUFSIZ,RCVTAB(4)			;RESET BUFFER
1910	014464	016464	002312	001312	MOV	RCVTAB(4),XMTTAB(4)			
1911	014472	110274	002312		MOVB	R2, @RCVTAB(4)			
1912	014476	005264	002312		INC	RCVTAB(4)			;UPDATE POINTER TO NEXT SPACE
1913	014502	161004			SUB	(R0),R4			
1914	014504	006204			ASR	R4			
1915	014506	005003			CLR	R3			
1916	014510	000261			SEC				
1917	014512	006103			7\$: ROI	R3			

1918	014514	005304			DEC	R4	
1919	014516	100375			BPL	7\$	
1920	014520	050361	000002		BIS	R3,2(R1)	;SET TCR BIT FOR LINE
1921	014524	000677			BR	1\$	
1922	014526			8\$:			
1923	014526	012604			MOV	(6)+,R4	;POP STACK INTO R4
1924	014530	012603			MOV	(6)+,R3	;POP STACK INTO R3
1925	014532	012602			MOV	(6)+,R2	;POP STACK INTO R2
1926	014534	012601			MOV	(6)+,R1	;POP STACK INTO R1
1927	014536	012600			MOV	(6)+,R0	;POP STACK INTO R0
1928	014540	000002			RTI		
1929	014542			DONE:			
1930	014542	004737	016310		JSR	PC, KBDINT	
1931	014546	062737	000001	001206	ADD	#1,PCNT+2	;ADD 1 TO THE PASS COUNT
1932	014554	005537	001204		ADC	PCNT	;MAKE IT DOUBLE PREC.
1933	014560	000004	016403		TYPE	,MEOP	;END OF PASS INDICATOR
1934	014564	032777	002000	164416	BIT	#SW10,@SWR	;RING THE BELL?
1935	014572	001004			BNE	4\$	;NO!
1936	014574	000004	000007		TYPE	,BELL	;RING THE BELL
1937	014600	000004	000177		TYPE	,177	;TYPE A FILLER FOR 11/05
1938	014604	013700	000042		MOV	@42,R0	;GET MONITOR ADDRESS
1939	014610	001405		4\$:	BEQ	3\$	;IF NONE
1940	014612	000005			RESET		;RESET AND
1941		014614			SENDAD =		
1942	014614	004710			JSR	7,(0)	;GO TO MONITOR
1943	014616	000240			NOP		;SAVE ROOM
1944	014620	000240			NOP		;FOR
1945	014622	000240			NOP		;ACT11
1946	014624	000137	007310		JMP	RECTAR	;RETURN
1947				3\$:			

```
1948
1949
1950      :      $HLT      ERROR TYPEOUT HANDLER
1951      :THIS ROUTINE PRINTS OUT ERROR MESSAGES STARTING WITH THE
1952      :ADDRESS OF THE 'HLT'. IT ALSO COUNTS THE NUMBER OF ERRORS
1953      :AND HAS THE CAPABILITY OF LOOPING ON ERROR, BELL ON ERROR,
1954      :'HALT' ON ERROR, AND INHIBIT TYPEOUTS. AN OPTIONAL ARGUMENT
1955      : (HLT+3) WILL BE PLACED IN 'HLTCT$' FOR ADITONAL TYPEOUTS.
1956
1957 014630 004737 016310      EMT$: JSR      PC,      KBDINT
1958 014634 032777 002000 164346      BIT      #SW10,@SWR      ;BELL ON ERROR?
1959 014642 001402      BEQ      1$      ;NO - SKIP
1960 014644 000004 000007      TYPE      BELL      ;RING BELL
1961 014650 005237 001202      1$: INC      ERRORS      ;COUNT THE NUMBER OF ERRORS
1962 014654 032777 020000 164326      BIT      #SW13,@SWR      ;SKIP TYPEOUT IF SET
1963 014662 001026      BNE      2$      ;SKIP TYPEOUTS
1964 014664 000004 014670      TYPE      ..+2      ;.ASCIZ <15><12>
1965 014674 011637 014760      MOV      (6),HLTADR      ;PUT ADDRESS OF INSTRUCTION ON STACK
1966 014700 162737 000002 014760      SUB      #2,HLTADR      ;FUDGE ADDRESS
1967 014706 117737 000046 014756      MOVEB   @HLTADR,HLTCT$      ;GET HLT ARGUEMENT
1968 014714 013705 014760      MOV      HLTADR,ITY      ;TYPE HLTADR IN OCTAL
1969 014720 004737 015366      JSR      PC,PRINTR      ;TYPE LEADING ZERO'S
1970 014724 000004 014730      TYPE      ..+2      ;.ASCIZ " "
1971 014734 004737 014762      JSR      PC,ERROR$      ;GO TO USER ERROR ROUTINE
1972 014740 005777 164244      2$: TST      @SWR      ;HALT ON ERROR
1973 014744 100001      BPL      .+4      ;SKIP IF CONTINUE
1974 014746 000000      HALT      ;HALT ON ERROR.
1975 014750 004737 016310      JSR      PC,KBDINT
1976 014754 000002      RTI      ;RETURN
1977
1978 014756 000000      HLTCT$: 0      ;HLT ARGUEMENT
1979 014760 000000      HLTADR: 0      ;LAST HLT INSTRUCTION EXECUTED
1980
1981 014762 042737 007700 015004      ERROR$: BIC      #7700,2$
1982 014770 105337 014756      1$: DECB     HLTCT$
1983 014774 100411      BMI      3$
1984 014776 062737 000100 015004      ADD      #100,2$
1985 015004 010005      2$: MOV      %0,ITY      ;TYPE REGISTER X IN OCTAL
1986 015006 004737 015366      JSR      %7,PRINTR
1987 015012 000004 016442      TYPE      SPACE
1988 015016 000764      BR      1$
1989 015020 000207      3$: RTS
```

```
1990
1991
1992
1993 015022 012737 000001 015314 READIN: MOV #1,INHRE
1994 015030 004737 015170 JSR PC, READ$ ;GO READ TTY UNTIL CR
1995 015034 005037 015314 CLR INHRE
1996 015040 010146 MOV R1,-(6) ;PUSH R1 ON STACK
1997 015042 010246 MOV R2,-(6) ;PUSH R2 ON STACK
1998 015044 010346 MOV R3,-(6) ;PUSH R3 ON STACK
1999 015046 012501 MOV (R5)+, R1
2000 015050 012737 000020 016206 MOV #20,CNT
2001 015056 012702 015316 MOV #INPUT,R2
2002 015062 122712 000120 CMPB #120,(R2) ;CHECK FOR 'P'
2003 015066 001425 BEQ 3$
2004 015070 005011 CLR (R1)
2005 015072 112203 1$: MOVB (R2)+,R3
2006 015074 120327 000015 CMPB R3,#15
2007 015100 001420 BEQ 3$ ;BRANCH WHEN DONE
2008 015102 162703 000060 SUB #60,R3
2009 015106 032703 177770 BIT #177770,R3
2010 015112 001013 BNE 3$ ;BRANCH IF BAD DATA
2011 015114 006511 ASL (R1)
2012 015116 103410 BCS 2$
2013 015120 006311 ASL (R1)
2014 015122 103406 BCS 2$
2015 015124 006311 ASL (R1)
2016 015126 103404 BCS 2$
2017 015130 050311 BIS R3,(R1)
2018 015132 005337 016206 DEC CNT
2019 015136 000755 BR 1$
2020 015140 000244 2$: CLZ
2021 015142 013737 177776 015166 3$: MOV @#PS, PSTEMP ;MAKE SURE Z-BIT IS CLR
2022 015150 012603 MOV (6)+,R3 ;SAVE CONDITION CODES
2023 015152 012602 MOV (6)+,R2 ;POP STACK INTO R3
2024 015154 012601 MOV (6)+,R1 ;POP STACK INTO R2
2025 015156 013737 015166 177776 MOV PSTEMP, @#PS ;POP STACK INTO R1
2026 015164 000205 RTS R5 ;RESTORE CONDITION CODES
2027
2028 015166 000000 PSTEMP: 0 ;TEMPORARY STORAGE FOR PS
2029
2030 015170 010346 READ$: MOV R3,-(6) ;SAVE R3
2031 015172 012703 015316 1$: MOV #INPUT,R3 ;GET ADDRESS
2032 015176 022703 015336 2$: CMP #INPUT+20,R3 ;BUFFER FULL?
2033 015202 001415 BEQ 4$ ;YES - TYPE '?'
2034 015204 105737 177560 TSTB @#177560 ;WAIT FOR
2035 015210 100375 BPL -4 ;A CHARACTER
2036 015212 113713 177562 MOVB @#177562,(3) ;GET CHARACTER
2037 015216 142713 000200 BICB #200,(3) ;GET RID OF JUNK
2038 015222 122713 000177 CMPB #177,(3) ;IS IT A RIUBOUT
2039 015226 001403 BEQ 4$ ;SKIP IF NOT
2040 015230 122713 000025 CMPB #25,(3)
2041 015234 001006 BNE 3$
2042 015236 4$:
2043 015236 000004 015242 TYPE -+2 ;.ASCIZ '?'<15><12>' '
2044 015250 000750 BR 1$ ;ZAP THE BUFFER AND LOOP
2045 015252 111337 016124 3$: MOVB (3),.TYPE ;SET UP FOR TYPING
```

Address	Offset	Hex	Label	Instruction	Comment
2046	015256	000004	016124	TYPE	TYPE
2047	015262	122723	000015	CMPB	#15,(3)+
2048	015266	001343		BNE	2\$
2049	015270	005737	015314	TST	INHRE
2050	015274	001401		BEQ	5\$
2051	015276	000402		BR	6\$
2052	015300	105063	177777	5\$: CLRB	-1(3)
2053	015304	000004	000012	6\$: TYPE	,12
2054	015310	012603		MOV	(6)+,R3
2055	015312	000207		RTS	PC
2056					
2057	015314	000000		INHRE:	0
2058	015316	000020		INPUT:	.BLKW 20
2059				:	\$OCTAL
2060					OCTAL TYPEOUT ROUTINE
2061					:THIS ROUTINE IS USED TO TYPE AN OCTAL NUMBER ON THE TTY. IT WILL TYPE
2062					:ALL 6 CHARACTERS, SUPPRESS LEADING ZEROES, TYPE AN 18 BIT ADDRESS, OR TYPE
2063					:THE 16 BITS. IT IS CALLED VIA THE DUMP, SDUMP, DUMP18, OR BITYPE MACRO'S.
2064					
2065	015356	012737	170101	015524	BITYP\$: MOV
2066	015364	000411			BR
2067	015366	112737	000001	015524	PRINTR: MOVB
2068	015374	000402			BR
2069	015376	005037	015524		PRINTS: CLR
2070	015402	112737	177772	015525	MOV
2071	015410	010446			.PTIT: MOV
2072	015412	012704	015526		MOV
2073	015416	105014			CLRB
2074	015420	000411			BR
2075	015422	105014			.PRL: CLRB
2076	015424	032737	000100	015524	BIT
2077	015432	001004			BNE
2078	015434	006105			ROL
2079	015436	106114			ROLB
2080	015440	006105			ROL
2081	015442	106114			ROLB
2082	015444	006105			.PRF: ROL
2083	015446	106114			ROLB
2084	015450	105714			TSTB
2085	015452	001402			BEQ
2086	015454	105237	015524		INCB
2087	015460	105737	015524		TSTB
2088	015464	001402			BEQ
2089	015466	152724	000060		BISB
2090	015472	105237	015525		INCB
2091	015476	001351			BNE
2092	015500	022704	015526		CMP
2093	015504	001002			BNE
2094	015506	112724	000060		MOVB
2095	015512	105014			CLRB
2096	015514	000004	015526		TYPE
2097	015520	012604			MOV
2098	015522	000207			RTS
2099	015524	000012			.PR: .BLKW

```
2100 015550 012777 015676 000126 PDOWN$: MOV #ILLUP,@PUVEC$ :SET FOR FAST UP
2101 015556 012777 000340 000122 MOV #340,@PUVEC$+2 :PRIO:7
2102 015564 010046 MOV R0,-(6) :PUSH R0 ON STACK
2103 015566 010146 MOV R1,-(6) :PUSH R1 ON STACK
2104 015570 010246 MOV R2,-(6) :PUSH R2 ON STACK
2105 015572 010346 MOV R3,-(6) :PUSH R3 ON STACK
2106 015574 010446 MOV R4,-(6) :PUSH R4 ON STACK
2107 015576 010546 MOV R5,-(6) :PUSH R5 ON STACK
2108 015600 010637 015702 MOV SP,.SAVR6 :SAVE SP
2109 015604 012777 015614 000072 MOV #PUP$,@PUVEC$ :SET UP VECTOR
2110 015612 000000 HALT :WAIT FOR PF
2111
2112 015614 013706 015702 PUP$: MOV .SAVR6,SP :GET SP
2113 015620 005001 CLR R1 :WAIT LOOP FOR THE TTY
2114 015622 005201 1$: INC R1 :WAIT FOR THE INC
2115 015624 001376 BNE 1$ :OF WORD
2116 015626 012605 MOV (6)+,R5 :POP STACK INTO R5
2117 015630 012604 MOV (6)+,R4 :POP STACK INTO R4
2118 015632 012603 MOV (6)+,R3 :POP STACK INTO R3
2119 015634 012602 MOV (6)+,R2 :POP STACK INTO R2
2120 015636 012601 MOV (6)+,R1 :POP STACK INTO R1
2121 015640 012600 MOV (6)+,R0 :POP STACK INTO R0
2122 015642 012737 015550 000024 MOV #PDOWN$,@#24 :SET UP THE POWER DOWN VECTOR
2123 015650 012737 000340 000026 MOV #340,@#26 :PRIO:7
2124 015656 000004 015662 TYPE .,+2 :.ASCIZ <15><12>'POWER'
2125 015672 000137 007310 JMP RESTAR :JMP TO USER ADDRESS
2126
2127 015676 000000 ILLUP: HALT :THE POWER UP SEQUENCE WAS STARTED
2128 015700 000776 BR .-2 : BEFORE THE POWER DOWN WAS COMPLETE
2129
2130 015702 000000 .SAVR6: 0 :PUT THE SP HERE
2131 015704 000024 000026 PUVEC$: 24,26 :POWER UP VECTOR
2132
2133
2134 015710 000002 YESRT: RTI :RETURN FROM TRACE TRAP
```

```
2135
2136
2137 ;*****
2138 ;IOT HANDLER - REENTERENT ROUTINE TO INDICATE A FALSE
2139 ; INTERRUPT OR TRAP, OR TO TYPE A MESSAGE
2140 ;*****
2141 IOTRAP: CMP      #1000, (SP)      ;CHECK RETURN ADDRESS
2142          BLT      IOT$,          ;BRANCH IF TYPE COMMAND
2143          SUB      #4, (SP)        ;GET VECTOR ADDRESS
2144          TYPE,    .+2             ;TYPE MESSAGE
2145          .ASCIIZ  <15><12>'UNEXPECTED INTERUPT TO '
2146
2147
2148
2149
2150          MOV      (SP)+,TTY        ;TYPE (SP)+ IN OCTAL
2151          JSR      PC,PRINTS        ;AND SUPPRESS LEADING ZERO'S
2152          TST      (SP)+           ;POP STACK
2153          TYPE,    .+2             ;TYPE MESSAGE
2154          .ASCIIZ  ' FROM '
2155
2156          .EVEN
2157          MOV      (SP),TTY         ;TYPE (SP) IN OCTAL
2158          JSR      PC,PRINTS        ;AND SUPPRESS LEADING ZERO'S
2159          HALT
2160          RTI                     ;CONTINUE IF DESIRED
2161
2162 :      $TYPE      MESSAGE TIMEOUT ROUTINE
2163
2164 ;THIS ROUTINE IS USE TO TYPE ASCII MESSAGES ON THE TTY. THE
2165 ;CALL CAN BE IN ONE OF 3 FORMS: 1) 'TYPE ,ADR' - TYPES THE
2166 ;MESSAGE STARTING IN LOCATION 'ADR:', 2) 'TYPE ,CHAR' - TYPES
2167 ;THE ASCII 'CHAR', AND 3) 'PRINT <<15><12>'MESSAGE'>' - TYPES
2168 ;THE MESSAGE WHICH IS INLINE ASCII.
2169
2170 IOT$:  MOV      TTY,-(6)           ;SAVE TTY
2171          MOV      @2(6),TTY        ;GET ADDRESS TO BE TYPED
2172          BIT      #177400,TTY      ;IS IT A TYPEN?
2173          BNE      1$              ;NO
2174          MOV      TTY,.TYPE        ;GET THE CHARACTER
2175          MOV      #.TYPE,TTY       ;FUDGE THE ADDRESS
2176          1$:    TSTB (TTY)         ;TERMINATOR?
2177          BEQ      2$              ;GET OUT IF SO
2178          MOVB     (TTY)+,@#177566 ;LOAD AND TYPE THE CHARACTER
2179          TSTB     @#177564         ;IS THE PRINTER READY
2180          BPL      -4              ;WAIT UNTIL IT IS
2181          BR       1$              ;GET THE NEXT CHARACTER
2182          2$:    MOV      @2(6),-(6) ;GET ADDRESS TO BE TYPED
2183          ADD      #2,4(6)          ;ADD 2 TO THE ADDRESS
2184          CMP      (6)+,2(6)        ;IS IT .+2?
2185          BNE      3$              ;NO
2186          ADD      #2,TTY           ;ADD 2 TO THE ADDRESS
2187          BIC      #1,TTY           ;BACK UP TO AN EVEN BYTE
2188          MOV      TTY,2(6)         ;RESTORE ADDRESS
2189          3$:    MOV      (6)+,TTY   ;RESTORE TTY
2190          RTI                     ;RETURN
```

```

2191 016124 000000 .TYPE: 0 ;CHARACTER TYPE LOCATION
2192
2193 016126 022737 000176 001210 CNTLU: CMP #SWREG,SWR
2194 016134 001023 BNE 1$
2195 016136 000004 016220 TYPE ,SWREQ
2196 016142 013705 000176 MOV SWREG,TTY ;TYPE SWREG IN OCTAL
2197 016146 004737 015366 JSR PC,PRINTR ;TYPE LEADING ZERO'S
2198 016152 000004 016210 TYPE ,NEWIS
2199 016156 004537 015022 JSR R5,READIN
2200 016162 016356 .WORD TMP1
2201 016164 001360 BNE CNTLU
2202 016166 022737 000020 016206 CMP #20,CNT
2203 016174 001403 BEQ 1$
2204 016176 013777 016356 163004 MOV TMP1,@SWR
2205 016204 000207 1$: RTS PC
2206
2207 016206 000000 CNT: 0
2208
2209 016210 020040 042516 036527 NEWIS: .ASCIIZ '' NEW- ''
2210 016216 000040
2211 016220 005015 053523 036522 SWREQ: .ASCIIZ <15><12>'SWR= '
2212 016226 000040
2213
2214
2215 016230 013746 000006 SUSWRR: MOV 6,-(SP)
2216 016234 013746 000004 MOV 4,-(SP)
2217 016240 012737 016260 000004 MOV #1$,4
2218 016246 022777 177777 162734 CMP #-1,@SWR
2219 016254 001402 BEQ 2$
2220 016256 000407 BR 3$
2221 016260 022626 1$: CMP (SP)+,(SP)+
2222 016262 012737 000176 001210 2$: MOV #SWREG,SWR
2223 016270 012737 000174 001212 MOV #DISPREG,DISPLAY
2224 016276 012637 000004 3$: MOV (SP)+,4
2225 016302 012637 000006 MOV (SP)+,6
2226 016306 000207 RTS PC
2227
2228
2229 016310 022737 000176 001210 KBDINT: CMP #SWREG,SWR
2230 016316 001016 BNE 1$
2231 016320 005037 016356 CLR TMP1
2232 016324 113737 177562 016356 MOVB 177562,TMP1
2233 016332 142737 000200 016356 BICB #200,TMP1
2234 016340 122737 000007 016356 CMPB #7,TMP1
2235 016346 001002 BNE 1$
2236 016350 004737 016126 JSR PC,CNTLU
2237 016354 000207 1$: RTS PC
2238
2239 016356 000000 TMP1: 0
2240
2241
2242 016360 005015 047516 042040 TRNERR: .ASCIIZ <15><12>'NO DATA RECEIVED''
2243 016366 052101 020101 042522
2244 016374 042503 053111 042105
2245 016402 000
2246 016403 015 042412 050117 MEOP: .ASCIIZ <15><12>'EOP''

```

:::++F

CZDJBF0 DJ11 EXER & ONLINE
CZDJBF.P11 21-JUN-78 13:56

MACY11 30A(1052) 21-JUN-78 14:01 PAGE 46
TYPE ROUTINE

SEQ 0045

```
2247 016410 000
2248 016411 043 000040
2249 016414 051440 046105 041505 MNUM: .ASCIZ '# '
2250 016422 020124 044514 042516 MSGSEL: .ASCIZ 'SELECT LINE - '
2251 016430 036440 000040
2252 016434 005015 177777 000377 RETURN: .ASCIZ <15><12><377><377><377>
2253 016442 020040 000 SPACE: .ASCIZ ' '
2254 016445 015 177412 055103 MSGMDN: .ASCIZ <15><12><377>'CZDJB-F-0 DJ11 EXER & ONLINE'
2255 016452 045104 026502 026506
2256 016460 020060 020040 045104
2257 016466 030461 042440 042530
2258 016474 020122 020046 047117
2259 016502 047114 000105
2260 016506 005015 044506 051522 MSGADR: .ASCIZ <15><12>'FIRST DJ11 ADDRESS: '
2261 016514 020124 045104 030461
2262 016522 040440 042104 042522
2263 016530 051523 020072 000040
2264 016536 005015 042526 052103 MSGVEC: .ASCIZ <15><12>'VECTOR ADDRESS: '
2265 016544 051117 040440 042104
2266 016552 042522 051523 020072
2267 016560 000040
2268 016562 005015 047516 020056 MSGNUM: .ASCIZ <15><12>'NO. OF DJ11'S: '
2269 016570 043117 042040 030512
2270 016576 023461 035123 020040
2271 016604 000
2272 016605 015 050012 047522 MSGPRG: .ASCIZ <15><12>'PROGRAM #: '
2273 016612 051107 046501 021440
2274 016620 020072 000040
2275 016624 005015 047516 042040 MSG01: .ASCIZ <15><12>'NO DJ11'S.'
2276 016632 030512 023461 020523
2277 016640 000
2278 016641 015 050012 047522 MSGP2: .ASCIZ <15><12>'PROG2: CONTINUOUS ECHO EXERCISER'<15><12>
2279 016646 031107 020072 041440
2280 016654 047117 044524 052516
2281 016662 052517 020123 041505
2282 016670 047510 042440 042530
2283 016676 041522 051511 051105
2284 016704 005015 000
2285 016707 015 025012 041505 MSGP3: .ASCIZ <15><12>'*ECHO TEST*'<15><12>
2286 016714 047510 052040 051505
2287 016722 025124 005015 000
2288 016727 040 020055 000 MSGDAS: .ASCIZ ' - '
2289 016733 015 051412 046111 MALARM: .ASCIZ <15><12>'SILO ALARM LEVEL FOR CSR'<15><12>
2290 016740 020117 046101 051101
2291 016746 020115 042514 042526
2292 016754 020114 047506 020122
2293 016762 051503 006522 000012
2294
2295 016770 000000 .EVEN
2296 000001 END: 0
.END
```

[illegible]

MALARM	016733	923	2289#											
MARK	001260	526#	817*	823	830*	1441*	1443	1448*	1567*	1569*	1572	1619*	1621*	1624
		1686*	1690	1706*										
MASK	005312	554#	555	686	1078	1609	1877							
MAXTAB	003312	549#	1704*	1828	1904	1908								
MEMX3	013542	1735#	1741											
MEOP	016403	1933	2246#											
MINUM	016411	726	2248#											
MSGADR	016506	648	2260#											
MSGDAS	016727	926	2288#											
MSGMDN	016445	646	2254#											
MSGNUM	016562	654	2268#											
MSGPRG	016605	717	2272#											
MSGP2	016641	1445	2278#											
MSGP3	016707	1695	2285#											
MSGSEL	016414	729	2249#											
MSGVEC	016536	651	2264#											
MSG01	016624	599	2275#											
N	000040	389#	1120#	1121#	1122#	1123#	1124#	1125#	1126#	1127#	1128#	1129#	1130#	1131#
		1132#	1133#	1134#	1135#	1136#	1137#	1138#	1139#	1140#	1141#	1142#	1143#	1144#
		1145#	1146#	1147#	1148#	1149#	1150#	1151#	1152#	1153#	1154#	1155#	1156#	1157#
		1158#	1159#	1160#	1161#	1162#	1163#	1164#	1165#	1166#	1167#	1168#	1169#	1170#
		1171#	1172#	1173#	1174#	1175#	1176#	1177#	1178#	1179#	1180#	1181#	1182#	1183#
		1184#	1185#	1186#	1187#	1188#	1189#	1190#	1191#	1192#	1193#	1194#	1195#	1196#
		1197#	1198#	1199#	1200#	1201#	1202#	1203#	1204#	1205#	1206#	1207#	1208#	1209#
		1210#	1211#	1212#	1213#	1214#	1215#	1216#	1217#	1218#	1219#	1220#	1221#	1222#
		1223#	1224#	1225#	1226#	1227#	1228#	1229#	1230#	1231#	1232#	1233#	1234#	1235#
		1236#	1237#	1238#	1239#	1240#	1241#	1242#	1243#	1244#	1245#	1246#	1247#	1248#
		1249#	1250#	1251#	1252#	1253#	1254#	1255#	1256#	1257#	1258#	1259#	1260#	1261#
		1262#	1263#	1264#	1265#	1266#	1267#	1268#	1269#	1270#	1271#	1272#	1273#	1274#
		1275#	1276#	1277#	1278#	1279#	1280#	1281#	1282#	1283#	1284#	1285#	1286#	1287#
		1288#	1289#	1290#	1291#	1292#	1293#	1294#	1295#	1296#	1297#	1298#	1299#	1300#
		1301#	1302#	1303#	1304#	1305#	1306#	1307#	1308#	1309#	1310#	1311#	1312#	1313#
		1314#	1315#	1316#	1317#	1318#	1319#	1320#	1321#	1322#	1323#	1324#	1325#	1326#
		1327#	1328#	1329#	1330#	1331#	1332#	1333#	1334#	1335#	1336#	1337#	1338#	1339#
		1340#	1341#	1342#	1343#	1344#	1345#	1346						

SEQ 0048

MACY11 30A(1052) 21-JUN-78 14:01 PAGE 51
CROSS REFERENCE TABLE -- USER SYMBOLS

K 4

[illegible]

X3SR5	013712	1769#																
X3SR6	013732	1773#																
X3SR7	013752	1777#																
YESRT	015710	2134#																
SENDAD=	014614	489	1941#															
.	= 016772	481#	485	488#	491#	495#	497#	500#	544#	546#	549#	551#	554#	1388#				
		1941	1948	1964	1965#	1970	1971#	1973	1978	1979	2035	2043	2058#	2068				
		2085	2088	2093	2099#	2124	2128	2144	2153	2156#	2180							
.BIT	= 132000	389#	394	395	396	397	398	399	400	401	402	1934	1957	1962				
		1976																
.PR	015524	2065*	2067*	2069*	2070*	2072	2076	2086*	2087	2090*	2092	2096	2099#					
.PRF	015444	2074	2077	2082#														
.PRL	015422	2075#	2091															
.PTIT	015410	2066	2071#															
.SAVR6	015702	2108*	2112	2130#														
.TYPE	016124	2045*	2046	2174*	2175	2191#												

. ABS. 016772 000

ERRORS DETECTED: 0

DSKZ:CZDJB.F,DSKZ:CZDJB.F.SEQ-DSKZ:CZDJB.F.MAC,DSKZ:CZDJB.F.P11

RUN-TIME: 6 9 .6 SECONDS

RUN-TIME RATIO: 124/17 7.2

CORE USED: 20K (39 PAGES)

DOCUMENT PAGES: 51