

DJ11

EXERCISER & ON LINE TEST
MD-11-DZDJB-E

EP-DZDJB-E-DL-B
COPYRIGHT © 72-77
FICHE 1 OF 1

OCT 1977
digital
MADE IN USA

The image shows a grid of 40 small, illegible data tables or charts arranged in 10 rows and 4 columns on the left side of the page. Each cell in the grid contains a small, dense table of numbers and text, which is too small to read. The tables appear to be organized in a structured manner, possibly representing test results or performance metrics. The right side of the page is mostly blank, with some faint, illegible markings.

B01

EJF1DZDPR25B0411

00010000 771026

IDENTIFICATION

48HOR1DZDJBESG

00010000

771026
SEG 0001

PRODUCT CODE: MAINDEC-11-DZDJB-E-D
PRODUCT NAME: DJ11 EXERCISER AND ON-LINE TESTS
PROGRAM DATE: MAY 1977
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.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1972, 1977 BY DIGITAL EQUIPMENT CORPORATION

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 BK, except absolute Loader.

2.3 Preliminary Programs

MAINDEC-11-DZDJA 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 00eop 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 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. ? 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 (CNTL 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 d11 is will set "done". the subroutine also adjusts the character counters to ensure that the maximum number of characters to be transferred 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 D - 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 8: ", at which time the operator enters "1", "2", or "3" depending on the sub-program he wishes to run.

If SW<9> is on a 1, the program will TYPE OUT " N SELECT

LINES = 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 ICR 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 DJ11'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.

9. MISCELLANEOUS

9.1 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

9.2 Stack Pointer

Stack is initially set to 1200

9.3 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.

9.4 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 jumpered 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 PROG1.

If the silo alarm level is set for anything other than 1 char. (all 3 jumpers wired in). You must press the key on the terminal, under test, that many times before an echo occurs. IE. if the silo alarm level is set for 5 you must press the key 5 times before an echo back.

2506		SWITCH SETTINGS
2574		DJ11 SPECIFICATIONS
2669		SET UP AREA
2696		DJ11 MAPPING ROUTINE
2721		DJ11 VECTOR MAPPING ROUTINE
2777		CHARACTER LENGTH MAPPING ROUTINE
2831		PROGRAM SELECTION
2869		RESTART POINT
2885	PROG1:	OFF-LINE EXERCISER
2980		BACKGROUND MONITOR
3074		ISR LINKERS
3088		TRANSMITTER ISR
3123		RECEIVER ISR
3164		DATA TABLES
3212	PROG2:	ON-LINE EXERCISER (TRANSMIT LAST CHARACTER RECEIVED)
3281		FOREGROUND ROUTINE
3296		ISR LINKERS
3310		TRANSMITTER ISF.
3354		RECEIVER ISR
3407	PROG3:	ECHO EXERCISER
3486		FOREGROUND ROUTINE
3516		ISR LINKERS
3530		TRANSMITTER ISR
3567		RECEIVER ISR
3638		BELL AND SCOPE ROUTINE
3641		HLT ROUTINE (ERROR TIMEOUT)
3654		OCTAL NUMBER INPUT ROUTINE
3692		TTY INPUT ROUTINE
3693		OCTAL DUMP OF A WORD
3695		POWER DOWN AND UP ROUTINES
3701		IOT HANDLER
3723		TYPE ROUTINE

.ENABLE ABS
.ENABLE AMA
;PROGRAM BY KEN CHAPMAN
;MODIFIED BY DAVID L. ADAMS MAY 1977

2565
2566
2567
2568
2569
2570
2571
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
2572
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610

100000
040000
020000
010000
004000
002000

001000
000400

SW	SWITCH	USE
SW15=	100000	; HALT ON ERROR
SW14=	40000	; NOT USED
SW13=	20000	; INHIBIT ERROR TIMEOUTS
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 TIME ONLY)
		; Sw/O: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

MO1

MAINDEC-11-DZDJB-E-D
DZDJB.E.P11 08-SEP-77 15:07

DJ11 EXERCISER AND ON-LINE TESTS
DJ11 SPECIFICATIONS

MACY11 30(1046) 08-SEP-77 15:13 PAGE 44-2

SEG 0012

```

2625      000000 000000 000000      . = 0 ; TRAP CATCHER IN LOCATIONS 0 THRU 776
(1) 000000 000000 000000      0,0 ; LOCATIONS 0 AND 2 CONTAIN "HALT" INSTRUCTIONS
(1) ; LOCATIONS 4 THRU 56 CONTAIN ".+2" AND "HALT" IN EVERY VECTOR
(1) ; LOCATIONS 60 THRU 776 CONTAIN ".+2" AND "IOT" IN EVERY VECTOR
(1)
(1)
(1)
(1)
(1)
(1)
(1) 000046 000046 013574      . = 46
(1) 000046 013574      SENDAD
(1)
(1)
(1) 000174 000000      . = 174
(1) 000174 000000      DISPREG:0
(1) 000176 000000      SWREG: 0
(1)
(1)
(1) 000200 000200 005312      . = 200
(1) 000200 000137 005312      JMP BEGIN ; 200 ALWAYS IS THE STARTING ADDRESS
2626      001000 001000 006310      . = 1000
2627 001000 000137 006310      JMP RESTAR ; RESTART ADDRESS
2628
2629
2630 001200 000000      . = 1200
2631 001202 000000      ICNT: 0 ; ITERATION COUNT-LH, TEST NO.-RH
2632 001204 000000 000000      ERRORS: 0 ; ERROR COUNT REGISTER
2633      PCNT: 0,0 ; PASS COUNT REGISTER
(1) 001210 177570      SWR: 177570
(1) 001212 177570      DISPLAY:177570
(1)
2634 001214 000000      SAVIT: 0
2635 001216 000020      TIMES: 20 ; MINIMUM NUMBER OF MESSAGES (PROG1)
2638 001220 000000      SVSW0: OPEN ; MAP OF LINES SELECTED, DJ11 #0
(1) 001222 000000      SVSW1: OPEN ; MAP OF LINES SELECTED, DJ11 #1
(1) 001224 000000      SVSW2: OPEN ; MAP OF LINES SELECTED, DJ11 #2
(1) 001226 000000      SVSW3: OPEN ; MAP OF LINES SELECTED, DJ11 #3
(1) 001230 000000      SVSW4: OPEN ; MAP OF LINES SELECTED, DJ11 #4
(1) 001232 000000      SVSW5: OPEN ; MAP OF LINES SELECTED, DJ11 #5
(1) 001234 000000      SVSW6: OPEN ; MAP OF LINES SELECTED, DJ11 #6
(1) 001236 000000      SVSW7: OPEN ; MAP OF LINES SELECTED, DJ11 #7
(1) 001240 000000      SVSW10: OPEN ; MAP OF LINES SELECTED, DJ11 #10
(1) 001242 000000      SVSW11: OPEN ; MAP OF LINES SELECTED, DJ11 #11
(1) 001244 000000      SVSW12: OPEN ; MAP OF LINES SELECTED, DJ11 #12
(1) 001246 000000      SVSW13: OPEN ; MAP OF LINES SELECTED, DJ11 #13
(1) 001250 000000      SVSW14: OPEN ; MAP OF LINES SELECTED, DJ11 #14
(1) 001252 000000      SVSW15: OPEN ; MAP OF LINES SELECTED, DJ11 #15
(1) 001254 000000      SVSW16: OPEN ; MAP OF LINES SELECTED, DJ11 #16
(1) 001256 000000      SVSW17: OPEN ; MAP OF LINES SELECTED, DJ11 #17
2639 001260 000000      MARK: 0
2640 001262 000030      BUFSIZ: 30 ; RECEIVE BUFFER SIZE (PROG3)
2641 001264 000000      BUFEND: 0 ; LAST ADDR OF PROG3 BUFFER
2642 001266 000001      UNITS: 1 ; NUMBER OF UNITS ON THE SYSTEM
2643 001270 160010      DEVAOR: 160010 ; FIRST DEVICE ADDRESS
2644 001272 000300      VECADR: 300 ; FIRST VECTOR ADDRESS
2645 001274 000240      RCVLVL: 240 ; RECEIVER BR LEVEL = 5
2646 001276 000240      XMTLVL: 240 ; TRANSMITTER BR LEVEL = 5
2647 001300 000000      ALMFLG: 0 ; SILO ALARM LEVEL FLAG
2648 001302 000000      TIMERA: 0 ; TIME COUNTERS
2649 001304 000000      TIMERB: 0

```

NO1

MAINDEC-11-DZDJB-E-D
DZDJB.E.P11 08-SEP-77 15:07

DJ11 EXERCISER AND ON-LINE TESTS
DJ11 SPECIFICATIONS

MACY11 30(1046) 08-SEP-77 15:13 PAGE 44-3

SEQ 0013

2650 001306 000000
2651 001310 000000
2652
2653
2654
2655
2656
2657 001312 000400
2658
2659 002312 000400
2660
2661
2662 003312 000400
2663 003313
2664
2665 004312 000400
2666 004313

COUNT: 0
SUM: 0

;VALUE OF THE SILO ALARM LEVEL

;TABLES

XMTTAB: .BLKW 400

;TRANSMIT DATA POINTER TABLE

RCVTAB: .BLKW 400

;RECEIVE DATA POINTER TABLE (PROG1)
;RECEIVE DATA TABLE (PROG2 AND PROG3)

XMTCNT: .BLKW 400

;TRANSMIT DATA COUNTER

RCVCNT=XMTCNT+1

;RECEIVE DATA COUNTER

MASK: .BLKW 400

;CHARACTER MASK TABLE

CNTTAB=MASK+1

;ITERATION COUNT AND FLAGS

```

2671
2672 005312 012706 001200      BEGIN:  MOV    #STACK, SP      ;SET UP STACK POINTER
2673 005316 004737 015210      JSR    PC,SUSARR
2674 005322 012700 000020      MOV    #20, R0
2675 005326 012720 014672      MOV    #IOTRAP, (R0)+      ;IOT VECTOR (20)
2676 005332 012720 000340      MOV    #340, (R0)+
2677 005336 012720 014530      MOV    #PDOWNS, (R0)+      ;POWER FAIL VECTOR (24)
2678 005342 012720 000340      MOV    #340, (R0)+
2679 005346 012720 013610      MOV    #EMTS, (R0)+        ;EMT VECTOR (30)
2680 005352 012720 000340      MOV    #340, (R0)+
2681 005356 005037 001202      CLR    ERRORS              ;CLEAR ERROR COUNTER
2682 005362 005037 001204      CLR    PCNT                ;CLEAR PASS COUNTER
2683 005366 005037 001206      CLR    PCNT+2
2684 005372 012700 000300      MOV    #300, R0            ;START OF FLOATING VECTOR AREA
2685 005376 005720      2S:  TST    (R0)+              ;UPDATE POINTER
2686 005400 010060 177776      MOV    R0, -2(R0)          ;PUT ".+2" IN EACH VECTOR
2687 005404 012720 000004      MOV    #IOT, (R0)+        ;AND "IOT"
2688 005410 022700 001000      CMP    #1000, R0          ;CHECK FOR END OF FLOATING VECTOR AREA
2689 005414 003370      BGT    2S                  ;BRANCH IF MORE
2690 005416 005037 001300      CLR    ALMFLG              ;CLEAR THE ALARM FLAG
2691 005422 012707 000400 010210      MOV    #256, CNTNIT        ;SET MAXIMUM SIZE VALUES
2692 005430 012737 000106 010214      MOV    #70, CNTNIT+4      ;FOR PROG #1
2693 005436 012737 000106 010220      MOV    #70, CNTNIT+10
2694 005444 012737 000001 001216      MOV    #1, TIMES          ;SET FOR QV ON FIRST PASS
2698
2699
2700      ;*****
2701      ;ROUTINE TO MAP ALL THE DJ11'S ON THE SYSTEM
2702      ;*****
2703 005452 013700 001270      DJMAP: MOV    DEVAR, R0      ;GET FIRST FLOATING ADDRESS
2704 005456 012702 000001      MOV    #1, R2              ;COUNTER FOR DJ11'S
2705 005462 012737 000002 000006      MOV    #RTI, #06          ;RTI WHEN TIME-OUT
2706 005470 005001      5S:  CLR    R1              ;SET UP COUNTER
2707 005472 000261      1S:  SEC                  ;SET CARRY
2708 005474 005710      TST    (R0)                ;CHECK FOR A DEVICE
2709 005476 103404      BCS    7S                  ;BRANCH IF NONE
2710 005500 062700 000010      6S:  ADD    #10, R0          ;GO TO NEXT DEVICE ADDRESS
2711 005504 005201      INC    R1                  ;COUNT DJ11'S
2712 005506 000771      BR     1S                  ;LOOK FOR MORE
2713
2714 005510 005037 000006      7S:  CLR    #06              ;RESTORE TIMEOUT VECTOR
2715 005514 010137 001256      MOV    R1, UNITS           ;SAVE COUNT
2716 005520 001005      BNE    GETVEC
2717 005522 000004 015567      TYPE, MSG01                ;TYPE "NO DJ11'S!"
2718 005526 000000      HALT                       ;FATAL ERROR
2719 005530 000137 013522      JMP    #DONE               ;RESTART
2723
2724      ;*****
2725      ;ROUTINE TO DETERMINE VECTOR ADDRESSES OF DJ11'S
2726      ;*****
2727
2728 005534 013746 000020      GETVEC: MOV    #20, -(SP)   ;SAVE IOT VECTOR
2729 005540 012737 005570 000020      MOV    #15, #20           ;RESET IOT VECTOR
2730 005546 013701 001270      MOV    DEVAR, R1          ;FIRST DJ ADDRESS
2731 005552 012711 040400      MOV    #40400, (R1)       ;SET CSR
2732

```

```

2733                                     ;BIT14= TRANS INTERRUPT ENABLE
2734 005556 012761 000001 000004      MOV    #1,    4(R1)      ;TCR LINE 0
2735 005564 000001                    WAIT                   ;WAIT FOR AN INTERRUPT
2736 005566 000407                    BR     3$              ;CONTINUE AFTER INTERUPT
2737
2738 005570 011602                    1$:  MOV    (SP),    R2      ;SAVE VECTOR ADR. (+4)
2739 005572 162716 000010            SUB    #10,    (SP)      ;REPOSITION ADR TO RCV. VEC.
2740 005576 011637 001272            MOV    (SP),    VECADR   ;SAVE FIRST VECTOR
2741 005602 022626                    CMP    (SP)+,    (SP)+  ;RESET STACK FROM IOT
2742 005604 000002                    RTI                       ;RETURN FROM INITIAL INTERUPT
2743
2744 005606 012637 000020            3$:  MOV    (SP)+,    @#20  ;RESTORE IOT VECTOR
2745 005612 005742                    TST   -(R2)            ;POINT TO XMT. VEC. +2
2746 005614 013703 001266            MOV    UNITS,    R3      ;SET UP UNIT COUNTER
2747
2748                                     ;CHECK THAT VECTORS ARE CONTIGUOUS
2749
2750 005620 005061 000004            2$:  CLR    4(R1)          ;CLEAR TCR
2751 005624 005011                    CLR    (R1)            ;CLEAR CSR
2752 005626 012712 000004            MOV    #IOT,    (R2)    ;RESTORE IOT TO XMT. VEC.+2
2753 005632 005303                    DEC    R3              ;CHECK FOR MORE DJ11'S
2754 005634 001415                    BEQ    REPORT          ;BRANCH IF DONE
2755 005636 062701 000010            ADD    #10,    R1      ;UPDATE DJ ADR. POINTER
2756 005642 062702 000010            ADD    #10,    R2      ;UPDATE VECTOR POINTER
2757 005646 012712 000002            MOV    #RTI,    (R2)   ;RTI ON INTERRUPT
2758 005652 012711 040400            MOV    #40400,    (R1) ;SET CSR
2759 005656 012761 000001 000004    MOV    #1,    4(R1)    ;TCR LINE 0
2760 005664 000001                    WAIT                   ;WAIT FOR AN INTERRUPT
2761 005666 000754                    BR     2$
2762
2763                                     ;REPORT CONFIGURATION
2764
2765 005670 032777 020000 173312    REPORT: BIT    #BIT13, @SWR ;CHECK FOR INHIBIT TYP0UT
2766 005676 001026                    BNE    GETLEN          ;SKIP REPORT IF SET
2767 005700 000004 015402            TYPE,  MSGMON
2768 005704 000004 015371            TYPE,  RETURN
2769 005710 000004 015451            TYPE,  MSGADR
2770 005714 013705 001270            MOV    DEVADR, TTY     ;TYPE DEVADR IN OCTAL
2771 005720 004737 014346            JSR    PC, PRINTR      ;TYPE LEADING ZERO'S
2772 005724 000004 015501            TYPE,  MSGVEC
2773 005730 013705 001272            MOV    VECADR, TTY    ;TYPE VECADR IN OCTAL
2774 005734 004737 014356            JSR    PC, PRINTS     ;AND SUPPRESS LEADING ZERO'S
2775 005740 000004 015525            TYPE,  MSGNUM
2776 005744 013705 001266            MOV    UNITS, TTY     ;TYPE UNITS IN OCTAL
2777 005750 004737 014356            JSR    PC, PRINTS     ;AND SUPPRESS LEADING ZERO'S

```

2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828

005754 022737 000176 001210
005762 001002
005764 004737 015106
005770 013700 001266
005774 013701 001270
006000 012702 000001
006004 012711 000415
006010 032711 000020
006014 001375
006016 010261 000004
006022 005711
006024 100376
006026 012761 000377 000006
006034 006302
006036 006302
006040 006302
006042 006302
006044 103364
006046 005061 000004
006052 062701 000010
006056 005300
006060 001347
006062 013700 001266
006066 013701 001270
006072 012702 004312
006076 012703 000004
006102 016104 000002
006106 100375
006110 010405
006112 000305
006114 042705 177760
006120 006305
006122 060205
006124 105104
006126 042704 177400
006132 010425
006134 010425
006136 010425
006140 010425
006142 005303
006144 001356
006146 062701 000010
006152 062702 000040
006156 005300
006160 001346

```
*****
:ROUTINE TO MAP CHARACTER LENGTHS
*****
GETLEN:  CMP    #SWREG, SWR
        BNE    6$
        JSR    PC, CNTLU
6$:      MOV    UNITS, R0      ;SET UP UNIT COUNTER
        MOV    DEVADR, R1    ;SET UP DEVICE ADDRESS POINTER
1$:      MOV    #1, R2       ;SET UP LINE MARKER
        MOV    #415, (R1)    ;RCV ENB, CMOS, MAINT., TRANS SCAN ENB
10$:     BIT    #BIT4, (R1)  ;WAIT FOR MOS TO CLEAR
        BNE    10$
2$:      MOV    R2, 4(R1)    ;TRANS CONTROL, ONE LINE AT A TIME
3$:      TST    (R1)        ;WAIT FOR TRANS READY
        BPL    3$
        MOV    #377, 6(R1)  ;SEND A RUBOUT
        ASL    R2           ;SKIP 4 LINES
        ASL    R2
        ASL    R2
        ASL    R2
        BCC    2$          ;BRANCH BACK IF MORE LINES
        CLR    4(R1)       ;CLEAR TCR
        ADD    #10, R1     ;UPDATE POINTER TO NEXT UNIT
        DEC    R0         ;CHECK FOR MORE UNITS
        BNE    1$
        MOV    UNITS, R0   ;SET UP UNIT COUNTER
        MOV    DEVADR, R1  ;SET UP DEVICE ADDRESS POINTER
        MOV    #MASK, R2   ;SET UP CHAR LEN TABLE POINTER
4$:      MOV    #4, R3      ;SET UP CHAR COUNTER
5$:      MOV    2(R1), R4   ;SAVE AND CHECK CHAR PRESENT
        BPL    5$
        MOV    R4, R5     ;DUP DATA
        SWAB   R5         ;LINE # IN LOW BYTE
        BIC    #177760, R5 ;CLEAR ALL BUT LINE #
        #2
        ADD    R2, R5     ;MAKE POINTER TO CHAR TABLE
        COMB   R4         ;MAKE DATA INTO MASK
        BIC    #177400, R4 ;CLEAR UPPER BYTE
        MOV    R4, (R5)+  ;SAVE THE MASK
        MOV    R4, (R5)+  ;SAVE THE MASK
        MOV    R4, (R5)+  ;SAVE THE MASK
        MOV    R4, (R5)+  ;SAVE THE MASK
        DEC    R3         ;COUNT TO 4
        BNE    5$
        ADD    #10, R1    ;ADDRESS POINTER TO NEXT DJ
        ADD    #40, R2    ;CHAR LEN TABLE POINTER
        DEC    R0         ;COUNT UNITS
        BNE    4$        ;BRANCH BACK IF MORE
```

2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882

006162 005737 000042
006166 001040
006170 000004 015550
006174 004537 014002
006200 006332
006202 001367
006204 032777 000400 172776
006212 001426
006214 005000
006216 012701 001220
006222 000004 015371
006226 000004 015346
006232 010005
006234 004737 014356
006240 000004 015351
006244 004537 014002
006250 001214
006252 013721 001214
006256 005200
006260 020037 001266
006264 001356
006266 000410
006270 013700 001266
006274 012701 001220
006300 012721 177777
006304 005300
006306 001374
006310 013700 006332
006314 001722
006316 022700 000003
006322 103717
006324 006300
006326 000170 006332
006332
006332 000001
006334 006342
006336 010762
006340 012166

```
*****
: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      2042          ;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, 2SWR   ;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,   NUM
        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  RESTART 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
```

```

*****
PROGRAM !: 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.
*****

```

2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942

```

006342 000005
006344 012706 001200
006350 052737 000340 177776
006356 012701 001312
006362 012702 001400
006366 005021
006370 005302
006372 001375
006374 012702 000400
006400 005201
006402 105021
006404 005302
006406 001374

006410 005000
006412 013701 001270
006416 013702 001272
006422 012703 007354
006426 010322
006430 013722 001274
006434 022323
006436 010113
006440 062723 000002
006444 005723
006446 010322
006450 013722 001276
006454 022323
006456 010123
006460 005011
006462 052711 050510

006466 032711 000020
006472 001375
006474 005737 001300
006500 001002
006502 004737 007006

```

```

PROG1: 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 #1400, 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

PIINIT: CLR R0
        MOV DEVAOR, R1
        MOV VECADR, R2
        MOV #RISR0, R3
11$: MOV R3, (R2)+ ;SET UP RECEIVER INTERRUPT VECTOR
     MOV RCVLVL, (R2)+
     CMP (R3)+, (R3)+ ;ADD 4 TO R3
     MOV R1, (R3) ;ADDRESS OF CSR
     ADD #2, (R3)+ ;ADDRESS OF RBUF
     TST (R3)+
     MOV R3, (R2)+ ;SET UP TRANSMITTER INTERRUPT VECTOR
     MOV XMTLVL, (R2)+
     CMP (R3)+, (R3)+
     MOV R1, (R3)+ ;ADDRESS OF CSR
     CLR (R1) ;CLEAR CSR
     BIS #50510, (R1) ;SET UP CSR
     ;BIT3 = CLEAR MOS
     ;BIT6 = RECEIVER INTERRUPT ENABLE
     ;BIT8 = TRANSMITTER SCAN ENABLE
     ;BIT12 = STATUS ENABLE
     ;BIT14 = TRANSMITTER INTERRUPT ENABLE
13$: BIT #BIT4, (R1) ;CHECK FOR MOS TO CLEAR
     BNE 13$
     TST ALMFLG ;HAS THE SILO ALARM LEVEL BEEN CHECKED?
     BNE 10$ ;YES
     JSR PC, ALMCK ;NO, GO DO IT

```

PROG1: OFF-LINE EXERCISER

```

2943 006506 006300 10$: ASL R0 ;UNIT * 2
2944 006510 016061 001220 000004 MOV SVSWO(R0),4(R1) ;SET TCR BITS (CSR + 4)
2945 006516 006200 ASR R0 ;RESTORE UNIT COUNTER
2946 006520 012737 000001 001260 MOV #1, MARK ;SET UP MARKER
2947 006526 017705 172456 MOV #SWR, R5 ;GET SWITCH SETTINGS
2948 006532 042705 1770 BIC #177770,R5 ;MASK MESSAGE #
2949 006536 006305 ASL R5
2950 006540 006305 ASL R5
2951 006542 012304 MOV (R3)+, R4 ;SET UP OFFSET TO TABLES
2952 006544 033761 001260 000004 14$: BIT MARK, 4(R1) ;CHECK FOR LINE SELECTED IN TCR
2953 006552 001414 BEQ 15$
2954 006554 016564 010206 001312 MOV ADNIT(5),XMTTAB(4)
2955 006562 016564 010206 002312 MOV ADNIT(5),RCVTAB(4)
2956 006570 116564 010210 003312 MOVB CNTNIT(5),XMTCNT(4)
2957 006576 116564 010210 003313 MOVB CNTNIT(5),RCVCNT(4)
2958 006604 005724 15$: TST (R4)+ ;INC OFFSET TO NEXT LINE
2959 006606 006337 001260 ASL MARK
2960 006612 103354 BCC 14$
2961 006614 032777 001000 172366 BIT #BIT9, #SWR ;CHECK FOR ON-LINE
2962 006622 001024 BNE 21$ ;BRANCH IF ON-LINE
2963 006624 052711 000014 BIS #14, (R1) ;SET THE MAINTENANCE BIT AND CLR MOS
2964 006630 032711 000020 20$: BIT #BIT4, (R1) ;WAIT FOR MOS TO CLEAR
2965 006634 001375 BNE 20$
2966 006636 052711 000001 BIS #1, (R1) ;TURN ON RCV ENABLE
2967 006642 062701 000010 12$: ADD #10, R1 ;POINT TO THE NEXT CSR
2968 006646 005200 INC R0
2969 006650 020037 001266 CMP R0, UNITS
2970 006654 001264 BNE 11$
2971 006656 012737 000001 001300 MOV #1,ALMFLG ;SET THE ALARM LEVEL FLAG
2972 006664 042737 000140 177776 BIC #140, #PS ;LOWER PROCESSOR PRIORITY
2973 006672 000406 BR FORGND ;GO DO IT
2974
2975 006674 052711 000001 21$: BIS #1, (R1) ;TURN ON RCV EN
2976 006700 005761 000002 22$: TST 2(R1) ;CLEAR JUNK OUT OF THE RBUF
2977 006704 100775 BMI 22$
2978 006706 000755 BR 12$
2982
2983 ;*****
2984 ;PROG1 BACKGROUND PROGRAM TO MONITOR TABLES
2985 ;*****
2986 ; NOTE - PROGRAM MAY HANG IN A LOOP.
2987 ; IF THIS HAPPENS, RUN DZDJA.
2988
2989 006710 012701 003312 FORGND: MOV #XMTCNT,R1
2990 006714 012702 000400 MOV #400,R2
2991 006720 105711 21$: TSTB (R1) ;CHECK FOR COUNT TABLE CLR
2992 006722 001376 BNE 21$ ;BRANCH IF NOT
2993 006724 062701 000002 ADD #2,R1 ;GO TO NEXT LINE ENTRY
2994 006730 005302 DEC R2 ;COUNT LINES
2995 006732 001372 BNE 21$ ;BRANCH IF MORE LINES
2996 006734 012701 003313 MOV #RCVCNT,R1
2997 006740 012702 000400 MOV #400,R2
2998 006744 121137 001306 22$: CMPB (R1),COUNT ;IS # OF CHAR LEFT IN RBUF LESS THAN
2999 ;THE SILO ALARM LEVEL
3000 ;IF NO WAIT FOR THEM
3001 006750 003375 BGT 22$
3002 006752 062701 000002 ADD #2,R1 ;IF YES IGNORE THEM

```

3002	006756	005302				DEC	R2		;COUNT LINES
3003	006760	001371				BNE	22\$;BRANCH IF MORE LINES
3004	006762	005337	001216			DEC	TIMES		;DO THIS AGAIN?
3005	006766	001402				BEQ	23\$;NO, GET OUT
3006	006770	000137	006342			JMP	PROG1		
3007	006774	012737	000020	001216	23\$:	MOV	#20, TIMES		;DO IT 16 TIMES THE NEXT TIME
3008	007002	000137	013522			JMP	#DONE		;SKIP ISR'S
3009	007006	012761	000001	000004	ALMCK:	MOV	#14(R1)		;SET LINE 0 IN THE TCR
3010	007014	052711	000004			BIS	#BIT2, (R1)		;SET THE MAINT BIT
3011	007020	052711	000001			BIS	#1, (R1)		;SET RCV EN AFTER MAINTENANCE BIT
3012	007024	005037	001306			CLR	COUNT		
3013	007030	005037	001310			CLR	SUM		
3014	007034	005037	001302			CLR	TIMERA		
3015	007040	012737	000200	001304	2\$:	MOV	#200, TIMERB		;SET UP TIME CONSTANTS
3016	007046	005711				TST	(R1)		;WAIT FOR TRANSFER READY BIT
3017	007050	100373				BPL	2\$		
3018	007052	112761	000377	000006		MOVB	#377, 6(R1)		;OUTPUT A CHAR TO TBUF
3019	007060	005237	001306			INC	COUNT		;COUNT EACH CHAR
3020	007064	105711			1\$:	TSTB	(R1)		;CHECK FOR DONE IN THE CSR
3021	007066	100405				BMI	3\$;IF SET GET OUT OF THE LOOP
3022	007070	004537	007320			JSR	R5, TIME		;GIVE DONE TIME TO SET
3023	007074	000773				BR	1\$;RETURN TO TEST FOR DONE AGAIN
3024	007076	000760				BR	2\$;RETURN TO OUTPUT ANOTHER CHAR
3025	007100	000742				BR	ALMCK		;ERROR RETURN TRY AGAIN
3026	007102	042711	000001		3\$:	BIC	#BIT0, (R1)		;TURN OFF RCV ENABLE
3027	007106	052711	000010			BIS	#BIT3, (R1)		;CLEAR MOS
3028	007112	022737	000001	001306		CMP	#1, COUNT		;IF SILO LEVEL SET FOR 1 THEN GET OUT
3029	007120	001437				BEQ	4\$		
3030	007122	063737	001306	001310	5\$:	ADD	COUNT, SUM		;GET THE LARGEST MULTIPLE OF THE
3031	007130	023727	001310	000106		CMP	SUM, #70.		;SILO ALARM LEVEL AND USE IT IN THE
3032	007136	002771				BLT	5\$;THREE SIZE LOCATIONS
3033	007140	001403				BEQ	6\$;IF EQUAL USE IT
3034	007142	163737	001306	001310		SUB	COUNT, SUM		;IF GREATER SUBTRACT ONE COUNT OFF
3035	007150	013737	001310	010214	6\$:	MOV	SUM, CNTNIT+4		;AS CLOSE TO 70 AS POSSIBLE
3036	007156	013737	001310	010220		MOV	SUM, CNTNIT+10		
3037	007164	063737	001306	001310	7\$:	ADD	COUNT, SUM		;CONTINUE TO A COUNT OF 256
3038	007172	023727	001310	000377		CMP	SUM, #377		
3039	007200	002771				BLT	7\$		
3040	007202	163737	001306	001310		SUB	COUNT, SUM		
3041	007210	013737	001310	010210	10\$:	MOV	SUM, CNTNIT		;AS CLOSE TO 256 AS POSSIBLE
3042	007216	000411				BR	11\$;ALL SET GET OUT
3043	007220	012737	000377	010210	4\$:	MOV	#377, CNTNIT		;PUT SIZE TO MAX VALUE
3044	007226	012737	000106	010214		MOV	#70., CNTNIT+4		
3045	007234	012737	000106	010220		MOV	#70., CNTNIT+10		
3046	007242	042711	000004		11\$:	BIC	#BIT2, (R1)		;TURN OFF THE MAINT BIT
3047	007246	032711	000020		12\$:	BIT	#BIT4, (R1)		;WAIT FOR MOS TO CLEAR
3048	007252	001375				BNE	12\$		
3049	007254	004737	015270			JSR	PC, KBDINT		;GET THE SWITCH REGISTER
3050	007260	032777	010000	171722		BIT	#SW12, JSWR		;PRINT ALARM LEVEL?
3051	007266	001413				BEQ	13\$;NO
3052	007270	000004	015676			TYPE,	MALARM		
3053	007274	010105				MOV	R1, TTY		;YES, PRINT CSR FIRST
3054	007276	004737	014346			JS	PC, PRINTR		
3055	007302	000004	015672			TYPE,	MSGDAS		
3056	007306	013705	001306			MOV	COUNT, TTY		;PRINT ALARM LEVEL
3057	007312	004737	014346			JSR	PC, PRINTR		

```

3058 007316 000207          13$:  RTS      PC
3059
3060
3061
3062 007320 105237 001302    TIME:  INCB    TIMERA      ; INCREMENT THROUGH ONE WORD
3063 007324 001012          BNE      1$          ; GO TEST FOR DONE AGAIN
3064 007326 005337 001304    DEC      TIMERB     ; MAKE TIMERB LARGER IF FAST PROCESSOR
3065 007332 001007          BNE      1$
3066 007334 023727 001306 000022  CMP      COUNT, #22 ; HAVE OUTPUTTED 18 TIMES
3067 007342 001002          BNE      2$          ; NO, GO OUTPUT ANOTHER CHAR
3068 007344 104001          HLT+1          ; YES, DONE DID NOT SET AFTER 18 OUTPLTS
3069
3070 007346 005725          TST      (R5)+      ; SET R5 FOR ERROR RETURN
3071 007350 005725          2$:  TST      (R5)+      ; SET R5 FOR NEXT OUTPUT RETURN
3072 007352 000205          1$:  RTS      R5      ; RETURN FROM ABOVE OR RETEST DONE
3076
3077
3078
3079
3080

```

```

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

```

```

3086 007354 004037 010070    RISR0:  JSR      RO, RCVISR
      (1) 007360 160012 000000      .WORD    <160012+(0*10)>, <40*0>
      (1) 007364 004037 007754    XISR0:  JSR      RO, XMTISR
      (1) 007370 160010 000000      .WORD    <160010+(0*10)>, <40*0>
      (1) 007374 004037 010070    RISR1:  JSR      RO, RCVISR
      (1) 007400 160022 000040      .WORD    <160012+(1*10)>, <40*1>
      (1) 007404 004037 007754    XISR1:  JSR      RO, XMTISR
      (1) 007410 160020 000040      .WORD    <160010+(1*10)>, <40*1>
      (1) 007414 004037 010070    RISR2:  JSR      RO, RCVISR
      (1) 007420 160032 000100      .WORD    <160012+(2*10)>, <40*2>
      (1) 007424 004037 007754    XISR2:  JSR      RO, XMTISR
      (1) 007430 160030 000100      .WORD    <160010+(2*10)>, <40*2>
      (1) 007434 004037 010070    RISR3:  JSR      RO, RCVISR
      (1) 007440 160042 000140      .WORD    <160012+(3*10)>, <40*3>
      (1) 007444 004037 007754    XISR3:  JSR      RO, XMTISR
      (1) 007450 160040 000140      .WORD    <160010+(3*10)>, <40*3>
      (1) 007454 004037 010070    RISR4:  JSR      RO, RCVISR
      (1) 007460 160052 000200      .WORD    <160012+(4*10)>, <40*4>
      (1) 007464 004037 007754    XISR4:  JSR      RO, XMTISR
      (1) 007470 160050 000200      .WORD    <160010+(4*10)>, <40*4>
      (1) 007474 004037 010070    RISR5:  JSR      RO, RCVISR
      (1) 007500 160062 000240      .WORD    <160012+(5*10)>, <40*5>
      (1) 007504 004037 007754    XISR5:  JSR      RO, XMTISR
      (1) 007510 160060 000240      .WORD    <160010+(5*10)>, <40*5>
      (1) 007514 004037 010070    RISR6:  JSR      RO, RCVISR
      (1) 007520 160072 000300      .WORD    <160012+(6*10)>, <40*6>
      (1) 007524 004037 007754    XISR6:  JSR      RO, XMTISR
      (1) 007530 160070 000300      .WORD    <160010+(6*10)>, <40*6>
      (1) 007534 004037 010070    RISR7:  JSR      RO, RCVISR
      (1) 007540 160102 000340      .WORD    <160012+(7*10)>, <40*7>
      (1) 007544 004037 007754    XISR7:  JSR      RO, XMTISR
      (1) 007550 160100 000340      .WORD    <160010+(7*10)>, <40*7>
      (1) 007554 004037 010070    RISR10: JSR      RO, RCVISR
      (1) 007560 160112 000400      .WORD    <160012+(10*10)>, <40*10>
      (1) 007564 004037 007754    XISR10: JSR      RO, XMTISR
      (1) 007570 160110 000400      .WORD    <160010+(10*10)>, <40*10>

```

(1) 007574 004037 010070
 (1) 007600 160122 000440
 (1) 007604 004037 007754
 (1) 007610 160120 000440
 (1) 007614 004037 010070
 (1) 007620 160132 000500
 (1) 007624 004037 007754
 (1) 007630 160130 000500
 (1) 007634 004037 010070
 (1) 007640 160142 000540
 (1) 007644 004037 007754
 (1) 007650 160140 000540
 (1) 007654 004037 010070
 (1) 007660 160152 000600
 (1) 007664 004037 007754
 (1) 007670 160150 000600
 (1) 007674 004037 010070
 (1) 007700 160162 000640
 (1) 007704 004037 007754
 (1) 007710 160160 000640
 (1) 007714 004037 010070
 (1) 007720 160172 000700
 (1) 007724 004037 007754
 (1) 007730 160170 000700
 (1) 007734 004037 010070
 (1) 007740 160202 000740
 (1) 007744 004037 007754
 (1) 007750 160200 000740

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

3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115

007754
 (2) 007754 010146
 (2) 007756 010246
 007760 012001
 007762 005711
 007764 100035
 007766 116102 000007
 007772 006302
 007774 061002
 007776 105762 003312
 010002 001410
 010004 117261 001312 00000E
 010012 105362 003312
 010016 005262 001312
 010022 000757
 010024 010346
 010026 005062 001312
 010032 161002
 010034 006202
 010036 005003
 010040 000261
 010042 006103
 010044 005302

 ;PROG1 TRANSMITTER INTERRUPT SERVICE ROUTINE
 ;*****

XMTISR:
 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 XMTCNT(2) ;TST FOR ZERO
 BEQ 2\$
 MOVB 2XMTTAB(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
 CLR R3
 3\$: SEC
 ROL R3
 DEC R2

```

3116 010046 100375          BPL      3$
3117 010050 040361 000004  BIC      R3,4(R1)      ;CLEAR TCR BIT FOR LINE
3118 010054 012603          MOV      (SP)+,R3      ;RESTORE R3
3119 010056 000741          BR       1$
3120 010060          4$:
(2) 010060 012602          MOV      (6)+,R2      ;POP STACK INTO R2
(2) 010062 012601          MOV      (6)+,R1      ;POP STACK INTO R1
(2) 010064 012600          MOV      (6)+,R0      ;POP STACK INTO R0
3121 010066 000002          RTI

```

```

;*****
;PROG1 RECEIVER INTERRUPT SERVICE ROUTINE
;*****

```

```

3130 010070          RCVISR:
(2) 010070 010146          MOV      R1,-(6)      ;PUSH R1 ON STACK
(2) 010072 010246          MOV      R2,-(6)      ;PUSH R2 ON STACK
(2) 010074 010346          MOV      R3,-(6)      ;PUSH R3 ON STACK
(2) 010076 010446          MOV      R4,-(6)      ;PUSH R4 ON STACK
3131 010100 012001          MOV      (R0)+,R1      ;GET RBUF ADDRESS
3132 010102 011102          1$: MOV      (R1),R2      ;READ THE DATA
3133 010104 100032          BPL      7$           ;BRANCH IF NO CHAR PRESENT
3134 010106 032702 070000  BIT      #70000,R2     ;CHECK FOR ERRORS
3135 010112 001403          BEQ      2$           ;BRANCH IF OK
3136 010114 104002          HLT+2      ;RECEIVER ERROR

```

```

3137          ;R1=RBUF ADDRESS
3138          ;R2=CONTENTS OF RBUF
3139          ;BIT12=PARITY ERROR
3140          ;BIT13=FRAMING ERROR
3141          ;BIT14=UART OVERRUN
3142 010116 042702 070000  2$: BIC      #70000, R2     ;CLEAR ERROR BITS FOR SPEED
3143 010122 010204          MOV      R2, R4      ;DUP THE RBUF
3144 010124 105004          CLRB    R4           ;CLEAR THE DATA
3145 010126 000304          SWAB   R4           ;LINE # TO LOW BYTE
3146 010130 106304          ASLB   R4           ;LINE # * 2, ALSO CLR CHAR PRESENT
3147 010132 061004          ADD     (R0),R4      ;ADD OFFSET
3148 010134 117402 002312  MOVB    @RCVTAB(R4),R3 ;GET EXPECTED DATA
3149 010140 046403 004312  BIC     MASK(4),R3     ;MASK CHARACTER LENGTH
3150 010144 120302          CMPB   R3,R2
3151 010146 001403          BEQ     3$           ;BRANCH IF OK
3152 010150 042703 177400  BIC     #177400,R3     ;MAKE SURE UPPER BYTE CLEAR
3153 010154 104003          HLT+3      ;DATA ERROR

```

```

3154          ;R1=RBUF ADDRESS
3155          ;R2=CONTENTS OF RBUF (DATA)
3156          ;R3=EXPECTED DATA
3157 010156 105364 003313  3$: DECB   RCVCNT(4)
3158 010162 001403          BEQ     7$
3159 010164 005264 002312  INC     RCVTAB(4)      ;UPDATE TABLE POINTER
3160 010170 000744          BR      1$           ;CONTINUE
3161 010172          7$:
(2) 010172 012604          MOV      (6)+,R4      ;POP STACK INTO R4
(2) 010174 012603          MOV      (6)+,R3      ;POP STACK INTO R3
(2) 010176 012602          MOV      (6)+,R2      ;POP STACK INTO R2
(2) 010200 012601          MOV      (6)+,R1      ;POP STACK INTO R1
(2) 010202 012600          MOV      (6)+,R0      ;POP STACK INTO R0
3162 010204 000002          RTI

```

3166
 3167
 3168
 3169
 3170
 3171 J10206 010252
 3172 010210 000377
 3173 010212 010652
 3174 010214 000106
 3175 010216 010244
 3176 010220 000106
 3177 010222 015734
 3178 010224 000001
 3179 010226 016334
 3180 010230 000001
 3181 010232 016734
 3182 010234 000001
 3183 010236 017334
 3184 010240 000001
 3185 010242 017734
 3186 010244 005015 177777 177777
 3187 010252 040
 3199 010253 041
 (2) 010254 042
 (2) 010255 043
 (2) 010256 044
 (2) 010257 045
 (2) 010260 046
 (2) 010261 047
 (2) 010262 050
 (2) 010263 051
 (2) 010264 052
 (2) 010265 053
 (2) 010266 054
 (2) 010267 055
 (2) 010270 056
 (2) 010271 057
 (2) 010272 060
 (2) 010273 061
 (2) 010274 062
 (2) 010275 063
 (2) 010276 064
 (2) 010277 065
 (2) 010300 066
 (2) 010301 067
 (2) 010302 070
 (2) 010303 071
 (2) 010304 072
 (2) 010305 073
 (2) 010306 074
 (2) 010307 075
 (2) 010310 076
 (2) 010311 077
 (2) 010312 100
 (2) 010313 101
 (2) 010314 102

```

;*****
;PROG1 DATA TABLES
;*****
ADRNIT: BINARY ;SW<2:0>=0 BINARY COUNT PATTERN
CNTNIT: 377 ;SIZE=256.
        PHRASE ;SW<2:0>=1 "THE QUICK SILVER GRAY FOX..."
        70. ;SIZE=70.
        SIXBIT ;SW<2:0>=2 040 THRU 137
        70. ;SIZE=70.
        END
        |
        | END+400
        |
        | END+1000
        |
        | END+1400
        |
        | END+2000
SIXBIT: .ASCII <15><12><377><377><377><377> ;CR-LF, FILLERS
BINARY: .BYTE 40
        .BYTE 41
        .BYTE 42
        .BYTE 43
        .BYTE 44
        .BYTE 45
        .BYTE 46
        .BYTE 47
        .BYTE 50
        .BYTE 51
        .BYTE 52
        .BYTE 53
        .BYTE 54
        .BYTE 55
        .BYTE 56
        .BYTE 57
        .BYTE 60
        .BYTE 61
        .BYTE 62
        .BYTE 63
        .BYTE 64
        .BYTE 65
        .BYTE 66
        .BYTE 67
        .BYTE 70
        .BYTE 71
        .BYTE 72
        .BYTE 73
        .BYTE 74
        .BYTE 75
        .BYTE 76
        .BYTE 77
        .BYTE 100
        .BYTE 101
        .BYTE 102

```

(R)	010315	103	.BYTE	103
(R)	010316	104	.BYTE	104
(R)	010317	105	.BYTE	105
(R)	010320	106	.BYTE	106
(R)	010321	107	.BYTE	107
(R)	010322	110	.BYTE	110
(R)	010323	111	.BYTE	111
(R)	010324	112	.BYTE	112
(R)	010325	113	.BYTE	113
(R)	010326	114	.BYTE	114
(R)	010327	115	.BYTE	115
(R)	010330	116	.BYTE	116
(R)	010331	117	.BYTE	117
(R)	010332	120	.BYTE	120
(R)	010333	121	.BYTE	121
(R)	010334	122	.BYTE	122
(R)	010335	123	.BYTE	123
(R)	010336	124	.BYTE	124
(R)	010337	125	.BYTE	125
(R)	010340	126	.BYTE	126
(R)	010341	127	.BYTE	127
(R)	010342	130	.BYTE	130
(R)	010343	131	.BYTE	131
(R)	010344	132	.BYTE	132
(R)	010345	133	.BYTE	133
(R)	010346	134	.BYTE	134
(R)	010347	135	.BYTE	135
(R)	010350	136	.BYTE	136
(R)	010351	137	.BYTE	137
(R)	010352	140	.BYTE	140
(R)	010353	141	.BYTE	141
(R)	010354	142	.BYTE	142
(R)	010355	143	.BYTE	143
(R)	010356	144	.BYTE	144
(R)	010357	145	.BYTE	145
(R)	010360	146	.BYTE	146
(R)	010361	147	.BYTE	147
(R)	010362	150	.BYTE	150
(R)	010363	151	.BYTE	151
(R)	010364	152	.BYTE	152
(R)	010365	153	.BYTE	153
(R)	010366	154	.BYTE	154
(R)	010367	155	.BYTE	155
(R)	010370	156	.BYTE	156
(R)	010371	157	.BYTE	157
(R)	010372	160	.BYTE	160
(R)	010373	161	.BYTE	161
(R)	010374	162	.BYTE	162
(R)	010375	163	.BYTE	163
(R)	010376	164	.BYTE	164
(R)	010377	165	.BYTE	165
(R)	010400	166	.BYTE	166
(R)	010401	167	.BYTE	167
(R)	010402	170	.BYTE	170
(R)	010403	171	.BYTE	171
(R)	010404	172	.BYTE	172

(2)	010405	173	.BYTE	173
(2)	010406	174	.BYTE	174
(2)	010407	175	.BYTE	175
(2)	010410	176	.BYTE	176
(2)	010411	177	.BYTE	177
(2)	010412	200	.BYTE	200
(2)	010413	201	.BYTE	201
(2)	010414	202	.BYTE	202
(2)	010415	203	.BYTE	203
(2)	010416	204	.BYTE	204
(2)	010417	205	.BYTE	205
(2)	010420	206	.BYTE	206
(2)	010421	207	.BYTE	207
(2)	010422	210	.BYTE	210
(2)	010423	211	.BYTE	211
(2)	010424	212	.BYTE	212
(2)	010425	213	.BYTE	213
(2)	010426	214	.BYTE	214
(2)	010427	215	.BYTE	215
(2)	010428	216	.BYTE	216
(2)	010431	217	.BYTE	217
(2)	010432	220	.BYTE	220
(2)	010433	221	.BYTE	221
(2)	010434	222	.BYTE	222
(2)	010435	223	.BYTE	223
(2)	010436	224	.BYTE	224
(2)	010437	225	.BYTE	225
(2)	010440	226	.BYTE	226
(2)	010441	227	.BYTE	227
(2)	010442	230	.BYTE	230
(2)	010443	231	.BYTE	231
(2)	010444	232	.BYTE	232
(2)	010445	233	.BYTE	233
(2)	010446	234	.BYTE	234
(2)	010447	235	.BYTE	235
(2)	010450	236	.BYTE	236
(2)	010451	237	.BYTE	237
(2)	010452	240	.BYTE	240
(2)	010453	241	.BYTE	241
(2)	010454	242	.BYTE	242
(2)	010455	243	.BYTE	243
(2)	010456	244	.BYTE	244
(2)	010457	245	.BYTE	245
(2)	010460	246	.BYTE	246
(2)	010461	247	.BYTE	247
(2)	010462	250	.BYTE	250
(2)	010463	251	.BYTE	251
(2)	010464	252	.BYTE	252
(2)	010465	253	.BYTE	253
(2)	010466	254	.BYTE	254
(2)	010467	255	.BYTE	255
(2)	010470	256	.BYTE	256
(2)	010471	257	.BYTE	257
(2)	010472	260	.BYTE	260
(2)	010473	261	.BYTE	261
(2)	010474	262	.BYTE	262

(U)	010	263	.BYTE	263
(U)	010	264	.BYTE	264
(U)	010	265	.BYTE	265
(U)	010	266	.BYTE	266
(U)	010	267	.BYTE	267
(U)	010	270	.BYTE	270
(U)	010	271	.BYTE	271
(U)	010	272	.BYTE	272
(U)	010	273	.BYTE	273
(U)	010	274	.BYTE	274
(U)	010	275	.BYTE	275
(U)	010	276	.BYTE	276
(U)	010	277	.BYTE	277
(U)	010	300	.BYTE	300
(U)	010	301	.BYTE	301
(U)	010	302	.BYTE	302
(U)	010	303	.BYTE	303
(U)	010	304	.BYTE	304
(U)	010	305	.BYTE	305
(U)	010	306	.BYTE	306
(U)	010	307	.BYTE	307
(U)	010	310	.BYTE	310
(U)	010	311	.BYTE	311
(U)	010	312	.BYTE	312
(U)	010	313	.BYTE	313
(U)	010	314	.BYTE	314
(U)	010	315	.BYTE	315
(U)	010	316	.BYTE	316
(U)	010	317	.BYTE	317
(U)	010	320	.BYTE	320
(U)	010	321	.BYTE	321
(U)	010	322	.BYTE	322
(U)	010	323	.BYTE	323
(U)	010	324	.BYTE	324
(U)	010	325	.BYTE	325
(U)	010	326	.BYTE	326
(U)	010	327	.BYTE	327
(U)	010	330	.BYTE	330
(U)	010	331	.BYTE	331
(U)	010	332	.BYTE	332
(U)	010	333	.BYTE	333
(U)	010	334	.BYTE	334
(U)	010	335	.BYTE	335
(U)	010	336	.BYTE	336
(U)	010	337	.BYTE	337
(U)	010	340	.BYTE	340
(U)	010	341	.BYTE	341
(U)	010	342	.BYTE	342
(U)	010	343	.BYTE	343
(U)	010	344	.BYTE	344
(U)	010	345	.BYTE	345
(U)	010	346	.BYTE	346
(U)	010	347	.BYTE	347
(U)	010	350	.BYTE	350
(U)	010	351	.BYTE	351
(U)	010	352	.BYTE	352

(R)	010565	353	.BYTE	353
(R)	010566	354	.BYTE	354
(R)	010567	355	.BYTE	355
(R)	010570	356	.BYTE	356
(R)	010571	357	.BYTE	357
(R)	010572	360	.BYTE	360
(R)	010573	361	.BYTE	361
(R)	010574	362	.BYTE	362
(R)	010575	363	.BYTE	363
(R)	010576	364	.BYTE	364
(R)	010577	365	.BYTE	365
(R)	010600	366	.BYTE	366
(R)	010601	367	.BYTE	367
(R)	010602	370	.BYTE	370
(R)	010603	371	.BYTE	371
(R)	010604	372	.BYTE	372
(R)	010605	373	.BYTE	373
(R)	010606	374	.BYTE	374
(R)	010607	375	.BYTE	375
(R)	010610	376	.BYTE	376
(R)	010611	377	.BYTE	377
(R)	010612	000	.BYTE	000
(R)	010613	001	.BYTE	001
(R)	010614	002	.BYTE	002
(R)	010615	003	.BYTE	003
(R)	010616	004	.BYTE	004
(R)	010617	005	.BYTE	005
(R)	010620	006	.BYTE	006
(R)	010621	007	.BYTE	007
(R)	010622	010	.BYTE	010
(R)	010623	011	.BYTE	011
(R)	010624	012	.BYTE	012
(R)	010625	013	.BYTE	013
(R)	010626	014	.BYTE	014
(R)	010627	015	.BYTE	015
(R)	010630	016	.BYTE	016
(R)	010631	017	.BYTE	017
(R)	010632	020	.BYTE	020
(R)	010633	021	.BYTE	021
(R)	010634	022	.BYTE	022
(R)	010635	023	.BYTE	023
(R)	010636	024	.BYTE	024
(R)	010637	025	.BYTE	025
(R)	010640	026	.BYTE	026
(R)	010641	027	.BYTE	027
(R)	010642	030	.BYTE	030
(R)	010643	031	.BYTE	031
(R)	010644	032	.BYTE	032
(R)	010645	033	.BYTE	033
(R)	010646	034	.BYTE	034
(R)	010647	035	.BYTE	035
(R)	010650	036	.BYTE	036
(R)	010651	037	.BYTE	037

010652 005015 177777 177777 PHRASE: .ASCII (15)<(12)<(377)<(377)<(377)<(377)
 010660 044124 020105 052521 .ASCIZ "THE QUICK SILVER GRAY FOX JUMPED OVER 9.876.543.210.0 LAZY DOGS!"

D03

MAINDEC-11-DZDJB-E-D
DZDJB.P11 08-SEP-77

DJ11 EXERCISER AND ON-LINE TESTS
15:07 DATA TABLES

MACY11 30(1046) 08-SEP-77 15:13 PAGE 48-11

SEG 0029

010666	041511	020113	044523
010674	053114	051105	043440
010702	040522	020131	047506
010710	020130	052512	050115
010716	042105	047440	042526
010724	020122	026071	033470
010732	026066	032065	026063
010740	030462	027060	020060
010746	040514	054532	042040
010754	043517	020523	000
010762			

3209

.EVEN

E03

MAINDEC-11-DZDJB-E-D
DZDJB.E.P11 08-SEP-77

DJ11 EXERCISER AND ON-LINE TESTS
15:07

MACY11 30(1046) 08-SEP-77 15:13 PAGE 49
PROG2: ON-LINE EXERCISER (TRANSMIT LAST CHARACTER RECEIVED)

SEG 0030

3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269

010762	000005		
010764	012706	001200	
010770	052737	000340	177776
010776	012701	001312	
011002	012702	001400	
011006	005021		
011010	005302		
011012	001375		
011014	012702	000400	
011020	005201		
011022	105021		
011024	005302		
011026	001374		
011030	005000		
011032	013701	001270	
011036	013702	001272	
011042	012703	011220	
011046	010322		
011050	013722	001274	
011054	022323		
011056	010113		
011060	062723	000002	
011064	005723		
011066	010322		
011070	013722	001276	
011074	022323		
011076	010123		
011100	012721	050501	
011104	005721		
011106	006300		
011110	016011	001220	
011114	006200		
011116	012737	000001	001260
011124	012304		
011126	033711	001260	
011132	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, 2#PS      ;PROCESSOR TO LEVEL 7
      MOV    #XMTTAB, R1     ;FIRST TABLE POINTER
      MOV    #1400, R2       ;LENGTH OF TABLES (WORDS)
1$:    CLR    (R1)+          ;CLEAR THE TABLE
      DEC    R2
      BNE   1$
      MOV    #400, R2        ;LENGTH OF MASK/CJUNT 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$:    MOV    R3, (R2)+      ;SET UP RECEIVER INTERRUPT VECTOR
      MOV    RCVLVL, (R2)+
      CMP    (R3)+, (R3)+   ;ADD 4 TO R3
      MOV    R1, (R3)       ;ADDRESS OF CSR
      ADD    #2, (R3)+      ;ADDRESS OF RBUF
      MOV    R3, (R2)+      ;SET UP TRANSMITTER INTERRUPT VECTOR
      MOV    XMTLVL, (R2)+
      CMP    (R3)+, (R3)+
      MOV    R1, (R3)+      ;ADDRESS OF CSR
      MOV    #50501, (R1)+  ;SET UP CSR
      BIT0 = RECEIVER ENABLE
      BIT6 = RECEIVER INTERRUPT ENABLE
      BIT8 = TRANSMITTER SCAN ENABLE
      BIT12 = STATUS ENABLE
      BIT14 = TRANSMITTER INTERRUPT ENABLE
      TST   (R1)+
      ASL   R0
      MOV   SVSWO(R0), (R1) ;UNIT # * 2
      MOV   R0
      ASR   R0
      MOV   #1, MARK       ;SET UP MARKER
      MOV   (R3)+, R4      ;SET UP OFFSET TO TABLES
4$:    BIT   MARK, (R1)    ;CHECK FOR LINE SELECTED
      BEG   5$

```

F03

MAINDEC-11-DZDJB-E-D
DZDJB.E.P11 08-SEP-77

DJ11 EXERCISER AND ON-LINE TESTS
15:07

MACY11 30(1046) 08-SEP-77 15:13 PAGE 49-1
PROG2: ON-LINE EXERCISER (TRANSMIT LAST CHARACTER RECEIVED)

SEG 0031

3270	011134	012764	015604	001312
3271	011142	012764	000045	003312
3272	011150	005724		
3273	011152	006337	001260	
3274	011156	103363		
3275	011160	022121		
3276	011162	005200		
3277	011164	020037	001266	
3278	011170	001326		
3279	011172	042737	000140	177776

```

MOV      #MSGP2, XMTTAB(4) ;SET UP XMTR TABLE
MOV      #45, XMTCNT(4) ;SET UP COUNT
SS:     TST      (R4)+      ;INC OFFSET TO NEXT LINE
        ASL      MARK
        BCC      45
        CMP      (R1)+, (R1)+ ;ADD 4
        INC      RO
        CMP      RO, UNITS
        BNE      1$
        BIC      #140, @#PS ;LOWER PROCESSOR PRIORITY

```

3283				
3284				
3285				
3286				
3287				

```

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

```

3288	011200	012700	020000	
3289	011204	000241		
3290	011206	005540		
3291	011210	001376		
3292	011212	005700		
3293	011214	001374		
3294	011216	000770		
3298				
3299				

```

FORP2:  MOV      #20000,RO ;TOP OF 4K BANK OF MEMORY
1$:     CLC
        ADC      -(RO) ;FAST READ/WRITE TO MEMORY
        BNE      1$ ;RAPID REPEAT
        TST      RO ;CHECK FOR LOC 0
        BNE      1$ ;BRANCH IF MORE MEMORY
        BR       FORP2 ;LOOP FOR EVER!

```

3300				
3301				
3302				

```

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

```

3308	011220	004037	012012	
(1)	011224	160012	000000	
(1)	011230	004037	011620	
(1)	011234	160020	000000	
(1)	011240	004037	012012	
(1)	011244	160022	000040	
(1)	011250	004037	011620	
(1)	011254	160030	000040	
(1)	011260	004037	012012	
(1)	011264	160032	000100	
(1)	011270	004037	011620	
(1)	011274	160040	000100	
(1)	011300	004037	012012	
(1)	011304	160042	000140	
(1)	011310	004037	011620	
(1)	011314	160050	000140	
(1)	011320	004037	012012	
(1)	011324	160052	000200	
(1)	011330	004037	011620	
(1)	011334	160060	000200	
(1)	011340	004037	012012	
(1)	011344	160062	000240	
(1)	011350	004037	011620	
(1)	011354	160070	000240	
(1)	011360	004037	012012	
(1)	011364	160072	000300	
(1)	011370	004037	011620	
(1)	011374	160100	000300	
(1)	011400	004037	012012	

```

R2SR0: JSR      RO,P2RISR
        .WORD    <160012+(0*10)>,<0*40>
X2SR0: JSR      RO,P2XISR
        .WORD    <160020+(0*10)>,<0*40>
R2SR1: JSR      RO,P2RISR
        .WORD    <160012+(1*10)>,<1*40>
X2SR1: JSR      RO,P2XISR
        .WORD    <160020+(1*10)>,<1*40>
R2SR2: JSR      RO,P2RISR
        .WORD    <160012+(2*10)>,<2*40>
X2SR2: JSR      RO,P2XISR
        .WORD    <160020+(2*10)>,<2*40>
R2SR3: JSR      RO,P2RISR
        .WORD    <160012+(3*10)>,<3*40>
X2SR3: JSR      RO,P2XISR
        .WORD    <160020+(3*10)>,<3*40>
R2SR4: JSR      RO,P2RISR
        .WORD    <160012+(4*10)>,<4*40>
X2SR4: JSR      RO,P2XISR
        .WORD    <160020+(4*10)>,<4*40>
R2SR5: JSR      RO,P2RISR
        .WORD    <160012+(5*10)>,<5*40>
X2SR5: JSR      RO,P2XISR
        .WORD    <160020+(5*10)>,<5*40>
R2SR6: JSR      RO,P2RISR
        .WORD    <160012+(6*10)>,<6*40>
X2SR6: JSR      RO,P2XISR
        .WORD    <160020+(6*10)>,<6*40>
R2SR7: JSR      RO,P2RISR

```

```

( ) 011404 160102 000340
( ) 011410 004037 011620
(1) 011414 160110 000340
(1) 011420 004037 012012
( ) 011424 160112 000400
(1) 011430 004037 011620
(1) 011434 160120 000400
(1) 011440 004037 012012
(1) 011444 160122 000440
(1) 011450 004037 011620
(1) 011454 160130 000440
(1) 011460 004037 012012
(1) 011464 160132 000500
(1) 011470 004037 011620
(1) 011474 160140 000500
(1) 011500 004037 012012
(1) 011504 160142 000540
(1) 011510 004037 011620
(1) 011514 160150 000540
(1) 011520 004037 012012
(1) 011524 160152 000600
(1) 011530 004037 011620
(1) 011534 160160 000600
(1) 011540 004037 012012
(1) 011544 160162 000640
(1) 011550 004037 011620
(1) 011554 160170 000640
(1) 011560 004037 012012
(1) 011564 160172 000700
(1) 011570 004037 011620
(1) 011574 160200 000700
(1) 011600 004037 012012
(1) 011604 160202 000740
(1) 011610 004037 011620
(1) 011614 160210 000740

```

```

X2SR7: .WORD <160012+<7*10>>,<7*40>
      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>

```

```

3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330

```

```

011620
(2) 011620 010146
(2) 011622 010246
011624 012001
011626 005711
011630 100064
011632 116102 000007
011636 006302
011640 061002
011642 105762 003312
011646 001413
011650 117261 001312 000006
011656 105362 003312
011662 105762 004313
011666 001357
011670 005262 001312

```

```

;*****
;PROG2 TRANSMITTER INTERRUPT SERVICE ROUTINE
;*****

```

```

P2XISR:
      MOV R1,-(6) ;PUSH R1 ON STACK
      MOV R2,-(6) ;PUSH R2 ON STACK
      MOV (R0)+,R1
IS:   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$ ;GET OUT
      MOVB @XMTTAB(2),6(R1) ;SEND A CHARACTER
      DECB XMTCNT(2) ;COUNT CHARACTERS
      TSTB CNTTAB(2) ;CHECK FOR MESSAGE OR DATA
      BNE 1$ ;BRANCH IF DATA
      INC XMTTAB(2) ;UPDATE TABLE POINTER

```

```

3331 011674 000754          BR      1$
3332 011676 105162 004313 2$:  COMB   CNTTAB(2)      ;CHANGE FLAG
3333 011702 001430          BEQ     3$           ;BRANCH IF WAS DATA
3334 011704 012762 002312 001312  MOV     #RCVTAB,XMTTAB(2) ;SET UP POINTER TO RECEIVER TABLE
3335 011712 060262 001312          ADD     R2,XMTTAB(2) ;ADD OFFSET
3336 011716 112762 000110 003312  MOVB   #72,XMTCNT(2) ;COUNT 72. CHARACTERS TO THE LINE
3337 011724 105762 002312          TSTB   RCVTAB(2)      ;CHECK FOR A BREAK
3338 011730 001336          BNE     1$           ;BRANCH IF REAL DATA
3339 011732 161002          SUB     (R0), R2      ;RECOVER LINE NUMBER
3340 011734 006202          ASR     R2
3341 011736 005037 001260          CLR     MARK          ;SET UP MARKER
3342 011742 000261          SEC
3343 011744 006137 001260 5$:  ROL     MARK          ;MOVE MARKER
3344 011750 005302          DEC     R2           ;COUNT LINES
3345 011752 100374          BPL     5$          ;BRANCH IF MORE
3346 011754 043761 001260 000004  BIC     MARK, 4(R1)   ;CLEAR TCR BIT
3347 011762 000721          BR      1$           ;CONTINUE
3348 011764 012762 015371 001312 3$:  MOV     #RETURN,XMTTAB(2) ;TYPE CARRIAGE RETURN, LINE FEED
3349 011772 112762 000002 003312  MOVB   #2,XMTCNT(2) ;COUNTER OF 2 CHARACTERS
3350 012000 000712          BR      1$
3351 012002 4$:  (2) 012002 012602  MOV     (6)+,R2      ;POP STACK INTO R2
      (2) 012004 012601  MOV     (6)+,R1      ;POP STACK INTO R1
      (2) 012006 012600  MOV     (6)+,R0      ;POP STACK INTO R0
3352 012010 000002          RTI
3356
3357 ;*****
3358 ;PROG2 RECEIVER INTERRUPT SERVICE ROUTINE
3359 ;*****
3360
3361 012012 P2RISR:
      (2) 012012 010146  MOV     R1,-(6)      ;PUSH R1 ON STACK
      (2) 012014 010246  MOV     R2,-(6)      ;PUSH R2 ON STACK
      (2) 012016 010346  MOV     R3,-(6)      ;PUSH R3 ON STACK
3362 012020 012001  MOV     (R0)+,R1     ;GET RBUF ADDRESS
3363 012022 011102 1$:  MOV     (R1),R2      ;READ THE DATA
3364 012024 100053  BPL     7$           ;BRANCH IF NO CHAR PRESENT
3365 012026 032702 070000  BIT     #70000,R2    ;CHECK FOR ERRORS
3366 012032 001402  BEQ     2$           ;BRANCH IF OK
3367 012034 104002  HLT+2  ;RECEIVER ERROR
3368 ;R1=RBUF ADDRESS
3369 ;R2=CONTENTS OF RBUF
3370 ;BIT12=PARITY ERROR
3371 ;BIT13=FRAMING ERROR
3372 ;BIT14=UART OVERRUN
3373 012036 000771          BR      1$           ;FORGET THE DATA
3374
3375 012040 010203 2$:  MOV     R2, R3      ;DUP THE RBUF
3376 012042 105003  CLRB   R3           ;CLEAR THE DATA
3377 012044 000303  SWAB   R3           ;LINE # TO LOW BYTE
3378 012046 106303  ASLB   R3           ;LINE # * 2, ALSO CLR CHAR PRESENT
3379 012050 061003  ADD     (R0),R3      ;ADD OFFSET
3380 012052 136302 004312  BITB   MASK(3),R2   ;CHECK CHARACTER LENGTH
3381 012056 001401  BEQ     3$           ;BRANCH IF OK
3382 012060 104002  HLT+2  ;CHARACTER LENGTH ERROR
3383 ;R1=RBUF ADDRESS

```

```

3384                                     ;R2=CONTENTS OF RBUF(DATA)
3385 012062 105763 002312 35:  TSTB  RCVTAB(3) ;CHECK FOR BREAK
3386 012066 001017          BNE   55   ;BRANCH IF REAL DATA
3387 012070 110263 002312  MOVB  R2,  RCVTAB(3) ;SAVE THE DATA
3388 012074 161003          SUB   (R0), R3 ;RECOVER LINE NUMBER
3389 012076 006203          ASR   R3
3390 012100 005037 001260  CLR   MARK ;SET UP MARKER
3391 012104 000261          SEC
3392 012106 006137 001260 45:  ROL   MARK ;UPDATE MARKER
3393 012112 005303          DEC   R3 ;COUNT LINES
3394 012114 100374          BPL   45   ;BRANCH IF MORE
3395 012116 053761 001260 000002 BIS   MARK, 2(R1) ;SET TCR BIT
3396 012124 000736          BR    15   ;CONTINUE
3397
3398 012126 110263 002312 55:  MOVB  R2,  RCVTAB(3) ;SAVE THE DATA
3399 012132 105163 004313  COMB  CNTTAB(3) ;SET MESSAGE FLAG
3400 012136 012763 015371 001312  MOV   #RETURN, XMTTAB(3) ;TYPE CARRIAGE RETURN, LINE FEED
3401 012144 112763 000002 003312  MOVB  #2,  XMTCNT(3) ;MESSAGE LENGTH
3402 012152 000723          BR
3403                                     75:
(2) 012154 012603          MOV   (6)+,R3 ;POP STACK INTO R3
(2) 012156 012602          MOV   (6)+,R2 ;POP STACK INTO R2
(2) 012160 012601          MOV   (6)+,R1 ;POP STACK INTO R1
(2) 012162 012600          MOV   (6)+,R0 ;POP STACK INTO R0
3404 012164 000002          RTI

```

09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

012166 000005
012170 012706 001200
012174 052737 000340 177776
012202 012701 001312
012206 012702 001400
012212 005021
012214 005302
012216 001375
012220 012702 000400
012224 005201
012226 105021
012230 005302
012232 001374
012234 012737 015734 001264
012242 063737 001262 001264
012250 012705 015734

012254 005000
012256 013701 001270
012262 013702 001272
012266 012703 012550
012272 010322
012274 013722 001274
012300 022323
012302 010113
012304 062723 000002
012310 005723
012312 010322
012314 013722 001276
012320 022323
012322 010123
012324 012721 050400

012330 005721
012332 006300
012334 016011 001220
012340 006200
012342 012737 000001 001260
012350 012304
012352 010246
012354 010346
012356 033711 001260
012362 001420

```
*****  
;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 #1400, R2 ;LENGTH OF TABLES (WORDS)  
15: CLR (R1)+ ;CLEAR THE TABLE  
DEC R2  
BNE 15  
MOV #400, R2 ;LENGTH OF MASK/COUNT TABLE  
25: INC R1 ;SKIP MASK  
CLRB (R1)+ ;CLEAR COUNT  
DEC R2  
BNE 25  
MOV #END, BUFEND ;GENERATE LAST BUFFER ADDRESS  
ADD BUFSIZ, BUFEND  
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 DEVAOR, R1  
MOV VECADR, R2  
MOV #R3SR0, R3 ;SET UP POINTER TO LINKERS  
15: 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  
25: BIT MARK, (R1) ;CHECK FOR LINE SELECTED  
BEG 65
```

```

3464 012364 010564 001312      MOV      R5, XMTTAB(4)      ;SET UP HEADER MESSAGE
3465 012370 010564 002312      MOV      R5, RCVTAB(4)     ;SET UP RECEIVER TABLE
3466 012374 013702 001262      MOV      BUFSIZ, R2        ;SET UP COUNTER
3467 012400 012703 015652      MOV      #MSGP3, R3       ;SET UP MESSAGE POINTER
3468 012404 112325          35:     MOVVB   (R3)+, (R5)+    ;MOVE MESSAGE INTO BUFFER
3469 012406 001404          BEQ      5$                ;BRANCH IF END OF MESSAGE
3470 012410 005302          DEC      R2                ;COUNT BUFFER SIZE
3471 012412 001374          BNE      3$                ;BRANCH IF MORE
3472 012414 000403          BR       6$                ;BRANCH IF DONE
3473 012416 105025          45:     CLRB   (R5)+        ;CLEAR REST OF BUFFER
3474 012420 005302          55:     DEC      R2                ;COUNT BUFFER SIZE
3475 012422 001375          BNE      4$                ;BRANCH IF MORE
3476 012424 005724          65:     TST     (R4)+        ;INC OFFSET TO NEXT LINE
3477 012426 006337 001260      ASL      MARK
3478 012432 103351          BCC      2$
3479 012434 012603          MOV      (6)+, R3          ;POP STACK INTO R3
3480 012436 012602          MOV      (6)+, R2          ;POP STACK INTO R2
3481 012440 022121          CMP      (R1)+, (R1)+     ;ADD 4
3482 012442 005200          INC      R0
3483 012444 020037 001266      CMP      R0, UNITS
3484 012450 001310          BNE      1$
3485 012452 042737 000140 177776 BIC      #140, #0PS      ;LOWER PROCESSOR PRIORITY

```

```

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

```

```

3493 012460 012701 001312      FORP3:  MOV      #XMTTAB, R1
3494 012464 012702 000400      MOV      #400, R2
3495 012470 005711          15:     TST     (R1)            ;CHECK FOR XMTR TABLE CLR
3496 012472 001376          BNE      1$                ;BRANCH IF NOT
3497 012474 062701 000002      ADD      #2, R1            ;GO TO NEXT LINE ENTRY
3498 012500 005302          DEC      R2                ;COUNT LINES
3499 012502 001372          BNE      1$                ;BRANCH IF MORE LINES
3500 012504 013700 001266      MOV      UNITS, R0         ;SET UP UNIT COUNTER
3501 012510 013701 001270      MOV      DEVADR, R1        ;AND DEVICE ADDRESS POINTER
3502 012514 052711 000101      25:     BIS      #101, (R1)   ;SET RECEIVER ENABLES OF CSR
3503          ;BIT0 = RECEIVER ENABLE
3504          ;BIT6 = RCV INTERRUPT ENABLE
3505 012520 062701 000010      ADD      #10, R1           ;UPDATE TO NEXT DJ11
3506 012524 005300          DEC      R0                ;COUNT DJ11'S
3507 012526 001372          BNE      2$
3508 012530 012700 020000      MEMX3:  MOV      #20000, R0    ;TOP OF 4K BANK OF MEMORY
3509 012534 000241          CLC
3510 012536 005540          15:     ADC     -(R0)          ;FAST READ/WRITE TO MEMORY
3511 012540 001376          BNE      1$                ;RAPID REPEAT
3512 012542 005700          TST     R0                 ;CHECK FOR LOC 0
3513 012544 001374          BNE      1$                ;BRANCH IF MORE MEMORY
3514 012546 000770          BR      MEMX3              ;LOOP FOR EVER!

```

```

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

```

```

3528 012550 004037 013300      R3SR0:  JSR     R0, P3RISR
3529 012554 160012 000000      .WORD   <160012+<0*10>>, <0*40>

```

(1)	012560	004037	013150	X3SR0:	JSR	RO, P3XISR
(1)	012564	160020	000000		.WORD	<160020+(0*10)>, <0*40>
(1)	012570	004037	013300	R3SR1:	JSR	RO, P3RISR
(1)	012574	160022	000040		.WORD	<160012+(1*10)>, <1*40>
(1)	012600	004037	013150	X3SR1:	JSR	RO, P3XISR
(1)	012604	160030	000040		.WORD	<160020+(1*10)>, <1*40>
(1)	012610	004037	013300	R3SR2:	JSR	RO, P3RISR
(1)	012614	160032	000100		.WORD	<160012+(2*10)>, <2*40>
(1)	012620	004037	013150	X3SR2:	JSR	RO, P3XISR
(1)	012624	160040	000100		.WORD	<160020+(2*10)>, <2*40>
(1)	012630	004037	013300	R3SR3:	JSR	RO, P3RISR
(1)	012634	160042	000140		.WORD	<160012+(3*10)>, <3*40>
(1)	012640	004037	013150	X3SR3:	JSR	RO, P3XISR
(1)	012644	160050	000140		.WORD	<160020+(3*10)>, <3*40>
(1)	012650	004037	013300	R3SR4:	JSR	RO, P3RISR
(1)	012654	160052	000200		.WORD	<160012+(4*10)>, <4*40>
(1)	012660	004037	013150	X3SR4:	JSR	RO, P3XISR
(1)	012664	160060	000200		.WORD	<160020+(4*10)>, <4*40>
(1)	012670	004037	013300	R3SR5:	JSR	RO, P3RISR
(1)	012674	160062	000240		.WORD	<160012+(5*10)>, <5*40>
(1)	012700	004037	013150	X3SR5:	JSR	RO, P3XISR
(1)	012704	160070	000240		.WORD	<160020+(5*10)>, <5*40>
(1)	012710	004037	013300	R3SR6:	JSR	RO, P3RISR
(1)	012714	160072	000300		.WORD	<16001+(6*10)>, <6*40>
(1)	012720	004037	013150	X3SR6:	JSR	RO, P3XISR
(1)	012724	160100	000300		.WORD	<160020+(6*10)>, <6*40>
(1)	012730	004037	013300	R3SR7:	JSR	RO, P3RISR
(1)	012734	160102	000340		.WORD	<160012+(7*10)>, <7*40>
(1)	012740	004037	013150	X3SR7:	JSR	RO, P3XISR
(1)	012744	160110	000340		.WORD	<160020+(7*10)>, <7*40>
(1)	012750	004037	013300	R3SR10:	JSR	RO, P3RISR
(1)	012754	160112	000400		.WORD	<160012+(10*10)>, <10*40>
(1)	012760	004037	013150	X3SR10:	JSR	RO, P3XISR
(1)	012764	160120	000400		.WORD	<160020+(10*10)>, <10*40>
(1)	012770	004037	013300	R3SR11:	JSR	RO, P3RISR
(1)	012774	160122	000440		.WORD	<160012+(11*10)>, <11*40>
(1)	013000	004037	013150	X3SR11:	JSR	RO, P3XISR
(1)	013004	160130	000440		.WORD	<160020+(11*10)>, <11*40>
(1)	013010	004037	013300	R3SR12:	JSR	RO, P3RISR
(1)	013014	160132	000500		.WORD	<160012+(12*10)>, <12*40>
(1)	013020	004037	013150	X3SR12:	JSR	RO, P3XISR
(1)	013024	160140	000500		.WORD	<160020+(12*10)>, <12*40>
(1)	013030	004037	013300	R3SR13:	JSR	RO, P3RISR
(1)	013034	160142	000540		.WORD	<160012+(13*10)>, <13*40>
(1)	013040	004037	013150	X3SR13:	JSR	RO, P3XISR
(1)	013044	160150	000540		.WORD	<160020+(13*10)>, <13*40>
(1)	013050	004037	013300	R3SR14:	JSR	RO, P3RISR
(1)	013054	160152	000600		.WORD	<160012+(14*10)>, <14*40>
(1)	013060	004037	013150	X3SR14:	JSR	RO, P3XISR
(1)	013064	160160	000600		.WORD	<160020+(14*10)>, <14*40>
(1)	013070	004037	013300	R3SR15:	JSR	RO, P3RISR
(1)	013074	160162	000640		.WORD	<160012+(15*10)>, <15*40>
(1)	013100	004037	013150	X3SR15:	JSR	RO, P3XISR
(1)	013104	160170	000640		.WORD	<160020+(15*10)>, <15*40>
(1)	013110	004037	013300	R3SR16:	JSR	RO, P3RISR
(1)	013114	160172	000700		.WORD	<160012+(16*10)>, <16*40>

(1) 013120 004037 013150
 (1) 013124 160200 000700
 (1) 013130 004037 013300
 (1) 013134 160202 000740
 (1) 013140 004037 013150
 (1) 013144 160210 000740

X3SR16: JSR R0,P3XISR
 .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>

3532
 3533
 3534
 3535
 3536
 3537

 ;PROG3 TRANSMITTER INTERRUPT SERVICE ROUTINE
 ;*****

013150
 (2) 013150 010146
 (2) 013152 010246
 3538 013154 012001
 3539 013156 005711
 3540 013160 100043
 3541 013162 116102 000007
 3542 013166 006302
 3543 013170 061002
 3544 013172 117261 001312 000006
 3545 013200 105072 001312
 3546 013204 005262 001312
 3547 013210 023762 001264 001312
 3548 013216 001003
 3549 013220 012762 015734 001312
 3550 013226 105772 001312
 3551 013232 001351
 3552 013234 010346
 3553 013236 005062 001312
 3554 013242 161002
 3555 013244 006202
 3556 013246 005003
 3557 013250 000261
 3558 013252 006103
 3559 013254 005302
 3560 013256 100375
 3561 013260 040361 000004
 3562 013264 012603
 3563 013266 000733
 3564 013270
 (2) 013270 012602
 (2) 013272 012601
 (2) 013274 012600
 3565 013276 000002

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 2XMTTAB(2),6(R1) ;SEND A CHARACTER
 CLRB 2XMTTAB(2) ;CLR TABLE AFTER USE
 INC XMTTAB(2) ;UPDATE TABLE POINTER
 CMP BUFEND,XMTTAB(2) ;CHECK FOR END OF BUFFER
 BNE 5\$;BRANCH IF NOT
 MOV #END,XMTTAB(2) ;RESET BUFFER POINTER
 5\$: TSTB 2XMTTAB(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
 SEC
 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

3569
 3570
 3571
 3572
 3573
 3574

 ;PROG3 RECEIVER INTERRUPT SERVICE ROUTINE
 ;*****

013300
 (2) 013300 010146
 (2) 013302 010246
 (2) 013304 010346
 (2) 013306 010446
 3575 013310 012001

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 (R0)+,R1 ;GET RBUF ADDRESS

```

3576 013312 011102          1$:  MOV      (R1),R2      ;READ THE DATA
3577 013314 100074          BPL      8$           ;BRANCH IF NO CHAR PRESENT
3578 013316 032702 070000  BIT      #70000,R2   ;CHECK FOR ERRORS
3579 013322 001402          BEQ      2$           ;BRANCH IF OK
3580 013324 104002          HLT+2        ;RECEIVER ERROR
3581                                     ;R1=RBUF ADDRESS
3582                                     ;R2=CONTENTS OF RBUF
3583                                     ;BIT12=PARITY ERROR
3584                                     ;BIT13=FRAMING ERROR
3585                                     ;BIT14=UART OVERRUN
3586 013326 000771          BR        1$         ;SKIP BAD DATA
3587 013330 010204          2$:  MOV      R2,      R4   ;DUP THE RBUF
3588 013332 105004          CLRB    R4          ;CLEAR THE DATA
3589 013334 000304          SWAB   R4          ;LINE # TO LOW BYTE
3590 013336 106304          ASLB   R4          ;LINE # * 2, ALSO CLR CHAR PRESENT
3591 013340 061004          ADD     (R0),R4    ;ADD OFFSET
3592 013342 136402 004312  BITB    MASK(4),R2  ;CHECK CHARACTER LENGTH
3593 013346 001401          BEQ     3$         ;BRANCH IF OK
3594 013350 104002          HLT+2        ;CHARACTER LENGTH ERROR
3595                                     ;R1=RBUF ADDRESS
3596                                     ;R2=CONTENTS OF RBUF(DATA)
3597 013352 005764 002312  3$:  TST     RCVTAB(4)  ;CHECK FOR UNSELECTED LINE
3598 013356 001002          BNE     4$         ;BRANCH IF OK
3599 013360 104002          HLT+2        ;RECEIVED DATA ON UNSELECTED LINE
3600                                     ;R1 = RBUF ADDRESS
3601                                     ;R2 = CONTENTS OF RBUF
3602 013362 000753          BR        1$         ;IGNORE THE DATA
3603 013364 105774 002312  4$:  TSTB   #RCVTAB(4) ;CHECK FOR DATA BUFFER FULL
3604 013370 001403          BEQ     5$         ;BRANCH IF OK
3605 013372 104002          HLT+2        ;SOFTWARE DATA BUFFER OVERFLOW
3606                                     ;POSSIBLE TRANSMITTER PROBLEM
3607                                     ;R1 = RBUF ADDRESS
3608                                     ;R2 = CONTENTS OF RBUF
3609 ;NOTE: IF THE ABOVE ERROR WAS DUE TO OVERLOAD, INCREASING THE CONTENTS
3610 ;       OF "BUFSIZ" MAY RECTIFY THE PROBLEM.
3611 ;       "BUFSIZ" MUST BE A MULTIPLE OF 2.
3612 ;       INCREASING IT MAY CAUSE THE BUFFERS TO OVERFLOW 4K.
3613 013374 000137 012166  JMP     PROG3      ;RESTART ON THIS TYPE ERROR
3614
3615 013400 005764 001312  5$:  TST     XMTTAB(4)  ;CHECK FOR TRANSMITTER ACTIVE
3616 013404 001414          BEQ     6$         ;BRANCH IF INACTIVE
3617 013406 110274 002312  MOVB   R2, #RCVTAB(4) ;PUT THE DATA IN THE BUFFER
3618 013412 005264 002312  INC     RCVTAB(4)   ;UPDATE POINTER TO NEXT SPACE
3619 013416 023764 001264 002312  CMP     BUFEND,RCVTAB(4) ;CHECK FOR END OF BUFFER
3620 013424 001332          BNE     1$         ;BRANCH IF NOT
3621 013426 012764 015734 002312  MOV     #END,RCVTAB(4) ;RESET BUFFER POINTER
3622 013434 000726          BR        1$
3623 013436 012764 015734 002312  6$:  MOV     #END, RCVTAB(4) ;RESET TABLE POINTER
3624 013444 016464 002312 001312  MOV     RCVTAB(4),XMTTAB(4)
3625 013452 110274 002312  MOVB   R2, #RCVTAB(4)
3626 013456 005264 002312  INC     RCVTAB(4)   ;UPDATE POINTER TO NEXT SPACE
3627 013462 161004          SUB     (R0),R4
3628 013464 006204          ASR    R4
3629 013466 005003          CLR   R3
3630 013470 000261          SEC
3631 013472 006103          7$:  ROL    R3

```

```

3632 013474 005304      DEC      R4
3633 013476 100375      BPL      7$
3634 013500 050361 000002      BIS      R3,2(R1)      ;SET TCR BIT FOR LINE
3635 013504 000702      BR
3636 013506      BS:
(2) 013506 012604      MOV      (6)+,R4      ;POP STACK INTO R4
(2) 013510 012603      MOV      (6)+,R3      ;POP STACK INTO R3
(2) 013512 012602      MOV      (6)+,R2      ;POP STACK INTO R2
(2) 013514 012601      MOV      (6)+,R1      ;POP STACK INTO R1
(2) 013516 012600      MOV      (6)+,R0      ;POP STACK INTO R0
3637 013520 000002      RTI
3638 013522      DONE:
(1) 013522 004737 015270      JSR      PC, KBDINT
(1) 013526 062737 000001 001206      ADD      #1,PCNT+2      ;ADD 1 TO THE PASS COUNT
(1) 013534 005537 001204      ADC      PCNT          ;MAKE IT DOUBLE PREC.
(1) 013540 000004 015340      TYPE    ,NEOP        ;END OF PASS INDICATOR
(1) 013544 032777 002000 165436      BIT      #SW10,2SWR    ;RING THE BELL?
(1) 013552 001004      BNE     4$           ;NO!
(1) 013554 000004 000007      TYPE    ,BELL        ;RING THE BELL
(1) 013560 000004 000177      TYPE    ,177         ;TYPE A FILLER FOR 11/05
(1) 013564 013700 000042      4$:     MOV      #42,R0   ;GET MC ITCR ADDRESS
(1) 013570 001405      BEQ     3$           ;IF NONE
(1) 013572 000005      RESET
(1) 013574      SENDAD =
(1) 013574 004710      JSR      7,(0)       ;GO TO MONITOR
(1) 013576 000240      NOP      ;SAVE ROOM
(1) 013600 000240      NOP      ;FOR
(1) 013602 000240      NOP      ;ACT11
(1) 013604 000137 006310      3$:     JMP      RESTAR     ;RETURN

```



```

3656                                     ;SUBROUTINE TO SAVE INPUT AS OCTAL NUMBER
3657
3658
3659 014002 012737 000001 014274 READIN: MOV    #1,INHRE
3660 014010 004737 014150          JSR    PC,    READS    ;GO READ TTY UNTIL CR
3661 014014 005037 014274          CLR    INHRE
3662 014020 010146          MOV    R1,-(6)      ;PUSH R1 ON STACK
(2) 014022 010246          MOV    R2,-(6)      ;PUSH R2 ON STACK
(2) 014024 010346          MOV    R3,-(6)      ;PUSH R3 ON STACK
3663 014026 012501          MOV    (R5)+,R1
3664 014030 012737 000020 015166  MOV    #20,CNT
3665 014036 012702 014276          MOV    #INPUT,R2
3666 014042 122712 000120  CMPB   #120,(R2)    ;CHECK FOR "P"
3667 014046 001425          BEQ    3$
3668 014050 005011          CLR    (R1)
3669 014052 112203          1$:   MOVB   (R2)+,R3
3670 014054 120327 000015  CMPB   R3,#15
3671 014060 001420          BEQ    3$          ;BRANCH WHEN DONE
3672 014062 162703 000060  SUB    #60,R3
3673 014066 032703 177770  BIT    #177770,R3
3674 014072 001013          BNE    3$          ;BRANCH IF BAD DATA
3675 014074 006311          ASL   (R1)
3676 014076 103410          BCS   2$
3677 014100 006311          ASL   (R1)
3678 014102 103406          BCS   2$
3679 014104 006311          ASL   (R1)
3680 014106 103404          BCS   2$
3681 014110 050311          BIS   R3,(R1)
3682 014112 005337 015166  DEC   CNT
3683 014116 000755          BR    1$
3684 014120 000244          2$:   CLZ
3685 014122 013737 177776 014146 3$:   MOV    #PS,    PSTEMP ;MAKE SURE Z-BIT IS CLR
3686 014130 012603          MOV    (6)+,R3      ;SAVE CONDITION CODES
(2) 014132 012602          MOV    (6)+,R2      ;POP STACK INTO R3
(2) 014134 012601          MOV    (6)+,R1      ;POP STACK INTO R2
3687 014136 013737 014146 177776  MOV    PSTEMP,#PS   ;POP STACK INTO R1
3688 014144 000205          RTS   R5           ;RESTORE CONDITION CODES
3689
3690 014146 000000          PSTEMP: 0          ;TEMPORARY STORAGE FOR PS
3691
3692 014150 010346          READS: MOV    R3,-(6) ;SAVE R3
(1) 014152 012703 014276 1$:   MOV    #INPUT,R3   ;GET ADDRESS
(1) 014156 022703 014316 2$:   CMP    #INPUT+20,R3 ;BUFFER FULL?
(1) 014162 001415          BEQ    4$          ;YES - TYPE "?"
(1) 014164 105737 177560  TSTB  #177560      ;WAIT FOR
(1) 014170 100375          BPL   -4           ;A CHARACTER
(1) 014172 113713 177562  MOVB  #177562,(3)  ;GET CHARACTER
(1) 014176 142713 000200  BICB  #200,(3)    ;GET RID OF JUNK
(1) 014202 122713 000177  CMPB  #177,(3)   ;IS IT A RUBOUT
(1) 014206 001403          BEQ    4$          ;SKIP IF NOT
(1) 014210 122713 000025  CMPB  #25,(3)
(1) 014214 001006          BNE    3$
(1) 014216          4$:
(1) 014216 000004 014222          TYPE  +2          ;ASCIZ "?"(15)(12)"= "
(1) 014230 000750          BR    1$          ;ZAP THE BUFFER AND LOOP
(1) 014232 111337 015104 3$:   MOVB  (3),.TYPE   ;SET UP FOR TYPING

```

```

(1) 014236 000004 015104 TYPE TYPE :ECHO IT
(1) 014242 122723 000015 CMPB #15,(3)+ :CHECK FOR RETURN
(1) 014246 001343 BNE 25 :LOOP IF NOT RETURN
(1) 014250 005737 014274 TST INHRE
(1) 014254 001401 BEQ 55
(1) 014256 000402 BR 65
(1) 014260 105063 177777 55: CLRB -1(3) :ZAP RETURN (THE 15)
(1) 014264 000004 000012 65: TYPE 12 :TYPE A LINE FEED
(1) 014270 012603 MOV (6)+,R3 :RESTORE R3
(1) 014272 000207 RTS PC :RETURN

```

```

(1) 014274 000000 INHRE: 0
(1) 014276 000020 INPUT: .BLKW 20 :TTY INPUT AREA
3693 ; $OCTAL OCTAL TYPEOUT ROUTINE

```

```

;THIS ROUTINE IS USED TO TYPE AN OCTAL NUMBER ON THE TTY. IT WILL TYPE
;ALL 6 CHARACTERS. SUPPRESS LEADING ZEROES, TYPE AN 18 BIT ADDRESS, OR TYPE
;THE 16 BITS. IT IS CALLED VIA THE DUMP, SDUMP, DUMP18, OR BITYPE MACRO'S.

```

```

(1) 014336 012737 170101 014504 BITYPS: MOV #170101,.PR :SET BIT FLAG ANS 16. CHARACTER COUNT
(1) 014344 000411 BR .PTIT :NOW TYPE IT IN BIT FORM
(1) 014346 112737 000001 014504 PRINTR: MOVB #1,.PR :SET ZERO FILL SWITCH
(1) 014354 000402 BR .+6 :SKIP
(1) 014356 005037 014504 PRINTS: CLR .PR :SUPPRESS LEADING ZERO'S
(1) 014362 112737 177772 014505 .PTIT: MOVB #-6,.PR+1 :SET COUNT
(1) 014370 010446 .PTIT: MOV R4 -(6) :SAVE R4
(1) 014372 012704 014506 MOV #.PR+2,R4 :SET POINTER TO FIRST ASCII CHAR.
(1) 014376 105014 CLRB (4) :CLEAR FIRST BYTE
(1) 014400 000411 BR .PRF :ROTATE FIRST BIT
(1) 014402 105014 .PRL: CLRB (4) :CLEAR BYTE OF CHARACTER
(1) 014404 032737 000100 014504 BIT #100,.PR :BIT TYPING MODE?
(1) 014412 001004 BNE .PRF :YES - SKIP 2 ROTATES
(1) 014414 006105 ROL TTY :ROTATE BIT INTO C
(1) 014416 106114 ROLB (4) :PACK IT
(1) 014420 006105 ROL TTY :ROTATE BIT INTO C
(1) 014422 106114 ROLB (4) :PACK IT
(1) 014424 006105 .PRF: ROL TTY :ROTATE BIT INTO C
(1) 014426 106114 ROLB (4) :PACK IT
(1) 014430 105714 TSTB (4) :IS IT ZERO?
(1) 014432 001402 BEQ .+6 :SKIP INC
(1) 014434 105237 014504 INCB .PR :SET FILL SWITCH
(1) 014440 105737 014504 TSTB .PR :CHECK FILL SWITCH
(1) 014444 001402 BEQ .+6 :SKIP BITSET
(1) 014446 152724 000060 BISB #'0,(4)+ :MAKE INTO ASCII CHAR
(1) 014452 105237 014505 INCB .PR+1 :INC COUNT
(1) 014456 001351 BNE .PRL :REPEAT
(1) 014460 022704 014506 CMP #.PR+2,R4 :EMPTY BUFFER?
(1) 014464 001002 BNE .+6 :SKIP IF NOT
(1) 014466 112724 000060 MOVB #'0,(4)+ :LOAD 1 ZERO
(1) 014472 105014 CLRB (4) :NULL TERMINATOR
(1) 014474 000004 014506 TYPE .PR+2 :TYPE IT
(1) 014500 012604 MOV (6)+,R4 :RESTORE R4
(1) 014502 000207 RTS PC :RETURN
(1) 014504 000012 .PR: .BLKW 12 :COUNT, SWITCH, AND OUTPUT BUFFER

```

F04

MAINDEC-11-DZDJB-E-D
DZDJB.E.P11 08-SEP-77

DJ11 EXERCISER AND ON-LINE TESTS
15:07

MACY11 30(1046) 08-SEP-77 15:13 PAGE 51-3
POWER DOWN AND UP ROUTINES

SEQ 0044

```

3695 014530 012777 014656 000126 PDOWN$: MOV #ILLUP, @PUVECS ;SET FOR FAST UP
(1) 014536 012777 000340 000122 MOV #340, @PUVECS+2 ;PRIO:7
(3) 014544 010046 MOV R0, -(6) ;PUSH R0 ON STACK
(3) 014546 010146 MOV R1, -(6) ;PUSH R1 ON STACK
(3) 014550 010246 MOV R2, -(6) ;PUSH R2 ON STACK
(3) 014552 010346 MOV R3, -(6) ;PUSH R3 ON STACK
(3) 014554 010446 MOV R4, -(6) ;PUSH R4 ON STACK
(3) 014556 010546 MOV R5, -(6) ;PUSH R5 ON STACK
(1) 014560 010637 014662 MOV SP, SAVR6 ;SAVE SP
(1) 014564 012777 014574 000072 MOV #PUPS, @PUVECS ;SET UP VECTOR
(1) 014572 000000 HALT ;WAIT FOR PF
(1) 014574 013706 014662 PUPS: MOV .SAVR6, SP ;GET SP
(1) 014600 005001 CLR R1 ;WAIT LOOP FOR THE TTY
(1) 014602 005201 IS: INC R1 ;WAIT FOR THE INC
(1) 014604 001376 BNE IS ;OF WORD
(3) 014606 012605 MOV (6)+, R5 ;POP STACK INTO R5
(3) 014610 012604 MOV (6)+, R4 ;POP STACK INTO R4
(3) 014612 012603 MOV (6)+, R3 ;POP STACK INTO R3
(3) 014614 012602 MOV (6)+, R2 ;POP STACK INTO R2
(3) 014616 012601 MOV (6)+, R1 ;POP STACK INTO R1
(3) 014620 012600 MOV (6)+, R0 ;POP STACK INTO R0
(1) 014622 012737 014530 000024 MOV #PDOWN$+2, @24 ;SET UP THE POWER DOWN VECTOR
(1) 014630 012737 000340 000026 MOV #340, @26 ;PRIO:7
(2) 014636 000004 014642 TYPE +2 ;ASCIZ <15><12>"POWER"
(1) 014652 000137 006310 JMP RESTAR ;JMP TO USER ADDRESS
(1) 014656 000000 ILLUP: HALT ;THE POWER UP SEQUENCE WAS STARTED
(1) 014660 000776 BR .-2 ;BEFORE THE POWER DOWN WAS COMPLETE
(1) 014662 000000 .SAVR6: 0 ;PUT THE SP HERE
(1) 014664 000024 000026 PUVECS: 24,26 ;POWER UP VECTOR
3696
3697
3698 014670 000002 YESRT: RTI ;RETURN FROM TRACE TRAP

```



```

(1) 015104 000000 .TYPE: 0 ; CHARACTER TYPE LOCATION
3724
(1) 015106 022737 000176 001210 CNTLU: CMP #SWREG,SWR
(1) 015114 001023 BNE 1$
(1) 015116 000004 015200 TYPE SWREG
(2) 015122 013705 000176 MOV SWREG,TTY ;TYPE SWREG IN OCTAL
(2) 015126 004737 014346 JSR PC,PRINTR ;TYPE LEADING ZERO'S
(1) 015132 000004 015170 TYPE NEWIS
(1) 015136 004537 014002 JSR AS,READIN
(1) 015142 015336 .WORD TMP1
(1) 015144 001360 BNE CNTLU
(1) 015146 022737 000020 015166 CMP #20,CNT
(1) 015154 001403 BEQ 1$
(1) 015156 013777 015336 164024 MOV TMP1,@SWR
(1) 015164 000207 1$: RTS PC
(1) 015166 000000 CNT: 0
(1) 015170 020040 042516 036527 NEWIS: .ASCIZ " NEW= "
(1) 015176 000040
(1) 015200 005015 053523 036522 SWREQ: .ASCIZ <15><12>"SWR= "
(1) 015206 000040
3725
(1) 015210 013746 000006 SUSWR: MOV 6,-(SP)
(1) 015214 013746 000004 MOV 4,-(SP)
(1) 015220 012737 015240 000004 MOV #1$ ,4
(1) 015226 022777 177777 163754 CMP #-1,@SWR
(1) 015234 001402 BEQ 2$
(1) 015236 000407 BR 3$
(1) 015240 022626 1$: CMP (SP)+,(SP)+
(1) 015242 012737 000176 001210 2$: MOV #SWREG,SWR
(1) 015250 012737 000174 001212 MOV #DISPREG,DISPLAY
(1) 015256 012637 000004 3$: MOV (SP)+,4
(1) 015262 012637 000006 MOV (SP)+,6
(1) 015266 000207 RTS PC
3726
(1) 015270 022737 000176 001210 KBDINT: CMP #SWREG,SWR
(1) 015276 001016 BNE 1$
(1) 015300 005037 015336 CLR TMP1
(1) 015304 013737 177562 015336 MOVB 177562,TMP1
(1) 015312 014273 000200 015336 BICB #200,TMP1
(1) 015320 122737 000007 015336 CMPB #7,TMP1
(1) 015326 001007 BNE 1$
(1) 015330 004737 015106 JSR PC,CNTLU
(1) 015334 000207 1$: RTS PC
(1) 015336 000000 TMP1: 0
3727
3728 015340 005015 047505 000120 MEOP: .ASCIZ <15><12>"EOP"
3729 015346 020043 000 MNUM: .ASCIZ " "
3730 015351 040 042523 042514 MSGSEL: .ASCIZ " SELECT LINE = "
(1) 015356 052103 046040 047111
(1) 015364 020105 020075 000

```

```

3731 015371 015 177412 177777 RETURN: .ASCIZ <15><12><377><377><377>
      015376 000
3732 015377 040 000040 SPACE: .ASCIZ " "
3733 015402 005015 046777 044501 MSGMDN: .ASCIZ <15><12><377>"MAINDEC-11-DZDJB-E DJ11 EXERCISER"
      015410 042116 041505 030455
      015416 026461 055104 045104
      015424 026502 020105 020040
      015432 045104 030461 042440
      015440 042530 041522 051511
      015446 051105 000
3734 015451 015 043012 051111 MSGADR: .ASCIZ <15><12>"FIRST DJ11 ADDRESS: "
      015456 052123 042040 030512
      015464 020061 042101 051104
      015472 051505 035123 020040
      015500 000
3735 015501 015 053012 041505 MSGVEC: .ASCIZ <15><12>"VECTOR ADDRESS: "
      015506 047524 020122 042101
      015514 051104 051505 035123
      015522 020040 000
3736 015525 015 047012 027117 MSGNUM: .ASCIZ <15><12>"NO. OF DJ11'S: "
      015532 047440 020106 045104
      015540 030461 051447 020072
      015546 000040
3737 015550 005015 051120 043517 MSGPRG: .ASCIZ <15><12>"PROGRAM #: "
      015556 040522 020115 035043
      015564 020040 000
3738 015567 015 047012 020117 MSG01: .ASCIZ <15><12>"NO DJ11'S!"
      015574 045104 030461 051447
      015602 000041
3739 015604 005015 051120 043517 MSGP2: .ASCIZ <15><12>"PROG2: CONTINUOUS ECHO EXERCISER"<15><12>
      015612 035062 020040 047503
      015620 052116 047111 047525
      015626 051525 042440 044103
      015634 020117 054105 051105
      015642 044503 042523 006522
      015650 000012
3740 015652 005015 042452 044103 MSGP3: .ASCIZ <15><12>"*ECHO TEST*"<15><12>
      015660 020117 042524 052123
      015666 006452 000012
3741 015672 026440 000040 MSGDAS: .ASCIZ " - "
3742 015676 005015 044523 047514 MALARM: .ASCIZ <15><12>"SILO ALARM LEVEL FOR CSR"<15><12>
      015704 040440 040514 046522
      015712 046047 053105 046105
      015720 043046 051117 041440
      015726 051123 005015 000
3743 015734 000000 .EVEN
3744 000000 ENC: 0
3745 000000 .ENC

```


BCP	3191#	3199	3205								
BITYPE	525#	2620#									
BLKBLK	1196#										
BUFHOR	1189#										
DUMP	515#	2620#	2770	3641	3724						
DUMP18	538#										
FILBLK	1181#										
LNKBLK	1173#										
OOT11	1224#										
OCT11X	1740#										
POP	571#	2620#	3120	3161	3351	3403	3479	3564	3636	3686	3695
PRINT	530#	2620#	3641	3692	3695						
PUSH	564#	2620#	3095	3130	3317	3361	3461	3537	3574	3662	3695
SCOPE.	353#										
SCOP.	382#										
SDUMP	520#	2620#	2772	2774	3714	3719					
TRACE	1154#										
TRMBLK	1202#										
TYPEN	543#										
WRITE	1210#										
SCATCH	578#										
SCHTAG	293#										
SCNTL	2475#	3724									
SEND	606#	3638									
SEQUAT	451#	2620									
SALT	725#	3641									
SILOCAT	2509#	2625									
SIOT	2538#										
SKBIN	2459#	3726									
SKRAT	134#										
SLOROR	397#										
SOCIAL	789#	3693									
SPOWER	871#	3695									
SRAND	1074#										
SRAND4	1117#										
SREAC	1023#	3692									
SSCOPE	654#										
SSETUP	311#										
SSRAT	191#										
SSWDOC	103#	2571									
SSWROF	2497#	2633									
SSWRRR	2442#	3725									
STRAP	978#										
STYPE	917#	3723									
SJRA*	242#										
.SCOP	374#										
.SCOPE	345#										

. RES. 015736 000

ERRORS DETECTED: 0

CZCJBE,CZCJBE/CRF+CZCJBE.MAC,CZCJBE.P11
RUN TIME: 57.5 SECONDS

C05

MAINDEC-11-DZDJB-E-D DJ11 EXERCISER AND ON-LINE TESTS
CZDJBE.P11 08-SEP-77 15:07

MACY11 30(1046) 08-SEP-77 15:13 PAGE 53-1
CROSS REFERENCE TABLE -- MACRO NAMES

SEG 0054

RUN-TIME RATIO: 228/14=16.0
CORE USED: 32K (63 PAGES)

D05