

DJ11

EXER ONLINE DIAG
CZDJBGO

AH-8509G-MC
FICHE 1 OF 1

SEP 1982
COPYRIGHT © 75-82
MADE IN USA



EXERCISE ONLINE DIAGNOSTIC

TEST	STATUS	DESCRIPTION
1	PASS	POWER SUPPLY
2	PASS	ENGINE OIL PRESSURE
3	PASS	ENGINE WATER PUMP
4	PASS	ENGINE FAN MOTOR
5	PASS	ENGINE BELT DRIVE
6	PASS	ENGINE VALVE TRAIN
7	PASS	ENGINE TIMING
8	PASS	ENGINE COMPRESSION
9	PASS	ENGINE OIL LEAK
10	PASS	ENGINE EXHAUST
11	PASS	ENGINE EMISSIONS
12	PASS	ENGINE NOISE
13	PASS	ENGINE VIBRATION
14	PASS	ENGINE TEMPERATURE
15	PASS	ENGINE OIL LEVEL
16	PASS	ENGINE OIL QUALITY
17	PASS	ENGINE OIL PRESSURE
18	PASS	ENGINE OIL TEMPERATURE
19	PASS	ENGINE OIL VISCOSITY
20	PASS	ENGINE OIL CLEANLINESS
21	PASS	ENGINE OIL CONSUMPTION
22	PASS	ENGINE OIL DRAINAGE
23	PASS	ENGINE OIL FILTRATION
24	PASS	ENGINE OIL STORAGE
25	PASS	ENGINE OIL DISTRIBUTION
26	PASS	ENGINE OIL RETURN
27	PASS	ENGINE OIL REGENERATION
28	PASS	ENGINE OIL REFINING
29	PASS	ENGINE OIL RECYCLING
30	PASS	ENGINE OIL REUSE

.REM

IDENTIFICATION

PRODUCT CODE: AC-8507G-MC
PRODUCT NAME: CZDJBG DJ11 EXER & ONLNE
PROGRAM DATE: JUNE 1982
MAINTAINER: DIAGNOSTIC ENGINEERING

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

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

COPYRIGHT (C) 1975, 1982 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

C 1
CONTENTS

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

HISTORY

JUNE 1982 REV F TO REV G

INSERTED THE LINE 'MOV #0,@MPS' IN THE ROUTINE 'BEGIN'
TO INSURE THAT INTERRUPTS WILL BE ALLOWED.



1. ABSTRACT

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

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

2. REQUIREMENTS

2.1 EQUIPMENT

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

2.2 STORAGE

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

2.3 PRELIMINARY PROGRAMS

CZDJA DJ11 LOGIC TESTS

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR ABS TAPES.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE 5.1 (ALL DOWN FOR WORST CASE TESTING)

4.2 STARTING ADDRESS

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

4.3 PROGRAM AND OPERATOR ACTION

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

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

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

THE SWITCH SETTINGS ARE:

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

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

(I.E.) SWR=XXXXXX NEW=

POSSIBLE RESPONSES ARE:

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

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

5.2 SUBROUTINE ABSTRACTS

5.2.1 HLT

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

5.2.2 ALMCK (PROG1 ONLY)

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

5.2.3 TRAPCATCHER

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

5.3 PROGRAM AND OPERATOR ACTION

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

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

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 TCR OF THE DJ11.) IF MORE DJ11'S ARE ON THE SYSTEM THE N WILL INDICATE THE NEXT DJ11 AND THE PROMPT IS REISSUED. WHEN ALL LINES ARE SELECTED THE PROGRAM WILL RUN THE SELECTED SUBPROGRAM.

6. ERRORS

6.1 ERROR PRINTOUT

THE FORMAT IS AS FOLLOWS:

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

WHERE:

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

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

6.2 ERROR RECOVERY

RESTART AT 200 OR 1000.

6.3 ERROR COUNTER

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

7. RESTRICTIONS

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

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

ONLY PROGRAM 1 IS RUN.

8. MISCELLANEOUS

8.1 EXECUTION TIME (PROG1 ONLY)

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

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

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1200

8.3 PASS COUNT (PROG1 ONLY)

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

8.4 POWER FAIL

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

9. PROGRAM DESCRIPTION

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

PROGRAM 1: EXERCISER (OFF-LINE)

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

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

DJ11 REGISTER BIT ASSIGNMENTS:

CONTROL STATUS REGISTER (CSR) XXXXX0

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

RECEIVER BUFFER REGISTER (RBUF) XXXXX2 (READ ONLY)

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

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

BIT0-15 STOP THE SCANNER ON CORRESPONDING LINE

TRANSMITTER BUFFER (TBUF) XXXXX6

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

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

BIT0-15 TRNSMIT A BREAK ON CORRESPONDING LINE!

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

491
 492 000000 000000 000000
 493
 494
 495
 496
 497
 498 000046 000046
 499 000046 014622
 500
 501 000174 000174
 502 000174 000000
 503 000176 000000
 504
 505 000200 000200
 506 000200 000137 006312
 507 001000 001000
 508 001000 000137 007316
 509
 510 001200

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

. = 46
 SENDAD

. = 174
 DISPREG: 0
 SWREG: 0

. = 200
 JMP BEGIN ;200 ALWAYS IS THE STARTING ADDRESS

. = 1000
 JMP RESTAR ;RESTART ADDRESS

. = 1200

511 001200 000000
 512 001202 000000
 513 001204 000000 000000
 514
 515 001210 177570
 516 001212 177570
 517
 518 001214 000000
 519 001216 000020
 520 001220 000000
 521 001222 000000
 522 001224 000000
 523 001226 000000
 524 001230 000000
 525 001232 000000
 526 001234 000000
 527 001236 000000
 528 001240 000000
 529 001242 000000
 530 001244 000000
 531 001246 000000
 532 001250 000000
 533 001252 000000
 534 001254 000000
 535 001256 000000
 536 001260 000000
 537 001262 000030
 538 001264 000001
 539 001266 160010
 540 001270 000300
 541 001272 000240
 542 001274 000240
 543 001276 000000
 544 001300 000000
 545 001302 000000
 546 001304 000000
 547 001306 000000
 548 001310 000000
 549
 550
 551
 552
 553
 554 001312 000400
 555
 556 002312 000400
 557
 558
 559 003312 000400
 560
 561 004312 000400
 562 004313
 563
 564 005312 000400
 565 005313

ICNT: 0
 ERRORS: 0
 PCNT: 0.0
 SWR: 177570
 DISPLAY:177570
 SAVIT: 0
 TIMES: 20
 SVSW0: OPEN
 SVSW1: OPEN
 SVSW2: OPEN
 SVSW3: OPEN
 SVSW4: OPEN
 SVSW5: OPEN
 SVSW6: OPEN
 SVSW7: OPEN
 SVSW10: OPEN
 SVSW11: OPEN
 SVSW12: OPEN
 SVSW13: OPEN
 SVSW14: OPEN
 SVSW15: OPEN
 SVSW16: OPEN
 SVSW17: OPEN
 MARK: 0
 BUFSIZ: 30
 UNITS: 1
 DEVADR: 160010
 VECADR: 300
 RCVLVL: 240
 XMTLVL: 240
 ISRFLG: 0
 ALMFLG: 0
 TIMERA: 0
 TIMERB: 0
 COUNT: 0
 SUM: 0

: ITERATION COUNT-LH, TEST NO.-RH
 : ERROR COUNT REGISTER
 : PASS COUNT REGISTER
 : MINIMUM NUMBER OF MESSAGES (PROG1)
 : MAP OF LINES SELECTED, DJ11 #0
 : MAP OF LINES SELECTED, DJ11 #1
 : MAP OF LINES SELECTED, DJ11 #2
 : MAP OF LINES SELECTED, DJ11 #3
 : MAP OF LINES SELECTED, DJ11 #4
 : MAP OF LINES SELECTED, DJ11 #5
 : MAP OF LINES SELECTED, DJ11 #6
 : MAP OF LINES SELECTED, DJ11 #7
 : MAP OF LINES SELECTED, DJ11 #10
 : MAP OF LINES SELECTED, DJ11 #11
 : MAP OF LINES SELECTED, DJ11 #12
 : MAP OF LINES SELECTED, DJ11 #13
 : MAP OF LINES SELECTED, DJ11 #14
 : MAP OF LINES SELECTED, DJ11 #15
 : MAP OF LINES SELECTED, DJ11 #16
 : MAP OF LINES SELECTED, DJ11 #17

: RECEIVE BUFFER SIZE (PROG3)
 : NUMBER OF UNITS ON THE SYSTEM
 : FIRST DEVICE ADDRESS
 : FIRST VECTOR ADDRESS
 : RECEIVER BR LEVEL = 5
 : TRANSMITTER BR LEVEL = 5
 : INTR SVC RTN FLAG
 : SILO ALARM LEVEL FLAG
 : TIME COUNTERS

::++F

:*****
 : TABLES
 :*****

XMTTAB: .BLKW 400
 RCVTAB: .BLKW 400
 MAXTAB: .BLKW 400
 XMTCNT: .BLKW 400
 RVCNT=XMTCNT+1
 MASK: .BLKW 400
 CNTTAB=MASK+1

: TRANSMIT DATA POINTER TABLE
 : RECEIVE DATA POINTER TABLE (PROG1)
 : RECEIVE DATA TABLE (PROG2 AND PROG3)
 : POINTER FOR XMIT/RCV BFRS.
 : TRANSMIT DATA COUNTER
 : RECEIVE DATA COUNTER
 : CHARACTER MASK TABLE
 : ITERATION COUNT AND FLAGS

::++F

```

566
567 006312 012706 001200 BEGIN: MOV #STACK, SP ;SET UP STACK POINTER
568 006316 012737 000000 177776 MOV #0,@#PS ;:ALLOW INTERUPTS VRG-060382
569 006324 004737 016236 JSR PC,SUSWRR
570 006330 012700 000020 MOV #20,R0
571 006334 012720 015720 MOV #IOTRAP,(R0)+ ;:IOT VECTOR (20)
572 006340 012720 000340 MOV #340,(R0)+
573 006344 012720 015556 MOV #PDOWNS,(R0)+ ;:POWER FAIL VECTOR (24)
574 006350 012720 000340 MOV #340,(R0)+
575 006354 012720 014636 MOV #EMTS,(R0)+ ;:EMT VECTOR (30)
576 006360 012720 000340 MOV #340,(R0)+
577 006364 005037 001202 CLR ERRORS ;:CLEAR ERROR COUNTER
578 006370 005037 001204 CLR PCNT ;:CLEAR PASS COUNTER
579 006374 005037 001206 CLR PCNT+2
580 006400 012700 000300 MOV #300, R0 ;:START OF FLOATING VECTOR AREA
581 006404 005720 2$: TST (R0)+ ;:UPDATE POINTER
582 006406 010060 177776 MOV RO, -2(R0) ;:PUT '+2' IN EACH VECTOR
583 006412 012720 000004 MOV #IOT, (R0)+ ;:AND 'IOT'
584 006416 022700 001000 CMP #1000, R0 ;:CHECK FOR END OF FLOATING VECTOR AREA
585 006422 003370 BGT 2$ ;:BRANCH IF MORE
586 006424 005037 001300 CLR ALMFLG ;:CLEAR THE ALARM FLAG
587 006430 012737 000400 011240 MOV #256,CNTNIT ;:SET MAXIMUM SIZE VALUES
588 006436 012737 000106 011244 MOV #70,CNTNIT+4 ;:FOR PROG #1
589 006444 012737 000106 011250 MOV #70,CNTNIT+10
590 006452 012737 000001 001216 MOV #1,TIMES ;:SET FOR QV ON FIRST PASS
591
592 ;*****
593 ;ROUTINE TO MAP ALL THE DJ11'S ON THE SYSTEM
594 ;*****
595
596 006460 013700 001266 DJMAP: MOV DEVADR, R0 ;:GET FIRST FLOATING ADDRESS
597 006464 012702 000001 MOV #1, R2 ;:COUNTER FOR DJ11'S
598 006470 012737 000002 000006 MOV #RTI, @#6 ;:RTI WHEN TIME-OUT
599 006476 005001 5$: CLR R1 ;:SET UP COUNTER
600 006500 000261 1$: SEC ;:SET CARRY
601 006502 005710 TST (R0) ;:CHECK FOR A DEVICE
602 006504 103404 BCS 7$ ;:BRANCH IF NONE
603 006506 062700 000010 6$: ADD #10, R0 ;:GO TO NEXT DEVICE ADDRESS
604 006512 005201 INC R1 ;:COUNT DJ11'S
605 006514 000771 BR 1$ ;:LOOK FOR MORE
606
607 006516 005037 000006 7$: CLR @#6 ;:RESTORE TIMEOUT VECTOR
608 006522 010137 001264 MOV R1, UNITS ;:SAVE COUNT
609 006526 001005 BNE GETVEC
610 006530 000004 016632 TYPE, MSGO1 ;:TYPE 'NO DJ11'S!'
611 006534 000000 HALT ;:FATAL ERROR
612 006536 000137 014550 JMP @#DONE ;:RESTART
613
614 ;*****
615 ;ROUTINE TO DETERMINE VECTOR ADDRESSES OF DJ11'S
616 ;*****
617
618 006542 013746 000020 GETVEC: MOV @#20, -(SP) ;:SAVE IOT VECTOR
619 006546 012737 006576 000020 MOV #1$, @#20 ;:RESET IOT VECTOR
620 006554 013701 001266 MOV DEVADR, R1 ;:FIRST DJ ADDRESS
621 006560 012711 040400 MOV #40400, (R1) ;:SET CSR

```

```

622
623
624 006564 012761 000001 000004      MOV      #1,      4(R1)      ;BIT8= TRANS SCAN ENABLE
625 006572 000001                                WAIT                                ;BIT14= TRANS INTERRUPT ENABLE
626 006574 000407                                BR      3$                      ;TCR, LINE 0
627                                         ;WAIT FOR AN INTERRUPT
628 006576 011602                                1$:  MOV      (SP),    R2      ;SAVE VECTOR ADR. (+4)
629 006600 162716 000010                        SUB      #10,    (SP)      ;REPOSITION ADR TO RCV. VEC.
630 006604 011637 001270                        MOV      (SP),    VECADR   ;SAVE FIRST VECTOR
631 006610 022626                        CMP      (SP)+,   (SP)+    ;RESET STACK FROM IOT
632 006612 000002                        RTI                                ;RETURN FROM INITIAL INTERRUPT
633
634 006614 012637 000020                        3$:  MOV      (SP)+,   @#20   ;RESTORE IOT VECTOR
635 006620 005742                        TST     -(R2)              ;POINT TO XMT. VEC. +2
636 006622 013703 001264                        MOV      UNITS,   R3      ;SET UP UNIT COUNTER
637
638                                         ;CHECK THAT VECTORS ARE CONTIGUOUS
639
640 006626 005061 000004                        2$:  CLR      4(R1)          ;CLEAR TCR
641 006632 005011                        CLR      (R1)             ;CLEAR CSR
642 006634 012712 000004                        MOV      #IOT,    (R2)    ;RESTORE IOT TO XMT. VEC.+2
643 006640 005303                        DEC      R3              ;CHECK FOR MORE DJ11'S
644 006642 001415                        BEQ     REPORT        ;BRANCH IF DONE
645 006644 062701 000010                        ADD      #10,    R1      ;UPDATE DJ ADR. POINTER
646 006650 062702 000010                        ADD      #10,    R2      ;UPDATE VECTOR POINTER
647 006654 012712 000002                        MOV      #RTI,   (R2)    ;RTI ON INTERRUPT
648 006660 012711 040400                        MOV      #40400, (R1)    ;SET CSR
649 006664 012761 000001 000004      MOV      #1,      4(R1)    ;TCR LINE 0
650 006672 000001                                WAIT                                ;WAIT FOR AN INTERRUPT
651 006674 000754                                BR      2$
652
653                                         ;REPORT CONFIGURATION
654
655 006676 032777 020000 172304  REPORT: BIT      #BIT13, @SWR  ;CHECK FOR INHIBIT TYP0UT
656 006704 001026                                BNE     GETLEN          ;SKIP REPORT IF SET
657 006706 000004 016453                                TYPE,  MSGMDN
658 006712 000004 016442                                TYPE,  RETURN
659 006716 000004 016514                                TYPE,  MSGADR
660 006722 013705 001266                        MOV      DEVADR,TTY      ;TYPE DEVADR IN OCTAL
661 006726 004737 015374                        JSR     PC,PRINTR        ;TYPE LEADING ZERO'S
662 006732 000004 016544                                TYPE,  MSGVEC
663 006736 013705 001270                        MOV      VECADR,TTY      ;TYPE VECADR IN OCTAL
664 006742 004737 015404                        JSR     PC,PRINTS        ;AND SUPRESS LEADING ZERO'S
665 006746 000004 015570                                TYPE,  MSGNUM
666 006752 013705 001264                        MOV      UNITS,TTY       ;TYPE UNITS IN OCTAL
667 006756 004737 015404                        JSR     PC,PRINTS        ;AND SUPRESS LEADING ZERO'S

```

```

668
669
670
671
672
673 006762 022737 000176 001210 GETLEN: CMP #SWREG,SWR
674 006770 001002 BNE 6$
675 006772 004737 016134 JSR PC,CNTLU
676 006776 013700 001264 6$: MOV UNITS, R0 ;SET UP UNIT COUNTER
677 007002 013701 001266 MOV DEVADR, R1 ;SET UP DEVICE ADDRESS POINTER
678 007006 012702 000001 1$: MOV #1, R2 ;SET UP LINE MARKER
679 007012 012711 000415 MOV #415, (R1) ;RCV ENB, CMOS, MAINT., TRANS SCAN ENB
680 007016 032711 000020 10$: BIT #BIT4, (R1) ;WAIT FOR MOS TO CLEAR
681 007022 001375 BNE 10$
682 007024 010261 000004 2$: MOV R2, 4(R1) ;TRANS CONTROL, ONE LINE AT A TIME
683 007030 005711 3$: TST (R1) ;WAIT FOR TRANS READY
684 007032 100376 BPL 3$
685 007034 012761 000377 000006 MOV #377, 6(R1) ;SEND A RUBOUT
686 007042 006302 ASL R2 ;SKIP 4 LINES
687 007044 006302 ASL R2
688 007046 006302 ASL R2
689 007050 006302 ASL R2
690 007052 103364 BCC 2$ ;BRANCH BACK IF MORE LINES
691 007054 005061 000004 CLR 4(R1) ;CLEAR TCR
692 007060 062701 000010 ADD #10, R1 ;UPDATE POINTER TO NEXT UNIT
693 007064 005300 DEC R0 ;CHECK FOR MORE UNITS
694 007066 001347 BNE 1$
695 007070 013700 001264 MOV UNITS, R0 ;SET UP UNIT COUNTER
696 007074 013701 001266 MOV DEVADR, R1 ;SET UP DEVICE ADDRESS POINTER
697 007100 012702 005312 MOV #MASK, R2 ;SET UP CHAR LEN TABLE POINTER
698 007104 012703 000004 4$: MOV #4, R3 ;SET UP CHAR COUNTER
699 007110 016104 000002 5$: MOV 2(R1), R4 ;SAVE AND CHECK CHAR PRESENT
700 007114 100375 BPL 5$
701 007116 010405 MOV R4, R5 ;DUP DATA
702 007120 000305 SWAB R5 ;LINE # IN LOW BYTE
703 007122 042705 177760 BIC #177760,R5 ;CLEAR ALL BUT LINE #
704 007126 006305 ASL R5 ;*2
705 007130 060205 ADD R2, R5 ;MAKE POINTER TO CHAR TABLE
706 007132 105104 COMB R4 ;MAKE DATA INTO MASK
707 007134 042704 177400 BIC #177400,R4 ;CLEAR UPPER BYTE
708 007140 010425 MOV R4, (R5)+ ;SAVE THE MASK
709 007142 010425 MOV R4, (R5)+ ;SAVE THE MASK
710 007144 010425 MOV R4, (R5)+ ;SAVE THE MASK
711 007146 010425 MOV R4, (R5)+ ;SAVE THE MASK
712 007150 005303 DEC R3 ;COUNT TO 4
713 007152 001356 BNE 5$
714 007154 062701 000010 ADD #10, R1 ;ADDRESS POINTER TO NEXT DJ
715 007160 062702 000040 ADD #40, R2 ;CHAR LEN TABLE POINTER
716 007164 005300 DEC R0 ;COUNT UNITS
717 007166 001346 BNE 4$ ;BRANCH BACK IF MORE

```

718
719
720
721
722
723
724
725
726 007170 005737 000042
727 007174 001040
728 007176 000004 016613
729 007202 004537 015030
730 007206 007340
731 007210 001367
732 007212 032777 000400 171770
733 007220 001426
734 007222 005000
735 007224 012701 001220
736 007230 000004 016442
737 007234 000004 016417
738 007240 010005
739 007242 004737 015404
740 007246 000004 016422
741 007252 004537 015030
742 007256 001214
743 007260 013721 001214
744 007264 005200
745 007266 020037 001264
746 007272 001356
747 007274 000410
748
749 007276 013700 001264
750 007302 012701 001220
751 007306 012721 177777
752 007312 005300
753 007314 001374
754
755
756 007316 013700 007340
757 007322 001722
758 007324 022700 000003
759 007330 103717
760 007332 006300
761 007334 000170 007340
762
763 007340
764 007340 000001
765 007342 007350
766 007344 012012
767 007346 013216

```

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

SELPRO: TST @#42 ;CHECK FOR AC1 11 OR DDP
        BNE ALL ;BRANCH IF MONITOR
        TYPE, MSGPRG
        JSR R5, READIN ;READ A NUMBER FROM THE CTY
        .WORD PROGNO
        BNE SELPRO
        BIT #BIT8, @SWR ;CHECK FOR SW<8>, SELECT LINES
        BEQ ALL ;BRANCH IF NOT
        CLR R0 ;SET UP UNIT COUNTER, DISPLAY
        MOV #SVSW0,R1 ;SET UP SWITCH TABLE POINTER
SWITCH: TYPE, RETURN
        TYPE, MNUM
        MOV R0,TTY ;PRINT THE NUMBER OF THE DR11C
        JSR PC,PRINTS
        TYPE, MSGSEL ;ASL FOR THE SELECTED LINES
        JSR R5,READIN
        .WORD SAVIT
        MOV SAVIT,(R1)+
        INC R0 ;COUNT UNITS
        CMP R0, UNITS ;CHECK FOR MORE UNITS
        BNE SWITCH ;BRANCH IF MORE
        BR RESTAR ;GO DO IT

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

.SBTTL RESTAR POINT

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

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

```



```

768
769
770
771
772
773
774
775
776
777
778 007350 000005
779 007352 005037 001276
780 007356 012706 001200
781 007362 052737 000340
782 007370 012701 001312
783 007374 012702 002000
784 007400 005021
785 007402 005302
786 007404 001375
787 007406 012702 000400
788 007412 005201
789 007414 105021
790 007416 005302
791 007420 001374
792
793
794
795
796
797
798
799 007422 005000
800 007424 013701 001266
801 007430 013702 001270
802 007434 012703 010400
803 007440 010322
804 007442 013722 001272
805 007446 022323
806 007450 010113
807 007452 062723 000002
808 007456 005723
809 007460 010322
810 007462 013722 001274
811 007466 022323
812 007470 010123
813 007472 005011
814 007474 052711 050510
815
816
817
818
819
820 007500 032711 000020
821 007504 001375
822 007506 005737 001300
823 007512 001002

```

```

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

```

```

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

```

```

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

```

```

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

```

```

824 007514 004737 010032      JSR      PC,ALMCK      :NO,GO DO IT
825 007520 006300              ASL      R0            :UNIT # * 2
826 007522 016061 001220 000004 10$:  MOV     SVSWO(R0),4(R1) :SET TCR BITS (CSR + 4)
827 007530 006200              ASR      R0            :RESTORE UNIT COUNTER
828 007532 012737 000001 001260  MOV     #1, MARK      :SET UP MARKER
829 007540 017705 171444      MOV     @SWR, R5       :GET SWITCH SETTINGS
830 007544 042705 177770      BIC     #177770,R5    :MASK MESSAGE #
831 007550 006305              ASL      R5
832 007552 006305              ASL      R5
833 007554 012304              MOV     (R3)+, R4     :SET UP OFFSET TO TABLES
834 007556 033761 001260 000004 14$:  BIT     MARK, 4(R1)   :CHECK FOR LINE SELECTED IN TCR
835 007564 001414              BEQ     15$
836 007566 016564 011236 001312  MOV     ADRNIT(5),XMTTAB(4)
837 007574 016564 011236 002312  MOV     ADRNIT(5),RCVTAB(4)
838 007602 116564 011240 004312  MOVB   CNTNIT(5),XMTCNT(4)
839 007610 116564 011240 004313  MOVB   CNTNIT(5),RCVCNT(4)
840 007616 005724              15$:  TST     (R4)+        :INC OFFSET TO NEXT LINE
841 007620 006337 001260      ASL     MARK
842 007624 103354              BCC     14$
843 007626 032777 001000 171354  BIT     #BIT9, @SWR   :CHECK FOR ON-LINE
844 007634 001024              BNE     21$           :BRANCH IF ON-LINE
845 007636 052711 000014      BIS     #14, (R1)    :SET THE MAINTENANCE BIT AND CLR MOS
846 007642 032711 000020 20$:  BIT     #BIT4,(R1)   :WAIT FOR MOS TO CLEAR
847 007646 001375              BNE     20$
848 007650 052711 000001      BIS     #1,(R1)     :TURN ON RCV ENABLE
849 007654 062701 000010 12$:  ADD     #10,R1       :POINT TO THE NEXT CSR
850 007660 005200              INC     R0
851 007662 020037 001264      CMP     R0, UNITS
852 007666 001264              BNE     11$
853 007670 012737 000001 001300  MOV     #1,ALMFLG    :SET THE ALARM LEVEL FLAG
854 007676 042737 000140 177776  BIC     #140, @MPS   :LOWER PROCESSOR PRIORITY
855 007704 000406              BR      FORGND       :GO DO IT
856
857 007706 052711 000001 21$:  BIS     #1,(R1)     :TURN ON RCV EN
858 007712 005761 000002 22$:  TST     2(R1)       :CLEAR JUNK OUT OF THE RBUF
859 007716 100775              BMI     22$
860 007720 000755              BR      12$
861
862
863
864
865
866
867
868 007722 012701 004312  FORGND: MOV     #XMTCNT,R1
869 007726 012702 000400      MOV     #400,R2
870 007732 105711 21$:  TSTB   (R1)         :CHECK FOR COUNT TABLE CLR
871 007734 001376              BNE     21$         :BRANCH IF NOT
872 007736 062701 000002      ADD     #2,R1       :GO TO NEXT LINE ENTRY
873 007742 005302              DEC     R2          :COUNT LINES
874 007744 001372              BNE     21$         :BRANCH IF MORE LINES
875 007746 012701 004313      MOV     #RCVCNT,R1
876 007752 012702 000400      MOV     #400,R2
877 007756 121137 001306 22$:  CMPB   (R1),COUNT  :IS # OF CHAR LEFT IN RBUF LESS THAN
878
879 007762 003375              BGT     22$         :THE SILO ALARM LEVEL
:IF NO WAIT FOR THEM

```

```

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

```

880	007764	062701	000002			ADD	#2,R1		:IF YES IGNORE THEM	
881	007770	005302				DEC	R2		:COUNT LINES	
882	007772	001371				BNE	22\$:BRANCH IF MORE LINES	
883	007774	005337	001216			DEC	TIMES		:DO THIS AGAIN?	
884	010000	001402				BEQ	23\$:NO, GET OUT	
885	010002	000137	007350			'MP	PROG1			
886	010006	005737	001276		23\$:	ST	ISRFLG		:SEE IF FLAG IS ZERO	::++F
887	010012	001002				JNE	1\$:IF NOT, BR	::++F
888	010014	000004	016366			TYPE,	TRNERR		:IF YES, RCV DATA ERROR	::++F
889	010020	012737	000020	001216	1\$:	MOV	#20,TIMES		:DO IT 16 TIMES THE NEXT TIME	::++F
890	010026	000137	014550			JMP	@#DONE		:SKIP ISR'S	
891	010032	012761	000001	000004	ALMCK:	MOV	#1,4(R1)		:SET LINE 0 IN THE TCR	
892	010040	052711	000004			BIS	#BIT2,(R1)		:SET THE MAINT BIT	
893	010044	052711	000001			BIS	#1,(R1)		:SET RCV EN AFTER MAINTENANCE BIT	
894	010050	005037	001306			CLR	COUNT			
895	010054	005037	001310			CLR	SUM			
896	010060	005037	001302			CLR	TIMERA			
897	010064	012737	000200	001304	2\$:	MOV	#200,TIMERB		:SET UP TIME CONSTANTS	
898	010072	005711				TST	(R1)		:WAIT FOR TRANSFER READY BIT	
899	010074	100373				BPL	2\$			
900	010076	112761	000377	000006		MOVB	#377,6(R1)		:OUTPUT A CHAR TO TBUF	
901	010104	005237	001306			INC	COUNT		:COUNT EACH CHAR	
902	010110	105711			1\$:	TSTB	(R1)		:CHECK FOR DONE IN THE CSR	
903	010112	100405				BMI	3\$:IF SET GET OUT OF THE LOOP	
904	010114	004537	010344			JSR	R5,TIME		:GIVE DONE TIME TO SET	
905	010120	000773				BR	1\$:RETURN TO TEST FOR DONE AGAIN	
906	010122	000760				BR	2\$:RETURN TO OUTPUT ANOTHER CHAR	
907	010124	000742				BR	ALMCK		:ERROR RETURN TRY AGAIN	
908	010126	042711	000001		3\$:	BIC	#BIT0,(R1)		:TURN OFF RCV ENABLE	
909	010132	052711	000010			BIS	#BIT3,(R1)		:CLEAR MOS	
910	010136	022737	000001	001306		CMP	#1,COUNT		:IF SILO LEVEL SET FOR 1 THEN GET OUT	
911	010144	001437				BEQ	4\$			
912	010146	063737	001306	001310	5\$:	ADD	COUNT,SUM		:GET THE LARGEST MULTIPLE OF THE	
913	010154	023727	001310	000106		CMP	SUM,#70.		:SILO ALARM LEVEL AND USE IT IN THE	
914	010162	002771				BLT	5\$:THREE SIZE LOCATIONS	
915	010164	001403				BEQ	6\$:IF EQUAL USE IT	
916	010166	163737	001306	001310		SUB	COUNT,SUM		:IF GREATER SUBTRACT ONE COUNT OFF	
917	010174	013737	001310	011244	6\$:	MOV	SUM,CNTNIT+4		:AS CLOSE TO 70 AS POSSIBLE	
918	010202	013737	001310	011250		MOV	SUM,CNTNIT+10			
919	010210	063737	001306	001310	7\$:	ADD	COUNT,SUM		:CONTINUE TO A COUNT OF 256	
920	010216	023727	001310	000377		CMP	SUM,#377			
921	010224	002771				BLT	7\$			
922	010226	163737	001306	001310		SUB	COUNT,SUM			
923	010234	013737	001310	011240	10\$:	MOV	SUM,CNTNIT		:AS CLOSE TO 256 AS POSSIBLE	
924	010242	000411				BR	11\$:ALL SET GET OUT	
925	010244	012737	000377	011240	4\$:	MOV	#377,CNTNIT		:PUT SIZE TO MAX VALUE	
926	010252	012737	000106	011244		MOV	#70.,CNTNIT+4			
927	010260	012737	000106	011250		MOV	#70.,CNTNIT+10			
928	010266	042711	000004		11\$:	BIC	#BIT2,(R1)		:TURN OFF THE MAINT BIT	
929	010272	032711	000020		12\$:	BIT	#BIT4,(R1)		:WAIT FOR MOS TO CLEAR	
930	010276	001375				BNE	12\$			
931	010300	004737	016316			JSR	PC,KBDINT		:GET THE SWITCH REGISTER	
932	010304	032777	010000	170676		BIT	#SW12,@SWR		:PRINT ALARM LEVEL?	
933	010312	001413				BEQ	13\$:NO	
934	010314	000004	016741			TYPE,	MALARM			
935	010320	010105				MOV	R1,TTY		:YES, PRINT CSR FIRST	

```

936 010322 004737 015374 JSR PC,PRINTR
937 010326 000004 016735 TYPE, MSGDAS
938 010332 013705 001306 MOV COUNT,TTY ;PRINT ALARM LEVEL
939 010336 004737 015374 JSR PC,PRINTR
940 010342 000207 13$: RTS PC
941
942
943
944 010344 105237 001302 TIME: INCB TIMERA ;INCREMENT THROUGH ONE WORD
945 010350 001012 BNE 1$ ;GO TEST FOR DONE AGAIN
946 010352 005337 001304 DEC TIMERB ;MAKE TIMERB LARGER IF FAST PROCESSOR
947 010356 001007 BNE 1$
948 010360 023727 001306 000022 CMP COUNT,#22 ;HAVE OUTPUTTED 18 TIMES
949 010366 001002 BNE 2$ ;NO, GO OUTPUT ANOTHER CHAR
950 010370 104001 HLT+1 ;YES,DONE DID NOT SET AFTER 18 OUTPUTS
951 ;R1 = CSR
952 010372 005725 TST (R5)+ ;SET R5 FOR ERROR RETURN
953 010374 005725 2$: TST (R5)+ ;SET R5 FOR NEXT OUTPUT RETURN
954 010376 000205 1$: RTS R5 ;RETURN FROM ABOVE OR RETEST DONE
955
956
957 ;*****
958 ;PROG1 LINKERS TO DJ11 INTERRUPT SERVICE ROUTINES
959 ;*****
960 010400 004037 011114 RISR0: JSR R0,RCVISR
961 010404 160012 000000 .WORD <160012+<0*10>>,<40*0>
962 010410 004037 011000 XISR0: JSR R0,XMTISR
963 010414 160010 000000 .WORD <160010+<0*10>>,<40*0>
964 010420 004037 011114 RISR1: JSR R0,RCVISR
965 010424 160022 000040 .WORD <160022+<1*10>>,<40*1>
966 010430 004037 011000 XISR1: JSR R0,XMTISR
967 010434 160020 000040 .WORD <160020+<1*10>>,<40*1>
968 010440 004037 011114 RISR2: JSR R0,RCVISR
969 010444 160032 000100 .WORD <160032+<2*10>>,<40*2>
970 010450 004037 011000 XISR2: JSR R0,XMTISR
971 010454 160030 000100 .WORD <160030+<2*10>>,<40*2>
972 010460 004037 011114 RISR3: JSR R0,RCVISR
973 010464 160042 000140 .WORD <160042+<3*10>>,<40*3>
974 010470 004037 011000 XISR3: JSR R0,XMTISR
975 010474 160040 000140 .WORD <160040+<3*10>>,<40*3>
976 010500 004037 011114 RISR4: JSR R0,RCVISR
977 010504 160052 000200 .WORD <160052+<4*10>>,<40*4>
978 010510 004037 011000 XISR4: JSR R0,XMTISR
979 010514 160050 000200 .WORD <160050+<4*10>>,<40*4>
980 010520 004037 011114 RISR5: JSR R0,RCVISR
981 010524 160062 000240 .WORD <160062+<5*10>>,<40*5>
982 010530 004037 011000 XISR5: JSR R0,XMTISR
983 010534 160060 000240 .WORD <160060+<5*10>>,<40*5>
984 010540 004037 011114 RISR6: JSR R0,RCVISR
985 010544 160072 000300 .WORD <160072+<6*10>>,<40*6>
986 010550 004037 011000 XISR6: JSR R0,XMTISR
987 010554 160070 000300 .WORD <160070+<6*10>>,<40*6>
988 010560 004037 011114 RISR7: JSR R0,RCVISR
989 010564 160102 000340 .WORD <160102+<7*10>>,<40*7>
990 010570 004037 011000 XISR7: JSR R0,XMTISR
991 010574 160100 000340 .WORD <160100+<7*10>>,<40*7>

```

```

992 010600 004037 011114
993 010604 160112 000400
994 010610 004037 011000
995 010614 160110 000400
996 010620 004037 011114
997 010624 160122 000440
998 010630 004037 011000
999 010634 160120 000440
1000 010640 004037 011114
1001 010644 160132 000500
1002 010650 004037 011000
1003 010654 160130 000500
1004 010660 004037 011114
1005 010664 160142 000540
1006 010670 004037 011000
1007 010674 160140 000540
1008 010700 004037 011114
1009 010704 160152 000600
1010 010710 004037 011000
1011 010714 160150 000600
1012 010720 004037 011114
1013 010724 160162 000640
1014 010730 004037 011000
1015 010734 160160 000640
1016 010740 004037 011114
1017 010744 160172 000700
1018 010750 004037 011000
1019 010754 160170 000700
1020 010760 004037 011114
1021 010764 160202 000740
1022 010770 004037 011000
1023 010774 160200 000740
1024
1025
1026
1027
1028
1029 011000
1030 011000 010146
1031 011002 010246
1032 011004 012001
1033 011006 005711
1034 011010 100035
1035 011012 116102 000007
1036 011016 006302
1037 011020 061002
1038 011022 105762 004312
1039 011026 001410
1040 011030 117261 001312 000006
1041 011036 105362 004312
1042 011042 005262 001312
1043 011046 000757
1044 011050 010346
1045 011052 005062 001312
1046 011056 161002
1047 011060 006202

```

```

RISR10: JSR RO,RCVISR
        .WORD <160012+<10*10>>,<40*10>
XISR10: JSR RO,XMTISR
        .WORD <160010+<10*10>>,<40*10>
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>

```

```

:*****
:PROG1 TRANSMITTER INTERRUPT SERVICE ROUTINE
:*****

```

```

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

```

```

1048 011062 005003      CLR      R3
1049 011064 000261      SEC
1050 011066 006103      3$:     ROL      R3
1051 011070 005302      DEC      R2
1052 011072 100375      BPL      3$
1053 011074 040361 000004    BIC      R3,4(R1)      ;CLEAR TCR BIT FOR LINE
1054 011100 012603      MOV      (SP)+,R3      ;RESTORE R3
1055 011102 000741      BR
1056 011104
1057 011104 012602      4$:     MOV      (6)+,R2      ;POP STACK INTO R2
1058 011106 012601      MOV      (6)+,R1      ;POP STACK INTO R1
1059 011110 012600      MOV      (6)+,R0      ;POP STACK INTO R0
1060 011112 000002      RTI

```

```

:*****
:PROG1 RECEIVER INTERRUPT SERVICE ROUTINE
:*****

```

```

1066 011114
1067 011114 010146      RCVISR:  MOV      R1,-(6)      ;PUSH R1 ON STACK
1068 011116 010246      MOV      R2,-(6)      ;PUSH R2 ON STACK
1069 011120 010346      MOV      R3,-(6)      ;PUSH R3 ON STACK
1070 011122 010446      MOV      R4,-(6)      ;PUSH R4 ON STACK
1071 011124 012001      MOV      (R0)+,R1      ;GET RBUF ADDRESS
1072 011126 011102      1$:     MOV      (R1),R2      ;READ THE DATA
1073 011130 100032      BPL      7$            ;BRANCH IF NO CHAR PRESENT
1074 011132 032702 070000    BIT      #70000,R2      ;CHECK FOR ERRORS
1075 011136 001403      BEQ      2$            ;BRANCH IF OK
1076 011140 104002      HLT+2                ;RECEIVER ERROR
1077
1078
1079
1080
1081
1082 011142 042702 070000      2$:     BIC      #70000, R2      ;CLEAR ERROR BITS FOR SPEED
1083 011146 010204      MOV      R2, R4        ;DUP THE RBUF
1084 011150 105004      CLRB     R4            ;CLEAR THE DATA
1085 011152 000304      SWAB    R4            ;LINE # TO LOW BYTE
1086 011154 106304      ASLB    R4            ;LINE # * 2, ALSO CLR CHAR PRESENT
1087 011156 061004      ADD     (R0),R4        ;ADD OFFSET
1088 011160 117403 002312    MOVB    @RCVTAB(R4),R3 ;GET EXPECTED DATA
1089 011164 046403 005312    BIC     MASK(4),R3     ;MASK CHARACTER LENGTH
1090 011170 120302      CMPB    R3,R2
1091 011172 001403      BEQ     3$
1092 011174 042703 177400    BIC     #177400,R3     ;BRANCH IF OK
1093 011200 104003      HLT+3                ;MAKE SURE UPPER BYTE CLEAR
1094
1095
1096
1097 011202 105364 004313      3$:     DECB    RCVCNT(4)     ;DATA ERROR
1098 011206 001403      BEQ     7$            ;R1=RBUF ADDRESS
1099 011210 005264 002312    INC     RCVTAB(4)     ;R2=CONTENTS OF RBUF(DATA)
1100 011214 000744      BR      1$            ;R3=EXPECTED DATA
1101 011216
1102 011216 012604      7$:     MOV     (6)+,R4        ;POP STACK INTO R4
1103 011220 012603      MOV     (6)+,R3        ;POP STACK INTO R3

```

1104 011222 012602
 1105 011224 012601
 1106 011226 012600
 1107 011230 005237 001276
 1108 011234 000002
 1109
 1110
 1111
 1112
 1113
 1114 011236 011302
 1115 011240 000377
 1116 011242 011702
 1117 011244 000106
 1118 011246 011274
 1119 011250 000106
 1120 011252 016776
 1121 011254 000001
 1122 011256 017376
 1123 011260 000001
 1124 011262 017776
 1125 011264 000001
 1126 011266 020376
 1127 011270 000001
 1128 011272 020776
 1129 011274 005015 177777 177777
 1130 011302 040
 1131 011303 041
 1132 011304 042
 1133 011305 043
 1134 011306 044
 1135 011307 045
 1136 011310 046
 1137 011311 047
 1138 011312 050
 1139 011313 051
 1140 011314 052
 1141 011315 053
 1142 011316 054
 1143 011317 055
 1144 011320 056
 1145 011321 057
 1146 011322 060
 1147 011323 061
 1148 011324 062
 1149 011325 063
 1150 011326 064
 1151 011327 065
 1152 011330 066
 1153 011331 067
 1154 011332 070
 1155 011333 071
 1156 011334 072
 1157 011335 073
 1158 011336 074
 1159 011337 075

MOV (6)+,R2 ;POP STACK INTO R2
 MOV (6)+,R1 ;POP STACK INTO R1
 MOV (6)+,R0 ;POP STACK INTO R0
 INC ISRFLG ;STEP INT SVC RTN FLAG
 RTI

::++F

 :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

1
 END+400
 1
 END+1000
 1
 END+1400
 1
 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

1160	011340	076	.BYTE	76
1161	011341	077	.BYTE	77
1162	011342	100	.BYTE	100
1163	011343	101	.BYTE	101
1164	011344	102	.BYTE	102
1165	011345	103	.BYTE	103
1166	011346	104	.BYTE	104
1167	011347	105	.BYTE	105
1168	011350	106	.BYTE	106
1169	011351	107	.BYTE	107
1170	011352	110	.BYTE	110
1171	011353	111	.BYTE	111
1172	011354	112	.BYTE	112
1173	011355	113	.BYTE	113
1174	011356	114	.BYTE	114
1175	011357	115	.BYTE	115
1176	011360	116	.BYTE	116
1177	011361	117	.BYTE	117
1178	011362	120	.BYTE	120
1179	011363	121	.BYTE	121
1180	011364	122	.BYTE	122
1181	011365	123	.BYTE	123
1182	011366	124	.BYTE	124
1183	011367	125	.BYTE	125
1184	011370	126	.BYTE	126
1185	011371	127	.BYTE	127
1186	011372	130	.BYTE	130
1187	011373	131	.BYTE	131
1188	011374	132	.BYTE	132
1189	011375	133	.BYTE	133
1190	011376	134	.BYTE	134
1191	011377	135	.BYTE	135
1192	011400	136	.BYTE	136
1193	011401	137	.BYTE	137
1194	011402	140	.BYTE	140
1195	011403	141	.BYTE	141
1196	011404	142	.BYTE	142
1197	011405	143	.BYTE	143
1198	011406	144	.BYTE	144
1199	011407	145	.BYTE	145
1200	011410	146	.BYTE	146
1201	011411	147	.BYTE	147
1202	011412	150	.BYTE	150
1203	011413	151	.BYTE	151
1204	011414	152	.BYTE	152
1205	011415	153	.BYTE	153
1206	011416	154	.BYTE	154
1207	011417	155	.BYTE	155
1208	011420	156	.BYTE	156
1209	011421	157	.BYTE	157
1210	011422	160	.BYTE	160
1211	011423	161	.BYTE	161
1212	011424	162	.BYTE	162
1213	011425	163	.BYTE	163
1214	011426	164	.BYTE	164
1215	011427	165	.BYTE	165

1216	011430	166	.BYTE	166
1217	011431	167	.BYTE	167
1218	011432	170	.BYTE	170
1219	011433	171	.BYTE	171
1220	011434	172	.BYTE	172
1221	011435	173	.BYTE	173
1222	011436	174	.BYTE	174
1223	011437	175	.BYTE	175
1224	011440	176	.BYTE	176
1225	011441	177	.BYTE	177
1226	011442	200	.BYTE	200
1227	011443	201	.BYTE	201
1228	011444	202	.BYTE	202
1229	011445	203	.BYTE	203
1230	011446	204	.BYTE	204
1231	011447	205	.BYTE	205
1232	011450	206	.BYTE	206
1233	011451	207	.BYTE	207
1234	011452	210	.BYTE	210
1235	011453	211	.BYTE	211
1236	011454	212	.BYTE	212
1237	011455	213	.BYTE	213
1238	011456	214	.BYTE	214
1239	011457	215	.BYTE	215
1240	011460	216	.BYTE	216
1241	011461	217	.BYTE	217
1242	011462	220	.BYTE	220
1243	011463	221	.BYTE	221
1244	011464	222	.BYTE	222
1245	011465	223	.BYTE	223
1246	011466	224	.BYTE	224
1247	011467	225	.BYTE	225
1248	011470	226	.BYTE	226
1249	011471	227	.BYTE	227
1250	011472	230	.BYTE	230
1251	011473	231	.BYTE	231
1252	011474	232	.BYTE	232
1253	011475	233	.BYTE	233
1254	011476	234	.BYTE	234
1255	011477	235	.BYTE	235
1256	011500	236	.BYTE	236
1257	011501	237	.BYTE	237
1258	011502	240	.BYTE	240
1259	011503	241	.BYTE	241
1260	011504	242	.BYTE	242
1261	011505	243	.BYTE	243
1262	011506	244	.BYTE	244
1263	011507	245	.BYTE	245
1264	011510	246	.BYTE	246
1265	011511	247	.BYTE	247
1266	011512	250	.BYTE	250
1267	011513	251	.BYTE	251
1268	011514	252	.BYTE	252
1269	011515	253	.BYTE	253
1270	011516	254	.BYTE	254
1271	011517	255	.BYTE	255

CZDJBG DJ11 EXFR & ONLINE
CZDJBG.P11 07-JUN-82 16:32MACY11 30A(1052) 07-JUN-82
DATA TABLES

16:36 PAGE 26

SEQ 25

1272	011520	256	.BYTE	256
1273	011521	257	.BYTE	257
1274	011522	260	.BYTE	260
1275	011523	261	.BYTE	261
1276	011524	262	.BYTE	262
1277	011525	263	.BYTE	263
1278	011526	264	.BYTE	264
1279	011527	265	.BYTE	265
1280	011530	266	.BYTE	266
1281	011531	267	.BYTE	267
1282	011532	270	.BYTE	270
1283	011533	271	.BYTE	271
1284	011534	272	.BYTE	272
1285	011535	273	.BYTE	273
1286	011536	274	.BYTE	274
1287	011537	275	.BYTE	275
1288	011540	276	.BYTE	276
1289	011541	277	.BYTE	277
1290	011542	300	.BYTE	300
1291	011543	301	.BYTE	301
1292	011544	302	.BYTE	302
1293	011545	303	.BYTE	303
1294	011546	304	.BYTE	304
1295	011547	305	.BYTE	305
1296	011550	306	.BYTE	306
1297	011551	307	.BYTE	307
1298	011552	310	.BYTE	310
1299	011553	311	.BYTE	311
1300	011554	312	.BYTE	312
1301	011555	313	.BYTE	313
1302	011556	314	.BYTE	314
1303	011557	315	.BYTE	315
1304	011560	316	.BYTE	316
1305	011561	317	.BYTE	317
1306	011562	320	.BYTE	320
1307	011563	321	.BYTE	321
1308	011564	322	.BYTE	322
1309	011565	323	.BYTE	323
1310	011566	324	.BYTE	324
1311	011567	325	.BYTE	325
1312	011570	326	.BYTE	326
1313	011571	327	.BYTE	327
1314	011572	330	.BYTE	330
1315	011573	331	.BYTE	331
1316	011574	332	.BYTE	332
1317	011575	333	.BYTE	333
1318	011576	334	.BYTE	334
1319	011577	335	.BYTE	335
1320	011600	336	.BYTE	336
1321	011601	337	.BYTE	337
1322	011602	340	.BYTE	340
1323	011603	341	.BYTE	341
1324	011604	342	.BYTE	342
1325	011605	343	.BYTE	343
1326	011606	344	.BYTE	344
1327	011607	345	.BYTE	345

1328	011610	346	.BYTE	346
1329	011611	347	.BYTE	347
1330	011612	350	.BYTE	350
1331	011613	351	.BYTE	351
1332	011614	352	.BYTE	352
1333	011615	353	.BYTE	353
1334	011616	354	.BYTE	354
1335	011617	355	.BYTE	355
1336	011620	356	.BYTE	356
1337	011621	357	.BYTE	357
1338	011622	360	.BYTE	360
1339	011623	361	.BYTE	361
1340	011624	362	.BYTE	362
1341	011625	363	.BYTE	363
1342	011626	364	.BYTE	364
1343	011627	365	.BYTE	365
1344	011630	366	.BYTE	366
1345	011631	367	.BYTE	367
1346	011632	370	.BYTE	370
1347	011633	371	.BYTE	371
1348	011634	372	.BYTE	372
1349	011635	373	.BYTE	373
1350	011636	374	.BYTE	374
1351	011637	375	.BYTE	375
1352	011640	376	.BYTE	376
1353	011641	377	.BYTE	377
1354	011642	000	.BYTE	0
1355	011643	001	.BYTE	1
1356	011644	002	.BYTE	2
1357	011645	003	.BYTE	3
1358	011646	004	.BYTE	4
1359	011647	005	.BYTE	5
1360	011650	006	.BYTE	6
1361	011651	007	.BYTE	7
1362	011652	010	.BYTE	10
1363	011653	011	.BYTE	11
1364	011654	012	.BYTE	12
1365	011655	013	.BYTE	13
1366	011656	014	.BYTE	14
1367	011657	015	.BYTE	15
1368	011660	016	.BYTE	16
1369	011661	017	.BYTE	17
1370	011662	020	.BYTE	20
1371	011663	021	.BYTE	21
1372	011664	022	.BYTE	22
1373	011665	023	.BYTE	23
1374	011666	024	.BYTE	24
1375	011667	025	.BYTE	25
1376	011670	026	.BYTE	26
1377	011671	027	.BYTE	27
1378	011672	030	.BYTE	30
1379	011673	031	.BYTE	31
1380	011674	032	.BYTE	32
1381	011675	033	.BYTE	33
1382	011676	034	.BYTE	34
1383	011677	035	.BYTE	35

1384	011700	036		
1385	011701	037		
1386				
1387	011702	005015	177777	177777
1388	011710	044124	020105	052521
1389	011716	041511	020113	044523
1390	011724	053114	051105	043440
1391	011732	040522	020131	047506
1392	011740	020130	052512	050115
1393	011746	042105	047440	042526
1394	011754	020122	026071	033470
1395	011762	026066	032065	026063
1396	011770	030462	027060	020060
1397	011776	040514	054532	042040
1398	012004	043517	020523	000
1399		012012		

.BYTE 36
.BYTE 37

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

.EVEN

1400
1401
1402
1403
1404
1405
1406
1407
1408 012012 000005
1409 012014 012706 001200
1410 012020 052737 000340 177776
1411 012026 012701 001312
1412 012032 012702 002000
1413 012036 005021
1414 012040 005302
1415 012042 001375
1416 012044 012702 000400
1417 012050 005201
1418 012052 105021
1419 012054 005302
1420 012056 001374
1421
1422
1423
1424
1425
1426
1427
1428 012060 005000
1429 012062 013701 001266
1430 012066 013702 001270
1431 012072 012703 012250
1432 012076 010322
1433 012100 013722 001272
1434 012104 022323
1435 012106 010113
1436 012110 062723 000002
1437 012114 005723
1438 012116 010322
1439 012120 013722 001274
1440 012124 022323
1441 012126 010123
1442 012130 012721 050501
1443
1444
1445
1446
1447
1448 012134 005721
1449 012136 006300
1450 012140 016011 001220
1451 012144 006200
1452 012146 012737 000001 001260
1453 012154 012304
1454 012156 033711 001260
1455 012162 001406

```

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

```

```

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

```

```

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

```

```

P2INIT: CLR    R0
        MOV    DEVADR, R1
        MOV    VECADR, R2
        MOV    #R2SRO,R3
1$:     MOV    R3, (R2)+     ;SET UP POINTER TO LINKERS
        MOV    RCVLVL, (R2)+ ;SET UP RECEIVER INTERUPT VECTOR
        CMP    (R3)+, (R3)+ ;ADD 4 TO R3
        MOV    R1, (R3)     ;ADDRESS OF CSR
        ADD    #2, (R3)+    ;ADDRESS OF RBUF
        TST   (R3)+
        MOV    R3, (R2)+     ;SET UP TRANSMITTER INTERUPT VECTOR
        MOV    XMTLVL, (R2)+
        CMP    (R3)+, (R3)+
        MOV    R1, (R3)+    ;ADDRESS OF CSR
        MOV    #50501, (R1)+ ;SET UP CSR
                               ;BIT0 = RECEIVER ENABLE
                               ;BIT6 = RECEIVER INTERUPT ENABLE
                               ;BIT8 = TRANSMITTER SCAN ENABLE
                               ;BIT12 = STATUS ENABLE
                               ;BIT14 = TRANSMITTER INTERUPT ENABLE
        TST   (R1)+
        ASL   R0
        MOV    SVSWO(R0), (R1) ;UNIT # * 2
        MOV    R0
        ASR   R0            ;SET TCR BITS FOR SELECTED LINES
        MOV    #1, MARK     ;RESET UNIT COUNTER
        MOV    (R3)+, R4    ;SET UP MARKER
4$:     BIT   MARK, (R1)    ;SET UP OFFSET TO TABLES
        BEQ  5$            ;CHECK FOR LINE SELECTED
        MOV    MARK, (R1)
        BEQ  5$

```

CZDJBG0 DJ11 EXER & ONLINE
CZDJBG.P11 07-JUN-82 16:32

MACY11 30A(1052) 07-JUN-82 16:36 PAGE 30
PROG2: ON-LINE EXERCISER (TRANSMIT LAST CHARACTER RECEIVED)

1456	012164	012764	016647	001312
1457	012172	012764	000045	004312
1458	012200	005724		
1459	012202	006337	001260	
1460	012206	103363		
1461	012210	022121		
1462	012212	005200		
1463	012214	020037	001264	
1464	012220	001326		
1465	012222	042737	000140	177776
1466				
1467				
1468				
1469				
1470				
1471	012230	012700	020000	
1472	012234	000241		
1473	012236	005540		
1474	012240	001376		
1475	012242	005700		
1476	012244	001374		
1477	012246	000770		
1478				
1479				
1480				
1481				
1482				
1483	012250	004037	013042	
1484	012254	160012	000000	
1485	012260	004037	012650	
1486	012264	160020	000000	
1487	012270	004037	013042	
1488	012274	160022	000040	
1489	012300	004037	012650	
1490	012304	160030	000040	
1491	012310	004037	013042	
1492	012314	160032	000100	
1493	012320	004037	012650	
1494	012324	160040	000100	
1495	012330	004037	013042	
1496	012334	160042	000140	
1497	012340	004037	012650	
1498	012344	160050	000140	
1499	012350	004037	013042	
1500	012354	160052	000200	
1501	012360	004037	012650	
1502	012364	160060	000200	
1503	012370	004037	013042	
1504	012374	160062	000240	
1505	012400	004037	012650	
1506	012404	160070	000240	
1507	012410	004037	013042	
1508	012414	160072	000300	
1509	012420	004037	012650	
1510	012424	160100	000300	
1511	012430	004037	013042	

```

MOV #M5GP2, XMTTAB(4) ;SET UP XMTR TABLE
MOV #45, XMTCNT(4) ;SET UP COUNT
5$: TST (R4)+ ;INC OFFSET TO NEXT LINE
ASL MARK
BCJ 4$
CMP (R1)+, (R1)+ ;ADD 4
INC R0
CMP R0, UNITS
BNE 1$
BIC #140, @#PS ;LOWER PROCESSOR PRIORITY

```

```

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

```

```

FORP2: MOV #20000,R0 ;TOP OF 4K BANK OF MEMORY
CLC
1$: ADC -(R0) ;FAST READ/WRITE TO MEMORY
BNE 1$ ;RAPID REPEAT
TST R0 ;CHECK FOR LOC 0
BNE 1$ ;BRANCH IF MORE MEMORY
BR FORP2 ;LOOP FOR EVER!

```

```

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

```

```

R2SR0: JSR R0,P2RISR
.WORD <160012+<0*10>>,<0*40>
X2SR0: JSR R0,P2XISR
.WORD <160020+<0*10>>,<0*40>
R2SR1: JSR R0,P2RISR
.WORD <160012+<1*10>>,<1*40>
X2SR1: JSR R0,P2XISR
.WORD <160020+<1*10>>,<1*40>
R2SR2: JSR R0,P2RISR
.WORD <160012+<2*10>>,<2*40>
X2SR2: JSR R0,P2XISR
.WORD <160020+<2*10>>,<2*40>
R2SR3: JSR R0,P2RISR
.WORD <160012+<3*10>>,<3*40>
X2SR3: JSR R0,P2XISR
.WORD <160020+<3*10>>,<3*40>
R2SR4: JSR R0,P2RISR
.WORD <160012+<4*10>>,<4*40>
X2SR4: JSR R0,P2XISR
.WORD <160020+<4*10>>,<4*40>
R2SR5: JSR R0,P2RISR
.WORD <160012+<5*10>>,<5*40>
X2SR5: JSR R0,P2XISR
.WORD <160020+<5*10>>,<5*40>
R2SR6: JSR R0,P2RISR
.WORD <160012+<6*10>>,<6*40>
X2SR6: JSR R0,P2XISR
.WORD <160020+<6*10>>,<6*40>
R2SR7: JSR R0,P2RISR

```

1512 012434 160102 000340
 1513 012440 004037 012650
 1514 012444 160110 000340
 1515 012450 004037 013042
 1516 012454 160112 000400
 1517 012460 004037 012650
 1518 012464 160120 000400
 1519 012470 004037 013042
 1520 012474 160122 000440
 1521 012500 004037 012650
 1522 012504 160130 000440
 1523 012510 004037 013042
 1524 012514 160132 000500
 1525 012520 004037 012650
 1526 012524 160140 000500
 1527 012530 004037 013042
 1528 012534 160142 000540
 1529 012540 004037 012650
 1530 012544 160150 000540
 1531 012550 004037 013042
 1532 012554 160152 000600
 1533 012560 004037 012650
 1534 012564 160160 000600
 1535 012570 004037 013042
 1536 012574 160162 000640
 1537 012600 004037 012650
 1538 012604 160170 000640
 1539 012610 004037 013042
 1540 012614 160172 000700
 1541 012620 004037 012650
 1542 012624 160200 000700
 1543 012630 004037 013042
 1544 012634 160202 000740
 1545 012640 004037 012650
 1546 012644 160210 000740
 1547
 1548
 1549
 1550
 1551

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>

 :PROG2 TRANSMITTER INTERRUPT SERVICE ROUTINE

1552 012650
 1553 012650 010146
 1554 012652 010246
 1555 012654 012001
 1556 012656 005711
 1557 012660 100064
 1558 012662 116102 000007
 1559 012666 006302
 1560 012670 061002
 1561 012672 105762 004312
 1562 012676 001413
 1563 012700 117261 001312 000006
 1564 012706 105362 004312
 1565 012712 105762 005313
 1566 012716 001357
 1567 012720 005262 001312

P2XISR:
 MOV R1,-(6) :PUSH R1 ON STACK
 MOV R2,-(6) :PUSH R2 ON STACK
 MOV (R0)+,R1
 1\$: TST (R1) :CHECK FOR TRANS READY
 BPL 4\$
 MOVB 7(R1),R2 :GET LINE NO.
 ASL R2
 ADD (R0),R2
 TSTB 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

```

1568 012724 000754
1569 012726 105162 005313
1570 012732 001430
1571 012734 012762 002312 U01312
1572 012742 060262 001312
1573 012746 112762 000110 004312
1574 012754 105762 002312
1575 012760 001336
1576 012762 161002
1577 012764 006202
1578 012766 005037 001260
1579 012772 000261
1580 012774 006137 001260
1581 013000 005302
1582 013002 100374
1583 013004 043761 001260 000004
1584 013012 000721
1585 013014 012762 016442 001312
1586 013022 112762 000002 004312
1587 013030 000712
1588 013032
1589 013032 012602
1590 013034 012601
1591 013036 012600
1592 013040 000002
1593
1594
1595
1596
1597
1598 013042
1599 013042 010146
1600 013044 010246
1601 013046 010346
1602 013050 012001
1603 013052 011102
1604 013054 100053
1605 013056 032702 070000
1606 013062 001402
1607 013064 104302
1608
1609
1610
1611
1612
1613 013066 000771
1614
1615 013070 010203
1616 013072 105003
1617 013074 000303
1618 013076 106303
1619 013100 061003
1620 013102 136302 005312
1621 013106 001401
1622 013110 104002
1623

2$: BR 1$
COMB CNTTAB(2) ;CHANGE FLAG
BEQ 3$ ;BRANCH IF WAS DATA
MOV #RCVTAB,XMTTAB(2) ;SET UP POINTER TO RECEIVER TABLE
ADD R2,XMTTAB(2) ;ADD OFFSET
MOVB #72,XMTCNT(2) ;COUNT 72 CHARACTERS TO THE LINE
TSTB RCVTAB(2) ;CHECK FOR A BREAK
BNE 1$ ;BRANCH IF REAL DATA
SUB (R0),R2 ;RECOVER LINE NUMBER
ASR R2
CLR MARK ;SET UP MARKER
5$: ROL MARK ;MOVE MARKER
DEC R2 ;COUNT LINES
BPL 5$ ;BRANCH IF MORE
BIC MARK,4(R1) ;CLEAR TCR BIT
BR 1$ ;CONTINUE
3$: MOV #RETURN,XMTTAB(2) ;TYPE CARRIAGE RETURN, LINE FEED
MOVB #2,XMTCNT(2) ;COUNTER OF 2 CHARACTERS
BR 1$
4$: MOV (6)+,R2 ;POP STACK INTO R2
MOV (6)+,R1 ;POP STACK INTO R1
MOV (6)+,R0 ;POP STACK INTO R0
RTI

;*****
;PROG2 RECEIVER INTERRUPT SERVICE ROUTINE
;*****

P2RISR:
MOV R1,-(6) ;PUSH R1 ON STACK
MOV R2,-(6) ;PUSH R2 ON STACK
MOV R3,-(6) ;PUSH R3 ON STACK
MOV (R0)+,R1 ;GET RBUF ADDRESS
1$: MOV (R1),R2 ;READ THE DATA
BPL 7$ ;BRANCH IF NO CHAR PRESENT
BIT #70000,R2 ;CHECK FOR ERRORS
BEQ 2$ ;BRANCH IF OK
HLT+2 ;RECEIVER ERROR
;R1=RBUF ADDRESS
;R2=CONTENTS OF RBUF
;BIT12=PARITY ERROR
;BIT13=FRAMING ERROR
;BIT14=UART OVERRUN
;FORGET THE DATA
2$: MOV R2,R3 ;DUP THE RBUF
CLRB R3 ;CLEAR THE DATA
SWAB R3 ;LINE # TO LOW BYTE
ASLB R3 ;LINE # * 2, ALSO CLR CHAR PRESENT
ADD (R0),R3 ;ADD OFFSET
BITB MASK(3),R2 ;CHECK CHARACTER LENGTH
BEQ 3$ ;BRANCH IF OK
HLT+2 ;CHARACTER LENGTH ERROR
;R1=RBUF ADDRESS

```



```

1624
1625 013112 105763 002312      3$:  TSTB   RCVTAB(3)      ;R2=CONTENTS OF RBUF(DATA)
1626 013116 001017              BNE     5$           ;CHECK FOR BREAK
1627 013120 110263 002312      MOVB   R2, RCVTAB(3) ;SAVE THE DATA
1628 013124 161003              SUB     (R0), R3    ;RECOVER LINE NUMBER
1629 013126 006203              ASR    R3
1630 013130 005037 001260      CLR    MARK        ;SET UP MARKER
1631 013134 000261              SEC
1632 013136 006137 001260      4$:  ROL    MARK        ;UPDATE MARKER
1633 013142 005303              DEC    R3          ;COUNT LINES
1634 013144 100374              BPL    4$          ;BRANCH IF MORE
1635 013146 053761 001260 000002  BIS    MARK, 2(n1) ;SET TCR BIT
1636 013154 000736              BR     1$          ;CONTINUE
1637
1638 013156 110263 002312      5$:  MOVB   R2, RCVTAB(3) ;SAVE THE DATA
1639 013162 105163 005313      COMB   CNTTAB(3)   ;SET MESSAGE FLAG
1640 013166 012763 016442 001312  MOV    #RETURN,XMTTAB(3) ;TYPE CARRIAGE RETURN, LINE FEED
1641 013174 112763 000002 004312  MOVB   #2, XMTCNT(3) ;MESSAGE LENGTH
1642 013202 000723
1643 013204
1644 013204 012603      7$:  MOV    (6)+,R3     ;POP STACK INTO R3
1645 013206 012602      MOV    (6)+,R2     ;POP STACK INTO R2
1646 013210 012601      MOV    (6)+,R1     ;POP STACK INTO R1
1647 013212 012600      MOV    (6)+,R0     ;POP STACK INTO R0
1648 013214 000002      RTI

```

```

1649
1650
1651
1652
1653
1654 013216 000005
1655 013220 012706 001200
1656 013224 052737 000340 177776
1657 013232 012701 001312
1658 013236 012702 002000
1659 013242 005021
1660 013244 005302
1661 013246 001375
1662 013250 012702 000400
1663 013254 005201
1664 013256 105021
1665 013260 005302
1666 013262 001374
1667 013264 012705 016776
1668
1669
1670
1671
1672
1673
1674
1675 013270 005000
1676 013272 013701 001266
1677 013276 013702 001270
1678 013302 012703 013570
1679 013306 010322
1680 013310 013722 001272
1681 013314 022323
1682 013316 010113
1683 013320 062723 000002
1684 013324 005723
1685 013326 010322
1686 013330 013722 001274
1687 013334 022323
1688 013336 010123
1689 013340 012721 050400
1690
1691
1692
1693 013344 005721
1694 013346 006300
1695 013350 016011 001220
1696 013354 006200
1697 013356 012737 000001 001260
1698 013364 012304
1699 013366 010246
1700 013370 010346
1701 013372 033711 001260
1702 013376 001420
1703 013400 010564 001312
1704 013404 010564 002312

```

```

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

```

```

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

```

::++f

```

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

```

```

P3INIT: CLR R0
MOV DEVADR, R1
MOV VECADR, R2
MOV #R3SR0, R3 ;SET UP POINTER TO LINKERS
1$: MOV R3, (R2)+ ;SET UP RECEIVER INTERRUPT VECTOR
MOV RCVLVL, (R2)+
CMP (R3)+, (R3)+ ;ADD 4 TO R3
MOV R1, (R3) ;ADDRESS OF CSR
ADD #2, (R3)+ ;ADDRESS OF RBUF
TST (R3)+
MOV R3, (R2)+ ;SET UP TRANSMITTER INTERRUPT VECTOR
MOV XMTLVL, (R2)+
CMP (R3)+, (R3)+
MOV R1, (R3)+ ;ADDRESS OF CSR
MOV #50400, (R1)+ ;SET UP CSR, TRANSMITTER ONLY
;BIT8 = TRANSMITTER SCAN ENABLE
;BIT12 = STATUS ENABLE
;BIT14 = TRANSMITTER INTERRUPT ENABLE
TST (R1)+
ASL R0 ;UNIT # * 2
MOV SVSW0(R0), (R1) ;SET TCR BITS FOR SELECTED LINES
ASR R0 ;RESET UNIT COUNTER
MOV #1, MARK ;SET UP MARKER
MOV (R3)+, R4 ;SET UP OFFSET TO TABLES
MOV R2, -(6) ;PUSH R2 ON STACK
MOV R3, -(6) ;PUSH R3 ON STACK
2$: BIT MARK, (R1) ;CHECK FOR LINE SELECTED
BEQ 6$
MOV R5, XMTTAB(4) ;SET UP HEADER MESSAGE
MOV R5, RCVTAB(4) ;SET UP RECEIVER TABLE

```

```

1705 013410 013702 001262      MOV      BUFSIZ, R2      ;SET UP COUNTER
1706 013414 012703 016715      MOV      #MSGP3, R3     ;SET UP MESSAGE POINTER
1707 013420 112325      3$:     MOV      (R3)+, (R5)+ ;MOVE MESSAGE INTO BUFFER
1708 013422 001404      BEQ      5$             ;BRANCH IF END OF MESSAGE
1709 013424 005302      DEC      R2             ;COUNT BUFFER SIZE
1710 013426 001374      BNE      3$             ;BRANCH IF MORE
1711 013430 000403      BR       6$             ;BRANCH IF DONE
1712 013432 105025      4$:     CLRB     (R5)+     ;CLEAR REST OF BUFFER
1713 013434 005302      5$:     DEC      R2             ;COUNT BUFFER SIZE
1714 013436 001375      BNE      4$             ;BRANCH IF MORE
1715 013440 010564 003312      6$:     MOV      R5,MAXTAB(4) ;SETUP BFR POINTER TABLE.
1716 013444 005724      TST      (R4)+         ;INC OFFSET TO NEXT LINE
1717 013446 006337 001260      ASL      MARK
1718 013452 103347      BCC      2$
1719 013454 012603      MOV      (6)+,R3        ;POP STACK INTO R3
1720 013456 012602      MOV      (6)+,R2        ;POP STACK INTO R2
1721 013460 022121      CMP      (R1)+, (R1)+   ;ADD 4
1722 013462 005200      INC      R0
1723 013464 020037 001264      CMP      R0, UNITS
1724 013470 001306      BNE      1$
1725 013472 042737 000140 177776 BIC      #140, @#PS     ;LOWER PROCESSOR PRIORITY
1726
1727
1728
1729
1730
1731 013500 012701 001312      FORP3:  MOV      #XMTTAB,R1
1732 013504 012702 000400      MOV      #400,R2
1733 013510 005711      1$:     TST      (R1)         ;CHECK FOR XMTR TABLE CLR
1734 013512 001376      BNE      1$             ;BRANCH IF NOT
1735 013514 062701 000002      ADD      #2,R1          ;GO TO NEXT LINE ENTRY
1736 013520 005302      DEC      R2             ;COUNT LINES
1737 013522 001372      BNE      1$             ;BRANCH IF MORE LINES
1738 013524 013700 001264      MOV      UNITS, R0      ;SET UP UNIT COUNTER
1739 013530 013701 001266      MOV      DEVADR, R1     ;AND DEVICE ADDRESS POINTER
1740 013534 052711 000101      2$:     BIS      #101, (R1)  ;SET RECEIVER ENABLES OF CSR
1741
1742
1743 013540 062701 000010      ADD      #10, R1        ;BIT0 = RECEIVER ENABLE
1744 013544 005300      DEC      R0             ;BIT6 = RCV INTERRUPT ENABLE
1745 013546 001372      BNE      2$             ;UPDATE TO NEXT DJ11
1746 013550 012700 020000      MEMX3:  MOV      #20000,R0 ;COUNT DJ11'S
1747 013554 000241      CLC
1748 013556 005540      1$:     ADC      -(R0)        ;TOP OF 4K BANK OF MEMORY
1749 013560 001376      BNE      1$             ;FAST READ/WRITE TO MEMORY
1750 013562 005700      TST      R0             ;RAPID REPEAT
1751 013564 001374      BNE      1$             ;CHECK FOR LOC 0
1752 013566 000770      BR       MEMX3          ;BRANCH IF MORE MEMORY
1753
1754
1755
1756
1757
1758 013570 004037 014320      R3SR0:  JSR      R0,P3RISR
1759 013574 160012 000000      .WORD   <160012+<0*10>>,<0*40>
1760 013600 004037 014170      X3SR0:  JSR      R0,P3XISR

```

```

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

```

```

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

```

1761	013604	160020	000000
1762	013610	004037	014320
1763	013614	160022	000040
1764	013620	004037	014170
1765	013624	160030	000040
1766	013630	004037	014320
1767	013634	160032	000100
1768	013640	004037	014170
1769	013644	160040	000100
1770	013650	004037	014320
1771	013654	160042	000140
1772	013660	004037	014170
1773	013664	160050	000140
1774	013670	004037	014320
1775	013674	160052	000200
1776	013700	004037	014170
1777	013704	160060	000200
1778	013710	004037	014320
1779	013714	160062	000240
1780	013720	004037	014170
1781	013724	160070	000240
1782	013730	004037	014320
1783	013734	160072	000300
1784	013740	004037	014170
1785	013744	160100	000300
1786	013750	004037	014320
1787	013754	160102	000340
1788	013760	004037	014170
1789	013764	160110	000340
1790	013770	004037	014320
1791	013774	160112	000400
1792	014000	004037	014170
1793	014004	160120	000400
1794	014010	004037	014320
1795	014014	160122	000440
1796	014020	004037	014170
1797	014024	160130	000440
1798	014030	004037	014320
1799	014034	160132	000500
1800	014040	004037	014170
1801	014044	160140	000500
1802	014050	004037	014320
1803	014054	160142	000540
1804	014060	004037	014170
1805	014064	160150	000540
1806	014070	004037	014320
1807	014074	160152	000600
1808	014100	004037	014170
1809	014104	160160	000600
1810	014110	004037	014320
1811	014114	160162	000640
1812	014120	004037	014170
1813	014124	160170	000640
1814	014130	004037	014320
1815	014134	160172	000700
1816	014140	004037	014170

R3SR1:	.WORD	<160020+<0*10>>,<0*40>
	JSR	RO,P3RISR
X3SR1:	.WORD	<160012+<1*10>>,<1*40>
	JSR	RO,P3XISR
R3SR2:	.WORD	<160020+<1*10>>,<1*40>
	JSR	RO,P3RISR
X3SR2:	.WORD	<160012+<2*10>>,<2*40>
	JSR	RO,P3XISR
R3SR3:	.WORD	<160020+<2*10>>,<2*40>
	JSR	RO,P3RISR
X3SR3:	.WORD	<160012+<3*10>>,<3*40>
	JSR	RO,P3XISR
R3SR4:	.WORD	<160020+<3*10>>,<3*40>
	JSR	RO,P3RISR
X3SR4:	.WORD	<160012+<4*10>>,<4*40>
	JSR	RO,P3XISR
R3SR5:	.WORD	<160020+<4*10>>,<4*40>
	JSR	RO,P3RISR
X3SR5:	.WORD	<160012+<5*10>>,<5*40>
	JSR	RO,P3XISR
R3SR6:	.WORD	<160020+<5*10>>,<5*40>
	JSR	RO,P3RISR
X3SR6:	.WORD	<160012+<6*10>>,<6*40>
	JSR	RO,P3XISR
R3SR7:	.WORD	<160020+<6*10>>,<6*40>
	JSR	RO,P3RISR
X3SR7:	.WORD	<160012+<7*10>>,<7*40>
	JSR	RO,P3XISR
R3SR10:	.WORD	<160020+<7*10>>,<7*40>
	JSR	RO,P3RISR
X3SR10:	.WORD	<160012+<10*10>>,<10*40>
	JSR	RO,P3XISR
R3SR11:	.WORD	<160020+<10*10>>,<10*40>
	JSR	RO,P3RISR
X3SR11:	.WORD	<160012+<11*10>>,<11*40>
	JSR	RO,P3XISR
R3SR12:	.WORD	<160020+<11*10>>,<11*40>
	JSR	RO,P3RISR
X3SR12:	.WORD	<160012+<12*10>>,<12*40>
	JSR	RO,P3XISR
R3SR13:	.WORD	<160020+<12*10>>,<12*40>
	JSR	RO,P3RISR
X3SR13:	.WORD	<160012+<13*10>>,<13*40>
	JSR	RO,P3XISR
R3SR14:	.WORD	<160020+<13*10>>,<13*40>
	JSR	RO,P3RISR
X3SR14:	.WORD	<160012+<14*10>>,<14*40>
	JSR	RO,P3XISR
R3SR15:	.WORD	<160020+<14*10>>,<14*40>
	JSR	RO,P3RISR
X3SR15:	.WORD	<160012+<15*10>>,<15*40>
	JSR	RO,P3XISR
R3SR16:	.WORD	<160020+<15*10>>,<15*40>
	JSR	RO,P3RISR
X3SR16:	.WORD	<160012+<16*10>>,<16*40>
	JSR	RO,P3XISR

```

1817 014144 160200 000700
1818 014150 004037 014320
1819 014154 160202 000740
1820 014160 004037 014170
1821 014164 160210 000740
1822
1823
1824
1825
1826
1827 014170
1828 014170 010146
1829 014172 010246
1830 014174 012001
1831 014176 005711
1832 014200 100043
1833 014202 116102 000007
1834 014206 006302
1835 014210 061002
1836 014212 117261 001312 000006
1837 014220 105072 001312
1838 014224 005262 001312
1839 014230 026262 003312 001312
1840 014236 001003
1841 014240 163762 001262 001312
1842 014246 105772 001312
1843 014252 001351
1844 014254 010346
1845 014256 005062 001312
1846 014262 161002
1847 014264 006202
1848 014266 005003
1849 014270 000261
1850 014272 006103
1851 014274 005302
1852 014276 100375
1853 014300 040361 000004
1854 014304 012603
1855 014306 000733
1856 014310
1857 014310 012602
1858 014312 012601
1859 014314 012600
1860 014316 000002
1861
1862
1863
1864
1865
1866 014320
1867 014320 010146
1868 014322 010246
1869 014324 010346
1870 014326 010446
1871 014330 012001
1872 014332 011102

```

```

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

```

```

:*****
:PROG3 TRANSMITTER INTERRUPT SERVICE ROUTINE
:*****

```

P3XISR:

```

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

```

```

:*****
:PROG3 RECEIVER INTERRUPT SERVICE ROUTINE
:*****

```

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
1$:     MOV  (R0)+,R1          ;GET RBUF ADDRESS
        MOV  (R1),R2          ;READ THE DATA

```

1873	014334	100077			BPL	8\$:BRANCH IF NO CHAR PRESENT	
1874	014336	032702	070000		BIT	#70000,R2			:CHECK FOR ERRORS	
1875	014342	001402			BEQ	2\$:BRANCH IF OK	
1876	014344	104002			HLT+2				:RECEIVER ERROR	
1877									:R1=RBUF ADDRESS	
1878									:R2=CONTENTS OF RBUF	
1879									:BIT12=PARITY ERROR	
1880									:BIT13=FRAMING ERROR	
1881									:BIT14=UART OVERRUN	
1882	014346	000771			BR	1\$:SKIP BAD DATA	
1883	014350	010204		2\$:	MOV	R2, R4			:DUP THE RBUF	
1884	014352	105004			CLRB	R4			:CLEAR THE DATA	
1885	014354	000304			SWAB	R4			:LINE # TO LOW BYTE	
1886	014356	106304			ASLB	R4			:LINE # * 2, ALSO CLR CHAR PRESENT	
1887	014360	061004			ADD	(R0),R4			:ADD OFFSET	
1888	014362	136402	005312		BITB	MASK(4),R2			:CHECK CHARACTER LENGTH	
1889	014366	001401			BEQ	3\$:BRANCH IF OK	
1890	014370	104002			HLT+2				:CHARACTER LENGTH ERROR	
1891									:R1=RBUF ADDRESS	
1892									:R2=CONTENTS OF RBUF (DATA)	
1893	014372	005764	002312	3\$:	TST	RCVTAB(4)			:CHECK FOR UNSELECTED LINE	
1894	014376	001002			BNE	4\$:BRANCH IF OK	
1895	014400	104002			HLT+2				:RECEIVED DATA ON UNSELECTED LINE	
1896									:R1 = RBUF ADDRESS	
1897									:R2 = CONTENTS OF RBUF	
1898	014402	000753			BR	1\$:IGNORE THE DATA	
1899	014404	105774	002312	4\$:	TSTB	@RCVTAB(4)			:CHECK FOR DATA BUFFER FULL	
1900	014410	001403			BEQ	5\$:BRANCH IF OK	
1901	014412	104002			HLT+2				:SOFTWARE DATA BUFFER OVERFLOW	
1902									:POSSIBLE TRANSMITTER PROBLEM	
1903									:R1 = RBUF ADDRESS	
1904									:R2 = CONTENTS OF RBUF	
1905										
1906										
1907										
1908										
1909	014414	000137	013216		JMP	PROG3			:RESTART ON THIS TYPE ERROR	
1910										
1911	014420	005764	001312	5\$:	TST	XMTTAB(4)			:CHECK FOR TRANSMITTER ACTIVE	
1912	014424	001414			BEQ	6\$:BRANCH IF INACTIVE	
1913	014426	110274	002312		MOVB	R2, @RCVTAB(4)			:PUT THE DATA IN THE BUFFER	
1914	014432	005264	002312		INC	RCVTAB(4)			:UPDATE POINTER TO NEXT SPACE	
1915	014436	026464	003312	002312	CMP	MAXTAB(4),RCVTAB(4)			:CHECK FOR END OF BUFFER	
1916	014444	001332			BNE	1\$:BRANCH IF NOT	::++F
1917	014446	163764	001262	002312	SUB	BUFSIZ,RCVTAB(4)			:RESET BUFFER POINTER	::++F
1918	014454	000726			BR	1\$				
1919	014456	016464	003312	002312	MOV	MAXTAB(4),RCVTAB(4)			:RESET TABLE POINTER	::++F
1920	014464	163764	001262	002312	SUB	BUFSIZ,RCVTAB(4)			:RESET BUFFER	::++F
1921	014472	016464	002312	001312	MOV	RCVTAB(4),XMTTAB(4)				::++F
1922	014500	110274	002312		MOVB	R2, @RCVTAB(4)				
1923	014504	005264	002312		INC	RCVTAB(4)			:UPDATE POINTER TO NEXT SPACE	
1924	014510	161004			SUB	(R0),R4				
1925	014512	006204			ASR	R4				
1926	014514	005003			CLR	R3				
1927	014516	000261			SEC					
1928	014520	006103		7\$:	ROL	R3				

:NOTE: IF THE ABOVE ERROR WAS DUE TO OVERLOAD, INCREASING THE CONTENTS OF 'BUFSIZ' MAY RECTIFY THE PROBLEM. 'BUFSIZ' MUST BE A MULTIPLE OF 2. INCREASING IT MAY CAUSE THE BUFFERS TO OVERFLOW 4K.

CZDJBG DJ11 EXER & ONLNE
CZDJBG.P11 07-JUN-82 16:32MACY11 30A(1052) 07-JUN-82 16:36 PAGE 39
RECEIVER ISR

SEQ 38

1929	014522	005304			DEC	R4	
1930	014524	100375			BPL	7\$	
1931	014526	050361	000002		BIS	R5,2(R1)	;SET TCR BIT FOR LINE
1932	014532	000677			BR	1\$	
1933	014534			8\$:			
1934	014534	012604			MOV	(6)+,R4	;POP STACK INTO R4
1935	014536	012603			MOV	(6)+,R3	;POP STACK INTO R3
1936	014540	012602			MOV	(6)+,R2	;POP STACK INTO R2
1937	014542	012601			MOV	(6)+,R1	;POP STACK INTO R1
1938	014544	012600			MOV	(6)+,R0	;POP STACK INTO R0
1939	014546	000002			RTI		
1940	014550						
1941	014550	004737	016316		DONE:		
1942	014554	062737	000001	001206	JSR	PC, KBDINT	
1943	014562	005537	001204		ADD	#1,PCNT+2	;ADD 1 TO THE PASS COUNT
1944	014566	000004	016411		ADC	PCNT	;MAKE IT DOUBLE PREC.
1945	014572	032777	002000	164410	TYPE	,MEOP	;END OF PASS INDICATOR
1946	014600	001004			BIT	#SW10,@SWR	;RING THE BELL?
1947	014602	000004	000007		BNE	4\$;NO!
1948	014606	000004	000177		TYPE	,BELL	;RING THE BELL
1949	014612	013700	000042		TYPE	,177	;TYPE A FILLER FOR 11/05
1950	014616	001405			4\$:	MOV	@#42,R0
1951	014620	000005			BEQ	3\$;GET MONITOR ADDRESS
1952		014622			RESET		;IF NONE
1953	014622	004710			SENDAD =		;RESET AND
1954	014624	000240			JSR	7,(0)	;GO TO MONITOR
1955	014626	000240			NOP		;SAVE ROOM
1956	014630	000240			NOP		;FOR
1957	014632	000137	007316		3\$:	NOP	;ACT11
1958					JMP	RESTAR	;RETURN

```

1959
1960
1961
1962
1963
1964
1965
1966
1967
1968 014636 004737 016316
1969 014642 032777 002000 164340
1970 014650 001402
1971 014652 000004 000007
1972 014656 005237 001202
1973 014662 032777 020000 164320
1974 014670 001026
1975 014672 000004 014676
1976 014702 011637 014766
1977 014706 162737 000002 014766
1978 014714 117737 000046 014764
1979 014722 013705 014766
1980 014726 004737 015374
1981 014732 000004 014736
1982 014742 004737 014770
1983 014746 005777 164236
1984 014752 100001
1985 014754 000000
1986 014756 004737 016316
1987 014762 000002
1988
1989 014764 000000
1990 014766 000000
1991
1992 014770 042737 007700 015012
1993 014776 105337 014764
1994 015002 100411
1995 015004 062737 000100 015012
1996 015012 010005
1997 015014 004737 015374
1998 015020 000004 016450
1999 015024 000764
2000 015026 000207

```

```

:      $HLT      ERROR TYPEOUT HANDLER

:THIS ROUTINE PRINTS OUT ERROR MESSAGES STARTING WITH THE
:ADDRESS OF THE 'HLT'. IT ALSO COUNTS THE NUMBER OF ERRORS
:AND HAS THE CAPABILITY OF LOOPING ON ERROR, BELL ON ERROR,
: 'HALT' ON ERROR, AND INHIBIT TYPEOUTS. AN OPTIONAL ARGUMENT
: (HLT+3) WILL BE PLACED IN 'HLTCT$:' FOR ADITIONAL TYPEOUTS.

EMTS:  JSR      PC,      KBDINT
        BIT      #SW10,@SWR      ;BELL ON ERROR?
        BEQ      1$              ;NO - SKIP
        TYPE     ,BELL           ;RING BELL
1$:     INC      ERRORS          ;COUNT THE NUMBER OF ERRORS
        BIT      #SW13,@SWR      ;SKIP TYPEOUT IF SET
        BNE     2$              ;SKIP TYPEOUTS
        TYPE     ,,+2            ;.ASCIZ <15><12>
        MOV     (6),HLTADR        ;PUT ADDRESS OF INSTRUCTION ON STACK
        SUB     #2,HLTADR        ;FUDGE ADDRESS
        MOVB    @HLTADR,HLTCT$    ;GET HLT ARGUMENT
        MOV     HLTADR,TTY        ;TYPE HLTADR IN OCTAL
        JSR     PC,PRINTR        ;TYPE LEADING ZERO'S
        TYPE     ,,+2            ;.ASCIZ " "
2$:     JSR     PC,ERROR$        ;GO TO USER ERROR ROUTINE
        TST     @SWR             ;HALT ON ERROR
        BPL     ,+4              ;SKIP IF CONTINUE
        HALT
        JSR     PC,KBDINT
        RTI                      ;RETURN

HLTCT$: 0
HLTADR: 0
;HLT ARGUMENT
;LAST HLT INSTRUCTION EXECUTED

ERROR$: BIC      #770n,PC
1$:     DFCB    HLTCT$
        EAI     3$
        ADD     #100,2$
2$:     MCV     X0,TTY           ;TYPE REGISTER X IN OCTAL
        JSR     X7,PRINTR
        TYPE    SPACE
        BR     1$
3$:     RTS      PC

```



```

2001
2002
2003
2004 015030 012737 000001 015322 READIN: MOV #1,INHRE
2005 015036 004737 015176 JSR PC, READS ;GO READ TTY UNTIL CR
2006 015042 005037 015322 CLR INHRE
2007 015046 010146 MOV R1,-(6) ;PUSH R1 ON STACK
2008 015050 010246 MOV R2,-(6) ;PUSH R2 ON STACK
2009 015052 010346 MOV R3,-(6) ;PUSH R3 ON STACK
2010 015054 012501 MOV (R5)+, R1
2011 015056 012737 000020 016214 MOV #20,CNT
2012 015064 012702 015324 MOV #INPUT,R2
2013 015070 122712 000120 CMPB #120,(R2) ;CHECK FOR 'P'
2014 015074 001425 BEQ 3$
2015 015076 005011 CLR (R1)
2016 015100 112203 1$: MOVB (R2)+,R3
2017 015102 120327 000015 CMPB R3,#15
2018 015106 001420 BEQ 3$ ;BRANCH WHEN DONE
2019 015110 162703 000060 SUB #60,R3
2020 015114 032703 177770 BIT #177770,R3
2021 015120 001013 BNE 3$ ;BRANCH IF BAD DATA
2022 015122 006311 ASL (R1)
2023 015124 103410 BCS 2$
2024 015126 006311 ASL (R1)
2025 015130 103406 BCS 2$
2026 015132 006311 ASL (R1)
2027 015134 103404 BCS 2$
2028 015136 050311 BIS R3,(R1)
2029 015140 005337 016214 DEC CNT
2030 015144 000755 BR 1$
2031 015146 000244 2$: CLZ
2032 015150 013737 177776 015174 3$: MOV @#PS, PSTEMP ;MAKE SURE Z-BIT IS CLR
2033 015156 012603 MOV (6)+,R3 ;SAVE CONDITION CODES
2034 015160 012602 MOV (6)+,R2 ;POP STACK INTO R3
2035 015162 012601 MOV (6)+,R1 ;POP STACK INTO R2
2036 015164 013737 015174 177776 MOV PSTEMP, @#PS ;POP STACK INTO R1
2037 015172 000205 RTS R5 ;RESTORE CONDITION CODES
2038
2039 015174 000000 PSTEMP: 0 ;TEMPORARY STORAGE FOR PS
2040
2041 015176 010346 READS: MOV R3,-(6) ;SAVE R3
2042 015200 012703 015324 1$: MOV #INPUT,R3 ;GET ADDRESS
2043 015204 022703 015344 2$: CMP #INPUT+20,R3 ;BUFFER FULL?
2044 015210 001415 BEQ 4$ ;YES - TYPE '?'
2045 015212 105737 177560 TSTB @#177560 ;WAIT FOR
2046 015216 100375 BPL .-4 ;A CHARACTER
2047 015220 113713 177562 MOVB @#177562,(3) ;GET CHARACTER
2048 015224 142713 000200 BICB #200,(3) ;GET RID OF JUNK
2049 015230 122713 000177 CMPB #177,(3) ;IS IT A RUBOUT
2050 015234 001403 BEQ 4$ ;SKIP IF NOT
2051 015236 122713 000025 CMPB #25,(3)
2052 015242 001006 BNE 3$
2053 015244 4$:
2054 015244 000004 015250 TYPE .,+2 ;ASCIZ "'?'<15><12>'="
2055 015256 000750 BR 1$ ;ZAP THE BUFFER AND LOOP
2056 015260 111337 016132 3$: MOVB (3),.TYPE ;SET UP FOR TYPING

```

2057	015264	000004	016132		TYPE	..TYPE		:ECHO IT
2058	015270	122723	000015		CMPB	#15,(3)+		:CHECK FOR RETURN
2059	015274	001343			BNE	2\$:LOOP IF NOT RETURN
2060	015276	005737	015322		TST	INHRE		
2061	015302	001401			BEQ	5\$		
2062	015304	000402			BR	6\$		
2063	015306	105063	177777	5\$:	CLRB	-1(3)		:ZAP RETURN (THE 15)
2064	015312	000004	000012	6\$:	TYPE	,12		:TYPE A LINE FEED
2065	015316	012603			MOV	(6)+,R3		:RESTORE R3
2066	015320	000207			RTS	PC		:RETURN
2067								
2068	015322	000000			INHRE:	0		
2069	015324	000020			INPUT:	.BLKW 20		:TTY INPUT AREA
2070					:	\$OCTAL		OCTAL TYPEOUT ROUTINE
2071								
2072								
2073								
2074								
2075								
2076	015364	012737	170101	015532	BITYP\$:	MOV #170101,.PR		:SET BIT FLAG ANS 16. CHARACTER COUNT
2077	015372	000411			BR	.PTIT		:NOW TYPE IT IN BIT FORM
2078	015374	112737	000001	015532	PRINTR:	MOVB #1,.PR		:SET ZERO FILL SWITCH
2079	015402	000402			BR	..+6		:SKIP
2080	015404	005037	015532		PRINTS:	CLR .PR		:SUPPRESS LEADING ZERO'S
2081	015410	112737	177772	015533		MOVB #-6,.PR+1		:SET COUNT
2082	015416	010446			.PTIT:	MOV R4,-(6)		:SAVE R4
2083	015420	012704	015534			MOV #.PR+2,R4		:SET POINTER TO FIRST ASCII CHAR.
2084	015424	105014			CLRB	(4)		:CLEAR FIRST BYTE
2085	015426	000411			BR	.PRF		:ROTATE FIRST BIT
2086	015430	105014			.PRL:	CLRB (4)		:CLEAR BYTE OF CHARACTER
2087	015432	032737	000100	015532		BIT #100,.PR		:BIT TYPING MODE?
2088	015440	001004			BNE	.PRF		:YES - SKIP 2 ROTATES
2089	015442	006105			ROL	TTY		:ROTATE BIT INTO C
2090	015444	106114			ROLB	(4)		:PACK IT
2091	015446	006105			ROL	TTY		:ROTATE BIT INTO C
2092	015450	106114			ROLB	(4)		:PACK IT
2093	015452	006105			.PRF:	ROL TTY		:ROTATE BIT INTO C
2094	015454	106114				ROLB (4)		:PACK IT
2095	015456	105714				TSTB (4)		:IS IT ZERO?
2096	015460	001402			BEQ	..+6		:SKIP INC
2097	015462	105237	015532		INCB	.PR		:SET FILL SWITCH
2098	015466	105737	015532		TSTB	.PR		:CHECK FILL SWITCH
2099	015472	001402			BEQ	..+6		:SKIP BITSET
2100	015474	152724	000060		BISB	#'0,(4)+		:MAKE INTO ASCII CHAR
2101	015500	105237	015533		INCB	.PR+1		:INC COUNT
2102	015504	001351			BNE	.PRL		:REPEAT
2103	015506	022704	015534		CMP	#.PR+2,R4		:EMPTY BUFFER?
2104	015512	001002			BNE	..+6		:SKIP IF NOT
2105	015514	112724	000060		MOVB	#'0,(4)+		:LOAD 1 ZERO
2106	015520	105014			CLRB	(4)		:NULL TERMINATOR
2107	015522	000004	015534		TYPE	..PR+2		:TYPE IT
2108	015526	012604			MOV	(6)+,R4		:RESTORE R4
2109	015530	000207			RTS	PC		:RETURN
2110	015532	000012			.PR:	.BLKW 12		:COUNT, SWITCH, AND OUTPUT BUFFER

: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, SCUMP, DUMP18, OR BITYPE MACRO'S.

```

2111 015556 012777 015704 000126 PDOWN$: MOV #ILLUP,@PUVECS :SET FOR FAST UP
2112 015564 012777 000340 000122 MOV #340,@PUVECS+2 :PRIO:7
2113 015572 010046 MOV R0,-(6) :PUSH R0 ON STACK
2114 015574 010146 MOV R1,-(6) :PUSH R1 ON STACK
2115 015576 010246 MOV R2,-(6) :PUSH R2 ON STACK
2116 015600 010346 MOV R3,-(6) :PUSH R3 ON STACK
2117 015602 010446 MOV R4,-(6) :PUSH R4 ON STACK
2118 015604 010546 MOV R5,-(6) :PUSH R5 ON STACK
2119 015606 010637 015710 MOV SP,.SAVR6 :SAVE SP
2120 015612 012777 015622 000072 MOV #PUPS,@PUVECS :SET UP VECTOR
2121 015620 000000 HALT :WAIT FOR PF
2122
2123 015622 013706 015710 PUPS: MOV .SAVR6,SP :GET SP
2124 015626 005001 CLR R1 :WAIT LOOP FOR THE TTY
2125 015630 005201 1$: INC R1 :WAIT FOR THE INC
2126 015632 001376 BNE 1$ :OF WORD
2127 015634 012605 MOV (6)+,R5 :POP STACK INTO R5
2128 015636 012604 MOV (6)+,R4 :POP STACK INTO R4
2129 015640 012603 MOV (6)+,R3 :POP STACK INTO R3
2130 015642 012602 MOV (6)+,R2 :POP STACK INTO R2
2131 015644 012601 MOV (6)+,R1 :POP STACK INTO R1
2132 015646 012600 MOV (6)+,R0 :POP STACK INTO R0
2133 015650 012737 015556 000024 MOV #PDOWN$,@#24 :SET UP THE POWER DOWN VECTOR
2134 015656 012737 000340 000026 MOV #340,@#26 :PRIO:7
2135 015664 000004 015670 TYPE .,+2 :.ASCIZ <15><12>'POWER'
2136 015700 000137 007316 JMP RESTAR :JMP TO USER ADDRESS
2137
2138 015704 000000 ILLUP: HALT :THE POWER UP SEQUENCE WAS STARTED
2139 015706 000776 BR .-2 : BEFORE THE POWER DOWN WAS COMPLETE
2140
2141 015710 000000 .SAVR6: 0 :PUT THE SP HERE
2142 015712 000024 000026 PUVECS: 24,26 :POWER UP VECTOR
2143
2144
2145 015716 000002 YESRT: RTI :RETURN FROM TRACE TRAP

```

2146
2147
2148
2149
2150
2151
2152 015720 022716 001000
2153 015724 002440
2154 015726 152716 000004
2155 015732 000004 015736
2156 015736 005015 047125 054105
2157 015744 042520 052103 042105
2158 015752 044440 052116 051105
2159 015760 050125 020124 047524
2160 015766 000040
2161 015770 012605
2162 015772 004737 015404
2163 015776 005726
2164 016000 000004 016004
2165 016004 043040 047522 020115
2166 016012 000
2167 016014 016014
2168 016014 011605
2169 016016 004737 015404
2170 016022 000000
2171 016024 000002
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181 016026 010546
2182 016030 017605 000002
2183 016034 032705 177400
2184 016040 001004
2185 016042 019537 016132
2186 016046 012705 016132
2187 016052 105715
2188 016054 001406
2189 016056 112537 177566
2190 016062 105737 177564
2191 016066 100375
2192 016070 000770
2193 016072 017646 000002
2194 016076 062766 000002 000004
2195 016104 022666 000002
2196 016110 001006
2197 016112 062705 000002
2198 016116 042705 000001
2199 016122 010566 000002
2200 016126 012605
2201 016130 000002

```

:*****
:IOT HANDLER - REENTERENT ROUTINE TO INDICATE A FALSE
: INTERRUPT OR TRAP, OR TO TYPE A MESSAGE
:*****

```

```

IOTRAP: CMP      #1000, (SP)      ;CHECK RETURN ADDRESS
        BLT      IOTS          ;BRANCH IF TYPE COMMAND
        SUB      #4, (SP)       ;GET VECTOR ADDRESS
        TYPE,   +2             ;TYPE MESSAGE
        .ASCIZ  <15><12>'UNEXPECTED INTERUPT TO '

        MOV      (SP)+,TTY      ;TYPE (SP)+ IN OCTAL
        JSR      PC,PRINTS     ;AND SUPPRESS LEADING ZERO'S
        TST      (SP)+        ;POP STACK
        TYPE,   +2             ;TYPE MESSAGE
        .ASCIZ  " FROM "

        .EVEN
        MOV      (SP),TTY      ;TYPE (SP) IN OCTAL
        JSR      PC,PRINTS     ;AND SUPPRESS LEADING ZERO'S
        HALT
        RTI                  ;CONTINUE IF DESIRED

```

: \$TYPE MESSAGE TYPEOUT ROUTINE

```

:THIS ROUTINE IS USE TO TYPE ASCII MESSAGES ON THE TTY. THE
:CALL CAN BE IN ONE OF 3 FORMS: 1) 'TYPE ,ADR' - TYPES THE
:MESSAGE STARTING IN LOCATION 'ADR:' 2) 'TYPE ,CHAR' - TYPES
:THE ASCII 'CHAR', AND 3) 'PRINT <<15><12>'MESSAGE'> - TYPES
:THE MESSAGE WHICH IS INLINE ASCII.

```

```

IOTS:  MOV      TTY,-(6)      ;SAVE TTY
        MOV      @2(6),TTY    ;GET ADDRESS TO BE TYPED
        BIT      #177400,TTY  ;IS IT A TYPEN?
        BNE     1$           ;NO
        MOV      TTY,.TYPE    ;GET THE CHARACTER
        MOV      #.TYPE,TTY   ;FUDGE THE ADDRESS
1$:    TSTB     (TTY)         ;TERMINATOR?
        BEQ     2$           ;GET OUT IF SO
        MOVB    (TTY)+,@#177566 ;LOAD AND TYPE THE CHARACTER
        TSTB    @#177564     ;IS THE PRINTER READY
        BPL     -4           ;WAIT UNTIL IT IS
        BR      1$          ;GET THE NEXT CHARACTER
2$:    MOV      @2(6),-(6)    ;GET ADDRESS TO BE TYPED
        ADD     #2,4(6)      ;ADD 2 TO THE ADDRESS
        CMP     (6)+,2(6)    ;IS IT .+2?
        BNE     3$           ;NO
        ADD     #2,TTY       ;ADD 2 TO THE ADDRESS
        BIC     #1,TTY       ;BACK UP TO AN EVEN BYTE
3$:    MOV      TTY,2(6)     ;RESTORE ADDRESS
        MOV      (6)+,TTY    ;RESTORE TTY
        RTI

```

```

2202 016132 000000          .TYPE: 0          ;CHARACTER TYPE LOCATION
2203
2204 016134 022737 000176 001210 CNTLU: CMP      #SWREG,SWR
2205 016142 001023          BNE      1$
2206 016144 000004 016226          TYPE     ,SWREQ
2207 016150 013705 000176          MOV     SWREG,TTY          ;TYPE SWREG IN OCTAL
2208 016154 004737 015374          JSR    PC,PRINTR          ;TYPE LEADING ZERO'S
2209 016160 000004 016216          TYPE     ,NEWIS
2210 016164 004537 015030          JSR    R5,READIN
2211 016170 016364          .WORD   TMP1
2212 016172 001360          BNE     CNTLU
2213 016174 022737 000020 016214          CMP     #20,CNT
2214 016202 001403          BEQ     1$
2215 016204 013777 016364 162776          MOV     TMP1,@SWR
2216 016212 000207          1$:    RTS     PC
2217
2218 016214 000000          CNT:    0
2219
2220 016216 020040 042516 036527 NEWIS:  .ASCIZ  '' NEW= ''
2221 016224 000040
2222 016226 005015 053523 036522 SWREQ:  .ASCIZ  <15><12>'SWR= ''
2223 016234 000040
2224
2225
2226 016236 013746 000006          SUSWRR: MOV     6,-(SP)
2227 016242 013746 000004          MOV     4,-(SP)
2228 016246 012737 016266 000004          MOV     #1$,4
2229 016254 022777 177777 162726          CMP     #-1,@SWR
2230 016262 001402          BEQ     2$
2231 016264 000407          BR      3$
2232 016266 022626          1$:    CMP     (SP)+,(SP)+
2233 016270 012737 000176 001210          2$:    MOV     #SWREG,SWR
2234 016276 012737 000174 001212          MOV     #DISPREG,DISPLAY
2235 016304 012637 000004          3$:    MOV     (SP)+,4
2236 016310 012637 000006          MOV     (SP)+,6
2237 016314 000207          RTS     PC
2238
2239
2240 016316 022737 000176 001210 KBDINT: CMP     #SWREG,SWR
2241 016324 001016          BNE     1$
2242 016326 005037 016364          CLR     TMP1
2243 016332 113737 177562 016364          MOVB   177562,TMP1
2244 016340 142737 000200 016364          BICB   #200,TMP1
2245 016346 122737 000007 016364          CMPB   #7,TMP1
2246 016354 001002          BNE     1$
2247 016356 004737 016134          JSR    PC,CNTLU
2248 016362 000207          1$:    RTS     PC
2249
2250 016364 000000          TMP1:  0
2251
2252
2253 016366 005015 047516 042040 TRNERR: .ASCIZ  <15><12>'NO DATA RECEIVED''
2254 016374 052101 020101 042522
2255 016402 042503 053111 042105
2256 016410 000
2257 016411 015 042412 050117 MEOP:  .ASCIZ  <15><12>'EOP''

```

::++F

2258	016416	000			
2259	016417	043	000040		
2260	016422	051440	046105	041505	MNUM: .ASCIZ '# ''
2261	016430	020124	044514	042516	MSGSEL: .ASCIZ '' SELECT LINE = ''
2262	016436	036440	000040		
2263	016442	005015	177777	000377	RETURN: .ASCIZ <15><12><377><377><377>
2264	016450	020040	000		SPACE: .ASCIZ '' ''
2265	016453	015	177412	055103	MSGMDN: .ASCIZ <15><12><377>'CZDJB-G-0 DJ11 EXER & ONLNE''
2266	016460	045104	026502	026507	
2267	016465	020060	020040	045104	
2268	016474	030461	042440	042530	
2269	016502	020122	020046	047117	
2270	016510	047114	000105		
2271	016514	005015	044506	051522	MSGADR: .ASCIZ <15><12>'FIRST DJ11 ADDRESS: ''
2272	016522	020124	045104	030461	
2273	016530	040440	042104	042522	
2274	016536	051523	020072	000040	
2275	016544	005015	042526	052103	MSGVEC: .ASCIZ <15><12>'VECTOR ADDRESS: ''
2276	016552	051117	040440	042104	
2277	016560	042522	051523	020072	
2278	016566	000040			
2279	016570	005015	047516	020056	MSGNUM: .ASCIZ <15><12>'NO. OF DJ11'S: ''
2280	016576	043117	042040	030512	
2281	016604	023461	035123	020040	
2282	016612	000			
2283	016613	015	050012	047522	MSGPRG: .ASCIZ <15><12>'PROGRAM #: ''
2284	016620	051107	046501	021440	
2285	016626	020072	000040		
2286	016632	005015	047516	042040	MSG01: .ASCIZ <15><12>'NO DJ11'S!''
2287	016640	030512	023461	020523	
2288	016646	000			
2289	016647	015	050012	047522	MSGP2: .ASCIZ <15><12>'PROG2: CONTINUOUS ECHO EXERCISER''<15><12>
2290	016654	031107	020072	041440	
2291	016662	047117	044524	052516	
2292	016670	052517	020123	041505	
2293	016676	047510	042440	042530	
2294	016704	041522	051511	051105	
2295	016712	005015	000		
2296	016715	015	025012	041505	MSGP3: .ASCIZ <15><12>'ECHO TEST''<15><12>
2297	016722	047510	052040	051505	
2298	016730	025124	005015	000	
2299	016735	040	020055	000	MSGDAS: .ASCIZ '' - ''
2300	016741	015	051412	046111	MALARM: .ASCIZ <15><12>'SILO ALARM LEVEL FOR CSR''<15><12>
2301	016746	020117	046101	051101	
2302	016754	020115	042514	042526	
2303	016762	020114	047506	020122	
2304	016770	051503	006522	000012	
2305					
2306	016776	000000			.EVEN
2307		000001			END: 0
					.END

CZDJBG DJ11 EXER & ONLNE
CZDJBG.P11 07-JUN-82 16:32

MACY11 30A(1052) 07-JUN-82 16:36 PAGE 56
CROSS REFERENCE TABLE -- MACRO NAMES

\$SWRDF	1#	514
\$SWRRR	1#	2225
\$TRAP	1#	
\$TYPE	1#	2173
\$URAT	1#	
.SCOP	1#	
.SCOPE	1#	

. ABS. 017000 000

ERRORS DETECTED: 0

CZDJBG,CZDJBG/SOL/CRF/NL:TOC=CZDJBG.MAC,CZDJBG.P11
 RUN-TIME: 5 7 .7 SECONDS
 RUN-TIME RATIO: 38/14=2.7
 CORE USED: 20K (39 PAGES)