

# TABLE OF CONTENTS

3-	1	COPYRIGHT NOTICE
4-	1	KXT11-A2 EDIT HISTORY
5-	1	Equates
6-	1	General DLART Equates
8-	1	General PPI Equates
9-	1	Program-specific Equates
11-	1	MACRO DEFINITIONS
13-	1	RAM Definition
14-	3	TRAPS-Trap-handling routines
14-	4	TRAPS-LTC Trap-Killer
14-	5	TRAPS-BREAK handler
15-	1	RESTART-Introduction
19-	1	RESTART-Entry point
20-	1	RESTART-See if stack exists
20-	19	RESTART-Exit if in IN-ROM state
21-	1	RESTART-Cause determination
22-	1	RESTART-Exits
23-	1	POWERUP-Introduction
24-	1	POWERUP-Turn on LED
24-	22	POWERUP-Test console DLART
25-	1	POWERUP-Test and set up I/O-page RAM
26-	1	POWERUP-Turn off LED
26-	29	POWERUP-Test for "low core"
27-	1	POWERUP-Exit
27-	20	POWERUP-Subroutine to initialize vectors
28-	1	AUTOBAUD-Synchronize with Console
30-	1	macroODT-Introduction
32-	1	macroODT-Save status and print prompt
33-	1	macroODT-Get ODT command
35-	1	macroODT- Go and Proceed
36-	1	macroODT-Register and PS command
37-	1	macroODT-Examine and Deposit
39-	1	macroODT-Get and echo character
40-	1	macroODT-Type ASCII string
41-	1	macroODT-Get octal digits
42-	1	macroODT-OCTSTR--type binary in R0 as ASCII
43-	1	macroODT-Output messages
44-	1	DIAGNOSTICS-for SLU2 and PPI
45-	1	HARDWARE ENTRY POINT
46-	1	DIAGNOSTICS-Continued
47-	1	BOOTS-Description
48-	9	BOOTS-RX Controller Definitions
48-	56	BOOTS-TU58 Definitions and Protocol Equates
48-	114	BOOTS-RT11 Definitions and Equates
49-	1	BOOTS-Program entry point
49-	42	-----> HALT AT PC=172234 INDICATES "Illegal device name"
49-	51	-----> HALT AT PC=172264 INDICATES "Illegal unit number"
49-	58	-----> HALT AT PC=172304 INDICATES "No low memory, can't boot"
49-	92	-----> HALT AT PC=172376 INDICATES "Unexpected timeout during boot"
50-	1	BOOTS-RX01/RX02 Bootstrap
51-	1	BOOTS-Distinguishing type of boot block
51-	23	-----> HALT AT PC=172454 INDICATES "No boot block on volume"
52-	1	BOOTS-TU58 Bootstrap
52-	29	-----> HALT AT PC=172542 INDICATES "TU58 initialization error"
52-	37	-----> HALT AT PC=172562 INDICATES "TU58 block 0 read error"
53-	1	BOOTS-Stand-alone volume bootstrap
53-	24	-----> HALT AT PC=172614 INDICATES "Directory read error"

KXT11-A2 1K FIRMWARE    MACRO V04.00    5-OCT-81 22:56:28  
TABLE OF CONTENTS

53-	36	-----> HALT AT PC=172652 INDICATES "File not found"
54-	1	BOOTS-Load Stand-Alone Program File
54-	8	-----> HALT AT PC=172732 INDICATES "Stand-alone file read error"
54-	12	-----> HALT AT PC=172750 INDICATES "Illegal transfer address"
55-	1	173000G ENTRY POINT
56-	1	BOOTS-Continued
57-	1	BOOTS-RX01/RX02 Read routines
57-	36	-----> HALT AT PC=173070 INDICATES "Floppy drive not ready"
57-	114	-----> HALT AT PC=173262 INDICATES "Floppy read error"
60-	1	BOOTS-TU58 Read routines
61-	27	-----> HALT AT PC=173556 INDICATES "TU58 END packet missing"
61-	37	-----> HALT AT PC=173610 INDICATES "TU58 checksum error"
63-	1	END STATEMENT

```
1                                    .TITLE KXT11-A2 1K FIRMWARE
2                                    .IDENT /V1.00/
3                                    .ENABL LC
4
5                                    ; Place identification number in the last ROM location:
6 000000                            .ASECT
7                                    .=173776
8 173776                            .BYTE 0
9 173777                            .BYTE 1.
```

```
1          .SBTTL COPYRIGHT NOTICE
2          ;
3          ; COPYRIGHT (C) 1980, 1981 BY
4          ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
5          ;
6          ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
7          ; ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
8          ; INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER
9          ; COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
10         ; OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
11         ; TRANSFERRED.
12         ;
13         ; THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE
14         ; AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
15         ; CORPORATION.
16         ;
17         ; DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
18         ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
19         ;
20         ; VERSION V1.00
21         ;
22         ; EDWARD P. LUWISH 29-SEP-81
```

KXT11-A2 1K FIRMWARE    MACRO V04.00    5-OCT-81 22:56:28 PAGE 4  
KXT11-A2 EDIT HISTORY

1		.SBTTL    KXT11-A2 EDIT HISTORY
2		
3		;EDIT HISTORY:
4		;

```
1
2
3
4
5          000001          BIT0   =      1
6          000002          BIT1   =      2
7          000004          BIT2   =      4
8          000010          BIT3   =     10
9          000020          BIT4   =     20
10         000040          BIT5   =     40
11         000100          BIT6   =    100
12         000200          BIT7   =    200
13         000400          BIT8   =    400
14         001000          BIT9   =   1000
15         002000          BIT10  =  2000
16         004000          BIT11  =  4000
17         010000          BIT12  = 10000
18         020000          BIT13  = 20000
19         040000          BIT14  = 40000
20         100000          BIT15  =100000
21
22          ; ASCII CHARACTER EQUATES
23
24         000012          LF      =      12          ;Line feed
25         000015          CR      =      15          ;Carriage return
26         000040          SPACE   =      40          ;Space
27
```

.SBTTL Equates

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

```

.SBTTL General DLART Equates			
		; DLART EQUATES	
177560	RCSR\$1 =	177560	;SLU1 Receive CSR
177562	RBUF\$1 =	177562	;SLU1 Receive buffer
177564	XCSR\$1 =	177564	;SLU1 Xmit CSR
177566	XBUF\$1 =	177566	;SLU1 Xmit buffer
176540	RCSR\$2 =	176540	;SLU2 Receive CSR
176542	RBUF\$2 =	176542	;SLU2 Receive buffer
176544	XCSR\$2 =	176544	;SLU2 Xmit CSR
176546	XBUF\$2 =	176546	;SLU2 Xmit buffer
		; DLART RECEIVE CSR BITS	
004000	RC.ACT =	BIT11	;Receiver active (R/O). Set
			; while character is being
			; received.
000200	RC.DUN =	BIT7	;Receiver done (R/O). A
			; character has been completely
			; received and now resides
			; in RBUF.
000100	RC.IEN =	BIT6	;Receiver int. enable (R/W).
			; when set, enables "keyboard"
			; interrupts, using vector
			; at 60.
		; DLART RECEIVE BUFFER BITS (R/O)	
100000	RB.ERR =	BIT15	;Error. Framing error or
			; overrun has occurred.
040000	RB.OVR =	BIT14	;Overrun error. Character was
			; received before previous one
			; was read.
020000	RB.FRM =	BIT13	;Framing error. No valid stop
			; bit was detected.
004000	RB.BRK =	BIT11	;Break detect. Set when break
			; is detected, reset when next
			; start bit arrives.

```

1      ; DLART TRANSMIT CSR BITS
2
3      000200      XC.RDY =      BIT7      ;Transmitter ready (R/O).
4      ; When set, indicates that the
5      ; last character was completely
6      ; sent and XBUF is ready for
7      ; a new one.
8      000100      XC.IEN =      BIT6      ;Transmit int. enable (R/W).
9      ;when set, enables "console
10     ; printer" interrupts, using
11     ; vector at 64.
12
13     ; Programmable baud rate bits
14
15     000010      PBR0   =      BIT3
16     000020      PBR1   =      BIT4
17     000040      PBR2   =      BIT5
18
19     ; PBR0-2 set baud rates as follows:
20
21     000000      BD.003 =      0          ;Baud rate = 300
22     000010      BD.006 =      PBR0      ;Baud rate = 600
23     000020      BD.012 =      PBR1      ;Baud rate = 1200
24     000030      BD.024 =      PBR1!PBR0 ;Baud rate = 2400
25     000040      BD.048 =      PBR2      ;Baud rate = 4800
26     000050      BD.096 =      PBR2!PBR0 ;Baud rate = 9600
27     000060      BD.192 =      PBR2!PBR1 ;Baud rate = 19200
28     000070      BD.384 =      PBR2!PBR1!PBR0 ;Baud rate = 38400
29
30     000004      XC.MNT =      BIT2      ;Maintenance (R/W). When set,
31     ; creates an internal "loop=
32     ; back" between the transitter
33     ; and receiver. Also dis-
34     ; connects the external
35     ; serial input.
36     000002      XC.PBE =      BIT1      ;Prog. baud rate enable. when
37     ; set, the baud rate is deter-
38     ; mined by bits 3-5 as
39     ; tabulated above. WHEN
40     ; CLEAR, BAUD RATE IS DETER-
41     ; MINED BY VOLTAGES APPLIED
42     ; TO DLART IC PINS.
43     000001      XC.BRK =      BIT0      ;Transmit break (R/W). When
44     ; set, serial output is a
45     ; continuous BREAK.

```



```

1                                     .SBTTL  General PPI Equates
2
3                                     ; PROGRAMMABLE PERIPHERAL INTERFACE (PPI) EQUATES
4
5         176206      PP.CWR =      176206      ;PPI Control Word Register
6         176200      PP.A  =      176200      ;PPI Port A Register
7         176202      PP.B  =      176202      ;PPI Port B Register
8         176204      PP.C  =      176204      ;PPI Port C Register
9
10                                    ; PPI MODE-SETTING BITS
11
12                                    ; KXT11-AA board configuration does not permit all combinations of
13                                    ; the mode bits. Consult the manual before using the PPI.
14
15         000200      PP.MOD =      BIT7        ;This MUST be or'd with other
16                                     ; bits to set mode.
17         000100      PP.MD2 =      BIT6        ;Sets mode 2
18         000040      PP.DMA =      BIT5        ;If bit 6 is low, determine
19                                     ; mode of port A
20                                     ; (hi-mode 1, lo=mode 0)
21         000020      PP.DRA =      BIT4        ;Direction of Port A.
22                                     ; Hi=IN, lo=OUT.
23         000010      PP.CHI =      BIT3        ;Direction of Port C upper half
24                                     ; Hi=IN, lo=OUT.
25         000004      PP.MDB =      BIT2        ;Direction of Port B.
26                                     ; Hi=IN, lo=OUT.
27         000002      PP.DRB =      BIT1        ;Direction of Port B.
28                                     ; Hi=IN, lo=OUT.
29         000001      PP.CLO =      BIT0        ;Direction of port C lower half
30                                     ; Hi=In, lo=OUT.
31
32                                    ; PPI BIT SET/RESET CONTROL BITS
33
34                                    ; When bit 7 is low, writing to the PPI CSR will set or reset
35                                    ; individual bits in Port C, depending on the mode and direction
36                                    ; of the port's bits, and on the combination fo bits you write.
37
38         000016      PP.BI7 =      BIT3!BIT2!BIT1 ;Use ONE
39         000014      PP.BI6 =      BIT3!BIT2      ;of these
40         000012      PP.BI5 =      BIT3!BIT1      ;to select
41         000010      PP.BI4 =      BIT3          ;which bit
42         000006      PP.BI3 =      BIT2!BIT1      ;is desired
43         000004      PP.BI2 =      BIT2          ;to be
44         000002      PP.BI1 =      BIT1          ;SET or
45         000000      PP.BI0 =      0             ;CLEARed
46
47         000001      PP.BIS =      BIT0          ;SET specified bit.
48         000000      PP.BIC =      0             ;CLEAR specified bit.

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34

```

```

.SBTTL Program-specific Equates

; EQUATES USED TO TURN ELD ON AND OFF

MODE = PP.MOD!PP.DRA!PP.CLO ;Port A = Mode 0 IN
;Port B = Mode 0 OUT
;Port C upper nibble = OUT
;Port C lower nibble = IN

LEDOFF = PP.BIS!PP.BI7 ;Set PC7

; EQUATES USED TO SET UP DLARTS

BAUDR$ = BD.024!XC.PBE ;Initial console baud rate to
; be 2400, with prog. baud
; rates enabled.

TUBAUD = BD.384!XC.PBE ;TU58 Baud rate = 38,400

; MEMORY CONFIGURATION EQUATES

RAMBOT = 160010 ;Bottom address of RAM
RAMTOP = 167776 ;Top address of RAM

; SOFTWARE FLAGS AND MASKS

PRI6 = 300 ;PS for priority of 6
PRI7 = 340 ;PS for priority of 7

; USED BY ODT MODULE

RFLAG = BIT7 ;Register flag bit- indicates
; register is being examined

T.BIT = BIT4 ;Trace bit in PSW

```

1			; RESTART TYPE WORD BITS	
2				
3	100000	R.HALT =	BIT15	;HALT or BREAK occurred
4	000200	R.NXM =	BIT7	;Accessed non-existent memory
5	000001	R.STAK =	BIT0	;Double-bus error
6				
7			; BOOT CONTROL WORD BITS	
8				
9	100000	NO.LOW =	BIT15	;No memory found at 000000-
10				; do not boot
11	000200	DEVBIT =	BIT7	;1 = RX01/02 floppy
12				;0 = TU58 cassette
13	000001	DEVNUM =	BIT0	;Unit no. (0 or 1)
14				
15			; DIAGNOSTIC MESSAGES	
16				
17	000100	E.EXT =	100	;SLU2 loopback test failed
18	000010	E.INT =	10	;SLU2 internal loopback failed
19	000001	E.PAR =	1	;Parallel port loopback failed

```
1                                   .SBTTL MACRO DEFINITIONS
2
3                                   ;
4                                   ;       MACRO DEFINITIONS
5                                   ;
6
7                                   ;*
8                                   ;
9                                   ; This macro will insert ABORTS into the code which will halt the
10                                  ; program, exit to IDT with the PC printed on the console, and generate
11                                  ; and entry in the table of contents which describes the error condition.
12                                  ;
13                                  ;-
14
15                   .MACRO   ABORT    TEXT
16                            HALT
17                   .IRP     PC$,\.
18                   .SBTTL   -----> HALT AT PC='PC$ INDICATES "'TEXT"
19                   .ENDR
20                            BR       .-2
21                   .ENDM
```

```

1      ;*
2      ; DELAY A,B,N
3      ;       where A and B are names of registers that are free (both will
4      ;       be clear when through) and N is an integer.
5      ;
6      ; This macro produces a delay whose duration (when running in KTX11-AA
7      ; ROM) is .2399N seconds.
8      ; When N<4, it is more efficient to use the following code:
9      ;
10     ;       CLR      Rn                ;1=      2.44
11     ;       SOB      Rn, .             ;1=      239861.76
12     ;       [SOB     Rn, .             ;1=      239861.76]
13     ;       [SOB     Rn, .             ;1=      239861.76]
14     ;
15     ; The macro generates code like the following:
16     ;
17     ;       MOV      #N,Ra              ;2W      3.66
18     ;n$  CLR      Rb                    ;1W      N*2.44
19     ;       SOB      Rb, .              ;1W      65536N*3.66
20     ;       SOB      Ra,n$              ;1W      N*3.66
21     ;-
22
23     .MACRO  DELAY  A,B,N,?L
24     MOV     #N,A
25 L:  CLR     B
26     SOB     B, .
27     SOB     A, L
28     .ENDM
29

```

```

1                                     .SBTTL RAM Definition
2
3                                     ; SCRATCH RAM AREA
4
5      167776      TRAP4  ==      167776      ;Enables trap-to-4 emulation
6
7      167774      ODTWHY ==      167774      ; when non-zero
8
9
10     167772      O.CNTL ==      167772      ;User-readable copy of R.TYPE.
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

```

167776	TRAP4	==	167776	;Enables trap-to-4 emulation
167774	ODTWHY	==	167774	; when non-zero
167772	O.CNTL	==	167772	;User-readable copy of R.TYPE.
				; Restart cause. See R.TYPE
				; table in RESTART routine.
				;ODT Control word. Set bit 15
				; to disable T-Bit filter, set
				; Bit 7 to disable Priority 7
				; filter.
167770	B.CNTL	==	167770	;Boot control word.
167766	R.PC	==	167766	;where restart saves top of stack
167764	IN.USR	==	167764	;Enables user-caused restart
167762	R.TYPE	==	167762	; and BREAK when non-zero
167760	USERSP	==	167760	;Restart cause. See table in
167756	RPOINT	==	167756	; RESTART routine.
167754	SAVPS	==	167754	;used by ODT to point to the image
167752	SAVPC	==	167752	; of user's R0 in its stack.
167750	ODTFLG	==	167750	;Used by ODT to point to the image
167746	ODTLOC	==	167746	; of user's R0 in its stack.
				;Store halted PS here for ODT
				;Store halted PC here for ODT
				;Used by ODT for internal flags.
				;Used by ODT to point to location
				; currently open.
167744	ODTSTK	==	167744	;Bottom of ODT's stack
167644	\$STACK	==	ODTSTK-100	;Bottom of default user stack.

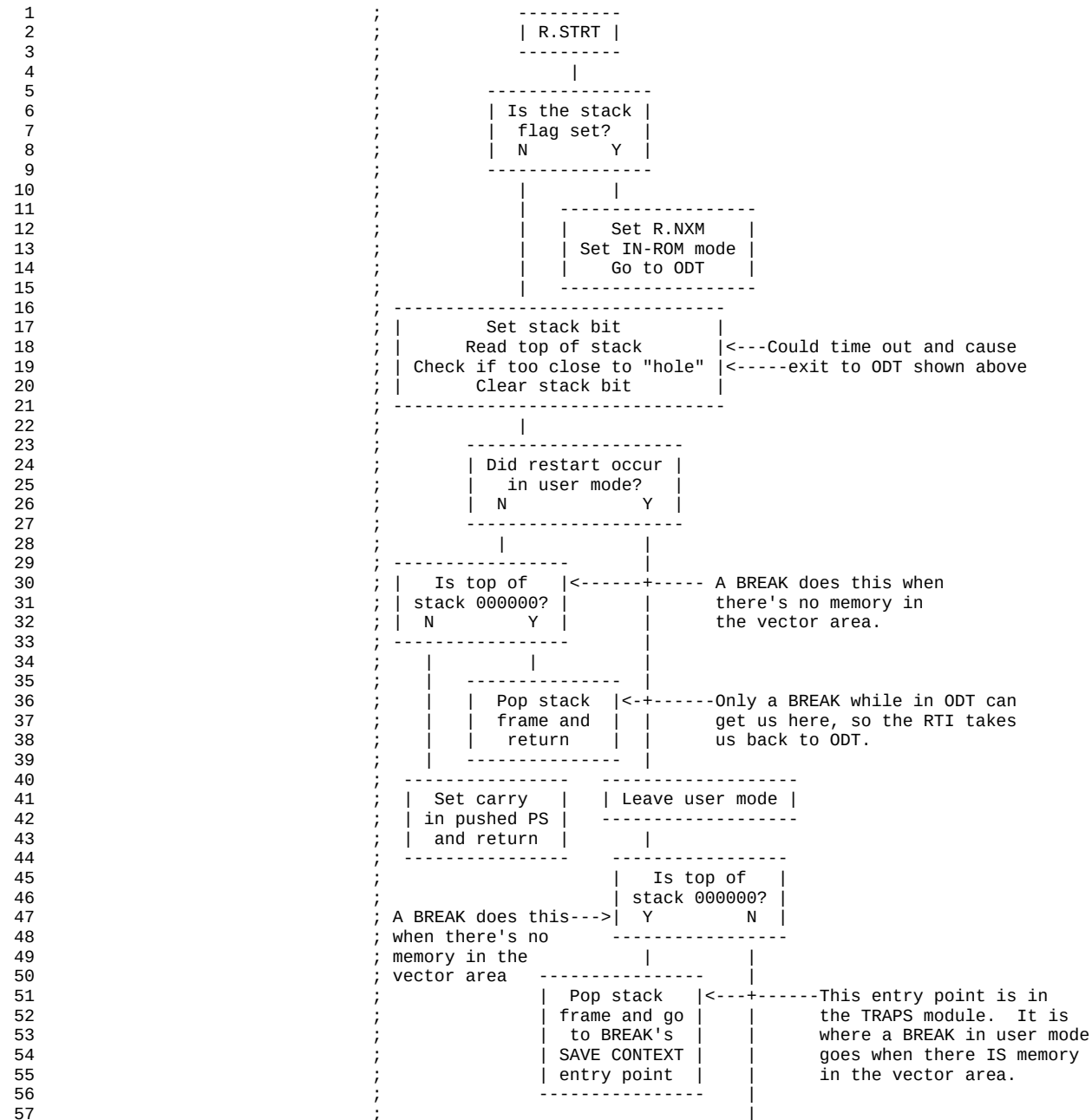
```

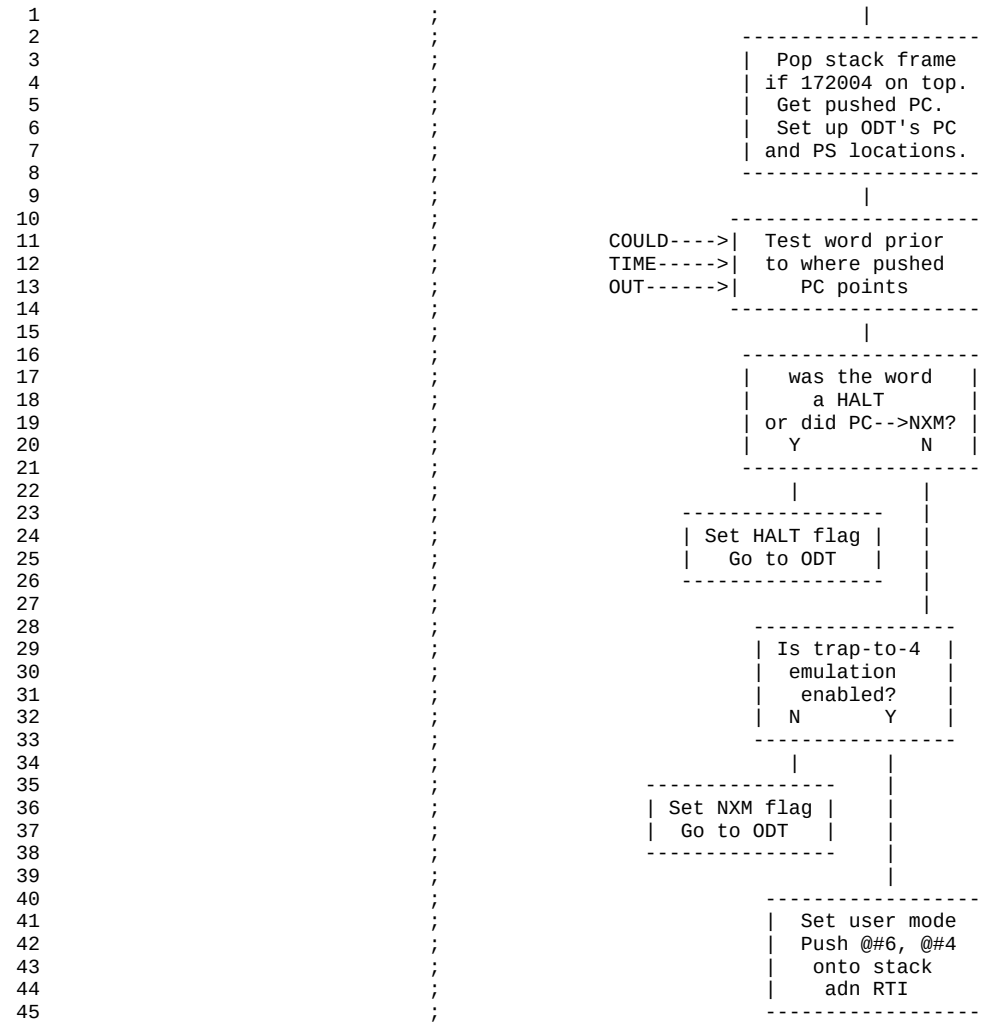
1          170000          . =170000
2
3          .SBTTL TRAPS-Trap-handling routines
4          .SBTTL TRAPS-LTC Trap-Killer
5          .SBTTL TRAPS-BREAK handler
6
7          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;
10         ;;;          BREAK-HANDLING ROUTINE          ;;;
11         ;;;          AND LINE TIME CLOCK INTERRUPT KILLER      ;;;
12         ;;;
13         ;;;
14         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
15         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
16 170000          $$$BRK::
17 170000 005767 177760          TST      IN.USR          ;Are we in user mode?
18 170004 001001          BNE      BRKN00          ;YES-Go to ODT
19
20 170006          $$$LTC::
21 170006 000002          RTI          ;NO-Go back to ROM program.
22          ; BREAKS are ignored by ODT,
23          ; RESTART, POWERUP and the
24          ;   DIAGNOSTICS. The BOOTS
25          ;   can be interrupted, though.
26 170010          BRKN00::
27 170010 012667 177736          MOV      (SP)+,SAVPC          ;Save context
28 170014 012667 177734          MOV      (SP)+,SAVPS          ;for ODT.
29 170020 012767 100000 177734          MOV      #R.HALT,R.TYPE          ;Causes PC to be printed
30          ; upon entry to ODT.
31 170026 005067 177732          CLR      IN.USR          ;Get out of user mode
32 170032 000167 000544          JMP      ODT

```

```
1          .SBTTL RESTART-Introduction
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          RESTART MODULE          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11         ;+
12         ;
13         ;The purpose of the RESTART routine is to restore the FALCON to a
14         ;known state following those exceptions which cause a RESTART hardware
15         ;action. This action consists of stacking the current PSW and program
16         ;counter, then setting the PSW to 340 and jumping to the hardwired
17         ;RESTART location. This location is at the address START+4 where
18         ;START is jumper selectable as 000000, 010000, 020000, 040000, 100000,
19         ;140000, 172000 or 173000 (all in octal). This program is designed
20         ;for a START location of 172000, thus RESTARTs jump to 172004.
21         ;
22         ;There are several different ways in which RESTART performs its
23         ;function, depending on the value in IN.USR, TRAP4, the contents
24         ;of the location the BP points to, and one bit (R.STAK) in R.TYPE.
25         ;
26         ;R.TYPE, the restart type word, is RESTART's output to ODT.
27         ;
28         ;-
29
30         ;+
31         ;
32         ;The goal is to maximize PDP-11 software compatibility and to provide
33         ;useful debugging information to the program developer.
34         ;
35         ;-
```







```
1      ;+
2      ;
3      ;Exception-type word (R.TYEP) is passed to ODT and is RESTARTs "best guess"
4      ;as to why a restart happened:
5      ;
6      ;      Note: A user-readable copy of this word is at ODTWHY.
7      ;
8      ;-----|-----|-----|-----|
9      ; EXIT  BIT   NAME   CAUSE
10     ;-----|-----|-----|-----|
11     ;-----|-----|-----|-----|
12     ; ODT   15    R.HALT  HALT instruction in user code-RESTART POPS STACK.
13     ;      |      |      | Note-BREAK also sets this bit (seet the TRAPS
14     ;      |      |      | module). ODT users this bit for PDP-11 ODT
15     ;      |      |      | compatibility.
16     ;-----|-----|-----|-----|
17     ; ODT   14    |      | Reserved
18     ; OR    13    |      | Reserved
19     ; TRAP  12    |      | Reserved
20     ; TO    11    |      | Reserved
21     ; FOUR  10    |      | Reserved
22     ;      | 9    |      | Reserved
23     ;      | 8    |      | Reserved
24     ;      | 7    | R.NXM  Timeout during user access of non-existant
25     ;      |      |      | memory
26     ;      | 6    |      | Reserved
27     ;      | 5    |      | Reserved
28     ;      | 4    |      | Reserved
29     ;      | 3    |      | Reserved
30     ;      | 2    |      | Reserved
31     ;      | 1    |      | Reserved
32     ;-----|-----|-----|-----|
33     ; ODT   0     R.STAK  Indicates that a timeout was caused by RESTART
34     ;      |      |      | itself accessing non-existant memory. This
35     ;      |      |      | occurs in conjunction with testing for
36     ;      |      |      | validity of the stack pointer.
37     ;      |      |      | In PDP-11 parlance, this is a
38     ;      |      |      | "double-bus error"
39     ;-----|-----|-----|-----|
40     ;
41     ;-
```

```

1          .SBTTL RESTART-Entry point
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          RESTART ENTRY POINT          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11 170036    R.STRT::
12
13          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
14          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
15          ;;;
16          ;;;          IF THE RESTART ROUTINE CAUSE THE RESTART          ;;;
17          ;;;          GO TO ODT AND PRINT "?"          ;;;
18          ;;;
19          ;;;          THIS EXCEPTION CAN BE CAUSED BY RESTART'S          ;;;
20          ;;;          STACK MANIPULATIONS          ;;;
21          ;;;
22          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
23          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
24
25          ; R.TYPE will have been cleared prior to entering
26          ; any ODT command. So, if the stack bit is set, only RESTART
27          ; itself could have caused the trap. Since the stack is always
28          ; valid in in-ROM mode, bad stack means we are in in-USER mode.
29
30          ;State: X=don't care, U=user, R=in-RUN----
31          ;
32 170036    005767 177720          TST      R.TYPE          ;X|Did the stack test fail?
33 170042    001406          BEQ      1$          ;X|No- go to next test
34 170044    052767 000200 177710    BIS      #R.NXM,R.TYPE    ;U|YES- set R.NXM
35                                     ;U| this forces '?' from ODT
36 170052    005067 177706          CLR      IN.USR          ;R|enter in-ROM mode
37 170056    000476          BR       8$          ;R|go to ODT.

```

```

1
2
3
4
5
6
7
8
9
10
11 170060 052767 000001 177674 1$: BIS #R.STAK, R.TYPE ;X|If we timeout, we want RESTART
12 ;X| to know we were diddling SP
13 170066 005716 TST (SP) ;X|see if stack is valid
14 170070 000240 NOP ;X|(in case times out)
15 170072 005766 000004 TST 4(SP) ;X|see if too close to top of
16 170076 000240 NOP ;X| valid memory
17 170100 005067 177656 CLR R.TYPE ;X|stack is OK
18
19 .SBTTL RESTART-Exit if in IN-ROM state
20
21
22
23
24
25
26
27
28
29
30
31 170104 005767 177654 TST IN.USR ;X|Are we in user mode?
32 170110 001007 BNE 3$ ;U|YES-go to next test
33 170112 005716 TST (SP) ;R|NO-see if BREAK brought
34 ;R| us here
35 170114 001002 BNE 2$ ;R|NO-just a restart
36 170116 022626 CMP (SP)+, (SP)+ ;R|YES-Behave like a BREAK that
37 170120 000002 RTI ;R| happened with RAM
38
39 170122 005266 000002 2$: INC 2(SP) ;R|Set carry in pushed PS
40 ;R| UNLESS ALREADY SET
41 170126 000002 RTI ;R|and return to ROM code that
42 ;R| caused timeout

```

```

1                                .SBTTL RESTART-Cause determination
2
3                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5                                ;;;
6                                ;;;          DETERMINE HOW USER CAUSED A RESTART          ;;;
7                                ;;;
8                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11 170130 005067 177630          3$:   CLR      IN.USR              ;U|we were in user mode,
12                                   ;R| but no longer.
13 170134 005716                  TST      (SP)                  ;R|See if BREAK brought
14                                   ;R| us here without low "core".
15 170136 001003                  BNE      4$                    ;R|NO-Just a RESTART
16 170140 022626                  CMP      (SP)+,(SP)+          ;R|YES-behave like a BREAK that
17 170142 000167 177642          JMP      BRKN00                ;R| happened while in user prog.
18
19
20                                ;
21                                ;   If the CPU attempts to fetch an instruction from non-existent
22                                ;   memory, two traps (the first from executing a HALT, the second
23                                ;   from timing out) will occur, the result being that second
24                                ;   trap pushes the restart address and 340 on the stack.
25                                ;   This information is useless and gets popped here.
26                                ;
27 170146 021627 172004          4$:   CMP      (SP),#RESTAR      ;X|Get rid of double stacking
28 170152 001001                  BNE      5$                    ;X|Caused by EXECUTION of NXM
29 170154 022626                  CMP      (SP)+,(SP)+          ;X|
30
31                                ;   Note: Because the contents of the stack is assumed to remain
32                                ;   unchanged following the first instruction below, it is imperative
33                                ;   that interrupts be disabled during the next three instructions.
34                                ;
35 170156 012667 177604          5$:   MOV      (SP)+,R.PC        ;R|Get pushed PC
36 170162 011667 177566          MOV      (SP),SAVPS            ;R|ODT would like
37 170166 014667 177560          MOV      -(SP),SAVPC           ;R|to see these
38
39 170172 162767 000002 177566          SUB      #2,R.PC        ;R|Set pointer to last word fetched
40                                   ;R| before restart occurred
41 170200 005777 177562          TST      @R.PC                 ;R|Is contents of pushed PC - 2
42                                   ;R| a zero (eg a HALT)?
43 170204 000240                  NOP                          ;R|Make sure next instruction
44                                   ;R| won't execute if we time out
45 170206 001005                  BNE      6$                    ;R|NO- it was an NXM
46 170210 052767 100000 177544          BIS      #R.HALT,R.TYPE  ;R|YES- flag a HALT.
47 170216 022626                  CMP      (SP)+,(SP)+          ;R|pop the non-PDP-11 stack frame
48 170220 000415                  BR       8$                    ;R|and go to ODT.
49

```

```

1                               .SBTTL RESTART-Exits
2
3                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5                               ;;;
6                               ;;;          EXIT APPROPRIATELY          ;;;
7                               ;;;
8                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11 170222 005767 177550      6$:   TST      TRAP4              ;R|Trap-to-4 emulation
12 170226 001407              BEQ      7$                    ;R|NO-go to ODT
13 170230 005167 177530      COM      IN.USR                ;U|YES-Set user mode
14 170234 013746 000006      MOV      @#6, -(SP)             ;U|Emulate a
15 170240 013746 000004      MOV      @#4, -(SP)             ;U|trap to
16 170244 000002              RTI                          ;U|four
17
18 170246 052767 000200 177506 7$:   BIS      #R.NXM, R.TYPE ;R|flag NXM error
19 170254 000167 000322      8$:   JMP      ODT              ;R|go to ODT

```

```
1          .SBTTL POWERUP-Introduction
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          POWER-UP MODULE          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10         ;
11         ; This module contains a series of routines which perform
12         ; tests on the on-board RAM and the console DLART. These
13         ; tests are preceded by the lighting of the LED on the
14         ; KXT11-AA board, and followed by its extinguishing. Should
15         ; the LED fail to either light or go out, there may be a
16         ; defect in the board or its configuration.
17
18         ; Following these tests, the on-board RAM is written with the
19         ; default values of certain control words, and, if there is
20         ; memory in the vector region (i.e., near 000000), the BREAK
21         ; and clock vectors are set up. If not, a bit is set in the
22         ; boot control word to disable the bootstraps.
```



```

1          .SBTTL POWERUP-Turn on LED
2
3          ;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;
5          ;;;;;;;;;;;;;;
6          ;;;          TURN ON LED          ;;;
7          ;;;          ;;;
8          ;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;
10         ;;;;;;;;;;;;;;
11
12 170260    PWR$UP::
13 170260 012706 167644      MOV      #$STACK,SP          ;Initialize stack pointer
14
15          ; Because a mode-setting command automatically clears all the internal
16          ; registers in the PPI, and clearing Port C Bit 7 turns on the LED, all
17          ; we have to do is set the mode, which is port A and lo half of C as
18          ; input, ports B and hi half of C as output.
19
20 170264 012737 000221 176206      MOV      #MODE,@#PP.CWR          ;Set proper PPI mode
21
22         .SBTTL POWERUP-Test console DLART
23
24         ;;;;;;;;;;;;;;
25         ;;;;;;;;;;;;;;
26         ;;;
27         ;;;          CHECK THE CONSOLE DLART          ;;;
28         ;;;
29         ;;;;;;;;;;;;;;
30         ;;;;;;;;;;;;;;
31
32 170272 005037 177564      CLR      @#XCSR$1          ;Disable XMIT interrupts,
33          ; BRK XMIT, maint. mode
34          ; Set buad rate to default
35 170276 005737 177562      TST      @#RBUF$1          ;Take out the trash.
36 170302 032737 000300 177560      BIT      #<RC.IEN!RC.DUN>,@#RCSR$1
37          ;Should be clear.
38 170310 001377      BNE      .          ;If not, drop dead.
39 170312 023727 177564 000200      CMP      @#XCSR$1,#XC.RDY          ;Should be set
40 170320 001377      BNE      .          ;If not, rest in peace.

```

```

1                                     .SBTTL POWERUP-Test and set up I/O-page RAM
2
3                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5                                     ;;;;
6                                     ;;;;          I/O PAGE RAM TEST          ;;;;
7                                     ;;;;          AND INITIALIZATION          ;;;;
8                                     ;;;;
9                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
11
12                                    ; Write the location's address into the location and read it back.
13                                    ; Do this for all I/O page RAM locations.
14                                    ; If it fails, enter tight loop.
15
16                                    ; In the process, clear all of this RAM. Note that the default
17                                    ; value of most of the control and flag words is zero.
18
19 170322 012700 160010                MOV    #RAMBOT,R0                ;Lowest address of RAM
20 170326 010010                1$:    MOV    R0,(R0)                ;Write the address
21 170330 020010                CMP    R0,(R0)                ;Read it back
22 170332 001377                2$:    BNE    2$                ;Tight failure loop
23 170334 005020                CLR    (R0)+                ;Clear and go on to next location
24 170336 020027 170000                CMP    R0,#RAMTOP+2        ;Until no more to test.
25 170342 103771                BLO    1$

```

```

1
2
3
4
5
6
7
8
9
10
11 170344 005000          CLR      R0
12 170346 077001          3$:      SOB      R0,3$          ;This leaves a 0 in R0, which
13 170350 077001          4$:      SOB      R0,4$          ; is essential for testing for
14 170352 077001          5$:      SOB      R0,5$          ; the presence of memory at
15 170354 077001          6$:      SOB      R0,6$          ; zero below.
16
17          ; Under no circumstances can R0 be altered until "low core" test below.
18
19
20
21
22
23          TURN OFF LED
24
25
26
27 170356 012737 000017 176206      MOV      #LEDOFF,@#PP.CWR
28
29          .SBTTL POWERUP-Test for "low core"
30
31
32
33
34          TEST FOR MEMORY AT 000000
35
36
37
38
39          ; Read memory at 000000, discard result.  If this fails, exit to
40          ; AUTOBAUD rather than continuing with normal powerup sequence.
41
42 170364 005710          TST      (R0)
43 170366 000240          NOP
44
45          ;This will execute even if
46          ; last instruction times out
47          ;Timed out, don't set vectors
48          ;Didn't time out, so go ahead
49          ;Don't set the NO.LOW flag
50
51 170370 103403          BCS      7$
52 170372 004767 000042      CALL      VECSET
53 170376 000403          BR       8$
54
55 170400 052767 100000 177362 7$:  BIS      #NO.LOW,B.CNTL          ;Did time out.
56
57          ; so let the world know.

```

```

1                               .SBTTL POWERUP-Exit
2
3                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5                               ;;;
6                               ;;;          EXIT FROM POWER-UP SEQUENCE          ;;;
7                               ;;;
8                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11 170406 012767 000300 177340 8$:      MOV      #PRI6,SAVPS          ;If P is typed in response to
12 170414 012767 170424 177330      MOV      #FAKOUT,SAVPC          ; ODT prompt before loading R7,
13 170422 000423                      BR       AUTOBA              ; will force yet more ODT.
14
15 170424                      FAKOUT:
16 170424 005067 177334      CLR      IN.USR          ; BUT IN THE RIGHT MODE!
17 170430 012706 167644      MOV      #$STACK,SP          ; And without running out of
18 170434 000167 000142      JMP      ODT              ; stack, either.
19
20                               .SBTTL POWERUP-Subroutine to initialize vectors
21
22                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
23                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
24                               ;;;
25                               ;;;          INITIALIZE VECTORS          ;;;
26                               ;;;
27                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
28                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
29
30                               ; Note: Thi ssubroutine is also used by the bootstrap module, to
31                               ; restore the vector area in the event that an invalid boot block
32                               ; was read into low memory.
33
34 170440                      VECSET::
35 170440 012737 170000 000140      MOV      #$$$BRK,@#140          ;Set up the BREAK-detect
36 170446 012737 000340 000142      MOV      #PRI7,@#142          ; vector.
37 170454 012737 170006 000100      MOV      #$$$LTC,@#100          ;Set up the line time clock
38 170462 012737 000340 000102      MOV      #PRI7,@#102          ; vector.
39 170470 000207                      RETURN

```

```

1          .SBTTL  AUTOBAUD-Synchronize with Console
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          AUTOBAUD MODULE          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11         ; Description:
12         ;
13         ;     AUTOBAUD allows the FALCON to automatically synchronize its
14         ;     console DLART to the baud rate of the console terminal.
15         ;     On power-up, the user must type a carriage return character.
16         ;     Upon synchronization, AUTOBAUD will proceed to OUT where an '@'
17         ;     character will be displayed on the console.
18         ;
19         ;     Autobaud will loop indefinitely until synchronization is successful.
20         ;
21         ;     The algorithm requires that the console terminal generates a
22         ;     zero (space) for the eighth bit in the carriage return. This
23         ;     will happen if the terminal is capable of sending eight-bit-
24         ;     no-parity or seven-bit-odd-parity characters.
25         ;
26         ; Environment:
27         ;
28         ;     Interrupts must be disabled for the algorithm to execute correctly
29         ;     since time durations are critical and delays due to long
30         ;     service routines may cause DLART overruns, which this routine
31         ;     ignores but cannot tolerate.
32         ;
33
34
35         ; VT103/FALCON configurations leave garbage in the DLART long after the
36         ; powerup sequence has begun. We must delay a bit before clearing garbage
37         ; out of the DLART, otherwise the garbage would arrive after the clear
38         ; (i.e., while polling for input). The "garbage" is an X-ON (<CTRL-q>)
39         ; that the VT-100 hardware sends after its power-up diagnostics have
40         ; completed successfully.
41         ;
42
43 170472     AUTOBA:
44 170472 012737 000032 177564      MOV     #BAUDR$,@#XCSR$1      ;Set 2400 baud
45 170500 005000                  CLR     R0                      ;Delay
46 170502 077001                  SOB     R0,.                    ;          .5
47 170504 077001                  SOB     R0,.                    ;          seconds

```

```

1
2
3 170506 105737 177562      10$:  TSTB  @RBUF$1          ; discard any garbage
4
5 170512 105737 177560      20$:  TSTB  @RCSR$1          ; wait for input
6 170516 100375
7 170520 113700 177562      BPL    20$
8 170524 012701 170550      MOV    @RBUF$1, R0      ; R0 = input character
9 170530 120021              MOV    #INBYTE, R1      ; R1 -> scrambled char table
10 170532 001411            30$:  CMPB  R0, (R1)+      ; in the table?
11 170534 020127 170556      BEQ    HVBAUD          ; yes
12 170540 001373            CMP    R1, #INBYT$      ; end of table reached?
13 170542 005000            BNE    30$              ; not yet
14 170544 077001            CLR    R0              ; uh oh, wait for DLART to clear
15 170546 000757            40$:  SOB   R0, 40$        ; wait for a while
16
17                          ; Table of what you would see if an octal 15 were sent at the following
18                          ; buad rates.
19
20 170550                  INBYTE:
21 170550          200      .BYTE  200              ; 300
22 170551          170      .BYTE  170              ; 600
23 170552          346      .BYTE  346              ; 1200
24 170553          015      .BYTE  15               ; 2400
25 170554          362      .BYTE  362              ; 4800
26 170555          377      .BYTE  377              ; 9600, 19200, 38400
27 170556                  INBYT$:
28
29                          ; We have a match. Set baud rate into DLART.
30
31 170556                  HVBAUD:
32 170556 162701 170551      SUB    #INBYTE+1, R1      ; turn pointer into bit mask
33 170562 006301              ASL    R1
34 170564 006301              ASL    R1
35 170566 005201              INC    R1              ; turn on XC.PBE
36 170570 006301              ASL    R1              ; set the baud rate
37 170572 010137 177564      MOV    R1, @XCSR$1      ; into CSR
38 170576 005000              CLR    R0              ; delay .24 seconds for rest
39 170600 077001              SOB    R0, .            ; of char. at slo baud rates
40
41                          ; Fall into ODT.

```

```

1      .SBTTL macroODT-Introduction
2
3      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4      ;
5      ;
6      ;                               macroODT                               ;
7      ;
8      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9
10     ;
11     ; macroODT is the user interface to the functions contained
12     ; in the KXT11-A2 firmware product. It interprets commands
13     ; entered via the console terminal keyboard (see tables below)
14     ; to permit the user to load a program into memory, execute
15     ; it and debug it.
16
17     ; COMMAND
18     ; 1- Slash (/)
19     ;    a-OPEN MEMORY LOCATION
20     ;    b-OPEN GENERAL REGISTER
21     ;    c-OPEN STATUS REGISTER
22     ; 2- Carriage return (<CR>)
23     ;    a-CHANGE AND CLOSE MEMORY LOCATION OR REGISTER
24     ;    b-CLOSE WITHOUT CHANGE
25     ; 3- Line feed (<LF>)
26     ;    a- CHANGE AND CLOSE MEMORY LOCATION AND OPEN NEXT
27     ;    b- CLOSE MEMORY LOCATION WITHOUT CHANGING AND OPEN NEXT
28     ; 4- Go (G)
29     ; 5- Proceed (P)
30     ; 6- Execute I/O diagnostics (X)
31     ; 7- Execute bootstraps (D)

```

```

1          ;SYNTAX OF COMMANDS LISTED ABOVE, SHOWING CONSOLE BEFORE,
2          ;DURING AND AFTER THE TYPING OF THE COMMAND.
3          ;      Key: n-an octal integer typed by the user, only
4          ;          last 6 digits significant
5          ;          x-a single octal digit
6          ;          u-the digits 0 or 1
7          ;          all other characters are literals
8          ;
9          ;      BEFORE          DURING          AFTER
10         ;
11         ;1a @                @n/                @n/xxxxxx
12         ;1b @                @Rx/               @Rx/xxxxxx
13         ;1c @                @RS/               @RS/xxxxxx
14         ;2a @n/xxxxxx        @n/xxxxxx n<CR>      @
15         ;2a @Rx/xxxxxx        @Rx/xxxxxx n<CR>     @
16         ;2a @RS/xxxxxx        @Rx/xxxxxx n<CR>     @
17         ;2a xxxxxx/xxxxxx xxxxxx/xxxxxx n<CR>    @
18         ;2b @n/xxxxxx        @n/xxxxxx <CR>       @
19         ;2b @Rx/xxxxxx        @Rx/xxxxxx <CR>      @
20         ;2b @RS/xxxxxx        @RS/xxxxxx <CR>      @
21         ;2b xxxxxx/xxxxxx xxxxxx/xxxxxx <CR>     @
22         ;3a @n/xxxxxx        @n/xxxxxx n<LF>       xxxxxx/xxxxxx
23         ;3a xxxxxx/xxxxxx xxxxxx/xxxxxx n<LF>     xxxxxx/xxxxxx
24         ;3b @n/xxxxxx        @n/xxxxxx <LF>        xxxxxx/xxxxxx
25         ;3b xxxxxx/xxxxxx xxxxxx/xxxxxx <LF>     xxxxxx/xxxxxx
26         ;4 @                @nG
27         ;5 @                @P
28         ;6 @                @X                xxxxxx
29         ;
30         ;7 @                @DDu
31         ;7 @                @DXu
32         ;7 @                @DYu
33         ;7 @                @DD<CR>
34         ;7 @                @DX<CR>
35         ;7 @                @DY<CR>

```



```

1          .SBTTL macroODT-Save status and print prompt
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;      SAVE CONTEXT, PRINT MESSAGES AND PROMPT      ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11 170602      ODT::
12 170602      105737      177562      TSTB      @RBUF$1          ;Clear out console garbage
13
14          ; Copy the restart type word into user area
15
16 170606      016767      177150      177160      MOV      R.TYPE,ODTWHY
17
18          ; Protect against stack timeouts, but save user's SP first
19
20 170614      010667      177140      MOV      SP,USERSP          ;SAVE USERS STACK POINTER
21 170620      012706      167744      MOV      #ODTSTK,SP          ;LOAD NEW SP
22
23          ; Save rest of the user program's context
24
25 170624      016716      177130      MOV      USERSP,(SP)          ;RESERVE LOCATION FOR R6
26 170630      010546      MOV      R5,-(SP)          ;SAVE
27 170632      010446      MOV      R4,-(SP)          ; ALL
28 170634      010346      MOV      R3,-(SP)          ; OF
29 170636      010246      MOV      R2,-(SP)          ; USER'S
30 170640      010146      MOV      R1,-(SP)          ; GENERAL
31 170642      010046      MOV      R0,-(SP)          ; REGISTERS
32 170644      010667      177106      MOV      SP,RPOINT          ;POINTER TO R0
33
34          ; Determine whether "?" or PC message is appropriate, and print it
35
36 170650      005767      177106      TST      R.TYPE          ;Did we get a HALT or BREAK?
37 170654      100004      BPL      QODT          ;NO-next question
38          ;YES-PRINT PC
39 170656      016700      177070      MOV      SAVPC,R0          ;GET STOPPED PC
40 170662      004767      000764      CALL      OCTST0          ;TYPE THE PC ON THE TERMINAL
41 170666      QODT:
42 170666      105767      177070      TSTB      R.TYPE          ;SEE IF RESTART OCCURRED
43          ;(NXM ONLY-BIT 7 SET)
44 170672      100003      BPL      KBD$          ;TYPE PROMPT
45
46          ; Here's where the prompt gets printed, with or without leading "?"
47
48 170674      KBDQ:
49 170674      012700      171730      MOV      #MSGQ,R0          ; GET ? ADDRESS
50 170700      000402      BR      PRINT          ;TYPE IN MESSAGE
51 170702      KBD$:
52 170702      012700      171731      MOV      #MSG$,R0          ;GET PROMPT MESSAGE ADDRESS
53
54 170706      005067      177050      PRINT: CLR      R.TYPE          ;So reentry gives no error msg.
55 170712      106427      000300      MTPS      #PRI6          ;Allow BREAKs to happen
56 170716      004767      000620      CALL      PUTSTR          ;TYPE THE PROMPT ALREADY
57 170722      005067      177022      CLR      ODTFLG          ;CLEAR FLAG FOR NEW ENTRY

```

```

1          .SBTTL  macroODT-Get ODT command
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          INTERPRET FIRST CHARACTER OF COMMAND          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11         ; Note: Following CALL GETCHR, the character (7 bit ASCII)
12         ; appears in R2.
13         ; Note: Following CALL GETNUM, if carry is clear, the octal integer
14         ; was followed by a carriage return.
15         ; Note: On exit to LCSET or falling through the GO routine, R0 contains
16         ; the address typed in.
17
18 170726 004767 000556          CALL  GETCHR          ;...INPUT CHARACTERS
19 170732 120227 000104          CMPB   R2,#'D          ;BOOTSTRAPS?
20 170736 001002          BNE    1$          ;NO
21 170740 000167 001220          JMP    BOOTS          ;YES
22
23 170744 120227 000130          1$:  CMPB   R2,#'X          ;DIAGNOSTICS?
24 170750 001002          BNE    2$          ;NO
25 170752 000167 000776          JMP    DIAGNO          ;YES
26
27 170756 120227 000120          2$:  CMPB   R2,#'P          ;PROCEED?
28 170762 001430          BEQ    PCMD          ;YES
29 170764 120227 000122          CMPB   R2,#'R          ;REGISTER?
30 170770 001465          BEQ    RCMD          ;YES
31 170772 120227 000060          CMPB   R2,#'0          ;OCTAL DIGIT?
32 170776 103736          BLO    KBDQ          ;NO,ERROR
33 171000 120227 000070          CMPB   R2,#'8          ;VALID DIGIT?
34 171004 103333          BHIS    KBDQ          ;NO,ERROR
35 171006 005000          CLR     R0          ;ITS A DIGIT
36 171010 004767 000576          CALL  GETNUM          ;GET REST OF THE DIGIT OR CMD
37 171014 103327          BCC     KBDQ          ;CR WAS ISSUED,ERROR
38
39         ; The last character at the end of the number could be a valid command-
40         ; Let's check:
41
42 171016 120227 000057          CMPB   R2,#'/'          ;EXAMINE LOCATION?
43 171022 001511          BEQ    LCSET          ;YES
44 171024 120227 000107          CMPB   R2,#'G          ;GO TO?
45 171030 001321          BNE    KBDQ          ;NO,ERROR

```

1	;	TABLE OF PERMISSABLE STATES			
2	;				
3	;	NO.	STATE	VALID INPUTS	COMMENT
4	;	1-	prompt @	0-7	-----> digit.
5	;			P	-----> proceed.
6	;			R	-----> register designator.
7	;			X	-----> execute diagnostic
8	;			D	-----> boot from device
9	;	2-	@175620	0-7	-----> another digit.
10	;		[input digit]	/	-----> examine loc.
11	;			G	-----> go from loc in.
12	;	3-	@176000/000002	0-7	-----> input new value.
13	;			LF	-----> display next loc.
14	;			CR	-----> close loc go to prompt.
15	;	4-	@200/000023 12	0-7	-----> input more digits.
16	;			LF	-----> save data display next.
17	;			CR	-----> save data go to prompt.
18	;	5-	@R	0-7	-----> register number.
19	;			S	-----> PSW.
20	;	6-	@R5	/	-----> examine.
21	;	7-	@R5/000024	0-7	-----> input new value.
22	;			CR	-----> close location.
23	;	8-	@R5/000024 16	0-7	-----> more digits input
24	;			CR	-----> save value go to prompt
25	;				

```

1          .SBTTL  macroODT- Go and Proceed
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          PROCESS GO AND PROCEED ODT COMMANDS          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11 171032 010067 176714          MOV      R0,SAVPC          ;PUT SUPPLIED PC IN MEMORY LOCATION
12
13          ; Pepare the environment for the Go command
14
15 171036 000005          RESET          ;BUS INITIALIZE
16 171040 005067 176710          CLR      SAVPS          ;CLEAR PSW
17
18          ; Entry point for the Proceed command
19
20          ; First, check for valid stack:
21
22 171044          PCMD:
23
24 171044 016600 000014          MOV      14(SP),R0          ;User's stack pointer
25 171050 005740          TST      -(R0)          ;Where SAVPS will go (see below)
26 171052 000240          NOP          ; (in case of time out)
27 171054 103403          BCS      1$          ;No good. Timed out.
28 171056 005740          TST      -(R0)          ;Where SAVPC will go
29 171060 000240          NOP          ;
30 171062 103004          BCC      2$          ;Sufficient stack.
31
32          ; EITHER Stack no good, so simulate a double bus trap without losing the
33          ; user's context as stored in the ODT stack.
34
35 171064 012767 000201 176702 1$: MOV      #R.STAK!R.NXM,ODTWHY          ;Sneaky! (R.TYPE untouched-
36          ; only the user image of it)
37 171072 000700          BR      KBDQ          ;Error prompt.
38
39          ; OR      Stack is OK, so restore user's context.
40
41 171074 012600          2$: MOV      (SP)+,R0          ;RESTORE
42 171076 012601          MOV      (SP)+,R1          ; ALL
43 171100 012602          MOV      (SP)+,R2          ; OF
44 171102 012603          MOV      (SP)+,R3          ; USER'S
45 171104 012604          MOV      (SP)+,R4          ; GENERAL
46 171106 012605          MOV      (SP)+,R5          ; REGISTERS
47
48 171110 106427 000340          MTPS     #PRI7          ;No BREAKS allowed until out of
49          ; ODT!
50 171114 042716 000001          BIC      #BIT0,(SP)          ;Odd stacks are too odd for the T-11
51 171120 011606          MOV      (SP),SP          ;RESTORE USER SP
52 171122 005167 176636          COM      IN.USR          ;Set user mode
53 171126 016746 176622          MOV      SAVPS,-(SP)          ;RESTORE PC AND PS TO ...
54 171132 016746 176614          MOV      SAVPC,-(SP)          ;...STACK WHERE RTT WILL LOOK
55 171136 000006          RTT          ;RETURN TO USERS PROGRAM
56 171140 000655          HKBDQ: BR      KBDQ          ;HELP IN BR
57 171142 000657          HKBD$: BR      KBD$          ;HELP IN BR

```

```

1          .SBTTL macroODT-Register and PS command
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          PROCESS ODT REGISTER COMMANDS          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11          ; Entry point for Rx and RS commands
12
13 171144          RCMD:
14 171144 052767 000200 176576          BIS      #RFLAG,ODTFLG          ;SET REGISTER FLAG
15 171152 004767 000420                  CALL    ONENUM              ;GET REGISTER NUMBER
16 171156 103246                  BCC      KBDQ              ;A VALID COMMAND DID NOT FOLLOW
17 171160 120227 000123                  CMPB    R2,#'S          ;IS IT THE RS?
18 171164 001412                  BEQ      SWCMD              ;YES,BRANCH
19 171166 120227 000057                  CMPB    R2,#' /          ;EXAMINE?
20 171172 001240                  BNE      KBDQ              ;NO,ERROR
21 171174 020027 000007                  CMP      R0,#7          ;>7?
22 171200 101235                  BHI      KBDQ              ;YES, ERROR
23 171202 001013                  BNE      RCMD1              ;IS IT EXACTLY SEVEN
24 171204 012700 167752                  MOV      #SAVPC,R0          ;YES,GET PC ADDRESS
25 171210 000413                  BR        REGOUT              ;DISPLAY
26
27          ; Status register (PS) selected:
28
29 171212          SWCMD:
30 171212 004767 000272                  CALL    GETCHR              ;WHAT YOU WANT TO DO WITH RS?
31 171216 120227 000057                  CMPB    R2,#' /          ;EXAMINE?
32 171222 001224                  BNE      KBDQ              ;NO,ERROR
33 171224 012700 167754                  MOV      #SAVPS,R0          ;GET ADDRESS WHERE PS IS
34 171230 000403                  BR        REGOUT              ;GO AND DISPLAY
35
36 171232 006300          RCMD1: ASL      R0              ;SHIFT FOR OFFSET IN MEMORY
37 171234 066700 176516                  ADD      RPOINT,R0          ;GET EXACT ADDRESS OF REG.
38 171240 010067 176502          REGOUT: MOV      R0,ODTLOC          ;STORE LOCATION
39 171244 000402                  BR        LOCDSR              ;DISPLAY

```

```

1          .SBTTL  macroODT-Examine and Deposit
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;  PROCESS ODT MEMORY AND REGISTER EXAMINE/DEPOSIT  ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11         ; ODTLOC points to register or memory location
12         ; Following CALL GETNUM, if carry is clear, CR followed digit.
13         ; ODTFLG: If register bit set indicates register is being examined
14
15         ;ENTRY FROM CMD ROUTINE AFTER LOC. VALUE IS GIVEN
16
17 171246 010067 176474      LCSET:  MOV     R0,ODTLOC          ;SAVE NEW LOCATION
18 171252 011000            LOCDSP: MOV     (R0),R0          ;GET DATA
19 171254 000240            NOP                               ;So next inst. does not execute
20                                     ;if we time out.
21 171256 103730            BCS     HKBDQ                    ;Print "?" if we timed out
22 171260 004767 000372      CALL    OCTSTR                  ;PRINT IT
23 171264 112702 000040      MOVB    #SPACE,R2              ;Print a space after the data
24 171270 004767 000226      CALL    PUTCHR
25 171274 004767 000210      CALL    GETCHR                  ;GET NEXT CHARACTER
26 171300 120227 000015      CMPB    R2,#CR                  ;FINISH
27 171304 001716            BEQ     HKBD$                    ;YES,CLOSE LOCATION
28 171306 120227 000060      CMPB    R2,#'0                  ;DEPOSIT?
29 171312 103450            BLO     4$                        ;NO,CHECK LF
30 171314 120227 000070      CMPB    R2,#'8                  ;MAYBE!
31 171320 103307            BHIS    HKBDQ                    ;NO,FORGET IT
32 171322 005000            CLR     R0                        ;YES
33 171324 004767 000262      CALL    GETNUM                  ;GET REST OF NUMBER
34 171330 103006            BCC     1$                        ;CR FOUND, STORE NEW VALUE
35
36 171332 120227 000012      CMPB    R2,#LF                  ;Not CR, must be LF
37 171336 001300            BNE     HKBDQ                    ;Print error message
38 171340 105767 176404      TSTB    ODTFLG                  ;If LF, cannot be register
39 171344 100675            BMI     HKBDQ                    ;(Error exit)
40
41         ;T-BIT FILTER.  The T-BIT can be set from the keyboard via ODT.
42         ;This can either be useful for debugging or disastrous.  So, you can
43         ;do it only if you first set FILT.T in 0.CNTL (BIT 15).
44
45 171346 022767 167754 176372 1$:  CMP     #SAVPS,ODTLOC      ;Are we diddling the PS?
46 171354 001021            BNE     3$                        ;No, we're not.
47 171356 042700 177400      BIC     #^C<377>,R0              ;PS is not a word.
48 171362 005767 176404      TST     0.CNTL                  ;Is BIT 15 (FILT.T) SET?
49 171366 100402            BMI     2$                        ;Yes, the filter's disabled
50 171370 042700 000020      BIC     #T.BIT,R0                ;KILL THE T-BIT
51
52                                     ;2$:                    ;Fall thru to Priority 7
53                                     ; filter

```

```

1          ;Priority-7 filter: unless FILT.7 (BIT 7) in 0.CNTL is set, you cannot
2          ;actually set the PS to priority 7 using ODT from the keyboard. This
3          ;protects the ability to break.
4
5 171374 105767 176372      2$:   TSTB   0.CNTL          ;ODT control word
6 171400 100407             BMI    3$                  ;Do nothing-filter disabled
7 171402 105700             TSTB   R0                   ;Inteded new PS
8 171404 100005             BPL     3$                  ;Do nothing-Priority < 4
9 171406 032700 000100     BIT     #BIT6,R0             ;Check again
10 171412 001402            BEQ     3$                  ;Do nothing-Priority < 6
11 171414 042700 000040     BIC     #BIT5,R0             ;LOWER THE BOOM
12 171420 010077 176322      3$:   MOV     R0,@ODTLOC      ;STORE NEW VALUE
13 171424 120227 000012     CMPB   R2,#LF              ;Go no to next location?
14 171430 001407            BEQ     5$                  ;Sure, why not.
15 171432 000643            BR      HKBD$               ;GO TO PROMPT
16
17 171434 120227 000012      4$:   CMPB   R2,#LF          ;IS A LF ISSUED
18 171440 001237            BNE     HKBDQ               ;NO,ERROR
19 171442 105767 176302     TSTB   ODTFLG              ;IS REGISTER FLAG SET
20 171446 100634            BMI     HKBDQ               ;YES, LF NOT PERMITTED
21 171450 112702 000015      5$:   MOVB   #CR,R2         ;TO LINE UP CURSOR
22 171454 004767 000042     CALL   PUTCHR              ;SEND IT
23 171460 062767 000002 176260  ADD     #2,ODTLOC        ;GET ADDRESS OF NEXT LOC.
24 171466 016700 176254     MOV     ODTLOC,R0           ;GET NEXT ADDRESS VALUE...
25 171472 004767 000160     CALL   OCTSTR              ;...AND PRINT IT
26 171476 112702 000057     MOVB   #'/,R2              ;SEND A SLASH BEFORE...
27 171502 004767 000014     CALL   PUTCHR              ;...SHOWING THE CONTENTS...
28 171506 000661            BR      LOCDSP             ;...OF THE LOCATIONS

```

```

1          .SBTTL  macroODT-Get and echo character
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          CHARACTER INPUT AND ECHO SUBROUTINE          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11         ; Get a character from the console keyboard and echo it back
12         ; exactly as received including parity bits if any.  Return with
13         ; character in R2, eighth bit (and high byte) zero.
14
15 171510  GETCHR:
16 171510  TSTB   @#RCSR$1          ;CHARACTER READY?
17 171514  BPL    GETCHR           ;BRANCH IF NOT AND KEEP TRYING
18 171516  MOVB   @#RBUF$1,R2      ;TRANSFER CHARACTER
19 171522
20 171522  PUTCHR:
21 171522  TSTB   @#XCSR$1          ;PRINTER READY
22 171526  BPL    PUTCHR           ;NO, TRY AGAIN
23 171530  MOVB   R2,@#XBUF$1      ;YES, XMIT CHARACTER
24 171534  BIC    #^C<177>,R2     ;CLEAR PARITY
25 171540  RETURN                  ;CONTINUE

```



```

1          .SBTTL  macroODT-Type ASCII string
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          MESSAGE PRINT SUBROUTINE          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11         ; Print message starting with character pointed to by R0 and
12         ; ending with first character with eighth bit set (this character
13         ; is no printed).
14
15 171542   PUTSTR:
16 171542   112002   MOVB      (R0)+,R2          ;GET ASCII CHAR
17 171544   100413   BMI      DONE              ;IS IT THE END MARK?
18 171546   004767   177750   CALL     PUTCHR      ;NO, PRINT IT
19 171552   000773   BR       PUTSTR            ;MOVE
20
21         ;ENTRY FOR CARRIAGE RETURN
22
23 171554   PUTCLF:
24 171554   112702   000015   MOVB      #CR,R2          ;PRINT CR
25 171560   004767   177736   CALL     PUTCHR      ;FALL THRU AND PRINT LF
26
27         ;ENTRY FOR LF
28
29 171564   112702   000012   PUTLF:   MOVB      #LF,R2          ;PRINT LF
30 171570   004767   177726   CALL     PUTCHR
31 171574   000207   DONE:   RETURN

```

```

1          .SBTTL macroODT-Get octal digits
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          NUMERIC INPUT ROUTINE          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11         ; On exit, R0 contains the binary representation of the number entered
12         ; If the carry bit is clear, a <CR> followed number
13         ; If the carry bit is set, some other character followed the number,
14         ; possibly a command.
15
16 171576      ONENUM:
17 171576 005000      CLR      R0          ;CLEAR ACCUMULATOR
18 171600
19 171600 004767 177704      CALL   GETCHR      ;GET DIGIT OR TERMINATOR
20 171604 120227 000015      CMPB   R2,#CR      ;CLEAR CARRY AND RETURN
21 171610 001412          BEQ    SRET          ;IF <CR> WAS TYPED
22 171612
23 171612 162702 000070      SUB     #'8,R2      ;COVERT TO BINARY...
24 171616 062702 000010      ADD     #'8-'0,R2    ;...AND TEST IF OCTAL OR NOT
25 171622 103007          BCC     NOCT          ;NOT VALID DIGIT.
26 171624 006300          ASL     R0          ;MAKE ROOM FOR NEW DIGIT
27 171626 006300          ASL     R0          ;DITTO
28 171630 006300          ASL     R0          ;DITTO
29 171632 050200          BIS     R2,R0          ;PUT IT IN PLACE
30 171634 000761          BR      NEXNUM        ;GET NEXT
31
32 171636 000241      SRET:   CLC          ;CLEAR CARRY
33 171640 000207          RETURN        ;CONTINUE
34
35 171642 062702 000060      NOCT:  ADD     #'0,R2      ;RESTORE ASCII BECUASE...
36 171646 000261          SEC          ;...POSSIBLE COMMAND
37 171650 000207          RETURN        ;CONTINUE

```

```

1          .SBTTL  macroODT-OCTSTR--type binary in R0 as ASCII
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          NUMERIC OUTPUT ROUTINE          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11         ; Prints, as a 6-digit octal integer, the value of the binary
12         ; number in R0.
13
14 171652 004767 177676      OCTST0: CALL    PUTCLF          ;NEED CRLF AT ODT ENTRY
15 171656                      OCTSTR:
16 171656 010046              MOV     R0,-(SP)                ;SAVE VALUE
17 171660 012746 000006      MOV     #6,-(SP)                ;SAVE VALUE
18 171664 005002              CLR     R2                      ;OUTPUT HOLD
19 171666 006100              5$:    ROL     R0                ;SHIFT MSB INTO LSB
20 171670 006102              ROL     R2                      ;* * * * *
21 171672 062702 000060      ADD     #'0,R2                  ;MAKE A DIGIT
22 171676 004767 177620      CALL    PUTCHR                  ;PUTPUT A CHARACTER
23 171702 005316              DEC     (SP)                    ;COUNT
24 171704 001406              BEQ     10$                      ;DONE
25 171706 005002              CLR     R2                      ;NEXT
26 171710 006100              ROL     R0                      ;GET NEXT DIGIT INTO
27 171712 006102              ROL     R2                      ;R2
28 171714 006100              ROL     R0                      ;FIRST TWO BITS
29 171716 006102              ROL     R2                      ;* * * * *
30 171720 000762              BR      5$                      ;CONTINUE
31 171722 005726              10$:   TST     (SP)+              ;CLEAR COUNT
32 171724 012600              MOV     (SP)+,R0                ;ORIGINAL VALUE
33 171726 000207              RETURN

```

```
1                               .SBTTL  macro0DT-Output messages
2
3                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5                               ;;;
6                               ;;;                                MESSAGES ;;;
7                               ;;;                                ;;;
8                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11                               .NLIST  BEX
12 171730      077      MSGQ:  .ASCII  '?'                               ; ERROR MESSAGE
13 171731      015      012      100 MSG$:  .ASCII  <CR><LF>'@'<200>      ; PROMPT
14                               .EVEN
15                               .LIST   BEX
```

```

1          .SBTTL  DIAGNOSTICS-for SLU2 and PPI
2
3          ;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;
5          ;;;
6          ;;;          DIAGNOSTIC MODULE          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;
10
11         ; Diagnose PPI in mode 0 with loopback connectors installed.
12         ; Diagnose SLU2 internal circuitry (maintenance mode) and
13         ; SLU2 drivers/receivers (with external loopback connector).
14
15         ; List of error bit definitions to return to user.
16
17 171736 000100          .WORD  E.EXT
18 171740 000010          .WORD  E.INT
19 171742
20         ERRBIT:
21
22         ; List of masks to put in XCSR$2. Perform internal loopback
23         ; test first, then external loopback test.
24 171742 000000          .WORD  0
25 171744 000002          .WORD  XC.PBE          ; 300 baud
26 171746 000006          .WORD  XC.PBE ! XC.MNT      ; 300 baud and maintenance
27 171750
28         INITS:
29
30         ; List of pattern bytes to loop around.
31         ; All bits on, alternating bits, all bits off.
32         ; Note: last byte must be 0.
33 171750      377      252      000  PATTERN: .BYTE  377, 252, 0
34          .EVEN
35
36          .ENABL  LSB
37 171754
38 171754 012737 000221 176206  DIAGNO:  MOV  #MODE,@#PP.CWR      ; set proper PPI mode- LED
39 171762 012737 000017 176206      MOV  #LEDOFF,@#PP.CWR      ; must immediately be turned
40          ; off as a consequence
41
42 171770 005000          CLR  R0          ; assume success
43
44         ; Perform parallel port diagnostic
45
46 171772 005001          CLR  R1          ; R1 = loop pattern
47 171774 000405          BR   AROUND2      ; SKIP OVER THE ENTRY POINT

```

```
1                                     .SBTTL HARDWARE ENTRY POINT
2
3      172000                        .=172000
4 172000                                START::
5 172000 000167 176254                JMP      PWR$UP
6 172004                                RESTAR::
7 172004 000167 176026                JMP      R.STRT
```

```

1
2 172010
3 172010 110137 176202
4 172014 123701 176200
5 172020 001402
6 172022 052700 000001
7 172026 077110
8
9
10
11 172030 012702 171742
12 172034 012701 176540
13 172040 016146 000002
14 172044 012704 171750
15 172050 014461 000004
16 172054 001436
17 172056 005742
18 172060 012716 000010
19 172064 012703 171750
20 172070 005005
21 172072 105761 000004
22 172076 100402
23 172100 077504
24 172102 000422
25
26 172104 111361 000006
27 172110 005005
28 172112 105711
29 172114 100402
30 172116 077503
31 172120 000413
32
33 172122 126113 000002
34 172126 001010
35 172130 105723
36 172132 001356
37 172134 005316
38 172136 001744
39 172140 062761 000010 000004
40 172146 000746
41
42 172150 051200
43 172152 005726
44 172154 004767 177472
45 172160 000167 176516
46

.SBTTL DIAGNOSTICS-Continued
AROUN2:
1$: MOVB R1, @PP.B ; send it out port B
CMPB @PP.A, R1 ; check input on port A
BEQ 2$ ; branch if same
BIS #E.PAR, R0 ; else set error flag
2$: SOB R1, 1$ ; loop for all values

; Perform SLU 2 diagnostic

MOV #ERRBIT, R2 ; R2->error flags
MOV #RCSR$2, R1 ; R1 -> SLU2
MOV 2(R1), -(SP) ; ignore garbage, make temp
MOV #INITS, R4 ; R4->initial XCSR value
3$: MOV -(R4), 4(R1) ; init XCSR
BEQ 11$ ; branch if done
TST -(R2) ; R2->next error flag
MOV #8., (SP) ; (SP)=baud rate counter
4$: MOV #PATERN, R3 ; R3->patterns
5$: CLR R5 ; init timeout counter
6$: TSTB 4(R1) ; loop pattern around
BMI 7$ ; branch if ready
SOB R5, 6$ ; else bump timeout counter
BR 10$ ; branch if timeout

7$: MOVB (R3), 6(R1)
CLR R5 ; initialize timeout counter
8$: TSTB (R1)
BMI 9$ ; branch if ready
SOB R5, 8$ ; else bump timeout counter
BR 10$ ; branch if timeout

9$: CMPB 2(R1), (R3) ; come back OK?
BNE 10$ ; no, set error bit & exit
TSTB (R3)+ ; done all bit patterns?
BNE 5$ ; no
DEC (SP) ; yes, done all bauds?
BEQ 3$ ; yes
ADD #10, 4(R1) ; no, to next baud rate
BR 4$

10$: BIS (R2), R0 ; set error bit
11$: TST (SP)+ ; rid of temp
CALL OCTST0 ; print error flags
JMP KBD$ ; and just get out.
.DSABL LSB

```

```

1                               .SBTTL BOOTS-Description
2
3                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5                               ;;;;
6                               ;;;;          BOOTSTRAP MODULE          ;;;;
7                               ;;;;
8                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
11      000000                      .REPT    0
12
13                              ;This is a short bootstrap program designed to handle floppy disks or TU58
14                              ;tape cassettes in either our standard bootable format or in the stand-alone
15                              ;volume format (RT-11 ".SAV"-structured files).
16                              ;
17                              ;The bootstrap sequence is as follows:
18                              ;
19                              ;1. Since entry is affected by typing D in response to ODT prompt, get next
20                              ;character (D, X or Y). Get optional device number next (default is 0).
21                              ;
22                              ;2. If floppy boot is selected:
23                              ;
24                              ;    a. Attempt to read 512 bytes from specified unit of the floppy
25                              ;       disk, starting from logical block zero, into memory locations
26                              ;       starting at 0 at the density of the medium present in the
27                              ;       drive at the time.
28                              ;
29                              ;    b. If the drive is not ready or does not contain a bootable
30                              ;       medium, go back to ODT.
31                              ;
32                              ;3. If TU58 boot is selected, read the first block from the selected
33                              ;drive into locations starting at 0.
34                              ;
35                              ;4. If the first byte read into RAM is 240 octal, jump to it. If the
36                              ;first byte is 260 octal, execute the stand-alone volume loader,
37                              ;using the selected device as input.
38
39                              .ENDR

```



```

1          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
2          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
3          ;;;
4          ;;;          EQUATES USED ONLY BY BOOTSTRAPS          ;;;
5          ;;;
6          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
7          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
8
9          .SBTTL BOOTS-RX Controller Definitions
10
11          ; RX01/RX02 (RXV11,RXV21) Register Definitions
12
13          177170      RXCS=   177170                      ;Control and Status
14          177172      RXDB=   RXCS+2                      ;Data Buffer
15
16          ; RX Control and Status Bits
17
18          100000      RX$$ER= 100000                      ;Error
19          040000      RX$$IN= 040000                      ;Initialize controller
20          030000      RX$$XA= 030000                      ;Extended address bits
21          004000      RX$$02= 004000                      ;1 if RX02; 0 if RX01
22          003000      RX$$XX= 003000                      ;Unused bits
23          000400      RX$$DE= 000400                      ;Density (1=double,0=single)
24          000200      RX$$TR= 000200                      ;Transfer function
25          000100      RX$$IE= 000100                      ;Interrupt enable
26          000040      RX$$DN= 000040                      ;Done
27          000020      RX$$UN= 000020                      ;Unit select
28          000016      RX$$FN= 000016                      ;Function select
29          000001      RX$$G0= 000001                      ;G0
30
31          ; RX Function Codes (in RX$$FN) with G0 bit present
32
33          000001      RX$FIL= 0*2+RX$$G0                  ;Fill buffer
34          000003      RX$EMP= 1*2+RX$$G0                  ;Empty buffer
35          000005      RX$WRT= 2*2+RX$$G0                  ;Write sector
36          000007      RX$RED= 3*2+RX$$G0                  ;Read sector
37          000011      RX$STD= 4*2+RX$$G0                  ;Set media density
38          000013      RX$RST= 5*2+RX$$G0                  ;Read status
39          000015      RX$WDD= 6*2+RX$$G0                  ;Write sector with deleted data
40          000017      RX$REC= 7*2+RX$$G0                  ;Read error code
41
42          ; RX Error Codes
43
44          000400      RXE$UN= 000400                      ;Unit selected
45          000200      RXE$DR= 000200                      ;Drive ready
46          000100      RXE$DD= 000100                      ;Deleted data
47          000040      RXE$DN= 000040                      ;Drive density
48          000020      RXE$DE= 000020                      ;Density error
49          000004      RXE$ID= 000004                      ;Initialize done
50          000001      RXE$CR= 000001                      ;CRC error
51
52          ; Miscellaneous Definitions
53
54          000010      RETRY= 8.                            ;Number of retries
55
56          .SBTTL BOOTS-TU58 Definitions and Protocol Equates
57

```

```

58                                     ;Absolute address definitions
59
60      000002      FILNAM = 000002      ;Address of RAD50 filename for
61                                     ; stand-alone program loading
62      001000      DIRBUF = 001000      ;Start of 512. word buffer used
63                                     ; for RT-11 directory operations
64                                     ; in stand-alone loading
65
66      ; TU58 Address definitions
67
68      176540      TI$CSR = RCSR$2      ;DL recevier control and status
69      176542      TI$BFR = RBUF$2      ;DL receiver data buffer
70      176544      TO$CSR = XCSR$2      ;DL transmitter control and status
71      176546      TO$BFR = XBUF$2      ;DL transmitter data buffer
72
73
74      ; TU58 Radial Serial Protocol codes
75
76      ; Flag Byte Definitions:
77
78      000001      R$$DAT = ^B<00001>      ;Data message flag
79      000002      R$$CTL = ^B<00010>      ;Control message flag
80      000004      R$$INT = ^B<00100>      ;Initialize flag
81      000020      R$$CON = ^B<10000>      ;Continue flag
82      000023      R$$XOF = ^B<10011>      ;XOFF
83
84      ; Control packet operation codes:
85
86      000000      R$NOP = 0.      ;No-operation
87      000001      R$INIT = 1.      ;Initialize
88      000002      R$READ = 2.      ;Read operation
89      000003      R$WRIT = 3.      ;Write operation
90      000004      R$COMP = 4.      ;Compare (NOP on TU58)
91      000005      R$POSI = 5.      ;Position operation
92      000006      R$ABRT = 6.      ;Abort (NOP on TU58)
93      000007      R$DIAG = 7.      ;Diagnose
94      000010      R$GETS = 8.      ;Get status
95      000011      R$SETS = 9.      ;Set status (NOP on TU58)
96      000012      R$GETC = 10.      ;Get characteristics
97      000013      R$SETC = 11.      ;Set characteristics (NOP on TU58)
98      000100      R$END = ^B<01000000>      ;*END message
99
100     ; END package success codes:
101
102     000000      S$NORM = 0.      ;Normal success
103     000001      S$RETR = 1.      ;Sucess but with retries
104     177776      S$PART = -2.      ;Partial operation (end of medium)
105     177770      S$UNIT = -8.      ;Invalid unit number
106     177767      S$CART = -9.      ;No cartridge
107     177765      S$WPRT = -11.      ;Cartridge write protected
108     177757      S$DCHK = -17.      ;Data check error
109     177740      S$SEEK = -32.      ;Seek error (block not found)
110     177737      S$MOTR = -33.      ;Motor stopped
111     177730      S$OPCD = -40.      ;Invalid operation code
112     177711      S$RECN = -55.      ;Invalid record number
113
114                                     .SBTTL BOOTS-RT11 Definitions and Equates

```

```

115
116           ; RT-11 Directory Structure Definitions
117
118           001000      SEGALO = DIRBUF           ;Number of segments allocated
119           001002      NXTSEG = DIRBUF+2         ;Number of next logical segment
120           001004      HGHSEG = DIRBUF+4         ;Highest segment in use
121           001006      XTRBYT = DIRBUF+6         ;Number of extra bytes per entry
122           001010      STRBLK = DIRBUF+10        ;Starting blocks for files
123                                           ; in this segment
124           000016      ENTSIZ = 7*2              ;Size of a directory entry
125           000010      D.FLEN = 10               ;Offset of file length in entry
126           000400      IENTAS$ = 000400          ;Flag for tentative file entry
127           001000      EMPTY$ = 001000           ;Flag for empty area entry
128           002000      PERMF$ = 002000           ;Flag for permanent file
129           004000      ENDSG$ = 004000           ;Flag for end of segment
130
131           ; RT-11 System Communications Area Definitions
132
133           000040      RT$STA = 000040            ;Start address for program
134           000042      RT$ISP = 000042            ;Initial stack pointer
135           000044      RT$JSW = 000044            ;Job status word
136           000046      RT$USR = 000046            ;USR load address
137           000050      RT$HGH = 000050            ;Job high memory limit
138           000052      RT$ENT = 000052            ;(Byte) ENT error code
139           000053      RT$UER = 000053            ;(Byte) User error code
140           000054      RT$RMN = 000054            ;Base address of resident monitor
141           000056      RT$FCH = 000056            ;(Byte) Console fill character
142           000057      RT$FCT = 000057            ;(Byte) Console fill count

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25 172164
26 172164 012737 000072 176544
27
28
29
30 172172
31 172172 010667 175566
32
33 172176 005004
34 172200 004767 177304
35 172204 120227 000104
36 172210 001412
37 172212 012704 000200
38 172216 020227 000130
39 172222 001405
40 172224 020227 000131
41 172230 001402
42 172232
43
44 172236 004767 177246
45 172242 022702 000015
46 172246 001410
47 172250 162702 000060
48 172254 001405
49 172256 005302
50 172260 001402
51 172262
52
53 172266 005204
54 172270 110467 175474
55 172274 005767 175470
56 172300 100002
57

```

```

58 172302                                ABORT    <No low memory, can't boot>
59
60                                ; Before proceeding, we set up the bus timeout trap vector, enable
61                                ; trap to 4 emulation and reset the bus. We do a delay (see
62                                ; explanation below) and set up the stack so the sand-alone booter and
63                                ; device primary bootstraps can get the information they need passed
64                                ; to them in R0 and R1 (see CHK240, below).
65
66 172306 012737 172370 000004 4$:    MOV     #BADBOT,@#4          ;If we time out, we want to re-
67 172314 012737 000300 000006        MOV     #PRI6,@#6          ;initialize everything.
68 172322 000005                      RESET          ;for now, init. the bus.
69
70                                ; Note: the previous instruction also screws up some devices
71                                ; which perform a long initialization sequence, such as RX02's,
72                                ; which do an automatic boot from drive 0. The long delay below
73                                ; is necessary in order to assure drive 1 is ready if a boot
74                                ; is desired from it.
75
76 172324                                DELAY    R0,R1,9.          ;Delay 2 seconds
77 172336 012706 167644                MOV     #$STACK,SP        ;Initialize the stack.
78 172342 010667 175430                MOV     SP,TRAP4          ;Set up Trap-to-4 emulation
79                                ;by making TRAP4 non-zero
80 172346 012716 037776                MOV     #37776,(SP)        ;Some boots need a memory-top
81                                ; address here, so 8K will do
82 172352 010402                      MOV     R4,R2              ;Boot control word here
83 172354 042702 177776                BIC     #^C<DEVNUM>,R2    ;Want only unit no. in R2
84 172360 010246                      MOV     R2,-(SP)          ;And we'll save it too.
85
86 172362 105704                      TSTB     R4                ;Bit 7 set for RX01/RX01
87 172364 100405                      BMI     RXBOOT            ;Go to floppy boot
88 172366 000436                      BR       TUBOOT            ;Go to TU58 boot
89
90 172370                                BADBOT:
91 172370 012706 167644                MOV     #$STACK,SP        ;Restore the stack
92 172374                                ABORT    <Unexpected timeout during boot>

```

```

1                               .SBTTL BOOTS-RX01/RX02 Bootstrap
2
3                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5                               ;;;;                                ;;;;
6                               ;;;;      FLOPPY BOOTSTRAP      ;;;;
7                               ;;;;                                ;;;;
8                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9                               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10                              ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
11                              ; This routine will bootstrap either floppy drive, at the density of the
12                              ; media mounted in that drive.
13
14 172400  RXBOOT:
15 172400 012746 177170      MOV    #RXCS, -(SP)          ;need floppy CSR for CHK240
16 172404 005737 177170      TST    @#RXCS                ;if no there, time out via 4
17 172410 000240              NOP                          ;to ST173 and reset the world
18 172412 012701 001000      MOV    #512., R1             ;Byte count
19 172416 005000              CLR    R0                    ;Starting block number
20 172420 005004              CLR    R4                    ;RAM buffer address = 000000
21 172422 004767 000402      CALL   DREAD                 ;LOAD IT ALL IN
  
```

```

1                                     .SBTTL BOOTS-Distinguishing type of boot block
2
3                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5                                     ;;;
6                                     ;;;      DISTINGUISH STANDARD FROM STAND-ALONE FROM      ;;;
7                                     ;;;      NON-BOOTABLE VOLUMES.                      ;;;
8                                     ;;;
9                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10                                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
11
12                                    ; The CHK240 routine will repeat powerup sequence if location 0 does not
13                                    ; contain a valid secondary bootstrap (i.e., does not have a 240 or 260
14                                    ; in it). It starts execution of the booted program if there's a 240,
15                                    ; and goes to the stand-alone program loader if there's a 260.
16
17 172426                                CHK240:
18 172426 022737 000240 000000          CMP      #240,@#0                ;Did we read a valid bootstrap?
19 172434 001410                        BEQ      1$
20 172436 022737 000260 000000          CMP      #260,@#0
21 172444 001447                        BEQ      STANDB                ;Stand-alone volume start with 260
22 172446 004767 175766                CALL     VECSET                ;Restore wiped-out vectors
23 172452                        ABORT    <No boot block on volume>
24
25 172456 012601                        1$:    MOV     (SP)+,R1          ;Unit CSR address
26 172460 012600                        MOV     (SP)+,R0          ;Unit number
27 172462 005007                        CLR      PC                ;Standard secondary boots

```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
```

```

.SBTTL BOOTS-TU58 Bootstrap

////////////////////////////////////
////////////////////////////////////
TU58 TAPE CASSETTE BOOTSTRAP
////////////////////////////////////
////////////////////////////////////

.ENABL  LSB
TUBOOT:
MOV     #TI$CSR, -(SP)          ;CHK240 wants TU58 CSR
MOV     #T0$CSR,R1             ;R1 -> output CSR for TU58 serial line
CLR     R3                     ;Set R3 = 0 (Two NULLs)
INC     @R1                    ;Start transmitting BREAK to TU58
CALL    CH8OUT                 ;Send eight NULLS
1$:     TSTB    @R1             ;Is transmitter ready again yet?
        BPL     1$             ;If PL no - wait
        BIC     #XC.BRK,@R1    ;Else stop sending BREAK now
        MOV     (PC)+,R3       ;Get two INIT commands for TU58
        .BYTE   R$$INT,R$$INT
        CALL    @R5            ;And transmit them
        TST     -(R1)          ;Dump any garbage char in TI$BUF
2$:     TSTB    @#TI$CSR       ;Is character available from TU58?
        BPL     2$             ;If PL, no - wait in loop
        CMPB    @R1,#R$$CON    ;If so, was it a CONTINUE flag?
        BEQ     3$             ;If EQ, yes- go ahead
        ABORT    <TU58 initialization error>

; TU58 is now initialized. Prepare to read block #0.
3$:     CLR     R0              ;Block number = 0
        MOV     #512.,R1       ;Byte count = one block
        CALL    READZU         ;Attempt to read the block
        BPL     CHK240         ;If PL, read was successful
        ABORT    <TU58 block 0 read error>
.DSABL  LSB
```



```

1          .SBTTL BOOTS-Stand-alone volume bootstrap
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          STAND-ALONE-VOLUME BOOTSTRAP          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11         ; This routine loads stand-alone programs (assumed to be in RT-11 .SAV
12         ; file format) from an RT-11 file structured TU58 cartridge. IT is
13         ; invoked if the first word in block 0 of the cartridge is 260.
14
15 172564      STANDB:
16 172564      012700      000001      MOV      #1,R0          ;Set directory segment #1
17 172570      006300      1$:      ASL      R0          ;Two blocks per segment
18 172572      022020      CMP      (R0)+,(R0)+      ;Add 4 to R0, as directory starts
19                                     ; in block#6
20 172574      012701      002000      MOV      #1024.,R1      ;Prepare to read two blocks
21 172600      012704      001000      MOV      #DIRBUF,R4      ;into the directory buffer
22 172604      004767      000162      CALL     READU          ;read the segment
23 172610      100002      BPL      2$          ;If PL, read was successful
24 172612      ABORT      <Directory read error>
25 172616      012704      001010      2$:      MOV      #STRBLK,R4      ;Else prepare to pick up
26                                     ; starting block
27 172622      012400      MOV      (R4)+,R0      ;R0 = starting block for files
28 172624      010403      3$:      MOV      R4,R3      ;Save pointer to current entry
29 172626      032724      002000      BIT      #PERMF$, (R4)+      ;Is this a permanent file?
30 172632      001010      BNE      4$          ;If bit set, yes - check if it matches
31 172634      022744      004000      CMP      #ENDSG$, -(R4)      ;Else is this the end-of-segment
32                                     ; marker
33 172640      001015      BNE      5$          ;If NE, no - go skip this entry
34 172642      013700      001002      MOV      @#NXTSEG,R0      ;Else get number of next segment
35 172646      001350      BNE      1$          ;If NE, there is one - go read it
36 172650      ABORT      <File not found>
37
38 172654      012705      000002      4$:      MOV      #FILNAM,R5      ;Point to RAD50 name of desired file
39 172660      022425      CMP      (R4)+,(R5)+      ;Check file name, first word
40 172662      001004      BNE      5$          ;if NE not desired file
41 172664      022425      CMP      (R4)+,(R5)+      ;...Check second word of filename
42 172666      001002      BNE      5$          ;if NE not desired one
43 172670      022425      CMP      (R4)+,(R5)+      ;...Finally, check extension
44 172672      001410      BEQ      LOAD          ;If EQ, got it - go load this
45                                     ; one into memory
46 172674      010304      5$:      MOV      R3,R4      ;Get entry pointer back
47 172676      062704      ADD      #D.FLEN,R4      ;Advance to file size of entry
48 172702      062400      ADD      (R4)+,R0      ;Update current file base
49 172704      022424      CMP      (R4)+,(R4)+      ;And skip to next file entry
50 172706      063704      001006      ADD      @#XTRBYT,R4      ;Plus any extra bytes in each entry
51 172712      000744      BR       3$          ;Continue file search

```

```

1                                     .SBTTL BOOTS-Load Stand-Alone Program File
2
3 172714 011401          LOAD:  MOV    @R4,R1                ;R1 = size of file in blocks
4 172716 000301          SWAB   R1                          ; * 256. = word count
5 172720 006301          ASL    R1                          ; * 2 = byte count
6 172722 004767 000042   CALL   READZU                     ;Read the program file into memory
7 172726 100002          BPL    1$                          ;If HI, error in read-ABORT
8 172730                ABORT  <Stand-alone file read error>
9 172734 013705 000040   1$:  MOV    @#RT$STA,R5           ;Get program start adrs
10 172740 032705 000001   BIT     #1,R5                    ;Is adrs even?
11 172744 001402          BEQ    START$                     ;If EQ yes - okay
12 172746                ABORT  <Illegal transfer address>
13
14 172752          START$:
15 172752 012601          MOV    (SP)+,R1                   ;Pass the CSR address
16 172754 112600          MOVB   (SP)+,R0                   ;Get unit number booted
17 172756 013706 000042   MOV    @#RT$ISP,SP               ;Load program's stack pointer
18 172762 005067 175010   CLR    TRAP4                     ;Disable trap to 4 feature
19 172766 000115          JMP    @R5                        ;Go start program execution
20
21 172770          READZU:
22 172770 005004          CLR    R4                          ;Load at 0
23 172772 016602 000004   READU: MOV    4(SP),R2            ;Get unit number
24 172776 000407          BR     AROUND$                    ; SKIP OVER THE ENTRY POINT
  
```

```
1                                     .SBTTL 173000G ENTRY POINT
2
3      173000                        .=173000
4 173000
5 173000 106427 000340      ST173:: MTPS    #PRI7      ;Can't assume anything here.
6 173004 000005            RESET      ;But PWR$UP usually does.
7 173006 005000            CLR        R0      ;DELAY for sake of DLART.
8 173010 077001            SOB        R0, .    ;(maint. bit cleard by RESET)
9
10 173012 000167 175242      JMP        PWR$UP
```

1					.SBTTL	BOOTS-Continued
2	173016			AROUN3:		
3	173016	105767	174746	TSTB	B.CNTL	
4	173022	100402		BMI	DREAD	;Bit 7 set for RX01/RX02
5	173024	000167	000370	JMP	TREAD	;Read from tape

```

1          .SBTTL BOOTS-RX01/RX02 Read routines
2
3          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5          ;;;
6          ;;;          FLOPPY DISK READ ROUTINES          ;;;
7          ;;;
8          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9          ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10
11         ; With register set up as below, read the appropriate number of
12         ; full sectors from the floppy, at either density, with either
13         ; RXV21 DMA or RXV11 Programmed I/O interface.
14         ;
15         ; R0: Starting block number for transfer.
16         ; R1: Byte count for transfer
17         ; R2: Unit number
18         ; R4: Address of buffer to receive data
19
20         .ENABL LSB
21 173030 010446 DREAD: MOV     R4,-(SP)          ;Save buffer address
22 173032 010046      MOV     R0,-(SP)          ;Save starting LBN
23 173034 010146      MOV     R1,-(SP)          ;Save byte count
24
25         ; Check status and media density of selected drive
26
27 173036 012701 177172      MOV     #RXDB,R1          ;Set up R1 for benefit of RXG0
28 173042 005000      CLR      R0                ;Initialize current unit/density word
29 173044 006002      ROR      R2                ;Bit 0 set = unit 1
30 173046 103002      BCC      1$
31 173050 052700 000020      BIS     #RX$$UN,R0        ;Set unit 1
32 173054 004567 000312 1$: JSR     R5,RXG0          ;Start a read status operation
33 173060 000013      .WORD    RX$RST              ; to determine status and density
34 173062 111102      MOVB     @R1,R2              ;Pick up low byte of status
35 173064 100402      BMI      2$                  ;If PL, drive not ready
36 173066      ABORT    <Floppy drive not ready>
37 173072 032702 000040 2$: BIT     #RXE$DN,R2        ;Check media density
38 173076 001411      BEQ      3$                  ;If EQ, single density
39
40         ; Double density.
41         ; Logical sector number = logical block number * 2
42         ; Sector count = byte count/256.
43
44 173100 052700 000400      BIS     #RX$$DE,R0        ;Set double density in command
45 173104 012602      MOV      (SP)+,R2            ;Byte count
46 173106 000302      SWAB     R2                  ;Divide by 256
47 173110 012603      MOV      (SP)+,R3            ;LBN
48 173112 006303      ASL      R3                  ;Multiply by 2
49 173114 012704 000200      MOV     #128.,R4        ;Words per sector
50 173120 000410      BR       4$
51
52         ; Single density.
53         ; Logical sector number = logical block number * 4
54         ; Sector count = byte count/128.
55
56 173122 012602 3$:  MOV      (SP)+,R2            ;Byte count
57 173124 000302      SWAB     R2                  ;Divide by 256

```

```

58 173126 006302          ASL      R2              ;And multiply by 2
59 173130 012603          MOV      (SP)+,R3        ;LBN
60 173132 006303          ASL      R3
61 173134 006303          ASL      R3              ;Multiply by 4
62 173136 012704 000100   MOV      #64.,R4        ;Words per sector
63
64                          ; Set up stack as follows:
65                          ; 0(SP) = Logical sector number
66                          ; 2(SP) = Sector count
67                          ; 4(SP) = Words per sector
68                          ; 6(SP) = Buffer address
69
70 173142 010446          4$:      MOV      R4,-(SP)      ;Words per sector
71 173144 010246          MOV      R2,-(SP)      ;Sector count
72 173146 010346          MOV      R3,-(SP)      ;Logcal Sector number
73
74                          ; Start the read operation.
75                          ; This is the top of the loop.
76
77 173150 004567 000216   5$:      JSR      R5,RXG0      ;Start a sector read
78 173154 000007          .WORD   RX$RED
79
80                          ; Convert Logical Sector Numbers to Physical tracks and sectors.
81
82 173156 011603          MOV      @SP,R3          ;Get Logical Sector Number
83 173160 012702 000010   MOV      #8.,R2          ;Loop count
84 173164 022703 006400   6$:      CMP      #26.*200,R3 ;Does 26 go into dividend?
85 173170 101002          BHI      7$              ;Branch if not, C clear (BHI >= BCC)
86 173172 062703 171400   ADD      #-26.*200,R3    ;Subtract 26 from dividend (C set)
87 173176 006103          7$:      ROL      R3          ;Shift dividend and quotient
88 173200 005302          DEC      R2              ;Decement loop count
89 173202 003370          BGT      6$              ;Branch till divide done
90 173204 110302          MOV      R3,R2          ;Copy track number
91 173206 105003          CLRB     R3              ;Remove track number from remainder
92 173210 000303          SWAB     R3              ;Get remainder
93 173212 022703 000014   CMP      #12.,R3        ;C=1 if 13<=R3<25, else C=0
94 173216 006103          ROL      R3              ;Sector*2 (2:1 interleave)
95                          ; [+1 (C) if sector 13-25]
96 173220 006302          ASL      R2              ;Double the track number
97 173222 060203          ADD      R2,R3          ;Skew the sector
98 173224 060203          ADD      R2,R3          ; by adding in
99 173226 060203          ADD      R2,R3          ; 6 * track number
100 173230 006202          ASR      R2              ;Undouble the track number
101 173232 005202          INC      R2              ; and make it 1-76 (Skip track 0
102                          ; for ANSI)
103 173234 162703 000032   8$:      SUB      #26.,R3    ;Put sector
104 173240 002375          BGE      8$              ; into range
105 173242 062703 000033   ADD      #27.,R3          ; 1-26
106
107                          ; Read the sector
108
109 173246 010311          MOV      R3,@R1          ;Set sector number
110 173250 004514          JSR      R5,@R4          ;
111 173252 010211          MOV      R2,@R1          ;Set track number
112 173254 004514          JSR      R5,@R4          ;Perform a sector read
113 173256 100002          BPL      9$              ;If HI, error
114 173260                ABORT    <Floppy read error>

```

```

1                                     ; Empty RXV11/RXV21 buffer into RAM
2
3 173264 004567 000102          9$:   JSR      R5,RXG0          ;Start empty buffer function
4 173270 000003                .WORD   RX$EMP          ; and wait for TR
5 173272 032737 004000 177170    BIT     #RX$$02,@#RXCS    ;Is DMA available?
6 173300 001407                BEQ      10$             ;IF EQ no - handle as RX01
7
8                                     ; RX02 DMA Operation
9
10 173302 016611 000004          MOV     4(SP),@R1          ;Else load word count
11 173306 004514                JSR      R5,@R4            ;Wait for TR
12 173310 016611 000006          MOV     6(SP),@R1          ;And load current bus address
13 173314 004514                JSR      R5,@R4            ;Wait for DONE
14 173316 000410                BR       12$
15
16                                     ; RX01 Programmed I/O Operation
17
18 173320 016603 000004    10$:   MOV     4(SP),R3          ;Get word count
19 173324 006303                ASL      R3                ;Turn word count into byte count
20 173326 016602 000006          MOV     6(SP),R2          ;Get starting bus address
21 173332 111122    11$:   MOVB      @R1,(R2)+            ;Move one byte from buffer to memory
22 173334 004514                JSR      R5,@R4            ;Wait for TR or DONE
23 173336 077303                SOB      R3,11$            ;Loop for all bytes in first sector
24
25                                     ; Loop back if not yet finished
26
27 173340 016603 000004    12$:   MOV     4(SP),R3          ;Get word count
28 173344 006303                ASL      R3                ;Turn into byte count
29 173346 060366 000006          ADD     R3,6(SP)          ;Update bus address
30 173352 005216                INC      @SP              ;Update Logical Sector Number
31 173354 005366 000002          DEC     2(SP)            ;Decrement Sector Count
32 173360 001273                BNE      5$                ;Read another sector
33 173362 062706 000010          ADD     #8.,SP            ;Pop the stack
34 173366 000257                CCC                      ;Clear condition codes
35                                     ; to show success.
36 173370 000207                RETURN          ;All done
37 .DSABL LSB

```

```

1          ; The main subroutine for sending disk commands and waiting for
2          ; their completion.
3          ;
4          ; Register usage:
5          ;      R0 = density bit ! unit select bit (proto for commands)
6          ;      R1 = RXDB address
7          ;      R4 = RXGO TR/DONE test routine pointer
8          ;
9
10 173372 012504      RXGO:  MOV     (R5)+,R4          ;Copy command word to use
11 173374 050004      BIS      R0,R4                ;Set unit # and density
12 173376 010437 177170  MOV     R4,@#RXCS          ;Start operation
13 173402 010704      MOV     PC,R4                ;Copy adrs for later calls
14 173404 005741      TST     -(R1)                ;R1 -> RXCS
15 173406 032711 000240 1$:  BIT     #RX$$TR!RX$$DN,@R1 ;Wait for TR or DONE
16 173412 001775      BEQ     1$                    ;If EQ, neither are true yet
17 173414 005721      TST     (R1)+                ;Reset R1 -> RXDB and check for error
18 173416 000205      RTS      R5                    ;Return to caller
19

```



```

1                                     .SBTTL BOOTS-TU58 Read routines
2
3                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
5                                     ;;;;
6                                     ;;;;      TU58 DECTape II READ ROUTINES      ;;;;
7                                     ;;;;
8                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
9                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
10                                    ;
11                                    ; Starts a read operation on the TU58 by transmitting a command packet
12                                    ;
13                                    ; Inputs:
14                                    ;      R0 = starting block # for transfer
15                                    ;      R1 = byte count for transfer
16                                    ;      R2 = unit number
17                                    ;      R4 = address of bufffer to receive data
18                                    ; Outputs:
19                                    ;      R0, R1, R2 unchanged
20                                    ; Destroys:
21                                    ;      R3, R4, R5
22
23                                    .ENABL LSB
24 173420 010446      TREAD:  MOV    R4,-(SP)                ;Save buffer address
25 173422 005004      CLR     R4                            ;Init checksum
26 173424 012703 005002  MOV    #10.*400+R$$CTL,R3          ;Set command flag and length
27 173430 004767 000206  CALL   CH2OUT                      ;Output 2 chars and set R5
28 173434 012703 000002  MOV    #R$READ,R3                 ;Send read command and modifier=0
29 173440 004715      CALL   @R5
30 173442 010203      MOV    R2,R3                          ;Then unit number and switches=0
31 173444 004715      CALL   @R5
32 173446 005003      CLR    R3                              ;Plus a zero sequence number
33 173450 004715      CALL   @R5
34 173452 010103      MOV    R1,R3                          ;Followed by the byte count
35 173454 004715      CALL   @R5
36 173456 010003      MOV    R0,R3                          ;And the block number
37 173460 004715      CALL   @R5
38 173462 010403      MOV    R4,R3                          ;Finally, transmit the checksum
39 173464 004715      CALL   @R5

```

```

1          ; Now ready to accept data messages from the TU58
2
3 173466 012600          MOV      (SP)+,R0          ;R0 -> data buffer
4          ;      CLC          ;(CH2OUT leaves C clear)
5 173470 006001          ROR      R1          ;R1 = word count for transfer
6 173472 004767 000116  1$:  CALL      7$          ;Get first word of packet
7 173476 122703 000001  CMPB     #R$DAT,R3        ;Is this indeed a data message?
8 173502 001017          BNE      3$          ;If NE no - may be END message
9 173504 105003          CLR      R3          ;Else clear clags
10 173506 000303          SWAB     R3          ;Move packet byte count to low byte
11 173510 106003          RORB     R3          ;And convert to word count
12 173512 160301          SUB      R3,R1          ;Remove from transfer count
13 173514 010305          MOV      R3,R5          ;And copy for loop counter
14 173516 004767 000102  2$:  CALL      9$          ;Get next two words
15 173522 010320          MOV      R3,(R0)+        ;Store in buffer
16 173524 077504          SOB      R5,2$          ;Loop for entire data message
17 173526 004767 000044  CALL      5$          ;Get checksum and compare
18 173532 005701          TST      R1          ;Have all data records been
19          ; transferred?
20 173534 001356          BNE      1$          ;If NE no
21 173536 004767 000052  CALL      7$          ;And get prospective
22          ; END packet start
23 173542 004767 000056  3$:  CALL      9$          ;Get opcode/success bytes
24          ; of END packet
25 173546 122703 000100  CMPB     #R$END,R3        ;Is this an END packet?
26 173552 001402          BEQ      4$          ;IF NE no - ABORT
27 173554          ABORT     <TU58 END packet missing>
28 173560 010300  4$:  MOV      R3,R0          ;Save success code in R0
29 173562 004767 000032  CALL      8$          ;Read remainder of END packet
30 173566 004767 000004  CALL      5$          ;And check its checksum
31 173572 000300          SWAB     R0          ;Set CC's on success code of transfer
32 173574 000207          RETURN          ;Return to caller
33
34 173576 004767 000064  5$:  CALL      CH2IN          ;Get two checksum bytes
35 173602 020403          CMP      R4,R3          ;Does it match calculated value?
36 173604 001402          BEQ      6$          ;If NE no - error
37 173606          ABORT     <TU58 checksum error>
38
39 173612 000207  6$:  RETURN          ;Else return with success
40
41 173614 005004  7$:  CLR      R4          ;Init checksum
42 173616 000402          BR       9$          ;And get the first word
43
44 173620 004717  8$:  CALL      @PC          ;Read 4 words
45 173622 004717          CALL      @PC
46 173624 004767 000036  9$:  CALL      CH2IN          ;Read next two bytes
47 173630 060304          ADD      R3,R4          ;Add into checksum
48 173632 005504          ADC      R4          ; with end-around carry
49 173634 000207          RETURN          ;Adn back to caller
50          .DSABL  LSB

```

```

1          ; CH2OUT -- write two bytes to the TU58
2          ;
3          ; Writes two bytes to interface and updates checksum.
4          ;
5          ; Inputs:
6          ;      R3 = two bytes to be output; low byte first
7          ;      R4 = current checksum word
8          ; Outputs:
9          ;      R3 unchanged
10         ;      R4 updated to new checksum
11         ;      R5 pointing to CH2OUT routine for easier future CALLs
12
13 173636   CH8OUT:
14 173636   004717   CALL    @PC                ;Entry point to output 8 characters
15 173640   004717   CALL    @PC
16 173642
17 173642   010705   CH2OUT:  MOV    PC,R5                ;Set R5 to collowing routine adrs
18 173644   060304           ADD    R3,R4                ;Update checksum word
19 173646   005504           ADC    R4                    ; with end-around carry
20 173650   004717           CALL    @PC                ;Repeat for both characters
21 173652   105737   176544   1$:  TSTB    @#$T$CSR        ;Is interface ready for output?
22 173656   100375           BPL     1$                  ;If PL no - wait
23 173660   110337   176546           MOVB   R3,@#$BFR      ;Else transmit character to TU58
24 173664   000407           BR      CHRET               ;Merge with other routine to retrn
25
26         ; CH2IN -- Read two bytes from the TU58
27         ; CHIN  -- Read a single byte from the TU58
28         ;
29         ; Inputs:
30         ;      none.
31         ; Outputs:
32         ;      R3 = character(s) read
33
34 173666   004717   CH2IN:  CALL    @PC                ;Read two, not one
35 173670   105003   CHIN:   CLR    R3                  ;And zero out space for new one
36 173672   105737   176540   1$:  TSTB    @#$I$CSR        ;Is a character available
37 173676   100375           BPL     1$                  ;IF PL no
38 173700   153703   176542           BISB    @#$I$BFR,R3    ;Else set into register
39 173704   000303   CHRET:  SWAB    R3                  ;Move current character over
40 173706   000207           RETURN                   ;And return to caller

```

KXT11-A2 1K FIRMWARE    MACRO V04.00    5-OCT-81 22:56:28 PAGE 63  
END STATEMENT

1  
2

000001

.END

.SBTTL END STATEMENT

AROUN2 172010	E.PAR = 000001	PP.BI5= 000012	RT\$UER= 000053	R.NXM = 000200
AROUN3 170416	FAKOUT 170424	PP.BI6= 000014	RT\$USR= 000046	R.PC = 167766 G
AUTOBA 170472 G	FILNAM= 000002	PP.BI7= 000016	RXBOOT 172400	R.STAK= 000001
BADBOT 172370	GETCHR 171510	PP.C = 176204	RXCS = 177170	R.STRT 170036 G
BAUDR\$= 000032	GETNUM 171612	PP.CHI= 000010	RXDB = 177172	R.TYPE= 167762 G
BD.003= 000000	HGHSEG= 001004	PP.CLO= 000001	RXESCR= 000001	SAVPC = 167752 G
BD.006= 000010	HKBDQ 171140	PP.CWR= 176206	RXESDD= 000100	SAVPS = 167754 G
BD.012= 000020	HKBD\$ 171142	PP.DMA= 000040	RXESDE= 000020	SEGALO= 001000
BD.024= 000030	HVBAUD 170556	PP.DRA= 000020	RXESDN= 000040	SPACE = 000040
BD.048= 000040	IENTAS= 000400	PP.DRB= 000002	RXESDR= 000200	SRET 171636
BD.096= 000050	INBYTE 170550	PP.MDB= 000004	RXESID= 000004	STANDB 172564
BD.192= 000060	INBYT\$ 170556	PP.MD2= 000100	RXESUN= 000400	START 172000 G
BD.384= 000070	INITS 171750	PP.MOD= 000200	RXGO 173372	START\$ 172752
BIT0 = 000001	IN.USR= 167764 G	PRINT 170706	RX\$EMP= 000003	STRBLK= 001010
BIT1 = 000002	KBDQ 170674	PRI6 = 000300	RX\$FIL= 000001	STTUBD 172172 G
BIT10 = 002000	KBD\$ 170702	PRI7 = 000340	RX\$REC= 000017	ST173 173000 G
BIT11 = 004000	LCSET 171246	PUTCHR 171522	RX\$RED= 000007	SWCMD 171212
BIT12 = 010000	LEDOFF= 000017	PUTCLF 171554	RX\$RST= 000013	\$SCART= 177767
BIT13 = 020000	LF = 000012	PUTLF 171564	RX\$STD= 000011	\$SDCHK= 177757
BIT14 = 040000	LOAD 172714	PUTSTR 171542	RX\$WDD= 000015	\$SMOTR= 177737
BIT15 = 100000	LOCDSP 171252	PWR\$UP 170260 G	RX\$WRT= 000005	\$SNORM= 000000
BIT2 = 000004	MODE = 000221	QODT 170666	RX\$\$DE= 000400	\$SOPCD= 177730
BIT3 = 000010	MSGQ 171730	RAMBOT= 160010	RX\$\$DN= 000040	\$SPART= 177776
BIT4 = 000020	MSG\$ 171731	RAMTOP= 167776	RX\$\$ER= 100000	\$SRECN= 177711
BIT5 = 000040	NEXNUM 171600	RBUF\$1= 177562	RX\$\$FN= 000016	\$SRETR= 000001
BIT6 = 000100	NOCT 171642	RBUF\$2= 176542	RX\$\$GO= 000001	\$SSEEK= 177740
BIT7 = 000200	NO.LOW= 100000	RB.BRK= 004000	RX\$\$IE= 000100	\$SUNIT= 177770
BIT8 = 000400	NXTSEG= 001002	RB.ERR= 100000	RX\$\$IN= 040000	\$SWPRT= 177765
BIT9 = 001000	OCTSTR 171656	RB.FRM= 020000	RX\$\$TR= 000200	TI\$BFR= 176542
BOOTS 172164 G	OCTST0 171652	RB.OVR= 040000	RX\$\$UN= 000020	TI\$CSR= 176540
BRKN00 170010 G	ODT 170602 G	RCMD 171144	RX\$\$XA= 030000	TO\$BFR= 176546
B.CNTL= 167770 G	ODTFLG= 167750 G	RCMD1 171232	RX\$\$XX= 003000	TO\$CSR= 176544
CHIN 173670	ODTL0C= 167746 G	RCSR\$1= 177560	RX\$\$02= 004000	TRAP4 = 167776 G
CHK240 172426	ODTSTK= 167744 G	RCSR\$2= 176540	R\$ABRT= 000006	TREAD 173420
CHRET 173704	ODTWHY= 167774 G	RC.ACT= 004000	R\$COMP= 000004	TUBAUD= 000072
CH2IN 173666	ONENUM 171576	RC.DUN= 000200	R\$DIAG= 000007	TUBOOT 172464
CH2OUT 173642	O.CNTL= 167772 G	RC.IEN= 000100	R\$END = 000100	T.BIT = 000020
CH8OUT 173636	PATERN 171750	READU 172772	R\$GETC= 000012	USERSP= 167760 G
CR = 000015	PBR0 = 000010	READZU 172770	R\$GETS= 000010	VECSET 170440 G
DEVBIT= 000200	PBR1 = 000020	REGOUT 171240	R\$INIT= 000001	XBUF\$1= 177566
DEVNUM= 000001	PBR2 = 000040	RESTAR 172004 G	R\$NOP = 000000	XBUF\$2= 176546
DIAGNO 171754	PCMD 171044	RETRY = 000010	R\$POSI= 000005	XCSR\$1= 177564
DIRBUF= 001000	PERMF\$= 002000	RFLAG = 000200	R\$READ= 000002	XCSR\$2= 176544
DONE 171574	PP.A = 176200	RPOINT= 167756 G	R\$SETC= 000013	XC.BRK= 000001
DREAD 173030	PP.B = 176202	RT\$ENT= 000052	R\$SETS= 000011	XC.IEN= 000100
D.FLEN= 000010	PP.BIC= 000000	RT\$FCH= 000056	R\$WRIT= 000003	XC.MNT= 000004
EMPTY\$= 001000	PP.BIS= 000001	RT\$FCT= 000057	R\$\$CON= 000020	XC.PBE= 000002
ENDSG\$= 004000	PP.BI0= 000000	RT\$HGH= 000050	R\$\$CTL= 000002	XC.RDY= 000200
ENTSI\$= 000016	PP.BI1= 000002	RT\$ISP= 000042	R\$\$DAT= 000001	XTRBYT= 001006
ERRBIT 171742	PP.BI2= 000004	RT\$JSW= 000044	R\$SINT= 000004	\$STACK= 167644 G
E.EXT = 000100	PP.BI3= 000006	RT\$JMN= 000054	R\$XOF= 000023	\$\$\$BRK 170000 G
E.INT = 000010	PP.BI4= 000010	RT\$STA= 000040	R.HALT= 100000	\$\$\$LTC 170006 G

. ABS. 174000 000  
000000 001  
ERRORS DETECTED: 0

KXT11-A2 1K FIRMWARE    MACRO V04.00    5-OCT-81 22:56:28 PAGE 63-2  
SYMBOL TABLE

VIRTUAL MEMORY USED: 9216 WORDS ( 36 PAGES)  
DYNAMIC MEMORY AVAILABLE FOR 67 PAGES  
,FALCON/C=FALCON

\$\$\$BRK	14-16#	27-35						
\$\$\$LTC	14-20#	27-37						
\$STACK	13-30#	24-13	27-17	49-77	49-91			
AROUN2	44-47	46-2#						
AROUN3	54-24	56-2#						
AUTOBA	27-13	28-43#						
B.CNTL	13-14#	26-49*	49-54*	49-55	56-3			
BADBOT	49-66	49-90#						
BAUDR\$	9-14#	28-44						
BD.003	7-21#							
BD.006	7-22#							
BD.012	7-23#							
BD.024	7-24#	9-14						
BD.048	7-25#							
BD.096	7-26#							
BD.192	7-27#							
BD.384	7-28#	9-18						
BIT0	5-5#	7-43	8-29	8-47	10-5	10-13	35-50	
BIT1	5-6#	7-36	8-27	8-38	8-40	8-42	8-44	
BIT10	5-15#							
BIT11	5-16#	6-16	6-37					
BIT12	5-17#							
BIT13	5-18#	6-35						
BIT14	5-19#	6-32						
BIT15	5-20#	6-30	10-3	10-9				
BIT2	5-7#	7-30	8-25	8-38	8-39	8-42	8-43	
BIT3	5-8#	7-15	8-23	8-38	8-39	8-40	8-41	
BIT4	5-9#	7-16	8-21	9-34				
BIT5	5-10#	7-17	8-18	38-11				
BIT6	5-11#	6-23	7-8	8-17	38-9			
BIT7	5-12#	6-19	7-3	8-15	9-32	10-4	10-11	
BIT8	5-13#							
BIT9	5-14#							
BOOTS	33-21	49-25#						
BRKN00	14-18	14-26#	21-17					
CH2IN	61-34	61-46	62-34#					
CH2OUT	60-27	62-16#						
CH8OUT	52-17	62-13#						
CHIN	62-35#							
CHK240	51-17#	52-36						
CHRET	62-24	62-39#						
CR	5-25#	37-26	38-21	40-24	41-20	43-13		
D.FLEN	48-125#	53-47						
DEVBIT	10-11#	49-37						
DEVNUM	10-13#	49-83						
DIAGNO	33-25	44-37#						
DIRBUF	48-62#	48-118	48-119	48-120	48-121	48-122	53-21	
DONE	40-17	40-31#						
DREAD	50-21	56-4	57-21#					
E.EXT	10-17#	44-17						
E.INT	10-18#	44-18						
E.PAR	10-19#	46-6						
EMPTY\$	48-127#							
ENDSG\$	48-129#	53-31						
ENTSI\$	48-124#							
ERRBIT	44-19#	46-11						

[illegible]



[illegible]



S\$SEEK	48-109#							
S\$UNIT	48-105#							
S\$WPRT	48-107#							
SAVPC	13-25#	14-27*	21-37*	27-12*	32-39	35-11*	35-54	36-24
SAVPS	13-24#	14-28*	21-36*	27-11*	35-16*	35-53	36-33	37-45
SEGALO	48-118#							
SPACE	5-26#	37-23						
SRET	41-21	41-32#						
ST173	55-4#							
STANDB	51-21	53-15#						
START	45-4#							
START\$	54-11	54-14#						
STRBLK	48-122#	53-25						
STTUBD	49-30#							
SWCMD	36-18	36-29#						
T.BIT	9-34#	37-50						
TI\$BFR	48-69#	62-38						
TI\$CSR	48-68#	52-13	52-25	62-36				
TO\$BFR	48-71#	62-23*						
TO\$CSR	48-70#	49-26*	52-14	62-21				
TRAP4	13-5#	22-11	49-78*	54-18*				
TREAD	56-5	60-24#						
TUBAUD	9-18#	49-26						
TUBOOT	49-88	52-12#						
USERSP	13-20#	32-20*	32-25					
VECSET	26-46	27-34#	51-22					
XBUF\$1	6-8#	39-22*						
XBUF\$2	6-12#	48-71						
XC.BRK	7-43#	52-20						
XC.IEN	7-8#							
XC.MNT	7-30#	44-26						
XC.PBE	7-36#	9-14	9-18	44-25	44-26			
XC.RDY	7-3#	24-39						
XCSR\$1	6-7#	24-32*	24-39	28-44*	29-37*	39-20		
XCSR\$2	6-11#	48-70						
XTRBYT	48-121#	53-50						

KXT11-A2 1K FIRMWARE    MACRO V04.00    5-OCT-81 22:56:28 PAGE M-1  
CROSS REFERENCE TABLE (CREF V04.00 )

[illegible]