6800 Terminal Card

Software

NATIONAL

53-110

Made in U.S.A.

6800 Software for the Terminal Card

I/O Registers

0		Display Address Register	<15:8>				18
1		Display Address Register	<7:0>				19
2	P	Control Register A	[CA1]	INIT		IRQ1	1A
3	P	Control Register B	- - -				1B
4		Cursor Address Register	<15:8>				1C
5		Cursor Address Register	<7:0>				1D
6	P	Control Register A	- - -				1E
7	P	Control Register B	[CB1]	Vertical Retrace		NMI2	1F
8	10	Current Scan Line					20
9	11	Interrupt Scan Line					21
A	12	P Control Register A	- - -				22
B	13	P Control Register B	[CB1]	Scan Line		NMI3	23
C	14	Characters per Line					24
D	15	Display Control					25
E	16	P Control Register A	[NMI Strobe]				26
F	17	P Control Register B	[CB1]	ODD FRAME		NMI1	27
10	20	Word Data from BUS	<15:08>				28
11	21	Word Data from BUS	<7:0>				29
12	22	P Control Register A	[CA1/CA2]	Data Ready		IRQ8	2A
13	23	P Control Register B	- - -				2B
14	24	Word Data to BUS	<15:08>				2C
15	25	Word Data to BUS	<7:0>				2D
16	26	P Control Register A					2E
17	27	P Control Register B	[CB1/CB2]	Data Ready		IRQ9	2F

I/O Registers

18	30		Terminal Data from 11		
19	31		Terminal Data to 11		
1A	32	P	Control Register A	[CA1/CA2]	$\overline{IRQ6}$
1B	33	P	Control Register B	[CB1/CB2]	$\overline{IRQ7}$
1C	34		Keyboard ASCII		
1D	35		Keyboard Buttons		
1E	36	P	Control Register A	[CA1]	$\overline{IRQ5}$
1F	37	P	Control Register B	Bell Control	
20	40				
21	41				
22	42	P	Control Register	60 Hz clock	$\overline{IRQ2}$
23	43	P	Control Register		$\overline{IRQ4}$
24	44		Photoreader Data		
25	45		NPR Control Register		
26	46	P	Control Register A	[CA1]	$\overline{IRQ3}$
27	47	P	Control Register B	[CB1/CB2]	$\overline{IRQ10}$
28	50		NPR Address	<15:08>	
29	51		NPR Address	<7:0>	
2A	52	P	Control Register A	Tape ←	$\overline{IRQ11}$
2B	53	P	Control Register B	Tape →	$\overline{IRQ12}$
2C	54		NPR Data to BUS	<15:08>	
2D	55		NPR Data to BUS	<7:0>	
2E	56	P	Control Register	Here Is	$\overline{IRQ13}$
2F	57	P	Control Register	Paper	$\overline{IRQ14}$

16 July 78
APP

I/O Registers

30	60		NPR Dat from BUS	<15:08>	
31	61		NPR Data from BUS	<7:0>	
32	62	P	Control Register A	Break	IRQ15
33	63	P	Control Register B	Repeat	IRQ16
34	64	}	Dummy		
35	65				
36	66				
37	67				
	70	}	Dummy		
	71				
	72				
	73				
	74	}	Dummy		
	75				
	76				
	77				

Direct variables

100 –	OPR OPR+1	} GRAPHIC Input Word
102 –	GCSR	GRAPHIC STATUS Register
103 –	LPCNTR	Division counter
104 –	DND	Dividend
106 –	DSR	Divisor
110 –	QT	Quotient
112 –	VCNTR	Vector plot counter
114 –	CITH	in range check word for plotting
116 –	SAVEX	Temporary save location
120 –	BT/LT	
122 –	CADR	
124 –	RDADD	
126 –	RAD	
130 –	HADD	
131 –		
132 –	BYTE	
133 –		
134	ROWADD	
136	ROW RST	
140	ROW UPD	
142	COL ADD	
144	COL RST	
146	COL UPD	

Division routine

16 Jul 78
AGD

Direct Variables

Quick variables

200-201	CURLOC
202-203	PGMSTK
204-205	CRSTK
206-207	PMSTK
210	SRTLIN
211	LINES
212	LOCK
213	TFLAG
214	RFLAG
215	CURCNT
216	UPDFLAG
217	GRAPH
220	ONLINE
221	NALL
222	NUMB
223	DVAL
224	CURS — Data in Cursor indicator
225	CNTR
226	DRCTN
227	
230	IBIS
231	IBIC
232	OBIS
233	OBIC
234	DCTL
235	CWORD — Write Character Routine (Graphic)
236	K1
237	K2

16 July 78
ARS

Direct Variables

240	CLPNTR
242	CLBASE
244	CLP BOT
246	CLP TOP
250	CL BOT
252	CLTOP
254	PLPNTR
256	PLBASE
260	TXT CBS
261	CURDSP
262	NJSR
264	GJSR
266	PRNT ON
270	TEMP
272	EVNSCN
274	ODD SCN
276	OVBASE

Printer Handler

used By Graphic Clear Routine

Direct Variables

300	DSPØ
302	DSP1
304	DSP2
306	
310	PAGEM
311	PAGEX
312	PAGEY
313	GCNT
314	ESH
315	ESL
316	GTABLE
320	GSTADD
322	GLBASE
324	GLCNT
326	GPCNT
327	
330	GRAMB
332	TCLR

16 Jul, 78
ARD

Direct Variables

340 -	TEMPA		1
341 -	TEMPB		2
342 - 343	TEMPX		2
344	PKX		
346	IVX		2
350	RVEC		2
352			2
354	VA		
355	VL		
356	VAL		
360-1	'TO' Address	} FILL	
362-3	'From' Address	} FILL	
364	length	} FILL	
366			
370			
372			
374			
376			

other variables

1 ∅ – 1 255_8	Text & Graphic Control Data
1 300 – 1 377	DSP STK
2 ∅ – 2 117	STRING
2 120 – 2 147	@STRNG
2 200 – 2220	Y Table
2 230 – 2 250	X TABLE
2 260 – 2 277	PLOT Table

16 Jul, 78
ABS

BASIC Control ROM (Top 2K Bytes)

174000_8

START	216	14	77	LDS	14	77	set stack pointer
	275	370	200	JSR	CLKD		start CLOCK
	275	371	110	JSR	RAMSET		set up character RAM
	275	371	150	JSR	DSINIT		start Display
	275	373	200	JSR	BRKSET		set up Break Key
	275	373	350	JSR	DATSET		set up TERMINAL INPUT
	275	374	60	JSR	KBSET		Set up Keyboard, Bell, Repeat & Photo Reader
	275	375	160	JSR	SETSCN		Set up Scrolling Buttons
	275	376	210	JSR	SYST		Print System ID
	376	360	0	LDX	360	0	} Is next 2K installed
	214	255	255	CPX	360	255	}
	46	5		BNE	SELF		NO - branch
	376	360	2	LDX	360	2	} else load Transfer vector and JMP
	156	0		JMP	0, X		}
SELF	40	376		BR	SELF		

* * * CONTROL PROCESSING HANDLER * * *

174120_8

CTLP	227	340	STA A	TEMPA	same character
	177	0 341	CLR	TEMPB	enable printing of character
	40	7	BR	SKIP	
	ϕ	ϕ ϕ			
	ϕ	ϕ			
	ϕ	ϕ			
SKIP	376	360 0	LDX	360 ϕ	get first location of next lower 2K Block
	214	255 255	CPX	# 255 255	is it 255 255
	46	5	BNE	NONEX	if not - NO transfer
	376	360 4	LDX	360 4	else load transfer vector
	255	ϕ	JSR	ϕ,X	JSR to transfer vector
NONEX	175	ϕ 341	TST	TEMPB	is printing inhibited
	46	11	BNE	DONE	if yes - branch
CHARX	226	34ϕ	LDA A	TEMPA	get character and mask
	232	230	ORA A	IBIS	
	224	231	AND A	IBIC	
(4354)	275	374 270	JSR	CHAR	print character
DONE	71		RTS		Return to caller

16 Jul 78
ARD

Clock Interrupt Setup

174200_8

CLKO	17		SEI	hold interrupts
(4402)	316 370 300		LDX #CLKINT	} set up Vector
	377 17 342		STX IRQ (=$FFE2)	
	206 5		LDA A #05	} set clock interrupt control
	227 42		STA A $22	
	226 40		LDA A $20	clear any pending interrupt
	206 40		LDA #' SP	get space character
	316 40 0		LDX 40 0	first character of Control line
LOOP	247 0		STA A 0,X	} put spaces in all 80 characters of Control line
	10		INX	
	214 40 120		CPX # 40 120	
	46 370		BNE LOOP	

316 40 70	LDX # 40 70	address for time display	
206 60	LDA A #'0	get a Zero character	
247 0	STA A 0,X		
247 1	STA A 1,X		
247 3	STA A 3,X	STORE 0's	
247 4	STA A 4,X		
247 6	STA A 6,X		
247 7	STA A 7,X		
206 72	LDA A #':'	get : character	
247 2	STA A 2,X	Store :'s	
247 5	STA A 5,X		
206 101	LDA A #'A	store A	
247 11	STA A 9,X		
206 115	LDA A #'M	store M	
247 12	STA A 10,X	time Display now	
		0 0 : 0 0 : 0 0 A M	
16	CLI	allow interrupts	
71	RTS	Return to Caller	

*** CLOCK NOW ON !!! ***

16 Jul 78
Bob

⑤ Clock Interrupt Handler

174300_8

CLKINT	175 0 214	TST RFLAG	test for Data inhibited
	47 11	BEQ TIME	if not - branch
	206 55	LDA A # $2D	Restart transfers from PDP 11
	227 32	STA A $1A	
	226 30	LDA A $18	set printer ready flag
	177 0 214	CLR RFLAG	data not inhibited
TIME	172 0 223	DEC DVAL	1/60th of a second
	46 6	BNE TDONE	1 second? no - branch
	206 74	LDA A # 60_{10}	60 60ths
	227 223	STA A DVAL	reset counter
	227 213	STA A TFLAG	set 1 second Flag
TDONE	226 40	LDA A $20	clear interrupt flag
	177 0 222	CLR NUMB	allow characters during next 60th
	73	RTI	return from interrupt

① Time clock routine (executed during Vertical Retrace)

174340_8

TIMCLK	175 0 213	TST TFLAG	a second yet ?
	47 134	BEQ TIMDON	NO - Done
	177 0 213	CLR TFLAG	Reset T-flag
	316 40 70	LDX # 40 70	time display addrs
	154 7	INC 7,X	increment seconds
	206 71	LDA A #'9	} are we past 9 ?
	241 7	CMP A 7,X	} 9-(7,X)
	54 116	BGE TIMDON	if not - Done
	306 60	LDA B #'0	} restore digit to 0
	347 7	STA B 7,X	}
	154 6	INC 6,X	increment 10's of seconds
	206 65	LDA A #'5	} are we past 5 tens ?
	241 6	CMP A 6,X	}
	54 102	BGE TIMDON	if not - Done
	347 6	STA B 6,X	restore digit to 0

16 Jul, 78
ARD

17441

154	4	INC 4,X	increment Minutes
206	71	LDA A #'9	} are we past 9 ?
241	4	CMP A 4,X	
54	70	BGE TIMDON	if not - Done
347	4	STA B 4,X	restore minutes
154	3	INC 3,X	increments 10's of Minutes
206	65	LDA A #'5	} are we past 5 ?
241	3	CMP A 3,X	
54	56	BGE TIMDON	if not - Done
347	3	STA B 3,X	restore count
154	1	INC 1,X	increment Hours
206	61	LDA A #'1	} above 10 hours ?
241	ø	CMP A ø,X	
47	14	BEQ ABove	yes - branch
206	71	LDA A #'9	} Past 9 Hrs ?
241	1	CMPA 1,X	
54	36	BGE TIMDON	if not - Done
347	1	STA B 1,X	reset Hours
154	0	INC ø,X	set to 10 hrs
40	30	BR TIMDON	Done

```
ABOVE   206  62     LDA A  #'2     } post 2 hr ?
        241  1      CMP A  1,X     }
        54   22     BGE    TIMDONE   if not - Done
        347  0      STA B  0,X       reset 10's of hours
        134         INC B            B=1
        347  1      STA B  1,X       reset Hours count

        206  101    LDA A  #'A     }
        241  11     CMP A  9,X     } are we in AM
        47   3      BEQ    NITE      if yes - Branch
        247  11     STA A  9,X       else make AM
        71          RTS              return

NITE    206  120    LDA A  #'P     } make PM
        247  11     STA A  9,X     }

TIMDON  71          RTS              return
```

16 July, 78
BOD

174510₈

Character RAM Set UP

RAMSET	316 17 377	LDX 17 377	Beginning RAM Addrs -1
	337 374	STX $FC	
	316 177 377	LDX 177 377	Beginning ROM Addrs -1
	337 376	STX $FE	
LOOP	336 376	LDX $FE	increment ROM Addrs
	10	INX	
	337 376	STX $FE	
	246 0	LDA A Ø, X	get character data
	336 374	LDX $FC	increment RAM Addrs
	10	INX	
	337 374	STX $Fc	
	247 0	STA A Ø, X	store character data
	214 37 377	CPX 37 377	are we finished?
	46 355	BNE LOOP	NO - Loop
	71	RTS	return

②

174550_8 Initialize Display System

DSINIT	316 14 100	LDX 14 100	get value for CLPBOT
	337 244	STX CLPBOT	define CLPBOT
	337 240	STX CLPNTR	} init value
	337 254	STX PLPNTR	
	206 1	LDA A #1	set all lines to 1 character Long
	40 1	BR ENTER1	go start
LOOP1	10	INX	point to next Location
ENTER1	247 ∅	STA A ∅,X	∅,X ← 1
	214 17 331	CPX 17 331	finished
	46 370	BNE LOOP1	NO – Loop
	337 246	STX CLPTOP	set top
	316 40 120	LDX 40 120	first Address for characters
	337 250	STX CLBOT	Define bottom
	337 242	STX CLBASE	} init value
	337 256	STX PLBASE	

20120

16 Jul 78
ARC

```
                  206  52        LDA A  #'*      get '*'
                  40   1         BR   ENTER2     go to ENTER2

LOOP2             10             INX             next location
ENTER2            247  0         STA A  0,X      set location = *
                  214  57 257     CPX    57 257   finished
                  46   370       BNE    LOOP2    if not - loop
                  337  252       STX    CLTOP    set top of
                                                 character memory

                  316  1  0       LDX    1  0     Base for Text
                  337  260       STX    TXTCBS
                                                 low order = CURDSP = 0

                  117            CLR A           set A = 0
                  227  212       STA A  LOCK     Display Locked
                  227  217       STA A  GRAPA    Graphics off
                  114            INC A
                  227  215       STA A  CURCNT   1 character
                  227  216       STA A  UPDFLAG  do immediate update
```

*** Set UP TEXT STOP LINE ***

```
316  1   0      LDX  1   Ø     } set to turn
377  1   222    STX  1   222   } Display off
316  7   26     LDX  7   26    } Restart at
377  1   224    STX  1   224   } Control line
```

*** Set UP Control START LINE ***

```
316  40  0      LDX  40  0     } Scan address
377  1   226    STX  1   226   }
316  120 1      LDX  120 1     } 80 character line
377  1   230    STX  1   230   } Display ON
316  7   16     LDX  7   16    } next interupt @ 16
377  1   232    STX  1   232   }
```

*** Set UP Control STOP LINE ***

```
316  1   0      LDX  1   0     } turn Display 'off'
377  1   236    STX  1   236   }
316  7   377    LDX  7   377   } inhibit further interupts
377  1   240    STX  1   240   }
```

16 Jul 78
BRB

*** Setup of Display Stack for TEXT ***

```
(5636)   275 372 250    JSR  LODATA      do first line
         206  27        LDA A #27   }    set for 23 iterations
         227 225        STA A CNTR  }
LOOP3    174 0 241      INC  CLPNTR+1 }  increment Low
         174 0 243      INC  CLBASE+1 }  order bytes
(5670)   275 372 310    JSR  UPDTXT      go update Text Line
         172  0 225     DEC  CNTR        finished?
         46  362        BNE  LOOP3       No - Loop
```

*** Set ALL Display Control Registers ***

```
         316  0  0      LDX  0. 0
         337  2         STX  $2   }
         337  6         STX  $6   }  clear ALL
         337 12         STX  $A   }  Display Control Registers
         337 16         STX  $E   }

         316 377 377    LDX  377 377   Data Direction out
         337  0         STX  $0        Display Address
         337  4         STX  $4        Cursor Address
```

```
316  0 377      LDX  Ø  377
337  1Ø         STX  $8          Scan counter IN
                                 Interrupt  OUT

316  377 177    LDX  377 177
337  14         STX  $C          characters per line
                                 Display Control

316  4   4      LDX  4   4
337  2          STX  $2
337  6          STX  $6          Set Controls to
337  12         STX  $A          Access Data Registers
337  16         STX  $E

206  377        LDA A 377.
227  11         STA A $9         inhibit scanline interrupts

316  1  0       LDX  1  0
337  14         STX  $C          Turn off Display

336  242        LDX  CLBASE
337  4          STX  CURSOR (= $4)   set Cursor Position
```

16 Jul 78
AAD

*** Set Interrupt Vectors ***

```
(6060)  316  372  100      LDX  # DISINT
        377   17  336      STX  # NMI3 = (17 336)   } set line scan interrupt

(6074)  316  372  350      LDX  # VERTIN
        377   17  334      STX  # NMI2 = (17 334)   } set vertical retrace interrupt

        316   54    4      LDX  54  4
        337   16           STX  $E                  } Set up NMI strobe

        316    4    7      LDX  4  7
        337   12           STX  $A   Turn on line scan interrupt
        337    6           STX  $6   Turn on Vertical retrace interrupt.

        *** Display is ON !!! ***

        71                 RTS       return to Caller
```

②

175100_8

*** Display Interrupt Handler ***

```
DISINT   336 300    LDX  DSP0   }  Load New Scan Address
         337 0      STX  $0     }
         336 302    LDX  DSP1   }  Load # of characters
         337 14     STX  $C     }  & New Display Control
         336 304    LDX  DSP2   }  Load New Line count
         337 10     STX  $8     }  & New interrupt line #
```

*** Display set for next line !!! ***

```
(6230)   275 372 130   JSR  DSPUPD   go setup for next interrupt
         226 11        LDA A $9      clear interrupt (dummy Load)
         226 14        LDA A $C      NMI strobe (dummy Load)
         73            RTI           return from interrupt
```

16 July 78
ARD

(16)

175130_8

*** Display UPDate Routine ***

DISUPO	237	202	STS	PGMSTH	save stack pointer
	336	204	LDX	CRSTH	get current Display stack position
	356	∅	LDX	∅,X	get address of next line Display control information
	256	∅	LDS	∅,X	Update Display data
	237	300	STS	DSP∅	
	256	2	LDS	2,X	
	237	302	STS	DSP1	
	256	4	LDS	4,X	
	237	304	STS	DSP2	
	336	204	LDX	CRSTH	update stack position
	10		INX		
	10		INX		
	337	204	STX	CRSTH	
	236	202	LDS	PGMSTH	restor stack Pointer
	71		RTS		return

⑤ *** SCROLL UP ROUTINES ***

175170$_8$

(6360) ROTUP 275 372 200 JSR CURUP update pointer to Control data
(6366) 275 372 250 JSR LD DATA
71 RTS return to Caller

④

175200$_8$

CURUP	226	261	LDA A CURDSP	
	201	212	CMP A #212	are we at top of area?
	45	3	BCS C1	NO - branch
	117		CLR A	else set at bottom
	40	2	BR C2	branch ahead
C1	213	6	ADD A #6	increment position
C2	227	261	STA A CURDSP	restore CurDsp
	71		RTS	return to calle

16 JUL 78
ARJ

*** Scroll Down routine ***

(2)

$175\,220_8$

(6440) ROTDWN	275	372	250	JSR LD DATA
(6446)	275	372	230	JSR CURDWN
	71			RTS

(3)

$175\,230_8$

CURDWN	226	261	LDA A CURDSP	get current Display position
	46	4	BNE C3	not at bottom - Branch
	206	212	LDA A # 212	else set at top
	40	2	BR C4	branch Ahead
C3	200	6	SUB A #6	Decrement Position
C4	227	261	STA A CURDSP	restor CURDSP
	71		RTS	return to Caller

③ *** Set UP Text Control Data ***

175250_8

```
LDDATA   336 254    LDX  PLPNTR    get address of current
                                   line character count
         346 0      LDA B Ø,X     get character count

         336 260    LDX  TXTCBS   get Address of Text Control
                                   Location
         347 2      STA B 2,X     set character count
         306 1      LDA B #1      } set Display Control
         347 3      STA B 3,X     }
         306 17     LDA B #17     } set next scan line interupt
         347 5      STA B 5,X     }
         306 7      LDA B #7      } set new line #
         347 4      STA B 4,X     }
         326 257    LDA B PLBASE+1} Low order Base Address
         347 1      STA B 1,X     }
         326 256    LDA B PLBASE  } High order Base Address
         347 0      STA B Ø,X     }

         71         RTS           return to Caller
```

16 Jul, 78
ABO

*** Text Update Subroutine ***

③

175310_8

```
UPDTXT   175  0  212     TST  LOCK      is display locked
                                        to incoming data
         47   1          BEQ  D1        yes- branch
         71              RTS            else return

D1       336  254       LDX  PLPNTR    } get current character
         246  0         LDA A 0, X     } count
         336  260       LDX  TXTCBS    } store current count
         247  2         STA A 2, X     } @ current Text control Loc.

         336  246       LDX  CLPNTR    } are we still in same
         234  254       CPX  PLPNTR    } line of text.
         46   1         BNE  D2        No- branch
         71              RTS            yes- return

D2       337  254       STX  PLPNTR    update to next line of text
         336  242       LDX  CLBASE    } update Data Base pointer
         337  256       STX  PLBASE    }
(6706)   275  372  170   JSR  ROTUP     go setup next text line
         71              RTS            return
```

VERT

NOGRP

E1

*** Vertical Retrace Interrupt Handler ***

(1)

175350_8

VERTIN	175	0	216	TST UPDFLAG	is update required ?
	47	132		BEQ RESTART	no - just go restart
	177	0	216	CLR UPDFLAG	doing update
	237	202		STS PGMSTK	save stack pointer
	316	1	370	LDX #DSPSTH+56.	Display stack address for last Text line
	216	1	234	LDS #CLSTP	Control Text line stop control Address
	257	6		STS 6,X	put on stack
	216	1	226	LDS #CLSRT	control Text line start control Address
	257	4		STS 4,X	put on stack
	216	1	220	LDS #TXTSTP	Text Stop Control
	257	2		STS 2,X	put on stack
	175	0	217	TST GRAPH	graphics ON ?
	47	4		BEQ NOGRPH	no - branch
	206	11		LDA A #9.	9 lines with graphic
	40	2		BR E1	
NOGRPH	206	26		LDA A #22	22 lines without graphic
E1	227	211		STA A LINES	Lines of Text !

16 Jul, 78
ARS

```
                226  260        LDA  A  TXTCBS   } Text Control Region Address of
                326  261        LDA  B  CURDSP   } last line of Text
                40   2          BR   EZ          branch to entry

LOOP            11              DEX              } point to preceding stack
                11              DEX              } position

EZ/             247  0          STA  A  0,X      put High order Address on Stack
                347  1          STA  B  1,X      put Low order Address on Stack
                300  6          SUB  B  #6       let CURDSP point to preceding
                                                 Text line
                44   2          BCC  CONT        have we passed bottom of stack?
                                                 NO - cont
                306  212        LDA  B  #212     else load Top of Stack area
CONT            172  0  211      DEC  LINES       are we finished yet?
                46   357        BNE  LOOP        NO - Loop

***  Text portion of Display stack is now Loaded  ***

                175  0  217      TST  GRAPH       is graphic ON?
                46   4          BNE  CONT1       yes - branch

                206  57         LDA  A  #STLT    else only Text scan line for
                                                 starting display
                40   20         BR   CONT2       branch

***  Set up graphics in Display stack  ***
```

```
CONT1   11          DEX
        11          DEX
        216 1 250   LDS #GRPSTP     } put Address of GRAPH Stop on Display stack
        257 0       STS Ø,X
        11          DEX
        11          DEX
        216 1 242   LDS #GRPSRT     } put Address of Graph Start on Display stack
        257 0       STS Ø,X

        206 17      LDAA#STLG       starting line in graphic
CONT2   227 210     STA A SRTLIN    store at start line
        337 206     STX PMSTK       save beginning of Display Stack
        236 202     LDS PGMSTK      restore stack
```

*** Restart Display for Current Scan ***

```
RESTART 175 0 217   TST GRAPH       is graphics on ?
        47 16       BEQ CONT3       NO - branch
        175 0 15    TST $D          is this an ODD frame
        52 4        BPL EVEN        even - branch
        336 274     LDX ODDSCN      odd scan Address
        40 2        BR ODD
EVEN    336 272     LDX EVNSCN      even scan address
ODD     377 1 242   STX GRPSRT      set up graph start
```

16 July 78
BBD

```
CONT3   336 206      LDX  PMSTH   } reset current DSP STACH
        337 204      STX  CRSTK   }

(7274)  275 372 130  JSR  DSP UPD   go set up Display Data
                                    for first Test interrupt

        226 210      LDA A SRTLIN } set interrupt for Starting
        227 11       STA A $9     }
        226 5        LDA A $5     clear interrupt (dummy load)
        226 14       LDA A $C     NMI strobe (dummy load)

(7322)  275 370 340  JSR  TIMCLK  Go Do Time update
        336 200      LDX  CURLOC } set cursor position
        337 4        STX  CURSOR } only during restore
```

*** PATCHES FOR Code to Executed At Retrace ***

377 377 376
377 377 376
377 377 376
377 377 376

```
73      RTI   Return from Interrupt
```

(12) *** BREAK KEY SETUP - LINE/LOCAL CONTROL ***

175600_8

```
BRKSET   17              SEI              hold interrupts
(7402)   316 373 230     LDX  #BRKINT     } set up Break Key
         377 17  374     STX  IRQ1S (#FFC)} interrupt Vector
         206 7           LDA A #7         } turn on interrupt
         227 62          STA A #32        }
         226 60          LDA A #30        clear pending interrupt

         177 0 220       CLR  ONLINE      make 'Local'
         16              CLI              allow interrupts
         71              RTS              Return to caller
```

⑯ *** BREAK KEY Interrupt Handler ***

175630_8

```
BRKINT   316 40 10       LDX  40 10       } address for control word
         337 360         STX  #F0         } "TO"
         206 7           LDA A #7         } # of characters
         227 364         STA A #F4        } to transfer
```

16 Jul 78

```
                163 0 220        COM   ONLINE
                47  7            BEQ   X1        Now LOCAL - branch

(7514)          316 373 300      LDX   #TXT1     'ON LINE'
                227 214          STA A RFLAG     Indicate Data was inhibited
                40  12           BR    X2

(7532)
X1              316 373 310      LDX   #TXT2     '_LOCAL_'
                206 5            LDA A #5        } inhibit ready flag
                227 32           STA A $1A       } to POP-11
                177 0 214        CLR   RFLAG     donot allow clock to
                                                 re enable Flag
X2              337 362          STX   $F2       FROM Address
(7562)          275 373 320      JSR   FILL      go transfer characters
                226 60           LDA A $30       clear interrupt Flag
                73               RTI             return from interrupt
```

①

175700_8

117	116	40		TXT1	'ON_LINE'
114	111	116	105		

②

175710_8

40	114	117		TXT2	'_LOCAL_'
103	101	114	40		

③

175720_8

FILL	336	362	LDX $F2	'FROM' Addres
	246	0	LDA A 0,X	get character
	10		INX	} update 'FROM' Addres
	337	362	STX $F2	
	336	360	LDX $F0	'TO' Addres
	247	0	STA A 0,X	store character
	10		INX	} updat 'TO' Addres
	337	360	STX $F0	
	172	0364	DEC $F4	finished ?
	46	355	BNE FILL	NO – go to Fill
	71		RTS	return to Caller

16Jul78
AM

(4) *** UNIBUS DATAIN HANDLER Setup ***

175750_8

DATSET	17		SEI	hold interrupts
(7722)	316	374	10	LDX # TERMDI } set up interrupt
	377	17	352	STX IRQ6 = ($FEA) } Vector
	177	0	32	CLR $1A clear Control Register
	177	0	30	CLR $18 set for input
	206	55		LDA A #20 set to access data, interrupt when data written into Buffer
	227	32		STA A $1A set control
	226	30		LDA A $18 clear pending interrupt (dummy Load)
	206	3		LDA A #3 } set for 180 char/sec
	227	221		STA A NALL }
	16			CLI enable interrupts
	71			RTS return to Calle

(10034)

① *** TERMINAL INTERRUPT HANDLER (INPUT) ***

176010_8

TERMDI	206	5	LDA A #5	} inhibit printer ready flag
	227	32	STA A $1A	}
	226	30	LDA A $18	get character in A reg
(1034)	275	370	120	JSR CTLP	go process character
	174	0	222	INC NUMB	increment counter for # of characters this 60th
	117			CLR A	
	175	0	220	TST ONLINE	are we online ?
	47	17		BEQ. TEND	No – branch
	226	221		LDA A NALL	} are we at limit of characters per 60th
	221	222		CMP A NUMB	}
	57	7		BLE CA	yes – branch
	206	55		LDA A #$20	} else set control to strobe printer flag & Allow more characters
	227	32		STA A $1A	}
	226	30		LDA A $18	set ready Flag
	73			RTI	return from interrupt
CA	206	377		LDA #$FF	printer inhibited
TEND	227	214		STA A RFLAG	set Flag for Clock
	73			RTI	return from interrupt

16 Jul 78
ABB

④ *** KRBP Control Set UP ***

176060_8

KBSET	17			SEI	Hold interrupts
(10142)	316	374	234	LDX #RPINT	} Repeat Button interrupt vector
	377	17	376	STX IRQ16 (=$FFE)	
(10156)	316	374	236	LDX #KBINT	} Keyboard interrupt vector
	377	17	350	STX IRQ5 (=$FE8)	
(10172)	316	374	230	LDX #PHRDR	} photoreader interrupt vector
	377	17	344	STX IRQ3 (=$FE4)	

*** Set up Keyboard ***

177	0	36	CLR $1E	Clear Keyboard Control
177	0	34	CLR $1C	Set for input
206	7		LDA A #7	} set to interrupt on every character
227	36		STA A $1E	
226	34		LDA A $1C	clear pending interrupt

*** Set up Bell ***

177	0	37	CLR $1F	clear switch control
177	0	35	CLR $1D	Set to Read Switches
206	54		LDA $2C	} set to strobe Bell on dummy write to $1D
227	37		STA A $1F	

*** Set Up Repeat Button ***

```
206 370        LDA A #$F8   } mask bits 0,1, & 2
224  63        AND A  $33   }
212   5        ORA A  #5    OR in Repeat Control
227  63        STA A  $33   Set Control Register
226  61        LDA A  $31   clear any pending interrupt
```

*** Set Up BUS output Control ***

```
177 0 33       CLR  $1B     Clear BUS0 control Register
206  377       LDA A #377   } set direction for output
227   31       STA A $19    }
206   54       LDA A #$2C   } Ready Flag strobe-control
227   33       STA A $1B    }
```

*** Photo Reader Set up ***

```
177  0  46     CLR $26      clear photoreader control
177  0  44     CLR $24      set direction in
206   7        LDA A #7     } set interrupt control
227  46        STA A $26    }
226  44        LDA A $24    clear pending interrupt
```

16 Jul 78
ABD

*** MASKING VARIABLES ***

```
206  0      LDA A  #Ø      }
227  230    STA A  IBIS    }  OR'd with incoming data

206  177    LDA A  #177    }
227  231    STA A  IBIC    }  And'd with incoming data

206  200    LDA A  #200    }
227  232    STA A  OBIS    }  OR'd with outgoing data

206  377    LDA A  #377    }
227  233    STA A  OBIC    }  And'd with outgoing data

177  0  234  CLR    DCTL      allow Control Character
                              printing.

16           CLI    enable interrupts
71           RTS    return to Caller
```

(9) *** KRP interrupt Handler ***

176230_8

PHRDR	226	44	LDA A $24	get character from reader auto clear interrupt flag
	40	4	BR IN	
RPINT	226	61	LDA A $31	clear repeat interrupt flag
KBINT	226	34	LDA A $1C	get Keyboard Data clear interrupt flag
IN	175	Ø 220	TST ONLINE	
	46	4	BNE DUMP	
(10512)	275	370 120	JSR CTLP	
	73		RTI	return from interrupt
DUMP	232	232	ORA A OBIS	or to set a bit
	224	233	AND A OBIC	And to clear a bit
	227	31	STA A $19	send character to PDP-11 auto set ready flag
	73		RTI	return from interrupt

16 Jul, 78
ADD

* * * CHAR ROUTINE * * *

176270₈

(See patch 19Sept78 pg 39)

CHAR	201	15	CMP A #(CR)	is character a CR ?
	46	11	BNE DA	if not - branch
	206	1	LDA A #1	get a count of 1 as position
	227	215	STA A CURCNT	
	336	242	LDX CLBASE	position Cursor
	337	200	STX CURLOC	
	71		RTS	return to caller
DA	201	12	CMP A #(LF)	is character a LF ?
	46	13	BNE DB	if not - branch
	275	375	100 JSR UPDLIN	reposition line
	226	215	LDA A CURCNT	get current count
	336	240	LDX CLPNTR	get CLPNTR Address
	247	0	STA A 0,X	set line length
	40	75	BR DX	go finish !
DB	201	7	CMP A #(BL)	is character a BL ?
	46	2	BNE DC	if not - branch
	227	35	STA A $1D	ring bell !!!

DC	175 ø 234		TST	DCTL	print controls ?
	47 5		BEQ	DD	yes - Branch
	221 234		CMP A	DCTL	do we print character ?
	44 1		BCC	DD	Yes - Branch
	71		RTS		if NO - return/finished
DD	336 200		LDX	'CURLOC	load address of Text Cursor
	247 ø		STA A	ø, X	put new character in position
	10		INX		update cursor position
	337 200		STX	CURLOC	
	306 40		LDA B	#(sp)	place a sp at the Cursor position
	347 ø		STA B	ø, X	
	174 0 215		INC	CURCNT	update CURRent count
	226 215		LDA	CURCNT	load count
	336 240		LDX	CLPNTR	get address of line length
	241 ø		CMP A	ø, X	is CURCNT > (CLPNTR)
	55 2		BLT	DE	NO - branch
	247 ø		STA A	ø, X	else update CLPNTR

16 Jul, 78
ADS

DE	206 121	LDA A # 81.	} are we past 80 characters ?	
	241 ∅	CMP A ∅,X		
	56 262	BGT PATCH	if not go finish	
	227 35	STA A $1D	else ring bell!!!	
	247 ∅	STA A ∅,X	and set count = 81.	
(11014)	275 375 100	JSR UPDLIN	reposition line	
	336 240	LDX CLPNTR	get address of new line character count	
	206 11	LDA A # 11.	} set line length = 11.	
	247 0	STA A ∅,X		
	227 215	STA CURCNT		
(11042) DX	275 375 60	JSR CLEAR	flush new line	
(11050)	275 375 40	JSR POSCUR	Reposition Cursor	
(11056)	275 372 310	JSR UPDTXT	update text from retrace	
	174 0 216	INC UPDFLAG	update screen during next retrace	
	71	RTS	return to Caller	
10550 PATCH	275 372 310	JSR UPDTXT	update text - no screen update	
	71	RTS		

④
1764
POSC

①
1764
CLEA

BA

*** POSITION CURSOR SUBROUTINE ***

(4)

176440_8

POSCUR	336 240	LDX CLPNTR	get address of current line length	
	246 0	LDA A Ø,X	get # of characters	
	112	DEC A	delete CURSOR position	
	137	CLR B	}	compute new position
	233 243	ADD A CLBASE+1	}	
	331 242	ADC B CLBASE	}	
	227 201	STA A CURLOC+1	}	set Cursor at position
	327 200	STA B CURLOC	}	
	71	RTS	return to caller	

(1) *** Clear characters Subroutine ***

176460_8

CLEAR	336 240	LDX CLPNTR	}	get character count of
	246 Ø	LDA A Ø,X	}	current line
	336 242	LDX CLBASE		get address of first character in line
	306 40	LDA B #sp		get a sp character
BA	347 Ø	STA B Ø,X		'clear' first Location
	10	INX		Next position
	112	DEC A		more characters ?
	46 372	BNE BA		yes - branch
	71	RTS		else return to Caller

16 Jul 78
AGS

① *** UPDate LINE Pointers Subroutine ***

176 500

UPDLIN	336	240	LDX CLPNTR	} get # of characters in current line
	246	Ø	LDA A Ø,X	
	201	1	CMP A #1	is count = 1
	47	3	BEQ AA	yes - branch always need 1 character
	152	Ø	DEC Ø,X	} else delete cursor space!
	112		DEC A	
AA	234	246	CPX CLPTOP	are we at top?
	47	3	BEQ AB	if yes - branch
	10		INX	else increment pointer
	40	2	BR AC	
AB	336	244	LDX CLPBOT	set to bottom
AC	337	240	STX CLPNTR	update counter stack

AD

* * * Check for character storage limit * * *

```
336 242        LDX  CLBASE      present base in X Reg
137            CLR B           }
233 243        ADD A CLBASE+1   } compute new
331 242        ADC B CLBASE     }   CLBASE
227 243        STA A CLBASE+1   } update CLBASE
327 242        STA B CLBASE     }
220 253        SUB A CLTOP+1    } are we past top
322 252        SBC B CLTOP      }   of character Area ?
45  6          BCS  AD          if not - branch & return

337 276        STX  OVBASE      else note last Base
                                  before going over top !!
336 250        LDX  CLBOT       } Reset CLBASE to Bottom
337 242        STX  CLBASE      }
AD  71         RTS              Return to Caller
```

16 July 78
ARD

② *** SCAN BUTTON Setup ***

176560_8

SETSCN	17	SEI	hold interrupts
(11342)	316 375 220	LDX #SCRLUP	} Scroll up interrupt vector
	377 17 366	STX IRQ1L (=$FFF6)	
(11356)	316 375 370	LDX #SCRLDN	} Scroll Down interrupt vector
	377 17 364	STX IRQ1L (=$FFF4)	
	206 5	LDA A #5	} enable interrupts
	227 52	STA A $2A	
	227 53	STA A $2B	
	226 50	LDA A $28	} clear pending interrupts
	226 51	LDA A $29	
	16	CLI	enable interrupts
	71	RTS	Return to Caller

(11504)

(11512)

(11520)

BOTH

11536

SAM

EXI

④ *** TAPE → Interrupt Handler ***

176620_8

SCRLUP	226	51	LDA $29	clear interrupt flag
	226	35	LDA $1D	read Keyboard switches
	205	40	BIT A *40	is Tape ⊢ depressed
	46	25	BNE BOTH	if yes - branch
	336	254	LDX PLPNTR	PLPNTR = CLPNTR ?
	234	240	CPX CLPNTR	
	47	21	BEQ SAME	if yes - branch
	206	377	LDA 'A # 377	unlock Display
	227	212	STA A LOCH	
(11504)	275	375	270	JSR UPSCRL	go set up for up scrolling
(11512)	275	375	320	JSR RLPUP	roll pointers up 1 line
(11520)	275	372	170	JSR ROTUP	rotate Display 1 line
	40	5		BR EXIT	
BOTH	226	50		LDA $28	clear interrupt flag
11536 SAME	275	376	140	JSR LCHDSP	go Lock Display
EXIT	174	0	216	INC UPDFLAG	update Display during next retrace
	73			RTI	Return from Interrupt

16 Jul 78
ARS

② *** UP Scroll Setup ***

176670_8

UPSCRL	175 0 226	TST DRCTN		BX1
	52 15	BPL UPDONE	+ = UP, − = DOWN; if up – we're done	BX2
	177 0 226	CLR DRCTN	else make (UP)	
	206 27	LDA A #23.	set counter = 23	
LOOPY	66	PSH A	save counter	
(11606)	275 375 320	JSR RLPUP	go roll pointers up 1 line	
	62	PUL A	get counter	
	112	DEC A	are we done?	
	56 370	BGT LOOPY	No – Loop	
UPDONE	71	RTS	Return to Caller	

③ *** ROLL Pointers UP 1 line Subroutine ***

176720_8

RLPUP	336 254	LDX PLPNTR	} get character count in current line	BX3
	246 0	LDA A 0,X	}	
	234 246	CPX CLPTOP	are we at Top	
	46 4	BNE BX1	if not – branch	
	336 244	LDX CLPBOT	else get bottom Location	
	40 1	BR BX2	branch	

```
BX1     10          INX          point to next line
BX2     337 254     STX PLPNTR   update pointer

        137         CLR B        current count = (B,A)
        233 257     ADD A PLBASE+1 }
        331 256     ADC B PLBASE   } compute and
        227 257     STA A PLBASE+1 } save new BASE
        327 256     STA B PLBASE   }

        226 253     LDA A CLTOP+1  }
        326 252     LDB B CLTOP    } compare
        220 257     SUB A PLBASE+1 } is PLBASE ≤ CLTOP
        322 256     SBC B PLBASE   }
        44  4       BCC BX3      yes - branch

        336 250     LDX CLBOT    } clear out
        337 256     STX PLBASE   } PLBASE

BX3     71          RTS          Return to Caller
```

16 July, 78
ARD

*** TAPE ← Interrupt Handler ***

176770_8

```
SCRLDN  226  50          LDA A  $28    clear interrupt flag
        226  35          LDA A  $1D    read Keyboard switches
        205  20          BIT A  #20    is Tape→ depressed
        46   17          BNE    BOTH   if yes - Branch

        206  377         LDA A  #377   } UNLock display
        227  212         STA A  LOCK   }

(12010) 275  376  30  :  JSR    DNSCRL  go setup for down scrolling
(12016) 275  376  60     JSR    RLPDN   roll pointers down 1 line
(12024) 275  372  220    JSR    ROTDWN  rotate display down 1 line
        40   5           BR     EXIT    branch

BOTH    226  51          LDA A  $29    clear interrupt Flag
(12042) 275  376  140    JSR    LCKDSP  go Lock display
EXIT    174  0    216    INC    UPDFLAG update display during next retrace
        73               RTI            Return from Interrupt
```

⑥ *** DOWN SCROLL Setup ***

1770 30₈

DNSCRL	175 Ø 226	TST DRCTN	+ = UP − = DOWN
	53 16	BMI DNDONE	if − we're Done
	206 377	LDA A #377	} set direction Down
	227 226	STA A DRCTN	
	206 27	LDA A #23.	set counter at 23
LOOPX	66	PSH A	save counter
(12110)	275 376 60	JSR RLPDN	Roll Pointer 1 line Down
	62	PUL A	get counter
	112	DEC A	are we finished
	56 370	BGT LOOPX	NO − Loop
DNDONE	71	RTS	Return to Caller

⑭ *** ROLL Pointers Down 1 line Subroutine ***

177 060₈

RLPDN	336 254	LDX PLPNTR	} is pointer at bottom ?
	234 244	CPX CLPBOT	
	46 4	BNE AX1	NO − branch
	336 246	LDX CLPTOP	get top
	40 1	BR AX2	branch

16 Jul 78
AM

```
AX1     11              DEX             point to previous line
AX2     337  254        STX  PLPNTR     update PLPNTR

        226  251        LDA  A  CLBOT+1 }
        326  250        LDA  B  CLBOT   }  CLBOT - PLBASE
        220  257        SUB  A  PLBASE+1}  is PLBASE > CLBOT
        322  256        SBC  B  PLBASE  }
        45   5          BCS  AX3        yes - branch
        336  276        LDX  OVBASE     } else reset PLBASE
        337  256        STX  PLBASE     } to Highest Base Last Calculated
        71              RTS             Return to Caller

AX3     226  257        LDA  A  PLBASE+1}
        326  256        LDA  B  PLBASE  }
        336  254        LDX  PLPNTR     }  compute previous
        240  0          SUB  A  0,X     }  Base Address
        302  0          SBC  B  #0      }
        227  257        STA  A  PLBASE+1} update PLBASE
        327  256        STA  B  PLBASE  }
        71              RTS             Return to Caller
```

⑤ *** LOCK DISPLAY SUBROUTINE ***

177140₈

LCKDSP	177 0 212	CLR LOCK	set for locked Display
	336 240	LDX CLPNTR	} Set for current pointer
	337 254	STX PLPNTR	
	336 242	LDX CLBASE	} Set for current Base
	337 256	STX PLBASE	
	177 0 226	CLR DRCTN	Set for scroll up
(12334)	275 376 30	JSR DNSCRL	DOWN scroll to 23rd previous line
	177 0 226	CLR DRCTN	set for Scroll up
	206 27	LDA A #23.	setup for 23 updates
LOOPZ	66	PSH A	save counter
(12356)	275 372 170	JSR ROTUP	rotate Display 1 line
(12364)	275 375 320	JSR RLPUP	roll pointers 1 line
	62	PUL A	get counter
	112	DEC A	are we finished
	56 365	BGT LOOPZ	no — Loop
12402	275 372 170	JSR ROTUP	update Last line
	71	RTS	Return to Caller

③ 177 210

16 July 78
AMD

* * * SYSTEM ID * * *

177 220₈

(12420) SYST	316	376	240	LDX	#SYSTD	point to character string
(12426)	275	376	220	JSR	PRINT	go print string
	77			RTS		

* * * PRINT Subroutine * * *

$177\ 220_8$

PRINT	337	342	STX	TEMPX	save pointer
	246	0	LDA	0,X	get character
	47	10	BEQ	DONE	if = 0 Done
(12454)	275	374	270	JSR	CHAR	go process character
	336	342	INX	TEMPX	} update pointer
	10		INX		
	40	362	BR	PRINT	go do another
DONE	71		RTS		return to caller

* * * ID Character Strings * * *

$177\ 240_8$

SYSTD /^{CR} ^{LF} ^{SP} 16 ^{SP} JULY ^{SP} 1978 ^{SP} – ^{SP} TEXT ^{SP}
DISPLAY ^{SP} ROM ^{CR} ^{LF} * ^{NU}/

	15	12	40	61	66	40	112	125	114	131
	40	61	71	67	70	40	55	40	124	105
130	124	40	104	111	123	120	114	101	131	40
	122	117	115	15	12	52	0			

$177\ 310_8$

19 July 78
AM

* * * Reset Subroutine * * *

177320_8

BEGIN	177	036	CLR $1E	Clear KYBD Control Reg.
	177	034	CLR $1C	Set Direction for Input
	206	6	LDA A #6	load 6
	227	36	STA A $1E	Set Access for data
	177	037	CLR $1F	Clear Switch Control Register
	177	035	CLR $1D	Set for Input
	206	54	LDA A #$2C	set Bell Control
	227	37	STA A $1F	
TEST	175	036	TST $1E	character typed ?
	52	373	BPL TEST	No – Test again
	226	34	LDA A $1C	get character
	201	114	CMP A #'L	is it an 'L'
	47	23	BEQ LOAD	if yes – go load memory
	201	107	CMP A #'G	is it an 'G'
	47	11	BEQ GO	if yes – go start ROM
	201	123	CMP A #'S	is it an 'S'
	47	10	BEQ START	if yes – go start in RAM
BELLRING	177	035	CLR $1D	else ring 'Bell'
	40	350	BR TEST	

```
GO      176 370 0   JMP   370 0      start top ROM
START   176   4 0   JMP     4 0      start Program in RAM
***     Set UP NPR  Logic            ***
LOAD    177   0 47  CLR   $27        clear NPR Control Reg
        206  37     LDA A #$1F  }    set Direction of Control
        227  45     STA A  $25  }
        206  54     LDA A #$2C  }    Set control for
        227  47     STA A  $27  }    CB2 strobe on write
        216  0 : 0  LDS   #$0        put zero in stack
        316 377 377 LD X  #$FFFF     all ones in X
        237  52     STS   $2A        clear NPR Address Cont.
        337  50     STX   $28        set for Output
        237  56     STS   $2E        clear Data Out Cntr
        337  54     STX   $2C        set for output
        237  62     STS   $32        Clear Data IN Cont.
        237  60     STS   $30        Set for Input

        316   4   4 LDX #$0404       Set up for Data Access
        337  52     STX  $2A  }
        337  56     STX  $2E  }      set control registers
        337  62     STX  $32  }
```

19 Jul 78
ABD

```
57          216  1  376     LDS  #$1FE      PDP-11 Address in S
            177  0  45      CLR  $25        NPR control Reg
            306  1           LDA B #1        get a 1
67          316  0  377     LDX  #$FF       internal Address
60
LOOP        10              INX             increment index address
            214  20  0       CPX  #$1000     at end of 4K ?
            47   276        BEQ  BELLRING   go ring bell if yes
            61              INS             } increment PDP11 word Address
            61              INS             }
            237  50         STS  $28        update NPR address Reg
            327  45         STA B $25       initiate NPR
AGAIN       175  0  45      TST  $25        finished ?
            52   373        BPL  AGAIN      if not - Test again

            226  61         LDA A $31       get low Byte
            247  0          STA A 0,X       store in RAM
            40   351        BR   LOOP       do another
```

RESTART Vectors

IRQ (177770)	376 320	#BEGIN
SWI 177772	376 320	# BEGIN
NMI (177774)	376 320	#BEGIN
RESTART (177776)	376 320	#BEGIN

Patch — 19 Sept 78 ABD

176270_8

CHAR:	176 377 120	JMP	PCHR:	go to char Patch
	1	NOP		

177520_8

PCHR:	376 350 0	LDX	350 0	} check for Rom 2
	214 255 255	CPX	#255 255	}
	46 5	BNE	PRTRN	if not there - branch
	376 350 10	LDX	350 10	} else jump to Rom 2
	156 0	JMP	0,X	}
177535 PRTRN	201 15	CMP A	#'C_R	test for C_R
	46 3	BNE	$Z	
	176 374 274	JMP	#176274	
$Z	176 374 305	JMP	#176305	

19 July 78
ABD

*** ROM 1 — LOCAL MONITOR ***

(4000)
$170\,000_8$

	255	255	#255 255	'Here' indicator
	360	10	# INIT	init Vector
	360	40	# DATA	data Vector

(4020)
$170\,010_8$

INIT	275	361	326	' JSR SYSTB — Print ROM ID

	376	350	0	LDX 350 Ø } is Next Rom There ?
	214	255	255	CPX #255 255 }
	46	5		BNE NONEX — No - check again
	376	350	2	LDX 350 2 } go to transfer Vector
	156	Ø		JMP Ø,X }
NONEX	40	376		BR NONEX

(4100
170 040

```
DATA    175  0  220      TST  ONLINE      online?
        46   4           BNE  SKIP        if yes- skip

        336  262         LDX  NJSR   }  JSR to designated
        255  0           JSR  0,X    }  Subroutine

SKIP    376  350  0       LDX  350  0  }
        214  255  255     CPX  #255 255 } is Next Rom there?
        46   5           BNE  OUT       if not - done
        376  350  4       LDX  350  4  } else load transfer
        156  0           JMP  0,X    } Vector & Jump

OUT     71                RTS
```

29 Jul, 78
ARB

*** CSRCH - Character Search Routine ***

170 400₈
(5000)

```
CSRCH   326 237        LDA B  K2   }
        327 236        STA B  K1   }  two character shifting buffer
        227 237        STA A  K2   }
        336 236        LDX    K1   }
        214 100 75     CPX  #'@=   }  search for control sequence @=
        47  1          BEQ    FND      if found - branch
        71             RTS             else return

FND     275 361 130    JSR    GETSTR   get a character string
        275 361 60     JSR    CHECK    check end of string
        316 367 240    LDX  #TABLE     table of monitor commands
        275 361 250    JSR    SRCHTB   go search table
        45  18         BCS    MATCH    if carry is set a match was found
        206 77         LDA A  '?       } no match found
        275 374 270    JSR    CHAR     } print '?
        275 361 77     JSR    CHECKX   Terminate sequence will not return here

MATCH   336 342        LDX    TEMAX    }
        356 4          LDX    4,X      }  Branch to vector specified
        158 0          JMP    0,X      }  in the table
```

*** CHECH Subroutine ***

(5140) 170460_8

CHECK	226	340	LDA A	TEMPA	} was last character typed a ^{C}R
	201	15	CMP A	#) ^{C}R	
	46	5	BNE	CHECKY	if not - branch
	175	2 0	TST	STRING	any characters in STRING ?
	47	4	BEQ	CHECKX	if not - branch to terminate
170473_8					
CHECKY	201	12	CMP A	#) ^{L}F	a ^{L}F ?
	46	15	BNE	END	if not - return

(5176) 170477_8

CHECKX	316	361	0	LDX	# CRSCH } Terminate current sequence
	337	262		STX	NJSR
	316	361	115	LDX	# PCL* } print end control
	275	376	220	JSR	PRINT
	62			PUL A	} remove one return level
	62			PUL A	
END	71			RTS	return to caller

PCL*	15	12	/ ^{C}R ^{L}F * ^{N}u /
	52	0	

29 July 78
ARO

*** Get STRING Subroutine ***

```
170530 8
(5260)
GETSTR   62               PULA                  } Save return Address
         227 350          STA  RVEC             } Until completion of
         62               PUL A                 } String Acquisition
         227 351          STA  RVEC+1           }
         316 361  161     LDX  GETS             ] set up transfer Vector
         337 262          STX  NJSR             ]

         316 2    120     LDX  #STRNG+80.         get top of string
LOOP     11               DEX                     Pop pointer
         157 Ø            CLR  Ø,X                clear string
         214 2 Ø          CPX  #STRNG             at bottom yet?
         46 370           BNE  LOOP               if not - go clear more
         337 346          STX  IVX                String pointer
         71               RTS                     return 2 levels back

170561 8
         *** Enter here to load string ***
(5244)
GETS     201 15           CMP A #$ CR             a carriage return?
         47 4             BEQ  END
         201 12           CMP A #$ LF             a Line feed?
         46 7             BNE  CONT               if not - branch
END      174 Ø 341        INC  TEMPB            } else return to line
         336 350          LDX  RVEC             } after load  JSR GETSTR
         156 Ø            JMP  Ø,X              } & inhibit character print
```

CONT

OU

DEL

EXI

```
CONT    336  346        LDX  IVX          pointer to string position
        201  177        CMP A #'DEL      } is this a 'delete' character
        47   13         BEQ  DELETE      }
        247  0          STA A 0,X        else some character
        214  2116       CPX  #STACK+78.   are we at end of buffer ?
        47   3          BEQ  OUT         yes - branch
        10              INX              else update pointer
        337  346        STX  IVX
OUT     71              RTS              Return to caller

DELETE  11              DEX              decrement pointer
        214  1  377      CPX  #STACK-1    are we below bottom
                                          of stack ?
        47   11         BEQ  EXIT        yes - Exit
        337  346        STX  IVX         else update pointer
        157  0          CLR  0,X         delete character
        206  134        LDA A #'\        get character
        275  374  270    JSR  CHAR        print 'Delete' mark
EXIT    174  0  341      INC  TEMPB       inhibit reprint.
        71              RTS              Return to Caller
```

29 July 78

* * * Search Table Subroutine * * *

```
5520
         170550₈
SRCHTB   337 342        STX  TEMPX       save table pointer
         356 ∅          LDX  ∅,X         get first two Bytes
         47  30         BEQ  DONE1       if =∅ are at end of Table
         274 2   0      CPX  STRING      first two characters match?
         46  11         BNE  CONT        no - branch
         336 342        LDX  TEMPX       } get second two Bytes
         356 2          LDX  2,X         }
         274 2   2      CPX  STRING+2    second two characters Match?
         47  14         BEQ  DONE2       yes - found entry - exit
CONT     336 342        LDX  TEMPX       }
         1∅             INX              }
         1∅             INX              }
         1∅             INX              } set table pointer to
         1∅             INX              } next entry
         1∅             INX              }
         1∅             INX              }
         40  342        BR   SRCHTB      go Search some more

DONE1    14             CLC              no match - clear carry
         71             RTS              return to caller

DONE2    15             SEC              found match - set carry
         71             RTS              return to caller
```

(5640) SYS
SYST
(5674) I
SYSTM

(5640) $171\,320_8$ (See patch 19 Sept 78 pg 63)

SYST	316	376	240	LDX	#SYSTD	} Print Text Rom ID
	275	376	220	JSR	PRINT	}
SYSTB	316	361	337	LDX	#SYSTM	} PRINT MONITOR ID
	275	376	220	JSR	PRINT	}
	275	361	77	JSR	CHECKX	Terminate Sequence (will not return Here)

(5676) $171\,337_8$

SYSTM	15	12	40	62	/^{CR} ^{LF} ^{SP} 29 ^{SP} JULY ^{SP} 1978
	71	40	112	125	^{SP}-^{SP} LOCAL ^{SP} MONITOR ^{SP}
	114	131	40	61	ROM ^{CR} ^{LF} *^{NU} /
	71	67	70	40	
	55	40	114	117	
	103	101	114	40	
	115	117	116	111	
	124	117	122	40	
	122	117	115	15	
	12	52	0		

30 July 78
ARS

*** Response Message ***

(6040) 171020_8

```
MSG     316 362  27     LDX  #PRES
        275 376 220     JSR  PRINT
        71              RTS

PRES    15  12          /CR LF * SP >> SP SP NUL/
        52  40
        76  76
        40  40  0
```

LOO

END

*** TIME SUBROUTINE ***

```
TIME    275 362  20     JSR  MSG        Print message
        275 361 130     JSR  GETSTR     get string
        275 361  60     JSR  CHECK      check termination

        316  40  70     LDX  # 40 70    } 'TO' addres
        337 360         STX  360        }
        316   2   0     LDX  # STRING   } 'from' addres
        337 362         STX  362        }
```

```
        206  13          LDA  A  #13₈     allow only 11 characters
        137              CLR  B           character count
LOOP    155  0           TST  0, X        } is this the end of
        47   5           BEQ  END         } the string — go to END
        134              INC  B           one more character
        10               INX              increment pointer
        112              DEC A            more allowed
        46   367         BNE  LOOP        yes — loop
END     327  364         STA B  $F4       save character count
        275  373  320     JSR  FILL        go load time
        275  361  77      JSR  CHECKX      Terminate Sequence
```

29 Jul 78
ASD

* * * CTLC ROUTine * * *

(6260) 171130₈

CTLC	275	362	20	JSR MSG	Print Message
	275	361	130	JSR GETSTR	Get character String
	275	361	60	JSR CHECK	check termination
	266	2	Ø	LDA A STRING	} a Y typed?
	201	131		CMP A #'Y	
	46	3		BNE NO	if not - go to NO
	117			CLR A	clear A
	402			BR END	branch
NO	206	40		LDA A #40	set A = 40
END	227	234		STA A DCTL	update DCTL
	275	361	77	JSR CHECKX	Terminate sequence

* * * Printing Subroutines * * *

(6360) 171170₈

PCL*A	316	362	177	LDX #CL*A	
	275	376	220	JSR PRINT	
	71			RTS	
CL*A	15	12		/CR LF * A NU/	
	52	101	Ø		

(6420)

(6460)

SHFT

(6520)

PBYTE

PWORD

GO

(6420) $171\,210_8$

```
PSP     316  362  217    LDX  #SP
        275  376  220    JSR  PRINT
        71              RTS

SP      40   40         / sp sp sp Nu /
        40   ø
```

* * * PRINT OCTAL VALUE Subroutine * * *

(6460) $171\,230_8$

```
SHFT    170  0    355    ASL  VL      }
        171  ø    354    ROL  VH      } 24 Bit shift
        171  ø    356    ROL  VAL     }
        71              RTS          return to caller
```

(6520) $171\,250_8$

```
PBYTE   206  3           LDA  A #3    count of 3 characters
        40   2           BR   GO      branch go
PWORD   206  6           LDA  A #6    count of 6 characters
GO      356  ø           LDX  ø,X     } get binary value to
        337  354         STX  VH      } convert to octal
        66              PSH  A       Same character count
        177  ø    356    CLR  VAL     clear character shift
        201  6           CMP  A #6    is cnt = 6
        47   3           BEQ  GC      if yes - do 6 characters
```

30 July 78
ARD

3C	275	362	230	JSR SHFT	shift a bit
6C	275	362	230	JSR SHFT	shift a bit
	206	60		LDA A #)Ø	develope a character
	233	356		ADD A VAL	from shifted data
	275	374	270	JSR CHAR	Print character
	62			PUL A	get count
	112			DEC A	decrement count
	47	11		BEQ EXIT	if count = Ø, finished
	66			PSH A	save count
	177	Ø	356	CLR VAL	clear character
	275	362	230	JSR SHFT	shift a bit
	40	346		BR 3C	go finish shifting
EXIT	71			RTS	Return to Caller

(6660)

EVA

LOOP

LOOP

OUT

*** EVALUATE STRING TO OCTAL CONSTANT ***

(6660) 171 330$_8$

```
EVAL    177  0  355     CLR  VL      } clear constant to
        177  0  354     CLR  VH      } zero

LOOP1   246  0          LDA  A  0,X     get first character
        47   25         BEQ  OUT        if = 0 finished with input
        200  60         SUB  A  '0      compute binary value
        306  3          LDA  B  #3      set to shift current value by
                                        3 bits
LOOP2   275  362  230    ;JSR  SHFT      go shift
        132              DEC  B          finished yet?
        46   372         BNE  LOOP2      NO — Loop2

        233  355         ADD  A  VL      }
        331  354         ADC  B  VH      } compute updated
        227  355         STA  A  VL      } value
        327  354         STA  B  VH      }
        10               INX             point to next character
        40   347         BR   LOOP1      go loop1

OUT     71               RTS             Return to Caller
```

30 July 1978
ARB

*** Print Address Routine ***

(6770) 171 374₈

PRINT ADD	275	362 170	JSR	PCL*A	Print Address Prefix
	316	0 344	LDX	#PKX	} Print Word
	275	362 254	JSR	PWORD	
	71		RTS		Return to Caller

*** PRINT DAT ***

(7020) 171 410₈

PRINT DAT	275	362 210	JSR	PSP	Print spaces
	336	344	LDX	PKX	} increment pointer
	10		INX		
	337	344	STX	PKX	
	11		DEX		point to last byte
	275	362 250	JSR	PBYTE	Print Byte
	71		RTS		Return to Caller

*** STRING EVALUATION ***

(7060) 171 430₈

PEVAL	316	2 0	LDX	#STRING	} go evaluate STRing
	275	362 330	JSR	EVAL	
	336	354	LDX	VH	} some evaluation
	337	344	STX	PKX	
	71		RTS		Return to Caller

* * * PEEK SUBROUTINE * * *

(7120) 171450_8

PEEK	275	362	20	JSR MSG	Print input designator
	275	361	130	JSR GETSTR	get Address
	275	361	60	JSR CHECH	check string Termination
	275	363	30	JSR PEVAL	evaluate string
PADD	275	362	374	JSR PRNT ADD	print Address
	206	10		LDA A #10_8	set up a count of 8
LOOP	66			PSH A	save count
	275	363	10	JSR PRNT DAT	go print first Byte
	62			PUL A	decrement counter =0 ?
	112			DEC A	
	46	370		BNE LOOP	if not LOOP
	275	361	130	JSR GETSTR	get character
	275	361	73	JSR CHECKY	check for termination
	40	353		BR PADD	if not go print another 8 Bytes !

30 July 78
ARJ

* * * POKE ROUTINE * * *

(7240) 177520_8

POKE	275	362	20	JSR	MSG	Print input designator
	275	361	130	JSR	GETSTR	get address
	275	361	60	JSR	CHECK	check string termination
	275	363	30	JSR	PEVAL	evaluate Address
AGAIN	275	362	374	JSR	PRINTADD	print Address
	275	362	210	JSR	PSP	print spaces
	275	361	130	JSR	GETSTR	get new data
	275	361	73	JSR	CHECKY	check termination
	316	2	ø	LDX	#STRING	evaluate data
	275	362	330	JSR	EVAL	
		226	355	LDA	A VL	
		336	344	LDX	PKX	'Poke' address !
		175	2 ø	TST	STRING	if STRING ≠ 0
		47	2	BEQ	NOUP	then poke value
		247	ø	STA	A ø, X	into position
NOUP		10		INX		increment position
		337	344	STX	PKX	update pointer
		40	336	BR	AGAIN	go loop Again

7400 BAUD

7460 PCL

7500 PUT

END

```
        * * *   BAUD    ROUTINE    * * *

7400  171 600₈
BAUD    275  362  20     JSR  MSG      print input designator
        275  361  130    JSR  GETSTR   get input string
        275  361  60     JSR  CHECH    check termination
        275  363  30     JSR  PEVAL    evaluate string
        226  355         LDA  A VL     } store evaluation
        227  221         STA  A NALL   } in NALL
        275  361  77     JSR  CHECKX   Terminate Sequence

        * * *   OTHER   PRINTING  ROUTINES    * * *

7460  171 630₈
PCLX    316  361  115    LDX  # PCLX
        275  376  220    JSR  PRINT
        71              RTS

7500    * * *   PUT  Value  Routine    * * *
        171640₈
PUT     175  2  ϕ       TST  STRING   any characters in string?
        47   15         BEQ  LEND     NO - finished
        316  2  0       LDX  # STRING } evaluate string
        275  362  330    JSR  EVAL     }
        336  344         LDX  PKX      } point address to
        11              DEC  X        } correct location
        226  355         LDA  A VL     } place value
        247  ϕ           STA  A ϕ,X    } in location
LEND    71              RTS           Return to Caller
```

3 JULY 78 AM

*** MASH LISTINGS ***

7800 171670$_8$

PIBIS	316	363	340	LDX	#LIBIS	point to print string
	40	15		BR	GO	go
PIBIC	316	363	351	LDX	#LIBIC	point to print string
	40	10		BR	GO	go
POBIS	316	363	362	LDX	#LOBIS	point to print string
	40	3		BR	GO	go
POBIC	316	363	373	LDX	#LOBIC	point to print string
GO	337	346		STX	IVX	save pointer
	275	363	230	JSR'	PCL*	print CL*
	275	362	210	JSR	PSP	space
	336	346		LDX	IVX	} print string
	275	376	220	JSR	PRINT	
	275	363	10	JSR	PRINTDAT	print current date
	275	362	20	JSR	MSG	print input designator
	71			RTS		Return to Caller

LIBIS	40	111	102	111	/ $^{S}_{P}$IBIS $^{S}_{P}$ $^{S}_{P}$ = $^{N}_{U}$ /
	123	40	40	75	ø
LIBIC	40	111	102	111	/ $^{S}_{P}$IBIC $^{S}_{P}$ $^{S}_{P}$ = $^{N}_{U}$ /
	103	40	40	75	ø
LOBIS	40	117	102	111	/ $^{S}_{P}$OBIS $^{S}_{P}$ $^{S}_{P}$ = $^{N}_{U}$ /
	123	40	40	75	ø
LOBIC	40	117	102	111	/ $^{S}_{P}$OBIC $^{S}_{P}$ $^{S}_{P}$ = $^{N}_{U}$ /
	103	40	40	75	ø

10040

MA

*** MASH ROUTINE ***

16040 172020_8

MASH	316	ø 230		LDX	# IBIS	} pointer for PUT Routine
	337	344		STX	PHX	
	275	363	270	JSR	PIBIS	Print sequence for IBIS
	275	361	130	JSR	GETSTR	get string
	275	361	73	JSR	CHECHY	check termination
	275	363	240	JSR	PUT	put data in place
	275	363	275	JSR	PIBIC	print sequence for IBIC
	275	361	130	JSR	GETSTR	get string
	275	361	73	JSR	CHECHY	check Termination
	275	363	240	JSR	PUT	Put data in place
	275	363	302	JSR	POBIS	print sequence for OBIS
	275	361	130	JSR	GETSTR	get string
	275	361	73	JSR	CHECHY	check Termination
	275	363	240	JSR	PUT	put data in place
	275	363	307	JSR	POBIC	print sequence for OBIC
	275	361	130	JSR	GETSTR	get string
	275	361	73	JSR	CHECHY	check Termination
	275	363	240	JSR	PUT	put data in place
	275	361	77	JSR	CHECKX	Terminate sequence

30 July 78
ARD

*** LOADER SUBROUTINES ***

10240 172120_8

ADDR	246	6	LDA A	6,X	mask everything except C1 controls
	204	3	AND A	#3	
	247	6	STA A	6,X	
	347	4	STA B	4,X	set direction for output
	212	74	ORA A	#74	C2 controls + Data Access
	247	6	STA A	6,X	set control
	10		INX		increment pointer
	71		RTS		Return to Caller

DAT

DATO	246	10	LDA A	10,X	mask everything except C1 controls
	204	3	AND A	#3	
	247	10	STA A	10,X	
	347	6	STA B	6,X	set for output
	212	4	ORA A	#4	Data access bit
	247	10	STA A	10,X	set control
	10		INX		increment pointer
	71		RTS		Return to caller

```
DATI    246  12      LDA A  12,X    }
        204   3      AND A  #3      }  most everything except
        247  12      STA A  12,X    }  C1 controls
        157  14      CLR    14,X      set for input
        212   4      ORA A  #4        set to access data
        247  12      STA A  12,X      set control
        10             INX            increment pointer
        71           ; RTS .          Return to Caller
```

30 July 78
ARD

* * * SETUP ALL NPR REGISTERS * * *

10400 172200₈

SETNPR	316	0	44		LDX # $24	get Base Address
	157	3			CLR 3, X	clear Pcontrol
	206	37			LDA A #37	set direction of NPR control
	247	1			STA A 1, X	
	206	56			LDA A #56	Set Pcontrol for data access & CB2 strobe
	247	3			STA A 3, X	
	157	1			CLR 1, X	Preset to NOP
	306	377		;	LDA B #377	for data direction Use
	275	364	120		JSR ADDR	set High order Address Byte
	275	364	120		JSR ADDR	set Low order Address Byte
	275	364	140		JSR DATO	set High order DATO Byte
	275	364	140		JSR DATO	set Low order DATO Byte
	275	364	160		JSR DATI	set High order DATI Byte
	275	364	160		JSR DATI	set Low order DATI Byte
	71				RTS	return to Caller

10520

TRNS

GO

MORE

* * * TRANSFER DATA SUBROUTINE * * *

10520 172250 8

```
TRNSFR   336  344        LDX  PKX   } get # of words to
         356  0          LDX  0,X   } transfer
         46   1          BNE  GO    if = 0 none to transfer
         71               RTS        Return to Caller
                                     transfer Complete

GO       337  225        STX  CNTR  set counter
         336  344        LDX  PKX   } get Transfer Address
         356  2          LDX  2,X   }
         337  50         STX  $28   set NPR Address

         336  344        LDX  PKX
         10              INX        update PKx
         10              INX        to point to first
         10              INX        Data word!
         10              INX
         337  344        STX  PKX

MORE     336  344        LDX  PKX   } get Data word for
         356  0          LDX  0,X   } transfer
         337  54         STX  $2C   Place data at DATO register

         206  5          LDA A #OPR } start NPR operation
         227  45         STA A $25  } will be complete
                                     within 20 μseconds
```

30 July 78
ASD

```
336  344        LDX  PHX   }
10              INX       }  update data pointer
10              INX       }
337  344        STX  PHX   }

336  50         LDX  $28   }
10              INX       }  update NPR Address
10              INX       }
337  50         STX  $28   }

336  225        LDX  CNTR  }
11              DEX       }  Decrement counter
337  225        STX  CNTR  }
46   343        BNE  MORE      ≠0  more to do
40   312        BR   TRANSFER      check for more
```

10720

BDS

LOA

LOO

* * * LOADING PROGRAM * * *

10720 172 350$_8$

BDSK	316	365	40	LDX #BOSKD	Boot Disk Data
	172 353$_8$				
LOAD	337	344		STX PKX	Save pointer
	275	364	200	JSR SETNPR	Set all NPR Registers
	226	53		LDA A $2B	Cause a "Power Fail"
	204	367		AND A #367	
	227	53		STA A $2B	
	316	Ø	Ø	LDX #ØØ	.7 sec wait Loop
LOOP	1Ø			INX	
	46	375		BNE LOOP	
	226	52		LDA A $2A	Enable forced NPR's
	204	367		AND A #367	
	227	52		STA A $2A	
	275	364	250	JSR TRNSFR	go transfer Data
	226	52		LDA A $2A	disable forced NPR's
	212	1Ø		ORA A #1Ø	
	227	52		STA A $2A	
	226	53		LDA A $2B	Terminate 'Power Fail' CPU restarts
	212	1Ø		ORA A #1Ø	
	227	53		STA A $2B	

30 July 78
ARD

	275	365	27	JSR	AHEAD	Push return address on stack
OUT	275	361	77	JSR	CHECKX	Terminate Sequence
AHEAD	7			TPA		get Status register
	66			PSHA		push 5 words on stack
	66			PSH A		
	66			PSH A		
	66			PSH A		
	66			PSH A		
	176	373	230	JMP	BRKINT	BRKINT will return via an RTI to 'OUT'

11100 BDSK

173/27

*** Floppy Disc Bootstrap Loader ***

$172\,440_8$

11100

BDSKL	Ø	2	CNT1 (=2 words)	Power fail Vector Load Data
	Ø	24	PFVECTOR (=24_8)	
	337	300	START Address (=157700_8)	
	Ø	340	Priority (=340_8)	
	Ø	40	CNT2 (=32 words)	Boot Data Loader
	337	300	START Address (=157700_8)	

*** PDP 11 Assembly Language Boot ***

	12	2	UNIT Ø	CLR R2	Default Boot for Unit Ø R2 = 2Ø For Unit 1
	125	302	UNIT 1	BIS #100247, R2	
	200	247			
	25	301	1$	MOV #RXCS, R1	put CSR address in R1
	376	170			
	260	211	2$	BITB R2, @R1	wait for Done
	3	376		BEQ 2$	
	225	303		MOV #7, R3	Set Sector/Track/Loop Counter
	Ø	7			
	20	100		MOV R1, RØ	Set up to point at RDBF
	20	220		MOV R2, (RØ)+	load read & go Function
	1	2		BR 4$	go wait

31 July 78
ARS

```
25   310        3$      MOV  #1, @R0      1st Sector /
                                           2nd Track /
 0     1                                   3rd Track /

14   203        4$      ASR  R3           step thru Sector/Track segment
207    2                BCS  6$           Branch to wait if S/T not done
225  311                MOVB (PC)+, @R1    load 'empty buffer' command
222   23        5$      MOVB @R0, (R3)+    mov data byte to Memory
 60  211        6$      BIT  R2, @R1       wait for TR, DONE, or Error
  3  376                BEQ  6$
201  356                BMI  1$            if error - retry
207  366                BCS  3$            branch if in S/T Loop
213  311                TSTB @R1           is data transfer done ?
201  371                BMI  5$            branch to do more
 12    0                CLR  R0           set to look at A0
 45  310                CMP  #240, @R0     is it = 240_8 ?
  0  240                                   if not go retry
  2  347                BNE  1$
245  302                CMPB #247, R2      sets C = 1 if
  0  247                                     unit 1
 13  100                ADC  R0           R0 = 0  unit 0
 12    7                CLR  PC           R0 = 1  unit 1
                                           go to secondary Boot

          * * *  Termination              * * *

 0     0
```

* * * Cassette Bootstrap Loader * * *

172560_8

11340

BCASD	Ø	2	CNT1 (=2)	Power Fail Vector Load
	Ø	24	PF Address ($=24_8$)	
	337	300	Start Address ($=157700_8$)	
	Ø	340	Priority ($=340_8$)	
	Ø	40	CNT2 ($=40_8$)	
	337	300	START Address ($=157700_8$)	

* * * PDP 11 Assembly language * * *

25	300	CBOOT	MOV #177500, R0	R0 Holds address of TA11
377	100			
12	(0)		CLR (R0)	select unit Ø
21	301	RESTART	MOV PC, R1	use for PIC
145	301		ADD #TABLE-., R1	R1 holds address of Command Table
0	52			
25	302		MOV #375, R2	Memory pointer & Data flag
Ø	375			
224	103		MOVB (R1)+, R3	Move Test Bits to R3
224	110	LOOP1	MOVB (R1)+, (R0)	Mov command from table to TA11
201	13		BMI DONE	if command is negative quit
260	310	LOOP2	BITB R3, (R0)	Test Ready and TR Bits
3	376		BEQ LOOP2	branch if bits not set!

31 July 78
ARD

212	202		INCB	R2	advance memory pointer
201	372		BMI	LOOP1	if minus - try another command
234	12		MOVB	2(R0), (R2)	read Data into memory
0	2				
240	337		CMPB	R3, @#0	First Byte should = 240_8
0	0				
3	367		BEQ	LOOP2	if equal go read another
0	0	STOP	HALT		Halt on error
1	355		BR	RESTART	Restart on Continue
13	310	DONE	TST	(R0)	check for error
201	374		BMI	STOP	Branch to Halt on error
12	7		CLR	PC	go to secondary boot

TABLE:

High Byte		Low Byte
37	240	ILBS + Rewind — Ready + TRansfer Request
5	15	Read + GO — SPACE FORWORD BLOCK + GO
224	24	Read + ILBS — Read + ILBS

* * * CASSETTE ENTRY POINT * * *

11600 172700_8

BOSK		316	365	160	LDX #BOSKD
		176	364	353	JMP LOAD

11640

BABLD

*** Absolute Binary Loader ***

1640 172 720$_8$

BABLD	0	2		CNT1 (=2)
	0	24		PT ADD (=24$_8$)
	377	100		STARTING ADDRESS (=157500)
	0	340		Priority = (340$_8$)
	0	140		CNT2 (=140$_8$)
	377	100		STARTING Address = (157500)

*** PDP-11 Assembly Language ***

```
                ; START of Loader
21   306   L.LD1   MOV  PC, SP          Setup stack
51   246         CMP  -(SP), -(SP)      to start @ L.LD1 -2
21   305         MOV  PC, L.PTR         Get Relocated
145  305         ADD  #L.READ-., L.PTR  Start Address of
                                        Read Routine
0    112
12   1           CLR  L.ADR             Clear the Load Address
27   316   L.LD1B  MOV  @#L.SR, @SP     pick up contents of SR
                                        and Save
377  170
14   16          ROR  @SP               check relocation factor
207  2           BCS  L.LD1C            jump if relocation needed
12   16          CLR  @SP               use tape address
1    3           BR   L.LD2             go do load
```

```
14      316     L.LDBC  ASL  @SP          check for non zero
2       1               BNE  L.LD2       jump if load address
                                          specified in SR
20      116             MOV  L.ADR, @SP  continue loading from
                                          last load

                ; Look for beginning of a Block

12      0       L.LD2   CLR  L.CHSM      initialize checksum
11      315             JSR  PC, @L.PTR  read a frame
212     303             DECB L.BYT       check for +1
2       374             BNE  L.LD2       loop until +1 found
11      315             JSR  PC, @L.PTR  read another frame

                ; input and save byte count, if
                  count = 6  go proms jump

11      367             JSR  PC, L.GWRD  get full Byte count
0       74
21      2               MOV  R4, L.BC
345     302             SUB  #4, L.BC    subtract 4 to
                                          make byte count correct
0       4
45      302             CMP  #2, L.BC    Was byte count = 6
0       2
3       43              BEQ  L.JMP       jump if no data
                                          ie jump block
```

```
11    367                JSR  PC, L.GWRD      get load address
0     54
143   204                ADD  @SP, R4         generate actual Address
21    1                  MOV  R4, L.ADR       and put it into the
                                                proper cell

        ; Read in remainder of Data
          if the Loader Halts @ L.BAD a checksum
          error has occurred, R3 will contain the
          expected checksum, and R0 will contain the
          deviation.

11    315     L.LD3      JSR  PC, @L.PTR      read a frame
        :
4     4                  BGE  L.LD4           branch if more data
                                                remains
213   300                TSTB L.CKSM          if checksum is correct
3     353                BEQ  L.LD2           then continue
0     0       L.BAD      HALT                 checksum error
1     351                BR   L.LD2           press continue to ignore
                                                checksum
220   321     L.LD4      MOVB L.BYT, (L.ADR)+ store 8 Bits at a time
1     370                BR   L.LD3           then Re-Loop

        ; input a frame, decrement byte count,
          and accumulate checksum

35    303     L.READ     MOV  L.DEV, L.BYT    get device address in L.BYT
0     152
212   213                INCB @L.BYT          Select Reader
213   313     L.R1       TSTB @L.BYT          DONE?
200   376                BPL  L.R1            NO - Loop
```

31 July 78
AOD

```
234   303          MOVB  2(L.BYT), L.BYT     get character
 0     2
140   300          ADD   L.BYT, L.CKSM       add to checksum
105   303          BIC   #177400, L.BYT      mask of junk
377    ø
 12   302          DEC   L.BC                decrement Byte
                                               count
 ø    207          RTS   PC                  Return
```

; assemble one full word of Data

```
 25   267  L.GWRD  MOV   (SP)+, L.TMP       Save return in
                                               temporary
 ø    52
 11   315          JSR   PC, @L.PTR          get one character
 20   304          MOV   L.BYT, R4           save R3 in temporary
 11   315          JSR   PC, @L.PTR          get another frame
 ø    303          SWAB  L.BYT               place frame in higher
                                               order Byte
105   303          BIC   #377, L.BYT         and clear lower Byte
 ø    377
120   304          BIS   L.BYT, R4           assemble frames into
                                               one word
 35   307          MOV   L.TMP, PC           Return
 ø    3ø
```

; check correctness of jump addres
Halt if address is ODD, Jump if Addres is even

11	367	L.JMP	JSR	PC, L.GWRD	get possible transfer address
377	346				
11	315		JSR	PC, @L.PTR	get checksum
213	300		TSTB	L.CHSM	if incorrect
2	340		BNE	L.BAD	go to checksum Halt
14	204		ASR	R4	get low order bit
206	2		BCC	L.JMP1	skip if even
ϕ	ϕ		HALT		else Halt
1	276		BR	L.LD1B	return to start of Loading loop
14	304	L.JMP1	ASL	R4	restore register
143	204		ADD	@SP, R4	
0	114		JMP	@R4	jump to User
ϕ	ϕ	L.TMP	.WORD	ϕ	tempory to save stack space

ϕ	ϕ	ϕ
ϕ	ϕ	ϕ
ϕ	ϕ	ϕ
ϕ	ϕ	ϕ
ϕ	ϕ	ϕ
ϕ	ϕ	ϕ

blank - bill's

31 July 78
BDD

*** Bootstrap Loader

12540
BMTD

35	301	START	MOV DEVICE, R1	get device address in R1
0	26			
25	302	LOOP	MOV #, -LOAD+2, R2	get address displacement
0	352			
12	211	ENABLE	INCB @R1	enable paper tape reader
213	311	WAIT	TSTB @R1	wait until
200	376		BPL WAIT	frame is available
234	162	:	MOVB 2(R1), LOAD(R2)	store frame read from tape in memory
0	2			
337	ø			
12	267		INC LOOP+2	increment load address displacement
377	356			
1	365	BRNCH	BR LOOP	go back & read more data
377	160	DEVICE	177560	device address
ø	ø		Termination	

*** ENTRY Point ***

12500 173 240ø

BADL	316	365	320	LDX #BADLD
	176	364	353	JMP LOAD

* ** MAGTAPE LOADER ** *

12540 173 260$_8$

BMTD

	∅	2	CNTL (=2)
	∅	24	PF Addrss (=24$_8$)
	337	76	START Addrss (=157476)
	∅	340	Priority (=340$_8$)

Power Fail Vector Loading

∅	141	CNTL (=141)
337	76	START Addrss (=157476)

*** PDP 11 Assembly Language

∅	∅	HALT
21	306	
13	346	
25	300	
∅	212	
141	200	
25	301	
365	122	
27	346	
377	170	
14	6016	
207	1	

Set up switch register

31 Jul 78
ARO

12	16
14	316
12	11
213	311
200	376
14	61
377	376
206	375
25	311
∅	17
213	311
200	376
∅	∅
27	302
377	170
44	121
20	11
25	341
377	362
25	341
140	3
213	311

200	376
40	210
3	10
25	361
377	377
ϕ	2
25	341
140	11
213	311
200	376
1	356
44	20
13	310
3	17
44	121
30	11
143	211
34	41
ϕ	2
25	341
140	3
213	311

31 July 78
ARB

200	376
13	340
13	311
200	337
22	100
ϕ	ϕ
1	376
30	2
13	340
14	202
207	321
14	302
143	202
ϕ	112
ϕ	ϕ
ϕ	ϕ
ϕ	ϕ
ϕ	ϕ
74	352
ϕ	200
0	22
200	ϕ
0	22

13420
BMT

12	15
11	14
11	11
101	120

0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1
0	1

ϕ ϕ Termination

* * * Entry Point * * *

13420 $173\,610_8$

BMT 316 366 266 LDX # BMTD

176 364 353 JMP LOAD

31 Jul 78

AHD

MONITOR TABLE

13500 173640_8

```
123  131  123  124   TABLE:    / SYST /
     361  320                  # SYST
124  111  115  105             / TIME /
     362  90                   # TIME
103  124  114  103             / CTLC /
     362  130                  # CTLC
120  105  105  113             / PEEK /
     363  50                   # PEEK
120  117  113  105             / POKE /
     363  120                  # POKE
102  101  125  104             / BAUD /
     363  200                  # BAUD
115  101  123  113             / MASH /
     364  20                   # MASH
102  104  123  113             / BOSH /
     364  350                  # BOSH
102  103  101  123             / BCAS /
     365  300                  # BCAS
102  101  102  114             / BABL /
     366  240                  # BABL
```

102 115 124 ∅ / BmT Nu /

367 210 #BmT

122 113 125 123 / RKUS /

220 ∅ #RKUS

31 July 78
AB

Patch — 19 Sept 78 ABD

in SYST: pg 44

171 334

	176	360 100	JMP	SPTCH:	go to Patch

170100$

SPTCH:	376	350 Ø	LDX	350 Ø	} check if Rom 2 is there
	214	255 255	CPX	#255 255	
	46	5	BNE	$PT	if not - skip
	376	350 6	LDX	350 6	} go do syst in other banks
	255	Ø	JSR	Ø,X	
$PT	275	361 77	JSR	CHECKX:	terminate operation will not return here!

PAGE MODE ROM

164000g
350 000

	255	255	#255 255	'Here' indicator
	350	20	# INIT	Init pointer
	350	70	# DATA	datascanning pointer
	350	110	# SYST	ID pointer
	354	320	# PCHK	character processing Series

350 20

INIT:	316	350	340	LDX # GSCAN	} set up scanning routine
	337	264		STX GJSR	
	206	377		LDA A #377	} go initialize to LOCAL
	227	220		STA A ONLINE	
	275	365	27	JSR AHEAD	
	275	352	211	JSR INTIMI:	go set for SCROLL mode
	206	2		LDA A #2	} go init all remaining banks
	275	350	130	JSR BNKTST	
$RPT	40	376		BR $RPT	} endless loop !
	40	374		BR $RPS	

19 Sept 78
AHO

350 70$_8$

350 130

DATA	336 264	LDX GJSR	}	go do designated subroutine
	255 Ø	JSR Ø,X	}	
	206 4	LDA A #4	}	go check all remaining banks
	275 350 130	JSR BNKTST	}	
	71	RTS		return to caller

BNKTST

$LOOP

SYST	275 350 250	JSR PSYST	print ID
	206 6	LDA A #6	} check remaining Banks
	275 350 130	JSR BNKTST	}
	71	RTS	return to caller

*** Bank Testing Routine ***

300 130

BNKTST	66		PSH A	push low order from interVector Address onto stack
	306	340	LDA B #340	push High order address
	67		PSH B	
	117		CLR A	Push Address of Beginning of Rom Bank #3
	66		PSH A	
	67		PSH B	
$LOOP	275	350 210	JSR BLOOP	go check bank
	60		TSX	set stack pointer
	346	0	LDA B 0,X	get bank address
	300	10	SUB B #10	next lower bank
	347	0	STA B 0,X	
	347	2	STA B 2,X	
	53	362	BMI $LOOP	if still in Rom - Loop
	306	4	LDA B #4	set up to check RAM area
	347	0	STA B 0,X	
	347	2	STA B 2,X	
	275	350 210	JSR BLOOP	go check bank
	62		PUL A	clean stack
	62		PUL A	
	62		PUL A	
	62		PUL A	
	71		RTS	return to Caller

```
350 210
BLOOP   60          TSX          get stack pointer
        358 2       LDX 2,X      get Bank Address
        358 Ø       LDX Ø,X      get Rom Indicator
        214 255 255 CPX #255 255 is Rom there?
        46  1Ø      BNE BSKP     if not - skip

        60          TSX          get stack pointer
        358 4       LDX 4,X      get Rom Jump Vector Address
        358 Ø       LDX Ø,X      get Vector
        255 Ø       JSR Ø,X      go to Bank
        71          RTS          Return to Caller

BSKP    306 200     LDA B #200   set to last bank
        60          TSX          set stack pointer
        347 2       STA B 2,X    } store bank address
        347 4       STA B 4,X    }
        71          RTS          Return to calle
```

350 25
PSYST

350 26
ID

* * PSYST ROM ID * * *

350 250

PSYST	316	350	260	LDX #ID	point to text
	275	376	220	JSR PRINT	go print ID
	71			RTS	

350 260

ID	15	12	40	61	/ CR LF SP 15 SP SEPT SP 1978 SP —
	65	40	123	105	SP PAGE SP MODE SP ROM SP CR LF x NA /
	120	124	40	61	
	71	67	70	40	
	55	40	120	101	
	107	105	40	115	
	117	104	105	40	
	122	117	115	40	
	15	12	52	0	

19 Sept 78
ASD

*** SCAN Routine ***

350 340

GSCAN:	326	315	LDA B	ESL	two character shifting Buffer for finding E_c =	$G4
	327	314	STA B	ESH		
	226	340	LDA A	TEMPA		
	227	315	STA A	ESL		
	201	33	CPA A	# E_c	is new character E_c ?	
	46	10	BNE	$G2	if not - branch	
	301	33	CPA B	# E_c	was previous also E_c ?	GCTL
	47	3	BEQ	$G1	if so - leave allowing printing of previous E_c !	
	174	0 341	INC	TEMPB	else inhibit printing of current E_c	
$G1	71		RTS		return to caller	
$G2	301	33	CPA B	# E_c	was previous character E_c	
	47	1	BEQ	$G3	if so - branch	
	71		RTS		else return allowing character to be printed	NCTL
$G3	201	75	CPA A	# =	is new character an = ?	
	47	5	BEQ	$G4	if so - branch	
	27		TBA		else print previous E_c	
	275	374 270	JSR	CHAR	print character	
	71		RTS		return allowing current character to be printed	

```
$G4     174 000 341     INC  TEMPB      inhibit printing of =
        316 351 200     LDX  #GCTL      } set up to transfer to
        337 264         STX  GJSR       } GCTL on next character
        71              RTS             return to caller
```

*** GCTL Routine ***

```
GCTL    226 340         LDA A TEMPA     get character
        316 350 340     LDX  #GSCAN     } reset transfer vector
        337 264         STX  GJSR       }
        200 60          SUB A #'0       check that character
        45  4           BCS  NTCL       is between 0 and
        201 12          CMP A #12       9 inclusively
        45  21          BCS  GOOD
NCTL    226 314         LDA  ESH        } print Ec
        275 374 270     JSR  CHAR       }
        226 315         LDA  ESL        } print =
        275 374 270     JSR  CHAR       }
        177 000 314     CLR  ESH        } clear buffer
        177 000 315     CLR  ESL        }
        71              RTS             return allowing printing
                                        of current character
```

19 sept 78
ASO

```
GOOD    174 0 341       INC TEMPB       inhibit printing
        26              TAB         }
        33              ABA         }   compute 3 * A
        33              ABA         }
        137             CLR B
        213   140       ADD A   #140   }
        311   351       ADC B   #351   }   compute Table Location
        227   317       STA A   GTABLE+1 }
        327   316       STA B   GTABLE  }  same Location
        336   316       LDX     GTABLE      get Location
        246   0         LDA     0,X         get # of characters
                                              needed
        47    15        BEQ     NONE        if none - branch
        227   313       STA A   GCNT        else save count
        316   351 200   LDX   #GETLST   }   set up transfer vector
        337   264       STX     GJSR    }
        316   2 120     LDX   #GSTRNG   }   Buffer for characters
        337   320       STX     GSTADD  }
        71              RTS

NONE    356 1           LDX     1,X         get jump vector
        156 0           JMP     0,X         jump to selected routine
```

164 540

351 140

TBLE:

Ø		Ø, #RAMSET
354	210	
1		1, #GBAUD
352	140	
Ø		Ø, #NTBI
352	150	
Ø		Ø, #LETBI
352	160	
Ø		Ø, #NTMI
352	170	
Ø		Ø, #PMI
352	300	
21		21, #DEFCHR
353	40	
2		2, #MOVCUR
353	130	
Ø		Ø, #DPAGE
353	170	
Ø		Ø, #UPage
353	210	

1980 78
ASO

xxx GETLST Routine xxx

351 20

GETLST	174	0 341		INC TEMPB	inhibit printing
	336	320		LDX GSTADD	get address of Buffer
	226	340		LDA TEMPA	get character
	247	0		STA A 0,X	store character
	10			INX	update Buffer address
	337	320		STX GSTADD	store update
	172	0 313		DEC GCNT	enough characters ?
	57	1	:	BLE. GETDON	if so — branch
	71			RTS	else return and wait for next character
GETDON	316	350 340		LDX #GSCAN	reset jump vector
	337	264		STX GJSR	
	336	316		LDX GTABLE	set table location
	356	1		LDX 1,X	set vector address
	156	0		JMP 0,X	jump to designated routine

GDO:

$PRNT:

$CR:

$LF

$FF

$VT

*** Page Mode character processing Routines ***

```
GDO:    201  40      CPA A #>40      do we have a control character?
        45   7       BCS $CR         if so - branch
$PRNT:  336  200     LDX CURLOC      else place character
        247  0       STA A 0,X       in array
        176  353 305  JMP ADD1:      go update position

$CR:    201  15      CPA A #>CR      Test for CR
        46   3       BNE $LF         if not - branch
        176  353 260  JMP CR:        do carriage return

$LF     201  12      CPA A #>LF      Test for LF
        46   3       BNE $FF         if not - branch
        176  353 272  JMP ADD80:     do a line feed

$FF     201  14      CPA A #>FF      test for FF  form feed
        46   3       BNE $VT         if not - branch
        176  352 300  JMP PMI:       do a new page

$VT     201  13      CPA A #>VT      Test for VT Vertical Tab
        46   3       BNE $HT
        176  353 277  JMP SUB80:     backup one line
```

19 Sept 78
ARO

$HT	201	11		CPA A #'$H_T	Test for H_T Horizontal Tab	$BL
	46	3		BNE $BS	if not - branch	
	176	353	305	JMP ADD1:	space ahead 1 position	
$BS	201	10		CPA A #'B_S	Test for B_S back space	$CTL
	46	3		BNE $RS	if not - branch	
	176	353	312	JMP SUB1:	back up one space	
$RS	201	36		CPA A #'R_S	Test for R_S Home Cursor	$DNE
	46	3		BNE $SUB	if not - branch	
	176	352	20	JMP HOME:	go home cursor	
$SUB	201	32		CPA A #'SUB	Test for S_B Clear Screen	
	46	3		BNE $FS	if not - branch	
	176	352	40	JMP CLR:	go clear screen	
$FS	201	34		CPA A #'F_S	Test for F_S go to previous page	
	46	3		BNE $US	if not - branch	
	176	353	170	JMP DPAGE:	go scroll to previous page	
$US	201	37		CPA A #'US	Test for U_S go to next page	
	46	3		BNE $BL	if not branch	
	176	353	210	JMP UPAGE:	go scroll to next page	

$BL	201	7	CPA A	#$07	Test for ^G Bell
	46	2	BNE	$CTL	if not - branch
	227	35	STA A	$1D	sound Bell
$CTL	221	234	CPA A	XCTL	printing of control characters ?
	45	3	BCS	$DNE	if not - branch
	176	351	244	JMP	$PRNT
$DNE	71			RTS	return to Caller

19 Sept 78
AFD

*** Home Cursor Routine ***

352 20

HOME:	117		CLR	A		clear position in line to beginning of line
	227	326	STA	A	GPCNT	

352 23

HOME2:	117		CLR	A		set character position to Home
	227	325	STA	A	GLCNT+1	
	227	324	STA	A	GLCNT	
	336	322	LDX		GLBASE	reposition cursor
	337	200	STX		CURLOC	
	71		RTS			Return to Caller

352

CLR:

$2CLR

*** Clear Screen Routine ***

352 40

CLR	275	352	20	JSR	HOME	position Cursor at Home
	275	352	113	JSR	POS	compute Address of first character
$LCLR	336	332		LDX	TCLR	} go clear text line
	275	354	161	JSR	CLINE	}
	275	352	100	JSR	NLINE	compute address of next line
	275	354	260	JSR	TMCK	this is a long routine so go check for clock tick
	336	324		LDX	GLCNT	} are we finished
	214	6	100	CPX	#6 100	}
	46	356		BNE	$LCLR	if not - Loop back
	176	352	20	JMP	HOME	go reset pointer to Home

19 sept 78
ARD

*** NLINE Routine ***

```
352 100
NLINE:  206 120      LDA A #120        }
        137          CLR B             }  update GLCNT to
        233 325      ADD A  GLCNT +1   }  Next line
        331 324      ADC B  GLCNT      }
        227 325      STA A  GLCNT +1   }
        327 324      STA B  GLCNT      }

352 113
POS:    316 ø 332    LDX  #TCLR        }  go compute address of
        176 354 13   JMP  CURPOS       }  character - result in TCLR
```

*** BAUD Rate Routine ***

```
352 140
GBAUD   226 340      LDA A TEMPA       get character
        227 221      STA A NALL        set rate
            71       RTS               return to caller
```

*** NORMAL TEXT Buffer Init ***

```
352 150
NTBI    316 57 257   LDX # 57 257      }  set for 4K Text
        337 252      STX  CLTOP        }
        71           RTS               return to caller
```

352 16
ETBI

352 1
NTMI

$A

INTMI

* * * Extended Text Buffer Init * * *

352 160

ETBI	316	177 257	LDX	# 177 257	} setup for 24K Text
	337	252	STX	CLTOP	
	71		RTS		

* * * Normal Text (scroll) mode Initialization * * *

352 170

NTMI	226	310	LDA A	PAGEM	} check if other modes active if not - then already in Text mode
	46	11	BNE	$A	
	226	311	LDA A	PAGEX	
	46	5	BNE	$A	
	226	312	LDA A	PAGEY	
	46	1	BNE	$A	
	71		RTS		return to caller
$A	336	242	LDX	CLBASE	} reset cursor position
	337	200	STX	CURLOC	
INTMI	117		CLR A		} turn off all other modes
	227	310	STA A	PAGEM	
	227	311	STA A	PAGEX	
	227	312	STA A	PAGEY	
	316	352	244	LDX # LIST	} set up to transfer data to Display Control line
	337	362		STX $FE	

20 Sept 78
ARP

352 225

```
DOFL    316  40  20    LDX   # CONTROL    } to Address
        337 360       STX   $F0          }                    $1:

        206  24       LDA A  #24         } set for 20 characters
        227 364       STA A  $F4         }

        176 373 320   JMP    FILL        go fill & return to calle
```

352 244

```
LIST    40 123 103 122      / SP SCROLL SP MODE
        117 114 114  40       SP SP SP SP SP SP SP SP Nu /
        115 117 104 105
        40  40  40  40
        40  40  40  40
        0
```

*** PAGE MODE Initialization ***

352 300

```
PMI     275 375 100   JSR   UPDLIN      terminate current line
        275 372 310   JSR   UPDTXT      update Text control

        336 242       LDX   GLBASE      } same new Base
        337 322       STX   GLBASE      }
```

```
        206  24          LDA A  #24        set for 20 Lines in
                                           new page
$1:     66              PSH A             save counter
        336  240        LDX  CLPNTR       get character count pointer
        206  121        LDA A  #121       } set line length to
        247  0          STA A  0,X        } 81(10) characters
        275  375  60    JSR  CLEAR        go clear line
        275  375  100   JSR  UPDLIN       update Line pointer
        275  372  310   JSR  UPDTXT       update text control
        275  354  260   JSR  TMCK:        go check time
        62              PUL A             get counter
        112             DEC A             more to do ?
        56   351        BGT  $1           if so - loop

        336  240        LDX  CLPNTR       get one extra line
        114             INC A             of 1 character
        227  215        STA A  CURCNT     } set count = 1
        247  0          STA A  0,X        }
        275  375  60    JSR  CLEAR        go clear line
        275  372  310   JSR  UPDTXT       update Text control
        275  352  20    JSR  HOME         position cursor at Home
```

20 Sept 78
ARD

```
206 377        LDA A  #377
227 216        STA A  UPDFLAG    set update flag
227 310        STA A  PAGEM      indicate in Page Mode
117            CLR A             } clear remaining
227 311        STA A  PAGEX      } modes
227 312        STA A  PAGEY
316 353 13     LDX    #PMLST     } point to PMLST
337 362        STX    $F2        }
176 352 225    JMP    DOFL       do fuel & return

353 13
PMLST  40 120 101 107    /sp PAGE sp MODE
       105 40 115 117    sp sp sp sp sp sp sp sp sp sp NL /
       104 105 40 40
       40 40 40 40
       40 40 40 40
       0
```

353
DEFCH

GLOOP

*** Define character Routine ***

353 40

DEFCHR	316	2120	LDX # GSTRNG	get string Address
	246	0	LDA A 0,X	first character is character to be redefined
	10		INX	set pointer to first data character
	337	320	STX GSTADD	
	137		CLR B	clear B
	275	353 114	JSR $*20	go multiply BA by 20_8
	213	0	ADD A #0	add in Base Address for character RAM
	311	20	ADC B #20	
	227	331	STA A GRAMB+1	save address
	327	330	STA B GRAMB	
	206	20	LDA A #20	20_8 Bytes/character
GLOOP	336	320	LDX GSTADD	get address
	346	0	LDA B 0,X	load character
	10		INX	update address
	337	320	STX GSTADD	
	336	330	LDX GRAMB	get RAM Address
	347	0	STA B 0,X	store character
	10		INX	update RAM Address
	337	330	STX GRAMB	
	112		DECA	finished ?
	56	357	BGT GLOOP	if not - Loop 20 Sept 78
	71		RTS	Return to Caller ARO

Selected Multiply Routines

353 114
$*20 110 ASL A } 16 bit shift left
131 ROL B

353 116
$*16 110 ASL A
131 ROL B

353 120
$*4 110 ASL A
131 ROL B

353 122
$*2 110 ASL A
131 ROL B

71 RTS

353 1
DPag
$LPD

MOVE CURSOR

353 130
MOVCUR 316 2 120 LDX #GSTRNG get Address of string
246 0 LDA A 0,X get Row #
137 CLR B
275 353 114 JSR $*20 multiply BA by 20₈
227 325 STA A GLCNT+1 } some product in GLCNT
327 324 STA 0 GLCNT
275 353 120 JSR $*4 get 100₈ * Row

```
233  325        ADD A  GLCNT+1   } compute
331  324        ADC B  GLCNT     }   120₈ × Row

227  325        STA A  GLCNT     } some row Base
327  324        STA B  GLCNT     }

177  0    326   CLR    GRCNT     initialize Column to 0
246  1          LDA A  1,X       get Column location

176  354  370   JMP    PMOV      go to pixel
```

* * * D Page Routine * *

```
353 170
DPage  206  25        LDA A  #25       setup for 21 lines
$LPD   316  375 370   LDX  #DNSCRL     get address of Down scroll routine
       275  353 240   JSR    HANDLER    go do scrolling
       275  354 260   JSR    TMCK       check for tick
       112             DEC    A        more to do?
       56   364       BGT    $LPD       if so - loop until finished
       71              RTS               Return to Caller
```

20 Sept 78
ABD

*** Scroll Up a page Routine ***

353 210

UPAGE	206 25	LDA A #25		set for 21 lines
$LPU	336 240	LDX	CLPNTR	} are we at some
	234 254	CPX	PLPNTR	} line ?
	46 3	BNE	$LAH	if not - go scroll
	176 376 140	JMP	LCHDSP	else Lock Display and return to caller
$LAH	316 375 220	LDX	# UPSCRL	Address of UPSCRL Routine
	275 353 240	JSR	HANDLER	go scroll
	275 354 260	JSR	TMCH	check Time
	112	DEC A		are we done ?
	56 353	BGT	$LPU	if not - loop until finished
	71	RTS		Return to Caller

353 240

HANDLER	66	PSH	A	} dump X reg into
	66	PSH	A	
	66	PSH	A	} push A & B into stack
	67	PSH	B	
	7	TPA		} push CC's
	66	PSH	A	
	156 ø	JMP	ø, X	go to designated handler

will return at location post JSR HANDLER

353 26
CR

353 2
ADD 80

353 27
SUB 80

353 3
ADD 1

353
SUB 1

* * * MOTION HANDLERS * * *

353 260

CR	226	326	LDA A GPCNT	get current column position
	137		CLR B	} set count to Ø
	327	326	STA B GPCNT	
	100		NEG A	negate number
	302	Ø	SBC B #Ø	extend sign of count through B
	40	45	BR UPDGL:	go update GLCNT

353 272

ADD 80	206	120	LDA A #120	} load 80_{10} count
	137		CLR B	
	40	40	BR UPDGL:	go update GLCNT

353 277

SUB 80	206	260	LDA A #260	} load -80_{10} count
	306	377	LDA B #377	
	40	32	BR UPDGL:	go update GLCNT

353 305

ADD 1	206	1	LDA A #1	} load 1 count
	137		CLR B	
	40	3	BR UPDGP	go update GPCNT & GLCNT

353 312

SUB 1	206	377	LDA A #377	} load -1 count
	26		TAB	

20 Sept 78
ARD

353 315

UPDGP1	66		PSH A		save count
	233	326	ADDA	GPCNT	add in count
	52	6	BPL	$U1	if still positive - branch
	213	120	ADDA	#120	else add 120_8
	40	6	BR	$U2	& go to $U2
$UX	200	120	SUB A	#120	
$U1	201	120	CMP A	#120	past 120 ?
	52	372	BPL	$UX	if so - branch
$U2	227	326	STA A	GPCNT	& store result in GPCNT
	62		PUL A		restore count

353 337

UPDGL	233	325	ADD A	GLCNT+1	} add in current position
	331	324	ADC B	GLCNT	
$U3	227	325	STA A	GLCNT+1	} save result
	327	324	STA B	GLCNT	
	52	6	BPL	$U4	if result positive - branch
	213	100	ADD A	#100	} else add in graphic buffer length
	311	6	ADC B	#6	
	40	364	BR	$U3	go save result

$U4

PAST

354 13

CURPOS

$U4	200	100		SUBA #100	} subtract length of Page
	302	6		SBCB #6	
	53	23		BMI PAST	within Page? - yes - branch
	275	354	70	JSR RPAGE	else go rotate Page one line
	226	326		LDA A GPCNT	} current colum position
	137			CLR B	
	213	360		ADD A #360	} add in row position
	311	5		ADC B #5	
	40	342		BR $U3	go store result
PAST	316	ø 200		LDX # CURLOC	get address CURLOC
354 13 CURPOS	226	323		LDA A GLBASE +1	} get Base value
	326	322		LDA B GLBASE	
	233	325		ADDA GLCNT +1	} add in GLCNT
	331	324		ADCB GLCNT	
	247	1		STA A 1, X	} store cursor position
	347	ø		STA B ø, X	

20 Sept 78
ABD

```
200  120      SUBA  #120
302  ϕ        SBCB  #ϕ
220  277      SUBA  DVBASE+1
322  276      SBCB  DVBASE
```

} is position past end of Buffer

```
53   1ϕ       BMI   $END
```

if not - branch

```
233  251      ADDA  CLBOT+1
331  250      ADCB  CLBOT
```

} else add in Base

```
247  1        STAA  1,X
:
347  ϕ        STAB  ϕ,X
```

} same position

```
$END  71      RTS
```

return to Caller

354 7ϕ
RPAGE
$LLP
$65ϕ

*** Rotate Page ***

354 70

RPAGE	275	352	23	JSR HOME2:	position Cursor leave column position unchanged
$LLP	316	∅	330	LDX # GRAMB	compute address of first character in line
	275	354	13	JSR CURPOS	
	275	352	100	JSR NLINE	update pointer to next line
	206	50		LDA A # 50	set up to replace 40₁₀ pairs of characters
$L5∅	66			PSH A	save counter
	336	332		LDX TCLR	get two characters from next line
	246	∅		LDA A ∅,X	
	1∅			INX	
	346	∅		LDA B ∅,X	
	1∅			INX	
	337	332		STX TCLR	
	336	330		LDX GRAMB	store two characters in new line
	247	∅		STA A ∅,X	
	1∅			INX	
	347	∅		STA B ∅,X	
	1∅			INX	
	337	33∅		STX GRAMB	

20 Sept 78
ARO

```
            62          PUL A       get counter
            112         DEC A       more to do
            56  347     BGT  $L50    Loop until line done

            275 354 260 JSR  TMCH   go check time
            336 324     LDX  GLCNT
            214 5 360   CPX  #5 360  to last line?
            46  322     BNE  $LLP   loop until there

            316 0 200   LDX  #CURLOC  } go position Cursor
            275 354 13  JSR  CURPOS   } at beginning of last line

            336 200     LDX  CURLOC   Curr Location
            206 120     LDA A #120    80 characters/line
            306 40      LDA B #40     Space Code

$80CL       347 0       STA B 0,X   }
            10          INX         } clear last line
            112         DEC A       }
            56  372     BGT  $80CL  }

            71          RTS         Return to Caller
```

* * * RAM Character Restoration * * *

354 210

RAMSET	316	17 377	LDX #17 377	} Beginning of Character RAM -1
	337	374	STX $FC	
	316	177 377	LDX #177 377	} Beginning of ROM -1
	337	376	STX $FE	
$LOOP	275	354 260	JSR TMCK	check time
	336	376	LDX $FE	} get byte update pointer
	10		INX	
	337	376	STX $FE	
	246	0	LDA A 0,X	
	336	374	LDX $FC	} store byte update pointer
	14		INX	
	337	374	STX $FC	
	247	0	STA A 0,X	
	214	37 377	CPX #37 377	finished
	416	352	BNE $LOOP	(copy not finished
	71		RTS	return to caller

20 Sept 78
ADD

*** Time Check Routine ***

354 260

TMCH	175	0 42	TST $22	test for IRQ Flag
	53	1	BMI $TM	if no tick then
	71		RTS	return to caller
$TM	66		PSH A	save A
	172	0 223	DEC DVAL	a second yet ?
	46	6	BNE TMCKDN	no - branch
	206	74	LDA A #74	} reset counter
	227	223	STA A DVAL	
	227	213	STA A TFlag	set Sec Flag
TMCKDN	226	40	LDA A $20	clear Clock Flag
	62		PUL A	restor A
	71		RTS	return to Caller

* * * Process character check Handler * * *

354 320

PCHK	175	0 316	TST PAGEM	in Page mode
	47	3	BEQ $K	if not - branch
	176	351 240	JMP BDD	go to page mode Handler
$K	66		PSH A	save A
	206	10	LDA A #10	} check all remaining
	275	350 130	JSR BNKTST	} Banks
	62		PUL A	restore A
	176	377 135	JMP PRTRN	return to CHAR scroll mode

354 350

PCKRTN	61		INS	} Pop return to BLOOP
	61		INS	
	61		INS	} Pop return to BNKTST
	61		INS	
	61		INS	} Pop Bank Test Vectors
	61		INS	
	61		INS	
	61		INS	
	61		INS	} pop Return to PCHK
	61		INS	
	61		INS	pop character
	71		RTS	Return to Caller of CHAR

*** Patch of MovCUR

354 370

PMOV	137		CLR	B	clear B
	66		PSH	A	save A
	52	4	BPL	$B	if positive - branch
$C	200	120	SUBA	#120	else decrease by one line
	53	374	BMI	$C	
$B	176	353	316	JMP	UPDGP+1

Software patches

in CLKO 174200_8

change of clock control Register

174211_8

227 26 STA A $16

226 24 LDA A $14

in CLKINT 174300_8

change of clock control Registers

174331_8

226 24 LDA A $14

20 Mar 1979
AM

Software Changes

in TIMCLK 174340₈

from ABOVE: (changes indicated by *)
corrects Am/pm changes

```
ABOVE:  206  62     LDA A H'2
        241  1      CMP A 1,X
*       56   22     BGT   TIMDON.
*       47   3      BEQ   DYNT
*       40   17     BR    13th Hr
*       Ø           — : —  .
DYNT    206  101    LDA A H'A
        241  9      CMP A 9,X
        47   3      BEQ   NITE
        247  11     STA A 9,X
        71          RTS
NITE    206  120    LDA A H'P
        247  11     STA A 9,X
TIMDON  71          RTS

*13th Hr 347 Ø      STA B Ø,X
*       134         INC B
*       347  1      STA B 1,X
*       71          RTS
```

Key Board Utility System Rom's

110000_8

RHUS	316	220	24	LDX * HUS
	176	364	353	JMP LOAD

110024_8

0	2	CNT1	length of first segment = 2 words
0	24	PF Vector	Address for transfer
320	0	150000_8	HUS starting location
0	340	340_8	Priority at start
7	200	CNT2	length of HUS 3600_8 words
320	0	START	Start address

HUS Text 3600_8 words

0 0 termination

28 Sept 78