

001 PDS-V3N

NAM PDS-V3N  
OPT Q, NOG

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

\*

PROGRAMMED BY ERIC JAMESON

COPYRIGHT 1976 SPHERE CORPORATION  
940 N. 4TH EA., NORTH SALT LAKE, UTAH 84054

SPHERE RESERVES ALL RIGHTS FOR THE REPRODUCTION,  
DISTRIBUTION AND USE OF THE PDS SOFTWARE.  
NO COPIES MAY BE MADE OR DISTRIBUTED WITHOUT THE  
WRITTEN PERMISSION OF SPHERE CORP.

\* THE PROGRAM DEVELOPMENT SYSTEM (PDS V3N) IS A SET OF  
\* PROGRAMS RESIDING ON ERASABLE PROGRAMMABLE READ ONLY  
\* MEMORY WHICH ALLOW EVEN THE SMALLEST USER TO USE HIS  
\* SPHERE SYSTEM AS A COMPLETE COMPUTER SYSTEM FOR THE  
\* DEVELOPMENT OF COMPUTER PROGRAMS.  
\* TOWARD THIS END, THE 4 FDS EPROMS CONTAIN A CURSOR  
\* BASED EDITOR, A MINI-ASSEMBLER, AND THE SPHERE DEBUGGING  
\* RID (SDA), AS WELL AS A SET OF UTILITY ROUTINES TO DO 16  
\* BIT MULTIPLY AND DIVIDE, ASCII-TO-BINARY, AND  
\* BINARY-TO-ASCII ROUTINES.

\* THE PDS-V3N PROM SET WAS WRITTEN IN ORDER TO RUN THE  
\* NEW KEYBOARD. CHANGES WERE MADE IN THE EDITOR AND THE  
\* DEBUGGER. AS THE NEW PROMS ARE A GREAT IMPROVEMENT OVER  
\* THE V3A PROM SET, A VERSION KNOWN AS PDS-V3D WAS MADE  
\* WHICH RUNS ON THE OLD KEYBOARDS. THE ONLY DIFFERENCE  
\* BETWEEN THE V3D AND THE V3N PROMS ARE THAT THE PIA  
\* ADDRESS IS CHANGED FROM F040 ON V3N TO F080 ON V3D.  
\* CHANGES WERE ALSO MADE IN THE DEBUGGER AND THE  
\* EDITOR. THE EDITOR CHANGES WERE THAT INSERT AND DELETE  
\* ARE NOW AT THE TOP OF THE PAGE AND THAT THERE IS A REEDIT  
\* COMMAND IN THE EXEC (CTRL R) TO ALLOW RE-EDITING. THE  
\* ENTRY TO THE DEBUGGER FROM THE BREAKPOINT INSTRUCTION  
\* HAS BEEN CHANGED SO THAT THE RETURN ADDRESS FOR THE  
\* BREAKPOINT IS NOW CALCULATED WHEN THE BREAKPOINT IS  
\* ENCOUNTERED AND GOES TO THE DEBUGGER. IN ADDITION  
\* THERE ARE 2 NEW INSTRUCTIONS: TJ FOR DOING A JSR TO  
\* A ROUTINE AND TX TO EXIT BACK TO THE EXEC.

				MEMORY MAP
3055	*			
3056	*			
3057	0000	TMP	EQU	\$00
0058	0002	TMP1	EQU	\$02
0059	0004	RRB	EQU	\$04
0060	0004	RR3	EQU	\$04
0061	0005	RR2	EQU	\$05
0062	0006	ARA	EQU	\$06
0063	0006	RR1	EQU	\$06
0064	0007	AR0	EQU	\$07
0065	0008	DIGIT	EQU	\$08
0066	000A	OUTEND	EQU	\$0A
0067	000C	BUFAADR	EQU	\$0C
0068	000E	BUFEND	EQU	\$0E
0069	0011	OUTBUF	EQU	\$11
0070	0014	SRCADR	EQU	\$14
0071	0016	DSTADR	EQU	\$16
00072	001A	ENDMEM	EQU	\$1A
00073	001C	CSRPTR	EQU	\$1C
00074	001E	BUFPTR	EQU	\$1E
00075	0020	BUFFLO	EQU	\$20
00076	0022	BUFFHI	EQU	\$22
00077	0024	SCNPTR	EQU	\$24
00078	0026	SRCAASM	EQU	\$26
00079	002A	ONDVAL	EQU	\$2A
00080	002C	SYMVAL	EQU	\$2C
00081	002E	BRKSAY	EQU	\$2E
00082	0030	BRKADR	EQU	\$30
00083	0032	EDIT	EQU	\$32
00084	0035	IOBUFF	EQU	\$35
00085	0040	PCVAL	EQU	\$40
00086	*			
00087	E01F	FRSTLN	EQU	\$E01F
00088	E1E0	LASTLN	EQU	\$E1E0
00089	E1FF	LASTCH	EQU	\$E1FF
00090	F040	KBDPIA	EQU	\$F040
00091		*KEYBOARD PIA ADDRESS FOR OLD KEYBOARD (KBD/1A) IS F000.		
00093	FE71	INPCHR	EQU	\$FE71
00094	FE64	DEBNG	EQU	\$FE64
00095	FF22	ASCBIN	EQU	\$FF22
				INPUTS A CHARACTER.
				DEBUGGER ROUTINE.
				ASCII TO BINARY ROUTINE.

0097 \*  
 0098 \*  
 0099 \*  
 0100 \*  
 0101 \* THE INITIALIZATION ROUTINES SET UP THE INITIAL VALUES  
 0102 \* UPON SYSTEM RESET.  
 0103 \*  
 0104 \*

0105 FC00	ORG	\$FC00	
0106 FC00 8E 01FF RESTRT	LDS	#\$1FF	SETS STACK POINTER
0107 FC03 30	TSX		MOVES STK PTR TO INDEX REG
0108 FC04 DF 26	STX	SRCASM	SETS ASSEMBLR OUTPUT PTR
00109 FC06 DF 0C	STX	BUFAADR	INIT INPUT BUFFR ADDR.
00110 FC08 86 1F	LDA A	#\$1F	INTLZ. KEYBOARD PIA.
00111 FC0A B7 F041	STA A	KBDPIA+1	PIA CONTROL REG. ADDRESS.
00112 FC0D CE 0FFFF	LDX	#\$FFF	LAST LOCATN OF MEMORY.
00113 FC10 DF 0E	STX	BUFEND	INIT END-OF-EDIT BUFFR.
00114 FC12 DF 1A	STX	ENDMEM	INIT END-OF-MEM ADDR.

00115 \*  
 00116 \*  
 00117 \*  
 00118 \*  
 00119 \*  
 0120 \*  
 121 \*  
 00122 \* THIS EXECUTIVE ACCEPTS COMMANDS FROM THE KEYBOARD TO  
 00123 \* DETERMINE WHAT UTILITY IS TO BE RUN. INVALID COMMANDS  
 00124 \* WILL SPACE THE CURSOR DOWN ONE LINE. DO NOT SPACE OFF THE  
 00125 \* BOTTOM OF THE SCREEN.  
 00126 \*  
 00127 \*

00128 FC14 SD 21	EXEC	BSR	HOME	CSRPTR IS HOMED.
00129 FC15 SD 25		BSR	CLEAR	CLEAR SCREEN
00130 FC18 SD 73	EXEC1	BSR	CR1	CRLF FOR NEW LINE.
00131 FC1A BD FE71		JSR	INPCHR	GETS & DISPLAYS CHR.
00132 FC1D S1 01		CMP A	#\$01	TESTS IF ASSEMBLY COMND.
00133 FC1F 26 03		BNE	EXEC2	SKIPS IF OTHER COMMAND.
00134 FC21 BD FDR1		JSR	ASMBLR	JUMPS TO ASSEMBL PRRogram.
00135 FC24 S1 05	EXEC2	CMP A	#\$05	TESTS IF CONTRL 'E'.
00136 FC26 26 02		BNE	EXEC3	SKIPS IF NOT EDIT CMD.
00137 FC28 SD 3D		BSR	EDITOR	JUMPS TO EDIT TEXT.
00138 FC2A S1 12	EXEC3	CMP A	#\$12	TESTS FOR A TR COMND.
00139 FC2C 26 02		BNE	EXEC4	SKIPS IF NOT A REEDIT COMND. 00140 FC2E
SD 3F	BSR	REEDIT		GOES TO REEDIT TEXT.
00141 FC30 S1 04	EXEC4	CMP A	#\$04	TESTS FOR CONTRL 'D'
00142 FC32 26 E4		BNE	EXEC1	SKIPS BACK FOR A NEW COMND.
00143 FC34 7E FE64		JMP	DEBUG	JUMPS TO DEBUGGER.

00145 \*

146 \*

~147 \*

00148 \*

00149 \*

00150 \*

00151 \*

00152 \*

00153 \*

00154 \*

00155 \*

00156 \*

00157 \*

00158 \*

00159 \*

00160 \*

00161 \*

00162 \*

00163 \*

00164 \*

00165 \*

00166 \*

00167 \*

00168 \*

00169 \*

00170 \*

8 71 \*

6..72 \*

00173 \*

00174 \*

00175 \*

00176 \*

00177 \*

00178 \*

00179 \*

00180 \*

00181 \*

00182 \*

00183 \*

00184 \*

00185 \*

00186 \*

00187 \*

00188 \*

00189 \*

00190 \*

10191 \*

00192 \*

0193 \*

0194 \*

0195 \*

~96 \*

~97 \*

0198 \*

0199 \*

0200 \*

0201 \*

## THE EDITOR

\* THE EDITOR ALLOWS INPUT FROM THE KEYBOARD INTO A  
 \* BUFFER MEMORY. INPUT IS DISPLAYED ON THE SCREEN. WHEN  
 \* IT IS TYPED IN, THE SCREEN TEXT CAN THEN BE EDITED BY USE  
 \* OF THE CURSOR. WHEN THE SCREEN IS FULL OR EDITING IS  
 \* FINISHED, THE DATA IS SCROLLED OFF THE SCREEN INTO THE  
 \* EDIT BUFFER. WHEN TEXT IS SCROLLED OFF THE TOP OF THE  
 \* SCREEN, IT IS STORED FROM THE BUFFER ADDRESS POINTER  
 \* (BUFADR) TO THE LOW BUFFER POINTER (BUFFLO). BUFFLO  
 \* POINTS TO THE END OF TEXT + ONE (I.E. IT POINTS TO THE  
 \* FIRST UNUSED BYTE). WHEN IT IS SCROLLED OFF THE BOTTOM  
 \* OF THE SCREEN, IT IS STORED IN THE TOP OF THE EDIT BUFFER.  
 \* THE TEXT GOES FROM THE HIGH BUFFER POINTER (BUFFHI) TO  
 \* THE END OF BUFFER POINTER (BUFEND). BUFFHI POINTS TO  
 \* THE END OF TEXT - ONE (I.E. IT POINTS TO THE LAST UNUSED  
 \* BYTE IN THE BUFFER). WHEN THE TEXT IS SCROLLED UP  
 \* OFF THE TOP OF THE SCREEN, TEXT IS TAKEN FROM THE HIGH  
 \* AREA OF THE EDIT BUFFER AND DISPLAYED ON THE LAST LINE OF  
 \* THE SCREEN. WHEN TEXT IS SCROLLED DOWN OFF THE BOTTOM,  
 \* TEXT, IF ANY EXISTS, IS MOVED FROM THE LOW EDIT BUFFER  
 \* AREA TO THE TOP LINE OF THE SCREEN.

## POINTERS USED

\* CSRPTR POSITION OF CHARACTERS INSERTED ON THE SCREEN.  
 \* SCNPTR POSITION OF START OF EDITED TEXT ON SCREEN.  
 \* BUFADR START OF TEXT BUFFER IN MAIN MEMORY.  
 \* BUFEND END OF TEXT BUFFER IN MAIN MEMORY.  
 \* BUFFLO END OF TEXT SCROLLED OFF TOP OF SCREEN  
 \* BUFFHI START OF TEXT SCROLLED OFF BOTTOM OF SCREEN.  
 \* BUflen NOT CURRENTLY USED.

## EDITOR COMMANDS

\* "UP ARROW" MOVES CURSOR UP ONE LINE; CSRPTR GETS  
 \* CSRPTR-32; CALL NDRFLG.  
 \*  
 \* "DOWN ARROW" MOVES CURSOR DOWN ONE LINE; CSRPTR GETS  
 \* CSRPTR+32; CALL OVRFLG.  
 \*  
 \* "RIGHT ARROW" MOVES CURSOR ONE POSITION RIGHT; CSRPTR  
 \* GETS CSRPTR+1; CALL OVRFLG.

\* "LEFT ARROW" MOVES CURSOR ONE POSITION LEFT; CSRPTR  
\* GETS CSRPTR-1; CALL NDRFLO.  
\*  
\* "CONTROL & LEFT ARROW (ON KEYBOARD)" LEFT JUSTIFY CURSOR;  
\* CSRPTR GETS CSRPTR TRUNCATED; CALL NDRFLO FOR SCNLOC CHK.  
\*  
\* "PUTCHR" OUTPUTS CHARACTER; CSRPTR GETS  
\* CSRPTR+1; GOES TO OVRFLD.  
\*  
\* "ENDCHR" TERMINATION CHAR; CLEAR EDIT FLAG;  
\* EXIT THE EDITOR.  
\*  
\* "HOME" HOMES CURSOR POINTER; CSRPTR GETS E000; NDRFLO.  
\*  
\* "CLEAR" CSRPTR TO END OF THE SCREEN GETS SPACES.  
\*  
\* "CTRL I" INSERT A LINE AT THE LAST LINE ON THE SCREEN;  
\* CALL OVR1 (SCROLLS UP ONE LINE); CSRPTR GETS E1E0.  
\*  
\* "CTRL D" DELETE LAST LINE; SCROLL DOWN (UNDR2);  
\* CSRPTR GETS E1E0.  
\*  
\*  
\*  
\* OVERFLOW CHECKS IF SCROLL UP IS NEEDED; IF IT IS, IT  
\* SCROLLS UP AND MOVES DATA TO & FROM THE BUFFERS.  
\*  
\* OVRFLD: IF CSRPTR < E200 THEN RETURN; IF EDIT IS ON THEN  
\* OVR1: BUFFLO+ GETS SCNPTR TO 'C. R.';  
\* DSTADR GETS CSRPTR GETS E1E0 (LAST LINE ON SCREEN);  
\* IF EDIT IS ON AND BUFFHI < BUFEND THEN MOVE THE TEXT  
\* (THE STRING FROM BUFFHI TO 'C. R.') TO THE LAST LINE.  
\*  
\*  
\*  
\* UNDERFLOW CHECKS IF SCROLL DOWN IS NEEDED AND MOVES  
\* DATA TO AND FROM THE BUFFERS. CURSOR HAD BEEN  
\* MOVED OFF THE TOP OF THE SCREEN AND IS NOW PUT AT THE  
\* HOME POSITION ON THE SCREEN.  
\*  
\* NDRFLO: IF CSRPTR > DFFF THEN RETURN (GO TO OVRFLD);  
\* IF EDIT FLAG IS ON THEN MOVE LAST LINE TO BUFFHI  
\* ON DOWN; SCRLDN; CSRPTR GETS E000; MOVE LINE FROM  
\* BUFFLO TO FIRST LINE ON THE CRT.  
\*  
\*  
\*  
\* NOTE: DON'T SCROLL OFF SCREEN IN EXEC UNTIL AFTER  
\* THE EDITOR HAS BEEN RUN.  
\*  
\* NOTE: EVERY LINE MUST HAVE A C.R. ON IT.

006 PDS-V3N

3	FC37	CE E000	HOME	LDX	#\$E000	LOADS HOME POSITION.
3	FC3A	DF 1C		STX	CSR PTR	STORES HOME IN CURSOR PTR.
3	FC3C	39		RTS		RETURNS TO CALLER.
4	*			*		
4	*			*		
3	FC3D	C6 60	CLEAR	LDA B	#\$60	LOADS BLANK (C. R.).
4	FC3F	CE E200		LDX	#LASTCH+1	LOADS END-OF-SCREEN PTR.
5	FC42	09	CLEAR1	DEX		DECREMENTS BLANKING PTR.
6	FC43	E7 00		STA B	0,X	BLANKS LOCATION.
7	FC45	9C 1C		CPX	CSR PTR	TESTS IF DONE.
8	FC47	26 F9		BNE	CLEAR1	BRANCHES BACK IF NOT DONE.
9	FC49	39		RTS		RETURNS.
10	*			*		
11	*			*		
12	*			*		
13	*		GETCHR	INPUTS A CHARACTER INTO ACC A WITHOUT		
14	*		MOVING THE CURSOR, AND BLINKS THE CURSOR.			
15	*			*		
16	FC4A	DE 1C	GETCHR	LDX	CSR PTR	LOADS CRT CURSOR POSITION.
17	FC4C	63 00		COM	0,X	COMPLEMENT (FLASH POSITION).
18	FC4E	CE 26F0		LDX	#9968	LOADS BLINK COUNT VALUE.
19	FC51	09	GET1	DEX		COUNT GETS COUNT-1.
20	FC52	27 F6		BEQ	GETCHR	RESETS CTR WHEN TIMED OUT.
21	FC54	86 40		LDA A	#\$40	LOADS MASK FOR CR2 FLAG.
22	FC56	B5 F041	← F041	BIT A	KBDPIA+1	TESTS IF A CHAR. TYPED IN.
23	FC59	27 F6	mv > P	BEQ	GET1	BRANCH IF CHAR NOT ENTERED.
24	FC5B	DE 1C		LDX	CSR PTR	LOADS CURSOR POSITION.
25	FC5D	A6 00		LDA A	0,X	TESTS IF BLINKMD (SOLID).
26	FC5F	2A 02		BPL	GET2	SKIPS IF NOT BLINKED.
27	FC61	63 00		COM	0,X	CLEAR THE CHARACTER.
28	FC63	B6 F040	GET2	LDA A	KBDPIA	LOADS A WITH KEYED CHAR.
29	FC66	39		RTS		RETURNS TO CALLER.
30	*			F262 ← V3D		
31	*			*		
32	*			*		
33	*			*		
34	*		EDITOR	IS THE MAIN ENTRY POINT FOR EDITING.		
35	*			*		
36	FC67	DE 0C	EDITOR	LDX	BUFAADR	BUFFLO GETS THE
37	FC69	DF 20		STX	BUFFLO	VALUE OF BUFAADR.
38	FC6B	DE 0E		LDX	BUFEND	BUFFHI GETS THE
39	FC6D	DF 22		STX	BUFFHI	VALUE OF BUFEND.
40	FC6F	8D C6	REEDIT	BSR	HOME	ENTRY POINT FOR
41	FC71	80 CB		BSR	CLEAR	RE-EDITING TEXT.
42	FC73	97 32	EDITRD	STH A	EDIT	TURNS ON EDIT MODE.
43	FC75	DE 1C	EDITIN	LDX	CSR PTR	SETS SCN PTR TO CSR PTR.
44	FC77	DF 24		STX	SCN PTR	
45	*			*		
46	*			*		
47	FC79	8D CF	EDREAD	BSR	GETCHR	X GETS CSR PTR & A GETS CHR.
48	FC7B	81 1B	ENDCHR	CMP A	#\$1B	TESTS FOR AN "ESC" CHAR.
49	FC7D	26 04		BNE	ED1	SKIPS IF NOT EDIT END.
50	FC7F	7F 0032		CLR	EDIT	TURNS OFF EDIT FLAG.
51	FC82	39		RTS		EXITS THE EDITOR.
52	FC83	SD EA	ED1	BSR	INSERT	EDITS CHARACTER.
53	FC85	20 F2		BRA	EDREAD	GOES FOR NEXT CHARACTER.

15 \*  
 316 \*  
 317 \* FOLLOWING IS THE MAIN EDITOR EXECUTION LOOP.  
 318 \*  
 319 \*  
 320 \*  
 321 \*  
 322 \*  
 323 FC87 81 0D CR CMP A #\\$0D TESTS FOR CARRIAGE RETURN.  
 324 FC89 2D AC BLT HOME SKIPS IF HOME CURSR COMMD.  
 325 FC88 2E 12 BGT RTCCSR GOES TO NEXT COMMD TEST.  
 326 FC8D 86 E0 CR1 LDA A #\\$60 LOADS INTERNAL C. R. VALUE.  
 327 \*  
 328 \*  
 329 \*  
 330 \*  
 331 FC8F 81 09 INSERT CMP A #\\$09 TESTS FOR A CONTROL 'I'.  
 332 FC91 2D 16 BLT DELETE SKIPS TO DELETE COMMD.  
 333 FC93 2E F2 BGT CR SKIPS FOR NEXT TEST.  
 334 FC95 DE 32 INSRT1 LDA B EDIT TESTS IF EDITOR IS ON.  
 335 FC97 27 03 BEQ INSRT2 SKIP TO EXIT IF EDITOR OFF.  
 336 FC99 BD FD46 JSR MOVE2 MOVES LAST LINE TO BUFFHI.  
 337 FC9C 7E FD74 INSRT2 JMF SCRLDN MOVES ALL LINES DOWN ONE.  
 338 \*  
 339 \*  
 340 \*  
 341 FC9F 81 12 RTCCSR CMP A #\\$12 TESTS FOR RIGHT ARROW.  
 342 FCA1 2D 28 BLT SUB32 SKIPS IF AN "UP ARROW".  
 343 FCA3 2E 08 BGT LFTCSR SKIPS IF A "LEFT ARROW".  
 344 FCA5 DE 1C RTARRO LDX CSRPTR LOADS CURSOR POINTER.  
 345 FCA7 20 1F BRA PUTCH1 STORES & INCREMENTS CSR.  
 346 \*  
 347 \*  
 348 \*  
 349 FCA9 80 67 DELETE BSR OVR1A SCROLLS UP ONE LINE.  
 350 FCA8 20 40 BRA OVR3 MOVES NEW LAST SCREEN LINE.  
 351 \*  
 352 \*  
 353 \*  
 354 FCAD 81 14 LFTCSR CMP A #\\$14 TESTS IF "LEFT ARROW".  
 355 FCAF 2D 24 BLT ADD32 SKIPS IF "DOWN ARROW".  
 356 FC81 2E 0C BGT CLER SKIPS FOR NEXT TEST.  
 357 FC83 09 DEX SUB 1 FROM CSRPTP.  
 358 FC84 20 25 BRA ADD2 STORES CURSOR POINTER.  
 359 \*  
 360 \*  
 361 \*  
 362 FC86 81 1F CLER CMP A #\\$1F TESTS FOR CTRL BACK ARROW.  
 363 FC88 2D 83 BLT CLEAR GOES TO CLEAR SCREEN.  
 364 FC8A 27 41 BEQ LFTJST MOVES CSR TO LEFT OF SCREEN.  
 365 \*  
 366 \*  
 367 \* ALL OTHER CHARACTERS FALL THRU TO PUTCHR.  
 368 \*

008 PDS-V3N

\* PUTCHR DISPLAYS A CHARACTER ON THE CRT DISPLAY AND  
\* INCREMENTS THE CURSOR POINTER AS WELL AS CHECKING  
\* AND HANDLING CARRIAGE RETURNS.

1  
2  
3  
4  
5  
6 FCBC DE 1C PUTCHR LDX CSRPTR LOADS OLD CSRPTR.  
7 FCBE 81 E0 CMP A #\$0D TESTS FOR EXTERNAL C. R.  
8 FCC0 27 E4 BEQ CRLF1 SKIPS TO DO A C. R. L. F.  
9 FCC2 A7 00 STA A 0,X DISPLAYS CHAR ON SCREEN.  
10 FCC4 81 E0 CMP A #\$60 TESTS FOR INTERNAL C. R.  
11 FCC6 27 4C CRLF1 BEQ CRLF SKIPS FOR CR. LF.  
32 FCC8 08 PUTCH1 INX INCREMENTS CSRPTR.  
33 FCC9 20 10 BRA ADD2 TESTS FOR OVRFLO & UNDRFLD.

185 FCCB DE 1C SUB32 LDX CSRPTR LOADS CURRENT CCSR POSITION.  
186 FCCD C6 20 LDA B #32 LOADS LOOP COUNT.  
187 FCCF 09 SUB32A DEX DECREMENTS CSRPTR.  
188 FCC0 5A DEC B - DECREMENTS LOOP COUNTR.  
189 FCC1 26 FC BNE SUB32A SKIPS BACK IF NOT DONE.  
190 FCC3 20 06 BRA ADD2 SKIPS TO CHECK UNDRFLD.

3392 FCC5 C6 20 ADD32 LDA B #32 LOADS LOOP COUNTER.  
3393 FCC7 08 ADD32A INX INCRE. CSRPTR IN INDEX.  
3394 FCC8 5A DEC B - DECREMENTS LOOP COUNTER.  
3395 FCC9 26 FC BNE ADD32A SKIPS BACK IF NOT DONE.  
3396 FCCB DF 1C ADD2 STX CSRPTR SAVES CSRPTR.

00408 \* NDRFLD (UNDERFLOW) CHECKS FOR THE CURSOR GOING OFF THE  
00409 \* TOP OF THE SCREEN. THE INDEX REG. CONTAINS THE CURSOR  
00400 \* POINTER WHEN THE ROUTINE IS ENTERED.  
00401 \*  
00402 \*  
00403 FCCD 8C E000 NDRFLD CPX #\$E000 TESTS IF CSRPTR >= DFFF.  
00404 FCE0 20 E1 BGE OVRFLO -- SKIPS IF CSRPTR GREATER.  
00405 FCE2 30 E1 BSR INSRTI SCRILLS DOWN & MOVES LINE.  
00406 FCE4 80 32 BSR MOVS3 MOVES BUFFL0 TO TOP OF CRT.

00408 \*  
00409 \*  
00410 \* OVERFLOW CHECKS FOR SCROLLING UP (CURSOR IS OFF  
L 1 \* THE BOTTOM OF THE SCREEN); INDEX CONTAINS THE CURSOR  
00412 \* POINTER UPON ENTRY.  
00413 \*  
00414 \*  
00415 FCE6 8C E200 OVRFL0 CPX #E200 TESTS AND EXITS IF  
00416 FCE9 2B 18 BMI OVREXT CURSOR ON SCREEN.  
00417 FCEB 8D 17 BSR OVR1 DOES OVR1 CHECKING.  
00418 FCED D6 32 OVR3 LDA B EDIT TESTS IF EDIT IS ON.  
00419 FCEF 27 12 BEQ OVREXT EXITS IF IT IS OFF.  
00420 FCF1 DE 22 LDX BUFFHI LOADS HI TEXT PTR.  
00421 FCF3 9C 0E CPX BUFEND TESTS IF PTRS NOT EQU.  
00422 FCF5 27 0C BEQ OVREXT EXITS IF NO TEXT.  
00423 FCF7 8D 3C BSR MOVE1A MOVES CHRS TO LAST LINE.  
00424 FCF9 DE 14 LDX SRCADR RESETS NEW BUFFHI  
00425 FCFB DF 22 STX BUFFHI LOCATION.  
00426 \*  
00427 \*  
00428 \*  
00429 \*  
00430 \* FOLLOWING ROUTINE MOVES THE CURSER TO THE LEFT.  
00431 \*  
00432 FCFD D6 1D LFTJST LDA B CSRPTR+1 LOADS LOW BYTE OF PTR.  
00433 FCFF C4 E0 AND B #\$E0 TRUNCATES TO LEFT OF LINE.  
00434 FD01 D7 1D STA B CSRPTR+1 SAVES L. J. ED PTR.  
00435 FD03 39 OVREXT RTS RETURNS TO EDITOR.  
00436 \*  
00437 \*  
00438 \*  
00439 \*  
00440 \* OVR1 DOES ACTUAL SCROLLING UP.  
00441 \*  
00442 FD04 D6 32 OVR1 LDA B EDIT TESTS IF EDIT IS ON.  
00443 FD06 27 0A BEQ OVR1A SKIPS IF EDIT OFF.  
00444 FD08 DE 20 LDX BUFFLO LOADS TEXT PTR LOW.  
00445 FD0A DF 16 STX DSTADR DESTINATION OF TEXT MOVE.  
00446 FD0C DE 24 LDX SCNPTR SOURCE FOR MOVE.  
00447 FD0E SD 26 BSR MOVE1 MOVES LIN1 TO BUFFFL0.  
00448 FD10 DF 20 STX BUFFLO SAVES NEW BUFFLO PTR.  
00449 FD12 20 4B OVR1A BRA SCRLUP SCROLLS SCREEN UP 1.  
  
00451 \* FOLLOWING ROUTINE MOVES THE CURSOR.  
00452 \*  
00453 FD14 SD BF CRLF BSR ADD32 LINE FEED.  
00454 FD16 20 E5 BRA LFTJST CARRIAGE RETURN.

00456 \* MOVE INSTRUCTIONS MOVE FROM ONE BUFFER AREA TO  
 00457 \* ANOTHER BUFFER AREA.  
 00458 \*  
 00459 \*  
 00460 \*  
 00461 \*  
 00462 \* MOVE3 CALCULATES THE SOURCE ADDRESS OF THE DATA IN  
 00463 \* BUFFLO (IF IT EXISTS) FOR MOVING TO THE FIRST LINE ON  
 00464 \* THE CRT. MOVE 1 IS THEN ENTERED TO DO THE MOVING.  
 00465 \*  
 00466 FD18 DE 24 MOVE3 LDX SCN PTR CSR PTR GETS E000 (HOME).  
 00467 FD1A DF 16 STX DST ADR SETS MOVE ADDRESS.  
 00468 FD1C DE 20 LDN BUFF LO LOADS LO BUFFR ADDR.  
 00469 FD1E 9C 0C CPX BUF ADR TESTS IF STRING EXISTS.  
 00470 FD20 27 23 BEQ MOVE EXT EXITS IF EMPTY.  
 00471 FD22 09 DEX  
 00472 FD23 9C 0C MV31 CPX BUF ADR MOVES BACK FROM BLANK.  
 00473 FD25 27 08 BEQ MV32 TESTS IF SRCADR = BUFFADR.  
 00474 FD27 09 DEX MOVES IF START OF LINE.  
 00475 FD28 E6 00 LDA B B,X. NEXT LOWER CHAR.  
 00476 FD2A C1 00 CMP B #\$60 GETS SOURCE CHAR FOR TEST.  
 00477 FD2C 26 F5 BNE MV31 TESTS FOR "C. R.".  
 00478 FD2E 08 INX SKIPS BACK UNTIL "C. R.".  
 00479 FD2F DF 20 MV32 STX BUFF LO POINTS BACK TO FIRST CHAR.  
 00480 FD31 20 03 BRA MOVE1 SAVES LO ADDRESS.  
 00481 MOVE DATA.  
 00482 \*  
 00483 \*  
 00484 \* MOVE1  
 00485 \*  
 00486 \* MOVE1 MOVES A SET OF CHARACTERS FROM EITHER THE TOP  
 00487 \* LINE OF THE SCREEN TO BUFFLO OR FROM BUFFHI TO THE  
 00488 \* BOTTOM LINE OF THE SCREEN. THE SOURCE ADDRESS IS PASSED  
 00489 \* IN THE INDEX REG., THE DESTINATION ADDRESS IN DSTADR.  
 00490 \* AND THE MOVE IS TERMINATED BY A "C. R." IN THE LINE OF  
 00491 \* TEXT BEING MOVED.  
 00492 \*  
 00493 \*  
 00494 FD33 DE 14 MOVE LDX SRC ADR LOADS SOURCE ADDRESS INTO X.E  
 00495 FD35 08 MOVE1A INX POINTS TO NEXT SOURCE CHAR.  
 00496 FD36 E6 00 MOVE1 LDA B B,X. LOADS SOURCE CHARACTER.  
 00497 FD38 DF 14 STX SRC ADR SAVES THE SOURCE POINTER.  
 00498 FD3A DE 16 LDN DST ADR LOADS DESTINATION ADDRESS.  
 00499 FD3C E7 00 STR B B,X. STORES CHR. IN DESTINATION.  
 00500 FD3E 08 INX NEXT DESTINATION ADDRESS.  
 00501 FD3F DF 16 STX DST ADR SAVES DESTINATION PTR.  
 00502 FD41 C1 00 CMP B #\$60 TESTS IF MOVE FINISHED (CR).  
 00503 FD43 26 E.E BNE MOVE - SKIPS BACK IF NOT DONE.  
 00504 FD45 39 RTS RETURNS TO CALLER.

00482 \*  
 00483 \*  
 00484 \* MOVE1  
 00485 \*  
 00486 \* MOVE1 MOVES A SET OF CHARACTERS FROM EITHER THE TOP  
 00487 \* LINE OF THE SCREEN TO BUFFLO OR FROM BUFFHI TO THE  
 00488 \* BOTTOM LINE OF THE SCREEN. THE SOURCE ADDRESS IS PASSED  
 00489 \* IN THE INDEX REG., THE DESTINATION ADDRESS IN DSTADR.  
 00490 \* AND THE MOVE IS TERMINATED BY A "C. R." IN THE LINE OF  
 00491 \* TEXT BEING MOVED.  
 00492 \*  
 00493 \*  
 00494 FD33 DE 14 MOVE LDX SRC ADR LOADS SOURCE ADDRESS INTO X.E  
 00495 FD35 08 MOVE1A INX POINTS TO NEXT SOURCE CHAR.  
 00496 FD36 E6 00 MOVE1 LDA B B,X. LOADS SOURCE CHARACTER.  
 00497 FD38 DF 14 STX SRC ADR SAVES THE SOURCE POINTER.  
 00498 FD3A DE 16 LDN DST ADR LOADS DESTINATION ADDRESS.  
 00499 FD3C E7 00 STR B B,X. STORES CHR. IN DESTINATION.  
 00500 FD3E 08 INX NEXT DESTINATION ADDRESS.  
 00501 FD3F DF 16 STX DST ADR SAVES DESTINATION PTR.  
 00502 FD41 C1 00 CMP B #\$60 TESTS IF MOVE FINISHED (CR).  
 00503 FD43 26 E.E BNE MOVE - SKIPS BACK IF NOT DONE.  
 00504 FD45 39 RTS RETURNS TO CALLER.

00506 \* MOVE2 SUBROUTINE  
 00507 \*  
 00508 \* THE MOVE2 SUBROUTINE MOVES THE LAST LINE ON THE  
 00509 \* SCREEN TO THE HIGH AREA OF THE BUFFER (BUFFHI) DURING  
 00510 \* SCROLLING. THE TEXT IS TEMPORARILY STORED ON THE STACK  
 . 11 \* DURING THE MOVE. THE MOVE IS TERMINATED BY A "C. R.". .  
 00512 \* THE TEXT IS STORED AT BUFFHI ON DOWN.  
 00513 \*  
 00514 \*  
 00515 \*  
 00516 \*

00517 FD46 CE E1E0	MOVE2	LDX #LASTLN	X GETS ADDR OF LAST LINE.
00518 FD49 5F		CLR B -	SETS TERMINATION FOR
00519 FD4A 37	MV21	PSH B STACK	POPPING.
00520 FD4B E6 00		LDA B 0,X	LOADS SOURCE CHAR.
00521 FD4D 00		INX	POINTS TO NEXT CHAR.
00522 FD4E C1 60		CMP B #\$60	TESTS IF LINE TO "C. R."
00523 FD50 26 F8		BNE MV21	MOVED TO STACK.
00524 FD52 DE 22	MV22	LDX BUFFHI	INIT. DESTINATION.
00525 FD54 E7 00	MV23	STA B 0,X	STORES CHAR.
00526 FD56 09		DEX	POINTS TO NEXT LOCATION.
00527 FD57 DF 22		STX BUFFHI	UPDATES BUFFER PTR.
00528 FD59 33		PUL B -	GETS NEXT CHAR.
00529 FD5A C1 00		CMP B #00	TESTS IF ALL CHR'S STORED.
00530 FD5C 26 F6		BNE MV23	SKIPS BACK IF NOT STORED.
00531 FD5E 39	MOVEX	RTS	RETURNS TO CALLER.

00533	*	SCROLLUP MOVES ALL LINES UP 1, & CLEARS LAST LINE.	
( 34	*		
00535 FD5F CE E000	SCRLUP	LDX #\$E000	SETS CRT HOME POSITION.
00536 FD62 E6 20	SCRPI	LDA B \$20,X	GETS CHAR FROM NEXT LINE.
00537 FD64 E7 00		STA B 00,X	STORES CHAR ON PREV. LINE.
00538 FD66 00		INX	POINTS TO NEXT LINE.
00539 FD67 8C E1E0		CPX #LASTLN	TESTS IF MOVE DONE.
00540 FD6A 26 F6		BNE SCRPI	GOES BACK IF NOT DONE.
00541 FD6C DF 1C		STX CSRPTR	SETS CSRPTR TO LAST LINE.
00542 FD6E DF 16		STX DSTADR	INIT DEST FOR NEXT MOVE.
00543 FD70 BD FC3D		JSR CLEAR	CLEARS LAST LINE.
00544 FD73 39		RTS	EXITS.

0546 \* SCRLDN MOVES ALL LINES DOWN ONE AND  
 0547 \* CLEARS THE TOP LINE ON THE SCREEN.  
 0548 \*  
 0549 FD74 CE E1DF SCRLDN LDX #LASTLN-1 INITIALIZES THE POINTER.  
 0550 FD77 E6 00 SCRD1 LDA B 0,X LOADS DATA TO BE MOVED.  
 0551 FD79 E7 20 STA B \$20,X MOVES DATA DOWN ONE LINE.  
 0552 FD7B DF 1C STX CSRPTR SAVES CURSOR.  
 0553 FD7D 09 DEX POINTS TO NEXT BYTE.  
 0554 FD7E 8C DFFF CPX #\$DFFF TESTS IF MOVE FINISHED.  
 0555 FD81 26 F4 BNE SCRD1 SKIPS BACK IF NOT DONE.  
 0556 FD83 C6 60 LDA B #\$60 LOADS BLANK TO CLEAR LINE.  
 0557 FD85 08 SCRD2 INX POINTS TO NEXT CHARACTER.  
 0558 FD86 E7 00 STA B 0,X CLEARS BYTE ON LINE 1.  
 0559 FD88 8C E01F CPX #FRSTLN TESTS IF LINE 1 CLEARED.  
 0560 FD8B 26 F8 BNE SCRD2 SKIPS BACK IF NOT CLEARED.  
 0561 FD8D 39 RTS RETURNS.

00563 \* OUTSTRING PRINTS OUT THE STRING BETWEEN THE  
 00564 \* OUTBUF POINTER AND THE BUFEND POINTER.  
 00565 \*  
 00566 FD8E DE 11 OUTSTR LDX OUTBUF BUFPTR GETS START OF TEXT.  
 00567 FD90 A6 00 OUT1 LDA A 0,X LOADS CHAR TO BE PUT OUT.  
 00568 FD92 DF 1E STA BUFPTR SAVES SOURCE POINTER.  
 ( 59 FD94 BD FCBC JSR PUTCHR PRINTS CHARACTER.  
 00570 FD97 DE 1E LDX BUFPTR RESTORES POINTER.  
 00571 FD99 9C 0A CPX OUTEND TESTS FOR END-OF-TEXT.  
 00572 FD9B 27 03 BEQ OUT2 EXITS IF END OF TEXT.  
 00573 FD9D 08 INX INCRE. PTR TO NEXT CHAR.  
 00574 FD9E 20 F0 BRA OUT1 GOES BACK FOR NEXT CHAR.  
 00575 FDA0 39 OUT2 RTS EXITS ROUTINE.  
 00576 \*  
 00577 \* END OF EDITOR PROGRAM.  
 00578 \*

```

*      THE MINI-ASSEMBLER
*
*
*      THE MINI-ASSEMBLER IS A FIXED-FIELD ONE INSTRUCTION
* PER LINE 2 PASS ASSEMBLER. THE MINI-ASSEMBLER FORMAT
* IS DESCRIBED ON PAGES 9-2 AND 9-3 OF THE SPHERE
* OPERATORS REFERENCE MANUAL.
*      THE TWO PASSES ARE REQUIRED TO FORM THE LABEL
* ADDRESSES. THE SECOND PASS EQUATES THE ADDRESS FOR
* LABELS REFERENCED BEFORE THEY ARE DEFINED IN THE PROGRAM.
*
*
*      ON ENTRY:
* SRCASM = ADDRESS OF SOURCE TEXT TO BE ASSEMBLED.
* BUFFLO = ADDRESS OF OBJECT CODE PRODUCED.
*
*      ON EXIT:
* PCVAL (PROGRAM COUNTER VALUE) = LAST LOCATION OF
* THE ASSEMBLED OBJECT PROGRAM.
*
*
*      ALGORITHM:
*
*ASMBLR: SET PASS COUNT TO ZERO; SET PCVAL TO DSTASM;
*ASM1A: OPERAND VALUE FORMED IN "ONDVAL":
*      A GETS CHAR IN X6 (OPERAND TYPE); X GETS X+7;
*      IF CHAR X6 IS A "0" THEN ONDVAL GETS VALUE FROM SYMBOL
* TABLE ELSE ONDVAL GETS VALUE FROM ASCBIN CONVERSION;
*SYMBL: EQUATES SYMBOL (PC VALUE IS THE " " SYMBOL
*[LABEL]) TO A LABEL VALUE;
*      SYMVAL GETS PCVAL;
*      IF X(1) IS AN "=" THEN SYMVAL GETS ONDVAL;
*      IF X(1) IS NOT A "=" OR A SPACE THEN IF SECOND PASS THEN
*          EXIT ELSE START SECOND PASS;
*      LABEL ENTRY IN SYMBOL TABLE GETS SYMVAL;
*LDOP: PUT OPERATION CODE INTO THE OBJECT CODE;
*      CONVERT X(2)-X(3) INTO BINARY;
*      SAVE PCVAL;
*      P. C. GETS P. C. +1;
*DPRND: FORM OPERAND IN OBJECT CODE:
*      FORM ONDVAL INTO FINGER SIZE BASED ON CODE IN X(6);
*      STORE NEW OPERAND VALUE IN MEMORY;
*      P. C. GETS P. C. +1 UP. 2;
*GETLN: GET NEXT LINE OF SOURCE;
*      GO TO ASM1H;
*
*      *

```

E 014 PDS-V3N

30	FDA1	7F 0004	ASMBLR CLR	AR3	INIT. PASS CTR TO FRST PASS.
31	FDA4	DE 20	ASM1 LDX .	BUFFLO	SETS PC CNTR TO START OF
32	FDA6	DF 40	STX PCVAL		OBJECT CODE.
33	FDA8	DE 26	LDX SRCASH		LOADS ADDR FOR FIRST LINE.
34	FDA9	DF 02	ASM1A STX TMP1		SAVES ADDR OF CURRENT LINE.
35	FDAC	A6 08	LDA A 8,X		LOADS SYMBOL (LABEL).
36	FDAC	E6 07	LDA B 7,X		LOADS OPERAND TYPE CODE.
37	FDB0	C1 40	CMP B #\$'@		IF @, LOADS DATA IN SYMBOL
38	FDB2	27 6B	BEQ INDADR		ADDRESS. GOES TO SYMBL.
39	FDB4	08	INX		SETS INDEX TO START OF
40	FDB5	08	INX		OPERAND NUMBER.
41	FDB6	08	INX		
42	FDB7	08	INX		
43	FDB8	08	INX		
44	FDB9	08	INX		
45	FDBA	08	INX		
46	FDBB	BD FF22	JSR ASCBIN		CONVRTS # TO BINARY IN B-A.
47	FDBE	D7 2A	ASM1B STA B ONDVAL		STORES OPERAND VALUE IN
48	FDC0	97 2B	STA A ONDVAL+1		ONDVAL.
49	*				
50	*				
51	*				
52	*				* FOLLOWING FORMS THE VALUE FOR THE LABEL.
53	*				
54	FDC2	DE 02	SYMBL LDX TMP1		LOADS ORIG LINE PTR INTO X.
55	FDC4	A6 00	LDA A 0,X		LOADS SYMBOL (LABEL).
56	FDC6	E6 01	LDA B 1,X		LOADS LABEL CONTROL CHAR.
57	FDC8	DE 40	LDX PCVAL		LABEL VALUE GETS PCVAL.
58	FDCA	DF 2C	STX SYMVAL		
59	FDCC	C1 3D	CMP B #\$'='		TESTS IF LABEL IS EQUATED.
60	FDCE	26 06	BNE ASM2		SKIPS IF NOT EQUATED.
61	FDD0	DE 2A	LDX ONDVAL		LABEL VALUE (SYMVAL) GETS
62	FDD2	DF 2C	STX SYMVAL		THE OPERND VALUE.
63	FDD4	20 0E	BRA ASM3		CONTINUES EVALUATION.
64	FDD6	C1 20	CMP B #\$'		TESTS FOR END-OF-PROGRAM.
65	FDD8	27 0A	BEQ ASM3		SKIPS IF SPACE (NOT END).
66	FDDA	7D 0004	TST AR3		TESTS IF SECOND PASS.
67	FDDC	27 01	BEQ ASM2A		EXITS IF SECOND PASS.
68	FDDF	39	RTS		EXITS THE ASSEMBLER.
69	FDE0	DF 04	ASM2A STA B AR3		SETS CTR TO SECOND PASS.
70	FDE2	20 C0	BRA ASM1		GOES BACK FOR SECOND PASS.
71	*				
72	*				
73	*				
74	*				* FOLLOWING PUTS THE LABEL VALUE IN THE SYMEOL TABLE.
75	*				
76	FDE4	80 41	ASM3 BSR SYNPTR	X	GETS SYMOL TABL ENTRY ADD. E
77	FDE6	96 2C	LDA A SYMVAL		STORES THE LABEL
78	FDE8	A7 00	STA A 0,X		ADDRESS (SYMVAL) INTO THE
79	FDEA	96 2D	LDA A SYMVAL+1		SYMBOL TABL.
80	FDEC	A7 01	STA A 1,X		

## \* FOLLOWING FORMS THE OPERATION CODE.

\*

1 FDEE DE 02	LLOOP	LDX	TMP1	LOADS ORIG LINE POINTER.
2 FDF0 08		INX		SETS X TO POINT TO
3 FDF1 08		INX		THE OP CODE CHARS.
4 FDF2 08		INX		
5 FDF3 A6 00		LDA A	0,X	GETS OP CODE CHAR INTO A.
6 FDF5 81 20		CMP A	#\$'	TESTS IF OP CODE EXISTS.
7 FDF7 27 0A		BEQ	OPRND	SKIPS IF NONEXISTANT.
8 FDF9 BD FF22		JSR	RSCBIN	CONVRTS OP CODE TO BINARY.
9 FDFC DE 40		LDX	PCVAL	LOADS PTR TO OBJECT CODE.
10 FDFE A7 00		STA A	0,X	STORS OP INTO OBJECT CODE.
11 FE00 08		INX		SETS TO NEXT OBJ CODE LOCTN.
12 FE01 DF 40		STX	PCVAL	SAVES P.C. POINTER.

6 \*

7 \*

8 \*

9 \*

10 \*

## \* FOLLOWING STORES INTO THE OBJECT CODE THE SIZED OPERAND.

\*

11 FE03 DE 02	OPRND	LDX	TMP1	LOADS SOURCE LINE POINTER.
12 FE05 A6 06		LDA A	6,X	LOADS OPERAND SIZE CHAR.
13 FE07 DE 40		LDX	PCVAL	LOADS X WITH OBJ CODE PTR.
14 FE09 81 45		CMP A	#\$'E	TESTS LENGTH TYPE.
15 FE0B 2E 31		BGT	RELTIV	SKIPS IF AN "R" OPERAND.
16 FE0D 27 21		BEQ	EXTEND	SKIPS IF AN "E" SIZE OPRND.
17 FE0F 81 44		CMP A	#\$'D	TESTS IF SIZE CHR EXISTS.
18 FE11 27 22		BEQ	DIRECT	SKIPS IF "D"CMDN EXISTS.

19 \*

20 \*

21 \*

22 \*

23 \*

24 \*

25 \*

26 \*

## \* FOLLOWING GETS THE NEXT LINE.

\*

27 FE13 DE 02	ASM4	LDX	TMP1	LOADS START OF LINE IN
28 FE15 08	ASM4R	INX		ORDER TO FIND NEXT LINE.
29 FE16 A6 E0		LDA A	0,X	LOADS CHAR FROM SOURCE LINE.
30 FE18 81 60		CMP A	#\$60	TESTS FOR A CARRAGE RETURN.
31 FE1A 26 F9		BNE	ASM4A	SKIPS BAK UNTIL C.R. FOUND.
32 FE1C 08		INX		POINTS TO FIRST LINE CHAR.
33 FE1D 20 8B		BRA	ASM1A	GOES BACK TO ASSM. NEXT LINE

34 \*

35 \*

36 \*

37 \*

38 \*

## \* THE FOLLOWING ARE SUBROUTINES USED BY THE MAIN CODE.

\*

39 \*

40 \*

41 \*

42 \*

43 \*

44 \*

45 \*

46 \*

47 \*

48 \*

49 \*

50 \*

51 \*

52 \*

53 \*

54 \*

55 \*

56 \*

57 \*

58 \*

59 \*

60 \*

61 \*

62 \*

63 \*

64 \*

65 \*

66 \*

67 \*

68 \*

69 \*

70 \*

71 \*

72 \*

73 \*

74 \*

75 \*

76 \*

77 \*

78 \*

79 \*

80 \*

81 \*

82 \*

83 \*

84 \*

85 \*

86 \*

87 \*

88 \*

89 \*

90 \*

91 \*

92 \*

93 \*

94 \*

95 \*

96 \*

97 \*

98 \*

99 \*

100 \*

101 \*

102 \*

103 \*

104 \*

105 \*

106 \*

107 \*

108 \*

109 \*

110 \*

111 \*

112 \*

113 \*

114 \*

115 \*

116 \*

117 \*

118 \*

119 \*

120 \*

121 \*

122 \*

123 \*

124 \*

125 \*

126 \*

127 \*

128 \*

129 \*

130 \*

131 \*

132 \*

133 \*

134 \*

135 \*

136 \*

137 \*

138 \*

139 \*

140 \*

141 \*

142 \*

143 \*

144 \*

145 \*

146 \*

147 \*

148 \*

149 \*

150 \*

151 \*

152 \*

153 \*

154 \*

155 \*

156 \*

157 \*

158 \*

159 \*

160 \*

161 \*

162 \*

163 \*

164 \*

165 \*

166 \*

167 \*

168 \*

169 \*

170 \*

171 \*

172 \*

173 \*

174 \*

175 \*

176 \*

177 \*

178 \*

179 \*

180 \*

181 \*

182 \*

183 \*

184 \*

185 \*

186 \*

187 \*

188 \*

189 \*

190 \*

191 \*

192 \*

193 \*

194 \*

195 \*

196 \*

197 \*

198 \*

199 \*

200 \*

201 \*

202 \*

203 \*

204 \*

205 \*

206 \*

207 \*

208 \*

209 \*

210 \*

211 \*

212 \*

213 \*

214 \*

215 \*

216 \*

217 \*

218 \*

219 \*

220 \*

221 \*

222 \*

223 \*

224 \*

225 \*

226 \*

227 \*

228 \*

229 \*

230 \*

231 \*

232 \*

233 \*

234 \*

235 \*

236 \*

237 \*

238 \*

239 \*

240 \*

241 \*

242 \*

243 \*

244 \*

245 \*

246 \*

247 \*

248 \*

249 \*

250 \*

251 \*

252 \*

253 \*

254 \*

255 \*

256 \*

257 \*

258 \*

259 \*

260 \*

261 \*

262 \*

263 \*

264 \*

265 \*

266 \*

267 \*

268 \*

269 \*

270 \*

271 \*

272 \*

273 \*

274 \*

275 \*

276 \*

277 \*

278 \*

279 \*

280 \*

281 \*

282 \*

283 \*

284 \*

285 \*

286 \*

287 \*

288 \*

289 \*

290 \*

291 \*

292 \*

293 \*

294 \*

295 \*

296 \*

297 \*

298 \*

299 \*

300 \*

301 \*

302 \*

303 \*

304 \*

305 \*

306 \*

307 \*

308 \*

## 016 PDS-V3N

1 FE30 D6 2A	EXTEND	LDA B	ONDVAL	STORES HI BYTE OF OPERAND
1 FE32 E7 00		STA B	0,X	INTO OBJECT CODE.
1 FE34 08		INX		INC PC TO POINT TO NXT WD.
2 FE35 96 2B	DIRECT	LDA A	ONDVAL+1	STORES LO BYTE OF OPERAND
3 FE37 A7 00		STA A	0,X	INTO OBJECT CODE.
4 FE39 08		INX		INC & SAVE P.C. TO POINT TO
5 FE3A DF 40		STX	PCVAL	NEXT BYTE.
6 FE3C 20 D5		BRA	ASM4	GOES TO WORK ON NEXT LINE.
7 *				
18 FE3E 08	RELTIV	INX		INCREMENT P.C. PTR TO POINT
19 FE3F DF 40		STX	PCVAL	TO NXT BYTE & SAVE P.C..
50 FE41 96 2B		LDA A	ONDVAL+1	LOADS LO BYTE OF OPERAND.
51 FE43 90 41		SUB A	PCVAL+1	FORMS RELATIVE OFFSET.
52 FE45 09		DEX		INSERTS RELATIVE BYTE INTO
53 FE46 A7 00		STA A	0,X	OBJECT CODE.
54 FE48 20 C9		BRA	ASM4	GOES TO ASSMBL NEXT LINE.
55 *				
56 *				END OF THE ASSEMBLER PROGRAM.
57 *				

## \* DEBUGGER

\* THE DEBUGGER FOR THE PDS SYSTEM WAS DESIGNED TO  
\* PROVIDE A VERSATILE TOOL FOR USE IN PROGRAM TESTING AND  
\* DEBUGGING. IT ALLOWS FOR BREAKPOINTS, MINI-ASSEMBLER  
\* SYMBOL TABLE REFERENCING, STACK MANIPULATION AND  
\* INPUT IN EITHER HEXADECIMAL, OCTAL OR DECIMAL.  
\* THE DEBUGGER PRINTS A PROMPT CHARACTER ">" ON EVERY NEW  
\* LINE. AN INSTRUCTION CAN BE TYPED IN WHENEVER THE CURSOR  
\* IS BLINKING, EXCEPT WHEN A NUMBER IS BEING TYPED IN.  
\* THE DEBUGGER CALLS THE EDITOR WHENEVER A NUMBER IS TO BE  
\* INPUT, SO CORRECTIONS CAN BE MADE IF THE WRONG DIGIT IS  
\* TYPED IN. THE POINTER "PCVAL" POINTS TO THE CURRENTLY  
\* OPENED BYTE LOCATION. THE DEBUGGER OPERATES ON WHATEVER  
\* BYTE IS POINTED TO BY PCVAL. FOR FURTHER DETAILS SEE THE  
\* SECTION ON THE DEBUGGER IN THE OPERATORS REFERENCE MANUAL.  
\*

\* THE DEBUGGER IS IMPLEMENTED BY A SMALL ROUTINE TO SET  
\* UP ENTRY <DEBUG> AND A LARGE ROUTINE WHICH DOES A RANGE  
\* COMPARE TO FIND THE PROPER COMMAND AND THEN EXECUTES THE  
\* COMMAND <RUNBUG>. NOTE THAT SINCE COMMANDS ARE  
\* DIFFERENTIATED BY RANGE, ANY KEY STRUCK WILL PRODUCE  
\* A COMMAND EXECUTION, SUCH AS A "," BEING INTERPRETED  
\* AS A "+" COMMAND.

## \* COMMANDS:

\* "C. R." LINE - PRINTS ">" OUT ON A NEW LINE.  
\* " " CHANGE - THE SPACE COMMND CHANGES CONTENTS FROM Y TO Z.  
\* "+" OPNNXT - OPENS NEXT LOCATION.  
\* "-" OPNPRE - OPENS PREVIOUS LOCATION.  
\* "↑B" BRKSET - SETS A BREAKPOINT AT THE OPENED LOCATION.  
\* "↑C" CLRBRK - CLEARS BREAKPOINT. MUST BE DONE BEFORE EXIT.  
\* "↑E" EXIT - PERFORM RTI - EXECUTE AT BRKPOINT LOCATION.  
\* "↑G" GOLOCN - STARTS EXECUTION AT OPENED LOCATION.  
\* "↑J" JUMP - JUMP TO USERS SUBROUTINE.  
\* "↑O" OPNLOC - OPENS LOCATION THAT IS TYPED IN AFTER "O".  
\* "↑R" OPNREG - OPENS THE TOP-OF-STACK LOCATION.  
\* "↑S" SETSTK - SETS THE STACK TO THE OPENED LOCATION.  
\* "↑T" QPNtbl - OPENS LOCATION IN SYMBOL TABLE OF NEXT CHR.  
\* "↑X" GOEXEC - EXITS THE DEBUGGER - GOES BACK TO ENEC.  
\*

## \* SUBROUTINES:

\* INPCHR - INPUTS A CHAR. INTO A AND PRINTS IT.  
\* INPNUM - INPUTS A NUMBER INTO B-A FROM THE KEYBOARD.  
\* PNTBYT - PRINTS ACC A AS 2 HEX DIGITS ON THE SCREEN.  
\* PNTDIG - PRINTS B-A AS 4 HEX DIGITS ON THE SCREEN.  
\* NEWLIN - PRINTS A C.R. AND A ">" ON THE SCREEN.  
\* DSPADR - PRINTS BYTE ADDRESS (XXXX) AND BYTE CONTENTS (YY)  
\* AS >XXXX YY ON THE SCREEN.  
\*

00138 FE4A	*	ORG	\$FE4A	
00139	*			
00140	*			
00141	*			
00142	*	FOLLOWING IS LOCATION OF ENTRY OF THE BRKPT VECTOR.		
00143	*			
00144 FE4A 30	BKENTR	TSX		INDEX GETS STACK POINTER.
00145 FE4B E6 05	LDA B	5,X		LOADS HI RETURN ADDRESS.
00146 FE4D A6 06	LDA A	6,X		LOADS LOW BYTE OF ADDRESS.
00147 FE4F 80 01	SUB A	#1		SUB 1 FROM RETURN ADDRESS.
00148 FE51 C2 00	SBC B	#0		
00149 FE53 E7 05	STA B	5,X		RESTORES RETURN ADDR. TO
00150 FE55 A7 06	STA A	6,X		THE BREAKPOINT LOCATION.
00151 FE57 20 0B	BRA	DEBUG		Goes to the debugger.
00152	*			
00153	*			
00154 FE59 81 0D	LINE	CMP A	#\$0D	TESTS FOR A C.R. (LINE).
00155 FE5B 2D 7D	BLT	IMPLCN		GOES TO 'JSR' ( $\uparrow$ J) ROUTINE.
00156 FE5D 2E 4E	BGT	OPNREG		SKIPS FOR NEXT ( $\uparrow$ R) TESTS.
00157 FE5F BD FCCB	JSR	SUB32		Moves cursor up one line.
00158 FE62 31	POPLIN	INS		CLEANS UP STACK FOR
00159 FE63 31		INS		DISPLAY OF C.R. >.
00160	*			
00161	*			
00162 FE64 8D 76	DEBUG	BSR	NEWLIN	PRINTS "C.R. >"
00163 FE66 8D 03	DEBUG1	BSR	INPCHR	READS IN COMMAND.
00164 FE68 BD FC45		JSR	RTARRO	INSERT BLANK.
00165 FE6B DE 40		LDX	FCVAL	LOADS CURRENTLY OPENED LOC.
00166 FE6D 8D 09		BSR	RUNBUG	EXECUTES DEBUG COMMAND.
00167 FE6F 20 F5		BRA	DEBUG1	GOES BACK FOR NEXT COMND.
00168	*			
00169	*			
00170 FE71 BD FC4A	INPCHR	JSR	GETCHR	READ IN CHAR INTO A.
00171 FE74 BD FCBC		JSR	PUTCHR	DISPLAYS CHARACTER.
00172 FE77 39		RTS		RETURNS TO CALLER.
00173	*			
00174	*			
00175	*			
00176 FE78 81 03	RUNBUG	CMP A	#\$03	TESTS FOR A "1C" COMMAND.
00177 FE7A 2D 25		BLT	EFKSET	SKIPS IF A "1B" COMMAND.
00178 FE7C 2E 45		BGT	EVIT	SKIPS FOR NEXT COMND TEST.
00179 FE7E 9E 30	CLFSRK	LDX	SEKACR	GETS ADDRESS OF BREAKPOINT.
00180 FE80 96 25		CPL A	EPKSHD	LOADS UP TO 16 BYTE CONTENTS.
00181 FE82 A7 00		STA A	O,X	RESTORES BYTE DATA.
00182 FE84 20 66		SRA	DSPADR	Goes to open the location.
00183	*			
00184	*			
00185 FE86 81 20	CHANGE	CMP A	#\$1	TESTS OFR A SPACE COMND.
00186 FE88 20 12		BLT	EXECUTV	SKIPS TO EXIT BACK TO EXEC.
00187 FE8A 2E 09		BGT	OPNPRE	SKIPS FOR OTHER CMND TESTS.
00188 FE8C 8D 56	SPACE	BSR	INPNUM	INPUTS NEW BYTE CONTENTS.
00189 FE8E DE 40		LDX	PCVAL	LOADS OPENED BYTE LOCATION.
00190 FE90 A7 00		STA A	0,X	STORES NEW BYTE CONTENTS.
00191	*			
00192	*			
00193 FE92 08	OPNNXT	INX		FORMS NEXT LOCATION ADDRES.
00194 FE93 20 57		BRA	DSPADR	GOES TO OPEN LOCATION BYTE.
0				

00200 FE95 81 2D	OPNPRE	CMP A	#\$1-	TESTS FOR A "-" COMMAND.
00201 FE97 2D F9		BLT	OPNNXT	SKIPS FOR A "+" COMMAND.
00202 FE99 09		DEX		FORMS PREV. LOCATION ADDR.
00203 FE9A 20 E0		BRA	DSPADR	GOES TO OPEN THE LOCATION.
00204	*			
00205	*			
00206	*			
00207 FE9C 31	EXECUT	INS		CLEANS UP THE STACK.
00208 FE9D 31		INS		
00209 FE9E 7E FC14		JMP	EXEC	RETURNS TO THE EXECUTIVE.
00210	*			
00211	*			
00212 FEAC 20 E5	BRKSET	LDA A	0,X	LOADS DATA OF OPENED LOCATN.
00213		STX A	BRKSAY	SAVES DATA OF OPENED BYTE.
00214		STX	BRKADDR	SAVES ADDR. OF BREAKPOINT.
00215 FEAD 81 12		LDA A	#\$3F	LOADS SOFTWARE INTUP COMND.
00216 FEAF 2D 0A		STA A	0,X	SETS AN SWI AT OPENED BYTE.
00217 FEB1 2E 19		BRA	POPLIN	GOES TO NEXT LINE FOR COMND.
00218 FEB3 30	*			
00219 FEB4 08	*			
00220 FEB5 08	*			
00221 FEB6 20 34	OPNREG	CMP A	#\$12	TESTS FOR "TR" (STACK TOP).
00222		BLT	OPNLOC	GOES TO OPEN A LOCATION.
00223		BGT	OPNTBL	SKIPS FOR NEXT TEST (TT).
00224 FEB8 35	SETSTK	TXS		OPENS TOP-OF-STACK.
00225 FEB9 20 F9		BRA	DEBUG	PCVAL GETS STACK POINTER.
00226	*			(CLEANS UP THE STACK).
00227	*			GOES TO DISPLAY THE T-O-S.
00228 FEBB 8D 27	OPNLOC	BSR	INPNUM	LOADS A 16 BIT NUMBER.
00229 FEBD D7 40	OPNLCL1	STA B	PCVAL	STORES NEWLY OPENED
00230 FEBF 97 41		STA A	PCVAL+1	LOCATION ADDRESS.
00231 FEC1 20 2B		BRA	DSPADR1	DISPLAYS CONTENTS OF LOCATN.
00232	*			
00233	*			
00234 FEC3 81 07	EXIT	CMP A	#\$07	TESTS IF AN EXIT (>E>) COMD.
00235 FEC5 27 11		BEQ	GOLOCN	SKIPS FOR THE "GO" COMMAND.
00236 FEC7 2E 90		BGT	LINE	SKIPS FOR NEXT COMND TEST.
00237 FEC9 31		INS		CLEAR'S UP THE STACK.
00238 FECB 31		INS		
00239 FECB 3B		RTI		RETURNS FROM BREAKPOINT.
00240	*			
00241	*			
00242 FECB 81 14	OPNTBL	CMP A	#\$14	TESTS IF A "TT" (TABLE).
00243 FECB 2D E8		BLT	SETSTK	GOES TO SET STACK PTR (T5).
00244 FECB 2E B4		BGT	CHANGE	SKIPS FOR NEXT TEST (SPACE).
00245 FECB 8D 9D		BSR	INPCHR	LOADS A WITH SYMBOL (LBL).
00246 FECB 48		RSL A	.	ALIGNS ADDRESS FOR
00247 FED5 5F		CLR B	.	SYMBOL TABLE ENTRY.
00248 FED6 20 E5		BRA	OPNLCL1	SAVES AND DISPLAYS ADDRESS.
00249	*			
00250	*			
00251 FED8 31	GOLOCN	INS		CLEANS UP THE STACK.
00252 FED9 31		INS		
00253 FEDB 6E 00	JMPLCN	JMP	0,X	JUMPS TO USERS PROGRAM.

6 55 \*  
00256 \* FOLLOWING ARE SUBROUTINES USED BY THE DEBUGGER.  
00257 \*  
00258 FEDC 86 0D NEWLIN LDA A #\$0D LOADS A CARRIAGE RETURN.  
00259 FEDE 8D 3E BSR PNTBF1 PRINTS A CARRIAGE RETURN.  
00260 FEE0 86 3E LDA A #'> LOADS A PROMPT CHARACTER.  
00261 FEE2 20 3A BRA PNTBF1 DISPLAYS PROMPTER CHAR.  
00262 \*  
00263 \*  
00264 \* FOLLOWING INPUTS A 16 BIT NUMBER INTO THEE BA REGISTER.  
00265 \*  
00266 FEE4 BD FC75 INPNUM JSR EDITIN INPUTS A STRING OF DIGITS.  
00267 FEE7 DE 24 LDX SCN PTR LOADS ADDR. OF FIRST DIGIT.  
00268 FEE9 8D 37 BSR ASCBIN CONVERTS TO BINARY # IN BA.  
00269 FEEB 39 RTS RETURNS TO CALLER.  
00270 \*  
00271 \*  
00272 \* FOLLOWING DISPLAYS THE LOCATION ADDR. & CONTENTS.  
00273 \*  
00274 FEEC DF 40 DESPA DR STX PCVAL SAVES OPENED LOCATION ADDR.  
00275 FEEE 8D EC DESPA D1 BSR NEWLIN PRINTS A "C. R." AND ">".  
00276 FEF0 8D 0A BSR PNTDIG PRINTS OUT "PCVAL" IN HEX.  
00277 FEF2 BD FC45 JSR RTARRO PRINTS A SPACE.  
00278 FEF5 DE 40 LDX PCVAL LOADS PTR. TO OPENED LOC.  
00279 FEF7 A6 00 LDA A 0,X LOADS DATA FROM LOCATN.  
00280 FEF9 8D 07 BSR PNTBYT PRINTS DATA IN HEX FORMAT.  
00281 FEFB 39 RTS RETURNS TO INPUT COMMAND.  
00282 \*  
00283 \*  
00284 FEFC 96 40 PNTDIG LDA A PCVAL PRINTS THE 2 HI HEX DIGITS  
00285 FEFE 8D 02 BSR PNTBYT OF OPENED ADDRESS.  
00286 FF00 96 41 LDA A PCVAL+1 PRINTS OUT 2 LOW HEX DIGITS  
00287 \*  
00288 \*  
00289 \* FOLLOWING PRINTS OUT 2 HEX DIGITS.  
00290 \*  
00291 FF02 CE 0010 PNTBYT LDX #16 LOADS 16 FOR BASE.  
00292 FF05 DF 04 STX ARB STORES FOR CONVERSION.  
00293 FF07 SF CLR B - CLEARS HI 2 DIGITS.  
00294 FF08' CE 0035 LDX #IOBUFF LOADS OUTPUT BUFF ADDRESS.  
00295 \*  
00296 \*  
00297 \* FOLLOWING CONVERTS BYTE TO HEX WITH LEADING ZEROS.  
00298 \*  
00299 FF0B D7 36 CONVRT STA B IOBUFF+1 CLERS SYTE FOR SECOND DIGIT.  
00300 FF0D BD FF64 JSR BINASC CONVERTS TO ASCII DIGITS.  
00301 FF10 96 35 LDA A IOBUFF LOADS HI DIGIT.  
00302 FF12 D6 36 LDA B IOBUFF+1 TESTS BOTH DIGITS CONVTD.  
00303 FF14 26 04 BNE PNTBUF SKIPS IF BOTH DIGITS CONVTD  
97 36 STA A IOBUFF+1 SETS UP LOW DIGIT.  
J0305 FF18 86 30 LDA A #\$10 HIGH DIGIT GETS A "0".  
J0306 \*  
J0307 FF1A 8D 02 PNTBUF BSR PNTSF1 PRINTS OUT HI DIGIT.  
J0308 FF1C 96 36 LDA A IOBUFF+1 LOADS LOW DIGI  
J0309 FF1E BD FCBC PNTBF1 JSR PUTCHR DISPLAYS CCHARACTER.  
00310 FF21 39 RTS RETURNS TO CALLING PROGRAM.  
00311 \*  
00312 \* END OF DEBUGGER

00314 \* UTILITY PROGRAMS  
00315 \*  
00316 \*  
00317 \*  
318 \*  
00319 \*  
00320 \*  
00321 \* THE ASCII TO BINARY ROUTINE CONVERTS FROM AN ASCII  
00322 \* NUMBER STRING POINTED TO BY X TO AN UNSIGNED 16 BIT  
00323 \* BINARY NUMBER IN BA (ACC B HAS THE HI BYTE, ACC A HAS  
00324 \* THE LO BYTE). THE ASCII STRING IS TERMINATED BY A NON  
00325 \* HEXADECIMAL CHARACTER. UPON EXITING, THE INDEX REGISTER  
00326 \* WILL POINT TO THE NEXT CHARACTER AFTER THE NUMBER  
00327 \* STRING. THE BASE OF THE NUMBER STRING IS PASSED TO  
00328 \* THE ROUTINE IN ARA (ARA IS THE ARITHMETIC REGISTER A  
00329 \* LOCATED IN BYTES 06 AND 07 OF LOW MEMORY). IF THE  
00330 \* ROUTINE IS ENTERED WITH A KNOWN BASE, PUT THE BASE  
00331 \* (BETWEEN 2 AND 16) IN ARA AND ENTER THE ROUTINE AT  
00332 \* THE ENTRY POINT ENTR2.  
00333 \*  
00334 \*  
00335 \*  
00336 \* CONVERSION FORMULA:  
00337 \* ASCII NUMBER STRING X[4], X[3], X[2], X[1] IN  
00338 \* BASE Y;  
00339 \* BINARY NUMBER =  
00340 \* X[4]\*Y↑3+X[3]\*Y↑2+X[2]\*Y↑1+X[1]\*Y↑0 OR  
00341 \* BINARY NUMBER =  
00342 \* (((0\*Y+X[4])\*Y+X[3])\*Y+X[2])\*Y+X[1]  
00343 \* WHERE ↑ IS THE EXPONENT OPERATOR,  
00344 \* X IS A CHARACTER & Y IS THE BASE.  
00345 \*  
00346 \*  
00347 \*  
00348 \* ALGORITHM:  
00349 \* ASCBIN: FORM THE BASE IN ARA BASED ON THE FIRST CHAR.  
00350 \* OF THE NUMBER STRING; INCREMENT CHAR. PTR. IN X;  
00351 \* ENTR2: NUMBER (IN BA) GETS 0;  
00352 \* NXTCHR: IF THE CURRENT CHAR. POINTED TO BY X IS NOT A  
00353 \* DIGIT THEN EXIT ELSE INCREMENT CHARACTER PTR IN INDEX;  
00354 \* CONVERT DIGIT TO BINARY;  
00355 \* NUMBER GETS NUMBER \* BASE;  
00356 \* NUMBER GETS NUMBER + DIGIT;  
00357 \* GO TO OPERATE ON THE NEXT DIGIT (NXTCHR);  
00358 \*

GE 008 PDS-V3N

0360 FF22 A6 00	ASCBIN	LDA A	0,X	GETS CHR TO FORM BASE.
0361 FF24 81 2E		CMP A	#\$1.	TESTS FOR DECM1 STRNG.
0362 FF26 2D 06		BLT	OCT	SKIPS IF BASE 8 (*).
0363 FF28 2E 09		BGT	HEX	SKIPS IF BASE 16.
0364 FF2A 86 0A		LDA A	#10	LOADS BASE 10 FOR CONVERSN.
0365 FF2C 20 02		BRA	ASC1	SKIPS TO INC. TEXT PTR.
0366 FF2E 86 08	OCT	LDA A	#8	LOADS BASE 8 FOR CONVERSION.
0367 FF30 08	ASC1	INX		INCREMENT PTR TO NEXT CHAR.
0368 FF31 20 02		BRA	ASC2	SKIPS TO SAVE BASE.
0369 FF33 86 10	HEX	LDA A	#16	LOADS BASE 16 FOR CONVERN.
0370 FF35 97 07	ASC2	STA A	AR0	SAVES BASE IN BASE#.
0371	*			
0372	*			
0373 FF37 5F	ENTR2	CLR B	NUMBER	GETS 0.
0374 FF38 37		PSH B	-	(LOW NUMBER ON STACK).
0375 FF39 D7 06		STA B	AR1	CLEAR5 HI OF BASE.
0376 FF3B A6 00	NXTCHR	LDA A	0,X	GETS CHAR TO CONVERT.
0377 FF3D 08		INX		INC TO NEXT CHARACTER.
0378 FF3E 81 30		CMP A	#\$'0	TESTS FOR END-OF-STRING.
0379 FF40 2D 20		BLT	REXIT	EXITS IF END.
0380 FF42 80 30		SUB A	#\$'0	FORMS B. C. D. NUMBER.
0381 FF44 81 0A		CMP A	#10	TESTS IF DECIMAL DIGIT.
0382 FF46 2D 0A		BLT	ASC3	SKIPS IF DECIMAL.
0383 FF48 81 10		CMP A	#16	TESTS FOR END OF STRING.
0384 FF4A 2F 16		BLE	REXIT	EXITS IF NOT A HEX DIGIT.
0385 FF4C 80 07		SUB A	#7	FORMS A HEX B. C. D. DIGIT.
0386 FF4E 81 10		CMP A	#16	TESTS FOR END-OF-STRING.
0387 FF50 2C 10		BGE	REXIT	EXITS IF CHAR > "F".
0388 FF52 97 06	ASC3	STA A	DIGIT	SAVES DIGIT FOR ADD.
0389	*			
0390 FF54 DF 00	CNVASC	STX	TMP	SAVES INDEX REG FOR MULT.
0391 FF56 32		PUL A	-	RESTORES LO OF "NUMBER".
0392 FF57 8D 3A		BSR	MULT	NUMBER GETS NUMBER * BASE.
0393 FF59 9B 08		ADD A	DIGIT	NUMBER GETS NUMBER + DIGIT.
0394 FF5B C9 00		ADC B	#0	
0395 FF5D 36		PSH A	-	SAVES LO OF NUMBER.
0396 FF5E 0E 00		LOX	TMP	RESTORES STRING PTR.
0397 FF60 20 09		BRA	NXTCHR	Goes TO CONVERT NEXT CHAR.
0398 FF62 32	REXIT	PUL A	-	RESTORES "NUMBER" IN BH.
0399 FF63 39		RTS		RETURNS TO CALLING PROGRAM.

1 \*  
 2 \*  
 3 \*  
 4 \* THE BINARY TO ASCII CONVERSION ROUTINE CONVERTS  
 5 \* A 16 BIT BINARY NUMBER IN THE BR REGISTER (REG B & REG R)  
 6 \* TO A STRING OF ASCII DIGITS. THE ASCII STRING CAN BE IN  
 7 \* ANY BASE FROM BASE 2 THROUGH BASE 41. THE VALUE OF THE  
 8 \* BASE IS LOCATED IN THE ARITHMETIC PSEUDO-REGISTER ARB  
 9 \* (ARB IS LOCATED IN BYTE AR3 [LOC 4] AND AR2 [LOC 5]).  
 10 \* WHEN THE ROUTINE IS ENTERED, THE POINTER TO THE OUTPUT  
 11 \* LOCATION IS PASSED IN THE INDEX REG. WHEN THE ROUTINE  
 12 \* EXITS, THE INDEX POINTS TO THE LAST DIGIT IN THE STRING  
 13 \* PLUS ONE.  
 14 \* CONVERSION IS DONE BY THE METHOD OF REPEATED  
 15 \* DIVISION. THE LOW ORDER DIGIT IS FORMED FIRST. THE  
 16 \* DIGITS ARE THEN PLACED ON THE STACK UNTIL CONVERSION IS  
 17 \* COMPLETED. THE DIGITS ARE THEN POPPED OFF THE STACK  
 18 \* AND PLACED IN THE OUTPUT STRING. THE TOP-OF-STACK IS  
 19 \* INITIALIZED TO HEX FF TO TELL WHEN ALL THE DIGITS  
 20 \* HAVE BEEN POPPED OFF THE STACK. AFTER THE DIVISION, THE  
 21 \* DIGIT (THE REMAINDER OF THE DIVISION OPERATION)  
 22 \* IS LOCATED IN THE HR 0 PART OF ARA (BYTE 7). WHEN  
 23 \* THE QUOTIENT OF THE DIVISION IS 0, THEN THE CONVERSION  
 24 \* IS COMPLETED.  
 25 \*  
 26 \*  
 27 \*

1428 FF64 DF 00	BINASC	STX	TMP	SAVES OUTPUT POINTER.
1429 FF66 34		DES		/SETS THE TOP-OF-STACK TO
1430 FF67 30		TSX		/ALL ONES TO TELL END OF
1431 FF68 6F 00		CLR	0,X	/CHAR STRING (LAST CHAR IS
1432 FF6A 63 00		COM	0,X	/PUT ON STACK FIRST).
1433 FF6C DE 04	BIN1	LDX	ARB	RESTORES DIVISOR (BASE).
1434 FF6E DF 06		STX	ARA	
1435 FF70 3D 3D		BSR	DIVIDE	* QUOTIENT IN BR GETS THE
1436				* REMAINDER OF # TO BE CONVERTED; REMAINDER IN ARA GETS
1437				* THE LOW ORDER DIGIT.
1438 FF72 97 02		STA R	TMP1	SAVES A OF BR.
1439 FF74 96 07		LDA R	AR0	LOAD DIGIT (REMAINDER).
1440 FF76 36		PSH R	-	STACK DIGIT (REVERSE ORDER). G
1441 FF77 96 02		LDA R	TMP1	RESTORES A OF BR.
1442 FF79 40		TST R	-	/TESTS IF QUOTIENT IS = 0
1443 FF7A 26 F0		BNE	BIN1	/ (SIGNIFYING THAT
1444 FF7C 5D		TST B	-	/ THE CONVERSRSION
1445 FFTD 26 ED		BNE	BIN1	/ IS DONE).
1446	*			
1447 FFTF DE 00	BINSTR	LDX	TMP	RESTORES OUTPUT POINTER.
1448 FF81 32	BIN3	PUL R	-	UNSTACK A DIGIT.
1449 FF82 4D		TST R	-	TESTS IF NEG (END?).
1450 FF83 2A 01		BPL	BIN4	SKIPS IF A DIGIT.
1451 FF85 39		RTS		EXITS FROM SUBROUTINE.
1452 FF86 81 09	BIN4	CMP R	#9	TESTS IF RESULT IS HEX.
1453 FF88 2F 02		BLE	BIN5	SKIPS IF DIGIT NOT HEX.
1454 FF8A 88 07		ADD R	#7	FORMS HEX VALUE OF DIGIT.
1455 FF8C 88 30	BIN5	ADD R	#\$'0	FORMS DECIMAL CHARACTER.
1456 FF8E A7 00		STA R	0,X	OUTPUTS CHARACTER.
1457 FF90 08		INX		POINTS TO NEXT CHARACTER.
1458 FF91 20 EE		BRA	BIN3	GOES BACK FOR NEXT DIGIT.

00460 \* MULTIPLY ROUTINE  
 00461 \*  
 00462 \* THE MULTIPLY ROUTINE MULTIPLIES TWO 16 BIT BINARY  
 00463 \* NUMBERS TOGETHER TO PRODUCE A 16 BIT RESULT. THE BA  
 00464 \* REGISTERS AND ARA (BYTES 6 & 7) REGISTER ARE USED.  
 00465 \* THE CONTENTS OF ARA ARE UNCHANGED UPON PROGRAM EXIT.  
 00466 \*  
 00467 \*  
 00468 \* BA GETS BA \* ARA  
 00469 \*  
 00470 \*  
 00471 \* MULTIPLYING IS ACCOMPLISHED BY REPEATED ADDITIONS  
 00472 \* OF ONE OF THE OPERATORS (OPERATOR ARA) INTO THE RESULT.  
 00473 \* THE RESULT STARTS OUT WITH A ZERO VALUE AND IS SHIFTED  
 00474 \* OVER ONE AFTER EACH ADDITION. THE HIGHEST ORDER VALUE  
 00475 \* IS ADDED IN FIRST AND THEN, GOING TO THE RIGHT,  
 00476 \* (THUS SHIFTING THE ANSWER LEFT ONE TO BRING IN THE NEXT  
 00477 \* RIGHTMOST DIGIT) GETTING THE NEXT LOWERMOST SIGNIFICANT  
 00478 \* DIGIT. THE NEXT RIGHTMOST BIT OF THE OTHER OPERAND  
 00479 \* (THE ONE ORIGINALLY IN BA) IS TESTED, AND IF ONE,  
 00480 \* ANOTHER ADDITION TAKES PLACE. THIS IS REPEATED UNTIL  
 00481 \* THE FINAL SUM IS FORMED.  
 00482 \*  
 00483 \*  
 00484 \*  
 00485 \* MULTIPLY ALGORITHM:  
 00486 \*  
 00487 \*MULT: STACK BA; BA GETS 0; SET COUNT VALUE TO 16;  
 00488 \*MUL1: SHIFT BA LEFT 1;  
 00489 \* SHIFT LEFT ORIG BA VALUE ON STACK INTO CARRY;  
 00490 \* IF CARRY = 0 THEN GO TO MUL2  
 00491 \* BA GETS BA + ARA;  
 00492 \*MUL2: DECREMENT COUNT;  
 00493 \* IF COUNT # 0 THEN GO TO MUL1 ELSE EXIT.  
 00494 \*  
 00495 \*  
 00496 \*  
 00497 \*  
 00498 FF93 36 MULT PSH A - PUTS THE ORIGINAL CONTENTS  
 00499 FF94 37 PSH B - OF BA onto the stack.  
 00500 FF95 86 1.0 LDA A #16 LOADS COUNT VALUE  
 00501 FF97 36 PSH A - onto the stack.  
 00502 FF98 4F CLR A - BA GETS ZEROED.  
 00503 FF99 5F CLR B -  
 00504 FF9A 30 TSX SET INDEX TO STACK.  
 00505 FF9B 48 MUL1 ASL A - SHIFT LEFT BA.  
 00506 FF9C 59 ROL B -  
 00507 FF9D 68 02 ASL 2,X SHIFTS ORIG. BA OPERAND  
 00508 FF9E 69 01 ROL 1,X ONE LEFT INTO CARRY.  
 00509 FFA1 24 04 BCC MUL2 SKIPS ADDING IF CARRY = 0.  
 00510 FFA3 9B 07 ADD A AR0 BA GETS BA + ARA.  
 00511 FFA5 D9 06 ADC B AR1  
 00512 FFA7 6A 00 MUL2 DEC 0,X TESTS IF DONE.  
 00513 FFA9 26 F0 BNE MUL1 GOES BACK IF NOT DONE.  
 00514 FFB8 31 INS CLEANS UP THE STACK.  
 00515 FFAC 31 INS  
 00516 FFAD 31 INS  
 00517 FFBE 39 RTS EXITS ROUTINE.

00519 \* DIVIDE ROUTINE  
 00520 \*  
 00521 \* THE DIVIDE SUBROUTINE DIVIDES THE 16 BIT NUMBER  
 00522 \* IN THE BA REGISTERS BY THE 16 BIT NUMBER IN THE PSEUDO  
 00523 \* REGISTER ARA (LOCATED IN BYTES 6 & 7). UPON EXITING,  
 00524 \* BA WILL CONTAIN THE QUOTIENT OF THE DIVISION AND ARA  
 00525 \* WILL CONTAIN THE REMAINDER. THE DIVIDEND BA IS DIVIDED  
 00526 \* BY THE DIVISOR ARA (I.E. BA/ARA ).  
 00527 \*  
 00528 \*  
 00529 \*  
 00530 \*  
 00531 \* DIVIDE ALGORITHM:  
 00532 \*  
 00533 \*DIVIDE: X3,4 GETS BA (BA IS PUT ON THE STACK);  
 00534 \* X1,2 GETS THE 16 BIT ARA VALUE (ARA PUT ON THE STACK);  
 00535 \* COUNT GETS 1 + THE NUMBER OF NONSIGNIFICANT BITS IN  
 00536 \* THE DIVISOR [LEFT JUSTIFY X1,2 TO FIRST 1 BIT, COUNT  
 00537 \* GETS 1 + THE # OF LEADING ZEROS IN X1,2] [COUNT WILL BE  
 00538 \* FROM 1 TO 17];  
 00539 \* BA GETS X3,4 [RESTORES BA];  
 00540 \* X3,4 GETS 0 [INITIALIZES THE QUOTIENT];  
 00541 \*DIV3: BA GETS BA - X1,2;  
 00542 \* IF THERE WAS A BORROW [I.E. DIVIDEND IN BA < DIVISOR  
 00543 \* IN X1,2 (ARA)] THEN BA GETS BA + X1,2 [ORIGINAL BA  
 00544 \* VALUE RESTORED] & CARRY CLEARED ELSE CARRY IS SET;  
 00545 \*DIV5: X3,4 [QUOTIENT] GETS CARRY LEFT SHIFTED IN;  
 00546 \* X1,2 [DIVISOR] GETS SHIFTED RIGHT ONE PLACE WITH ZERO  
 00547 \* FILLED IN FROM THE LEFT SIDE;  
 00548 \* DECREMENT COUNT;  
 00549 \* EXIT IF DONE ELSE GO TO DIV3;  
 00550 \*  
 00551 \*  
 00552 \*  
 00553 \*  
 00554 \*  
 00555 FFAF 36 DIVIDE PSH A - LOADS DIVIDEND INTO X3,4.  
 00556 FFB0 37 PSH B  
 00557 FFB1 96 E6 LDA A AR1 LOADS DIVISOR FROM ARA.  
 00558 FFB3 D6 07 LDA B AR0  
 00559 FFB5 37 PSH B - PUTS DIVISOR INTO X1,2.  
 00560 FFB6 36 PSH A  
 00561 FFB7 34 DES SET UP SPACE FOR COUNT.  
 00562 FFB8 30 TSX INDEX GETS STACK POINTER  
 00563 FFB9 86 E1 LDA A #1 INITIALIZE COUNT.  
 00564 FFBB 60 E1 TST 1,X TESTS FOR HI DIVISOR BIT ON  
 00565 FFED 2B 06 BMI DIV2 SKIPS IF ON.  
 00566 FFBF 4C DIV1 INC A - COUNTS LEADING ZEROS.  
 00567 FFC0 68 02 ASL 2,X LEFT JUSTIFIES X1,2.  
 00568 FFC2 69 E1 ROL 1,X  
 00569 FFC4 2B E4 BMI DIV2 SKIPS IF NO LEADING ZERO.  
 00570 FFC6 81 11 CNP A #17 TESTS FOR ALL ZERO DIVISOR.  
 00571 FFC8 26 F5 BNE DIV1 GOES BACK IF BITS LEFT.  
 00572 FFCA A7 00 DIV2 STA A 0,X SETS COUNTER.  
 00573 FFCC E6 E3 LDA B 3,X BA GETS ORIGINAL  
 00574 FFCE A6 E4 LDA A 4,X DIVIDEND VALUE.  
 00575 FFDD 6F E3 CLR 3,X CLEARS X3,4 FOR FORMATION  
 00576 FFD2 6F E4 CLR 4,X OF THE QUOTIENT.

00578 FFD4 AB 02	DIV3	SUB A 2,X	START OF DIVIDE LOOP.
00579 FFD6 E2 01		SBC B 1,X	
00580 FFDB 24 07		BCC DIV4	SKIP IF DIVIDEND < DIVISOR.
00581 FFDA AB 02		ADD A 2,X	RESTORES DIVIDEND IN BA.
00582 FFDC E9 01		ADC B 1,X	
00583 FFDE 0C		CLC	CLEARS THE CARRY.
00584 FFDF 20 01		BRA DIV5	SKIPS WITH CARRY CLEAR.
00585 FFE1 0D	DIV4	SEC	SETS CARRY TO 1.
00586 FFE2 69 04	DIV5	ROL 4,X	SHIFT CARRY INTO
00587 FFE4 69 03		ROL 3,X	QUOTIENT X3.4
00588 FFE6 64 01		LSR 1,X	SHIFTS DIVISOR X1.2
00589 FFE8 66 02		ROR 2,X	RIGHT ONE.
00590 FFEA 6A 00		DEC 0,X	DECREMENTS COUNTER.
00591 FFEC 26 E6		BNE DIV3	GOES BACK IF NOT DONE.
00592 FFEE D7 E6		STA B AR1	STORES REMAINDER IN BAA.
00593 FFF0 97 07		STA A AR0	
00594 FFF2 31		INS	CLEANS UP THE STACK.
00595 FFF3 31		INS	
00596 FFF4 31		INS	
00597 FFF5 33		PUL B	STORES QUOTIENT IN BA.
00598 FFF6 32		PUL A	
00599 FFF7 39	*	RTS	EXITS ROUTINE.
00600	*		
00601	*		
00602	*		
00603 FFFF8 0104	IRQ	FDB \$0104	INTERRUPT REQUEST VECTOR.
34 FFFFA FE4FI	SWI	FDB BKENTR	SOFTWARE INT. VECTOR ADDR.
00605 FFFFC 0108	NMI	FDB \$0108	NON-MASKABLE-INT. VECT.
00606 FFFE FC000	RST	FDB \$FC00	RESTART VECTOR ADDRESS.
00607	*		
00608	*		
00609	*		
00610	*		END OF PDS SOURCE LISTING.
00611	*		
00612	*		
00613			END
TOTAL ERRORS 00000			