

000.001

```

2
3 .SYS. EQU 1 ASSEMBLE SYSTEM USER'S CODE
4 LON 1 SHOW IF-SKIPPED LINES
5
6 *** PATCH - PATCH SYSTEM AND USER FILES.
7 *
8 * PATCH IS USED TO PATCH BINARY (ABS AND PIC) FILES ON THE
9 * SYSTEM. PATCH RUNS IN TWO MODES, *USER* AND *SYSTEM*.
10 *
11 * IF THE FILE TO BE PATCHED HAS A PATCH HISTORY TABLE ON THE
12 * END OF IT, PATCH RUNS IN SYSTEM MODE. OTHERWISE, PATCH RUNS
13 * IN USER MODE.
14 *
15 * SYSTEM MODE:
16 *
17 *
18 * IN SYSTEM MODE, PATCH WILL VIOLATE SOFTWARE WRITE-PROTECTION ON
19 * THE FILES. FOR THIS REASON, PATCH IS VERY PARTICULAR ABOUT ENTERING
20 * PATCHES.
21 *
22 * FIRST, A PATCH SERIES CODE MUST BE GIVEN. THIS CODE REPRESENTS A
23 * PATCH SERIES BYTE, AND A CRC-16 OF THAT BYTE.
24 * NEXT, A PATCH PREREQUISITE CODE IS REQUIRED. THIS CODE REPRESENTS
25 * A 6 BYTE FIELD, WITH A BIT SET FOR EVERY PREREQUISITE PATCH NEEDED.
26 * THE PSC IS FOLLOWED BY A CRC-16 OF THE PREREQUISITE BYTES PRECEDED
27 * BY THE PATCH SERIES BYTE.
28 *
29 * AFTER THE PATCHES ARE ENTERED, A PATCH CHECK CODE MUST BE ENTERED.
30 * THIS PRODUCES A 2 BYTE CRC-16 FOR ALL OF THE ABOVE ENTERED VALUES,
31 * AND ALSO ENCLUDES A CTC-16 FOR THE PATCH CHECK CODE ITSELF.
32 *
33 * IF ALL OF THESE CODES ARE ALL OK, THEN THE PATCH IS MADE.
34 *
35 * USER MODE:
36 *
37 *
38 * IN USER MODE, THE USER SPECIFIES THE PROGRAM NAME, AND THE PATCHES.
39 * WHEN HE CTL-D'S, THE PATCHES ARE ENTERED. A PROGRAM MUST BE WRITE
40 * ACCESSABLE FOR PATCH TO WORK IN USER MODE.
41
42

```

43 **** ASSEMBLY CONSTANTS

000.000

```

45 CN.FIL EQU 0 PATCH FILE CHANNEL NUMBER
46

```

47 ****

000.000

```

48
49 XTEXT ECDEF

```

ECDEF

16:23:54 16-MAY-80

51X ** ERROR CODE DEFINITIONS.

000.000	53X	ORG	0	
000.000	54X	DS	1	NO ERROR #0
000.001	55X EC.EOF	DS	1	END OF FILE
000.002	56X EC.EOM	DS	1	END OF MEDIA
000.003	57X EC.ILC	DS	1	ILLEGAL SYSCALL CODE
000.004	58X EC.CNA	DS	1	CHANNEL NOT AVAILABLE
000.005	59X EC.DNS	DS	1	DEVICE NOT SUITABLE
000.006	60X EC.IDN	DS	1	ILLEGAL DEVICE NAME
000.007	61X EC.IFN	DS	1	ILLEGAL FILE NAME
000.010	62X EC.NRD	DS	1	NO ROOM FOR DEVICE DRIVER
000.011	63X EC.FND	DS	1	CHANNEL NOT OPEN
000.012	64X EC.ILR	DS	1	ILLEGAL REQUEST
000.013	65X EC.FUC	DS	1	FILE USAGE CONFLICT
000.014	66X EC.FNF	DS	1	FILE NAME NOT FOUND
000.015	67X EC.UND	DS	1	UNKNOWN DEVICE
000.016	68X EC.ICN	DS	1	ILLEGAL CHANNEL NUMBER
000.017	69X EC.DIF	DS	1	DIRECTORY FULL
000.020	70X EC.IFC	DS	1	ILLEGAL FILE CONTENTS
000.021	71X EC.NEM	DS	1	NOT ENOUGH MEMORY
000.022	72X EC.RF	DS	1	READ FAILURE
000.023	73X EC.WF	DS	1	WRITE FAILURE
000.024	74X EC.WPV	DS	1	WRITE PROTECTION VIOLATION
000.025	75X EC.WP	DS	1	DISK WRITE PROTECTED
000.026	76X EC.FAP	DS	1	FILE ALREADY PRESENT
000.027	77X EC.DDA	DS	1	DEVICE DRIVER ABORT
000.030	78X EC.FL	DS	1	FILE LOCKED
000.031	79X EC.FAD	DS	1	FILE ALREADY OPEN
000.032	80X EC.IS	DS	1	ILLEGAL SWITCH
000.033	81X EC.UUN	DS	1	UNKNOWN UNIT NUMBER
000.034	82X EC.FNR	DS	1	FILE NAME REQUIRED
000.035	83X EC.DIW	DS	1	DEVICE IS NOT WRITABLE (OR WRITE LOCKED)
000.036	84X EC.UNA	DS	1	UNIT NOT AVAILABLE
000.037	85X EC.ILV	DS	1	ILLEGAL VALUE
000.040	86X EC.ILO	DS	1	ILLEGAL OPTION
000.041	87X EC.VPM	DS	1	VOLUME PRESENTLY MOUNTED ON DEVICE
000.042	88X EC.NVM	DS	1	NO VOLUME PRESENTLY MOUNTED
000.043	89X EC.FOD	DS	1	FILE OPEN ON DEVICE
000.044	90X EC.NPM	DS	1	NO PROVISIONS MADE FOR REMOUNTING MORE DISKS
000.045	91X EC.DNI	DS	1	DISK NOT INITIALIZED
000.046	92X EC.DNR	DS	1	DISK IS NOT READABLE
000.047	93X EC.DSC	DS	1	DISK STRUCTURE IS CORRUPT
000.050	94X EC.NCV	DS	1	NOT CORRECT VERSION OF HDOS
000.051	95X EC.NOS	DS	1	NO OPERATING SYSTEM MOUNTED
000.052	96X EC.IOI	DS	1	ILLEGAL OVERLAY INDEX
000.053	97X EC.OTL	DS	1	OVERLAY TOO LARGE
000.054	98	XTEXT	DIRDEF	

DIR

16:23:58 16-MAY-80

```

100X **      DIRECTORY ENTRY FORMAT.
101X
000.000      102X      ORG      0
103X
104X
000.377      105X DF.EMP EQU      3770      FLAGS ENTRY EMPTY
000.376      106X DF.CLR EQU      3760      FLAGS ENTRY EMPTY, REST OF DIR ALSO CLEAR
107X
000.000      108X DIR.NAM DS      8      NAME
000.010      109X DIR.EXT DS      3      EXTENSION
000.013      110X DIR.PRO DS      1      PROJECT
000.014      111X DIR.VER DS      1      VERSION
000.015      112X DIRIDL EQU      *      FILE IDENTIFICATION LENGTH
113X
000.015      114X DIR.CLU DS      1      CLUSTER FACTOR
000.016      115X DIR.FLG DS      1      FLAGS
000.017      116X      DS      1      RESERVED
000.020      117X DIR.FGN DS      1      FIRST GROUP NUMBER
000.021      118X DIR.LGN DS      1      LAST GROUP NUMBER
000.022      119X DIR.LSI DS      1      LAST SECTOR INDEX (IN LAST GROUP)
000.023      120X DIR.CRD DS      2      CREATION DATE
000.025      121X DIR.ALD DS      2      LAST ALTERATION DATE
122X
000.027      123X DIRELEN EQU      *      DIRECTORY ENTRY LENGTH
000.027      124      XTEXT      IOCDEF

126X **      I/O CHANNEL DEFINITIONS.
127X
000.000      128X      ORG      0
129X
000.000      130X IOC.LNK DS      2      ADDRESS OF NEXT CHANNEL, =0 IF LAST
000.002      131X IOC.DDA DS      2      THREAD JUMP TO DEVICE DRIVER (VIA DEV. TABLE)
132X
000.004      133X IOC.FLG DS      1      FILE TYPE FLAGS
000.001      134X FT.DD EQU      00000001B      =1 IF DIRECTORY DEVICE
000.002      135X FT.OR EQU      00000010B      =1 IF OPEN FOR READ
000.004      136X FT.OW EQU      00000100B      =1 IF OPEN FOR WRITE
000.010      137X FT.OU EQU      00001000B      =1 IF OPEN FOR UPDATE
000.003      138X IOC.SQL EQU      *-IOC.DDA      LENGTH OF INFO FOR SEQUENTIAL FILE (FROM IOC)
139X
000.005      140X IOC.GRT DS      2      ADDRESS OF GROUP RESERVATION TABLE
000.007      141X IOC.SPG DS      1      SECTORS PER GROUP, THIS DEVICE
000.010      142X IOC.CGN DS      1      CURRENT GROUP NUMBER
000.011      143X IOC.CSI DS      1      CURRENT SECTOR INDEX (IN CURRENT GROUP)
000.012      144X IOC.LGN DS      1      LAST GROUP NUMBER
000.013      145X IOC.LSI DS      1      LAST SECTOR INDEX (IN LAST GROUP)
000.010      146X IOC.DRL EQU      *-IOC.FLG      LENGTH OF INFO NORMALLY COPIED BACK TO
147X *      THE CHANNEL TABLE
000.014      148X IOC.DTA DS      2      DEVICE TABLE ADDRESS FOR THIS DEVICE
000.016      149X IOC.DES DS      2      SECTOR NUMBER OF DIRECTORY ENTRY
000.020      150X IOC.DEV DS      2      DEVICE CODE
000.022      151X IOC.UNI DS      1      UNIT NUMBER (0-9)
000.021      152X IOC.DIL EQU      *-IOC.DDA      LENGTH OF INFO FOR DIRECTORY FILE (FROM IOC)
153X

```

IOC

16:24:02 16-MAY-80

000.023	154X	IOC.DIR DS	DIRELEN	DIRECTORY ENTRY
	155X			
000.052	156X	IOCELEN EQU	*	IOC ENTRY LENGTH
	157X			
000.001	158X	IOCTD EQU	1	INDEX OF USER CHANNEL #0 IN CHANTAB (FIRST = 0)
000.052	159	XTEXT	DEVDEF	

	161X	**	DEVICE TABLE ENTRIES.	
	162X			
000.000	163X		ORG	0
	164X			
000.000	165X	DEV.NAM DS	2	DEVICE NAME
000.000	166X	DV.EL EQU	00000000B	END OF DEVICE LIST FLAG
000.001	167X	DV.NU EQU	00000001B	DEVICE ENTRY NOT IN USE
	168X			
000.002	169X	DEV.RES DS	1	DRIVER RESIDENCE CODE
000.001	170X	DR.IM EQU	00000001B	DRIVER IN MEMORY
000.002	171X	DR.PR EQU	00000010B	DRIVER PERMINANTLY RESIDENT
	172X			
000.003	173X	DEV.JMP DS	1	JMP TO PROCESSOR
000.004	174X	DEV.DDA DS	2	DRIVER ADDRESS
000.006	175X	DEV.FLG DS	1	FLAG BYTE
000.001	176X	DT.DD EQU	00000001B	DIRECTORY DEVICE
000.002	177X	DT.CR EQU	00000010B	CAPABLE OF READ OPERATION
000.004	178X	DT.CW EQU	00000100B	CAPABLE OF WRITE OPERATION
	179X			
000.007	180X	DEV.SPG DS	1	SECTORS PER GROUP THIS DEVICE
000.010	181X	DEV.MUM DS	1	MOUNTED UNIT MASK
000.011	182X	DEV.MNU DS	1	MAXIMUM NUMBER OF UNITS
000.012	183X	DEV.UNT DS	2	ADDRESS OF UNIT SPECIFIC DATA TABLE
	184X			
000.014	185X	DEV.DVL DS	2	DRIVER BYTE LENGTH
000.016	186X	DEV.DVG DS	1	DRIVER ROUTINE GROUP ADDRESS
	187X			
000.017	188X	DEVELEN EQU	*	DEVICE TABLE ENTRY LENGTH

	190X	**	UNIT SPECIFIC DEVICE DATA TABLE ENTRIES	
	191X			
000.000	192X		ORG	0
	193X			
000.000	194X	UNT.FLG DS	1	UNIT SPECIFIC *DEV.FLG*
000.001	195X	UNT.GRT DS	2	ADDRESS OF GROUP RESERVATION TABLE (IF DT.DD)
000.003	196X	UNT.GTS DS	2	GRT SECTOR NUMBER
000.005	197X	UNT.DIS DS	2	DIRECTORY FIRST SECTOR NUMBER
	198X			
000.007	199X	UNT.SIZ EQU	*	SIZE OF UNIT SPECIFIC DATA TABLE PER UNIT
000.007	200	XTEXT	HOSDEF	

```
202X **      HOSDEF - DEFINE HOS PARAMETER.
203X *
204X
205X
000.026      206X VERS      EQU      1*16+6      VERSION 1.6
207X
000.377      208X SYSCALL EQU      377R      SYSCALL INSTRUCTION
209X
000.000      210X
211X          ORG      0
212X
213X *      RESIDENT FUNCTIONS
214X
000.000      215X .EXIT      DS      1      EXIT (MUST BE FIRST)
000.001      216X .SCIN      DS      1      SCIN
000.002      217X .SCOUT     DS      1      SCOUT
000.003      218X .PRINT     DS      1      PRINT
000.004      219X .READ      DS      1      READ
000.005      220X .WRITE     DS      1      WRITE
000.006      221X .CONSL     DS      1      SET/CLEAR CONSOLE OPTIONS
000.007      222X .CLRCD     DS      1      CLEAR CONSOLE BUFFER
000.010      223X .LOADD     DS      1      LOAD AN OVERLAY
000.011      224X .VERS      DS      1      RETURN HDOS VERSION NUMBER
000.012      225X .SYSRES    DS      1      PRECEDING FUNCTIONS ARE RESIDENT
226X
227X
228X *      *HDOSOVLO.SYS* FUNCTIONS
229X
000.040      230X          ORG      40A
231X
000.040      232X .LINK      DS      1      LINK (MUST BE FIRST)
000.041      233X .CTLCD     DS      1      CTL-C
000.042      234X .OPENR     DS      1      OPENR
000.043      235X .OPENW     DS      1      OPENW
000.044      236X .OPENU     DS      1      OPENU
000.045      237X .OPENC     DS      1      OPENC
000.046      238X .CLOSE     DS      1      CLOSE
000.047      239X .POSIT     DS      1      POSITION
000.050      240X .DELET     DS      1      DELETE
000.051      241X .RENAM     DS      1      RENAME
000.052      242X .SETTP     DS      1      SETTOP
000.053      243X .DECODE     DS      1      NAME DECODE
000.054      244X .NAME      DS      1      GET FILE NAME FROM CHANNEL
000.055      245X .CLEAR     DS      1      CLEAR CHAN
000.056      246X .CLEARA    DS      1      CLEAR ALL CHANS
000.057      247X .ERROR     DS      1      LOOKUP ERROR
000.060      248X .CHFLG     DS      1      CHANGE FLAGS
000.061      249X .DISMT     DS      1      FLAG SYSTEM DISK DISMOUNTED
000.062      250X .LOADD     DS      1      LOAD DEVICE DRIVER
251X
252X
253X *      *HDOSOVLI.SYS* FUNCTIONS
254X
000.200      255X          ORG      200R
256X
000.200      257X .MOUNT     DS      1      MOUNT (MUST BE FIRST)
```

HOSDEF

16:24:06 16-MAY-80

000.201	258X	.DISMOUNT	DS	1	DISMOUNT
000.202	259X	.MOUNT	DS	1	MOUNT/NO MESSAGE
000.203	260X	.DISMOUNT	DS	1	DISMOUNT/NO MESSAGE
000.204	261X	.RESET	DS	1	RESET = DISMOUNT/MOUNT OF UNIT
000.205	262		XTEXT	HOSEQU	

264X ** HDOS SYSTEM EQUIVALENCES.

	265X	*			
	266X				
024.000	267X	S.GRT0	EQU	24000A	SYSTEM AREA FOR GRT0
025.000	268X	S.GRT1	EQU	25000A	SYSTEM AREA FOR GRT1
026.000	269X	S.GRT2	EQU	26000A	SYSTEM AREA FOR GRT2
	270X				
030.000	271X	ROMBOOT	EQU	30000A	ROM BOOT ENTRY
	272X				
040.100	273X		ORG	40100A	FREE SPACE FROM PAM-8
	274X				
040.100	275X		DS	8	JUMP TO SYSTEM EXIT
040.110	276X	D.CON	DS	16	DISK CONSTANTS
040.130	277X	SYDD	EQU	*	SYSTEM DISK ENTRY POINT
040.130	278X	D.VEC	DS	24*3	SYSTEM ROM ENTRY VECTORS
040.240	279X	D.RAM	DS	31	SYSTEM ROM WORK AREA
040.277	280X	S.VAL	DS	36	SYSTEM VALUES
040.343	281X	S.INT	DS	115	SYSTEM INTERNAL WORK AREAS
041.126	282X		DS	16	
041.146	283X	S.SOVR	DS	2	STACK OVERFLOW WARNING
041.150	284X		DS	42200A-*	SYSTEM STACK
001.032	285X	STACKL	EQU	*-S.SOVR	STACK SIZE
	286X				
042.200	287X	STACK	EQU	*	LWA+1 SYSTEM STACK
042.200	288X	USERFWA	EQU	*	USER FWA
042.200	289		XTEXT	ESVAL	

291X ** S.VAL - SYSTEM VALUE DEFINITIONS.

	292X	*			
	293X	*			THESE VALUES ARE SET AND MAINTAINED BY THE SYSTEM.
	294X	*			
	295X	*			THE DECK HOSEQU MUST BE MODIFIED WHEN THIS IS MODIFIED.
	296X				
	297X				
040.277	298X		ORG	S.VAL	
	299X				
040.277	300X	S.DATE	DS	9	SYSTEM DATE (IN ASCII)
040.310	301X	S.DATC	DS	2	CODED DATE
040.312	302X	S.TIME	DS	4	TIME FROM MIDNIGHT (IN TICS)
040.316	303X	S.HIMEM	DS	2	HARDWARE HIGH MEMORY ADDRESS+1
	304X				
040.320	305X	S.SYSM	DS	2	FWA RESIDENT SYSTEM
	306X				
040.322	307X	S.USRM	DS	2	LWA USER MEMORY

```

040.324 308X
309X S.OMAX DS 2 MAX OVERLAY SIZE FOR SYSTEM
310X
311X
312X ** THE FOLLOWING FIVE CELLS SHOULD BE MODIFIED/READ ONLY VIA THE .CONSL SYSCALL
313X
000.200 314X CSL.ECH EQU 10000000B SUPPRESS ECHO
000.002 315X CSL.WRP EQU 00000010B WRAP LINES AT WIDTH
000.001 316X CSL.CHR EQU 00000001B OPERATE IN CHARACTER MODE
317X
000.000 318X I.CSLMD EQU 0 S.CSLMD IS FIRST BYTE
040.326 319X S.CSLMD DS 1 CONSOLE MODE
320X
000.200 321X CTP.BKS EQU 10000000B TERMINAL PROCESSES BACKSPACES
000.040 322X CTP.MLI EQU 00100000B MAP LOWER CASE TO UPPER ON INPUT
000.020 323X CTP.MLO EQU 00010000B MAP LOWER CASE TO UPPER ON OUTPUT
000.010 324X CTP.2SB EQU 00001000B TERMINAL NEEDS TWO STOP BITS
000.002 325X CTP.BKM EQU 00000010B MAP BKSP (UPON INPUT) TO RUBOUT
000.001 326X CTP.TAB EQU 00000001B TERMINAL SUPPORTS TAB CHARACTERS
327X
000.001 328X I.CONTY EQU 1 S.CONTY IS 2ND BYTE
000.000 329X ERRNZ *-S.CSLMD-I.CONTY
040.327 330X S.CONTY DS 1 CONSOLE TYPE FLAGS
000.002 331X I.CUSOR EQU 2 S.CUSOR IS 3RD BYTE
000.000 332X ERRNZ *-S.CSLMD-I.CUSOR
040.330 333X S.CUSOR DS 1 CURRENT CURSOR POSITION
000.003 334X I.CONWI EQU 3 S.CONWI IS 4TH BYTE
000.000 335X ERRNZ *-S.CSLMD-I.CONWI
040.331 336X S.CONWI DS 1 CONSOLE WIDTH
337X
000.001 338X CO.FLG EQU 00000001B CTL-D FLAG
000.200 339X CS.FLG EQU 10000000B CTL-S FLAG
340X
000.004 341X I.CONFL EQU 4 S.CONFL IS 5TH BYTE
000.000 342X ERRNZ *-S.CSLMD-I.CONFL
040.332 343X S.CONFL DS 1 CONSOLE FLAGS
344X
040.333 345X S.CAADR DS 2 ADDRESS FOR ABORT PROCESSING (>256 IF VALID)
040.335 346X S.CCTAB DS 6 ADDR FOR CTL-A, CTL-B, CTL-C PROCESSING
040.343 347 XTEXT ESINT

349X ** S.INT - SYSTEM INTERNAL WORKAREA DEFINITIONS.
350X *
351X * THESE CELLS ARE REFERENCED BY OVERLAYS AND MAIN CODE, AND
352X * MUST THEREFORE RESIDE IN FIXED LOW MEMORY.
353X
354X
040.343 355X ORG S.INT
356X
357X ** CONSOLE STATUS FLAGS
358X
040.343 359X S.CDB DS 1 CONSOLE DESCRIPTOR BYTE
000.000 360X CDB.HB5 EQU 00000000B

```

ESINT

16:24:13 16-MAY-80

000.001	361X	CDB.HB4	EQU	00000001B	=0 IF HB-5, =1 IF HB-4
040.344	362X	S.BAUD	DS	2	[0-14] HB-4 BAUD RATE, =0 IF HB-5
	363X	*			[15] =1 IF BAUD RATE => 2 STOP BITS
	364X				
	365X	**			TABLE ADDRESS WORDS
	366X				
040.346	367X	S.DLINK	DS	2	ADDRESS OF DATA IN HDOS CODE
040.350	368X	S.OFWA	DS	2	FWA OVERLAY TABLE
040.352	369X	S.CFWA	DS	2	FWA CHANNEL TABLE
040.354	370X	S.DFWA	DS	2	FWA DEVICE TABLE
040.356	371X	S.RFWA	DS	2	FWA RESIDENT HDOS CODE
	372X				
	373X	**			DEVICE DRIVER DELAYED LOAD FLAGS
	374X				
040.360	375X	S.DDLDA	DS	2	DRIVER LOAD ADDRESS (HIGH BYTE=0 IF NO LOAD PENDING)
040.362	376X	S.DDLEN	DS	2	CODE LENGTH IN BYTES
040.364	377X	S.DDGRP	DS	1	GROUP NUMBER FOR DRIVER
040.365	378X		DS	1	HOLD PLACE
	379X	*S.DDSEC	DS	2	SECTOR NUMBER FOR DRIVER (* OBSOLETE ! *)
040.366	380X	S.DDDTA	DS	2	DEVICE'S ADDRESS IN DEVLST +DEV.RES
040.370	381X	S.DDOPC	DS	1	OPEN OP CODE PENDING
	382X				
	383X	**			OVERLAY MANAGEMENT FLAGS
	384X				
000.001	385X	OVL.IN	EQU	00000001B	IN MEMORY
000.002	386X	OVL.RES	EQU	00000010B	PERMANENTLY RESIDENT
000.014	387X	OVL.NUM	EQU	00001100B	OVERLAY NUMBER MASK
000.200	388X	OVL.UCS	EQU	10000000B	USER CODE SWAPPED FOR OVERLAY
	389X				
040.371	390X	S.OVLFL	DS	1	OVERLAY FLAG
040.372	391X	S.UCSF	DS	2	FWA SWAPPED USER CODE
040.374	392X	S.UCSL	DS	2	LENGTH SWAPPED USER CODE
040.376	393X	S.OVLS	DS	2	SIZE OF OVERLAY CODE
041.000	394X	S.OVLE	DS	2	ENTRY POINT OF OVERLAY CODE
	395X				
041.002	396X	S.SSN	DS	2	SWAP AREA SECTOR NUMBER
041.004	397X	S.OSN	DS	2	OVERLAY SECTOR NUMBER
	398X				
	399X	*			SYSCALL PROCESSING WORK AREAS
	400X				
041.006	401X	S.CACC	DS	1	(ACC) UPON SYSCALL
041.007	402X	S.CODE	DS	1	SYSCALL INDEX IN PROGRESS
	403X				
	404X	*			JUMPS TO ROUTINES IN RESIDENT HDOS CODE
	405X				
041.010	406X	S.JUMPS	DS	0	START OF DUMP VECTORS
041.010	407X	S.SDD	DS	3	JUMP TO STAND-IN DEVICE DRIVER
041.013	408X	S.FASER	DS	3	JUMP TO FATSERR (FATAL SYSTEM ERROR)
041.016	409X	S.DIREA	DS	3	JUMP TO DIREAD (DISK FILE READ)
041.021	410X	S.FCI	DS	3	JUMP TO FCI (FETCH CHANNEL INFO)
041.024	411X	S.SCI	DS	3	JUMP TO SCI (STORE CHANNEL INFO)
041.027	412X	S.GUP	DS	3	JUMP TO GUP (GET UNIT POINTER)
	413X				
041.032	414X	S.MOUNT	DS	1	<>0 IF THE SYSTEM DISK IS MOUNTED
041.033	415X	S.DCS	DS	1	DEFAULT CLUSTER SIZE-1
	416X				

ESINT

16:24:14 16-MAY-80

041.034	417X S.BOOTF DS	1	BOOT FLAGS
000.001	418X BOOT.P EQU	00000001B	EXECUTE PROLOGUE UPON BOOTUP
	419X		
	420X *		STACK VALUE SAVED FOR OVERLAY SYSCALLS
	421X		
041.035	422X S.OVSTK DS	2	VALUE OF SP UPON SYSCALLS USING OVERLAY
	423X		
041.037	424X	DS 1	RESERVED
	426X **		ACTIVE I/O AREA.
	427X *		
	428X *		THE AIO.XXX AREA CONTAINS INFORMATION ABOUT THE I/O OPERATION
	429X *		CURRENTLY BEING PERFORMED. THE INFORMATION IS OBTAINED FROM
	430X *		THE CHANNEL TABLE, AND WILL BE RESTORED THERE WHEN DONE.
	431X *		
	432X *		NORMALLY, THE AIO.XXX INFORMATION WOULD BE OBTAINED DIRECTLY
	433X *		FROM VARIOUS SYSTEM TABLES VIA POINTER REGISTERS. SINCE THE
	434X *		8080 HAS NO GOOD INDEXED ADDRESSING, THE DATA IS MANUALLY
	435X *		COPIED INTO THE AIO.XXX CELLS BEFORE PROCESSING, AND
	436X *		BACKDATED AFTER PROCESSING.
	437X		
041.040	438X AIO.VEC DS	3	JUMP INSTRUCTION
041.041	439X AIO.DDA EQU	*-2	DEVICE DRIVER ADDRESS
041.043	440X AIO.FLB DS	1	FLAG BYTE
041.044	441X AIO.GRT DS	2	ADDRESS OF GROUP RESERV. TABLE
041.046	442X AIO.SPG DS	1	SECTORS PER GROUP
041.047	443X AIO.CGN DS	1	CURRENT GROUP NUMBER
041.050	444X AIO.CSI DS	1	CURRENT SECTOR INDEX
041.051	445X AIO.LGN DS	1	LAST GROUP NUMBER
041.052	446X AIO.LSI DS	1	LAST SECTOR INDEX
041.053	447X AIO.DTA DS	2	DEVICE TABLE ADDRESS
041.055	448X AIO.DES DS	2	DIRECTORY SECTOR
041.057	449X AIO.DEV DS	2	DEVICE CODE
041.061	450X AIO.UNI DS	1	UNIT NUMBER (0-9)
	451X		
041.062	452X AIO.DIR DS	DIRELEN	DIRECTORY ENTRY
	453X		
041.111	454X AIO.CNT DS	1	SECTOR COUNT
041.112	455X AIO.EOM DS	1	END OF MEDIA FLAG
041.113	456X AIO.EOF DS	1	END OF FILE FLAG
041.114	457X AIO.TFP DS	2	TEMP FILE POINTERS
041.116	458X AIO.CHA DS	2	ADDRESS OF CHANNEL BLOCK (IOC.DDA)
041.120	460X S.SCR DS	2	SYSTEM SCRATCH AREA ADDRESS
041.122	461	XTEXT ASCII	

ASCII

16:24:18 16-MAY-80

463X ** ASCII CHARACTER EQUIVALENCES.

000.015	464X				
000.012	465X CR	EQU	13		CARRIAGE RETURN
000.200	466X LF	EQU	10		LINE FEED
000.000	467X NULL	EQU	2000		PAD CHARACTER
000.007	468X NUL2	EQU	0		
000.177	469X BELL	EQU	7		BELL CHARACTER
000.010	470X RUBOUT	EQU	177Q		
000.026	471X BKSP	EQU	10Q		CTL-H
000.002	472X C.SYN	EQU	26Q		SYNC
000.047	473X C.STX	EQU	2		STX
000.011	474X QUOTE	EQU	47Q		
000.033	475X TAB	EQU	11Q		
000.012	476X ESC	EQU	33Q		
000.212	477X NL	EQU	12Q		NEW LINE (HDOS SYSTEMS)
000.014	478X ENL	EQU	NL+200Q		NL + END-OF-LINE-FLAG
000.001	479X FF	EQU	14Q		FORM FEED
000.002	480X CTLA	EQU	01Q		CTL-A
000.003	481X CTLB	EQU	02Q		CTL-B
000.004	482X CTLC	EQU	03Q		CTL-C
000.017	483X CTLD	EQU	04Q		CTL-D
000.020	484X CTLO	EQU	17Q		CTL-O
000.021	485X CTLP	EQU	20Q		CTL-P
000.023	486X CTLQ	EQU	21Q		CTL-Q
000.032	487X CTLS	EQU	23Q		CTL-S
041.122	488X CTLZ	EQU	32Q		CTL-Z
	489	XTEXT	PICDEF		

491X ** PIC FORMAT EQUIVALENCES.

000.000	492X				
	493X	ORG	0		
	494X				
000.000	495X PIC.ID	DS	1		377Q = BINARY FILE FLAG
000.001	496X	DS	1		FILE TYPE (FT.PIC)
000.002	497X PIC.LEN	DS	2		LENGTH OF ENTIRE RECORD
000.004	498X PIC.PTR	DS	2		INDEX OF START OF PIC TABLE
	499X				
000.006	500X PIC.COD	DS	0		CODE STARTS HERE
000.006	501	XTEXT	ABSDEF		

503X ** ABS FORMAT EQUIVALENCES.

000.000	504X				
	505X	ORG	0		
	506X				
000.000	507X ABS.ID	DS	1		377Q = BINARY FILE FLAG
000.001	508X	DS	1		FILE TYPE (FT.ABS)
000.002	509X ABS.LDA	DS	2		LOAD ADDRESS
000.004	510X ABS.LEN	DS	2		LENGTH OF ENTIRE RECORD
000.006	511X ABS.ENT	DS	2		ENTRY POINT
	512X				
000.010	513X ABS.COD	DS	0		CODE STARTS HERE
000.010	514	XTEXT	FILDEF		

FILDEF

16:24:28 16-MAY-80

516X ** FILDEF - FILE TYPE DEFINITIONS.

517X *

518X * DB 377Q,FT.XXX

519X

520X

000.000

521X FT.ABS EQU 0

ABSOLUTE BINARY

000.001

522X FT.PIC EQU 1

POSITION INDEPENDANT CODE

000.002

523X FT.REL EQU 2

RELOCATABLE CODE

000.003

524X FT.BAC EQU 3

COMPILED BASIC CODE

525

042.170

526 ORG USERFWA-ABS.COD

042.170 377.000

527 DB 377Q,FT.ABS

042.172 200 042

528 DW USERFWA LOAD ADDR

042.174 171 011

529 DW MEML-USERFWA LOAD SIZE

042.176 200 042

530 DW START ENTRY

```

533 *** PATCH - MAIN ROUTINE.
534 *
535 * ENTERED HERE FROM *PRS*
536
537
042.200 538 PATCH EQU *
042.200 539 START EQU * PROGRAM ENTERS HERE
042.200 540 RESTART EQU *
541
042.200 061 200 042 542 LXI SP,STACK CLEAN STACK
042.203 315 372 042 543 CALL PRS PRESET
042.206 315 214 042 544 CALL PATCH1 DO THE PATCH
042.211 303 200 042 545 JMP RESTART DO ANOTHER
546
042.214 315 144 043 547 PATCH1 CALL OFF OPEN FILE FOR PATCH
042.217 322 225 042 548 JNC PATCH2 GOT NAME
042.222 257 549 XRA A
042.223 377 000 550 DB SYSCALL,.EXIT CTL-D, EXIT
551
042.225 552 PATCH2 EQU *
000.001 553 IF .SYS. SYSTEM OPTIONS
554 LDA PHTSWI
555 ANA A
556 JNZ APF ADD PHT TO FILE
557 ENDIF
042.225 072 261 053 558 LDA PHIST
042.230 247 559 ANA A
042.231 312 244 042 560 JZ PATCH3 USER MODE, NO PHT
042.234 315 211 044 561 CALL GFI GET PATCH ID
042.237 330 562 RC ERROR
042.240 315 315 044 563 CALL GPC GET PREREQ CODE
042.243 330 564 RC CTL-D
565
566 * GET NEXT ADDRESS
567
042.244 315 215 045 568 PATCH3 CALL GPA GET PATCH ADDRESS
042.247 332 260 042 569 JC PATCH5 GOT CTL-D, ENTER PATCH
042.252 315 021 046 570 PATCH4 CALL APP ACCEPT PROGRAM PATCHES
042.255 303 244 042 571 JMP PATCH3 CTL-D
572
573 * ALL DONE. GET CHECKSUM, IF NEEDED
574
042.260 072 261 053 575 PATCH5 LDA PHIST
042.263 247 576 ANA A
042.264 304 212 046 577 CNZ CFC CHECK PATCH CRC, IF SYSTEM MODE
042.267 330 578 RC ERROR
579
580 * OFF CTL-C'S, START MODIFYING FILE
581
042.270 041 000 000 582 LXI H,0
042.273 076 003 583 MVI A,CTL-C
042.275 377 041 584 DB SYSCALL,.CTLC DISABLE CTL-C'S
042.277 315 101 047 585 CALL EFF ENTER PATCHES IN FILE
042.302 315 023 050 586 CALL FVB FLUSH VIEW BUFFER
042.305 315 025 051 587 CALL UBH UPDATE BINARY HEADER
042.310 072 261 053 588 LDA PHIST

```

```

042.313 247      589      ANA      A
042.314 304 141 051 590      CNZ      WPH      WRITE PATCH HISTORY, IF ANY
042.317 076 000      591      MVI      A,CN.FIL
042.321 377 046      592      DB      SYSCALL,,CLOSE  CLOSE PROGRAM FILE
042.323 311      593      RET      RESTART      EXIT TO EXEC LOOP

```

595 *** ERROR - FATAL SYSTEM ERROR OCCURED.

596 *

597 * EXIT TO RESTART

598 *

599 * ENTRY (A) = ERROR CODE

600

601

```

042.324 315 332 042 602 ERROR CALL ERROR.
042.327 303 200 042 603 JMP RESTART

```

604

```

042.332 365      605 ERROR. PUSH PSW      SAVE CODE
042.333 315 301 052 606 CALL $CC0      CLEAR CTL-0

```

```

042.336 315 136 031 607 CALL $TYPTX
042.341 012 007 105 608 DB NL,BELL,'ERROR -', ' '+2000

```

```

042.353 361      609 POP PSW
042.354 046 012      610 MVI H,NL

```

```

042.356 377 057      611 DB SYSCALL,,ERROR
042.360 311      612 RET

```

614 *** CCHIT - ENTERED WHEN CTL-C HIT.

615 *

616

617

```

042.361 373      618 CCHIT EI
042.362 315 136 031 619 CALL $TYPTX
042.365 136 303      620 DB '/', 'C'+2000
042.367 303 200 042 621 JMP RESTART

```

```

624 *** PRS - PRESET PROGRAM.
625 *
626 * PRS PRESETS THE PROGRAM, BY
627 *
628 * 1) CLEARING ALL CHANNELS (INCLUDING THE OVERLAY CHANNEL)
629 * 2) PRINTING A PROGRAM TITLE
630 * 3) SETING UP THE MEMORY USAGE
631 * 4) SETING UP THE DATA TABLES.
632 * 5) SETING UP THE CTL-C ADDRESS
633 *
634 * ENTRY NONE
635 * EXIT NONE
636 * USES ALL
637 *
638 *
042,372 639 PRS EQU *
640 *
641 * VERIFY THE VERSION OF HDOS /79.12.GC/
642 *
042,372 377 011 643 DB SYSCALL,VERS /79.12.GC/
042,374 332 130 043 644 JC PRSERR1 PROBABLY NO .VERS SYSCALL /79.12.GC/
042,377 376 026 645 CFI VERS /79.12.GC/
043,001 302 130 043 646 JNZ PRSERR1 NOT CORRECT VERSION /79.12.GC/
647 *
043,004 377 056 648 DB SYSCALL,CLEARA CLEAR ALL CHANNELS BUT OVERLAY
043,006 076 377 649 MVI A,377H
043,010 377 055 650 DB SYSCALL,CLEAR CLEAR OVERLAY CHANNEL
651 *
652 * SETUP BUFFERS, POINTERS, AND VALUES
653 *
043,012 257 654 XRA A
043,013 062 236 053 655 STA SIVBU SECTOR IN VIEW BUFFER NOT VALID
043,016 062 261 053 656 STA PHIST NO PHT
043,021 041 000 060 657 LXI H,PATLIST
043,024 042 246 053 658 SHLD PLPTR CLEAR PATCH LIST
659 *
660 * ANNOUNCE SELF
661 *
043,027 315 136 031 662 CALL $TYPTX
043,032 012 120 101 663 DB NL,'PATCH Issue #50.05.00.',NL,ENL /79.12.GC/
664 *
665 * COMPUTE FINAL PATCH ADDRESS, REQUEST FROM HDOS
666 *
043,067 052 320 040 667 LHLD S,SYSM
043,072 353 668 XCHG
043,073 052 324 040 669 LHLD S,OMAX
043,076 173 670 MOV A,E
043,077 225 671 SUB L
043,100 157 672 MOV L,A
043,101 172 673 MOV A,D
043,102 234 674 SBB H
043,103 147 675 MOV H,A (HL) = MAX WITH OVERLAY STILL RESIDENT
043,104 053 676 DCX H
043,105 053 677 DCX H
043,106 053 678 DCX H
043,107 053 679 DCX H 4 BYTES OF SLOP

```

043.110	042 250 053	680	SHLD	PLMAX	SET MAX	
043.113	377 052	681	DB	SYSCALL,,SETTP	REQUEST MAX	
043.115	332 324 042	682	JC	ERROR	PROBLEMS, WILL PROBABLY LOOP GENERATING THIS MESSAGE	
		683				
		684 *		SETUP CTL-C		
		685				
043.120	041 361 042	686	LXI	H,CCHIT		
043.123	076 003	687	MVI	A,CTLC		
043.125	377 041	688	DB	SYSCALL,,CTLC	SETUP CTL-C PROCESSING	
043.127	311	689	RET		EXIT	
		690				
043.130	076 050	691	PRSEERR1	MVI	A,EC.NCV	NOT CORRECT VERS. OF HDQS /79.12.GC/
		692				
043.132	315 332 042	693	PRSEERR	CALL	ERROR.	/79.12.GC/
043.135	076 001	694	MVI	A,1		/79.12.GC/
043.137	377 000	695	DB	SYSCALL,,EXIT		/79.12.GC/
		696				
043.141	303 141 043	697	JMP	*	SHOULD NEVER HAPPEN	/79.12.GC/

```

700 *** OFF - OPEN FILE FOR PATCHES.
701 *
702 * OFF PROMPTS THE USER FOR A FILE NAME TO PATCH, AND
703 *
704 * 1) DETERMINES IF THE DEVICE IS SUITABLE (MUST BE H17)
705 * 2) DETERMINES IF THE FILE EXISTS
706 * 3) DETERMINES ITS TYPE (ABS, PIC)
707 * 4) SEES IF THERE IS A PATCH HISTORY TABLE (PHT) ON IT
708 * 5) DETERMINES THE PROGRAMS FWA AND LWA.
709 *
710 * ENTRY NONE
711 * EXIT 'C' CLEAR IF OK
712 * FILE DESCRIPTOR VALUES SETUP.
713 * 'C' SET IF CTL-D
714 * USES ALL
715
043.144 315 136 031 717 OFF CALL $TYPTX
043.147 106 151 154 718 DB 'File Name?', '+2000
043.142 041 071 054 719 LXI H,LINE
043.165 315 027 053 720 CALL $RTL. READ LINE IN UPPER CASE
043.170 330 721 RC CTL-D
722
723 * SEE IF ANY SWITCHES SPECIFIED
724
043.171 315 137 044 725 CALL DCS DECODE COMMAND SWITCHES
726
727 * GOT FILE NAME.
728
043.174 001 077 044 729 LXI B,OFFA LOOK UP DEVICE TYPE
043.177 021 240 053 730 LXI D,DEFAULT
043.202 377 053 731 DB SYSCALL,,DECODE
043.204 332 324 042 732 JC ERROR ERROR
043.207 072 077 044 733 LDA OFFA+0 (A) = DEVICE TYPE
043.212 057 734 CMA
043.213 346 005 735 ANI DT,DD+DT,CW MUST BE DIRECTORY DEVICE, CAPABLE OF WRITE
043.215 076 005 736 MVI A,EC,DNS
043.217 302 324 042 737 JNZ ERROR ERROR
043.222 021 240 053 738 LXI D,DEFAULT
043.225 041 071 054 739 LXI H,LINE
043.230 076 000 740 MVI A,CN,FIL
043.232 377 042 741 DB SYSCALL,,OPENR OPEN FILE FOR READ
043.234 332 324 042 742 JC ERROR ERROR
743
744 * DETERMINE PROGRAM TYPE
745
043.237 001 000 000 746 LXI B,0
043.242 315 235 050 747 CALL PPF POSITION PROGRAM FILE TO BEGINNING
043.245 001 000 001 748 LXI B,256
043.250 021 000 056 749 LXI D,BUFFER
043.253 076 000 750 MVI A,CN,FIL
043.255 377 004 751 DB SYSCALL,,READ READ FIRST BLOCK
043.257 332 324 042 752 JC ERROR ERROR
043.262 052 000 056 753 LHLD BUFFER
043.265 054 754 INR L
043.266 076 020 755 MVI A,EC,IFC

```


043.270	302 324 042	756	JNZ	ERROR	NOT BINARY FILE
043.273	174	757	MOV	A,H	
043.274	062 252 053	758	STA	FILTYF	SET FILE TYPE
043.277	376 000	759	CPI	FT,ABS	SEE IF ABS
043.301	312 332 043	760	JE	OFF1	IS ABS
043.304	376 001	761	CPI	FT,PIC	
043.306	076 020	762	MVI	A,EC,IFC	ASSUME ILLEGAL FILE CONTENT
043.310	302 324 042	763	JNE	ERROR	ERROR
		764			
		765	*	IS PIC FILE	
		766			
043.313	041 000 000	767	LXI	H,0	
043.316	042 230 053	768	SHLD	SKEW	ADDRESS N IS BYTE N
043.321	042 262 053	769	SHLD	PGMFWA	PROGRAM STARTS AT 0
043.324	052 002 056	770	LHLD	BUFFER+PIC.LEN	(HL) = PROGRAM LENGTH
043.327	303 355 043	771	JMP	OFF1.5	
		772			
		773	*	IS ABS FILE	
		774			
043.332	052 002 056	775	OFF1	LHLD	BUFFER+ABS.LDA
043.335	042 262 053	776	SHLD	PGMFWA	STORE PROGRAM'S FWA
043.340	315 224 030	777	CALL	%CHL	(HL) = -USERFWA
043.343	021 010 000	778	LXI	D,ABS.COD	
043.346	031	779	DAD	D	(HL) = ABS.COD-USERFWA
043.347	042 230 053	780	SHLD	SKEW	PGM ADDRESS-USERFWA+ABS.COD = FILE ADDRESS
043.352	052 004 056	781	LHLD	BUFFER+ABS.LEN	(HL) = PROGRAM LENGTH
		782			
043.355	042 266 053	783	OFF1.5	SHLD	FILSIZ
043.360	353	784	XCHG		SET FILE SIZE
043.361	052 262 053	785	LHLD	PGMFWA	(DE) = LENGTH
043.364	031	786	DAD	D	
043.365	053	787	DCX	H	(HL) = PGM LWA
043.366	042 264 053	788	SHLD	PGMLWA	
043.371	353	789	XCHG		
043.372	052 230 053	790	LHLD	SKEW	
043.375	031	791	DAD	D	(HL) = FILE INDEX OF PROGRAM LWA
043.376	044	792	INR	H	(H) = SECTOR NUMBER OF LAST PGM SECTOR+1
043.377	114	793	MOV	C,H	
044.000	006 000	794	MVI	B,0	(BC) = SECTOR NUMBER WHERE PHT MIGHT BE
044.002	315 235 050	795	CALL	PPF	POSITION BEFORE PHT
044.005	332 051 044	796	JC	OFF2	MISSING
		797			
044.010	001 000 001	798	LXI	B,256	
044.013	021 000 057	799	LXI	D,PHT	
044.016	076 000	800	MVI	A,CN.FIL	
044.020	377 004	801	DB	SYSCALL,.READ	TRY TO READ SUPPOSED PHT
044.022	315 324 047	802	CALL	AEE	ALLOW END OF FILE ERROR
044.025	332 051 044	803	JC	OFF2	NO PHT
044.030	021 175 053	804	LXI	D,PHTFORM+PHT.HDR	
044.033	041 000 057	805	LXI	H,PHT	
044.036	016 011	806	MVI	C,PHTHDRL	
044.040	315 060 030	807	CALL	%COMP	SEE IF IN PROPER FORMAT FOR PHT
044.043	076 001	808	MVI	A,1	
044.045	062 261 053	809	STA	PHIST	ASSUME HAVE PHT
044.050	310	810	RE		ALL OK, HAVE PHT
		811			

OFF - OPEN FILE FOR PATCHES

16:24:34 16-MAY-80

```

812 *      DONT HAVE A PHT, AM IN USER MODE
813
044.051 257      814 OFF2   XRA      A
044.052 062 261 053 815      STA      PHIST
044.055 076 000      816      MVI      A,CN.FIL
044.057 377 046      817      DB       SYSCALL,.CLOSE  CLOSE FILE WHICH IS OPEN FOR READ
044.061 076 000      818      MVI      A,CN.FIL
044.063 021 240 053 819      LXI      D,DEFAULT
044.066 041 071 054 820      LXI      H,LINE
044.071 377 044      821      DB       SYSCALL,.OPENU  OPEN USER FILE FOR UPDATE!
044.073 332 324 042 822      JC       ERROR
044.076 311      823      RET
824
825
044.077      826 OFFPA   DS       32      .DECODE BUFFER

```

```

828 **      DCS - DECODE COMMAND SWITCHES.
829 *
830 *      DCS DECODES AND REMOVES SWITCHES FROM THE COMMAND LINE
831 *      IN *LINE*
832 *
833 *      ENTRY (HL) = *LINE
834 *      EXIT  'C' SET IF ERROR
835 *           MESSAGE ALREADY GIVEN
836 *           'C' CLEAR IF OK
837 *      USES  ALL
838
044.137 257      839 DCS     XRA      A
044.140 062 232 053 840      STA      SDISP      PRESET SWITCHES
000.001      841      IF      .SYS.
842      STA      CHECKC     CLEAR CHECK FLAG
843      STA      PHTSWI     CLEAR PHT ADD SWITCH
844      ENDF
044.143 021 160 044 846      LXI      D,DCSA
044.146 315 064 052 847      CALL     $DRS      DECODE AND REMOVE SWITCHES
044.151 332 324 042 848      JC       ERROR     SWITCH ERROR
044.154 041 071 054 849      LXI      H,LINE
044.157 311      850      RET
851
852
044.160      853 DCSA    DS       0      FWA SWITCH TABLE
044.160 104 111 123 854      DB       'DISP','L'+200Q,'A'+200Q,'C'+200Q,'E'+200Q,200Q
044.171 174 044      855      DW       SW.DISP      DISPLACE SWITCH
856
000.001      857      IF      .SYS.
858      DB       'PHT',200Q
859      DW       SW.PHT
860
861      DB       'CHECK',200Q
862      DW       SW.CHK
863      ENDF
864

```

044.173 000 865 DB 0 END OF TABLE

867 ** SW.DISP - PROCESS /DISP:NN SWITCH
 868
 044.174 315 235 051 869 SW.DISP CALL \$DNS. DECODE NUMERIC SWITCH
 044.177 076 032 870 MVI A,EC.IS
 044.201 332 324 042 871 JC ERROR ILLEGAL SWITCH
 044.204 173 872 MOV A,E
 044.205 062 232 053 873 STA SDISP SET SECTOR DISPLACEMENT
 044.210 311 874 RET

876 ** SW.PHT - /PHT SWITCH
 877 *
 878 * ADD.PHT TO FILE
 879
 000,001 880 IF ,SYS,
 881 SW.PHT MVI A,1
 882 STA PHTSWI
 883 RET
 884 SW.CHK SPACE 3,10
 885 ** SW.CHK - /CHECK SWITCH
 886 *
 887 * SHOW CHECKSUM VALUES
 888
 889 SW.CHK MVI A,1
 890 STA CHECKC
 891 RET
 892 ENDDIF

```

895 *** GPI - GET PATCH ID.
896 *
897 * GPI PROMPTS THE USER WITH
898 *
899 * PATCH ID?
900 *
901 * AND READS IN A PATCH ID SEQUENCE.
902 *
903 * THE ID IS A 6 CHARACTER SEQUENCE, REPRESENTING THE ID NUMBER
904 * (1 TO 256) AND A 2 BYTE CRC-16 FOR IT.
905 *
906 * SEE 'RES' FOR A DESCRIPTION OF THE ENCODING TECHNIQUE.
907 *
908 * ENTRY NONE
909 * EXIT 'C' CLEAR IF OK
910 * PATCHID = ID BYTE
911 * 'C' SET IF CTL-D
912 * USES ALL
913 *
914 *
044.211 315 136 031 915 GPI CALL $TYPTX
044.214 120 141 164 916 DB 'Patch ID?','+200Q
044.226 006 001 917 MVI B,1 1 BYTE REPLY
044.230 315 253 050 918 CALL RES READ ENCODED STRING
044.233 330 919 RC CTL-D
044.234 312 305 044 920 JE GPI2 IS OK
921 *
922 * BAD PATCH ID
923 *
044.237 315 136 031 924 CALL $TYPTX
044.242 007 111 156 925 DB BELL,'Invalid Patch ID. Try Again...','ENL
044.302 303 211 044 926 JMP GPI
927 *
928 * GOT GOOD ID
929 *
044.305 072 071 054 930 GPI2 LDA LINE (A) = SEQUENCE BYTE
044.310 062 260 053 931 STA PATCHID SET PATCH ID CODE
044.313 247 932 ANA A CLEAR CARRY
044.314 311 933 RET

```

```

936 *** GPC - GET PREREQ CODE.
937 *
938 * GPC PROMPTS THE USER FOR THE ENTRY OF THE PREREQUISITE CODE.
939 * THIS CODE CONSISTS OF 14 CHARACTERS, ENCODEING 5 BYTES OF
940 * CODE, AND 2 BYTES OF CRC-16 FOR THAT CODE.
000.000 941 ERRNZ PATPRQL-5 DOCUMENTATION ASSUMES 5 BYTES OF PREREQ CODE
942 *
943 * ENTRY NONE
944 * EXIT 'C' CLEAR IF OK
945 * 'C' SET IF CTL-D
946 * USES ALL
947
948
044.315 257 949 GPC XRA A
044.316 062 214 045 950 STA GPCC FLAG MISSING NO PREREQUISITES
044.321 315 136 031 951 CALL $TYPTX
044.324 120 162 145 952 DB 'Prerequisite Code?',+200Q
044.347 006 005 953 MVI B,PATPRQL SET NUMBER OF BYTES TO GET
044.351 315 253 050 954 CALL RES READ ENCODED STRING
044.354 330 955 RC CTL-D
044.355 312 022 045 956 JZ GPC1 CODE OK
957
958 * ERROR IN CODE ENTRY
959
960
044.360 315 136 031 961 CALL $TYPTX
044.363 007 111 154 962 DB BELL,'Illegal Code. Try Again...','ENL
045.017 303 315 044 963 JMP GPC TRY AGAIN
964
965 * GOT GOOD CODE. COPY INTO TABLE, SEE IF ANY MISSING
966
045.022 021 071 054 967 GPC1 LXI D,LINE
045.025 001 005 000 968 LXI B,PATPRQL
045.030 041 253 053 969 LXI H,PATPRQ
045.033 315 252 030 970 CALL $MOVE MOVE 5 BYTES INTO PATPRQ TABLE
971
972 * SEE IF ANY PREREQS MISSING
973
045.036 041 253 053 974 LXI H,PATPRQ
045.041 021 023 057 975 LXI D,PHT+PHT.HIS
045.044 006 005 976 MVI B,PATPRQL
045.046 032 977 GPC3 LDAX D (A) = HISTORY BYTE
045.047 057 978 CMA
045.050 246 979 ANA M NON-ZERO IF ONE MISSING
045.051 312 136 045 980 JZ GPC5 NOT MISSING IN THIS BYTE
981
982 * MISSING A PATCH.
983 * (A) = BIT INDEX OF MISSING PATCH
984 * (B) = 5-BYTE INDEX OF MISSING PATCH BIT
985
045.054 365 986 PUSH PSW SAVE BIT
045.055 076 005 987 MVI A,5
045.057 220 988 SUB B (A) = BYTE INDEX
045.060 207 989 ADD A
045.061 207 990 ADD A
045.062 207 991 ADD A (A) = 8*BYTE INDEX

```

```

045.063 117 992 MOV C,A (C) = BIT INDEX OF LOW/ORDER BIT
045.064 361 993 POP PSW (A) = BIT INDEX
045.065 014 994 GPC4 INR C
045.066 247 995 ANA A
045.067 312 136 045 996 JZ GPC5 NO MORE MISSING PATCHES
045.072 037 997 RAR
045.073 322 065 045 998 JNC GPC4 THIS ONE NOT MISSING
999
1000 * (C) = # OF MISSING PATCH. REPORT IT
1001
045.076 365 1002 PUSH PSW
045.077 305 1003 PUSH B
045.100 325 1004 PUSH D
045.101 345 1005 PUSH H SAVE ALL REGS
045.102 006 000 1006 MVI B,0 (BC) = PATCH NUMBER +1
045.104 013 1007 DCX B CORRECT
045.105 041 176 045 1008 LXI H,GPCB
045.110 076 003 1009 MVI A,3
045.112 315 074 053 1010 CALL $UDDN UNPACK PATCH NUMBER
045.115 041 154 045 1011 LXI H,GPCA
045.120 377 003 1012 DB SYSCALL,.PRINT FINK ON HIM
045.122 076 001 1013 MVI A,1
045.124 062 214 045 1014 STA GPCC FLAG ONE MISSING
045.127 341 1015 POP H
045.130 321 1016 POP D
045.131 301 1017 POP B
045.132 361 1018 POP PSW RESTORE REGS
045.133 303 065 045 1019 JMP GPC4 TRY SOME MORE
1020
1021 * NONE MISSING IN THIS BYTE. TRY NEXT
1022
045.136 023 1023 GPC5 INX D
045.137 043 1024 INX H
045.140 005 1025 DCR B
045.141 302 046 045 1026 JNZ GPC3 MORE TO TRY
1027
1028 * DONE WITH CHECKS. SEE IF ANY MISSING
1029
045.144 072 214 045 1030 LDA GPCC
045.147 247 1031 ANA A
045.150 302 200 042 1032 JNZ RESTART IF ANY MISSING
045.153 311 1033 RET ALL OK
1034
045.154 007 122 145 1035 GPCA DB BELL,'Required Patch #
045.176 060 060 060 1036 GPCB DB '000 missing.',BELL,ENL
1037
045.214 000 1038 GPCC DB 00 <>0 IF PATCH MISSING

```

```

1041 *** GPA - GET PATCH ADDRESS.
1042 *
1043 * GPA GETS THE ADDRESS FOR THE NEXT ROUND OF PATCHES.
1044 *
1045 * THE PATCH ADDRESS IS ENTERED IN THE PATCH LIST.
1046 *
1047 * ENTRY NONE
1048 * EXIT 'C' CLEAR IF OK
1049 * PATADR=ADDRESS
1050 * 'C' SET IF CTL-D
1051 * USES ALL
1052
1053
045,215 315 136 031 1054 GPA CALL $TYPTX
045,220 012 101 144 1055 DB NL,'Address?',' +2000
045,232 041 077 054 1056 LXI H,LINE+6
045,235 315 027 053 1057 CALL $RTL. READ LINE IN UPPER CASE
045,240 330 1058 RC CTL-D
045,241 041 071 054 1059 LXI H,LINE
045,244 076 006 1060 MVI A,6
045,246 066 060 1061 GPA1 MVI M,'0' PUT IN 6 PRECEDING ZEROS
045,250 043 1062 INX H
045,251 075 1063 DCR A
045,252 302 246 045 1064 JNZ GPA1
045,255 053 1065 DCX H
1066
1067 * FIND LAST CHARACTER IN ENTERED LINE
1068
045,256 043 1069 GPA2 INX H
045,257 176 1070 MOV A,M
045,260 247 1071 ANA A
045,261 302 256 045 1072 JNZ GPA2 FIND LAST CHARACTER
045,264 021 372 377 1073 LXI D,-6
045,267 031 1074 DAD D (HL) = ADDRESS OF LAST 6 DIGITS
045,270 315 373 047 1075 CALL DOB DECODE OCTAL BYTE
045,273 332 363 045 1076 JC GPA3 ERROR
045,276 365 1077 PUSH PSW SAVE VALUE
045,277 315 373 047 1078 CALL DOB
045,302 332 363 045 1079 JC GPA3 ERROR
045,305 321 1080 POP D (D) = HIGH ORDER BYTE
045,306 137 1081 MOV E,A (DE) = ADDRESS
1082
1083 * SEE IF ADDRESS IS BEFORE FILE
1084
045,307 052 230 053 1085 LHLD SKEW
045,312 315 224 030 1086 CALL $CHL (HL) = SUBTRACT FACTOR
045,315 173 1087 MOV A,E
045,316 225 1088 SUB L
045,317 172 1089 MOV A,D
045,320 234 1090 SBB H 'C' SET IF ADDRESS TOO LOW
045,321 353 1091 XCHG (HL) = ADDRESS
045,322 042 233 053 1092 SHLD CPA SET CURRENT PATCH ADDRESS
045,325 320 1093 RNC IS OK, EXIT
1094
1095 * PATCH ADDRESS TOO LOW
1096

```

```
045.326 315 136 031 1097 CALL $TYPTX
045.331 007 120 141 1098 DB BELL,'Patch Address Too Low',ENL
045.360 303 215 045 1099 JMP GPA TRY AGAIN
1100
1101 * ERROR IN INPUT
1102
045.363 315 136 031 1103 GPA3 CALL $TYPTX
045.366 007 111 154 1104 DB BELL,'Illegal Address Value.',ENL
046.016 303 215 045 1105 JMP GPA TRY AGAIN
```



```

1108 *** APP - ACCEPT PROGRAM PATCHES.
1109 *
1110 * APP READS THE PROGRAM PATCHES FROM THE CONSOLE.
1111 * STARTING AT THE CURRENT PATCH ADDRESS, AND INCREMENTING BY
1112 * ONE BYTE EACH TIME, APP PROMPTS
1113 *
1114 * AAAAAA = VVV/
1115 *
1116 * WHERE 'AAAAAA' = ADDRESS, 'VVV' = OLD VALUE. THE USER MAY
1117 * THEN TYPE
1118 *
1119 * NNN NEW VALUE
1120 * CR LEAVE THE SAME
1121 * CTL-D START NEW ADDRESS
1122 *
1123 * THE VALUES RECEIVED ARE ENTERED IN THE PATCH LIST.
1124 *
1125 * ENTRY NONE
1126 * EXIT WHEN CTL-D HIT
1127 * USES ALL
1128 *
1129 *
046.021 1130 APP EQU *
046.021 052 233 053 1131 LHL D CPA (HL) = DESIRED BYTE ADDRESS
046.024 315 111 050 1132 CALL LBF LOCATE BYTE IN TILE
046.027 353 1133 XCHG (DE) = ADDRESS IN VIEWFR
1134 *
1135 * GOT CURRENT VALUE, DISPLAY IT
1136 *
046.030 052 233 053 1137 APP4 LHL D CPA
046.033 174 1138 MOV A,H
046.034 315 147 053 1139 CALL $TOD TYPE OCTAL DIGITS
046.037 175 1140 MOV A,L
046.040 315 147 053 1141 CALL $TOD
046.043 315 136 031 1142 CALL $TYPTX
046.046 040 075 240 1143 DB ' ',' ' +200Q
046.051 032 1144 LDAX D (A) = OLD VALUE
046.052 315 147 053 1145 CALL $TOD
046.055 315 136 031 1146 CALL $TYPTX
046.060 257 1147 DB ' '//+200Q
1148 *
1149 * ACCEPT NEW VALUE
1150 *
046.061 325 1151 PUSH D SAVE ADDRESS OF CURRENT VALUE
046.062 041 074 054 1152 LXI H,LINE+3
046.065 315 027 053 1153 CALL $RTL. READ IN UPPER CASE
046.070 321 1154 POP D
046.071 330 1155 RC CTL-D
046.072 072 074 054 1156 LDA LINE+3
046.075 247 1157 ANA A
046.076 032 1158 LDAX D (A) = CURRENT VALUE
046.077 312 132 046 1159 JZ APP6 LEFT TO CURRENT VALUE
1160 *
1161 * GOT THE NEW VALUE
1162 *
046.102 076 060 1163 MVI A,'0'

```

```

046.104 041 071 054 1164 LXI H,LINE PUT 2 LEADING '0'S BEFORE
046.107 167 1165 MOV M,A
046.110 043 1166 INX H
046.111 167 1167 MOV M,A
046.112 043 1168 APP5 INX H
046.113 176 1169 MOV A,M
046.114 247 1170 ANA A LOOK FOR END OF VALUE
046.115 302 112 046 1171 JNZ APP5 NOT AT END OF LINE
046.120 021 375 377 1172 LXI D,-3
046.123 031 1173 DAD D (HL) = FWA OF 3 DIGIT VALUE
046.124 315 373 047 1174 CALL DOB DECODE OCTAL BYTES
046.127 332 165 046 1175 JC APP7 ERROR
1176
1177 * ADD VALUE TO PATCH LIST
1178 *
1179 * (A) = NEW VALUE
1180
046.132 365 1181 APP6 PUSH PSW SAVE VALULE
046.133 072 233 053 1182 LDA CPA
046.136 315 170 047 1183 CALL ABL INSERT ADDRESS IN LIST
046.141 072 234 053 1184 LDA CPA+1
046.144 315 170 047 1185 CALL ABL
046.147 361 1186 POP PSW
046.150 315 170 047 1187 CALL ABL ADD BYTE TO LIST
046.153 052 233 053 1188 LHLD CPA
046.156 043 1189 INX H
046.157 042 233 053 1190 SHLD CPA INCREMENT ADDRESS
046.162 303 021 046 1191 JMP APP GET ANOTHER
1192
1193 * BAD CALL
1194
046.165 315 136 031 1195 APP7 CALL $TYPTX
046.170 007 111 154 1196 DB BELL,'Illegal Value',ENL
046.207 303 021 046 1197 JMP APP TRY AGAIN

```

```

1201 *** CPC - CHECK PATCH CRC.
1202 *
1203 * CPC COMPUTES A CRC-16 FOR THE ENTIRE PATCH SERIES, INCLUDING
1204 * THE SERIAL NUMBER, PREREQUESTIE LIST, AND ALL ADDRESSES AND
1205 * VALUES, AND THEN CHECKS THAT CRC AGAINST THE ONE ENTERED
1206 * BY THE USER.
1207 *
1208 * ENTRY NONE
1209 * EXIT 'C' CLEAR IF OK
1210 * 'C' SET IF ERROR
1211 * USES ALL
1212
1213
046.212 315 136 031 1214 CPC CALL $TYPTX
046.215 012 120 141 1215 DB NL,'Patch Check Code?',' '+200Q
046.240 021 000 060 1216 LXI D,FATLIST
046.243 052 246 053 1217 LHLD PLPTR
046.246 175 1218 MOV A,L
046.247 223 1219 SUB E
046.250 117 1220 MOV C,A (BC) = LENGTH OF VALUES IN FATLIST
046.251 174 1221 MOV A,H
046.252 232 1222 SBB D
046.253 107 1223 MOV B,A
046.254 315 335 047 1224 CALL CBS CRC BYTE STRING
046.257 042 077 047 1225 SHLD CPCA STORE EXPECTED CRC
000.001 1226 IF .SYS.
1227
1228 * TYPE THE EXPECTED CRC, AS A SUBTLE HINT!
1229
1230 LDA CHECKC
1231 ANA A
1232 JZ CPC1 DONT SHOW IT, HE'S PLAYING DUMB
1233 CALL $TYPTX
1234 DB 'Expecting:',' '+200Q
1235 LDA CPCA
1236 CALL $TOD
1237 LDA CPCA+1
1238 CALL $TOD
1239 CALL $TYPTX
1240 DB ENL
1241 CPC1 EQU *
1242 ENDIF
046.262 006 002 1243 MVI B,2
046.264 315 253 050 1244 CALL RES READ ENCRYPTED STRING
046.267 330 1245 RC CTL-D
046.270 312 355 046 1246 JE CPC2 IS OK
046.273 315 136 031 1247 CALL $TYPTX
046.276 007 103 150 1248 DB BELL,'Check Code Entered Incorrectly. Try Again.','ENL
046.352 303 212 046 1249 JMP CPC TRY AGAIN
1250
1251 * GOT GOOD CHECK CODE. SEE IF MATCHES
1252
046.355 052 071 054 1253 CPC2 LHLD LINE
046.360 353 1254 XCHG
046.361 052 077 047 1255 LHLD CPCA
046.364 315 216 030 1256 CALL $CDEHL COMPARE

```

CPC - CHECK PATCH CRC.

CPC

16:24:40 16-MAY-80

046.367	310	1257	RE	EVERYTHING OK
046.370	315 136 031	1258	CALL	\$TYPTX
046.373	007 103 150	1259	DB	BELL; 'Check Code Does Not Match Patch'
047.033	012 105 162	1260	DB	NL; 'Error In Patch Entry. Try Again.', ENL
047.075	067	1261	STC	
047.076	311	1262	RET	
		1263		
047.077	000 000	1264	CPCA	DW 0
				TEMP STORE FOR COMPUTED CRC

```

1268 ***      EPF - ENTER PATCHES IN FILE.
1269 *
1270 *      EPF IS CALLED TO ENTER THE PATCHES STORED IN PATLIST INTO
1271 *      THE FILE.
1272 *
1273 *      IF WE ARE IN USER MODE, THE FILE MUST ALREADY BE OPEN FOR UPDATE.
1274 *      IF WE ARE IN SYSTEM MODE, EPF WILL KLUDGE THE FILE OPEN.
1275 *
1276 *      ENTRY  NONE
1277 *      EXIT   NONE
1278 *      USES   ALL
1279
1280
1281 EPF      EQU      *
1282      LDA      PHIST
1283      ANA      A
1284      CNZ      CFU          CLUDGE FILE TO UPDATE STATUS
1285
1286 *      PUT IN PATCHES, A BYTE AT A TIME
1287
1288 EPF1     LXI      D,PATLIST
1289      LHLD     PGMLWA
1290      MOV      B,H
1291      MOV      C,L          (BC) = CURRENT LWA
1292 EPF2     LHLD     PLPTR
1293      CALL     $CDEHL        SEE IF ALL IN
1294      RE              ALL IN
1295
1296 *      GOT ANOTHER PATCH. INSTALL IT
1297
1298      LDAX     D
1299      MOV      L,A
1300      INX      D
1301      LDAX     D
1302      MOV      H,A          (HL) = ADDRESS
1303      INX      D
1304
1305 *      SEE IF EXTENDING PROGRAM LWA
1306
1307      MOV      A,C
1308      SUB      L
1309      MOV      A,B
1310      SBB      H
1311      JNC      EPF3          NOT EXTENDING
1312      MOV      B,H
1313      MOV      C,L          UPDATE (BC)
1314      SHLD     PGMLWA        SET NEW LENGTH
1315 EPF3     CALL     LBF        LOCATE BYTE IN FILE
1316      LDAX     D
1317      INX      D
1318      MOV      H,A          PATCH SECTOR
1319      MVI      A,1
1320      STA      SIVBA        FLAG SECTOR AALTERED
1321      INX      H
1322      JMP      EPF2          GET NEXT PATCH

```

APF - ADD PHT TO FILE

APF

16:24:42 16-MAY-80

000.001

```
1326 **      APF - ADD PHT TO FILE.
1327 *
1328 *      APF IS CALLED TO ADD A PHT TO A FILE.
1329 *
1330 *      THE PROPER PHT SECTOR ADDRESS IS COMPUTED, THE FILE IS KLUDGED OPEN
1331 *      FOR UPDATE, AND A BLANK PHT IS ADDED.
1332 *
1333 *      ENTRY      NONE
1334 *      EXIT      TO RESTART
1335 *      USES      ALL
1336
1337
1338          IF      .SYS.
1339 APF      CALL    CFU          KLUDGE FILE FOR UPDATE
1340          LXI     B,PHTL
1341          LXI     D,PHTFORM
1342          LXI     H,PHT
1343          CALL    $MOVE      MOVE IN ARCHTYPE PHT FORM
1344          LXI     B,9
1345          LXI     D,S.DATE
1346          LXI     H,PHT+PHT.DAT
1347          CALL    $MOVE      MOVE IN DATE
1348          CALL    WPH.        ADD PHT
1349          MVI     A,CN.FIL
1350          DB      SYSCALL,.,CLOSE  CLOSE IT
1351          JMP     RESTART     TRY AGAIN
1352          ENDIF
```

```

1356 **      ABL - ADD BYTE TO LIST.
1357 *
1358 *      ABL ADDS A SINGLE BYTE TO THE PATCH LIST (PATLIST).
1359 *
1360 *      ENTRY      (A) = BYTE
1361 *      EXIT      NONE
1362 *      USES      A,F,D,E,H,L
1363
1364
047.170 052 246 053 1365 ABL      LHLD      PLPTR
047.173 167      1366      MOV      M,A          STORE IN LIST
047.174 043      1367      INX      H
047.175 042 246 053 1368      SHLD     PLPTR          UPDATE COUNT
047.200 353      1369      XCHG
047.201 052 250 053 1370      LHLD     PLMAX          SEE IF OVERFLOW SOON OR NOW
047.204 045      1371      DCR      H          ALLOW 256 BYTE WARNING
047.205 173      1372      MOV      A,E
047.206 225      1373      SUB      L
047.207 172      1374      MOV      A,D
047.210 234      1375      SBB      H
047.211 330      1376      RC
047.212 247      1377      ANA      A          PLENTY OF ROOM
047.213 302 273 047 1378      JNZ      ABL1          CLEAN OUT OF ROOM!
1379
1380 *      RUNNING OUT OF ROOM, WARN HIM BEFORE ITS TOO LATE!
1381
047.216 315 136 031 1382      CALL     $TYPTX
047.221 007 122 165 1383      DB      BELL,'Running low on space. Please finish up!','ENL
047.272 311      1384      RET
1385
1386 *      OUT OF ROOM
1387
047.273 315 136 031 1388 ABL1      CALL     $TYPTX
047.276 007 117 165 1389      DB      BELL,'Out of RAM Space!','ENL
047.321 303 200 042 1390      JMP      RESTART
1391
1392 **      AEE - ALLOW EOF ERROR.
1393 *
1394 *      AEE IS CALLED IMMEDIATELY AFTER A SYSTEM CALL, TO CHECK
1395 *      FOR THE TYPE OF A RETURNED ERROR. IF THERE IS NO ERROR, OR THE
1396 *      ERROR TYPE IS 'EC.EOF', THEN AEE RETURNS TO THE CALLER.
1397 *      IF THE ERROR WAS NOT EOF, THEN AEE JUMPS TO *ERROR*
1398 *
1399 *      ENTRY      'C' SET IF ERROR
1400 *      (A) = ERROR CODE
1401 *      EXIT      TO *ERROR* IF ERROR <> EC.EOF
1402 *      TO CALLER IF ERROR = EC.EOF
1403 *      TO CALLER IF NO ERROR
1404 *      USES      NONE
1405
1406
047.324 320      1407 AEE      RNC          NO ERROR
047.325 365      1408      PUSH     PSW          SAVE PSW

```

047.326	378 001	1409	CFI	EC.EOF	
047.330	302 324 042	1410	JNE	ERROR	NOT RIGHT TYPE OF ERROR
047.333	361	1411	POP	PSW	
047.334	311	1412	RET		RETURN WITH EOF ERROR

```

1414 **      CBS - CRC BYTE STRING.
1415 *
1416 *      CBS COMPUTES THE CRC OF A STRING OF MEMORY BYTES.
1417 *      THE CRC IS COMPUTED BASED ON A PREFIX OF 377377A.
1418 *
1419 *      * * WARNING * * ;THIS ROUTINE IS NOT ESPECIALLY COMPATIBLE
1420 *      WITH OTHER CRC GENERATING ROUTINES. ITS CRC SHOULD BE COMPARED
1421 *      ONLY WITH OTHER CRC'S GENERATED BY *CBS*, AND
1422 *      NO OTHER ROUTINES.
1423 *
1424 *      ENTRY      (DE) = ADDRESS OF STRING
1425 *                (BC) = COUNT
1426 *      EXIT      (HL) = NEW CRC VALUE
1427 *                (DE) = (DE) + (BC)
1428 *      USES      ALL
1429
1430

```

047.335	041 377 377	1431	CBS	LXI	H,377377A	
047.340	170	1432	CBS1	MOV	A,B	
047.341	261	1433		ORA	C	
047.342	310	1434		RZ		ALL DONE
047.343	032	1435		LDAX	D	
047.344	315 316 052	1436		CALL	%CRC	ADD TO CRC
047.347	023	1437		INX	D	
047.350	013	1438		DCX	B	
047.351	303 340 047	1439		JMP	CBS1	DO NEXT ONE

```

1441 **      CFU - KLUDGE FILE TO UPDATE STATUS.
1442 *
1443 *      *****
1444 *      *       *
1445 *      * NOTE *
1446 *      *       *
1447 *      *****
1448 *
1449 *      FOR PROTECTION, SYSTEM FILES (WHICH PROBABLY HAVE THE WRITE-PROTECT)
1450 *      BIT SET ARE OPENED ONLY FOR READ ACCESS (THEY CANNOT BE OPENED FOR
1451 *      UPDATE ACCESS, IF WRITE PROTECTED)
1452 *
1453 *      CFU SCREWS AROUND WITH THE CANNEL AND SETS THE APPROPRIATE
1454 *      BITS TO ALLOW READ/WRITE (I.E., UPDATE) ACCESS.
1455 *
1456 *      ENTRY      FILE OPEN ON CN.FIL
1457 *      EXIT      NONE
1458 *      USES      A,F,D,E,H,L

```



```

1459
1460
047.354 052 352 040 1461 CFU LHL D S,CFWA
000.000 1462 ERRNZ CN,FIL
047.357 315 211 030 1463 CALL $HLIHL (HL) = CHANNEL FWA
047.362 021 004 000 1464 LXI D,IOC.FLG
047.365 031 1465 DAD D (HL) = ADDRESS OF IOC.FLG BYTE
047.366 076 014 1466 MVI A,FT.OW+FT.OU
047.370 266 1467 ORA M
047.371 167 1468 MOV M,A
047.372 311 1469 RET CHANGE FLAGS TO 'OPEN FOR UPDATE'
EXIT

```

```

1471 ** DOB - DECODE OCTAL BYTE.
1472 *
1473 * DOB DECODES A THREE DIGIT OCTAL BYTE INTO BINARY.
1474 *
1475 * ENTRY (HL) = ADDRESS OF THREE DIGITS
1476 * EXIT 'C' CLEAR IF OK
1477 * (HL) = (HL)+3
1478 * (A) = VALUE
1479 * 'C' SET IF ERROR
1480 * USES A,F,E,H,L
1481
1482
047.373 021 003 000 1483 DOB LXI D,3 (E) = DIGIT COUNT, (D) = ACCUM
047.376 172 1484 DOB1 MOV A,D
047.377 207 1485 ADD A
050.000 207 1486 ADD A
050.001 207 1487 ADD A (A) = ACCUM*8
050.002 127 1488 MOV D,A
050.003 176 1489 MOV A,M (A) = DIGIT
050.004 043 1490 INX H
050.005 326 060 1491 SUI '0'
050.007 330 1492 RC ERROR
050.010 376 010 1493 CPI 8
050.012 077 1494 CMC
050.013 330 1495 RC ERROR
050.014 202 1496 ADD D ADD TO ACCUM
050.015 127 1497 MOV D,A
050.016 035 1498 DCR E
050.017 302 376 047 1499 JNZ DOB1 GET ANOTHER DIGIT
050.022 311 1500 RET

```

```

1502 ** FVB - FLUSH VIEW BUFFER.
1503 *
1504 * FVB FLUSHES THE CONTENTS OF THE VIEW BUFFER OUT TO THE DISK FILE,
1505 * IF THEY HAVE BEEN CHANGED (SIVBA <> 0).
1506 *
1507 * THE FILE MUST BE ALREADY OPEN FOR UPDATE.
1508 *

```

```

1509 *      ENTRY  NONE
1510 *      EXIT   NONE
1511 *      USES   ALL
1512
1513
050.023 072 237 053 1514 FVB  LDA    SIVBA
050.026 247        1515      ANA    A
050.027 310        1516      RZ          NOT ALTERED
050.030 072 235 053 1517      LDA    SIVB
050.033 117        1518      MOV    C,A
050.034 006 000    1519      MVI    B,0      (BC) = SECTOR NUMBER
050.036 315 235 050 1520      CALL  PPF      POSITION OVER IT
050.041 315 324 047 1521      CALL  AEE      ALLOW EOF ERROR, ONLY
050.044 322 067 050 1522      JNC    FVB1     GOT.THERE
1523
1524 *      COULDN'T POSITION THERE, MUST BE EXTENDING THE FILE, WE'LL
1525 *      WRITE GARBAGE ENOUGH TO DO THE EXTEND...
1526
050.047 101        1527      MOV    B,C
050.050 016 000    1528      MVI    C,0      (BC) = BYTES NEEDED TO EXTEND
050.052 021 000 004 1529      LXI    D,4000A    (DE) = ADDRESS OF GARBAGE (PROBABLY 0'S)
050.055 076 000    1530      MVI    A,CN.FIL
050.057 377 005    1531      DB     SYSCALL,.WRITE WRITE IT
050.061 332 324 042 1532      JC     ERROR    ERROR ON WRITE
050.064 303 023 050 1533      JMP     FVB     NOW TRY IT
1534
1535 *      GOT THERE, WILL WRITE REPLACEMENT SECTOR
1536
050.067 001 000 001 1537 FVB1  LXI    B,256
050.072 021 000 055 1538      LXI    D,VIEWBFR
050.075 076 000    1539      MVI    A,CN.FIL
050.077 377 005    1540      DB     SYSCALL,.WRITE WRITE IT
050.101 332 324 042 1541      JC     ERROR    ERROR
050.104 257        1542      XRA    A
050.105 062 237 053 1543      STA    SIVBA    CLEAR ALTERED FLAG
050.110 311        1544      RET          EXIT

```

```

1546 **      LBF - LOCATE BYTE IN FILE.
1547 *
1548 *      LBF LOCATES A BYTE IN THE USER PROGRAM FILE, AND BRINGS THE
1549 *      SECTOR CONTAINING IT INTO VIEWBFR.
1550 *
1551 *      IF A NEW SECTOR IS NEEDED, THE VIEWBFR IS
1552 *      FLUSHED BACK TO THE DISK (IF NECESSARY) FIRST.
1553 *
1554 *      IF THE REQUIRED BYTE CANNOT BE REACHED (OFF THE END OF THE FILE)
1555 *      THEN A SECTOR FULL OF ZEROS IS 'IMAGINED' FOR THE OCCASION.
1556 *      WHEN THIS SECTOR IS FLUSHED BACK TO THE DISK (VIA FVB)
1557 *      THE DISK FILE WILL BE EXTENDED FOR IT.
1558 *
1559 *      ENTRY  (HL) = USER PROGRAM ADDRESS FOR BYTE
1560 *      EXIT   (HL) = ADDRESS IN VIEWBFR OF VALUE
1561 *      USES   A,F,H,L

```

```

1562
1563
050.111 305 1564 LBF PUSH B
050.112 325 1565 PUSH D SAVE REGISTERS
050.113 353 1566 XCHG
050.114 052 230 053 1567 LHLD SKEW (HL) = ADDRESS TO FILE INDEX FACTOR
050.117 031 1568 DAD D (H) = SECTOR, (L) = INDEX IN SECTOR
050.120 353 1569 XCHG (DE) = FILE POINTER
050.121 325 1570 PUSH D SAVE FOR LATER USE
050.122 052 235 053 1571 LHLD SIVB (L) = SECTOR # IN VIEWBFR
000.000 1572 ERRNZ SIVBV-SIVB-1 (H) < 0 IF VIEWBFR VALID
050.125 174 1573 MOV A,H
050.126 247 1574 ANA A
050.127 312 150 050 1575 JZ LBF4 ISNT VALID, SO JUST READ A NEW ONE
050.132 172 1576 MOV A,D
050.133 275 1577 CMP L SEE IF ONE WE WANT
050.134 302 145 050 1578 JNE LBF1 NOT RIGHT ONE
050.137 341 1579 POP H
050.140 046 055 1580 MVI H,VIEWBFR/256 (HL) = ADDRESS
050.142 321 1581 POP D RESTORE REGS
050.143 301 1582 POP B
050.144 311 1583 RET EXIT WITH BYTE
1584
1585 * WILL HAVE TO REPLACE SECTOR IN BFR, FLUSH IT
1586
050.145 315 023 050 1587 LBF1 CALL FVR FLUSH VIEW BUFFER
1588
1589 * READ IN THE SECTOR CONTAINING THE BYTE.
1590
050.150 321 1591 LBF4 POP D (D) = SECTOR NEEDED
050.151 325 1592 PUSH D
050.152 112 1593 MOV C,D
050.153 006 000 1594 MVI B,0
050.155 315 235 050 1595 CALL PPF POSITION TO IT
050.160 315 324 047 1596 CALL AEE ALLOW EOF ERROR, ONLY
050.163 332 206 050 1597 JC LBF5 IS EOF
050.166 001 000 001 1598 LXI B,256
050.171 021 000 055 1599 LXI D,VIEWBFR
050.174 076 000 1600 MVI A,CN.FIL
050.176 377 004 1601 DB SYSCALL,,READ READ IT
050.200 315 324 047 1602 CALL AEE ALLOW EOF ERROR, ONLY
050.203 322 216 050 1603 JNC LBF6 GOT THE SECTOR
1604
1605 * SECTOR IS OFF END OF FILE, MAKE ONE UP OF ALL ZEROS.
1606
050.206 006 000 1607 LBF5 MVI B,0
050.210 041 000 055 1608 LXI H,VIEWBFR
050.213 315 212 031 1609 CALL $ZERO ZERO IT
1610
1611 * BUFFER IS READ IN. SETUP DESCRIPTOR CELLS.
1612
050.216 341 1613 LBF6 POP H (H) = SECTOR, (L) = INDEX
050.217 174 1614 MOV A,H
050.220 062 235 053 1615 STA SIVB SET SECTOR IN VIEWBFR
050.223 076 001 1616 MVI A,1
050.225 062 236 053 1617 STA SIVBV SET VIEWBFR VALID

```

050.230	046	055	1618	MOV	H,VIEWBFR/256	(HL) = ADDRESS IN VIEWBFR
050.232	321		1619	POP	D	RESTORE REGISTERS
050.233	301		1620	POP	B	
050.234	311		1621	RET		

1623 ** PPF - POSITION PROGRAM FILE.

1624 *

1625 * PPF IS CALLED TO POSITION THE PROGRAM FILE IMMEDIATELY BEFORE
1626 * A GIVEN SECTOR.

1627 *

1628 * ENTRY (BC) = SECTOR NUMBER

1629 * EXIT 'C' SET IF ERROR

1630 * (A) = ERROR CODE

1631 * 'C' CLEAR IF OK

1632 * USES ALL

1633

1634

050.235	072	232	053	1635	PPF	LDA	SDISP	(A) = SECTORS PROGRAM IS DISPLACED
---------	-----	-----	-----	------	-----	-----	-------	------------------------------------

050.240	201			1636		ADD	C	
---------	-----	--	--	------	--	-----	---	--

050.241	117			1637		MOV	C,A	
---------	-----	--	--	------	--	-----	-----	--

050.242	076	000		1638		MVI	A,0	
---------	-----	-----	--	------	--	-----	-----	--

050.244	210			1639		ADC	B	
---------	-----	--	--	------	--	-----	---	--

050.245	107			1640		MOV	B,A	
---------	-----	--	--	------	--	-----	-----	--

050.246	076	000		1641		MVI	A,CN.FIL	
---------	-----	-----	--	------	--	-----	----------	--

050.250	377	047		1642		DB	SYSCALL,.POSIT	
---------	-----	-----	--	------	--	----	----------------	--

050.252	311			1643		RET		
---------	-----	--	--	------	--	-----	--	--

1645 ** RES - READ ENCODED STRING.

1646 *

1647 * RES READS AN ENCODED CHARACTER STRING.

1648 *

1649 * MANY OF THE USER ENTRIES ARE ENCODED, TO MAKE THEM HARD TO UNDERSTAND.

1650 * THIS ENCOURAGES THE USER TO ENTER THEM CAREFULLY, AND DISCOURAGES

1651 * THE USER FROM OMITTING THINGS BECAUSE HE'S LAZY, OR DOESN'T THINK

1652 * THAT THEY'RE NEEDED.

1653 *

1654 * A N BYTE STRING IS ENCODED AS N*2+4 CHARACTERS.

1655 *

1656 * THE STRING IS DECODED BY DECREMENTING THE ASCII CHARACTER BY 1,

1657 * AND XOR'ING IT AGAINST A CHARACTER PATTERN

1658 * (RESA). THIS STRING IS REPEATED EVERY 32 CHARACTERS UNTIL

1659 * THE ENTIRE ENTRY IS XOR'ED. THEN, THE LOW 4 BITS OF EACH

1660 * CHARACTER ARE TAKEN, AND COMPRESSED TO FORM N+2 BYTES. THE 'N'

1661 * BYTES ARE THE VALUE, THE 2 END BYTES ARE A CRC-16 OF THE N BYTES.

1662 * THUS, THE ENTRY CAN BE CHECKED FOR INTERNAL CONSISTENCY.

1663 *

1664 * ENTRY (B) = N (NUMBER OF BYTES)

1665 * EXIT 'C' SET IF CTL-D STRUCK

1666 * 'C' CLEAR IF ENTRY MADE

1667 * 'Z' SET IF ENTRY VALID

```

1668 *      (LINE, LINE+N-1 = VALUE BYTES)
1669 *      'Z' CLEAR IF ENTRY BAD
1670 *      USES  ALL
1671
1672
050.253      1673 RES  EQU  *
000.001      1674      IF  .SYS.
1675      LDA  CHECKC
1676      ANA  A
1677      CNZ  EES      ENCODE ENCRPYTED STRING FOR WIZARDS
1678      ENIIF
050.253 041 071 054 1679      LXI  H,LINE
050.256 315 027 053 1680      CALL $RTL.      READ LINE, MAP TO UPPER CASE
050.261 330      1681      RC      CTL-D
050.262 170      1682      MOV  A,B      (A) = N
050.263 306 002 1683      ADI  2      +2 FOR CRC
050.265 107      1684      MOV  B,A
050.266 305      1685      PUSH B      SAVE N+2
050.267 001 071 054 1686      LXI  B,LINE      (BC) = TARGET FOR DECODED AND COMPRESSED STRING
050.272 353      1687      XCHG      (DE) = ENTERED STRING
050.273 041 364 050 1688      LXI  H,RESA      (HL) = XOR STRING
1689
1690 *      XOR STRING
1691
050.276 365      1692 RES1  PUSH  PSW      SAVE REMAINING COUNT
050.277 032      1693      LDAX  D
050.300 075      1694      DCR  A      DECREMENT
050.301 256      1695      XRA  M
050.302 346 017 1696      ANI  170
050.304 207      1697      ADD  A      MOVE FIRST NIBBLE LEFT
050.305 207      1698      ADD  A
050.306 207      1699      ADD  A
050.307 207      1700      ADD  A
050.310 365      1701      PUSH  PSW
050.311 023      1702      INX  D
050.312 043      1703      INX  H
050.313 032      1704      LDAX  D
050.314 075      1705      DCR  A      DECREMENT
050.315 256      1706      XRA  M
050.316 346 017 1707      ANI  170
050.320 343      1708      XTHL      (H) = FIRST NIBBLE
050.321 204      1709      ADD  H      (A) = FULL BYTE VALUE
050.322 341      1710      POP  H      (HL) = ENTERED LINE POINTER
050.323 002      1711      STAX  B      STORE IN LINE
050.324 003      1712      INX  B
050.325 023      1713      INX  D
050.326 043      1714      INX  H
050.327 175      1715      MOV  A,L
050.330 376 024 1716      CFI  $RESAE
050.332 302 340 050 1717      JNE  RES2      STRING NOT USED UP, YET
050.335 041 364 050 1718      LXI  H,RESA      START OVER
050.340 361      1719 RES2  POP  PSW      (A) = COUNT LEFT
050.341 075      1720      DCR  A
050.342 302 276 050 1721      JNZ  RES1
1722
1723 *      CRC VALUE TO SEE IF OK

```

RES

16:24:48 16-MAY-80

```

1724
050.345 021 071 054 1725 LXI D,LINE
050.350 301 1726 POP B (B) = N+2
050.351 110 1727 MOV C,B
050.352 006 000 1728 MVI B,0 (BC) = N+2
050.354 315 335 047 1729 CALL CBS CRC BYTE STRING
050.357 174 1730 MOV A,H
050.360 265 1731 ORA L CRC OF STRING AND CRC MUST BE 0
050.361 311 1732 RET
1733
050.362 012 014 1734 DB NL,FF SHOW MESSAGE TO SNOOPS
1735 ** * WARNING: THIS MESSAGE MUST BE AN EVEN NUMBER OF BYTES * *
050.364 110 105 101 1736 RESA DB 'HEATH SOFTWARE PATCH UTILITY'
051.024 014 1737 RESAE DB FF END ADDRESS+1 OF MESSAGE

1739 ** EES - ENCODE ENCODED STRING.
1740 *
1741 * IF THE 'C' OPTION IS SELECTED IN '.SYS.' VERSIONS
1742 * OF PATCH, THEN EES IS CALLED BEFORE RES.
1743 *
1744 * EES ACCEPTS A VALUE FROM THE CONSOLE, IN THE FORM
1745 * OF OCTAL BYTES, ALL OF WHICH MUST BE 3 DIGITS, BUT
1746 * WHICH MAY BE SEPERATED BY BLANKS.
1747 *
1748 * THESE N BYTES WILL BE ENCRYPTED INTO THE 2*N+4 BYTE ALPHA STRING
1749 * RES REQUIRES, AND THAT STRING WILL BE TYPED FOR THE USER.
1750 *
1751 * ENTRY (B) = NUMBER OF BYTES WANTED
1752 * EXIT NONE
1753 * USES NONE
1754
000.001 1755 IF .SYS.
1756
1757 EES CAL $SAVALL SAVE REGISTERS
1758 EES0 CALL $TYPTX
1759 DB NL,'Enter Value as Octal Bytes:',',','+2000
1760 LXI H,LINE
1761 CALL $RTL.
1762 JC $RSTALL RESTORE ALL AND EXIT, CTL-D
1763 LXI D,EESA
1764 MOV C,B (C) = BYTES WANTED
1765 EES1 MOV A,C
1766 ANA A
1767 JZ EES3 ALL DONE INPUTTING, TYPE VALUE
1768 PUSH B
1769 PUSH D
1770 CALL $SOB
1771 CALL DOB DECODE OCTAL BYTE
1772 POP D
1773 POP B
1774 JC EES0 TRY AGAIN
1775 STAX D STORE VALUE
1776 INX D INCR ADDRESS
1777 DCR C

```

```

1778      JMP      EES1
1779
1780      *      GOT VALUES. CRC IT
1781
1782      EES3    LXI      D,EESA
1783      MOV      C,B
1784      PUSH     B      SAVE COUNT
1785      MVI      B,0      (BC) = BYTE COUNT
1786      CALL     CBS      CRC BYTE STRING
1787      XCHG
1788      MOV      M,D
1789      INX      H
1790      MOV      M,E      STORE CRC
1791      POP      B
1792
1793      *      TYPE ENCRYPTED STRING.
1794      *      (B) = # OF BYTES -2
1795
1796      CALL     $TYPTX
1797      DB      'Encoded Entry =',',',',t2000
1798      INR      B
1799      INR      B
1800      LXI      D,EESA
1801      LXI      H,RESA
1802      EES4    LDAX     D      (A) = VALUE
1803      RAR
1804      RAR
1805      RAR
1806      RAR
1807      XRA      M
1808      ANI      170
1809      ORI      '0'
1810      INR      A
1811      CALL     $WCHAR      TYPE CHARACTER
1812      LDAX     D
1813      INX      H
1814      XRA      M
1815      ANI      170
1816      ORI      '0'
1817      INR      A
1818      CALL     $WCHAR      TYPE 2ND CHARACTER
1819      INX      H
1820      INX      D
1821      MVI      A,$RESAE
1822      CMP      L
1823      JNE      EES5
1824      LXI      H,RESA
1825      EES5    DCR      B
1826      JNZ      EES4      MORE TO GO
1827      CALL     $TYPTX
1828      DB      ENL      CRLF
1829      JMP      $RSTALL      RESTORE AND EXIT
1830
1831      EESA    DS      80      LINE BUFFER
1832
1833      ENDIF

```

```

1835 **      UBH - UPDATE BINARY HEADER.
1836 *
1837 *      UBH UPDATES THE LENGTH FIELD IN THE PROGRAMS BINARY HEADER
1838 *      TO REFLECT ANY PATCHES ADDED TO THE END OF THE PROGRAM.
1839 *
1840 *      ENTRY  NONE
1841 *      EXIT   NONE
1842 *      USES   ALL
1843
1844
051.025 001 000 000 1845 UBH LXI B,0
051.030 315 235 050 1846 CALL PPF REWIND FILE
051.033 332 324 042 1847 JC ERROR
051.034 001 000 001 1848 LXI B,256
051.041 021 000 056 1849 LXI D,BUFFER
051.044 076 000 1850 MVI A,CN,FIL
051.046 377 004 1851 DB SYSCALL,.READ READ IN HEADER TABLE
051.050 332 324 042 1852 JC ERROR
1853
1854 *      SET NEW PROGRAM LENGTH.
1855
051.053 052 264 053 1856 LHLD PGMLOW
051.056 353 1857 XCHG
051.057 052 262 053 1858 LHLD PGMFWA (HL) = FWA
051.062 173 1859 MOV A,E
051.063 225 1860 SUB L
051.064 157 1861 MOV L,A
051.065 172 1862 MOV A,D
051.066 234 1863 SBB H
051.067 147 1864 MOV H,A (HL) = NEW LENGTH-1
051.070 043 1865 INX H (HL) = NEW LENGTH
051.071 072 252 053 1866 LDA FILTYP
051.074 376 000 1867 CPI FT.ABS
051.076 312 107 051 1868 JE UBH1 IS ABSOLUTE BINARY
1869
1870 *      IS PIC PROGRAM
1871
051.101 042 002 056 1872 SHLD BUFFER+PIC.LEN
051.104 303 112 051 1873 JMP UBH2
1874
1875 *      IS ABS PROGRAM
1876
051.107 042 004 056 1877 UBH1 SHLD BUFFER+ABS.LEN
051.112 001 000 000 1878 UBH2 LXI B,0
051.115 315 235 050 1879 CALL PPF REWIND AGAIN
051.120 332 324 042 1880 JC ERROR
051.123 001 000 001 1881 LXI B,256
051.126 021 000 056 1882 LXI D,BUFFER
051.131 076 000 1883 MVI A,CN,FIL
051.133 377 005 1884 DB SYSCALL,.WRITE REPLACE HEADER
051.135 320 1885 RNC RETURN IF OK
051.136 303 324 042 1886 JMP ERROR

```



```

1888 **      WPH - WRITE PATCH HISTORY.
1889 *
1890 *      WPH UPDATES THE PATCH HISTORY TABLE, AND APPENDS IT TO THE END OF
1891 *      THE BINARY FILE.
1892 *
1893 *      ENTRY  NONE
1894 *      EXIT  NONE
1895 *      USES  ALL
1896
1897
051.141 072 260 053 1898 WPH  LDA  PATCHID
051.144 107          1899  MOV  B,A          (B) = PATCH ID
051.145 346 370     1900  ANI  370H
051.147 037         1901  RAR
051.150 037         1902  RAR
051.151 037         1903  RAR          (A) = PATCHID/8
051.152 041 023 057 1904  LXI  H,PHT+PHT.HIS
051.155 315 101 030 1905  CALL $DADA.      (HL) = ADDRESS OF BYTE FOR PATCH CODE
1906
1907 *      COMPUTE BIT TO SET TO INDICATE THIS PATCH
1908
051.160 170         1909  MOV  A,B
051.161 346 007     1910  ANI  7
051.163 107         1911  MOV  B,A          (B) = BIT INDEX
051.164 257         1912  XRA  A
051.165 067         1913  STC
051.166 027         1914 WPH1  RAL
051.167 005         1915  DCR  B
051.170 362 166 051 1916  JF  WPH1
051.173 266         1917  ORA  M          SET BIT
051.174 167         1918  MOV  M,A
1919
1920 **      WPH. - WRITE PHT TO END OF FILE
1921 *
1922 *      WRITE PHT TO END OF FILE
1923
051.175 052 264 053 1924 WPH.  LHLD  PGMLWA
051.200 353         1925  XCHG
051.201 052 230 053 1926  LHLD  SKEW          COMPUTE LAST PROGRAM SECTOR USED
051.204 031         1927  DAD  D          (HL) = FILE ADDRESS OF LAST PROGRAM BYTE
051.205 114         1928  MOV  C,H
051.206 006 000     1929  MVI  B,0          (BC) = SECTOR NUMBER
051.210 003         1930  INX  B          POINT TO ONE AFTER, FOR PHT
051.211 315 235 050 1931  CALL PPF          POSITION TO IT
051.214 332 324 042 1932  JC  ERROR          COULDN'T GET THERE
051.217 001 000 001 1933  LXI  B,256
051.222 021 000 057 1934  LXI  D,PHT
051.225 076 000     1935  MVI  A,CN.FIL
051.227 377 005     1936  DB  SYSCALL,WRITE  WRITE PHT ON FILE
051.231 320         1937  RNC          RETURN IF NO ERROR
051.232 303 324 042 1938  JMP  ERROR

```

051.235

1941

XTEXT DNS

```

1943X **      $DNS - DECODE NUMERIC SWITCH.
1944X *
1945X *      $DNS DECODES A NUMERIC SWITCH OF THE FORM:
1946X *
1947X *      :NNN
1948X *
1949X *      A POSTRADIX OF D, Q, O, OR B IS ALLOWED. IF THE VALUE
1950X *      IS SYNTACTICALLY VALID, IT IS REPLACED WITH BLANKS.
1951X *
1952X *      ENTRY (HL) = ADDRESS IF ':'
1953X *      (A) = DEFAULT BASE (2, 8 OR 10)
1954X *      EXIT 'C' CLEAR IF OK
1955X *      (HL) ADVANCED PAST VALUE
1956X *      VALUE BLANKED
1957X *      (DE) = VALUE
1958X *      'C' SET IF ERROR
1959X *      USES ALL
1960X
1961X
051.235 076 012 1962X $DNS. MVI A,10      BASE 10 DEFAULT
051.237 107      1963X $DNS MOV B,A      (B) = DEFAULT BASE
051.240 176      1964X MOV A,M
051.241 376 072 1965X CPI ':'
051.243 067      1966X STC
051.244 300      1967X RNE              NOT ':'
051.245 345      1968X PUSH H          SAVE ADDRESS OF SWITCH START
051.246 043      1969X INX H
051.247 170      1970X MOV A,B
051.250 315 274 051 1971X CALL $DNV    DECODE NUMERIC VALUE
051.253 301      1972X POP B          (BC) = ADDRESS OF ':'
051.254 330      1973X RC              ERROR
051.255 076 040 1974X $DNS1 MVI A,' '
051.257 002      1975X STAX B          BLANK LINE
051.260 003      1976X INX B          INCREMENT ADDRESS
051.261 175      1977X MOV A,L
051.262 271      1978X CMP C
051.263 302 255 051 1979X JNE $DNS1
051.266 170      1980X MOV A,B
051.267 274      1981X CMP H          SEE IF IN RIGHT BANK
051.270 302 255 051 1982X JNE $DNS1
051.273 311      1983X RET            RETURN WITH 'C' CLEAR AND VALUE
051.274      1984X XTEXT DNV

```

*DNV

```

1986X **      $DNV - DECODE NUMERIC VALUE.
1987X *
1988X *      $DNV DECODES A NUMERIC VALUE (IN THE FORM OF AN ASCII STRING)
1989X *      INTO A BINARY NUMBER. THE MAXIMUM MAGNITUDE IS
1990X *      65535D.
1991X *
1992X *      THE NUMBER MAY CONTAIN A POSTRADIX OF 'B' (BINARY)
1993X *      'O' OR 'Q' (OCTAL) OR 'D' (DECIMAL)
1994X *
1995X *      ENTRY (HL) = ADDRESS OF FIRST BYTE OF NUMBER
1996X *      (A) = DEFAULT BASE (2 FOR BINARY, 10 FOR DECIMAL, ETC.)
1997X *      EXIT 'C' CLEAR IF OK
1998X *      (HL) ADVANCED PAST NUMBER (AND POSTRADIX)
1999X *      (DE) = VALUE
2000X *      'C' SET IF ERROR
2001X *      USES ALL
2002X
2003X
051.274 062 011 052 2004X $DNV STA $DNVA SET DEFAULT BASE
051.277 104 2005X MOV B,H
051.300 115 2006X MOV C,L (BC) = TEXT ADDRESS
2007X
2008X *      SCAN FOR POSTRADIX
2009X
051.301 176 2010X $DNV1 MOV A,M
051.302 315 055 052 2011X CALL $CVD CHECK FOR VALID DECIMAL DIGIT
051.305 043 2012X INX H
051.306 322 301 051 2013X JNC $DNV1 MORE TO GO
051.311 053 2014X DCX H REMOVE EXTRA INCREMENT
051.312 171 2015X MOV A,C
051.313 275 2016X CMP L SEE IF THERE WERE ANY NUMBERS
051.314 067 2017X STC ASSUME NOT
051.315 310 2018X RE ERROR
2019X
2020X *      OUT OF NUMBERS. SEE IF POSTRADIX FOLLOWS
2021X
051.316 176 2022X MOV A,M (A) = PROPOSED POSTRADIX
051.317 345 2023X PUSH H SAVE END ADDRESS
051.320 041 012 052 2024X LXI H,$DNVB
051.323 247 2025X ANA A
051.324 312 344 051 2026X JZ $DNV2 NO POSTRADIX
051.327 315 023 052 2027X CALL $TRLS
051.332 176 2028X MOV A,M
051.333 302 344 051 2029X JNE $DNV2 NOT POSTRADIX
051.336 341 2030X POP H
051.337 043 2031X INX H SKIP POSTRADIX
051.340 345 2032X PUSH H
051.341 062 011 052 2033X STA $DNVA SET NEW POSTRADIX
051.344 021 000 000 2034X $DNV2 LXI D,0 (DE) = ACCUMULATOR
2035X
2036X *      BUILD NUMBER
2037X
051.347 072 011 052 2038X $DNV3 LDA $DNVA (A) = BASE
051.352 365 2039X PUSH PSW SAVE BASE
051.353 315 007 031 2040X CALL $MUB6 MULTIPLY
051.356 321 2041X POP D (D) = BASE

```

\$DNU

16:24:59 16-MAY-80

051.357	332	007	052	2042X	JC	\$DNU4	OVERFLOW
051.362	012			2043X	LDAX	B	(A) = DIGIT
051.363	326	060		2044X	SUI	'0'	
051.365	003			2045X	INX	B	
051.366	272			2046X	CMF	D	COMPARE TO BASE
051.367	077			2047X	CMC		
051.370	332	007	052	2048X	JC	\$DNU4	TOO LARGE A DIGIT
051.373	315	101	030	2049X	CALL	\$DADA.	ADD TO VALUE
051.376	353			2050X	XCHG		(DE) = VALUE
051.377	012			2051X	LDAX	B	
052.000	315	055	052	2052X	CALL	\$CVD.	
052.003	322	347	051	2053X	JNC	\$DNU3	MORE TO GO
052.006	247			2054X	ANA	A	CLEAR CARRY
052.007	341			2055X	\$DNU4	POP	RESTORE POINTER
052.010	311			2056X	RET		EXIT
				2057X			
052.011	000			2058X	\$DNUA	DB	0
052.012	102	002		2059X	\$DNUB	DB	'B',2
052.014	117	010		2060X		DB	'0',8
052.016	121	010		2061X		DB	'R',8
052.020	104	012		2062X		DB	'D',10
052.022	000			2063X		DB	0
052.023				2064	XTEXT	TBLS	

2066X ** \$TBLS - TABLE SEARCH

2067X *

2068X * TABLE FORMAT

2069X *

2070X * DB KEY1,VAL1,

2071X *

2072X *

2073X * DB KEYN,VALN

2074X *

2075X *

2076X * ENTRY (A) = PATTERN

2077X *

(H,L) = TABLE FWA

2078X * EXIT (A) = PATTERN IF FOUND

2079X *

'Z' SET IF FOUND

2080X * 'Z' CLEAR IF NOT FOUND OR PATTERN=0

2081X *

2082X

2083X

052.023 305 2084X \$TBLS

PUSH

B

052.024 376 000

2085X

CPI

0

052.026 312 050 052

2086X

JZ

TBL2

052.031 107

2087X

MOV

B,A

052.032 176

2088X TBL1

MOV

A,M

(A) = CHARACTER

052.033 043

2089X

INX

H

052.034 270

2090X

CMF

B

052.035 312 052 052

2091X

JZ

TBL3

IF MATCH

052.040 247

2092X

ANA

A

052.041 043

2093X

INX

H

SKIP PAST

052.042 302 032 052

2094X

JNZ

TBL1

IF NOT END OF TABLE

052.045	053	2095X	DCX	H	
052.046	053	2096X	DCX	H	
052.047	257	2097X	XRA	A	
052.050	376 001	2098X TBL2	CPI	1	SET TO ZERO FOR OLD USERS /78.10.GC/
		2099X			CLEAR ZERO /78.10.GC/
		2100X *	DONE		
		2101X			
052.052	301	2102X TBL3	POP	B	
052.053	311	2103X	RET		
052.054		2104	XTEXT	MU86	

2106X **	\$MU86 - MULTIPLY 8X16 UNSIGNED.
2107X *	
2108X *	\$MU86 MULTIPLIES A 16 BIT VALUE BY A 8
2109X *	BIT VALUE.
2110X *	
2111X *	ENTRY (A) = MULTIPLIER
2112X *	(DE) = MULTIPLICAND
2113X *	EXIT (HL) = RESULT
2114X *	'Z' SET IF NOT OVERFLOW
2115X *	USES A,F,H,L
2116X	
2117X	
031.007	2118X \$MU86 EQU 31007A IN H17 ROM
052.054	2119 XTEXT CVD

2121X **	\$CVD - CHECK FOR VALID DIGIT.
2122X *	
2123X *	CVD EXAMINES A DIGIT TO SEE IF IT IS A VALID DECIMAL DIGIT.
2124X *	
2125X *	ENTRY (HL) = ADDRESS OF CHARACTER
2126X *	EXIT 'C' SET IF ILLEGAL
2127X *	(A) = VALUE
2128X *	USES A,F
2129X	
2130X	
052.054	2131X \$CVD MOV A,M (A) = CHARACTER
052.055	2132X \$CVD. SUI '0'
052.057	2133X RC ILLEGAL
052.060	2134X CPI 9+1
052.062	2135X CMC
052.063	2136X RET
052.064	2137 XTEXT DRS

```

2139X ** $DRS - DECODE AND REMOVE SWITCHES.
2140X *
2141X * $DRS IS CALLED TO DECODE COMMAND SWITCHES FROM A LINE
2142X * OF TEXT. SWITCHES TAKE THE FORM:
2143X *
2144X * /XXXXX
2145X *
2146X * AFTER A SWITCH HAS BEEN LOCATED, IT (AND THE PRECEDING '/')
2147X * ARE REPLACED WITH BLANKS.
2148X *
2149X * VALID SWITCH DESCRIPTIONS ARE ENCODED INTO A TABLE
2150X * SUPPLIED BY THE CALLER, IN THE FORMAT:
2151X *
2152X * DB 'X...X' REQUIRED SWITCH CHARACTERS
2153X * DB 'C'+200Q,...,'C'+200Q OPTIONAL CHARACTERS
2154X * DB 200Q END OF CHARACTERS
2155X * DW ADDR PROCESSOR ADDRESS (CALLED WHEN SWITCH DETECTED)
2156X *
2157X * DB 'Y...Y' NEXT SWITCH
2158X *
2159X *
2160X *
2161X *
2162X * DB 0 FLAGS END OF TABLE
2163X *
2164X * SWITCHES MUST BE FOLLOWED BY A ':', A '/' (ANOTHER SWITCH)
2165X * A ',', OR A 00 BYTE.
2166X *
2167X * UPON DETECTION OF A VALID SWITCH, $DRS CALLS THE USER PROCESS
2168X * ROUTINE. UPON ENTRY,
2169X * (HL) = ADDRESS OF THE FIRST BYTE FOLLOWING THE SWITCH
2170X * 'Z' CLEAR IF CHARACTER = '/', ',', OR 00
2171X * 'Z' SET IF CHARACTER = ':'
2172X *
2173X * THE USER ROUTINE CAN DECODE SWITCH SUB-OPTIONS, IF DESIRED.
2174X * THE USER ROUTINE MAY USE ALL REGISTERS.
2175X *
2176X * ENTRY (DE) = SWITCH TABLE FWA
2177X * (HL) = LINE FWA
2178X * EXIT 'C' CLEAR IF OK
2179X * 'C' SET IF ERROR
2180X * (HL) = ADDRESS OF START OF BAD SWITCH
2181X * (A) = ERROR CODE
2182X * USES ALL
2183X *
2184X *
052.064 2185X $DRS EQU *
2186X *
2187X * LOOK FOR SWITCHES
2188X *
052.064 176 2189X $DRS1 MOV A,M
052.065 247 2190X ANA A
052.066 310 2191X RZ END OF LINE
052.067 043 2192X INX H
052.070 376 057 2193X CPI '/'
052.072 302 064 052 2194X JNE $DRS1 NOT A SWITCH

```

```

052.075 042 261 052 2195X SHLD *DRSB ($DRSB) = SWITCH FWA (AFTER '/')
2196X
2197X * GOT A SWITCH. LOOK FOR A MATCH IN THE CALLER'S TABLE
2198X
052.100 325 2199X PUSH D SAVE TABLE FWA
052.101 052 261 052 2200X $DRS2 LHLD $DRSB (HL) = SWITCH FWA
052.104 032 2201X $DRS3 LDAX D (A) = TABLE ENTRY
052.105 346 177 2202X ANI 177H
052.107 312 157 052 2203X JZ $DRS6 GOT A MATCH
052.112 276 2204X CMP H
052.113 302 123 052 2205X JNE $DRS4 NO MATCH
052.116 023 2206X INX D
052.117 043 2207X INX H
052.120 303 104 052 2208X JMP $DRS3 SEE IF MORE MATCH
2209X
2210X * HAVE MIS-MATCH. SEE IF THE MISSING CHARACTER IS SIGNIFICANT
2211X
052.123 176 2212X $DRS4 MOV A,M (A) = LINE CHARACTER WE COULDN'T MATCH
052.124 315 230 052 2213X CALL $DRS15 SEE IF OK TERMINATOR
052.127 302 137 052 2214X JNE $DRS4.5 NO MATCH ON THIS SWITCH
052.132 032 2215X LDAX D (A) = NEXT CHARACTER IN SWITCH PATTERN
052.133 247 2216X ANA A
052.134 372 157 052 2217X JM $DRS6 HAVE SUFFICIENT MATCH
052.137 315 243 052 2218X $DRS4.5 CALL $DRS20 SKIP TABLE ENTRY
052.142 032 2219X LDAX D
052.143 247 2220X ANA A
052.144 302 101 052 2221X JNZ $DRS2 MORE SWITCHES IN TABLE TO CHECK
2222X
2223X * BAD SWITCH
2224X
052.147 321 2225X $DRS5 POP D RESTORE STACK
052.150 052 261 052 2226X LHLD $DRSB POINT TO BAD SWITCH
052.153 067 2227X STC
052.154 076 032 2228X MVI A,EC,IS ILLEGAL SWITCH
052.156 311 2229X RET
2230X
2231X * HAVE SWITCH. CHECK IT'S FOLLOWING CHARACTER
2232X
052.157 315 263 052 2233X $DRS6 CALL $SOB SKIP OVER BLANKS
052.162 176 2234X MOV A,M
052.163 315 230 052 2235X CALL $DRS15 CHECK CHARACTER
052.166 302 147 052 2236X JNE $DRS5 IN ERROR
052.171 315 243 052 2237X CALL $DRS20 GET PROCESSOR ADDRESS
052.174 021 206 052 2238X LXI D,$DRS7
052.177 345 2239X PUSH H
052.200 325 2240X PUSH D SAVE (HL)
052.201 305 2241X PUSH B SET RETURN ADDRESS FOR TABLE CODE
052.202 176 2242X MOV A,M SAVE PROCESSOR ADDRESS
052.203 376 072 2243X CPI '/' (A) = NEXT CHARACTER
052.205 311 2244X RET SET CONDITION CODES
2245X CALL USER PROCESS
2246X * USER PROCESS RETURNS HERE
2247X
052.206 321 2248X $DRS7 POP D (DE) = LAST CHARACTER OF SWITCH+1
052.207 052 261 052 2249X LHLD $DRSB (HL) = FIRST CHARACTER OF SWITCH AFTER /
052.212 053 2250X DCX H (HL) = ADDRESS OF '/'

```

```

2251X
2252X *      REPLACE SWITCH WITH BLANKS
2253X
052.213 066 040 2254X $DRS8 MVI    M, ' '
052.215 043      2255X      INX    H
052.216 315 216 030 2256X      CALL  $CDEHL
052.221 302 213 052 2257X      JNE    $DRS8      NOT THERE YET
052.224 321      2258X      POP    D      (DE) = SWITCH TABLE FWA
052.225 303 064 052 2259X      JMP    $DRS1      LOOK FOR MORE SWITCHES

2261X **      $DRS15 - CHECK FOR VALID DELIMITER CHARACTER.
2262X *
2263X *      $DRS15 CHECKS THE NEXT TEXT CHARACTER TO SEE IF IT IS
2264X *
2265X *      00, '/', ',', ';
2266X *
2267X *      ENTRY    (A) = CHARACTER
2268X *      EXIT    'Z' SET IFF CHARACTER IS ONE OF THE ABOVE
2269X *      USES    F
2270X
052.230 247      2271X $DRS15 ANA    A
052.231 310      2272X      RZ              IS 00
052.232 376 057 2273X      CPI    '/'
052.234 310      2274X      RE
052.235 376 054 2275X      CPI    ','
052.237 310      2276X      RE
052.240 376 072 2277X      CPI    ';'
052.242 311      2278X      RET

2280X **      $DRS20 - GET PROCESSOR ADDRESS.
2281X *
2282X *      $DRS20 IS CALLED TO GET THE PROCESSOR ADDRESS FIELD OUT OF
2283X *      AN ENTRY IN THE SWITCH TABLE. THE CALLER SUPPLIES A POINTER
2284X *      TO SOMEWHERE IN THE TEXT PART OF THE SWITCH DESCRIPTION;
2285X *      $DRS20 ADVANCES THE POINTER TO THE PROCESSOR ADDRESS.
2286X *
2287X *      ENTRY    (DE) = POINTER TO TEXT PART OF SWITCH ENTRY
2288X *      EXIT    (DE) = POINTER TO 1ST BYTE OF NEXT SWITCH TABLE ENTRY
2289X *      (BC) = PROCESSOR ADDRESS FROM TABLE
2290X *      USES    A,F,B,C,D,E
2291X
2292X
052.243 032      2293X $DRS20 LDAX   D
052.244 023      2294X      INX    D
052.245 376 200 2295X      CPI    200H
052.247 302 243 052 2296X      JNE    $DRS20
052.252 032      2297X      LDAX   D      (A) = LOW BYTE OF PROCESSOR ADDRESS
052.253 117      2298X      MOV    C,A
052.254 023      2299X      INX    D
052.255 032      2300X      LDAX   D
052.256 107      2301X      MOV    B,A      (BC) = PROCESSOR ADDRESS
052.257 023      2302X      INX    D
052.260 311      2303X      RET
2304X

```


052.261 000 000 2305X \$DRSB DW 0 POINTER TO SWITCH BEING PROCESSED
052.263 2306 XTEXT SOB

2308X ** \$SOB - SKIP OVER BLANKS.
2309X *
2310X * \$SOB IS CALLED TO SKIP AN ARBITRARILY LONG STRING OF BLANKS AND TABS.
2311X *
2312X * ENTRY (HL) = FWA OF (POSSIBLE) BLANK STRING
2313X * EXIT (HL) = LWA+1 OF BLANK STRING (UNCHANGED IF NO BLANKS)
2314X * (A) = FIRST NON-BLANK, NON-TAB CHARACTER EEN
2315X * USES A,F,H,L
2316X
2317X
052.263 053 2318X \$SOB DCX H PRE-DECREMENT
052.264 043 2319X \$SOB1 INX H
052.265 176 2320X MOV A,M
052.266 376 040 2321X CPI /
052.270 312 264 052 2322X JE \$SOB1 GOT BLANK
052.273 376 011 2323X CPI TAB
052.275 312 264 052 2324X JE \$SOB1 GOT TAB
052.300 311 2325X RET
052.301 2326 XTEXT CHL

2328X ** \$CHL = COMPLEMENT (HL).
2329X *
2330X * (HL) = -(HL) TWO'S COMPLEMENT
2331X *
2332X * ENTRY NONE
2333X * EXIT NONE
2334X * USES A,F,H,L
2335X
2336X
030.224 2337X \$CHL EQU 30224A IN H17 ROM
052.301 2338 XTEXT MOVE

2340X ** \$MOVE - MOVE DATA
2341X *
2342X * \$MOVE MOVES A BLOCK OF BYTES TO A NEW MEMORY ADDRESS.
2343X * IF THE MOVE IS TO A LOWER ADDRESS, THE BYTES ARE MOVED FROM
2344X * FIRST TO LAST.
2345X *
2346X * IF THE MOVE IS TO A HIGHER ADDRESS, THE BYTES ARE MOVED FROM
2347X * LAST TO FIRST.
2348X *
2349X * THIS IS DONE SO THAT AN OVERLAPED MOVE WILL NOT 'RIPPLE'.
2350X *
2351X * ENTRY (BC) = COUNT

```

2352X *      (DE) = FROM
2353X *      (HL) = TO
2354X *      EXIT  MOVED
2355X *      (DE) = ADDRESS OF NEXT FROM BYTE
2356X *      (HL) = ADDRESS OF NEXT *TO* BYTE
2357X *      'C' CLEAR
2358X *      USES  ALL
2359X
2360X
030.252      2361X $MOVE EQU 30252A      IN H17 ROM
052.301      2362      XTEXT  CCO

```

```

2364X **      $CCO - CLEAR CONTROL-0
2365X *
2366X *      $CCO IS CALLED TO CLEAR THE EFFECT OF THE CTL-0 CHARACTER.
2367X *
2368X *      ENTRY  NONE
2369X *      EXIT  NONE
2370X *      USES  NONE
2371X
2372X
052.301      315 054 031 2373X $CCO CALL $SAVALL      SAVE REGISTERS
052.304      076 004      2374X MVI A,I,CONFL
052.306      001 001 000 2375X LXI B,CO.FLG      CLEAR CO.FLG
052.311      377 006      2376X DB SYSCALL,CONSL
052.313      303 047 031 2377X JMP $RSTALL      RESTORE REGISTERS AND RETURN
052.316      2378      XTEXT  SAVALL

```

```

2380X **      $RSTALL - RESTORE ALL REGISTERS.
2381X *
2382X *      $RSTALL RESTORES ALL THE REGISTERS OFF THE STACK, AND
2383X *      RETURNS TO THE PREVIOUS CALLER.
2384X *
2385X *      ENTRY  (SP) = PSW
2386X *      (SP+2) = BC
2387X *      (SP+4) = DE
2388X *      (SP+6) = HL
2389X *      (SP+8) = RET
2390X *      EXIT  TO *RET*, REGISTERS RESTORED
2391X *      USES  ALL
2392X
2393X
031.047      2394X $RSTALL EQU 31047A      IN H17 ROM

```

```

2396X **      $SAVALL - SAVE ALL REGISTERS ON STACK.
2397X *
2398X *      $SAVALL SAVES ALL THE REGISTERS ON THE STACK.
2399X *
2400X *      ENTRY  NONE
2401X *      EXIT   (SF) = PSW
2402X *      (SF+2) = BC
2403X *      (SF+4) = DE
2404X *      (SF+6) = HL
2405X *      USES   H,L
2406X
2407X
031.054      2408X $SAVALL EQU 31054A      IN H17 ROM
052.316      2409      XTEXT  HLIHL

```

```

2411X **      $HLIHL - LOAD HL INDIRECT THROUGH HL.
2412X *
2413X *      (HL) = ((HL))
2414X *
2415X *      ENTRY  NONE
2416X *      EXIT   NONE
2417X *      USES   A,H,L
2418X
030.211      2419X $HLIHL EQU 30211A      IN H17 ROM
052.316      2420      XTEXT  CDEHL

```

```

2422X **      $CDEHL - COMPARE (DE) TO (HL)
2423X *
2424X *      $CDEHL COMPARES (DE) TO (HL) FOR EQUALITY.
2425X *
2426X *      ENTRY  NONE
2427X *      EXIT   'Z' SET IF (DE) = (HL)
2428X *      USES   A,F
2429X
030.216      2431X $CDEHL EQU 30216A      IN H17 ROM
052.316      2432      XTEXT  CRC

```

```

2434X **      $CRC - COMPUTE CRC16
2435X *
2436X *
2437X *      COMPUTE THE CRC16 CHECKSUM.
2438X *
2439X *      ENTRY  (HL) = CURRENT CHECKSUM
2440X *      (A) = BYTE
2441X *      EXIT  (HL) UPDATED
2442X *      (A) UNCHANGED.

```

\$CRC

```

2443X *      USES      F,H,L
2444X
2445X
052.316 305   2446X $CRC  PUSH      B      SAVE (BC)
052.317 006 010 2447X      MVI      B,B      (B) = BIT COUNT
052.321 007   2448X $CRC1  RLC
052.322 117   2449X      MOV      C,A      (C) = BIT
052.323 175   2450X      MOV      A,L
052.324 207   2451X      ADD      A
052.325 157   2452X      MOV      L,A
052.326 174   2453X      MOV      A,H
052.327 027   2454X      RAL
052.330 147   2455X      MOV      H,A
052.331 027   2456X      RAL
052.332 251   2457X      XRA      C
052.333 017   2458X      RRC
052.334 322 347 052 2459X      JNC      $CRC2      IF NOT TO XOR
052.337 174   2460X      MOV      A,H
052.340 356 200   2461X      XRI      200H
052.342 147   2462X      MOV      H,A
052.343 175   2463X      MOV      A,L
052.344 356 005   2464X      XRI      50
052.346 157   2465X      MOV      L,A
052.347 171   2466X $CRC2  MOV      A,C
052.350 005   2467X      DCR      B
052.351 302 321 052 2468X      JNZ      $CRC1      IF MORE TO GO
052.354 301   2469X      POP      B      RESTORE (BC)
052.355 311   2470X      RET      EXIT
052.356       2471X      XTEXT    COMP

```

```

2473X **      $COMP - COMPARE TWO CHARACTER STRINGS.
2474X *
2475X *      $COMP COMPARES TWO BYTE STRINGS.
2476X *
2477X *      ENTRY      (C) = COMPARE COUNT
2478X *      (DE) = FWA OF STRING #1
2479X *      (HL) = FWA OF STRING #2
2480X *      EXIT      'Z' CLEAR, IS MIS-MATCH
2481X *      (C) = LENGTH REMAINING
2482X *      (DE) = ADDRESS OF MISMATCH IN STRING#1
2483X *      (HL) = ADDRESS OF MISMATCH IN STRING #2
2484X *      'C' SET, HAVE MATCH
2485X *      (C) = 0
2486X *      (DE) = (DE) + (0C)
2487X *      (HL) = (HL) + (0C)
2488X *      USES      A,F,C,D,E,H,L
2489X
2490X
030.060       2491X $COMP  EQU      30060A      IN H17 ROM
052.356       2492X      XTEXT    MCU

```

```

2494X **      MCU - MAP LOWER CASE TO UPPER CASE.
2495X *
2496X *      MCU MAPS A LOWER CASE ALPHABETIC TO UPPER
2497X *      CASE.
2498X *
2499X *      ENTRY (A) = CHARACTER
2500X *      EXIT (A) = CHARACTER RESULT
2501X *      USES A,F
2502X
2503X
052.356 376 141 2504X $MCU CPI 'a'
052.360 330 2505X RC NOT LOWER CASE
052.361 376 173 2506X CPI 'Z'+1
052.363 320 2507X RNC NOT LOWER CASE
052.364 326 040 2508X SUI 'a'-'A'
052.366 311 2509X RET
052.367 2510 XTEXT TYPTX

```

```

2512X **      $TYPTX - TYPE TEXT.
2513X *
2514X *      $TYPTX IS CALLED TO TYPE A BLOCK OF TEXT ON THE SYSTEM CONSOLE.
2515X *
2516X *      IMBEDDED ZERO BYTES INDICATE A CARRIAGE RETURN LINE FEED.
2517X *      A BYTE WITH THE 200Q BIT SET IS THE LAST BYTE IN THE MESSAGE.
2518X *
2519X *      ENTRY (RET) = TEXT
2520X *      EXIT TO (RET+LENGTH)
2521X *      USES A,F
2522X
2523X
031.136 2524X $TYPTX EQU 31136A IN H17 ROM
2525X
031.144 2526X $TYPTX EQU 31144A IN H17 ROM
052.367 2527 XTEXT RCHAR

```

```

2529X **      $RCHAR - READ SINGLE CHARACTER FROM CONSOLE.
2530X *
2531X *      ENTRY NONE
2532X *      EXIT (A) = CHARACTER
2533X *      USES A,F
2534X
2535X
052.367 377 001 2536X $RCHAR DB SYSCALL, SCIN
052.371 332 367 052 2537X JC $RCHAR NOT READY
052.374 311 2538X RET
2539X
052.375 377 002 2540X $WCHAR DB SYSCALL, SCOUT
052.377 311 2541X RET
053.000 2542 XTEXT DADA

```

```

2544X **      $DADA - PERFORM (H,L) = (H,L) + (O,A)
2545X *
2546X *      ENTRY (H,L) = BEFORE VALUE
2547X *      (A) = BEFORE VALUE
2548X *      EXIT (H,L) = (H,L) + (O,A)
2549X *      'C' SET IF OVERFLOW
2550X *      USES F,H,L
2551X
2552X
030.072      2553X $DADA EQU 30072A IN H17 ROM
053.000      2554      XTEXT DADA2

```

```

2556X **      $DADA. - ADD (O,A) TO (H,L)
2557X *
2558X *      ENTRY NONE
2559X *      EXIT (HL) = (HL) + (OA)
2560X *      USES A,F,H,L
2561X
2562X
030.101      2563X $DADA. EQU 30101A IN H17 ROM
053.000      2564      XTEXT DU66

```

```

2566X **      $DU66 - UNSIGNED 16 / 16 DIVIDE.
2567X *
2568X *      (HL) = (BC)/(DE)
2569X *
2570X *      ENTRY (BC), (DE) PRESET
2571X *      EXIT (HL) = RESULT
2572X *      (DE) = REMAINDER
2573X *      USES ALL
2574X
2575X
030.106      2576X $DU66 EQU 30106A IN H17 ROM
053.000      2577      XTEXT MLU

```

```

2579X **      MLU - MAP LOWER CASE LINE TO UPPER CASE.
2580X *
2581X *      MLU MAPS THE LOWER CASE ALPHABETICS IN A LINE TO UPPER CASE.
2582X *
2583X *      ENTRY (HL) = LINE FWA
2584X *      EXIT NONE
2585X *      USES NONE
2586X
2587X
053.000 345      2588X $MLU PUSH PSW SAVE (PSW)
053.001 345      2589X      PUSH H SAVE FWA
053.002 053      2590X      DCX H ANTICIPATE INX H

```

```

053.003 043      2591X $MLU1 INX    H
053.004 176      2592X      MOV    A,M      (A)= CHARACTER
053.005 315 356 052 2593X      CALL  $MCU      MAP CHAR TO UPPER
053.010 167      2594X      MOV    M,A
053.011 247      2595X      ANA    A
053.012 302 003 053 2596X      JNZ   $MLU1      MORE TO GO
053.015 341      2597X      POP    H      RESTORE (HL)
053.016 361      2598X      POP    PSW      RESTORE (PSW)
053.017 311      2599X      RET
053.020      2600      XTEXT  TYPCH

```

```

2602X **      $TYPCH - TYPE SINGLE CHARACTER.
2603X *
2604X *      ENTRY  (RET) = CHARACTER
2605X *      EXIT   TO (RET)+1
2606X *      (A) = CHARACTER TYPED
2607X
2608X
053.020 343      2609X $TYPCH XTHL      (HL) = RETURN ADDRESS
053.021 176      2610X      MOV    A,M      (A) = CHARACTER
053.022 043      2611X      INX     H
053.023 343      2612X      XTHL      RESTORE ADVANCED EXIT ADDRESS
2613X
2614X **      $TYPC. - TYPE SINGLE CHARACTER.
2615X *
2616X *      ENTRY  (A) = CHARACTER
2617X *      EXIT   TO (RET)
2618X
053.024 377 002 2619X $TYPC. DB      SYSCALL, SCOUT
053.026 311      2620X      RET
053.027      2621      XTEXT  RTL

```

```

2623X **      $RTL - READ TEXT LINE.
2624X *
2625X *      $RTL READS A LINE FROM THE TERMINAL.
2626X *
2627X *      CHARACTER ARE ACCEPTED FROM THE TERMINAL, RUBOUT AND BACKSPACE
2628X *      CHARACTERS ARE PROCESSED. WHEN A CARRIAGE RETURN IS ENTERED,
2629X *      $RTL RETURNS.
2630X *
2631X *      ENTRY  (HL) = BUFFER FWA
2632X *      EXIT   'C' CLEAR IF OK
2633X *      DATA IN BUFFER
2634X *      (A) = TEXT LENGTH
2635X *      'C' SET IF CTL-D STRUCK
2636X *      USES   A,F
2637X
2638X
053.027 315 036 053 2639X $RTL. CALL  $RTL      $RTL IN UPPER CASE
053.032 330      2640X      RC      CTL-D

```

\$RTL

053.033 303 000 053 2641X JMP \$MLU MAP LINE TO UPPER CASE

2642X

053.036 2643X \$RTL EQU *

053.036 345 2644X PUSH H

SAVE FWA

053.037 315 367 052 2645X \$RTL1 CALL \$RCHAR

053.042 376 004 2646X CPI CTLD

053.044 312 071 053 2647X JE \$RTL2

CTL-D STRUCK

053.047 167 2648X MOV M,A

053.050 043 2649X INX H

053.051 376 012 2650X CPI NL

053.053 302 037 053 2651X JNE \$RTL1

053.056 053 2652X DCX H

053.057 066 000 2653X MVI M,0

053.061 043 2654X INX H

2655X

2656X

* ALL DONE. COMPUTE LENGTH

2657X

053.062 353 2658X

XCHG (DE) = LWA+1

053.063 343 2659X

XTHL (HL) = FWA

053.064 173 2660X MOV A,E

053.065 225 2661X SUB L

(A) = LENGTH

053.066 247 2662X ANA A

CLEAR CARRY

053.067 321 2663X POP D

RESTORE (DE)

053.070 311 2664X RET

2665X

2666X * CTL-D STRUCK

2667X

053.071 341 2668X \$RTL2 POP H

(HL) = FWA

053.072 067 2669X STC

053.073 311 2670X RET

053.074 2671 XTEXT UDDN

2672X

2673X

2674X

2675X

2676X

2677X

2678X

2679X

2680X

2681X

2682X

2683X

2684X

2685X

2686X

2687X

2688X

2689X

2690X

2691X

2692X

2693X

** \$UDDN - UNPACK DECIMAL DIGITS.

UDDN CONVERTS A 16 BIT VALUE INTO A SPECIFIED NUMBER OF
DECIMAL DIGITS. THE RESULT IS NULL FILLED TO THE LEFT.

ENTRY (B,C) = ADDRESS VALUE

(A) = DIGIT COUNT

(H,L) = MEMORY ADDRESS

EXIT (HL) = (HL) + (A)

USES ALL

053.074 2685X \$UDDN EQU *

053.074 315 072 030 2686X CALL \$DADA

053.077 345 2687X PUSH H

SAVE FINAL (H,L) VALUE

053.100 365 2688X

053.101 345 2689X UDDN1 PUSH PSW

053.102 021 012 000 2690X PUSH H

053.105 315 106 030 2691X LXI D,10

(H,L) = VALUE/10

053.110 104 2692X CALL \$DU66

053.110 104 2693X MOV B,H


```

053.111 115      2694X      MOV      C,L      (BC) = QUOTIENT
053.112 341      2695X      POP      H
053.113 076 060   2696X      MVI      A,0
053.115 203      2697X      ADD      E      ADD REMAINDER
053.116 053      2698X      DCX      H
053.117 167      2699X      MOV      M,A      STORE DIGIT
053.120 170      2700X      MOV      A,B
053.121 261      2701X      ORA      C
053.122 312 134 053 2702X      JZ      UDDN2      ALL ZEROS
053.125 361      2703X      POP      PSW
053.126 075      2704X      DCR      A
053.127 302 100 053 2705X      JNZ      UDDN1      IF MORE TO GO
                2706X
                2707X *      ALL DONE. EXIT
                2708X
053.132 341      2709X UDDN1.5 POP      H      RESTORE H
053.133 311      2710X      RET              RETURN
                2711X
                2712X *      DIGITS LEADING THIS ONE ARE ZERO. STORE NULLS INSTEAD.
                2713X
053.134 361      2714X UDDN2  POP      PSW
053.135 075      2715X UDDN3  DCR      A
053.136 312 132 053 2716X      JE      UDDN1.5      ALL DONE
053.141 053      2717X      DCX      H
053.142 066 000   2718X      MVI      M,0
053.144 303 135 053 2719X      JMP      UDDN3
053.147          2720      XTEXT    ZERO

```

```

                2722X **      $ZERO - ZERO MEMORY
                2723X *
                2724X *      $ZERO ZEROS A BLOCK OF MEMORY.
                2725X *
                2726X *      ENTRY    (HL) = ADDRESS
                2727X *              (B) = COUNT
                2728X *      EXIT      (A) = 0
                2729X *      USES     A,B,F,H,L
                2730X
                2731X
031.212          2732X $ZERO  EQU      31212A      IN H17 ROM
053.147          2733      XTEXT    '00'

```

```

                2735X **      $TOD - TYPE OCTAL DIGITS.
                2736X *
                2737X *      $TOD TYPES AN OCTAL BYTE AS 3 OCTAL DIGITS, ZERO FILL.
                2738X *
                2739X *      ENTRY    (A) = VALUE
                2740X *      EXIT      VALUE TYPES
                2741X *      USES     A,F
                2742X
                2743X

```

053.147	305	2744X	\$TOD	PUSH	B	
053.150	006 003	2745X		MVI	B,3	
053.152	247	2746X		ANA	A	CLEAR CARRY
		2747X				
053.153	027	2748X	TOD1	RAL		
053.154	027	2749X		RAL		
053.155	027	2750X		RAL		
053.156	365	2751X		PUSH	PSW	
053.157	346 007	2752X		ANI	7	
053.161	306 060	2753X		ADI	'0'	
053.163	315 024 053	2754X		CALL	\$TYPC.	TYPE CHARACTER
053.166	361	2755X		POP	PSW	
053.167	005	2756X		DCR	B	
053.170	302 153 053	2757X		JNZ	TOD1	IF MORE TO GO
053.173	301	2758X		POP	B	
053.174	311	2759X		RET		EXIT

```

2762 *** THIS SECTION DESCRIBES THE PATCH HISTORY TABLE. THE ACTUAL TABLE
2763 * WILL BE READ INTO THE BUFFER AREA *PHT*, DECLARED BELOW.
2764 *
2765 * THE PATCH HISTORY TABLE (PHT) IS USED TO KEEP TRACK OF WHAT PATCHES
2766 * HAVE BEEN ENTERED INTO A FILE. SYSTEM FILES ALL HAVE
2767 * BUILT-ON PHT'S, ANY FILE WITHOUT ONE IS THEREFORE NOT A SYSTEM
2768 * FILE.
2769 *
2770 * Note that the PHT stuck on the end of a binary file,
2771 * after the binary information. Thus, the PHT is not loaded into memory
2772 * during normal execution.
2773 *
2774 * The PHT occupies an entire sector, always the last sector in the file.
2775 * If the patches cause the file to be lengthened, then the
2776 * old PHT will be overlaid, and the new one added to the end of the
2777 * longer file.
2778
2779
2780
053.175 2781 PHTFORM DS 0 START OF 'STANDARD FORM' FOR PHT
000.000 2782 PHT.HDR EQU *-PHTFORM
053.175 374.001 2783 DB 3760,0010 PHT HEADER
053.177 120 110 124 2784 DB 'PHT/HSG' IDENTIFIES THE REAL THING!
000.011 2785 PHTHDRL EQU *-PHTFORM PHT HEADER LENGTH
000.011 2786 PHT.DAT EQU *-PHTFORM
053.206 122 127.055 2787 DB 'RW-JGL-RW' DATE OF LAST PATCH IN DD-MMM-YY
000.022 2788 PHT.CNT EQU *-PHTFORM
053.217 000 2789 DB 0 NUMBER OF PATCHES MADE
000.023 2790 PHT.HIS EQU *-PHTFORM
053.220 000.000.000 2791 DB 0,0,0,0,0,0,0,0 BIT-BY-BIT PATCH HISTORY (PATCH 0-63)
000.033 2792 PHTL EQU *-PHTFORM PHT ENTIRE LENGTH

```

DATA VALUES

16:26:16 16-MAY-80

```

2795 ** PATCH SERIES VALUES.
2796 *
2797 * THESE CELLS KEEP TRACK OF THE CURRENT PATCH ADDRESS
2798 * AND ASSOCIATED VALUES
2799
053.230 000 000 2800 SKEW DW 0 CORRECTION FACTOR TO MAP PROGRAM ADDRESS
053.232 000 2801 SDISP DB 0 SECTORS TO DISPLACE INTO FILE TO FIND PROGRAM
2802 * INTO FILE BYTE NUMBER
053.233 000 000 2803 CPA DW 0 CURRENT PATCH ADDRESS
053.235 000 2804 SIVB DB 0 CURRENT SECTOR IN VIEW BUFFER
053.236 000 2805 SIVBV DB 0 <<0 IF SIVB VALID
053.237 000 2806 SIVBA DB 0 <<0 IF SECTOR-IN-VIEWBFR ALTERED

2808 ** MISCELLANEOUS WORK AREAS
2809
053.240 123 131 060 2810 DEFALT DB 'SYOABS' DEFAULT FOR FILE NAMES
053.246 000 060 2811 PLPTR DW PATLIST POINTER TO NEXT FREE BYTE IN PATLIST
053.250 000 000 2812 PLMAX DW 0 MAX. SIZE OF PATCH LIST
053.252 000 2813 FILTYP DB 0 FT.ABS OR FT.PIC

2815 ** PATCH VALUES (MEANINGFUL ONLY FOR SYSTEM PATCHES)
2816
053.253 2817 PATPRQ DS 5 PATCH PREREQUISITE LIST
000.005 2818 PATPRQL EQU *-PATPRQ NUMBER OF BYTES IN LIST
053.260 2819 PATCHID DS 1 PATCH ID VALUE

2821 ** PHT FLAGS
2822
053.261 000 2823 PHIST DB 0 <<0 IF PHT PRESENT
053.262 000 000 2824 PGMFWA DW 0 FWA OF USER PROGRAM
053.264 000 000 2825 PGMLWA DW 0 LWA OF USER PROGRAM
053.266 000 000 2826 FILSIZ DW 0 SIZE OF FILE
2827
000.001 2828 IF .SYS.
2829 CHECKC DB 0 <<0 IF /CHECK SWITCH SELECTED
2830 PHTSWI DB 0 <<0 IF /PHT SWITCH SELECTED
2831 PHT DS 256 FILL PHT BUFFER WITH COPYRIGHT MESSAGE
2832 ENDIF
2833
053.270 2834 .PAT1. DS 32 PATCH AREA WHICH DOESNT REQUIRE EXPANSION OF FILE
053.330 014 2835 DB FF
053.331 266 271 337 2836 DB 377Q-'I',377Q-'F',377Q-' ',377Q-'U',377Q-' ',377Q-'C',377Q-'N'
053.340 337 255 273 2837 DB 377Q-' ',377Q-'R',377Q-'D',377Q-' ',377Q-'T',377Q-'H',377Q-'S'
053.347 337 252 337 2838 DB 377Q-' ',377Q-'U',377Q-' ',377Q-'C',377Q-'N',377Q-' ',377Q-'G'
053.356 253 337 276 2839 DB 377Q-'T',377Q-' ',377Q-'A',377Q-' ',377Q-'B',377Q-'D',377Q-' '
053.365 265 275 336 2840 DB 377Q-'J',377Q-'B',377Q-'!',FF
2841
2842
2843
2844 ** BUFFERS

```

```

2845
053.371 2846 MEML EQU * END OF LOAD IMAGE
2847
053.371 2848 .PAT. DS 64 PATCH AREA
2849
054.071 2850 LINE DS 80
054.211 2851 DS **255/256*256-* PUT VIEWBFR ON EVEN BOUNDARY
055.000 2852 VIEWBFR DS 256 BUFFER TO PROVIDE 'OLD VALUE' DISPLAY
000.000 2853 ERRNZ #VIEWBFR MUST BE ON EVEN BOUNDARY
056.000 2854 BUFFER DS 256 WORKING BUFFER
000.001 2855 IF .SYS.
2856 ELSE
057.000 2857 PHT DS 256 PUT BUFFER OFF END, NORMALLY
2858 ENDDIF PATCH HISTORY TABLE

```

```

2860 ** PATCH LIST.
2861 *
2862
060.000 2863 PATLIST EQU * PATCH LIST STARTS HERE
2864
060.000 2865 END
ASSEMBLY COMPLETE
2865 STATEMENTS
0 ERRORS DETECTED
11946 BYTES FREE

```

```

XREF V1.1

```

PAGE 62

[illegible]

```
..XREF V1.1
```

PAGE 63

\$WCHAR	052375	2540L																	
\$ZERO	031212	1609	2732E																
.CHFLG	000060	248L																	
.CLEAR	000055	245L	650																
.CLEARA	000056	246L	648																
.CLOSE	000046	238L	592	817															
.CLRCO	000007	222L																	
.CONSL	000006	221L	2376																
.CTLC	000041	233L	584	688															
.DECODE	000053	243L	731																
.DELET	000050	240L																	
.DISMT	000061	249L																	
.DMNMS	000203	260L																	
.DMOUN	000201	258L																	
.ERROR	000057	247L	611																
.EXIT	000000	215L	550	695															
.LINK	000040	232L																	
.LOADD	000062	250L																	
.LOADD	000010	223L																	
.MONMS	000202	259L																	
.MOUNT	000200	257L																	
.NAME	000054	244L																	
.OPENC	000045	237L																	
.OPENR	000042	234L	741																
.OPENU	000044	236L	821																
.OPENW	000043	235L																	
.PAT.	053371	2848L																	
.PAT1.	053270	2834L																	
.POSIT	000047	239L	1642																
.PRINT	000003	218L	1012																
.READ	000004	219L	751	801	1601	1851													
.RENAM	000051	241L																	
.RESET	000204	261L																	
.SCIN	000001	216L	2536																
.SCOUT	000002	217L	2540	2619															
.SETTP	000052	242L	681																
.SYS.	000091	3E	553	842	857	880	1226	1338	1674	1755	2828	2855							
.SYSRES	000012	225L																	
.VERS	000011	224L	643																
.WRITE	000005	220L	1531	1540	1884	1936													
ABL	047170	1183	1185	1187	1365L														
ABL1	047273	1378	1388L																
ABS.COD	000010	513L	526	778															
ABS.ENT	000006	511L																	
ABS.ID	000000	507L																	
ABS.LDA	000002	509L	775																
ABS.LEN	000004	510L	781	1877															
AEE	047324	802	1407L	1521	1596	1602													
AIO.CGN	041047	443L																	
AIO.CHA	041116	458L																	
AIO.CNT	041111	454L																	
AIO.CSI	041050	444L																	
AIO.DDA	041041	439E																	
AIO.DES	041055	448L																	
AIO.DEV	041057	449L																	
AIO.DIR	041062	452L																	
AIO.DTA	041053	447L																	
AIO.EOF	041113	456L																	

```
..XREF V1.1
```

PAGE 64

[illegible]

PATCH - PATCH SYSTEM AND USER FILES
CROSS REFERENCE TABLE

XREF V1.1
PAGE 65

DCSA	044160	846	853L		
DEFAULT	053240	730	738	819	2810L
DEV.DDA	000004	174L			
DEV.DVG	000016	186L			
DEV.DVL	000014	185L			
DEV.FLG	000006	175L			
DEV.JMP	000003	173L			
DEV.MNU	000011	182L			
DEV.MUM	000010	181L			
DEV.NAM	000000	165L			
DEV.RES	000002	169L			
DEV.SFG	000007	180L			
DEV.UNT	000012	183L			
DEVELEN	000017	188E			
DF.CLR	000376	106E			
DF.EMP	000377	105E			
DIR.ALD	000025	121L			
DIR.CLU	000015	114L			
DIR.CRD	000023	120L			
DIR.EXT	000010	109L			
DIR.FGN	000020	117L			
DIR.FLG	000016	115L			
DIR.LGN	000021	118L			
DIR.LSI	000022	119L			
DIR.NAM	000000	108L			
DIR.PRO	000013	110L			
DIR.VER	000014	111L			
DIRELEN	000027	123E	154	452	
DIRIDL	000015	112E			
DOB	047373	1075	1078	1174	1483L
DOB1	047376	1484L	1499		
DR.IM	000001	170E			
DR.PR	000002	171E			
DT.CR	000002	177E			
DT.CW	000004	178E	735		
DT.DD	000001	176E	735		
DV.EL	000000	166E			
DV.NU	000001	167E			
EC.CNA	000004	58L			
EC.BDA	000027	77L			
EC.DIF	000017	69L			
EC.DIW	000035	83L			
EC.DNI	000045	91L			
EC.DNR	000046	92L			
EC.DNS	000005	59L	736		
EC.DSC	000047	93L			
EC.EOF	000001	55L	1409		
EC.EOM	000002	56L			
EC.FAD	000031	79L			
EC.FAP	000026	76L			
EC.FL	000030	78L			
EC.FNF	000014	66L			
EC.FNO	000011	63L			
EC.FNR	000034	82L			
EC.FOD	000043	89L			
EC.FUC	000013	65L			
EC.ICN	000016	68L			
EC.IDN	000006	60L			

```

XREF V1.1

```

PAGE 66

[illegible]

```

XREF V1.1
PAGE 67

```

[illegible]

CROSS REFERENCE TABLE

		558	575	588	656	809	815	1282	2823L
PHIST	053261	558	575	588	656	809	815	1282	2823L
PHT	057000	799	805	975	1904	1934	2857L		
PHT.CNT	000022	2788E							
PHT.DAT	000011	2786E							
PHT.HDR	000000	804	2782E						
PHT.HIS	000023	975	1904	2790E					
PHTFORM	053175	804	2781L	2782	2785	2786	2788	2790	2792
PHTHDL	000011	806	2785E						
PHTL	000033	2792E							
PIC.COD	000006	500L							
PIC.ID	000000	495L							
PIC.LEN	000002	497L	770	1872					
PIC.PTR	000004	498L							
PLMAX	053250	680	1370	2812L					
PLPTR	053244	658	1217	1292	1365	1368	2811L		
PPF	050235	747	795	1520	1595	1635L	1846	1879	1931
PRS	042372	543	639E						
PRSERR	043132	693L							
PRSERR1	043130	444	646	691L					
QUOTE	000047	474E							
RES	050253	918	954	1244	1673E				
RES1	050276	1692L	1721						
RES2	050340	1717	1719L						
RESA	050364	1688	1718	1736L					
RESAE	051024	1716	1737L						
RESTART	042200	540E	545	603	621	1032	1390		
RQMBOQT	030000	271E							
RUBOUT	000177	470E							
S.BAUD	040344	362L							
S.BOOTF	041034	417L							
S.CAADR	040333	345L							
S.CACC	041006	401L							
S.CCTAB	040335	346L							
S.CDB	040343	359L							
S.CFWA	040352	369L	1461						
S.CODE	041007	402L							
S.CONFL	040332	343L							
S.CONTY	040327	330L							
S.CONWI	040331	336L							
S.CSLMD	040326	319L	329	332	335	342			
S.CUSUR	040330	333L							
S.DATC	040310	301L							
S.DATE	040277	300L							
S.DCS	041033	415L							
S.DDATA	040366	380L							
S.DDGRF	040364	377L							
S.DDLDA	040360	375L							
S.DDLEN	040362	376L							
S.DDOPC	040370	381L							
S.DFWA	040354	370L							
S.DIREA	041016	409L							
S.DLINK	040346	367L							
S.FASER	041013	408L							
S.FCI	041021	410L							
S.GRT0	024000	267E							
S.GRT1	025000	268E							
S.GRT2	026000	269E							
S.GUP	041027	412L							

XREF V1.1
 PAGE 69

[illegible]

WPH1 051166 1914L 1916

23384 BYTES FREE