

3 *** H8ASM - H8 ASSEMBLER.
4 *
5 * J. G. LETWIN, 09/76, FOR *WINTEK* CORP.
6 *
7 * COPYRIGHT 09/76, *WINTEK* CORPORATION,
8 * LAFAYETTE, IND.

10 *** H8ASM - H8 RESIDENT ASSEMBLER.
11 *
12 * SOFTWARE ISSUE NUMBER:
13 *
14 * 0104.02.00.
15 * 79/12 --.05.00

/78.10.6C/

17 **** ASSEMBLY CONSTANTS.

000.002 19 VER EQU 2 VERSION NUMBER
000.000 20 LEV EQU 0 VERSION LEVEL
000.101 21 MOD EQU 'A' MODIFICATION LEVEL

23 ** ERROR FLAGS

000.001 25 ERR.U EQU 0010 *U* - UNDEFINED
000.002 26 ERR.R EQU 0020 *R* - ILLEGAL REGISTER SPECIFICATION
000.004 27 ERR.D EQU 0040 *D* - DOUBLY DEFINED SYMBOL
000.010 28 ERR.A EQU 0100 *A* - EXPRESSION ERROR
000.020 29 ERR.V EQU 0200 *V* - VALUE TOO LARGE FOR FIELD
000.040 30 ERR.F EQU 0400 *F* - ILLEGAL STATEMENT FORMAT
000.100 31 ERR.O EQU 1000 *O* - OPCODE ERROR
000.200 32 ERR.P EQU 2000 *P* - PROGRAMMER FLAGGED ERROR

34 ** LIST OPTIONS.

000.001 36 LST.L EQU 0010 MASTER LIST FLAG
000.002 37 LST.I EQU 0020 LIST IF-SKIPPED LINES
000.004 38 LST.C EQU 0040 LIST INCLUDED CODE
000.200 39 LST.G EQU 2000 LIST ALL GENERATED BYTES

41 ** SYMBOL DEFINITION TYPES.

000.000 43 ST.UND EQU 0 UNDEFINED
000.001 44 ST.LAB EQU 1 LABEL
000.002 45 ST.EQU EQU 2 DEFINED VIA *EQU*
000.003 46 ST.SET EQU 3 DEFINED VIA *SET*
000.100 47 ST.REL EQU 1000 RELOCATABLE
000.200 48 ST.DBL EQU 2000 DOUBLY DEFINED

51 ** CHARACTER TYPES

000.200 53 CT.ALPH EQU 10000000B ALPHA CHARACTER

HEADING:

15:17:45 16-MAY-80

```

54
55
56 **      CHANNEL NUMBERS
57
000.000 58 CN.BIN EQU 0      BINARY FILE
000.001 59 CN.LST EQU 1      LISTING FILE
000.002 60 CN.SOU EQU 2      SOURCE INPUT FILE
000.003 61 CN.XTX EQU 3      XTEXT
62
63

```

```

64 **      MACHINE INSTRUCTIONS
65
000.303 66 MI.JMP EQU 303Q    JMP
000.072 67 MI.LDA EQU 072Q    LDA
000.311 68 MI.RET EQU 311Q    RET
000.043 69 MI.INXH EQU 43Q     INX H INSTRUCTIIN
000.325 70 MI.PSHD EQU 325Q    PUSH D
71
000.000 72      XTEXT  ASCII

```

```

74X **      ASCII CHARACTER EQUIVALENCES.
75X
000.015 76X CR EQU 13        CARRIAGE RETURN
000.012 77X LF EQU 10        LINE FEED
000.200 78X NULL EQU 200Q     PAD CHARACTER
000.000 79X NUL2 EQU 0
000.007 80X BELL EQU 7        BELL CHARACTER
000.177 81X RUBOUT EQU 177Q
000.010 82X BKSP EQU 10Q      CTL-H
000.026 83X C.SYN EQU 26Q     SYNC
000.002 84X C.STX EQU 2       STX
000.047 85X QUOTE EQU 47Q
000.011 86X TAB EQU 11Q
000.033 87X ESC EQU 33Q
000.012 88X NL EQU 12Q        NEW LINE (HDOS SYSTEMS)
000.212 89X ENL EQU NL+200Q    NL + END-OF-LINE-FLAG
000.014 90X FF EQU 14Q        FORM FEED
000.001 91X CTLA EQU 01Q      CTL-A
000.002 92X CTLB EQU 02Q      CTL-B
000.003 93X CTLC EQU 03Q      CTL-C
000.004 94X CTLD EQU 04Q      CTL-D
000.017 95X CTLO EQU 17Q      CTL-O
000.020 96X CTLP EQU 20Q      CTL-P
000.021 97X CTLQ EQU 21Q      CTL-Q
000.023 98X CTLS EQU 23Q      CTL-S
000.032 99X CTLZ EQU 32Q      CTL-Z
000.000 100 XTEXT DIRDEF

```

HEADING.

DIR

15:17:59 16-MAY-80

102X ** DIRECTORY ENTRY FORMAT.

000.000	103X				
	104X	ORG	0		
	105X				
	106X				
000.377	107X	DF.EMP	EQU	3770	FLAGS ENTRY EMPTY
000.376	108X	DF.CLR	EQU	3760	FLAGS ENTRY EMPTY; REST OF DIR ALSO CLEAR
	109X				
000.000	110X	DIR.NAM	DS	8	NAME
000.010	111X	DIR.EXT	DS	3	EXTENSION
000.013	112X	DIR.PRO	DS	1	PROJECT
000.014	113X	DIR.VER	DS	1	VERSION
000.015	114X	DIRIDL	EQU	*	FILE IDENTIFICATION LENGTH
	115X				
000.015	116X	DIR.CLU	DS	1	CLUSTER FACTOR
000.016	117X	DIR.FLG	DS	1	FLAGS
000.017	118X		DS	1	RESERVED
000.020	119X	DIR.FGN	DS	1	FIRST GROUP NUMBER
000.021	120X	DIR.LGN	DS	1	LAST GROUP NUMBER
000.022	121X	DIR.LSI	DS	1	LAST SECTOR INDEX (IN LAST GROUP)
000.023	122X	DIR.CRD	DS	2	CREATION DATE
000.025	123X	DIR.ALD	DS	2	LAST ALTERATION DATE
	124X				
000.027	125X	DIRELEN	EQU	*	DIRECTORY ENTRY LENGTH
000.027	126	XTEXT		DEVDEF	

128X ** DEVICE TABLE ENTRIES.

000.000	129X				
	130X	ORG	0		
	131X				
000.000	132X	DEV.NAM	DS	2	DEVICE NAME
000.000	133X	DEV.EL	EQU	00000000B	END OF DEVICE LIST FLAG
000.001	134X	DEV.NU	EQU	00000001B	DEVICE ENTRY NOT IN USE
	135X				
000.002	136X	DEV.RES	DS	1	DRIVER RESIDENCE CODE
000.001	137X	DR.IM	EQU	00000001B	DRIVER IN MEMORY
000.002	138X	DR.FR	EQU	00000010B	DRIVER PERMANENTLY RESIDENT
	139X				
000.003	140X	DEV.JMP	DS	1	JMP TO PROCESSOR
000.004	141X	DEV.DDA	DS	2	DRIVER ADDRESS
000.006	142X	DEV.FLG	DS	1	FLAG BYTE
000.001	143X	DT.DD	EQU	00000001B	DIRECTORY DEVICE
000.002	144X	DT.CR	EQU	00000010B	CAPABLE OF READ OPERATION
000.004	145X	DT.CW	EQU	00000100B	CAPABLE OF WRITE OPERATION
	146X				
000.007	147X	DEV.SPG	DS	1	SECTORS PER GROUP THIS DEVICE
000.010	148X	DEV.MUM	DS	1	MOUNTED UNIT MASK
000.011	149X	DEV.MNU	DS	1	MAXIMUM NUMBER OF UNITS
000.012	150X	DEV.UNT	DS	2	ADDRESS OF UNIT SPECIFIC DATA TABLE
	151X				
000.014	152X	DEV.DVL	DS	2	DRIVER BYTE LENGTH
000.016	153X	DEV.DVG	DS	1	DRIVER ROUTINE GROUP ADDRESS
	154X				

000.017 155X DEVELEN EQU * DEVICE TABLE ENTRY LENGTH

157X ** UNIT SPECIFIC DEVICE DATA TABLE ENTRIES

158X
000.000 159X ORG 0
160X
000.000 161X UNT.FLG DS 1 UNIT SPECIFIC *DEV.FLG*
000.001 162X UNT.GRT DS 2 ADDRESS OF GROUP RESERVATION TABLE (IF DT.DD)
000.003 163X UNT.GTS DS 2 GRT SECTOR NUMBER
000.005 164X UNT.DIS DS 2 DIRECTORY FIRST SECTOR NUMBER
165X
000.007 166X UNT.SIZ EQU * SIZE OF UNIT SPECIFIC DATA TABLE PER UNIT
000.007 167 XTEXT HOSDEF

169X ** HOSDEF - DEFINE HOS PARAMETER.

170X *
171X
172X
000.026 173X VERS EQU 1*16+6 VERSION 1.6
174X
000.377 175X SYSCALL EQU 3770 SYSCALL INSTRUCTION
176X
177X

000.000 178X ORG 0

179X
180X * RESIDENT FUNCTIONS

181X
000.000 182X .EXIT DS 1 EXIT (MUST BE FIRST)
000.001 183X .SCIN DS 1 SCIN
000.002 184X .SCOUT DS 1 SCOUT
000.003 185X .PRINT DS 1 PRINT
000.004 186X .READ DS 1 READ
000.005 187X .WRITE DS 1 WRITE
000.006 188X .CONSL DS 1 SET/CLEAR CONSOLE OPTIONS
000.007 189X .CLRCD DS 1 CLEAR CONSOLE BUFFER
000.010 190X .LOADO DS 1 LOAD AN OVERLAY
000.011 191X .VERS DS 1 RETURN HDOS VERSION NUMBER
000.012 192X .SYSRES DS 1 PRECEDING FUNCTIONS ARE RESIDENT
193X

194X
195X * *HDOSOVLO.SYS* FUNCTIONS

196X
000.040 197X ORG 40A
198X
000.040 199X .LINK DS 1 LINK (MUST BE FIRST)
000.041 200X .CTLCD DS 1 CTL-C
000.042 201X .OPENR DS 1 OPENR
000.043 202X .OPENW DS 1 OPENW
000.044 203X .OPENU DS 1 OPENU
000.045 204X .OPENC DS 1 OPENC
000.046 205X .CLOSE DS 1 CLOSE

HEADING.

HOSDEF

15:18:06 16-MAY-80

000.047	206X	.POSIT	DS	1	POSITION
000.050	207X	.DELET	DS	1	DELETE
000.051	208X	.RENAM	DS	1	RENAME
000.052	209X	.SETTP	DS	1	SETTOP
000.053	210X	.DECODE	DS	1	NAME DECODE
000.054	211X	.NAME	DS	1	GET FILE NAME FROM CHANNEL
000.055	212X	.CLEAR	DS	1	CLEAR CHAN
000.056	213X	.CLEARA	DS	1	CLEAR ALL CHANS
000.057	214X	.ERROR	DS	1	LOOKUP ERROR
000.060	215X	.CHFLG	DS	1	CHANGE FLAGS
000.061	216X	.DISMT	DS	1	FLAG SYSTEM DISK DISMOUNTED
000.062	217X	.LOADD	DS	1	LOAD DEVICE DRIVER
	218X				
	219X				
	220X	*			*HDOSVOL1.SYS* FUNCTIONS
	221X				
000.200	222X		ORG	2000	
	223X				
000.200	224X	.MOUNT	DS	1	MOUNT (MUST BE FIRST)
000.201	225X	.DMOUN	DS	1	DISMOUNT
000.202	226X	.MONMS	DS	1	MOUNT/NO MESSAGE
000.203	227X	.DMNMS	DS	1	DISMOUNT/NO MESSAGE
000.204	228X	.RESET	DS	1	RESET = DISMOUNT/MOUNT OF UNIT
000.205	229	XTEXT		H0SEQU	

	231X	**			HDOS SYSTEM EQUIVALENCES.
	232X	*			
	233X				
024.000	234X	S.GRT0	EQU	24000A	SYSTEM AREA FOR GRT0
025.000	235X	S.GRT1	EQU	25000A	SYSTEM AREA FOR GRT1
026.000	236X	S.GRT2	EQU	26000A	SYSTEM AREA FOR GRT2
	237X				
030.000	238X	ROMBOOT	EQU	30000A	ROM BOOT ENTRY
	239X				
040.100	240X		ORG	40100A	FREE SPACE FROM PAM-8
	241X				
040.100	242X		DS	8	JUMP TO SYSTEM EXIT
040.110	243X	D.CON	DS	16	DISK CONSTANTS
040.130	244X	SYDD	EQU	*	SYSTEM DISK ENTRY POINT
040.130	245X	D.VEC	DS	24*3	SYSTEM ROM ENTRY VECTORS
040.240	246X	D.RAM	DS	31	SYSTEM ROM WORK AREA
040.277	247X	S.VAL	DS	36	SYSTEM VALUES
040.343	248X	S.INT	DS	115	SYSTEM INTERNAL WORK AREAS
041.126	249X		DS	16	
041.146	250X	S.SOVR	DS	2	STACK OVERFLOW WARNING
041.150	251X		DS	42200A-*	SYSTEM STACK
001.032	252X	STACKL	EQU	*-S.SOVR	STACK SIZE
	253X				
042.200	254X	STACK	EQU	*	LWA+1 SYSTEM STACK
042.200	255X	USERFWA	EQU	*	USER FWA
042.200	256	XTEXT		ESVAL	

```
258X **      S.VAL - SYSTEM VALUE DEFINITIONS.
259X *
260X *      THESE VALUES ARE SET AND MAINTAINED BY THE SYSTEM.
261X *
262X *      THE DECK HOSEQU MUST BE MODIFIED WHEN THIS IS MODIFIED.
263X
264X
040.277      265X      ORG      S.VAL
266X
040.277      267X S.DATE DS      9      SYSTEM DATE (IN ASCII)
040.310      268X S.DATC DS      2      CODED DATE
040.312      269X S.TIME DS      4      TIME FROM MIDNIGHT (IN TICS)
040.316      270X S.HIMEM DS     2      HARDWARE HIGH MEMORY ADDRESS+1
271X
040.320      272X S.SYSM DS      2      FWA RESIDENT SYSTEM
273X
040.322      274X S.USRM DS      2      LWA USER MEMORY
275X
040.324      276X S.OMAX DS      2      MAX OVERLAY SIZE FOR SYSTEM
277X
278X
279X **      THE FOLLOWING FIVE CELLS SHOULD BE MODIFIED/READ ONLY VIA THE .CONSL SYSCALL
280X
000.200      281X CSL.ECH EQU     10000000B  SUPPRESS ECHO
000.002      282X CSL.WRP EQU     00000010B  WRAP LINES AT WIDTH
000.001      283X CSL.CHR EQU     00000001B  OPERATE IN CHARACTER MODE
284X
000.000      285X I.CSLMD EQU      0      S.CSLMD IS FIRST BYTE
040.326      286X S.CSLMD DS      1      CONSOLE MODE
287X
000.200      288X CTP.BKS EQU     10000000B  TERMINAL PROCESSES BACKSPACES
000.040      289X CTP.MLI EQU     00100000B  MAP LOWER CASE TO UPPER ON INPUT
000.020      290X CTP.MLO EQU     00010000B  MAP LOWER CASE TO UPPER ON OUTPUT
000.010      291X CTP.2SB EQU     00001000B  TERMINAL NEEDS TWO STOP BITS
000.002      292X CTP.BKM EQU     00000010B  MAP BKSP (UPON INPUT) TO RUBOUT
000.001      293X CTP.TAB EQU     00000001B  TERMINAL SUPPORTS TAB CHARACTERS
294X
000.001      295X I.CONTY EQU      1      S.CONTY IS 2ND BYTE
000.000      296X      ERRNZ *S.CSLMD-I.CONTY
040.327      297X S.CONTY DS      1      CONSOLE TYPE FLAGS
000.002      298X I.CUSOR EQU      2      S.CUSOR IS 3RD BYTE
000.000      299X      ERRNZ *S.CSLMD-I.CUSOR
040.330      300X S.CUSOR DS      1      CURRENT CURSOR POSITION
000.003      301X I.CONWI EQU      3      S.CONWI IS 4TH BYTE
000.000      302X      ERRNZ *S.CSLMD-I.CONWI
040.331      303X S.CONWI DS      1      CONSOLE WIDTH
304X
000.001      305X CQ.FLG EQU     00000001B  CTL-0 FLAG
000.200      306X CS.FLG EQU     10000000B  CTL-S FLAG
307X
000.004      308X I.CONFL EQU      4      S.CONFL IS 5TH BYTE
000.000      309X      ERRNZ *S.CSLMD-I.CONFL
040.332      310X S.CONFL DS      1      CONSOLE FLAGS
311X
040.333      312X S.CAADR DS      2      ADDRESS FOR ABORT PROCESSING (>256 IF VALID)
040.335      313X S.CCTAB DS      6      ADDR FOR CTL-A, CTL-B, CTL-C PROCESSING
```

040.343

314

XTEXT ESINT

040.343

040.343

000.000

000.001

040.344

040.346

040.350

040.352

040.354

040.356

040.360

040.362

040.364

040.365

040.366

040.370

000.001

000.002

000.014

000.200

040.371

040.372

040.374

040.376

041.000

041.002

041.004

316X ** S.INT - SYSTEM INTERNAL WORKAREA DEFINITIONS.

317X *

318X *

319X *

320X

321X

322X

323X

324X **

325X

326X

327X

328X

329X

330X *

331X

332X **

333X

334X

335X

336X

337X

338X

339X

340X **

341X

342X

343X

344X

345X

346X

347X

348X

349X

350X **

351X

352X

353X

354X

355X

356X

357X

358X

359X

360X

361X

362X

363X

364X

365X

366X *

ORG S.INT

CONSOLE STATUS FLAGS

CONSOLE DESCRIPTOR BYTE

DS 1 EQU 00000000B

DS 2 EQU 00000001B

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

DS 2

THESE CELLS ARE REFERENCED BY OVERLAYS AND MAIN CODE, AND
MUST THEREFORE RESIDE IN FIXED LOW MEMORY.

=0 IF H8-5, =1 IF H8-4
[0-14] H8-4 BAUD RATE, =0 IF H8-5
[15] =1 IF BAUD RATE => 2 STOP BITS

TABLE ADDRESS WORDS

ADDRESS OF DATA IN HDOS CODE

FWA OVERLAY TABLE

FWA CHANNEL TABLE

FWA DEVICE TABLE

FWA RESIDENT HDOS CODE

DEVICE DRIVER DELAYED LOAD FLAGS

DRIVER LOAD ADDRESS (HIGH BYTE=0 IF NO LOAD PENDING)

CODE LENGTH IN BYTES

GROUP NUMBER FOR DRIVER

HOLD PLACE

SECTOR NUMBER FOR DRIVER (* OBSOLETE ! *)

DEVICE'S ADDRESS IN DEVLST +DEV.RES

OPEN OPCODE PENDING

OVERLAY MANAGEMENT FLAGS

IN MEMORY

PERMANENTLY RESIDENT

OVERLAY NUMBER MASK

USER CODE SWAPPED FOR OVERLAY

OVERLAY FLAG

FWA SWAPPED USER CODE

LENGTH SWAPPED USER CODE

SIZE OF OVERLAY CODE

ENTRY POINT OF OVERLAY CODE

SWAP AREA SECTOR NUMBER

OVERLAY SECTOR NUMBER

SYSCALL PROCESSING WORK AREAS

HEADING.

ESINT

15:18:24 16-MAY-80

	367X				
041.006	368X	S.CACC	DS	1	(ACC) UPON SYSCALL
041.007	369X	S.CODE	DS	1	SYSCALL INDEX IN PROGRESS
	370X				
	371X	*			JUMPS TO ROUTINES IN RESIDENT HDOS CODE
	372X				
041.010	373X	S.JUMPS	DS	0	START OF DUMP VECTORS
041.010	374X	S.SDD	DS	3	JUMP TO STAND-IN DEVICE DRIVER
041.013	375X	S.FASER	DS	3	JUMP TO FATSEERR (FATAL SYSTEM ERROR)
041.016	376X	S.DIREA	DS	3	JUMP TO DIREAD (DISK FILE READ)
041.021	377X	S.FCI	DS	3	JUMP TO FCI (FETCH CHANNEL INFO)
041.024	378X	S.SCI	DS	3	JUMP TO SCI (STORE CHANNEL INFO)
041.027	379X	S.GUP	DS	3	JUMP TO GUP (GET UNIT POINTER)
	380X				
041.032	381X	S.MOUNT	DS	1	0 IF THE SYSTEM DISK IS MOUNTED
041.033	382X	S.DCS	DS	1	DEFAULT CLUSTER SIZE-1
	383X				
041.034	384X	S.BOOTF	DS	1	BOOT FLAGS
000.001	385X	BOOT.P	EQU	00000001B	EXECUTE PROLOGUE UPON BOOTUP
	386X				
	387X	*			STACK VALUE SAVED FOR OVERLAY SYSCALLS
	388X				
041.035	389X	S.OVSTK	DS	2	VALUE OF SP UPON SYSCALLS USING OVERLAY
	390X				
041.037	391X		DS	1	RESERVED
	393X	**			ACTIVE I/O AREA.
	394X	*			
	395X	*			THE AIO.XXX AREA CONTAINS INFORMATION ABOUT THE I/O OPERATION
	396X	*			CURRENTLY BEING PERFORMED. THE INFORMATION IS OBTAINED FROM
	397X	*			THE CHANNEL TABLE, AND WILL BE RESTORED THERE WHEN DONE.
	398X	*			
	399X	*			NORMALLY, THE AIO.XXX INFORMATION WOULD BE OBTAINED DIRECTLY
	400X	*			FROM VARIOUS SYSTEM TABLES VIA POINTER REGISTERS. SINCE THE
	401X	*			8080 HAS NO GOOD INDEXED ADDRESSING, THE DATA IS MANUALLY
	402X	*			COPIED INTO THE AIO.XXX CELLS BEFORE PROCESSING, AND
	403X	*			BACKDATED AFTER PROCESSING.
	404X				
041.040	405X	AIO.VEC	DS	3	JUMP INSTRUCTION
041.041	406X	AIO.DDA	EQU	*-2	DEVICE DRIVER ADDRESS
041.043	407X	AIO.FLG	DS	1	FLAG BYTE
041.044	408X	AIO.GRT	DS	2	ADDRESS OF GROUP RESERV TABLE
041.046	409X	AIO.SPG	DS	1	SECTORS PER GROUP
041.047	410X	AIO.CGN	DS	1	CURRENT GROUP NUMBER
041.050	411X	AIO.CSI	DS	1	CURRENT SECTOR INDEX
041.051	412X	AIO.LGN	DS	1	LAST GROUP NUMBER
041.052	413X	AIO.LSI	DS	1	LAST SECTOR INDEX
041.053	414X	AIO.DTA	DS	2	DEVICE TABLE ADDRESS
041.055	415X	AIO.DES	DS	2	DIRECTORY SECTOR
041.057	416X	AIO.DEV	DS	2	DEVICE CODE
041.061	417X	AIO.UNI	DS	1	UNIT NUMBER (0-9)
	418X				
041.062	419X	AIO.DIR	DS	DIRELEN	DIRECTORY ENTRY

	420X				
041.111	421X	AIO.CNT	DS	1	SECTOR COUNT
041.112	422X	AIO.EOM	DS	1	END OF MEDIA FLAG
041.113	423X	AIO.EOF	DS	1	END OF FILE FLAG
041.114	424X	AIO.TFF	DS	2	TEMP FILE POINTERS
041.116	425X	AIO.CHA	DS	2	ADDRESS OF CHANNEL BLOCK (IOC.DDA)

041.120	427X	S.SCR	DS	2	SYSTEM SCRATCH AREA ADDRESS
041.122	428	XTEXT	FILDEF		

430X ** FILDEF - FILE TYPE DEFINITIONS.

	431X	*			
	432X	*	DB	377Q,FT.XXX	
	433X				
	434X				
000.000	435X	FT.ABS	EQU	0	ABSOLUTE BINARY
000.001	436X	FT.PIC	EQU	1	POSITION INDEPENDANT CODE
000.002	437X	FT.REL	EQU	2	RELOCATABLE CODE
000.003	438X	FT.BAC	EQU	3	COMPILED BASIC CODE
041.122	439	XTEXT	ABSDEF		

441X ** ABS FORMAT EQUIVALENCES.

	442X				
000.000	443X		ORG	0	
	444X				
000.000	445X	ABS.ID	DS	1	377Q = BINARY FILE FLAG
000.001	446X		DS	1	FILE TYPE (FT.ABS)
000.002	447X	ABS.LDA	DS	2	LOAD ADDRESS
000.004	448X	ABS.LEN	DS	2	LENGTH OF ENTIRE RECORD
000.006	449X	ABS.ENT	DS	2	ENTRY POINT
	450X				
000.010	451X	ABS.COD	DS	0	CODE STARTS HERE
000.010	452	XTEXT	PICDEF		

454X ** PIC FORMAT EQUIVALENCES.

	455X				
000.000	456X		ORG	0	
	457X				
000.000	458X	PIC.ID	DS	1	377Q = BINARY FILE FLAG
000.001	459X		DS	1	FILE TYPE (FT.PIC)
000.002	460X	PIC.LEN	DS	2	LENGTH OF ENTIRE RECORD
000.004	461X	PIC.PTR	DS	2	INDEX OF START OF PIC TABLE
	462X				
000.006	463X	PIC.COD	DS	0	CODE STARTS HERE
000.006	464	XTEXT	FBDEF		

HEADING.

FBDEF

15:18:49 16-MAY-80

466X ** FILE BLOCK DEFINITIONS.

000.000	467X				
000.000	468X	ORG	0		
000.000	469X	FB.CHA	DS	1	CHANNEL NUMBER
000.001	470X	FB.FLG	DS	1	FLAGS
000.002	471X	FB.FWA	DS	2	BUFFER FWA
000.004	472X	FB.PTR	DS	2	BUFFER POINTER
000.006	473X	FB.LIM	DS	2	LIMIT OF DATA IN BUFFER (READ OPERATIONS)
000.010	474X	FB.LWA	DS	2	LWA OF BUFFER
000.012	475X	FB.NAM	DS	4+8+4+1	NAME OF FILE
000.021	476X	FB.NAML	EQU	*-FB.NAM	
000.033	477X	FBENL	EQU	*	ENTRY LENGTH
000.033	478	XTEXT	ECDEF		

480X ** ERROR CODE DEFINITIONS.

000.000	481X				
000.000	482X	ORG	0		
000.000	483X	DS	1		NO ERROR #0
000.001	484X	EC.EOF	DS	1	END OF FILE
000.002	485X	EC.EOM	DS	1	END OF MEDIA
000.003	486X	EC.ILC	DS	1	ILLEGAL SYSCALL CODE
000.004	487X	EC.CNA	DS	1	CHANNEL NOT AVAILABLE
000.005	488X	EC.DNS	DS	1	DEVICE NOT SUITABLE
000.006	489X	EC.IDN	DS	1	ILLEGAL DEVICE NAME
000.007	490X	EC.IFN	DS	1	ILLEGAL FILE NAME
000.010	491X	EC.NRD	DS	1	NO ROOM FOR DEVICE DRIVER
000.011	492X	EC.FND	DS	1	CHANNEL NOT OPEN
000.012	493X	EC.ILR	DS	1	ILLEGAL REQUEST
000.013	494X	EC.FUC	DS	1	FILE USAGE CONFLICT
000.014	495X	EC.FNF	DS	1	FILE NAME NOT FOUND
000.015	496X	EC.UND	DS	1	UNKNOWN DEVICE
000.016	497X	EC.ICN	DS	1	ILLEGAL CHANNEL NUMBER
000.017	498X	EC.DIF	DS	1	DIRECTORY FULL
000.020	499X	EC.IFC	DS	1	ILLEGAL FILE CONTENTS
000.021	500X	EC.NEM	DS	1	NOT ENOUGH MEMORY
000.022	501X	EC.RF	DS	1	READ FAILURE
000.023	502X	EC.WF	DS	1	WRITE FAILURE
000.024	503X	EC.WPV	DS	1	WRITE PROTECTION VIOLATION
000.025	504X	EC.WP	DS	1	DISK WRITE PROTECTED
000.026	505X	EC.FAP	DS	1	FILE ALREADY PRESENT
000.027	506X	EC.DDA	DS	1	DEVICE DRIVER ABORT
000.030	507X	EC.FL	DS	1	FILE LOCKED
000.031	508X	EC.FAO	DS	1	FILE ALREADY OPEN
000.032	509X	EC.IS	DS	1	ILLEGAL SWITCH
000.033	510X	EC.UUN	DS	1	UNKNOWN UNIT NUMBER
000.034	511X	EC.FNR	DS	1	FILE NAME REQUIRED
000.035	512X	EC.DIW	DS	1	DEVICE IS NOT WRITABLE (OR WRITE LOCKED)
000.036	513X	EC.UNA	DS	1	UNIT NOT AVAILABLE
000.037	514X	EC.ILV	DS	1	ILLEGAL VALUE
000.040	515X	EC.ILO	DS	1	ILLEGAL OPTION
000.041	516X	EC.VPM	DS	1	VOLUME PRESENTLY MOUNTED ON DEVICE
000.042	517X	EC.NVM	DS	1	NO VOLUME PRESENTLY MOUNTED
000.043	518X	EC.FOD	DS	1	FILE OPEN ON DEVICE
000.044	519X	EC.NPM	DS	1	NO PROVISIONS MADE FOR REMOUNTING MORE DISKS

HEADING.

ECDEF

15:18:54 16-MAY-80

```

000.045      520X EC.DNI DS      1      DISK NOT INITIALIZED
000.046      521X EC.DNR DS      1      DISK IS NOT READABLE
000.047      522X EC.DSC DS      1      DISK STRUCTURE IS CORRUPT
000.050      523X EC.NCV DS      1      NOT CORRECT VERSION OF HDOS
000.051      524X EC.NOS DS      1      NO OPERATING SYSTEM MOUNTED
000.052      525X EC.IOI DS      1      ILLEGAL OVERLAY INDEX
000.053      526X EC.OTL DS      1      OVERLAY TOO LARGE
000.054      527      XTEXT  IOCDEF

529X **      I/O CHANNEL DEFINITIONS.
530X
000.000      531X      ORG      0
532X
000.000      533X IOC.LNK DS      2      ADDRESS OF NEXT CHANNEL, =0 IF LAST
000.002      534X IOC.DDA DS      2      THREAD JUMP TO DEVICE DRIVER (VIA DEV TABLE)
535X
000.004      536X IOC.FLG DS      1      FILE TYPE FLAGS
000.001      537X FI.ID   EQU      00000001B  =1 IF DIRECTORY DEVICE
000.002      538X FT.OR   EQU      00000010B  =1 IF OPEN FOR READ
000.004      539X FI.OW   EQU      00000100B  =1 IF OPEN FOR WRITE
000.010      540X FT.OU   EQU      00001000B  =1 IF OPEN FOR UPDATE
000.003      541X IOC.SQL EQU      *-IOC.DDA  LENGTH OF INFO FOR SEQUENTIAL FILE (FROM IOC)
542X
000.005      543X IOC.GRT DS      2      ADDRESS OF GROUP RESERVATION TABLE
000.007      544X IOC.SPG DS      1      SECTORS PER GROUP, THIS DEVICE
000.010      545X IOC.CGN DS      1      CURRENT GROUP NUMBER
000.011      546X IOC.CSI DS      1      CURRENT SECTOR INDEX (IN CURRENT GROUP)
000.012      547X IOC.LGN DS      1      LAST GROUP NUMBER
000.013      548X IOC.LSI DS      1      LAST SECTOR INDEX (IN LAST GROUP)
000.010      549X IOC.DRL EQU      *-IOC.FLG  LENGTH OF INFO NORMALLY COPIED BACK TO
550X *      THE CHANNEL TABLE
000.014      551X IOC.ITA DS      2      DEVICE TABLE ADDRESS FOR THIS DEVICE
000.016      552X IOC.DES DS      2      SECTOR NUMBER OF DIRECTORY ENTRY
000.020      553X IOC.DEV DS      2      DEVICE CODE
000.022      554X IOC.UNI DS      1      UNIT NUMBER (0-9)
000.021      555X IOC.DIL EQU      *-IOC.DDA  LENGTH OF INFO FOR DIRECTORY FILE (FROM IOC)
556X
000.023      557X IOC.DIR DS      DIRELEN  DIRECTORY ENTRY
558X
000.052      559X IOCELEN EQU      *      IOC ENTRY LENGTH
560X
000.001      561X IOCCTD EQU      1      INDEX OF USER CHANNEL #0 IN CHANTAB (FIRST = 0)
562
563 ****

```

```

566
042.170          567      ORG      USERFWA-ABS.COD
042.170 377 000    568      DB      3770,FT.ABS          ABS HEADER
042.172 200 042    569      DW      USERFWA          LOAD ADDRESS
042.174 205 032    570      DW      MEML-USERFWA        SIZE
042.176 055 070    571      DW      PRS              ENTRY
572
042.200          573      START EQU      *          START HERE AFTER *PRS*
574
042.200 315 317 052 575      CALL     BDT          BUILD DYNAMIC TABLES
042.203 332 325 042 576      JC       EXIT        PROBLEMS
042.206 076 001     577      MVI      A,1          PASS 1
042.210 315 344 042 578      CALL     ASM          ASSEMBLE PASS 1
579
042.213 315 366 056 580      CALL     RSF          REAWIND SOURCE FILE
042.216 076 002     581      MVI      A,2
042.220 315 344 042 582      CALL     ASM          ASSEMBLE PASS 2
042.223 072 144 067 583      LDA      FTFLAG
042.226 376 001     584      CPI      FT.PIC
042.230 302 263 042 585      JNE      H8ASM1        NOT PIC CODE
586
587 *          IS PIC CODE, UPDATE LENGTH POINTER IN HEADER
588
042.233 052 134 067 589      LHL     ORG
042.236 353         590      XCHG          (DE) = CODE LENGTH
042.237 041 003 000 591      LXI      H,PIC.LEN+1
042.242 042 134 067 592      SHLD    ORG          SET ADDRESS FOR HEADER FIELD+1
042.245 173         593      MOV      A,E
042.246 315 070 052 594      CALL     ABV          WRITE LENGTH
042.251 041 004 000 595      LXI      H,PIC.LEN+2        (HL) = ORG+1
042.254 042 134 067 596      SHLD    ORG          WRITE 2ND BYE OF PIC.LEN
042.257 172         597      MOV      A,D
042.260 315 070 052 598      CALL     ABV          2ND HALF
042.263 072 150 067 599      H8ASM1  LDA      BINFNAM
042.266 247         600      ANA      A
042.267 312 301 042 601      JZ       H8ASM2        NO FILE, DONT FINISH IT
042.272 315 123 060 602      CALL     WBB          FLUSH BUFFER TO DISK
042.275 076 000     603      MVI      A,CN.BIN
042.277 377 046     604      DB       SYSCALL,.CLOSE  CLOSE FILE
605
606 *          ASSEMBLY COMPLETE.
607
042.301 315 051 056 608      H8ASM2  CALL     PAS          PRINT ASSEMBLY STATISTICS
042.304 315 104 055 609      CALL     FNF          FORCE NEW PAGE
042.307 041 226 067 610      LXI      H,LISTFB
042.312 315 052 062 611      CALL     $FCLO        CLOSE LISTING FILE
042.315 315 003 061 612      CALL     $CCO          CLEAR CTL-0
042.320 041 163 056 613      LXI      H,PASA
042.323 377 003     614      DB       SYSCALL,.PRINT  PRINT FINAL MESSAGE
615
042.325 257         616      EXIT    XRA      A
042.326 377 000     617      DB       SYSCALL,.EXIT

```

```

619 **      CCHIT - CTL-C HIT.
620 *
621
622
042.330 315 138 031 623 CCHIT CALL $TYPYX
042.333 136 303 624 DB 'C','C'+2000
042.335 303 325 042 625 JMP EXIT

627 **      RESTART - RECOVER FROM ERRORS.
628 *
629 *      THIS CODE IS ENTERED AFTER FILE ERRORS ARE
630 *      DISCOVERED AND COMPLAINED ABOUT.
631
632
042.340 076 001 633 RESTART MVI A,1
042.342 377 000 634 DB SYSCALL,.EXIT EXIT WITH RESET

```

```

638 **      ASM IS CALLED TO MAKE AN ASSEMBLY PASS.
639 *
640 *      IF PASS = 1, ASSEMBLE TEXT, CREATE SYMBOL TABLE, PRODUCE
641 *      NO BINARY OR LISTING.
642 *
643 *      IF PASS = 2, ASSEMBLE TEXT, DEFINE NO SYMBOLS, PRODUCE BINARY
644 *      AND LISTING.
645 *
646 *      ENTRY (A) = PASS NUMBER (1 OR 2)
647 *      EXIT 'END' STATEMENT READ
648 *      ERRCNT = NUMBER OF STATEMENTS WITH ERRORS
649 *      STATNO = NUMBER OF STATEMENTS READ
650
651
042.344 062 122 067 652 ASM STA PASS
042.347 021 115 045 653 LXI D,NULTITL /78.10.6C/
042.352 315 031 045 654 CALL TITLE. /78.10.6C/
042.355 021 115 045 655 LXI D,NULTITL /78.10.6C/
042.360 315 104 045 656 CALL STL. /78.10.6C/
042.363 041.000.000 657 LXI H,0
042.366 042 123 067 658 SHLD ERRCNT CLEAR ERROR COUNT
042.371 042 126 067 659 SHLD STATNO
042.374 257 660 XRA A
042.375 062 347 067 661 STA CNDFLG CLEAR CONDITIONAL ASSEMBLY
043.000 062 133 067 662 STA ENDFLG CLEAR END FLAG
043.003 062 352 067 663 STA PAGNUM CLEAR PAGE NUMBER /10.04.77/
043.006 062 141 067 664 STA GRPFLG SET IN FIRST GROUP
043.011 062 146 067 665 STA CODEFLG CLEAR 'CODE' PSEUDO SEEN FLAG
043.014 062 144 067 666 STA FTFLAG CLEAR CODE GENERATION TYPE FLAG
043.017 076 001 667 MVI A,LST.L
043.021 062 130 067 668 STA LSTCTL
043.024 071 669 DAD SP (HL) = (SP)
043.025 042 352 043 670 SHLD ASMB SAVE STACK POINTER VALUE
043.030 041.200.042 671 LXI H,USERFWA DEFAULT ORG
043.033 042 134 067 672 SHLD ORG
673
674 *      SET INITIAL LISTING CONTROL BITS
675
043.036 041 131 067 676 LXI H,LSTCTL
043.041 072 130 067 677 LDA LSTCTL
043.044 266 678 ORA M SET FORCED ON BITS
043.045 043 679 INX H
000.000 680 ERRNZ LSTCTL-LSTCTL-1 (HL) = #LSTCTL
043.046 246 681 ANA M CLEAR FORCED OFF BITS
043.047 062 130 067 682 STA LSTCTL
683
684 *      ASSEMBLE ANOTHER LINE OF PROGRAM.
685
043.052 052 134 067 686 ASM1 LHLD ORG
043.055 042 136 067 687 SHLD SORG SAVE COPY OF ORG
043.060 315 275 056 688 CALL PDL PREPARE DISPLAY LINE
043.063 052 126 067 689 LHLD STATNO
043.066 043 690 INX H
043.067 042 126 067 691 SHLD STATNO INCREMENT STATEMENT NUMBER
043.072 315 145 057 692 CALL UNL UNPACK NEXT LINE
043.075 312 230 043 693 JZ ASM4 IS COMMENT

```

```

694
695 *      CRACK OPCODE.
696
043.100 041 174 064 697      LXI      H,OPCTAB
043.103 021 067 070 698      LXI      D,OPCODE
043.106 257 699      XRA      A
043.107 315 310 055 700      CALL     LVT          LOCATE VALUE IN TABLE
043.112 332 130 043 701      JC          ASM2          FOUND
043.115 315 003 057 702      CALL     SEF          *0* ERROR
043.120 100 703      DB          ERR.0
043.121 001 000 000 704      LXI      B,0
043.124 120 705      MOV      D,B          GENERATE 3 00 BYTES
043.125 303 364 043 706      JMP      ASM7
707
043.130 176 708      ASM2      MOV      A,M          (A) = OPCODE INDEX
000.000 709      ERRNZ     OF,CE-2000      CODE ASSUMES = 2000
043.131 027 710      RAL          CHECK FOR CONDITIONAL ELIGIBILITY
043.132 332 144 043 711      JC          ASM3          WILL PROCESS REGARDLESS OF *IF*
043.135 072 347 067 712      LDA      CNDFLG
043.140 017 713      RRC
043.141 332 237 043 714      JC          ASM5          ASM TO SKIP ASSEMBLING
043.144 176 715      ASM3      MOV      A,M          (A) = OPCODE INDEX
043.145 346 100 716      ANI      OF,LD      SEE IF TO DEFINE LABEL
043.147 345 717      PUSH     H
043.150 314 335 053 718      CZ          DLH          IF TO DEFINE LABEL
043.153 341 719      POP      H
043.154 176 720      MOV      A,M          (A) = OPCODE INDEX
043.155 346 077 721      ANI      770          CLEAR FLAG BITS
043.157 043 722      INX      H
043.160 106 723      MOV      B,M          (B) = OPCODE
043.161 345 724      PUSH     H          SAVE ADDRESS IN OPCTAB
043.162 365 725      PUSH     PSW          SAVE INDEX
043.163 376 015 726      CPI      PSIND
043.165 076 001 727      MVI      A,1          ASSUME IS MACHINE CODE, WHICH IS GROUP 2
043.167 332 173 043 728      JC          ASM3.1      IS MACHINE CODE
043.172 170 729      MOV      A,B          IS PSEUDO. USE IT'S GROUP FLAG
043.173 041 141 067 730      ASM3.1 LXI      H,GRPFLG
043.176 266 731      ORA      M          <0 IF THIS OPERATION IN MAIN GROUP
043.177 167 732      MOV      M,A
043.200 304 076 044 733      CNZ     GCP          GENERATE 'CODE' PSEUDO, IF NECESSARY
043.203 361 734      POP      PSW          (A) = OPERATION INDEX
043.204 007 735      RLC
043.205 041 252 043 736      LXI      H,ASMA
043.210 315 101 030 737      CALL     $DADA.          (HL) = ADDRESS OF ADDRESS
043.213 315 211 030 738      CALL     $HLIHL
043.216 072 122 067 739      LDA      PASS
043.221 017 740      RRC          (C) SET IF PASS = 1
043.222 170 741      MOV      A,B          (A) = OPCODE
043.223 343 742      XTHL          (HL) = ADDRESS OF OPCTAB ENTRY
043.224 021 074 070 743      LXI      D,EXPWRK      (DE) = EXPRESSION POINTER
043.227 311 744      RET          ENTER PROCESSOR
745
746 *      HAVE COMMENT
747
043.230 072 347 067 748      ASM4      LDA      CNDFLG
043.233 017 749      RRC

```

```

043.234 322 044 044 750 JNC ASM13 LIST IF NOT CONDITIONAL ASSEMBLY
751
752 * AM SKIPPING LINE. SEE IF TO LIST.
753
043.237 072 130 067 754 ASMS LDA LSTCTL
043.242 346 002 755 ANI LST.I
043.244 312 053 044 756 JZ ASM14 IGNORE
043.247 303 044 044 757 JMP ASM13 LIST BY ITSELF
758
043.252 759 ASMA EQU * PROCESSOR JUMP TABLE
760
761 * MACHINE OPCODES.
762
043.252 001 044 763 DW SNG SINGLE BYTE - NO OPERAND
043.254 110 047 764 DW IMM IMMEDIATE ARITHMETIC
043.256 117 047 765 DW THR THREE-BYTE OPCODES
043.260 126 047 766 DW RAO REG ARITH - TYPE 1
043.262 135 047 767 DW RAT REG ARIT - TYPE 2
043.264 147 047 768 DW RPO REG PAIR GP 1
043.266 156 047 769 DW RPT REG PAIR GP 2
043.270 165 047 770 DW INX INX INSTRUCTION
043.272 206 047 771 DW MVI MVI INSTRUCTION
043.274 231 047 772 DW INDX LDAX, STAX INSTRUCTIONS
043.276 256 047 773 DW RST RST INSTRUCTION
043.300 303 047 774 DW LXI LXI INSTRUCTION
043.302 322 047 775 DW MOV MOV INSTRUCTION
776
777 * PSEUDO OPCODES.
778
000.015 779 PSIND EQU *-ASMA/2 INDEX OF 1ST PSEUDO OP
043.304 116 044 780 DW DB DB
043.306 220 044 781 DW DS DS
043.310 240 044 782 DW DW DW
043.312 273 044 783 DW EJECT EJECT
043.314 151 045 784 DW ELSE ELSE
043.316 007 046 785 DW END END
043.320 167 045 786 DW ENDIF ENDIF
043.322 102 046 787 DW EQU EQU
043.324 204 045 788 DW ERRXX ERRZR, ERRNZ, ERRPL, ERRMI
043.326 120 045 789 DW IF IF
043.330 271 045 790 DW LOF LOF
043.332 252 045 791 DW LON LON
043.334 354 045 792 DW FORG ORG
043.336 150 046 793 DW SET SET
043.340 311 044 794 DW SPACE SPACE
043.342 073 045 795 DW STL STL
043.344 020 045 796 DW TITLE TITLE
043.346 164 046 797 DW CODE CODE
043.350 357 046 798 DW XTEXT XTEXT
799
043.352 000 000 800 ASME DW 0 SAVED STACKPOINTER

```



```

802 **      MACHINE AND PSEUDO OP CODE PROCESSORS EXIT TO
803 *
804 *      THESE POINTS:

```

```

806 **      ASM6 - EXIT WITH 2 BYTES OF DATA
807 *
808 *      PLACE 2 BYTES IN LINE AND LIST IT.
809 *
810 *      ENTRY  (B) = 1ST BYTE, (C) = 2ND
811 *
812 *

```

```

043.354 170      813 ASM6  MOV    A,B
043.355 315 351 055 814      CALL   ORB          OUTPUT 1ST BYTE
043.360 171      815      MOV    A,C
043.361 303 001 044 816      JMP     ASM8

```

```

818 **      ASM7 - EXIT WITH 3 BYTES OF DATA.
819 *
820 *      PLACE 3 BYTES IN LINE AND LIST IT
821 *
822 *      ENTRY  (D) = 1ST BYTE
823 *            (C) = 2ND BYTE
824 *            (B) = 3RD BYTE
825 *
826 *

```

```

043.364 172      827 ASM7  MOV    A,D
043.365 315 351 055 828      CALL   ORB          1ST BYTE
043.370 315 323 056 829      CALL   RRI          RECORD RELOCATION INFORMATION
043.373 171      830      MOV    A,C
043.374 315 351 055 831      CALL   ORB          2ND BYTE
043.377 257      832      XRA     A

```

```

834 **      ASM8 - EXIT WITH ONE BYTE OF CODE.
835 *
836 *      PLACE 1 BYTE IN LINE AND LIST IT.
837 *
838 *      ENTRY  (A) = VALUE
839 *
840 *

```

```

044.000 200      841 ASM8.  ADD    B          ENTRY TO OUTPUT A+B
044.001 315 351 055 842 ASM8  CALL   ORB          OUTPUT BYTE
044.004 303 032 044 843      JMP     ASM11       LIST WITH ORG

```

845 ** ASM10 - REQUIRE STATEMENT END, THEN LIST STATEMENT

846 * WITHOUT ORG.

847 *

848 * USED BY DW AND DB

849

850

044.007 033

851 ASM10

DCX D

044.010 032

852

LDAX D

(A) = LAST CHARACTER

044.011 315 031 053

853

CALL CEF

CHECK FOR END OF FIELD CHARACTER

044.014 312 044 044

854

JZ ASM13

GOT A LEGAL TERMINATOR

044.017 315 003 057

855 FLGERA

CALL SEF

** ERROR

044.022 010

856

DB ERR.A

044.023 303 044 044

857

JMP ASM13

859 ** ERR.O. - FLAG *O* ERROR, LIST LINE WITH ORG

860 *

861 *

USED WHEN THE OPCODE IS SYNTACTICALLY VALID, JUST ILLEGAL

862 *

IN THAT CONTEXT

863

864

044.026 315 003 057

865 ERR.O.

CALL SEF

044.031 100

866

DB ERR.O

867 *

JMP ASM11

LIST LINE WITH ORG

869 ** ASM11 - LIST LINE WITH ORG.

870 *

871

872

044.032 052 136 067

873 ASM11

LHLD SORG

(HL) = SAVED ORG VALUE

044.035 104

874

MOV B,H

044.036 115

875

MOV C,L

877 ** ASM11. - LIST LINE WITH (BC) = ORG

878 *

879

880

044.037 140

881 ASM11.

MOV H,B

044.040 151

882

MOV L,C

```

884 **      ASM12 - LIST LINE WITH (HL) AS ORG
885 *
886
887
044.041 315 121 057 888 ASM12 CALL UDL. UNPACK (HL) TO LINE

```

```

890 **      ASM13 - LIST WITHOUT ORG.
891 *
892
893
044.044 315 370 053 894 ASM13 CALL DLL DISPLAY LISTING LINE
044.047 257 895 XRA A
044.050 062 140 067 896 STA ERRFLG CLEAR ERROR

```

```

898 **      ASM14 - NO LIST.
899 *
900
901
044.053 072 140 067 902 ASM14 LDA ERRFLG
044.056 247 903 ANA A
044.057 302 044 044 904 JNZ ASM13 ERROR - MUST LIST
044.062 052 352 043 905 LHLD ASMB
044.065 371 906 SPHL RESET STACK POINTER
044.066 072 133 067 907 LDA ENDFLG
044.071 247 908 ANA A
044.072 300 909 RNZ
044.073 303 052 043 910 JMP ASM1 EXIT IF *END* READ
PROCESS ANOTHER CARD

```

```

912 **      GCP - GENERATE 'CODE' PSEUDO.
913 *
914 *
915 *      GCP IS CALLED AFTER THE GROUP CLASSIFICATION OF EVERY STATEMENT
916 *      HAS BEEN DETERMINED. IF THIS TO-BE-ASSEMBLED STATEMENT HAS
917 *      PUT US INTO GROUP 2, AND NO 'CODE' PSEUDO HAS BEEN ENCOUNTERED,
918 *      THEN WE MUST FAKE UP A
919 *
920 *
921 *      CODE ABS
922 *
923 *      STATEMENT.
924 *
925 *      ENTRY (A) = (GRPFLG)
926 *
927 *      EXIT NONE
928 *      USES A,F
929
044.076 072 146 067 928 GCP LDA CODEFLG
044.101 247 929 ANA A
044.102 300 930 RNZ CODE PSEUDO ENCOUNTERED

```

ASM - HDOS RESIDENT ASSEMBLER

ASM - MAKE ASSEMBLY PASS.

GCP

HEATH HBASM V1.4 01/20/78

PAGE 20

15:19:06 16-MAY-80

044.103 315 054 031 931

CALL \$SAVALL

SAVE REGS

044.106 006 000 932

MVI R,FT.ABS

DO ABS

044.110 315 231 046 933

CALL CODE2

PROCESS CODE PSEUDO

044.113 303 047 031 934

JMP \$RSTALL

RESTORE AND RETURN

```

937 **      DB - DEFINE BYTE.
938 *
939 *      DB      VAL,...,VAL
940
941
044.116      942 DB      EQU      *
044.116 315 116 057 943      CALL      UOL      UNPACK ORG INTO LINE
044.121 325      944 DB1     PUSH      D
945
946 *      EXAMINE NEXT ELEMENT.
947
044.122 032      948      LDAX      D
044.123 376 047 949      CPI      QUOTE
044.125 302 157 044 950      JNE      DB3      NOT QUOTE
951
952 *      HAVE QUOTED STRING. SEE IF IS PART OF EXPRESSION, OR JUST
953 *      A STAND-ALONE.
954
044.130 023      955      INX      D
044.131 315 252 055 956 DB2     CALL      GSC      GET STRING CHARACTER
044.134 302 131 044 957      JNZ      DB2
958
044.137 032      958      LDAX      D
044.140 247      959      ANA      A
044.141 372 172 044 960      JM      DB4      MARKED CHARACTER /80.02.GC/
044.144 315 031 053 961      CALL      CEF      CHECK FOR END OF FIELD /80.02.GC/
044.147 312 172 044 962      JZ      DB4      /80.02.GC/
044.152 376 054 963      CPI      ','      CHECK FOR EXPRESSION /80.02.GC/
044.154 312 172 044 964      JZ      DB4      /80.02.GC/
965
966 *      HAVE BYTE EXPRESSION
967
044.157 321      968 DB3     POP      D
044.160 315 341 054 969      CALL      E8B      EVALUATE TO 8 BITS
044.163 171      970      MOV      A,C      (A) = VALUE
044.164 315 351 055 971      CALL      OBB      OUTPUT BINARY BYTE
044.167 303 206 044 972      JMP      DB6
973
974 *      HAVE QUOTED STRING
975
044.172 321      976 DB4     POP      D
044.173 023      977      INX      D
044.174 257      978      XRA      A
044.175 304 351 055 979 DB5     CNZ      OBB      OUTPUT BINARY BYTE (SKIP 1ST TIME)
044.200 315 252 055 980      CALL      GSC      GET STRING CHARACTER
044.203 302 175 044 981      JNZ      DB5      IF MORE
982
983 *      END OF BYTE VALUE. SEE IF MORE FOLLOW
984
044.206 032      985 DB6     LDAX      D
044.207 023      986      INX      D
044.210 376 054 987      CPI      ','
044.212 312 121 044 988      JE      DB1      IF MORE
044.215 303 007 044 989      JMP      ASM10     REQUIRE END AND LIST LINE

```

DS - DEFINE STORAGE.

DS

15:19:08 16-MAY-80

```

          993 **      DS - DEFINE STORAGE.
          994 *
          995 *      DS      EXPR
          996
          997
044.220          998 DS      EQU      *
044.220 315 372 054 999      CALL     EPO      EVALUATE FOR PASS 1
044.223 302 032 044 1000     JNZ      ASM11     EXIT - ERROR
044.226 052 134 067 1001     LHLD     ORG
044.231 011      1002     DAD      B
044.232 042 134 067 1003     SHLD     ORG
044.235 303 032 044 1004     JMP      ASM11     LIST WITH ORG

```

DW - DEFINE WORD.

15:19:08 16-MAY-80

```
1007 ** DW - DEFINE WORD.
1008 *
1009 * DW EXP1,...,EXPN
1010
1011
044.240 1012 DW EQU *
044.240 315 116 057 1013 CALL UOL UNPACK ORG INTO LINE
1014
1015 * DECODE NEXT ELEMENT
1016
044.243 315 215 051 1017 DW1 CALL EVL
044.246 315 323 056 1018 CALL RRI RECORD RELOCATION INFORMATION
044.251 171 1019 MOV A,C
044.252 315 351 055 1020 CALL ORB OUTPUT BINARY BYTE
044.255 170 1021 MOV A,B
044.256 315 351 055 1022 CALL ORB OUTPUT 2ND HALF
044.261 032 1023 LDAX D
044.262 023 1024 INX D
044.263 376 054 1025 CPI ','
044.265 312 243 044 1026 JE DW1 IF MORE TO GO
044.270 303 007 044 1027 JMP ASM10 ENSURE END AND LIST
```

```

1030 **      EJECT - SET PAGE EJECT.
1031 *
1032
1033
044.273 315 041 053 1034 EJECT CALL CLE CHECK LISTING ELIGIBILITY
044.276 312 053 044 1035 JZ ASM14 NOT TO LIST
044.301 076 001 1036 MVI A,1
044.303 062 350 067 1037 STA EJEFLG FORCE PAGE EJECT
044.306 303 053 044 1038 JMP ASM14 NO LIST

1040 **      SPACE N,M
1041 *
1042 *      SPACE N LINES IF < M LINES REMAIN ON THE PAGE. OTHERWISE, EJECT.
1043
1044
044.311 332 053 044 1045 SPACE JC ASM14 IGNORE IF PASS 1
044.314 315 041 053 1046 CALL CLE CHECK LISTING ELIGIBILITY
044.317 312 053 044 1047 JZ ASM14 NO LISTING
044.322 315 341 054 1048 CALL EBB EVALUATE 8 BIT EXPRESSION
044.325 305 1049 PUSH B SAVE N
044.326 032 1050 LDAX D
044.327 376 054 1051 CPI 7
044.331 302 347 044 1052 JNE SPC1 NO M
044.334 023 1053 INX D
044.335 315 341 054 1054 CALL EBB EVALUATE M
044.340 072 351 067 1055 LDA LINCNT
044.343 271 1056 CMP C
044.344 332 273 044 1057 JC EJECT IF TO FORCE EJECT
1058
044.347 301 1059 SPC1 POP B (C) = N
044.350 072 214 067 1060 LDA PAGEDF
044.353 127 1061 MOV D,A (D) = PAGESIZ
044.354 072 351 067 1062 LDA LINCNT
044.357 272 1063 CMP D
044.360 312 053 044 1064 JE ASM14 AT TOP OF PAGE, DONT SPACE
044.363 221 1065 SUB C (A) = PROPOSED NEW LINE NUMBER
044.364 332 273 044 1066 JC EJECT WILL BRING NEW PAGE
044.367 315 100 053 1067 SPC2 CALL COL COUNT OUTPUT LINE
044.372 305 1068 PUSH B SAVE COUNT
044.373 041 226 067 1069 LXI H,LISTFB
044.376 021 017 045 1070 LXI D,SPCA
045.001 001 001 000 1071 LXI B,1
045.004 315 047 063 1072 CALL $FWRIB WRITE NEW LINE
045.007 301 1073 POP B (C) = COUNT
045.010 015 1074 DCR C
045.011 302 367 044 1075 JNZ SPC2 IF NOT DONE
045.014 303 053 044 1076 JMP ASM14 EXIT WITH NO LIST
1077
045.017 012 1078 SPCA DB NL SPACE LINE

```



```

1081 **      TITLE - SETUP PAGE TITLE.
1082 *
1083 *      TITLE 'NEW TITLE'
1084
1085
045.020 315 031 045 1086 TITLE CALL TITLE.
045.023 332 017 044 1087 JC FLGERA
045.026 303 053 044 1088 JMP ASM14
1089
045.031 041 233 066 1090 TITLE. LXI H,TTLTXT-1 (HL) = ADDRESS FOR TEXT
045.034 006 062 1091 MVI B,TXTL (B) = MAX LENGTH
045.036 032 1092 TTL1 LDAX D
045.037 376 047 1093 CPI QUOTE
045.041 302 071 045 1094 JNE TTL4 NO TITLE
045.044 023 1095 INX D
045.045 005 1096 TTL2 DCR B
045.046 312 071 045 1097 JZ TTL4 TOO MANY CHARACTERS
045.051 043 1098 INX H
045.052 315 252 055 1099 CALL GSC GET STRING CHARACTER
045.055 167 1100 MOV M,A
045.056 302 045 045 1101 JNZ TTL2 IF MORE TO GO
1102
1103 *      FILL REMAINDER OF LINE WITH BLANKS.
1104
045.061 066 040 1105 TTL3 MVI M,' '
045.063 043 1106 INX H
045.064 005 1107 DCR B
045.065 302 061 045 1108 JNZ TTL3
045.070 311 1109 RET
1110
045.071 067 1111 TTL4 STC FLAG ERROR
045.072 311 1112 RET

```

```

1114 **      STL - SUBTITLE LINE
1115 *
1116 *      STL 'NEW SUB-TITLE'
1117
1118
045.073 315 104 045 1119 STL CALL STL.
045.076 332 017 044 1120 JC FLGERA
045.101 303 053 044 1121 JMP ASM14
1122
045.104 006 063 1123 STL. MVI B,STXTL
045.106 041 343 066 1124 LXI H,STLTXT-1
045.111 315 036 045 1125 CALL TTL1
045.114 311 1126 RET
1127
045.115 047 040 047 1128 MULTITL DB 0470, ,0470

```

```

1131 **      IF - INITIATE CONDITIONAL ASSEMBLY.
1132 *
1133 *      IF      EXPR
1134 *
1135 *      ASSEMBLE IF EXPR = 0
1136
1137
045.120 315 372 054 1138 IF      CALL      EPD      EVALUATE PASS 1
045.123 302 017 044 1139      JNZ      FLGERA      ERROR
045.126 041 347 067 1140      LXI      H,CNDIFLG
045.131 176      1141      MOV      A,M
045.132 247      1142      ANA      A
045.133 302 017 044 1143      JNZ      FLGERA      CONDITIONAL ASSEMBLY ALREADY IN EFFECT
045.136 170      1144      MOV      A,B
045.137 261      1145      ORA      C
045.140 066 200 1146      MVI      M,2000
045.142 312 146 045 1147      JZ      IF1      AM TO ASSEMBLE
045.145 064      1148      INR      M      AM TO SKIP
045.146 303 037 044 1149 IF1      JMP      ASM11.      LIST WITH (BC) = ORG

```

```

1151 **      ELSE - TOGGLE CONDITIONAL ASSEMBLY.
1152 *
1153 *      ELSE
1154
1155
045.151 041 347 067 1156 ELSE      LXI      H,CNDIFLG
045.154 176      1157      MOV      A,M
045.155 247      1158      ANA      A
045.156 362 017 044 1159      JP      FLGERA      CONDITIONAL ASSEMBLY NOT IN EFFECT
045.161 356 001 1160      XRI      1
045.163 167      1161      MOV      M,A
045.164 303 044 044 1162      JMP      ASM13      PRINT NO INFORMATION

```

```

1164 **      ENDIF - COMPLETE CONDITIONAL PROCESSING.
1165 *
1166 *      ENDIF
1167
1168
045.167 041 347 067 1169 ENDIF      LXI      H,CNDIFLG
045.172 176      1170      MOV      A,M
045.173 066 000 1171      MVI      M,0
045.175 247      1172      ANA      A
045.176 312 017 044 1173      JZ      FLGERA      CONDITIONAL ASSEMBLY NOT IN EFFECT
045.201 303 044 044 1174      JMP      ASM13      LIST WITH NO INFO

```

```

1177 **      ERRXX - CONDITIONAL ERRORS.
1178 *
1179 *      ERRZR EXPR
1180 *      ERRNZ EXPR
1181 *      ERRPL EXPR
1182 *      ERRMI EXPR
1183 *
1184 *      FLAG A *P* ERROR IF EXPR MATCHES CONDITION.
1185
1186
045.204 315 215 051 1187 ERRXX CALL EVL
045.207 176 1188 MOV A,M (A) = TYPE INDEX
045.210 041 037 044 1189 LXI H,ASM11. LIST WITH (BC) = ORG
045.213 315 223 045 1190 CALL ERR1
045.216 315 003 057 1191 CALL SEF *P* ERROR
045.221 200 1192 DB ERR.F
045.222 351 1193 PCHL GO TO ASM12
1194
045.223 315 076 031 1195 ERR1 CALL $TBRA
045.226 004 1196 DB ERRZR-*
045.227 007 1197 DB ERRNZ-*
045.230 012 1198 DB ERRPL-*
045.231 015 1199 DB ERRMI-*

045.232 170 1201 ERRZR MOV A,B
045.233 261 1202 ORA C
045.234 310 1203 RZ ERR1 *P* ERROR
045.235 351 1204 PCHL LIST WITH (HL) = ORG

045.236 170 1206 ERRNZ MOV A,B
045.237 261 1207 ORA C
045.240 300 1208 RNZ ERR1 *P* ERROR
045.241 351 1209 PCHL

045.242 170 1211 ERRPL MOV A,B
045.243 247 1212 ANA A
045.244 360 1213 RP ERR1 *P* ERROR
045.245 351 1214 PCHL

045.246 170 1216 ERRMI MOV A,B
045.247 247 1217 ANA A
045.250 370 1218 RM ERR1 *P* ERROR
045.251 351 1219 PCHL

```

```

1222 **      LON - LISTING ON.
1223 *
1224 *      LON      CCC
1225 *
1226 *      TURN OPTIONS ON, OPTIONS =
1227 *
1228 *      L      MASTER LISTING
1229 *      I      IF-SKIPPED LINES
1230 *      C      INCLUDED CODE
1231 *      G      GENERATED CODE
1232
1233
045.252 315 314 045 1234 LON CALL LST
045.255 312 044 044 1235 JZ ASM13 ALL DONE
045.260 266 1236 ORA M
045.261 041 132 067 1237 LXI H,LSTCTL
045.264 246 1238 ANA M CLEAR BITS MENTIONED IN /N: SWITCH
045.265 002 1239 STAX B
045.266 303 252 045 1240 JMP LON PROCESSES NEXT

```

```

1242 **      LOF - LISTING OFF.
1243 *
1244 *      LOF      CCC
1245 *
1246 *      TURN LON OPTIONS BACK OFF.
1247
1248
045.271 315 314 045 1249 LOF CALL LST
045.274 312 044 044 1250 JZ ASM13 DONE
045.277 365 1251 PUSH PSW SAVE OLD VALUE
045.300 174 1252 MOV A,M
045.301 057 1253 CMA
045.302 341 1254 POP H (H) = OLD (A)
045.303 244 1255 ANA H (A) = (.NOT.BIT).AND.LSTCTL
045.304 041 131 067 1256 LXI H,LSTCTL
045.307 266 1257 ORA M SET BITS MENTIONED IN /L: SWITCH
045.310 002 1258 STAX B
045.311 303 271 045 1259 JMP LOF

```

```

1261 **      LST - PERFORM LON AND LOF PRESET.
1262 *
1263 *      LST PERFORMS SOME FIXED TASKS FOR LON AND LOF.
1264 *
1265 *      ENTRY (DE) = NEXT EXPRESSION CHARACTER
1266 *      EXIT 'Z' SET IF END OF LIST
1267 *      'Z' CLEAR IF VALID CHARACTER
1268 *      IF NOT AT END:
1269 *      (DE) UPDATED
1270 *      (BC) = #LSTCTL
1271 *      (A) = (LSTCTL)
1272 *      (HL) = ADDRESS OF OPTION BIT

```

			1273			
			1274			
045.314	032		1275	LST	LDAX	D
045.315	315 031 053		1276		CALL	CEF
045.320	310		1277		RE	CHECK FOR END OF FIELD CHARACTER
045.321	023		1278		INX	END OF FIELD
045.322	041 342 045		1279		LXI	D
045.325	315 261 061		1280		CALL	H,LSTA
045.330	302 017 044		1281		JNZ	\$TBLS
045.333	366 001		1282		ORI	TABLE SEARCH
045.335	001 130 067		1283		LXI	NOT GOOD OPTION
045.340	012		1284		LDAX	CLEAR 'Z'
045.341	311		1285		RET	
			1286			
			1287			
045.342			1288	LSTA	EDU	*
045.342	114 001		1289		DB	OPTION TABLE
045.344	107 200		1290		DB	'L',LST.L
045.346	111 002		1291		DB	'G',LST.G
045.350	103 004		1292		DB	'I',LST.I
045.352	000 000		1293		DB	'C',LST.C
					DB	0,0

		1296	**	ORG - SET ORIGIN COUNTER.	
		1297	*		
		1298	*	ORG EXPR	
		1299	*		
		1300	*	EXPRESSION MUST EVALUATE PASS 1	
		1301			
		1302			
045,354	072 144 067	1303	FORG	LDA	FTFLAG
045,357	247	1304		ANA	A
000,000		1305		ERRNZ	FT,ABS
045,360	302 026 044	1306		JNZ	ERR.O.
045,363	315 372 054	1307		CALL	EPD
045,366	302 017 044	1308		JNZ	FLGERA
045,371	140	1309		MOV	H,B
045,372	151	1310		MOV	L,C
045,373	042 134 067	1311		SHLD	ORG
045,376	042 136 067	1312		SHLD	SORG
046,001	315 335 053	1313		CALL	DLH
046,004	303 032 044	1314		JMP	ASM11

OPCODE ERROR
 EVALUATE PASS 1
 BAD VALUE

SET NEW ORG
 SET TO COME OUT ON LISTING
 DEFINE LABEL HERE
 LIST WITH ORG

```

1317 **      END - END OF PROGRAM.
1318 *
1319 *      END
1320
1321
046.007 315 116 057 1322 END      CALL      UOL          UNPACK ORG INTO LINE
046.012 072 142 067 1323      LDA      XTXFLG
046.015 247      1324      ANA      A
046.016 302 026 044 1325      JNZ      ERR.O.      AM IN XTEXT
046.021 072 144 067 1326      LDA      FTFLAG
000.000      1327      ERRNZ    FT.ABS
046.024 247      1328      ANA      A
046.025 302 047 046 1329      JNZ      END1      IS PIC, CANNOT TAKE ENTRY POINT
046.030 315 372 054 1330      CALL      EP0      EVALUATE PASS 1
046.033 151      1331      MOV      L,C
046.034 140      1332      MOV      H,B
046.035 042 201 067 1333      SHLD     ABSENT      SET PROGRAM ENTRY POINT ADDRESS
046.040 257      1334      XRA      A
046.041 315 351 055 1335      CALL      DBB      ADD 00 BYTE TO END
046.044 303 072 046 1336      JMP      END3      FLAG END AND EXIT
1337
1338 *      IS PIC, DO PASS-DEPENDANT STUFF..
1339
046.047 072 122 067 1340 END1    LDA      PASS
046.052 075      1341      DCR      A
046.053 302 067 046 1342      JNZ      END2      PASS 2
046.056 052 134 067 1343      LHLD     ORG
046.061 042 211 067 1344      SHLD     PICPTR      SET ADDRESS OF RELOCATION TABLE
046.064 303 072 046 1345      JMP      END3      FLAG END AND EXIT
1346
046.067 315 202 055 1347 END2    CALL      GRT      PIC AND PASS2 - GENERATE RELOC TABLE
1348
1349
1350 **      ENTER HERE TO FORCE END OF PASS
1351
046.072      1352 END.      EQU      *
1353
046.072 076 001      1354 END3    MVI      A,1
046.074 062 133 067 1355      STA      ENDFLG
046.077 303 044 044 1356      JMP      ASM13      LIST, ORG ALREADY DECODED

```

```

1359 **      EQU - EQUIVALENCE SYMBOL.
1360 *
1361 * LAB    EQU      EXPR
1362 *
1363 *      ASSIGN VALUE OF *EXPR* TO LABEL.
1364 *
1365 *      EXPRESSION MUST EVALUATE PASS 1
1366 *
1367
046.102 365      1368 EQU      PUSH      PSW      SAVE PASS FLAG
046.103 315 372 054 1369      CALL     EPQ      EVALUATE PASS ONE
046.106 302 044 044 1370      JNZ      ASM13    ERROR
046.111 361      1371      POP      PSW      RESTORE PASS FLAG
046.112 322 126 046 1372      JNC      EQU1      PASS 2
1373
1374 *      PASS 1
1375
046.115 021 000 002 1376      LXI      D,ST,EQU*256+ST,UND
046.120 315 255 053 1377      CALL     DEF      DEFINE SYMBOL
046.123 303 044 044 1378      JMP      ASM13    EXIT
1379
1380 *      PASS 2
1381
046.126 021 056 070 1382 EQU1    LXI      D,LABEL
046.131 315 016 057 1383      CALL     SST
046.134 176      1384      MOV      A,M
000.000      1385      ERNZ     ST,DBL-2000  ASSUMES = 2000
046.135 027      1386      RAL
046.136 322 037 044 1387      JNC      ASM11.    OK, LIST WITH (BC) = ORG
046.141 315 003 057 1388      CALL     SEF      *D* ERROR
046.144 004      1389      DB      ERR,D
046.145 303 037 044 1390      JMP      ASM11.

```

```

1392 **      SET - SET VALUE.
1393 *
1394 * LAB    SET      EXPR
1395 *
1396 *      SET PERFORMS THE SAME FUNCTION AS EQU, BUT THE ASSIGNMENT IS MADE
1397 *      PASS 2, AND A VALUE MAY BE RE-SET.
1398
1399
046.150      1400 SET     EQU      *
046.150 315 215 051 1401      CALL     EVL      EVALUATE
046.153 021 003 003 1402      LXI      D,ST,SET*256+ST,SET
046.156 315 255 053 1403      CALL     DEF
046.161 303 037 044 1404      JMP      ASM11.    LIST WITH (BC) = ORG

```


CODE - PROCESS CODE PSEUDO

CODE

15:19:19 16-MAY-80

```

1408 ** CODE - PROCESS CODE PSEUDO.
1409 *
1410 * CODE ABS GENERATE ABS CODE
1411 * CODE PIC GENERATE PIC CODE
1412
1413
1414 CODE EQU *
1415 CALL CODE0 PROCESS PSEUDO
1416 JMP ASM11 LIST WITH ORG
1417
1418 CODE0 LXI H,GRPFLG
1419 MOV A,M
1420 INR M
1421 ANA A
1422 JZ CODE1 IN GROUP 1
1423 JMP ERR.0. *0* ERROR, NOT IN 1ST GROUP
1424
1425 * AM IN 1ST GROUP.
1426
1427 CODE1 LDAX D
1428 CPI 'A'
1429 MVI B,FT.ABS
1430 JE CODE2 GOT TYPE
1431 INR B (B) = FT.PIC
1432 CPI 'P'
1433 ERRNZ FT.PIC-FT.ABS-1
1434 JE CODE2 GOT IT
1435 CALL SEF
1436 DB ERR.A CANT UNDERSTAND OPERAND
1437 RET
1438
1439 * GOT A TYPE SPECIFIED
1440 *
1441 * (B) = FT.XXX
1442
1443 CODE2 MOV A,B
1444 STA FTFLAG SET TYPE
1445 ERRNZ FT.PIC*64-ST.REL
1446 RRC
1447 RRC
1448 STA RELFLG RELFLG = ST.REL IF FT.PIC
1449 MVI A,1
1450 STA CODEFLG SET CODE PSEUDO ENCOUNTERED
1451 MOV A,B
1452 ERRNZ FT.ABS
1453 ANA A
1454 JZ CODE2.5 IS ABS
1455 LXI H,0
1456 SHLD ABSFWA SET CODE DISPLACEMENT = 0
1457 LXI H,PIC.COD
1458 SHLD ORG SET DEFAULT ORG = 0 (PIC.COD FOR 1ST USER GENERATED BYTE)
1459 SHLD SORG
1460 CODE2.5 LDA PASS
1461 DCR A
1462 RZ PASS 1
1463

```

```

1464 *      IS PASS 2. GENERATE BINARY HEADER
1465
046.300 005 1466 DCR B
000.000 1467 ERRNZ FT,PIC-1
046.301 312 341 046 1468 JZ CODE3 IS PIC
1469
1470 *      IS ABSOLUTE. GENERATE
1471 *
1472 *      377Q,FT,ABS,FWA,LWA,ENTRY
1473
046.304 052 203 067 1474 LHLD ABSLWA
046.307 353 1475 XCHG
046.310 052 175 067 1476 LHLD ABSFWA (HL) = FWA GENED CODE
046.313 315 224 030 1477 CALL $CHL
046.316 031 1478 DAD D (HL) = LENGTH
046.317 042 177 067 1479 SHLD ABSLEN SET LENGTH
046.322 076 010 1480 MVI A,ABS.COD
046.324 062 172 067 1481 STA BINSKW SET BINARY SKEW IN FILE
046.327 315 141 061 1482 CALL $MOVEL
046.332 010 000 173 1483 DW ABS.COD,ABSHDR,BINBFR SET HEADER IN BUFFER
046.340 311 1484 RET
1485
1486 *      IS PIC, GENERATE
1487 *
1488 *      377Q,FT,PIC,LEN,POINTER
1489
046.341 257 1490 CODE3 XRA A
046.342 062 172 067 1491 STA BINSKW NO BINARY SKEW DUE TO HEADER
046.345 315 141 061 1492 CALL $MOVEL SET HEADER IN BUFFER
046.350 006 000 205 1493 DW PIC.COD,PICHDR,BINBFR
046.356 311 1494 RET

```

XTEXT - PROCESS XTEXT PSEUDO

XTEXT

15:19:21 16-MAY-80

```

1498 **      XTEXT - PROCESS XTEXT PSEUDO.
1499 *
1500 *      XTEXT NAME
1501 *
1502 *      LOOK ON SAME DISK AS SOURCE FILE, THEN LOOK ON OTHER
1503
1504
046.357 072 142 067 1505 XTEXT LDA   RTXFLG
046.362 247          1506 ANA   A
046.363 302 026 044 1507 JNZ   ERR.D,      ALREADY IN XTEXT
1508
1509 *      GET CURRENT DEVICE
1510
046.366 041 326 067 1511 LXI   H,RTXFB+FB.NAM
046.371 315 164 061 1512 CALL  $CPF,      COPY FILE NAME
046.374 021 074 047 1513 LXI   D,XTEXTA
046.377 041 074 070 1514 LXI   H,XTEXTB
047.002 325          1515 PUSH  D,      SAVE ADDRESS OF DEFAULT BLOCK
047.003 076 002      1516 MVI   A,CN.SOU
047.005 377 054      1517 DB     SYSCALL,NAME GET NAME OF INPUT FILE
047.007 322 021 047 1518 JNC   XTEXT0,    NO ERROR
047.012 315 003 057 1519 CALL  SEF
047.015 010          1520 DB     ERR.A
047.016 303 032 044 1521 JMP   ASM11
1522
047.021 315 141 061 1523 XTEXT0 CALL  $MOVE,    SET DEFAULT EXTENSION
047.024 003 000 105 1524 DW     3,XTEXTD,XTEXTA+3
047.032 321          1525 POP   D,      (DE) = DEFAULT BLOCK ADDRESS
047.033 041 314 067 1526 LXI   H,RTXFB
047.036 315 337 061 1527 CALL  $FOPER,    OPEN WITH DEVICE AS DEFAULT
047.041 322 064 047 1528 JNC   XTEXT1,    GOT IT
1529
1530 *      CANT OPEN ON THAT DEVICE, TRY THE OTHER
1531
047.044 021 102 047 1532 LXI   D,XTEXTC
047.047 315 337 061 1533 CALL  $FOPER,
047.052 322 064 047 1534 JNC   XTEXT1,    GOT IT
1535
1536 *      CANT FIND IT ANYWHERE !
1537
047.055 315 003 057 1538 CALL  SEF
047.060 001          1539 DB     ERR.U
047.061 303 032 044 1540 JMP   ASM11,      LIST WITH ORG
1541
1542 *      GOT IT OPEN
1543
047.064 076 001      1544 XTEXT1 MVI   A,1
047.066 062 142 067 1545 STA   RTXFLG
047.071 303 032 044 1546 JMP   ASM11,      LIST WITH ORG
1547
047.074          1548 XTEXTA DS     6,      DEFAULT BLOCK FOR FIRST TRY TO OPEN
047.102 123 131 060 1549 XTEXTC DB     'SYO',  DEFAULT BLOCK FOR 2ND TRY
047.105 101 103 115 1550 XTEXTD DB     'ACH',  EXTENSION FOR ANY FETCH

```

		1554	**	SNG - SINGLE BYTE, NO OPERAND.		
		1555	*			
		1556				
		1557				
044.001		1558	SNG	EQU	ASMB	GENERATE 1 BYTE

		1560	**	IMM - IMMEDIATE ARITHMETIC.		
		1561	*			
		1562	*	OPC	VAL	
		1563				
		1564				
047.110	315 341 054	1565	IMM	CALL	EBB	EVALUATE TO 8 BITS
047.113	106	1566		MOV	B,M	
047.114	303 354 043	1567		JMP	ASMB	GENERATE 2 BYTES

		1569	**	THR - THREE BYTE OPCODES.		
		1570	*			
		1571	*	OPC	EXPR	
		1572	*			
		1573	*	JMP, CALL, LHLD, SHLD, LDA, STA		
		1574				
		1575				
047.117	315 215 051	1576	THR	CALL	EVL	
047.122	126	1577		MOV	D,M	
047.123	303 364 043	1578		JMP	ASMB	GENERATE 3 BYTES

		1580	**	RAO - REGISTER ARITHMETIC, TYPE 1.		
		1581	*			
		1582	*	REGISTER SPECIFIED IN LOW 3 BITS.		
		1583				
		1584				
047.126	315 225 054	1585	RAO	CALL	DRS	DECODE REGISTER SPECIFICATION
047.131	276	1586		DB	*DRSA	GROUP 1
047.132	303 000 044	1587		JMP	ASMB.	GENERATE 1 BYTE

		1589	**	RAT - REGISTER ARITHMETIC, TYPE 2		
		1590	*			
		1591	*	REGISTER SPECIFICATION IN MID 3 BITS		
		1592				
		1593				
047.135	315 225 054	1594	RAT	CALL	DRS	DECODE REGISTER SPECIFICATION
047.140	276	1595		DB	*DRSA	GROUP 1
047.141	007	1596		RLC		
047.142	007	1597		RLC		

047.143	007	1598	RLC			
047.144	303 000 044	1599	JMP	ASMB.	GENERATE 1 BYTE	
		1601	**	RPO - REGISTER PAIR, GROUP 1		
		1602	*			
		1603	*	B=1, D=0, H=5, S=7		
		1604				
		1605				
047.147	315 225 054	1606	RPO	CALL DRS	DECODE REGISTER SPECIFICATION	
047.152	317	1607	DB	#DRSB	GROUP 2	
047.153	303 000 044	1608	JMP	ASMB.	GENERATE 1 BYTE	
		1610	**	RPT - REGISTER PAIR GROUP 2		
		1611	*			
		1612	*	PUSH, POP		
		1613	*			
		1614	*	B=0, D=2, H=4, P=6		
		1615				
		1616				
047.156	315 225 054	1617	RPT	CALL DRS	DECODE REGISTER SPECIFICATION	
047.161	330	1618	DB	#DRSC	GROUP 3	
047.162	303 000 044	1619	JMP	ASMB.	GENERATE 1 BYTE	
		1621	**	INX - PROCESS INX INSTRUCTION.		
		1622	*			
		1623				
		1624				
047.165	315 173 047	1625	INX	CALL INX1	DECODE REGISTER SPECIFICATION	
047.170	303 000 044	1626	JMP	ASMB.	RETURN WITH CODE	
		1627				
		1628				
		1629	**	INX1 - B=00, D=20, H=40, S=60		
		1630				
047.173	315 225 054	1631	INX1	CALL DRS		
047.176	317	1632	DB	#DRSB	GROUP 2	
047.177	075	1633	DCR	A		
047.200	027	1634	RAL			
047.201	027	1635	RAL			
047.202	027	1636	RAL			
047.203	346 070	1637	ANI	0700		
047.205	311	1638	RET			

```
1640 **      MVI - PROCESS MVI INSTRUCTIIN.
1641 *
1642 *      MVI      REG,VAL
1643
1644
047.206 315 225 054 1645 MVI      CALL      DRS      DECODE REG SPEC
047.211 276      1646      DB      #DRSA      GROUP 1
047.212 007      1647      RLC
047.213 007      1648      RLC
047.214 007      1649      RLC
047.215 200      1650      ADD      B
047.216 147      1651      MOV      H,A      (H) = OPCODE BYTE
047.217 315 066 053 1652      CALL      CMA      REQUIRE COMA
047.222 315 341 054 1653      CALL      EBB      EVALUTE TO 8 BITS
047.225 104      1654      MOV      B,H
047.226 303 354 043 1655      JMP      ASM6      OUTPUT 2 BYTES
```

```
1657 **      INDX - PROCESS LDAX, STAX INSTRUCTIONS
1658 *
1659 *      LDAX      B
1660 *      LDAX      D
1661 *      STAX      B
1662 *      STAX      D
1663
1664
047.231 032      1665 INDX      LDAX      D
047.232 376 102 1666      CPI      'B'
047.234 312 252 047 1667      JE      INDX1      IS 'B'
047.237 376 104 1668      CPI      'D'
047.241 076 020 1669      MVI      A,200
047.243 312 000 044 1670      JE      ASM8      OUTPUT 1 BYTE
047.246 315 003 057 1671      CALL      SEF      ERROR
047.251 002      1672      DB      ERR,R      BAD RESIGER SPECIFIED
047.252 170      1673 INDX1      MOV      A,B
047.253 303 001 044 1674      JMP      ASM8      OUTPUT 1 BYTE
```

```
1676 **      RST - RESTART INSTRUCTION
1677 *
1678 *      RST      EXPR
1679 *
1680 *      EXPR MUST BE 0-7
1681
1682
047.256 315 341 054 1683 RST      CALL      EBB      EVALUATE TO 8 BITS
047.261 171      1684      MOV      A,C
047.262 346 370 1685      ANI      3700
047.264 312 273 047 1686      JZ      RST1      IF OK
047.267 315 003 057 1687      CALL      SEF
047.272 020      1688      DB      ERR,V
047.273 171      1689 RST1      MOV      A,C
```

047.274	007	1690	RLC		
047.275	007	1691	RLC		
047.276	007	1692	RLC		
047.277	206	1693	ADD	M	
047.300	303 001 044	1694	JMP	ASMB	OUTPUT 1

1696 ** LXI - PROCESS LXI INSTRUCTION.

1697 *

1698 * LXI REG,EXPR

1699

1700

047.303	315 173 047	1701	LXI	CALL	INX1	DECODE SPECIFICATION
047.306	200	1702		ADD	B	
047.307	147	1703		MOV	H,A	(H) = OPCODE
047.310	315 066 053	1704		CALL	CMA	GOBBLE COMMA
047.313	315 215 051	1705		CALL	EVL	EVALUATE EXPRESSION
047.316	124	1706		MOV	D,H	(D) = 3RD BYTE
047.317	303 364 043	1707		JMP	ASM7	OUTPUT 3 BYTES

1709 ** MOV - PROCESS MOV INSTRUCTION.

1710 *

1711 * MOV REG,REG

1712

1713

047.322	315 225 054	1714	MOV	CALL	DRS	
047.325	276	1715		DB	#DRSA	GROUP 1
047.326	007	1716		RLC		
047.327	007	1717		RLC		
047.330	007	1718		RLC		
047.331	260	1719		ORA	B	
047.332	107	1720		MOV	B,A	(B) = OPCODE AND 1ST REG
047.333	315 066 053	1721		CALL	CMA	READ :
047.336	315 225 054	1722		CALL	DRS	
047.341	276	1723		DB	#DRSA	GROUP 1
047.342	200	1724		ADD	B	
047.343	303 001 044	1725		JMP	ASMB	SINGLE BYTE

```

1729 **      DNT - DECODE NEXT TOKEN.
1730 *
1731 *      DNT IS CALLED TO DECODE THE NEXT TOKEN.
1732 *
1733 *      IF TOKEN = OPERATOR, (L) = INDEX
1734 *          =0 +
1735 *          =1 -
1736 *          =2 *
1737 *          =3 /
1738 *
1739 *      IF TOKEN = SYMBOL, (BC) = VALUE
1740 *
1741 *      DNT EXITS THROUGH A BRANCH TABLE.
1742 *
1743 *      CALL      DNT
1744 *      DB      ADRA-*      IF +
1745 *      DB      ADRB-*      IF -
1746 *      DB      ADRC-*      IF *
1747 *      DB      ADRD-*      IF /
1748 *      DB      ADRE-*      IF SYMBOL
1749 *      DB      ADRF-*      IF END OF EXPR
1750 *
1751 *      ENTRY    (DE) = EXPRESSION POINTER
1752 *      EXIT      (BC) = VALUE IF SYMBOL
1753 *              (L) = INDEX IF OPERATOR
1754 *              TOKREL = ST.REL IF RELOCATABLE VALUE
1755 *      USES      ALL
1756 *
1757 *
1758 DNT EQU *
1759 LXI H,$TBRA
1760 PUSH H          SET $TBRA EXIT VIA *RET*
1761 XRA A
1762 STA TOKREL      CLEAR RELOCATION FLAG
1763 LDAX D
1764 CPI QUOTE
1765 JNE DNT2        NOT QUOTE
1766
1767 *      HAVE 'C' OR 'CC'
1768
1769 INX D
1770 CALL GSC        GET STRING CHARACTER
1771 JZ DNT13       NULL STRING ILLEGAL
1772 MOV C,A
1773 MVI B,0        ASSUME ONE CHARACTER
1774 CALL GSC        GET STRING CHARACTER
1775 JZ DNT1        ONLY 1 CHARACTER
1776 MOV B,C
1777 MOV C,A
1778 CALL GSC        GET STRING CHARACTER
1779 JNZ DNT13      TOO MANY CHARACTERS
1780 DNT1 MVI A,4
1781 RET            RETURN VIA $TBRA
1782
1783 *      HAVE OPERATOR OR SYMBOL OR NULL
1784

```


050.017	315 031 053	1785	DNT2	CALL	CEF	CHECK FOR END OF FIELD
050.022	076 005	1786		MVI	A,5	
050.024	310	1787		RZ		IF END OF FIELD
050.025	032	1788		LDAX	D	
050.026	376 054	1789		CPI	'','	
050.030	076 005	1790		MVI	A,5	
050.032	310	1791		RE		IF '','', FLAG AS END OF EXPRESSION
050.033	315 027 051	1792		CALL	LCT	LOOKUP CHARACTER
050.036	157	1793		MOV	L,A	
050.037	007	1794		RLC		
050.040	332 061 050	1795		JC	DNT3	IS SYMBOL
050.043	007	1796		RLC		
050.044	332 163 050	1797		JC	DNT6	IS NUMBER
050.047	007	1798		RLC		
050.050	322 353 050	1799		JNC	DNT13	ERROR
		1800				
		1801	*	HAVE	OPERATOR	
		1802				
050.053	175	1803		MOV	A,L	
050.054	346 003	1804		ANI	3	
050.056	157	1805		MOV	L,A	
050.057	023	1806		INX	D	
050.060	311	1807		RET		EXIT VIA \$TBRA
		1808				
		1809	*	HAVE	SYMBOL. BUILD IT UP	
		1810				
050.061	041 004 051	1811	DNT3	LXI	H,DNTA	(HL) = WORKAREA POINTER
050.064	006 007	1812		MVI	B,7	7 CHAR MAX
050.066	345	1813		PUSH	H	
050.067	053	1814		DCX	H	
050.070	315 027 051	1815	DNT4	CALL	LCT	LOOKUP CHARACTER TYPE
050.073	346 300	1816		ANI	3000	
050.075	312 110 050	1817		JZ	DNT5	NOT ALPHANUMERIC
050.100	032	1818		LDAX	D	
050.101	043	1819		INX	H	
050.102	167	1820		MOV	M,A	
050.103	023	1821		INX	D	
050.104	005	1822		DCR	B	
050.105	302 070 050	1823		JNZ	DNT4	IF MORE TO COPY
		1824				
		1825	*	HAVE	SYMBOL. LOOKUP VALUE	
		1826				
050.110	176	1827	DNT5	MOV	A,M	SET SIGN ON LAST CHARACTER
050.111	366 200	1828		ORI	2000	
050.113	167	1829		MOV	M,A	
050.114	353	1830		XCHG		
050.115	343	1831		XTHL		SAVE DE, (HL) = DNTA
050.116	353	1832		XCHG		
050.117	315 016 057	1833		CALL	SST	SEARCH SYMBOL TABLE
050.122	176	1834		MOV	A,M	(A) = TYPE
050.123	043	1835		INX	H	
000.000		1836		ERRNZ	ST.UND	CODE ASSUMES = 0
050.124	247	1837		ANA	A	
050.125	302 135 050	1838		JNZ	DNT5.5	DEFINED
050.130	315 003 057	1839		CALL	SEF	*U* ERROR
050.133	001	1840		DB	ERR.U	

DNT

15:19:31 16-MAY-80

050.134	257	1841	XRA	A	CLEAR FLAG
050.135	365	1842	DNT5.5	PUSH	PSW
050.136	346 100	1843	ANI	ST.REL	SAVE CODE
050.140	062 354 067	1844	STA	TOKREL	SET RELOCATABLE FLAG
050.143	361	1845	POP	PSW	(A) = FLAG BITS
050.144	027	1846	RAL		
050.145	322 154 050	1847	JNC	DNT5.7	NOT REFERENCE TO DOUBLE DEFINED
050.150	315 003 057	1848	CALL	SEF	FLAG ** FOR DOUBLE REFERENCE
050.153	200	1849	DB	ERR.P	
050.154		1850	DNT5.7	EQU	*
050.154	116	1851	MOV	C,M	
050.155	043	1852	INX	H	
050.156	106	1853	MOV	B,M	(BC) = VALUE
050.157	321	1854	POP	D	RESTORE (DE)
050.160	076 004	1855	MVI	A,4	
050.162	311	1856	RET	\$TBRA	EXIT VAI \$TBRA
		1857			
		1858	*	HAVE NUMBER	
		1859			
050.163	041 003 051	1860	DNT6	LXI	H,DNTA-1
050.166	006 022	1861	MVI	B,18	18 DIGITS MAX
050.170	315 027 051	1862	DNT7	CALL	LCT
050.173	346 120	1863	ANI	1200	LOOKUP TYPE
050.175	312 210 050	1864	JZ	DNT8	SEE IF NUMBER OR POSTRADIX
050.200	032	1865	LDAX	D	OUT OF NUMBER
050.201	043	1866	INX	H	
050.202	167	1867	MOV	M,A	COPY TO WORK AREA
050.203	023	1868	INX	D	
050.204	005	1869	DCR	B	
050.205	302 170 050	1870	JNZ	DNT7	
		1871			
		1872	*	HAVE ACCUMULATED NUMBER. SEE IF HAS POSTRADIX.	
		1873			
050.210		1874	DNT8	EQU	*
050.210	257	1875	XRA	A	
050.211	062 332 050	1876	STA	DNTD	FLAG NO OVERFLOW
050.214	176	1877	MOV	A,M	
050.215	062 333 050	1878	STA	DNTC	SAVE POSTRADIX
050.220	315 034 051	1879	CALL	LCT.	LOOKUP CHARACTER TYPE
050.223	346 020	1880	ANI	200	
050.225	302 233 050	1881	JNZ	DNT9	HAS POSTRADIX
050.230	043	1882	INX	H	
050.231	066 104	1883	MVI	M, 'D'	
050.232		1884	DNTB	EQU	*-1
		1885			DEFAULT POSTRADIX
		1886	*	COMPUTE BASE	
		1887			
050.233	176	1888	DNT9	MOV	A,M
050.234	066 200	1889	MVI	M,2000	(A) = POSTRADIX
050.236	315 034 051	1890	CALL	LCT.	FLAG END OF NUMBER
050.241	346 017	1891	ANI	170	LOOKUP CHARACTER TYPE
050.243	074	1892	INR	A	(A) = POSTRADIX
050.244	325	1893	PUSH	D	SAVE EXPRESSION POINTER
050.245	137	1894	MOV	E,A	
050.246	026 000	1895	MVI	D,0	(DE) = BASE
		1896			

			1897	*	DECODE NUMBER.	
			1898			
050.250	041 004 051	1899		LXI	H,DNTA	
050.253	001 000 000	1900		LXI	B,0	PRESET ACCUMULATOR TO 0
050.256	176	1901	DNT10	MOV	A,H	
050.257	247	1902		ANA	A	
050.260	372 330 050	1903		JH	DNT11	ALL DONE
050.263	315 232 053	1904		CALL	DHD	DECODE HEX DIGITS
050.266	332 352 050	1905		JC	DNT12	ERROR
050.271	043	1906		INX	H	
050.272	345	1907		PUSH	H	
050.273	325	1908		PUSH	D	
050.274	365	1909		PUSH	PSW	
050.275	315 337 030	1910		CALL	\$MU66	ACCUM = ACCUM*BASE
050.300	345	1911		PUSH	H	
050.301	041 332 050	1912		LXI	H,DNTD	ACCUMULATE OVERFLOW FLAGS
050.304	206	1913		ADD	H	
050.305	167	1914		MOV	M,A	
050.306	341	1915		POP	H	
050.307	361	1916		POP	PSW	
050.310	117	1917		MOV	C,A	
050.311	006 000	1918		MVI	B,0	(BC) = DIGIT VALUE
050.313	011	1919		DAD	B	(HL) = ACCUM*BASE + DIGIT
050.314	321	1920		POP	D	
050.315	171	1921		MOV	A,C	
050.316	273	1922		CMP	E	COMPARE DIGIT TO BASE
050.317	104	1923		MOV	B,H	
050.320	115	1924		MOV	C,L	
050.321	341	1925		POP	H	
050.322	322 352 050	1926		JNC	DNT12	ERROR
050.325	303 256 050	1927		JMP	DNT10	
		1928				
050.330	321	1929	DNT11	POP	D	NUMBER ACCUMULATED OK
050.331	041 000 000	1930		LXI	H,0	(H) = POSTRADIX, (L) = OVERFLOW
050.332		1931	DNTD	EQU	*-2	OVERFLOW FLAG
050.333		1932	DNTC	EQU	*-1	POSTRADIX
050.334	076 101	1933		MVI	A,'A'	
050.336	274	1934		CMP	H	
050.337	314 362 050	1935		CE	DNT14	IS 'A' POSTRADIX
050.342	175	1936		MOV	A,L	
050.343	247	1937		ANA	A	
050.344	304 377 050	1938		CNZ	DNT15	IS OVERFLOW
		1939				
050.347	076 004	1940		MVI	A,4	
050.351	311	1941		RET	\$TBRA	EXIT VIA \$TBRA
		1942				
050.352	321	1943	DNT12	POP	D	ERROR WHILE CRACKING NUMBER
		1944				
		1945	*		ERROR DETECTED	
		1946				
050.353	315 003 057	1947	DNT13	CALL	SEF	** ERROR
050.356	010	1948		DB	ERR,A	
050.357	076 005	1949		MVI	A,5	SET TYPE = NULL
050.361	311	1950		RET	\$TBRA	EXIT THROUGH \$TBRA
		1951				
050.362	175	1952	DNT14	MOV	A,L	

```

050.363 037 1953 RAR
050.364 170 1954 MOV A,B SHIFT HIGH BYTE RIGHT WITH CARRY
050.365 037 1955 RAR
050.366 107 1956 MOV B,A
050.367 334 377 050 1957 CC DNT15 BAD DIGIT
050.372 175 1958 MOV A,L (A) = OVERFLOW REGISTER
050.373 346 376 1959 ANI 3760 CLEAR ALLOWED OVERFLOW
050.375 157 1960 MOV L,A CLEAR SINGLE CARRY
050.376 311 1961 RET
1962
050.377 315 003 057 1963 DNT15 CALL SEF FLAG OVERFLOW
051.002 020 1964 DB ERR,V
051.003 311 1965 RET
1966
051.004 1967
1968 DNTA DS 19 WORK AREA

```

```

1970 ** LCT - LOOKUP CHARACTER TYPE.
1971 *
1972 * LCT LOOKS UP THE CHARACTER TYPE INDEX FOR A CHARACTER.
1973 *
1974 * ENTRY (DE) = STRING POINTER
1975 * EXIT (A) = INDEX
1976 * 1000 VALID ALPHA
1977 * 0100 VALID NUMBER
1978 * 0010 VALID OPERATOR
1979 * 0001 VALID POSTRADIX
1980 * NNNN OPCODE INDEX IF OPERATOR, BASE IF POSTRADIX
1981 * USES A,F
1982 *
1983
000.000 1984 ERNZ CT.ALPH-2000 /80.02.GC/
1985
051.027 032 1986 LCT LDAX D
051.030 247 1987 ANA A
051.031 372 053 051 1988 JM LCT1
051.034 1989 LCT, EQU * ENTRY WITH (A) = CHARACTER
051.034 326 040 1990 SUI 7
051.036 332 053 051 1991 JC LCT1 TOO SMALL
051.041 345 1992 PUSH H SAVE (HL)
051.042 041 055 051 1993 LXI H,LCTA
051.045 315 101 030 1994 CALL $DATA.
051.050 176 1995 MOV A,M (A) = FLAG BYTE
051.051 341 1996 POP H
051.052 311 1997 RET
1998
051.053 257 1999 LCT1 XRA A END OF LINE
051.054 311 2000 RET
2001
051.055 2002 LCTA EQU * CHARACTER TABLE
2003
051.055 000 2004 DB 00000000B BLANK
051.056 000 2005 DB 00000000B !

```

051.057	000	2006	DB	00000000B	*
051.060	000	2007	DB	00000000B	#
051.061	200	2008	DB	10000000B	\$
051.062	000	2009	DB	00000000B	PERCENT
051.063	000	2010	DB	00000000B	%
051.064	000	2011	DB	00000000B	'
051.065	000	2012	DB	00000000B	(
051.066	000	2013	DB	00000000B)
051.067	042	2014	DB	00100010B	*
051.070	040	2015	DB	00100000B	+
051.071	000	2016	DB	00000000B	,
051.072	041	2017	DB	00100001B	-
051.073	200	2018	DB	10000000B	.
051.074	043	2019	DB	00100011B	/
051.075	100	2020	DB	01000000B	0
051.076	100	2021	DB	01000000B	1
051.077	100	2022	DB	01000000B	2
051.100	100	2023	DB	01000000B	3
051.101	100	2024	DB	01000000B	4
051.102	100	2025	DB	01000000B	5
051.103	100	2026	DB	01000000B	6
051.104	100	2027	DB	01000000B	7
051.105	100	2028	DB	01000000B	8
051.106	100	2029	DB	01000000B	9
051.107	000	2030	DB	00000000B	:
051.110	000	2031	DB	00000000B	;
051.111	000	2032	DB	00000000B	<
051.112	000	2033	DB	00000000B	=
051.113	000	2034	DB	00000000B	>
051.114	000	2035	DB	00000000B	?
051.115	000	2036	DB	00000000B	@
051.116	327	2037	DB	11010111B	A
051.117	321	2038	DB	11010001B	B
051.120	300	2039	DB	11000000B	C
051.121	331	2040	DB	11011001B	D
051.122	300	2041	DB	11000000B	E
051.123	300	2042	DB	11000000B	F
051.124	200	2043	DB	10000000B	G
051.125	237	2044	DB	10011111B	H
051.126	200	2045	DB	10000000B	I
051.127	200	2046	DB	10000000B	J
051.130	200	2047	DB	10000000B	K
051.131	200	2048	DB	10000000B	L
051.132	200	2049	DB	10000000B	M
051.133	200	2050	DB	10000000B	N
051.134	227	2051	DB	10010111B	O
051.135	200	2052	DB	10000000B	P
051.136	227	2053	DB	10010111B	Q
051.137	200	2054	DB	10000000B	R
051.140	200	2055	DB	10000000B	S
051.141	200	2056	DB	10000000B	T
051.142	200	2057	DB	10000000B	U
051.143	200	2058	DB	10000000B	V
051.144	200	2059	DB	10000000B	W
051.145	200	2060	DB	10000000B	X
051.146	200	2061	DB	10000000B	Y

051.147	200	2062	DE	10000000B	Z
051.150	000	2063	DE	00000000B	[
051.151	000	2064	DE	00000000B	\
051.152	000	2065	DE	00000000B]
051.153	000	2066	DE	00000000B	^
051.154	000	2067	DE	00000000B	_
051.155	000	2068	DE	00000000B	`
051.156	327	2069	DE	11010111B	a
051.157	321	2070	DE	11010001B	b
051.160	300	2071	DE	11000000B	c
051.161	331	2072	DE	11011001B	d
051.162	300	2073	DE	11000000B	e
051.163	300	2074	DE	11000000B	f
051.164	200	2075	DE	10000000B	g
051.165	237	2076	DE	10011111B	h
051.166	200	2077	DE	10000000B	i
051.167	200	2078	DE	10000000B	j
051.170	200	2079	DE	10000000B	k
051.171	200	2080	DE	10000000B	l
051.172	200	2081	DE	10000000B	m
051.173	200	2082	DE	10000000B	n
051.174	227	2083	DE	10010111B	o
051.175	200	2084	DE	10000000B	p
051.176	227	2085	DE	10010111B	q
051.177	200	2086	DE	10000000B	r
051.200	200	2087	DE	10000000B	s
051.201	200	2088	DE	10000000B	t
051.202	200	2089	DE	10000000B	u
051.203	200	2090	DE	10000000B	v
051.204	200	2091	DE	10000000B	w
051.205	200	2092	DE	10000000B	x
051.206	200	2093	DE	10000000B	y
051.207	200	2094	DE	10000000B	z
051.210	000	2095	DE	00000000B	{
051.211	000	2096	DE	00000000B	
051.212	000	2097	DE	00000000B	}
051.213	000	2098	DE	00000000B	~
051.214	000	2099	DE	00000000B	DEL

```

2101 **      EVL - EVALUATE OPERAND EXPRESSION.
2102 *
2103 *      EVL EVALUATES AN OPERAND EXPRESSION. IT IS PROCESSED
2104 *      LEFT TO RIGHT, WITH NO OPERATOR PRECEDENCE, AND NO PARANETHSIS.
2105 *
2106 *      VALID OPERATORS
2107 *      +
2108 *      -
2109 *      *
2110 *      /
2111 *
2112 *      VALID SYMBOLS
2113 *
2114 *      LABEL

```

```
2115 *      *      LOCATION COUNTER
2116 *      'C'      8 BIT ASCII
2117 *      'CC'     16 BIT ASCII
2118 *      NNN      NUMBER, POSTRADIX =
2119 *      Q OCTAL
2120 *      D DECIMAL
2121 *      B BINARY
2122 *      H HEX
2123 *
2124 *      IF PASS1, UNDEFINED ERROR FLAGS WILL BE IGNORED.
2125 *
2126 *      ENTRY      (DE) = STRING POINTER
2127 *      EXIT       (BC) = VALUE
2128 *      (DE) UPDATED
2129 *      EXPREL = ST.REL IF RELOCATABLE
2130 *      'C' SET IF ERROR
2131 *      USES      A,F,B,C,D,E
2132
2133
051.215 345 2134 EVL PUSH H SAVE (HL)
051.216 001 000 000 2135 LXI B,0
051.221 257 2136 XRA A
051.222 062 353 067 2137 STA EXPREL CLEAR RELOCATABLE FLAG
051.225 305 2138 PUSH B SAVE ACCUMULATOR ON STACK
051.226 032 2139 LDAX D
051.227 376 043 2140 CPI '*'
051.231 076 377 2141 MVI A,3770 ASSUME NO *
051.233 302 240 051 2142 JNE EVL1 NO *
051.236 074 2143 INR A (A) = 0
051.237 023 2144 INX D SKIP *
051.240 062 342 051 2145 EVL1 STA EVLA SET MASK FOR RESULT
2146
2147 *      HAVE NULL
2148
051.243 315 346 047 2149 CALL DNT DECODE NEXT TOKEN
051.246 032 2150 DB EVL5-* + - UNARY +
051.247 031 2151 DB EVL5-* - - UNARY -
051.250 004 2152 DB EVL2-* * - ORG
051.251 061 2153 DB EVL8-* / - ERROR
051.252 005 2154 DB EVL3-* VAL - VALUE
051.253 057 2155 DB EVL8-* NUL - ERROR
2156
051.254 315 054 052 2157 EVL2 CALL EVL20 (BC) = (ORG)
051.257 341 2158 EVL3 POP H DISCARD INITIAL VALUE
051.260 305 2159 PUSH B SET INITIAL VALUE = ORG
051.261 072 354 067 2160 LIA TOKREL
051.264 062 353 067 2161 STA EXPREL SET RELOCATABILITY OF EXPRESSION
2162
2163 *      HAVE VALUE.
2164
051.267 315 346 047 2165 EVL4 CALL DNT DECODE NEXT TOKEN
051.272 006 2166 DB EVL5-* +
051.273 005 2167 DB EVL5-* -
051.274 004 2168 DB EVL5-* *
051.275 003 2169 DB EVL5-* /
051.276 034 2170 DB EVL8-* VAL - ERROR
```

```

051.277 037      2171      DB      EVL9-*      NUL - DONE
                  2172
                  2173 *      HAVE OPERATOR
                  2174
051.300 345      2175 EVL5    PUSH    H      SAVE OPERATOR INDEX
051.301 315 346 047 2176      CALL    DNT    DECODE NEXT TOKEN
051.304 025      2177      DB      EVL7.5-*    + - ERROR
051.305 024      2178      DB      EVL7.5-*    - - ERROR
051.306 004      2179      DB      EVL6-*      * - ORG
051.307 022      2180      DB      EVL7.5-*    / - ERROR
051.310 005      2181      DB      EVL7-*      VAL - DO OPEARTION
051.311 020      2182      DB      EVL7.5-*    NUL - ERROR
                  2183
051.312 315 054 052 2184 EVL6    CALL    EVL20    (BC) = (ORG)
051.315 341      2185 EVL7    POP     H
051.316 175      2186      MOV     A,L      (A) = OPERATOR INDEX
051.317 341      2187      POP     H      (HL) = OLD VALUE, (BC) = NEW
051.320 325      2188      PUSH    D
051.321 315 345 051 2189      CALL    EVL10    PERFORM OPERATION
051.324 321      2190      POP     D
051.325 345      2191      PUSH    H      SAVE RESULT
051.326 303 267 051 2192      JMP     EVL4
                  2193
                  2194 *      ERROR
                  2195
051.331 361      2196 EVL7.5  POP     PSW      CLEAN STACK
051.332 315 003 057 2197 EVL8    CALL    SEF      SET ERROR FLAG
051.335 010      2198      DB      ERR.A
                  2199
                  2200 *      DONE
                  2201
051.336 301      2202 EVL9    POP     B      (BC) = VALUE
051.337 341      2203      POP     H      RESTORE HL
051.340 170      2204      MOV     A,B
051.341 346 000      2205      ANI     0      MASK OFF IF #
051.342      2206 EVLA    EQU     *-1
051.343 107      2207      MOV     B,A
051.344 311      2208      RET
                  2209
                  2210
                  2211 *      PERFORM ARITHMETIC.
                  2212 *
                  2213 *      ENTRY    (L) = OPERATOR INDEX
                  2214 *      (BC) = Y
                  2215 *      (HL) = X
                  2216 *
                  2217 *      EXIT    (HL) = X OP Y
                  2218
051.345      2219 EVL10    EQU     *
051.345 315 076 031 2220      CALL    $TBRA
051.350 004      2221      DB      EVL11-*    +
051.351 022      2222      DB      EVL12-*    -
051.352 045      2223      DB      EVL13-*    *
051.353 053      2224      DB      EVL14-*    /
                  2225
051.354 011      2226 EVL11    DAD     B      +

```


051.355	345	2227	PUSH	H	SAVE SUM
051.356	041 353 067	2228	LXI	H,EXPREL	
051.361	072 354 067	2229	LDA	TOKREL	
051.364	206	2230	ADD	M	SUM RELOCATION FLAGS
000.000		2231	ERRNZ	ST.REL-1000	
051.365	167	2232	MOV	M,A	
051.366	341	2233	POP	H	RESTORE RESULT
051.367	372 047 052	2234	JM	EVL16	REL+REL IS ILLEGAL
051.372	311	2235	RET		
		2236			
051.373	175	2237	EVL12	MOV	A,L
051.374	221	2238	SUB	C	
051.375	157	2239	MOV	L,A	
051.376	174	2240	MOV	A,H	
051.377	230	2241	SBB	B	
052.000	147	2242	MOV	H,A	
052.001	345	2243	PUSH	H	SAVE RESULT
052.002	041 353 067	2244	LXI	H,EXPREL	
052.005	072 354 067	2245	LDA	TOKREL	
052.010	226	2246	SUB	M	
052.011	167	2247	MOV	M,A	STORE RESULT
052.012	341	2248	POP	H	RESTORE RESULT
052.013	332 047 052	2249	JC	EVL16	ABS-REL ILLEGAL
052.016	311	2250	RET		
		2251			
052.017	353	2252	EVL13	XCHG	*
052.020	315 337 030	2253	CALL	\$MU66	
052.023	303 035 052	2254	JMP	EVL15	CHECK FOR RELOCATION ERROR
		2255			
052.026	120	2256	EVL14	MOV	D,B
052.027	131	2257	MOV	E,C	/
052.030	104	2258	MOV	B,H	
052.031	115	2259	MOV	C,L	
052.032	315 106 030	2260	CALL	\$DU66	
052.035	345	2261	EVL15	PUSH	H
052.036	041 353 067	2262	LXI	H,EXPREL	SAVE RESULT
052.041	072 354 067	2263	LDA	TOKREL	
052.044	267	2264	ORA	A	
052.045	341	2265	POP	H	RESTORE RESULT
052.046	310	2266	RZ		ABS 'OP' ABS IS OK
		2267			
		2268	*	RELOCATION ERROR	
		2269			
052.047	315 003 057	2270	EVL16	CALL	SEF
052.052	002	2271	DB	ERR,R	
052.053	311	2272	RET		

			2274	**	EVL20 - USE ORG AS TOKEN VALUE	
			2275	*		
			2276	*	ENTRY	NONE
			2277	*	EXIT	(BC) = ORG
			2278	*		TOKREL SET PROPERLY
			2279	*	USES	A,F,B,C,H,L
			2280			
			2281			
052.054	072	145	067	2282	EVL20	LDA RELFLG
052.057	062	354	067	2283	STA	TOKREL SET FLAG PROPERLY
052.062	052	134	067	2284	LHLD	ORG
052.065	104			2285	MOV	B,H
052.066	115			2286	MOV	C,L (BC) = (ORG)
052.067	311			2287	RET	

SUBROUTINES.

ABV

15:19:43 16-MAY-80

```

2291 **      ABV - ACCUMULATE BYTE VALUE.
2292 *
2293 *      ABV ADDS A BYTE TO THE BINARY BUFFER.
2294 *
2295 *      THE ORG UPON ENTRY IS THE ADDRESS+1 OF THE BYTE
2296 *
2297 *      ENTRY (A) = VALUE
2298 *      EXIT  NONE
2299 *      USES  NONE
2300
2301
2302 ABV      CALL  $SAVALL      SAVE REGS
2303          MOV   B,A          (B) = VALUE
2304          LDA   BINFNAM
2305          ANA   A
2306          JZ    $RSTALL      NO BINARY FILE
2307          LDA   PASS
2308          CPI   2
2309          JNE   ABV2         NOT PASS 2
2310          PUSH  B            SAVE VALUE
2311          LHLD  ABSFWA
2312          XCHG              (DE) = ORG OF FIRST BINARY BYTE
2313          LHLD  ORG
2314          DCX   H            (HL) = REAL ORG
2315          MOV   A,L
2316          SUB   E
2317          MOV   L,A
2318          MOV   A,H
2319          SBB   D
2320          MOV   H,A          (HL) = INDEX OF BYTE IN BINARY FILE
2321          LDA   BINSKW
2322          CALL  $DADA,        (HL) = NUMBER OF BYTE IN BINARY FILE
2323          LDA   BINCSEN
2324          CMP   H
2325          JE    ABV1         CAN GO IN THIS SECTOR
2326
2327 *      WILL NOT GO IN THIS SECTOR. PUT THIS SECTOR BACK, GET
2328 *      THE PROPER ONE.
2329
2330          CALL  WBB          WRITE BINARY BUFFER
2331          MOV   A,H
2332          STA   BINCSEN      SET CURRENT SECTOR NUMBER
2333
2334 *      NOW READ THE NEW SECTOR INTO THE BUFFER AREA. IF IT DOES NOT
2335 *      YET EXIST, WRITE ENOUGH GARBAGE UNTIL IT DOES.
2336
2337          PUSH  H
2338          MOV   C,H          (C) = SECTOR NUMBER
2339          MVI   B,0
2340          MVI   A,CN.BIN
2341          DB    SYSCALL,POSIT POSITION TO WHERE WE WANT
2342          JNC   ABV0         GOT THERE
2343          MOV   B,C          (B) = SECTORS TO WRITE
2344          MVI   C,0
2345          LXI   D,4096        POINT TO GARBAGE (MOSTLY 0, I THINK..)
2346          MVI   A,CN.BIN

```

052,201	377	005	2347	DB	SYSCALL,,WRITE	WRITE IT
052,203	332	334 060	2348	JC	BINERR	ERROR
052,206	001	000 001	2349	ABV0 LXI	B,256	
052,211	021	237 070	2350	LXI	D,BINBFR	
052,214	076	000	2351	MVI	A,CN.BIN	
052,216	377	004	2352	DB	SYSCALL,,READ	READ IN NEW SECTOR
052,220	322	230 052	2353	JNC	ABV00	OK
052,223	376	001	2354	CPI	EC.EOF	OK IF SIMPLE EOF
052,225	302	334 060	2355	JNE	BINERR	ERROR
052,230	341		2356	ABV00 POP	H	(L) = INDEX FOR BYTE
			2357			
			2358	*	BYTE WILL GO IN THIS SECTOR	
			2359			
052,231	021	237 070	2360	ABV1 LXI	D,BINBFR	
052,234	046	000	2361	MVI	H,0	
052,236	031		2362	DAD	D	(HL) = ADDRESS IN BINBFR
052,237	361		2363	POP	PSW	(A) = VALUE
052,240	167		2364	MOV	M,A	SET
052,241	303	047 031	2365	JMP	\$RSTALL	RESTORE AND EXIT
			2366			
			2367	*	IS PASS 1. IF AN ABS FILE, KEEP TRACK OF THE SMALLEST	
			2368	*	AND LARGEST ADDRESS WHICH GOT DATA...	
			2369			
052,244	072	144 067	2370	ABV2 LDA	FTFLAG	
052,247	247		2371	ANA	A	
052,250	302	047 031	2372	JNZ	\$RSTALL	NOT ABS
052,253	052	175 067	2373	LHLD	ABSFVA	
052,256	353		2374	XCHG		
052,257	052	134 067	2375	LHLD	ORG	COMPARE OLD LOWEST TO NOW
052,262	053		2376	DCX	H	(HL) = ORG FOR THIS BYTE
052,263	175		2377	MOV	A,L	
052,264	223		2378	SUB	E	
052,265	174		2379	MOV	A,H	
052,266	232		2380	SBB	D	
052,267	322	275 052	2381	JNC	ABV3	NOT NEW LOW
052,272	042	175 067	2382	SHLD	ABSFVA	NEW LOW
052,275	353		2383	ABV3 XCHG		
052,276	052	203 067	2384	LHLD	ABSLVA	
052,301	353		2385	XCHG		
052,302	173		2386	MOV	A,E	SEE IF NEW HIGH
052,303	225		2387	SUB	L	
052,304	172		2388	MOV	A,D	
052,305	234		2389	SBB	H	
052,306	322	047 031	2390	JNC	\$RSTALL	NOT NEW HIGH
052,311	042	203 067	2391	SHLD	ABSLVA	SET IT
052,314	303	047 031	2392	JMP	\$RSTALL	RESTORE AND RETURN

```
2394 **      BDT - BUILD DYNAMIC TABLES.
2395 *
2396 *      BDT INITIALIZES THE SYMBOL TABLE AND THE RELOCATION TABLE.
2397 *
2398 *      THE SYMBOL TABLE STARTS AT THE FIRST AVAILABLE ADDRESS,
2399 *      AND CONTINUES UP TO THE END OF THE RELOCATION TABLE.
2400 *
2401 *      THE RELOCATION TABLE STARTS IN HIGH MEMORY, AND GOES DOWN.
2402 *
2403 *      THIS ROUTINE IS USED ONE TIME ONLY, BUT CANNOT BE ENCLOSED
2404 *      WITH THE OVERLAID CODE, IN THAT THIS ROUTINE ZAPS THAT OVERLAID AREA.
2405 *
2406 *      ENTRY  NONE
2407 *      EXIT   'C' CLEAR IF OK
2408 *            'C' SET IF ERROR, ERROR MESSAGE PRINTED
2409 *      USES   ALL
2410
2411
2412 BDT      LHLD  S,SYSH      (HL) = FWA SYSTEM
2413          LDA   LARGE
2414          ANA   A
2415          JNZ  BDT1      WILL USEE ALL WE CAN
2416          XCHG
2417          LHLD  S,OMAX
2418          CALL  $CHL
2419          DAD   D      (HL) = AMOUNT WHICH WILL NOT CAUSE OVERLAY SWAPING
2420 BDT1     LXI   D,-12
2421          DAD   D
2422          XCHG      (DE) = LIMIT FOR REL TABLE
2423          LXI   H,-SYMTAB-256
2424          DAD   D
2425          JNC  BDT4      NOT AT LEAST 256 BYTES
2426          XCHG      (HL) = REL LIMIT
2427          SHLD  RELLWA      SET LIMIT FOR REL TABLE
2428          SHLD  RELPTR      SET REL TABLE EMPTY
2429          DB    SYSCALL,.SETTP  REQUEST IT
2430          JNC  BDT2      OK
2431          MVI   H,BELL
2432          DB    SYSCALL,.ERROR      PROBLEMS
2433          STC      FLAG ERROR
2434          RET
2435
2436 BDT2     LHLD  SYMFWA
2437          XCHG
2438          LHLD  RELLWA
2439 BDT3     DCX   H
2440          MVI   M,0      CLEAR TABLE AREA
2441          CALL  $CDEHL
2442          JNE  BDT3      MORE TO GO
2443          ANA   A
2444          RET      RETURN WITH 'C' CLEAR
2445
2446 *      NOT AT LEAST 256 BYTES IN SYMTAB. FORCE /LARGE
2447
2448 BDT4     MVI   A,1
2449          STA   LARGE      SET LARGE
```

053.026 303 317 052 2450 JMP BDT TRY AGAIN

2452 ** CEF - CHECK FOR END OF FIELD CHARACTER.
2453 *
2454 * CEF CHECKS A CHARACTER TO SEE IF IT IS A
2455 *
2456 * 00, BLANK, OR TAB
2457 *
2458 * ENTRY (A) = CHARACTER
2459 * EXIT 'Z' SET IF 00, BLANK OR TAB
2460 * 'Z' CLEAR OTHERWISE
2461 * USES F
2462
2463
053.031 247 2464 CEF ANA A
053.032 310 2465 RZ 00
053.033 376 011 2466 CPI TAB
053.035 310 2467 RE TAB
053.036 376 040 2468 CPI
053.040 311 2469 RET RETURN WITH CODE

2471 ** CLE - CHECK LISTING ELIGIBILITY.
2472 *
2473 * CLE IS CALLED TO SEE IF THE CURRENT LINE SHOULD BE LISTED TO
2474 * THE OUTPUT FILE.
2475 *
2476 * IF LST.L = FALSE, DONT LIST
2477 * IF XTEXT LINE AND LST.C = FALSE, DONT LIST
2478 *
2479 * ENTRY NONE
2480 * EXIT 'Z' CLEAR IFF TO LIST
2481 * USES A,F,H,L
2482
2483
053.041 041 130 067 2484 CLE LXI H,LSTCTL
053.044 176 2485 MOV A,M
053.045 346 001 2486 ANI LST.L
053.047 310 2487 RZ DONT LIST
053.050 072 143 067 2488 LDA XTXLINE
053.053 247 2489 ANA A
053.054 302 062 053 2490 JNZ CLE1 IS XTEXT
053.057 366 001 2491 ORI 1 LIST
053.061 311 2492 RET
2493
053.062 176 2494 CLE1 MOV A,M
053.063 346 004 2495 ANI LST.C
053.065 311 2496 RET

```
2498 **      CMA - READ COMMA.
2499 *
2500 *      CMA IS CALLED WHEN A COMMA IS EXPECTED TO APPEAR IN THE
2501 *      EXPRESSION. IT IS CHECKED, AND ADVANCED OVER.
2502 *
2503 *      ENTRY (DE) = LINE POINTER
2504 *      EXIT (DE) ADVANCED
2505 *      USES A,F,D,E
2506
2507
2508 CMA      LDAX D
2509          INX D
2510          CPI ','
2511          RE
2512          CALL SEF          OK
2513          DB ERR,A          ** ERROR
2514          RET

2516 **      COL - COUNT OUTPUT LINES.
2517 *
2518 *      COL IS CALLED TO COUNT AN OUTPUT LINE BEFORE IT IS WRITTEN.
2519 *
2520 *      ENTRY NONE
2521 *      EXIT NONE
2522 *      USES A,F
2523
2524
2525 COL      PUSH H          SAVE (HL)
2526          PUSH D
2527          PUSH B          SAVE REGISTERS
2528
2529          LXI H,EJEFLG
2530          MOV A,M          (A) = EJEFLG
2531          ANA A
2532          MVI M,0          CLEAR FLAG
2533          ERNZ LINCNT-EJEFLG-1
2534          INX H
2535          MOV A,M          (A) = LINCNT
2536          JNZ COL1          EJEFLG < 0
2537          ANA A
2538          JNZ COL3          NOT TIME YET
2539
2540 *      FORCE NEW PAGE.
2541
2542 COL1     PUSH H          /78.10.GC/
2543          CALL FNF          FORCE NEW PAGE
2544          LXI H,PAGNUM      /78.10.GC/
2545          MOV A,M
2546          ADI 1             DECODE PAGE NUMBER
2547          DAA               INCREMENT
2548          MOV M,A
2549          PUSH PSW
2550          RAR
```

```

053.141 037      2551      RAR
053.142 037      2552      RAR
053.143 037      2553      RAR
053.144 346 017    2554      ANI      17Q      (A) = HIGH DIGIT
053.146 302 153 053 2555      JNZ      COL2.5      NON-ZERO
053.151 076 360    2556      MVI      A,'-' '0'
053.153 306 040    2557 COL2.5 ADI      '0'
053.155 062 052 067 2558      STA      HEADA
053.160 361      2559      POP      PSW
053.161 346 017    2560      ANI      17Q      2ND DIGIT
053.163 306 060    2561      ADI      '0'
053.165 062 053 067 2562      STA      HEADB
053.170 001 222 000 2563      LXI      B,HEADLEN
053.173 021 234 066 2564      LXI      D,HEADING
053.176 041 226 067 2565      LXI      H,LISTFB
053.201 315 047 063 2566      CALL     $FWRIB      WRITE HEADING
053.204 341      2567      POP      H
                                /78.10.GC/
053.205 065      2568
053.205 065      2569 COL3   DCR      M      COUNT LINE      /78.10.GC/
                                2570
053.206 301      2571      POP      B
053.207 321      2572      POP      D
053.210 341      2573      POP      H
053.211 311      2574      RET

```

```

2576 **      CUS - COMPUTE UNUSED SPACE.
2577 *
2578 *      CUS COMPUTES THE FREE SPACE LEFT TO THE ASSEMBLER.
2579 *
2580 *      IF NOT ENOUGH IS FREE TO CONTINUE, CUS RETURNS A NEGATIVE VALUE.
2581 *
2582 *      ENTRY      NONE
2583 *      EXIT      (HL) = BYTES FREE
2584 *      'C' SET IF NOT ENOUGH TO CONTINUE
2585 *      USES      A,E,H,L
2586
2587
053.212 325      2588 CUS    PUSH     D
053.213 052 220 067 2589      LHLD     SYMPTR
053.216 353      2590      XCHG
053.217 052 224 067 2591      LHLD     RELPTR
053.222 175      2592      MOV      A,L
053.223 223      2593      SUB      E
053.224 157      2594      MOV      L,A
053.225 174      2595      MOV      A,H
053.226 232      2596      SBB      D
053.227 147      2597      MOV      H,A      COMPUTE DIFFERENCE
053.230 321      2598      POP      D      RESTORE (DE)
053.231 311      2599      RET

```



```
2601 **      DHD - DECODE HEX DIGIT.
2602 *
2603 *      DHD DECODES AN ASCII CHARACTER INTO A 4 BIT VALUE.
2604 *
2605 *      ENTRY (A) = CHARACTER
2606 *      EXIT  (A) = VALUE
2607 *      'C' SET IF ERROR
2608 *      USES  A,F
2609
2610
2611 DHD      SUI      '0'
2612          RC
2613          ERROR
2614          CPI      10
2615          JC      DHD1      IS 0-9
2616          SUI      'A'-10
2617          RC
2618          ERROR
2619          CPI      6
2620          CMC
2621          RC
2622          NOT A-F
2623          ADI      10
2624          ANA      A
2625          CLEAR CARRY
2626          EXIT WITH VALUE
2627
2628
2629 **      DEF - DEFINE SYMBOL.
2630 *
2631 *      DEF IS CALLED TO DEFINE THE SYMBOL IN *SYMBOL*.
2632 *      THE SYMBOL IS DEFINED ABSOLUTE OR RELOCATABLE, ACCORDING TO
2633 *      (EXPREL)
2634 *
2635 *      ENTRY (D) = NEW SYMBOL TYPE
2636 *      (E) = OLD TYPE. IF SYMBOL IS PRESNET, AND ITS TYPE
2637 *      IS NOT ST.UND OR (E), FLAG 'D' ERROR.
2638 *      (BC) = VALUE
2639 *      LABEL = LINE LABEL
2640 *
2641 *      EXIT  TO RET
2642 *      USES  A,F,D,E,H,L
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
```

```

053.307 247      2654      ANA      A
000.000      2655      ERNZ     ST,UND      CODE ASSUMES = 0.
053.310 312 327 053 2656      JZ      DEF1      UNDEFINED
053.313 273      2657      CMP      E
053.314 312 327 053 2658      JE      DEF1      IS PROPER OLD TYPE
053.317 366 200      2659      ORI      ST,DBL
053.321 167      2660      MOV      M,A      FLAG DOUBLE DEFINITION
053.322 315 003 057 2661      CALL     SEF
053.325 004      2662      DB      ERR,D      ** ERROR
053.326 311      2663      RET      DONT RE-DEFINE
      2664
053.327 162      2665      DEF1     MOV      M,D      SET TYPE
053.330 043      2666      INX      H
053.331 161      2667      MOV      M,C
053.332 043      2668      INX      H
053.333 160      2669      MOV      M,B      SET VALUE
053.334 311      2670      RET

      2672 **      DLH - DEFINE LABEL HERE.
      2673 *
      2674 *      DLH IS CALLED TO DEFINE A LABEL (IF ONE EXISTS) AT THE
      2675 *      CURRENT ORG.
      2676 *
      2677 *      ENTRY (LABEL) = LABEL STRING
      2678 *      EXIT (HL) = ORG VALUE
      2679 *      'D' SET ON LABEL IF DOUBLY DEFINED
      2680 *      USES ALL
      2681
      2682
053.335 072 056 070 2683      DLH     LDA      LABEL
053.340 247      2684      ANA      A
053.341 310      2685      RZ      NO LABEL EXISTS
053.342 072 122 067 2686      LDA      PASS
053.345 075      2687      DCR      A
053.346 300      2688      RNZ      RETURN IF PASS = 1
053.347 052 134 067 2689      LHLD     ORG      (HL) = LABEL'S VALUE
053.352 021 000 001 2690      LXI      D,ST,LAB*256+0
053.355 072 145 067 2691      LDA      RELFLG
053.360 062 353 067 2692      STA      EXPREL      DEFINE SYMBOL REL IF GENERATING REL CODE
053.363 104      2693      MOV      B,H
053.364 115      2694      MOV      C,L      (BC) = VALUE
053.365 303 255 053 2695      JMP      DEF      DEFINE SYMBOL HERE

```

```

2697 **      DLL - DISPLAY LISTING LINE.
2698 *
2699 *      DLL TYPES THE LISTING LINE IF THE 'L' LIST OPTION IS SET,
2700 *      OR IF AN ERROR IS PRESENT.
2701 *
2702 *      ENTRY NONE
2703 *      EXIT  NONE

```

DLL

```

2704 *      USES      ALL
2705
2706
053.370 072 122 067 2707 DLL      LDA      PASS
053.373 037      2708      RAR
053.374 330      2709      RC              DO NOTHING PASS 1
2710
2711 *      SET XTEXT FLAG
2712
053.375 072 143 067 2713      LDA      XTXLINE      /80.02.GC/
054.000 247      2714      ANA      A      /80.02.GC/
054.001 312 011 054 2715      JZ      DLL0      IS NOT CURRENTLY AN *XTEXT* /80.02.GC/
2716
054.004 076 130      2717      MVI      A,X'      /80.02.GC/
054.006 062 113 067 2718      STA      DSPLNE      FLAG THE XTEXT      /80.02.GC/
2719
054.011 072 140 067 2720 DLL0      LDA      ERRFLG
054.014 107      2721      MOV      B,A
054.015 247      2722      ANA      A
054.016 302 030 054 2723      JNZ      DLL1      HAVE ERROR
054.021 315 041 053 2724      CALL     CLE      CHECK LISTING ELIGIBILITY
054.024 310      2725      RZ      NOT TO LIST
054.025 303 130 054 2726      JMP      DLL3      DONT TRY TO INSERT ERROR MESSAGES
2727
2728 *      GENERATE ERROR CHARACTERS FOR FLAGGED ERRORS.
2729
054.030 315 003 061 2730 DLL1      CALL     $CC0      CLEAR CONTROL-0
054.033 016 003      2731      MVI      C,3      (C) = MAX NUMBER OF ERROR MESSAGES
054.035 052 123 067 2732      LHLD     ERRCNT
054.040 043      2733      INX      H
054.041 042 123 067 2734      SHLD     ERRCNT      COUNT LINES IN ERROR
054.044 041 056 067 2735      LXI      H,DSPLN
054.047 021 203 054 2736      LXI      D,DLLB-1      (DE) = TABLE POINTER
2737
054.052 023      2738 DLL2      INX      D      LOOK UP ERROR CHARACTERS
054.053 032      2739      LDAX     D
054.054 023      2740      INX      D
054.055 247      2741      ANA      A
054.056 312 074 054 2742      JZ      DLL2.5      ALL MESSAGES TYPED
054.061 240      2743      ANA      B
054.062 312 052 054 2744      JZ      DLL2      NO ERROR OF THIS TYPE
054.065 032      2745      LDAX     D      (A) = CHARACTER
054.066 167      2746      MOV      M,A      STORE IN LINE
054.067 043      2747      INX      H
054.070 015      2748      DCR      C
054.071 302 052 054 2749      JNZ      DLL2      MORE ROOM FOR ERRORS
2750
2751 *      HAVE JUST FORMATTED ERROR LINE. SEE IF TO GO TO CONSOLE
2752
054.074 072 227 067 2753 DLL2.5      LDA      LISTFB+FB.FLG
054.077 247      2754      ANA      A
054.100 312 112 054 2755      JZ      DLL2.7      SEND TO CONSOLE, FOR SURE
054.103 072 125 067 2756      LDA      ERRSHO
054.106 247      2757      ANA      A
054.107 312 130 054 2758      JZ      DLL3      JUST WRITE TO FILE
2759

```

```

2760 *      TYPE LINE (WITH ERROR) ON CONSOLE
2761
054.112 041 056 067 2762 DLL2.7 LXI    H,DSPLIN
054.115 377 003 2763 DB      SYSCALL,.PRINT PRINT HEADER
054.117 041 237 075 2764 LXI    H,LINE
054.122 315 051 061 2765 CALL   $TYPLZ      TYPE LINE TO 00
054.125 315 226 061 2766 CALL   $CRLF      END OF LINE AFTER IT
2767
2768 *      TYPE OUT LISTING LINE.
2769
054.130 2770 DLL3 EQU    *
054.130 315 100 053 2771 CALL   COL          COUNT OUTPUT LINE
054.133 041 237 075 2772 LXI    H,LINE
054.136 315 342 060 2773 CALL   $DTB          DELETE TRAILING BLANKS
054.141 075 2774 DCR    A
054.142 312 167 054 2775 JZ      DLL4          NO LINE TO LIST, MAYBE JUST HEADER
2776
2777 *      PRINT LINE HEADER AND BODY
2778
054.145 001 040 000 2779 LXI    B,DSPLEN
054.150 021 056 067 2780 LXI    D,DSPLIN
054.153 041 226 067 2781 LXI    H,LISTER
054.156 315 047 063 2782 CALL   $FWRIB        WRITE LINE
054.161 021 237 075 2783 LXI    D,LINE
054.164 303 002 063 2784 JMP     $FWRIL        WRITE LINE AND RETURN
2785
2786 *      HAVE NO LINE BODY, SEND JUST HEADER
2787
054.167 041 056 067 2788 DLL4 LXI    H,DSPLIN
054.172 315 342 060 2789 CALL   $DTB          DELETE TRAILING BLANKS
054.175 353 2790 XCHG
054.176 041 226 067 2791 LXI    H,LISTER
054.201 303 002 063 2792 JMP     $FWRIL        WRITE LINE AND RETURN
2793
054.204 2794 DLLB EQU    *
054.204 001 125 2795 DB      ERR,U,'U'
054.206 002 122 2796 DB      ERR,R,'R'
054.210 004 104 2797 DB      ERR,D,'D'
054.212 010 101 2798 DB      ERR,A,'A'
054.214 020 126 2799 DB      ERR,V,'V'
054.216 040 106 2800 DB      ERR,F,'F'
054.220 100 117 2801 DB      ERR,O,'O'
054.222 200 120 2802 DB      ERR,P,'P'
054.224 000 2803 DB      0

```

```

2805 **      DRS - DECODE REGISTER SPECIFICATION
2806 *
2807 *      DRS DECODES A REGISTER SPECIFICATION.
2808 *
2809 *      CALL   DRS
2810 *      DB      CODE
2811 *
2812 *      CODE =

```

	2813	*	1	2	3	
	2814	*	B 0	B 1	B 00	
	2815	*	C 1	D 3	D 20	
	2816	*	D 2	H 5	H 40	
	2817	*	E 3	S 7	P 60	
	2818	*	H 4			
	2819	*	L 5			
	2820	*	M 6			
	2821	*	A 7			
	2822	*				
	2823	*	ENTRY	(DE) = OPERAND POINTER		
	2824	*	EXIT	(DE) UPDATED		
	2825	*		(A) = REGISTER INDEX		
	2826	*	ERR.R SET IF ERROR			
	2827	*	USES	A,F,D,E		
	2828					
	2829					
054,225	343	DRS	XIHL		(HL) = ADDRESS OF CODE	
054,226	176	2831	MOV	A,M	(A) = INDEX	
054,227	043	2832	INX	H		
054,230	343	2833	XTHL			
054,231	345	2834	PUSH	H	SAVE (HL)	
054,232	046 054	2835	MVI	H,DRSA/256		
054,234	157	2836	MOV	L,A		
054,235	032	2837	LDAX	D		
054,236	023	2838	INX	D		
054,237	315 261 061	2839	CALL	\$TBL5		
054,242	176	2840	MOV	A,M	(A) = REGISTER SPECIFICATION	
054,243	341	2841	POP	H		
054,244	037	2842	RAR			
054,245	302 270 054	2843	JNZ	DRS3	NO GOOD	
		2844				
		2845	*	HAVE VALID REGISTER. DISCARD EXTRA CHARACTERS		
		2846				
054,250	365	2847	PUSH	PSW	SAVE REGISTER CODE	
054,251	033	2848	DCX	D		
054,252	023	2849	INX	D		
054,253	032	2850	LDAX	D		
054,254	376 101	2851	CPI	'A'		
054,256	332 266 054	2852	JC	DRS2	NOT ALPHA	
054,261	376 133	2853	CPI	'Z'+1		
054,263	332 252 054	2854	JC	DRS1	IS ALPHA	
054,266	361	2855	POP	PSW	(A) = CODE	
054,267	311	2856	RET			
		2857				
		2858	*	ILLEGAL REGISTER SPECIFICATION		
		2859				
054,270	315 003 057	2860	DRS3	CALL	SEF	
054,273	002	2861	DB	ERR.R	** ERROR	
054,274	257	2862	XRA	A		
054,275	311	2863	RET			
		2865	**	REGISTER VALUE TABLES.		
		2866				
054,276		2867	DRSA	EQU	*	GROUP 1

```
054.276 101 017 2868 DB 'A',7*2+1
054.300 102 001 2869 DB 'B',0*2+1
054.302 103 003 2870 DB 'C',1*2+1
054.304 104 005 2871 DB 'D',2*2+1
054.306 105 007 2872 DB 'E',3*2+1
054.310 110 011 2873 DB 'H',4*2+1
054.312 114 013 2874 DB 'L',5*2+1
054.314 115 015 2875 DB 'M',6*2+1
054.316 000 2876 DB 0
2877
054.317 2878 DRSE EQU * GROUP 2
054.317 102 003 2879 DB 'B',1*2+1
054.321 104 007 2880 DB 'D',3*2+1
054.323 110 013 2881 DB 'H',5*2+1
054.325 123 017 2882 DB 'S',7*2+1
054.327 000 2883 DB 0
2884
054.330 2885 DRSC EQU * GROUP 3
054.330 102 001 2886 DB 'B',00Q*2+1
054.332 104 041 2887 DB 'D',20Q*2+1
054.334 110 101 2888 DB 'H',40Q*2+1
054.336 120 141 2889 DB 'P',60Q*2+1
054.340 000 2890 DB 0

000.054 2892 .B SET DRSC/256
000.000 2893 ERRNZ DRSA/256-.B

2895 ** EBB - EVALUATE 8 BIT EXPRESSION
2896 *
2897 * EBB IS CALLED TO EVALUATE AN EXPRESSION AND TO INSURE THAT
2898 * IS EVALUATES TO 8 BITS OR LESS. IF NOT, THE *V* ERROR
2899 * IS FLAGGED.
2900 *
2901 * ENTRY (DE) = OPERAND POINTER
2902 * EXIT (C) = VALUE
2903 * (DE) UPDATED
2904 * USES A,B,C,D,E,F
2905
2906
054.341 315 215 051 2907 EBB CALL EVL EVALUATE EXPRESSION
054.344 072 353 067 2908 LDA EXPREL
054.347 247 2909 ANA A
054.350 304 365 054 2910 CNZ EBB1 RELOCATION ERROR
054.353 170 2911 MOV A,B
054.354 247 2912 ANA A
054.355 310 2913 RZ IF 0
054.356 074 2914 INR A
054.357 310 2915 RZ IF -0
054.360 315 003 057 2916 CALL SEF *V* ERROR
054.363 020 2917 DB ERR,V
054.364 311 2918 RET
2919
```

```

2920 *      RELOCATION ERROR
2921
054.365 315 003 057 2922 E8B1 CALL SEF
054.370 002          2923 DB   ERR.R      FLAG ERROR
054.371 311          2924 RET

2926 **      EPO - EVALUATE FOR PASS 1.
2927 *
2928 *      EPO IS CALLED WHEN AN EVALUATION IS REQUIRED DURING PASS 1.
2929 *
2930 *      IF PASS = 1, EVALUATE THE EXPRESSION. IF IT CONTAINS UNDEFINED
2931 *      SYMBOLS, DEFINE A SYMBOL NNNNNN, WHERE NNNNNN = THE
2932 *      STATEMENT NUMBER (IN OCTAL)..
2933 *
2934 *      IF PASS = 2, SEE IF NNNNNN IS DEFINED. IF SO, WAS AN ERROR
2935 *      PASS 1, FLAG 'U' THIS PASS.
2936 *
2937 *      ENTRY (DE) = EXPRESSION POINTER
2938 *      EXIT  (DE) UPDATED
2939 *      (BC) = VALUE
2940 *      'Z' SET IF NO ERROR
2941 *      USES  A,B,C,D,E,F
2942
2943
054.372 345          2944 EPO   PUSH   H
054.373 072 127 067 2945 LDA   STATN+1
054.376 041 075 055 2946 LXI   H,EPOA
055.001 315 234 061 2947 CALL  $UOD      UNPACK OCTAL DIGITS
055.004 072 126 067 2948 LDA   STATN
055.007 315 234 061 2949 CALL  $UOD      UNPACK OCTAL DIGITS
055.012 315 215 051 2950 CALL  EVL      EVALUATE EXPRESSION
055.015 072 122 067 2951 LDA   PASS
055.020 017          2952 RRC
055.021 322 045 055 2953 JNC   EP02      PASS = 2
2954
2955 *      PASS = 1
2956
055.024 072 140 067 2957 LDA   ERRFLG
000.000          2958 ERNZ  ERR.U-1    COSE ASSUMES = 1
055.027 037          2959 RAR
055.030 322 067 055 2960 JNC   EP04      OK
2961
2962 *      HAVE 'U' ERROR, DEFINE NNNNNN
2963
055.033 325          2964 PUSH   D
055.034 021 075 055 2965 LXI   D,EPOA    (DE) = ADDRESS OF SYMBOL
055.037 315 016 057 2966 CALL  SST      SEARCH SYMBOL TABLE
055.042 303 066 055 2967 JMP   EP03      RETURN WITH ERROR
2968
2969 *      PASS = 2
2970
055.045 325          2971 EP02  PUSH   D
055.046 021 075 055 2972 LXI   D,EPOA

```

```
055.051 052 216 067 2973      LHL D    SYMFWA
055.054 315 306 055 2974      CALL   LVT.      LOCATE VALUE IN TABLE
055.057 322 066 055 2975      JNC     EPO3      NOT FOUND, OK
055.062 315 003 057 2976      CALL   SEF       *U* AND *A* ERRORS
055.065 011      2977      DB      ERR.U+ERR.A
055.066 321      2978      EPO3    POP     D
055.067 341      2979      EPO4    POP     H
055.070 072 140 067 2980      LDA     ERRFLG
055.073 247      2981      ANA     A      SET ERROR CODE
055.074 311      2982      RET      RETURN
055.075 060 060 060 2983
055.075 060 060 060 2984      EPOA    DB      '000000','X'+80H

2986 **      FNP - FORCE NEW PAGE.
2987 *
2988 *      FNP CAUSES A PAGE EJECT, BY FORMFEED OR BY LINE FEED,
2989 *      WHICHEVER IS REQUIRED.
2990 *
2991 *      ENTRY  NONE
2992 *      EXIT   NONE
2993 *      USES   ALL
2994
055.104 072 215 067 2996      FNP     LDA     FORMDP
055.107 247      2997      ANA     A
055.110 302 132 055 2998      JNZ     FNP1      MUST LINE FEED
2999
3000 *      DEVICE WILL TAKE A FORM FEED
3001
055.113 001 001 000 3002      LXI     B,1
055.116 021 200 055 3003      LXI     D,FNPA
055.121 041 226 067 3004      LXI     H,LISTFB
055.124 315 047 063 3005      CALL   $FWRIB    WRITE
055.127 303 167 055 3006      JMP     FNP3      ADJUST LINE COUNT
3007
3008 *      MUST USE CRLF'S TO GET THERE
3009
055.132 041 214 067 3010      FNP1    LXI     H,PAGEDP
055.135 226      3011      SUB     M      (A) = .GAP SPACE
055.136 041 351 067 3012      LXI     H,LINCNT
055.141 206      3013      ADD     M      (A) = AMOUNT NEEDED
055.142 312 167 055 3014      JZ      FNP3      IF NO LINES (IS THE CASE AT START OF ASSEMBLY)
055.145 001 001 000 3015      FNP2    LXI     B,1      (BC) = COUNT
055.150 021 201 055 3016      LXI     D,FNPB
055.153 041 226 067 3017      LXI     H,LISTFB
055.156 365      3018      PUSH    PSW      SAVE COUNT
055.157 315 047 063 3019      CALL   $FWRIB    WRITE BYTE
055.162 361      3020      POP     PSW
055.163 075      3021      DCR     A
055.164 302 145 055 3022      JNZ     FNP2      GO SOME MORE
3023
3024 *      ADJUST PAGE LINE COUNT
3025
```


055.167	072 214 067	3026	FNP3	LDA	PAGEDF	
055.172	326 003	3027		SUI	3	(A) = SPACES ON PAGE -HEADING SIZE
055.174	062 351 067	3028		STA	LINCNT	SET LINES REMAINING
055.177	311	3029		RET		DONE
		3030				
055.200	014	3031	FNPA	DB	FF	FORM FEED
055.201	012	3032	FNPB	DB	NL	NEW LINE
		3034	**	GRT	- GENERATE RELOCATION TABLE.	
		3035	*			
		3036	*	GRT	IS CALLED AT THE EEND OF PASS 2 TO GENERATE	
		3037	*		ANY RELOCATION TABLES NEEDED.	
		3038	*			
		3039	*	ENTRY	NONE	
		3040	*	EXIT	NONE	
		3041	*	USES	ALL	
		3042				
		3043				
055.202	072 144 067	3044	GRT	LDA	FTFLAG	
000.000		3045		ERRNZ	FT.PIC-1	
055.205	075	3046		DCR	A	
055.206	300	3047		RNZ		NOT PIC
055.207	052 224 067	3048		LHLD	RELPTR	
055.212	353	3049		XCHG		
055.213	052 222 067	3050		LHLD	RELLWA	(DE) = LOW TABLE ADDR, (HL) = HIGH TABLE ADDR
		3051				
055.216	315 216 030	3052	GRT1	CALL	\$CDEHL	
055.221	312 242 055	3053		JE	GRT2	ALL DONE
		3054				
		3055	*	WRITE	ELEMENT TO BINARY	
		3056				
055.224	053	3057		DCX	H	
055.225	106	3058		MOV	B,M	
055.226	053	3059		DCX	H	
055.227	176	3060		MOV	A,M	
055.230	315 351 055	3061		CALL	0BB	OUTPUT
055.233	170	3062		MOV	A,B	
055.234	315 351 055	3063		CALL	0BB	OUTPUT 2ND HALF
055.237	303 216 055	3064		JMP	GRT1	SEE IF MORE
		3065				
055.242	257	3066	GRT2	XRA	A	
055.243	315 351 055	3067		CALL	0BB	FINISH TABLE
055.246	257	3068		XRA	A	
055.247	303 351 055	3069		JMP	0BB	WRITE 2ND 00 AND EXIT

```
3071 **      GSC - GET STRING CHARACTER
3072 *
3073 *      GSC READS A CHARACTER FROM A QUOTED STRING IN THE SOURCE LINE.
3074 *
3075 *      A DOUBLE QUOTE (") IS TAKEN AS ONE QUOTE CHARACTER.
3076 *
3077 *      ENTRY (DE) = POINTER TO NEXT CHARACTER
3078 *      EXIT (DE) UPDATED
3079 *      (A) = CHARACTER
3080 *      'Z' SET IF END OF STRING
3081 *      USES A,F,D,E
3082
3083
3084 GSC      LDAX D      (A) = NEXT LINE CHARACTER
3085      INX D
3086      ANA A      SEE IF END OF LINE
3087      JZ GSC4     GONE PAST END
3088      CPI QUOTE
3089      RNE      NOT END QUOTE
3090
3091 *      HAVE END-QUOTE
3092
3093      LDAX D
3094      CPI QUOTE
3095      JNE GSC3     NOT DOUBLE QUOTE, IS END QUOTE
3096      INX D
3097      ORA A      CLEAR 'Z'
3098      RET      RETURN WITH QUOTE
3099
3100 GSC3     XRA A      SET 'Z'
3101      RET
3102
3103 *      GONE PAST END OF LINE WITHOUT TRAILING QUOTE
3104
3105 GSC4     CALL SEF
3106      DB ERR.A
3107      DCX D
3108      DCX D
3109      XRA A      FLAG END OF STRING
3110      RET
3111
3112 **      LVT - LOCATE VALUE IN TABLE.
3113 *
3114 *      LVT LOOKS UP A VALUE IN A TABLE.
3115 *
3116 *      ENTRY (HL) = TBL ADDRESS
3117 *      (DE) = VALUE ADDRESS
3118 *      (A) = NOP IF 2 DATA BYTES PER ENTRY, = INX H INSTRUCTION
3119 *      IF 3 DATA BYTES PER ENTRY.
3120 *      EXIT 'C' SET IF FOUND
3121 *      (HL) = ADDRESS
3122 *      'C' CLEAR IF NOT FOUND
3123 *      (HL) = ADDRESS OF NEXT ENTRY
```

```
3124 *      USES      A,F,H,L
3125
3126
055.306 076 043 3127 LVT.  MVI      A,MI,INXH      SYMTAB SEARCH ENTRY POINT
3128
055.310 3129 LVT      EQU      *
055.310 062 344 055 3130 STA      LUTA      SET INX OR NOP INSTRUCTION
055.313 176 3131 LUTO    MOV      A,M      (A) = TABLE FIRST ENTRY
055.314 247 3132 ANA      A
055.315 310 3133 RZ
055.316 325 3134 PUSH     D      TABLE EXHAUSTED
3135                          SAVE (DE)
3136 *
3137
055.317 032 3138 LVT1    LDAX     D      COMPARE CHARACTERS
055.320 276 3139 CMP      M
055.321 302 334 055 3140 JNE      LVT2      NO MATCH
055.324 023 3141 INX      D
055.325 043 3142 INX      H
055.326 027 3143 RAL
055.327 322 317 055 3144 JNC      LVT1      MORE TO CHECK
055.332 321 3145 POP      D
055.333 311 3146 RET
3147
3148
3149 *      NOT FOUND
3150
055.334 176 3151 LVT2    MOV      A,M
055.335 043 3152 INX      H
055.336 247 3153 ANA      A
055.337 362 334 055 3154 JP      LVT2      NOT AT END OF ENTRY
055.342 043 3155 INX      H
055.343 043 3156 INX      H
055.344 043 3157 LVTA    INX      H
055.345 321 3158 POP      D
055.346 303 313 055 3159 JMP      LUTO      LOOK AGAIN
```

```
3161 **      ORB - OUTPUT BINARY BYTE.
3162 *
3163 *      ORB IS CALLED TO OUTPUT A BINARY BYTE.
3164 *      THE BYTE IS ADDED TO THE BINARY FILE, AND IS ADDED TO THE
3165 *      LISTING (IF APPROPRIATE).
3166 *      ORG IS INCREMENTED.
3167 *
3168 *      ** NOTE **      EVERYBODY WHO GENERATES ANY BINARY
3169 *      INFORMATION MUST DO IT VIA A CALL TO ORB! THE ONLY EXCEPTION
3170 *      IS THE TWO BYTES BACK-GENERATED INTO THE PIC HEADERS
3171 *      BY THE MAIN LOOP. ORB CALLS ABV, AND ABV (DURING PASS 1) KEEPS
3172 *      TRACK OF THE RANGE OF BINARY GENERATED.
3173 *
3174 *      IF PASS = 1, DO NOTHING
3175 *
3176 *      ENTRY      (A) = VALUE
```

```

3177 *      EXIT      NONE
3178 *      USES      A,F
3179
3180
055.351 345 3181 0BB      PUSH      H
055.352 052 134 067 3182      LHL      D
055.355 043 3183      INX      H
055.356 042 134 067 3184      SHLD     D
055.361 062 031 056 3185      STA      D
055.364 072 122 067 3186      LDA      D
055.367 017 3187      RRC
055.370 332 041 056 3188      JC       D
055.373 052 120 067 3189 0BB1     LHL      D
055.376 076 106 3190      MVI      A, #DISPLIM
056.000 275 3191      CMP      L
056.001 302 030 056 3192      JNE      D
3193
3194 *      LINE IS FULL. IF *G* SET, PRINT AND ADD ENTRY
3195
056.004 072 130 067 3196      LDA      LSTCTL
000.000 3197      ERNZ     LST.G-2000      CODE ASSUMES = 2000
056.007 027 3198      RAL
056.010 322 041 056 3199      JNC      D
056.013 325 3200      PUSH     D
056.014 305 3201      PUSH     B
056.015 315 370 053 3202      CALL     DLL
056.020 315 275 056 3203      CALL     PDL
056.023 301 3204      POP      B
056.024 321 3205      POP      D
056.025 303 373 055 3206      JMP      D
3207
3208 *      ADD TO LINE
3209
056.030 076 000 3210 0BB2     MVI      A, 0
056.031 3211 0BB3     EQU      *-1      VALUE STORED HERE
056.032 315 234 061 3212      CALL     $UOD      UNPACK OCTAL DIGITS
056.035 043 3213      INX      H
056.036 042 120 067 3214      SHLD     D
056.041 072 031 056 3215 0BB3     LDA      D
056.044 315 070 052 3216      CALL     ABV      ADD BINARY VALUE
056.047 341 3217      POP      H
056.050 311 3218      RET

```

```

3220 **      PAS - PRINT ASSEMBLY STATISTICS.
3221 *
3222 *      PAS PRINTS THE FINAL ASSEMBLY STATISTICS.
3223 *
3224 *      STATEMENTS = NNN
3225 *      NO ERRORS DETECTED      [OR]
3226 *      ERRORS = NN
3227 *
3228 *      ENTRY      ERRCNT = # OF ERRORS
3229 *      LINCNT = # OF LINES

```

```
3230 *      EXIT      NONE
3231 *      USES      ALL
3232
3233
056.051 052 126 067 3234 PAS      LHL D      STATNO
056.054 104      3235      MOV      B,H
056.055 115      3236      MOV      C,L
056.056 041 165 056 3237      LXI      H,PASB
056.061 076 005      3238      MVI      A,5
056.063 315 157 031 3239      CALL     $UDD      UNPACK STATEMENT COUNT
056.066 315 212 053 3240      CALL     CUS      COMPUTE UNUSED SPACE
056.071 104      3241      MOV      B,H
056.072 115      3242      MOV      C,L      (BC) = COUNT
056.073 322 101 056 3243      JNC      PAS0      NOT ALL USED UP
056.076 001 000 000 3244      LXI      B,0      ALL USED UP
056.101 041 220 056 3245 PAS0    LXI      H,PASD
056.104 076 005      3246      MVI      A,5
056.106 315 157 031 3247      CALL     $UDD      UNPACK FREE BYTES COUNT
056.111 052 123 067 3248      LHL D      ERRCNT
056.114 104      3249      MOV      B,H
056.115 115      3250      MOV      C,L      (DE) = ERROR COUNT
056.116 041 241 056 3251      LXI      H,PASD
056.121 076 005      3252      MVI      A,5
056.123 305      3253      PUSH     B      SAVE COUNT
056.124 315 157 031 3254      CALL     $UDD      UNPACK COUNT
056.127 301      3255      POP      B
056.130 170      3256      MOV      A,B
056.131 261      3257      ORA      C
056.132 302 146 056 3258      JNZ      PAS2      HAVE ERROR COUNT
056.135 315 141 061 3259      CALL     $MOVE L
056.140 005 000 270 3260      DW      5,PASE,PASD      USE NO IN PLACE OF COUNT
056.146 001 104 000 3261 PAS2    LXI      B,PASAL
056.151 021 163 056 3262      LXI      D,PASA
056.154 041 226 067 3263      LXI      H,LISTFB
056.157 315 047 063 3264      CALL     $FWRIB      WRITE TO LISTING FILE
056.162 311      3265      RET      EXIT
3266
056.163 012 012      3267 PASA    DB      NL,NL
056.165 060 060 060 3268 PASB    DB      '00000 Statements Assembled',NL
056.220 060 060 060 3269 PASC    DB      '00000 Bytes Free',NL
056.241 060 060 060 3270 PASD    DB      '00000 Errors Detected',NL
000.104      3271 PASAL    EQU      *-PASA
056.267 212      3272      DB      ENL      END FOR PRINT STATEMENT
056.270 040 116 157 3273 PASE    DB      ' No',0,0
```

```
3275 **      PDL - PREPARE DISPLAY LINE
3276 *
3277 *      PDL PRESETS THE DISPLAY LINE BY BLANKING IT OUT.
3278 *
3279 *      ENTRY      NONE
3280 *      EXIT      LSTLIN = BLANKS
3281 *      USES      A,F,H,L
3282
```

```
3283
056.275 041 056 067 3284 PDL LXI H,DSPLIN
056.300 076 040 3285 MVI A,DSPLEN
056.302 066 040 3286 PDL1 MVI M,
056.304 043 3287 INX H
056.305 075 3288 DCR A
056.306 302 302 056 3289 JNZ PDL1
056.311 042 237 075 3290 STA LINE ZERO LINE
056.314 041 072 067 3291 LXI H,DSPLNB
056.317 042 120 067 3292 SHLD OBBPTR SET NEW POINTER VALUE
056.322 311 3293 RET
```

```
3295 ** RRI - RECORD RELOCATION INFORMATION.
3296 *
3297 * RRI IS CALLED WHEN AN BINARY ADDRESS VALUE IS
3298 * ABOUT TO BE PRODUCED. IF IT IS RELOCATABLE (EXPREL M<> 0)
3299 * THEN ITS ADDRESS IS ENTERED IN THE RELOCATION TABLE.
3300 *
3301 * ENTRY ORG = ORG FOR VALUEE
3302 * EXPREL <> 0 IF RELOCATABLE
3303 * EXIT NONE
3304 * USES A,F
3305
056.323 072 353 067 3307 RRI LDA EXPREL
056.326 247 3308 ANA A
056.327 310 3309 RZ NOT RELOCATABLE
056.330 072 122 067 3310 LDA PASS
056.333 075 3311 DCR A
056.334 310 3312 RZ IS PASS 1
056.335 325 3313 PUSH D SAVE REGS
056.336 345 3314 PUSH H SAVE REGS
056.337 315 212 053 3315 CALL CUS COMPUTE UNUSED SPACE
056.342 332 036 057 3316 JC MEMOVR OVERFLOW
056.345 052 134 067 3317 LHLD ORG
056.350 353 3318 XCHG
056.351 052 224 067 3319 LHLD RELPTR
056.354 053 3320 DCX H ADD VALUE TO LIST
056.355 162 3321 MOV M,D
056.356 053 3322 DCX H
056.357 163 3323 MOV M,E
056.360 042 224 067 3324 SHLD RELPTR
056.363 341 3325 POP H
056.364 321 3326 POP D RESTORE REGS
056.365 311 3327 RET
3328
3329
```

```

3331 **      RSF - REWIND SOURCE FILE.
3332 *
3333 *      RSF IS CALLED TO REWIND THE INPUT SOURCE FILE.
3334 *
3335 *      ENTRY  RSFA = TEXT NAME
3336 *            RSFB = TEXT LENGTH
3337 *      EXIT  TAPE POSITIONED
3338 *      USES  ALL
3339
3340
056.366 001 000 000 3341 RSF  LXI    B,0
056.371 076 002     3342     MVI    A,CN,SOU
056.373 377 047     3343     DB      SYSCALL,,POSIT      REWIND SOURCE FILE
056.375 041 261 067 3344     LXI    H,SORCFB
057.000 303 157 062 3345     JMP     $FCLEAR      CLEAR FILE BLOCK AND EXIT

```

```

3347 **      SEF - SET ERROR FLAGS.
3348 *
3349 *      SEF SETS THE SPECIFIED ERROR IN *ERRFLG*. THEN RETURN TO
3350 *      RET+1
3351 *
3352 *      CALL  SEF
3353 *      DB    ERRBIT
3354 *
3355 *      ENTRY (RET) = ERROR
3356 *      EXIT  ERROR SET
3357 *      RETURN TO (RET)+1
3358 *      USES  A,F
3359
3360
057.003 343     3361 SEF  XTHL      (HL) = RETURN ADDRESS
057.004 072 140 067 3362     LDA     ERRFLG
057.007 266     3363     ORA     M
057.010 062 140 067 3364     STA     ERRFLG
057.013 043     3365     INX     H
057.014 343     3366     XTHL      ADVANCE EXIT
057.015 311     3367     RET      RETURN PAST CODE

```

```

3369 **      SST - SEARCH SYMBOL TABLE.
3370 *
3371 *      SST SCANS THE SYMTAB FOR A GIVEN ENTRY. IF FOUND, RETURN
3372 *      IF NOT FOUND, CREATE AS TYPE *U*
3373 *
3374 *      ENTRY (DE) = ADDRESS OF SYMBOL
3375 *      EXIT  (DE) UNCHANGED
3376 *      (HL) = ADDRESS OF START OF VALUE BYTES.
3377 *      USES  A,F,H,L
3378
3379
057.016 052 216 067 3380 SST  LHLD    SYMFWA

```

```
057.021 315 306 055 3381 CALL LUT. LOCATE VALUE IN TABLE
057.024 330 3382 RC FOUND IT
057.025 325 3383 PUSH D SAVE (DE)
3384
3385 * WILL CREATE NEW ENTRY. SEE IF TABLE OVERFLOW.
3386
057.026 345 3387 PUSH H
057.027 315 212 053 3388 CALL CUS COMPUTE UNUSED SPACE
057.032 341 3389 POP H
057.033 322 071 057 3390 JNC SST1 OK
057.036 315 136 031 3391 MEMOVR CALL $TYPTX
057.041 012 012 007 3392 DB NL,NL,BELL,'Symtab Overflow',NL,NL,BELL+2000
057.066 303 072 046 3393 JMP END. FORCE END OF THIS PASS
3394
057.071 321 3395 SST1 POP D REFRESH (DE)
057.072 325 3396 PUSH D
3397
3398 * NOT FOUND. PUT IN TABLE, SET TYPE = ST.UND
3399
057.073 032 3400 SST2 LDAX D
057.074 167 3401 MOV M,A
057.075 023 3402 INX D
057.076 043 3403 INX H
057.077 007 3404 RLC
057.100 322 073 057 3405 JNC SST2
057.103 345 3406 PUSH H
057.104 021 020 000 3407 LXI D,16
057.107 031 3408 DAD D
057.110 042 220 067 3409 SHLD SYMPTR SET SYMBOL TABLE LIMIT
057.113 341 3410 POP H
057.114 321 3411 POP D
057.115 311 3412 RET SYMBOL CREATED IN TABLE
```

```
3414 ** UOL - UNPACK ORG INTO LINE.
3415 *
3416 * UOL UNPACKS THE ORIGIN VALUE INTO THE LISTING LINE.
3417 *
3418 * ENTRY NONE
3419 * EXIT (HL) = SORG
3420 * USES A,F,H,L
3421
3422
057.116 052 136 067 3423 UOL LHLD SORG
057.121 345 3424 UOL. PUSH H
057.122 325 3425 PUSH D
057.123 353 3426 XCHG (DE) = VALUE
057.124 041 061 067 3427 LXI H,DSPLNA
057.127 172 3428 MOV A,D
057.130 315 234 061 3429 CALL $UOD UNPACK BANK
057.133 066 056 3430 MVI M,' ' SET PERIOD BETWEEN BANKS
057.135 043 3431 INX H
057.136 173 3432 MOV A,E
057.137 315 234 061 3433 CALL $UOD UNPACK ADDR
```



```
057.142 321      3434      POP      D
057.143 341      3435      POP      H
057.144 311      3436      RET

                                3438  **      UNL - UNPACK NEXT LINE.
                                3439  *
                                3440  *      UNL UNPACKS THE NEXT SOURCE LINE FROM THE TEXT BUFFER. IF THE
                                3441  *      IS NO MORE TEXT, AND *END* LINE IS GENERATED.
                                3442  *
                                3443  *      ENTRY      NONE
                                3444  *      EXIT      'Z' SET IF COMMENT
                                3445  *      USES      ALL
                                3446
                                3447
057.145      3448  UNL      EQU      *                                /80.02.6C/
                                3449
057.145 052 126 067 3450      LHL      STATNO                                /80.02.6C/
057.150 104      3451      MOV      B,H                                /80.02.6C/
057.151 115      3452      MOV      C,L                                /80.02.6C/
057.152 041 106 067 3453      LXI      H,DSPLND      BC = STATEMENT NUMBER
057.155 076 005      3454      MVI      A,5      HL = ADDRESS TO PAT AT
057.157 315 157 031 3455      CALL     $UDD      A = DIGIT COUNT
                                3456      OUTPUT THE LINE NUMBER
                                3457
057.162 001 144 000 3457      LXI      B,LINEMAX
057.165 021 237 075 3458      LXI      D,LINE
057.170 072 142 067 3459      LDA      TXFLG
057.173 062 143 067 3460      STA      TXLINE      TXLINE < 0 IFF READING FROM XTEXT
057.176 247      3461      ANA      A
057.177 312 225 057 3462      JZ      UNL00      FROM MAIN SOURCE
057.202 041 314 067 3463      LXI      H,XTXFB
057.205 315 177 062 3464      CALL     $FREAL      READ LINE
057.210 322 250 057 3465      JNC      UNL0      GOT IT
                                3466
                                3467  *      EOF ON XTEXT FILE
                                3468
057.213 315 052 062 3469      CALL     $FCLO      CLOSE
057.216 257      3470      XRA      A
057.217 062 142 067 3471      STA      TXFLG
057.222 303 145 057 3472      JMP      UNL      TRY AGAIN
                                3473
057.225 041 261 067 3474  UNL00  LXI      H,$ORCFB
057.230 315 177 062 3475      CALL     $FREAL      READ LINE
057.233 322 250 057 3476      JNC      UNL0      OK
                                3477
                                3478  *      EOF ON MAIN PROGRAM, FAKE AN END STATEMENT
                                3479
057.236 315 141 061 3480      CALL     $MOVEL
057.241 027 000 074 3481      DW      UNLAL,UNLA,LINE      USE END STATEMENT
057.247 353      3482      XCHG      (DE) = LINEE LWA
057.250 072 237 075 3483  UNL0  LDA      LINE
057.253 376 052      3484      CPI      '*'
057.255 310      3485      RE
057.256 041 237 075 3486      LXI      H,LINE      (HL) = LINE POINTER
```

```

3487
3488 *      STRIP OFF LABEL.
3489
057.261 021 056 070 3490      LXI      D,LABEL
057.264 006 011 3491      MVI      B,B+1      8 CHARACTER MAX      /80.02.6C/
057.266 315 015 060 3492      CALL     UNL3      STRIP LABEL
057.271 032 3493      LDAX      D      CHECK FOR '?'
057.272 326 272 3494      SUI      /'+2000
057.274 302 305 057 3495      JNZ      UNL0.5      NOT :
057.277 022 3496 UNL0.3 STAX      D      CLEAR IT
057.300 033 3497      DCX      D
057.301 032 3498      LDAX      D
057.302 366 200 3499      ORI      2000
057.304 022 3500      STAX      D
057.305 076 065 3501 UNL0.5 MVI      A,#LABEL+7      FLAG LAST
057.307 223 3502      SUB      E      MAKE SURE AM NOW 7 OR LESS      /80.02.6C/
057.310 302 323 057 3503      JNE      UNL0.7      IS OK
057.313 315 003 057 3504      CALL     SEF
057.316 040 3505      DB      ERR.F
057.317 257 3506      XRA      A
057.320 303 277 057 3507      JMP      UNL0.3
3508
3509 *      VERIFY LABEL STARTS WITH ALPHA      /80.02.6C/
3510
057.323 021 056 070 3511 UNL0.7 LXI      D,LABEL
057.326 032 3512      LDAX      D      /80.02.6C/
057.327 247 3513      ANA      A      /80.02.6C/
057.330 312 351 057 3514      JZ      UNL0.9      NO LABEL DEFINED      /80.02.6C/
057.333 346 177 3515      ANI      1770      STRIP OFF ANY *END* BIT      /80.02.6C/
057.335 315 034 051 3516      CALL     LCT.      LOOK-UP CHARACTER TYPE      /80.02.6C/
057.340 346 200 3517      ANI      CT.ALPH      /80.02.6C/
057.342 302 351 057 3518      JNZ      UNL0.9      CHARACTER IS ALPHA      /80.02.6C/
057.345 315 003 057 3519      CALL     SEF
057.350 040 3520      DB      ERR.F      FLAG FORMAT ERROR      /80.02.6C/
3521
3522 *      STRIP OFF OPCODE.
3523
057.351 021 067 070 3524 UNL0.9 LXI      D,OPCODE
057.354 006 006 3525      MVI      B,B+1
057.356 315 015 060 3526      CALL     UNL3
3527
3528 *      COPY EXPRESSION TO WORKAREA.
3529
057.361 021 074 070 3530      LXI      D,EXPWRK
057.364 176 3531 UNL1      MOV      A,M
057.365 022 3532      STAX      D
057.366 023 3533      INX      D
057.367 043 3534      INX      H
057.370 247 3535      ANA      A
057.371 302 364 057 3536      JNZ      UNL1      NOT AT END OF LINE
057.374 023 3537      INX      D
057.375 022 3538      STAX      D      SET ZERO BYTE AT END
057.376 033 3539      DCX      D
057.377 076 040 3540      MVI      A,' '
060.001 022 3541      STAX      D      GUARANTEE /',000 ENDING
3542

```

```

3543 *      SEE IF COMMENT.
3544 *
3545 *      IF NO LABEL OR OPCODE, IS COMMENT
3546
060.002 072 056 070 3547 UNL2.5 LDA LABEL
060.005 346 177 3548 ANI 1770
060.007 300 3549 RNZ NOT COMMENT
060.010 072 067 070 3550 LDA OPCODE
060.013 247 3551 ANA A
060.014 311 3552 RET

3554 **     UNL3 - PARSE LABEL OR OPCODE.
3555 *
3556 *      COPY A CHARACTER STRING FROM (HL) TO (DE)
3557 *      UNTIL TAB, SPACE, OR END OF LINE IS SEEN.
3558 *
3559 *      IF NONE COPIED, ZERO (DE)
3560 *      IF SOME COPIED, SET 80H ON LAST CHARACTER
3561 *
3562 *      ENTRY (B) = MAX CHARACTER COUNT
3563 *      EXIT (DE) = ADDRESS OF LAST CHARACTER
3564
060.015 176 3565 UNL3
060.016 022 3566 MOV A,M
060.017 247 3567 STAX D
060.020 312 047 060 3568 ANA A
060.023 043 3569 JZ UNL5 IS END OF LINE
060.024 376 040 3570 INX H
060.026 312 047 060 3571 CPI ' '
060.031 376 011 3572 JE UNL5 IS SPACE
060.033 312 047 060 3573 CPI TAB
060.036 023 3574 JE UNL5 IS TAB
060.037 005 3575 INX D
060.040 302 015 060 3576 DCR B
3577 JNZ UNL3 MORE TO GO
3578
3579 *      TOO MANY CHARACTERS. FLAG AN *F* ERROR
3580
060.043 315 003 057 3581 CALL SEF *F* ERROR
060.046 040 3582 DB ERR.F
3583
3584 *      ITEM COPIED. SET SIGN BIT AND 0 NEXT BYTE
3585
060.047 257 3586 UNL5 XRA A
060.050 022 3587 STAX D CLEAR FIELD
060.051 033 3588 DCX D SET SIGN OVER LAST CHARACTER
060.052 032 3589 LDAX D
060.053 366 200 3590 ORI 80H
060.055 022 3591 STAX D
3592
3593 *      SKIP BLANKS AND TABS
3594
060.056 053 3595 DCX H

```

```

060.057 043      3596 UNL9 INX      H
060.060 176      3597 UNL10 MOV     A,M
060.061 376 040  3598      CFI      /
060.063 312 057 060 3599      JE      UNL9      IS BLANK
060.066 376 011  3600      CFI      TAB
060.070 312 057 060 3601      JE      UNL9      IS TAB
060.073 311      3602      RET      EXIT
                                3603
060.074 011 105 116 3604 UNLA     DB      /      END Statement Missing,0
000.027      3605 UNLAL     EQU      *-UNLA

```

```

3607 **      WBB - WRITE BINARY BUFFER.
3608 *
3609 *      WBB WRITES THE BINARY BUFFER TO THE DISK.
3610 *
3611 *      IF IT IS A REPLACEMENT FOR AN EXISIGING SECTOR, JUST WRITE IT.
3612 *
3613 *      IF NECESSARY, EXTEND THE FILE WITH GARBAGE UNTIL THE PROPER SECTOR
3614 *      IS REACHED.
3615 *
3616 *      ENTRY  NONE
3617 *      EXIT   NONE
3618 *      USES   NONE
3619
3620
060.123 315 054 031 3621 WBB     CALL    $SAVALL      SAVE REGISTERS
060.126 072 171 067 3622 WBB1    LDA      BINCSN      (A) = CURRENT SECTOR NUMBER
060.131 117      3623      MOV      C,A
060.132 006 000  3624      MVI      B,0
060.134 076 000  3625      MVI      A,CN.BIN
060.136 377 047  3626      DB      SYSCALL,.POSIT      POSITION
060.140 322 317 060 3627      JNC      WBB2      OK
060.143 376 001  3628      CFI      EC.EOF
060.145 302 334 060 3629      JNE      BINERR      NOT EOF, SERIOUS ERROR

```

```

3630
3631 *      SHOULD NOT OCCUR
3632
060.150 315 136 031 3633      CALL    $TYPTX
060.153 012 007 111 3634      DB      NL,BELL,'Internal Error #1'
060.176 012 124 150 3635      DB      NL,'This should not occur.'
060.225 103 157 156 3636      DB      'Contact HEATH Technical Correspondence for Assistance.'
060.313 212      3637      DB      ENL
060.314 303 325 042 3638      JMP      EXIT
3639
3640 *      GOT THERE. WRITE SECTOR
3641
060.317 001 000 001 3642 WBB2    LXI      B,256
060.322 021 237 070 3643      LXI      D,BINBFR
060.325 076 000  3644      MVI      A,CN.BIN
060.327 377 005  3645      DB      SYSCALL,.WRITE
060.331 322 047 031 3646      JNC      $RSTALL      EXIT IF OK
3647
3648 *      ERROR ON BINARY FILE

```

15:20:18 16-MAY-80

060.342

3654

XTEXT DTB

3656X ** \$DTB - DELETE TRAILING BLANKS.

3657X *

3658X * \$DTB DELETES THE TRAILING BLANKS FROM A CODED LINE.

3659X *

3660X * ENTRY (HL) = LINE FWA

3661X * EXIT (A) = LENGTH OF RESULT (INCLUDING '00' TERMINATOR BYTE)

3662X * USES A,F

3663X

3664X

060.342 325

3665X \$DTB PUSH D SAVE (DE)

060.343 124

3666X MOV D,H

060.344 135

3667X MOV E,L (DE) = FWA

060.345 033

3668X DCX D (DE) = FWA-1

060.346 176

3669X \$DTB1 MOV A,M

060.347 043

3670X INX H

060.350 247

3671X ANA A FIND END OF LINE

060.351 302 346 060

3672X JNZ \$DTB1

060.354 053

3673X DCX H (HL) = ADDRESS OF TERMINATING ZERO BYTE

3674X

3675X * GOT END OF LINE. DELETE TRAILING BLANKS

3676X

060.355 053

3677X \$DTB2 DCX H BACKUP ONE CHARACTER

060.356 315 216 030

3678X CALL \$CDEHL

060.361 312 372 060

3679X JE \$DTB3 GONE PAST FRONT OF LINE, MUST BE ALL BLANKS

060.364 176

3680X MOV A,M

060.365 376 040

3681X CPI /

060.367 312 355 060

3682X JE \$DTB2 GOT BLANK

3683X

3684X * HAVE TRIMED LINE, COMPUTE LENGTH

3685X

060.372 043

3686X \$DTB3 INX H

060.373 066 000

3687X MVI M,0 TERMINATE LINE

060.375 175

3688X MOV A,L

060.376 223

3689X SUB E (A) = LENGTH +1 (FOR '00' BYTE)

060.377 353

3690X XCHG

061.000 043

3691X INX H (HL) = LINE FWA

061.001 321

3692X POP D RESTORE (DE)

061.002 311

3693X RET

061.003

3694 XTEXT CCO

3696X ** \$CCO - CLEAR CONTROL-0

3697X *

3698X * \$CCO IS CALLED TO CLEAR THE EFFECT OF THE CTL-0 CHARACTER.

3699X *

3700X * ENTRY NONE

3701X * EXIT NONE

3702X * USES NONE

3703X

```
.....
061.003 315 054 031 3704X
061.006 078 004 3705X $CCO CALL $SAVALL SAVE REGISTERS
061.010 001 001 000 3706X MVI A,I:CONFL
061.013 377 006 3707X LXI B,CD.FLG CLEAR CD.FLG
061.015 303 047 031 3708X DB SYSCALL,CONSL
061.020 3710 XTEXT MCU RESTORE REGISTERS AND RETURN
.....
```

```
.....
3712X ** MCU - MAP LOWER CASE TO UPPER CASE.
3713X *
3714X * MCU MAPS A LOWER CASE ALPHABETIC TO UPPER
3715X * CASE.
3716X *
3717X * ENTRY (A) = CHARACTER
3718X * EXIT (A) = CHARACTER RESULT
3719X * USES A,F
3720X
3721X
061.020 376 141 3722X $MCU CPI 'a'
061.022 330 3723X RC NOT LOWER CASE
061.023 376 173 3724X CPI 'z'+1
061.025 320 3725X RNC NOT LOWER CASE
061.026 326 040 3726X SUI 'a'-'A'
061.030 311 3727X RET
061.031 3728 XTEXT MLU
.....
```

```
.....
3730X ** MLU - MAP LOWER CASE LINE TO UPPER CASE.
3731X *
3732X * MLU MAPS THE LOWER CASE ALPHABETICS IN A LINE TO UPPER CASE.
3733X *
3734X * ENTRY (HL) = LINE FWA
3735X * EXIT NONE
3736X * USES NONE
3737X
3738X
061.031 345 3739X $MLU PUSH PSW SAVE (PSW)
061.032 345 3740X PUSH H SAVE FWA
061.033 053 3741X DCX H ANTICIPATE INX H
061.034 043 3742X $MLU1 INX H
061.035 176 3743X MOV A,M (A)= CHARACTER
061.036 315 020 061 3744X CALL $MCU MAP CHAR TO UPPER
061.041 167 3745X MOV M,A
061.042 247 3746X ANA A
061.043 302 034 061 3747X JNZ $MLU1 MORE TO GO
061.046 341 3748X POP H RESTORE (HL)
061.047 361 3749X POP PSW RESTORE (PSW)
061.050 311 3750X RET
061.051 3751 XTEXT TYPLZ
.....
```

```

3753X **      $TYPLZ - TYPE LINE UNTIL ZERO BYTE ENCOUNTERED.
3754X *
3755X *      NO NEW-LINE IS SENT.
3756X *
3757X *      ENTRY  (HL) = FWA OF TEXT
3758X *      EXIT   (HL) ADVANCED PAST ZERO BYTE
3759X *      USES   A,F,H,L
3760X
3761X
061.051 176    3762X $TYPLZ MOV    A,M
061.052 043    3763X      INX    H
061.053 247    3764X      ANA    A
061.054 310    3765X      RZ
061.055 315 136 061 3766X      CALL $WCHAR      ALL DONE
061.060 303 051 061 3767X      JMP    $TYPLZ      WRITE LINE
061.063      3768      XTEXT  HLIHL      DO MORE

```

```

3770X **      $HLIHL - LOAD HL INDIRECT THROUGH HL.
3771X *
3772X *      (HL) = ((HL))
3773X *
3774X *      ENTRY  NONE
3775X *      EXIT   NONE
3776X *      USES   A,H,L
3777X
030.211      3778X $HLIHL EQU    30211A      IN H17 ROM
061.063      3779      XTEXT  CDEHL

```

```

3781X **      $CDEHL - COMPARE (DE) TO (HL)
3782X *
3783X *      $CDEHL COMPARES (DE) TO (HL) FOR EQUALITY.
3784X *
3785X *      ENTRY  NONE
3786X *      EXIT   'Z' SET IF (DE) = (HL)
3787X *      USES   A,F
3788X
030.216      3790X $CDEHL EQU    30216A      IN H17 ROM
061.063      3791      XTEXT  CHL

```

```

3793X **      $CHL - COMPLEMENT (HL).
3794X *
3795X *      (HL) = -(HL)      TWO'S COMPLEMENT
3796X *
3797X *      ENTRY  NONE
3798X *      EXIT   NONE
3799X *      USES   A,F,H,L

```



```
3800X
3801X
030.224 3802X *CHL EQU 30224A IN H17 ROM
061.063 3803 XTEXT DADA2
```

```
3805X ** $DADA. - ADD (0,A) TO (H,L)
3806X *
3807X * ENTRY NONE
3808X * EXIT (HL) = (HL) + (0A)
3809X * USES A,F,H,L
3810X
3811X
030.101 3812X $DADA. EQU 30101A IN H17 ROM
061.063 3813 XTEXT SAVALL
```

```
3815X ** $RSTALL - RESTORE ALL REGISTERS.
3816X *
3817X * $RSTALL RESTORES ALL THE REGISTERS OFF THE STACK, AND
3818X * RETURNS TO THE PREVIOUS CALLER.
3819X *
3820X * ENTRY (SP) = PSW
3821X * (SP+2) = BC
3822X * (SP+4) = DE
3823X * (SP+6) = HL
3824X * (SP+8) = RET
3825X * EXIT TO *RET*, REGISTERS RESTORED
3826X * USES ALL
3827X
3828X
031.047 3829X $RSTALL EQU 31047A IN H17 ROM
```

```
3831X ** $SAVALL - SAVE ALL REGISTERS ON STACK.
3832X *
3833X * $SAVALL SAVES ALL THE REGISTERS ON THE STACK.
3834X *
3835X * ENTRY NONE
3836X * EXIT (SP) = PSW
3837X * (SP+2) = BC
3838X * (SP+4) = DE
3839X * (SP+6) = HL
3840X * USES H,L
3841X
3842X
031.054 3843X $SAVALL EQU 31054A IN H17 ROM
061.063 3844 XTEXT RTL
```

```

3846X **      $RTL - READ TEXT LINE.
3847X *
3848X *      $RTL READS A LINE FROM THE TERMINAL.
3849X *
3850X *      CHARACTER ARE ACCEPTED FROM THE TERMINAL, RUBOUT AND BACKSPACE
3851X *      CHARACTERS ARE PROCESSED. WHEN A CARRIAGE RETURN IS ENTERED,
3852X *      $RTL RETURNS.
3853X *
3854X *      ENTRY (HL) = BUFFER FWA
3855X *      EXIT 'C' CLEAR IF OK
3856X *      DATA IN BUFFER
3857X *      (A) = TEXT LENGTH
3858X *      'C' SET IF CTL-D STRUCK
3859X *      USES A,F
3860X
3861X
061.063 315 072 061 3862X $RTL. CALL $RTL $RTL IN UPPER CASE
061.066 330 3863X RC CTL-D
061.067 303 031 061 3864X JMP $MLU MAP LINE TO UPPER CASE
3865X
061.072 3866X $RTL EQU *
061.072 345 3867X PUSH H SAVE FWA
061.073 315 130 061 3868X $RTL1 CALL $RCHAR
061.076 376 004 3869X CPI CTLD
061.100 312 125 061 3870X JE $RTL2 CTL-D STRUCK
061.103 167 3871X MOV M,A
061.104 043 3872X INX H
061.105 376 012 3873X CPI NL
061.107 302 073 061 3874X JNE $RTL1
061.112 053 3875X DCX H
061.113 066 000 3876X MVI M,0
061.115 043 3877X INX H
3878X
3879X *      ALL DONE. COMPUTE LENGTH
3880X
061.116 353 3881X XCHG (DE) = LWA+1
061.117 343 3882X XTHL (HL) = FWA
061.120 173 3883X MOV A,E
061.121 225 3884X SUB L (A) = LENGTH
061.122 247 3885X ANA A CLEAR CARRY
061.123 321 3886X POP D RESTORE (DE)
061.124 311 3887X RET
3888X
3889X *      CTL-D STRUCK
3890X
061.125 341 3891X $RTL2 POP H (HL) = FWA
061.126 067 3892X STC
061.127 311 3893X RET
061.130 3894 XTEXT RCHAR

```

```
3896X **      $RCHAR - READ SINGLE CHARACTER FROM CONSOLE.
3897X *
3898X *      ENTRY  NONE
3899X *      EXIT   (A) = CHARACTER
3900X *      USES   A,F
3901X
3902X
061.130 377 001 3903X $RCHAR DB      SYSCALL,.SCIN
061.132 332 130 061 3904X JC      $RCHAR      NOT READY
061.135 311      3905X RET
3906X
061.136 377 002 3907X $WCHAR DB      SYSCALL,.SCOUT
061.140 311      3908X RET
061.141      3909 XTEXT UDD

3911X **      $UDD - UNPACK DECIMAL DIGITS.
3912X *
3913X *      UDD CONVERTS A 16 BIT VALUE INTO A SPECIFIED NUMBER OF
3914X *      DECIMAL DIGITS. THE RESULT IS ZERO FILLED.
3915X *
3916X *      ENTRY  (B,C) = ADDRESS VALUE
3917X *      (A) = DIGIT COUNT
3918X *      (H,L) = MEMORY ADDRESS
3919X *      EXIT   (HL) = (HL) + (A)
3920X *      USES   ALL
3921X
3922X
031.157 3923X $UDD EQU 31157A      IN H17 ROM
061.141 3924 XTEXT MOVEL

3926X **      $MOVEL - MOVE DATA
3927X *
3928X *      $MOVEL MOVES A BLOCK OF BYTES TO A NEW MEMORY ADDRESS.
3929X *      IF THE MOVE IS TO A LOWER ADDRESS, THE BYTES ARE MOVED FROM
3930X *      FIRST TO LAST.
3931X *
3932X *      IF THE MOVE IS TO A HIGHER ADDRESS, THE BYTES ARE MOVED FROM
3933X *      LAST TO FIRST.
3934X *
3935X *      THIS IS DONE SO THAT AN OVERLAPED MOVE WILL NOT 'RIPPLE'.
3936X *
3937X *      CALL  $MOVEL
3938X *      DW    COUNT
3939X *      DW    FROM
3940X *      DW    TO
3941X *
3942X *      ENTRY  ((SP)) = RET
3943X *      (RET+0) = COUNT (WORD VALUE)
3944X *      (RET+2) = FROM
3945X *      (RET+4) = TO
```

```

3946X *      EXIT      TO (RET+6)
3947X *      (DE) = ADDRESS OF NEXT FROM BYTE
3948X *      (HL) = ADDRESS OF NEXT *TO* BYTE
3949X *      'C' CLEAR
3950X *      USES      ALL
3951X
3952X
061.141 341 3953X $MOVE POP      H      (HL) = RET
061.142 116 3954X      MOV      C,M
061.143 043 3955X      INX      H
061.144 106 3956X      MOV      B,M      (BC) = COUNT
061.145 043 3957X      INX      H
061.146 136 3958X      MOV      E,M
061.147 043 3959X      INX      H
061.150 126 3960X      MOV      D,M      (DE) = FROM
061.151 043 3961X      INX      H
061.152 325 3962X      PUSH     D      ((SP)) = FROM
061.153 136 3963X      MOV      E,M
061.154 043 3964X      INX      H
061.155 126 3965X      MOV      D,M      (DE) = TO
061.156 043 3966X      INX      H
061.157 343 3967X      XTHL      ((SP)) = RET, (HL) = FROM
061.160 353 3968X      XCHG      (DE) = FROM, (HL) = TO
061.161 303 252 030 3969X      JMP      $MOVE      MOVE IT
061.164      3970      XTEXT     CPF

```

```

3972X **      $CPF - COPY FILE NAME
3973X *
3974X *      $CPF COPIES A FILE NAME FROM ONE LOCATION TO ANOTHER.
3975X *
3976X *      THE CHARACTERS ARE COPIED UNTIL A DELIMITER (',', ' ', '=', OR 00)
3977X *      IS FOUND.
3978X *
3979X *      THE FILENAME IS THEN TERMINATED WITH A '00' BYTE.
3980X *
3981X *      ENTRY      (DE) = FROM ADDRESS
3982X *      (HL) = TO ADDRESS
3983X *      EXIT      'C' CLEAR IF OK
3984X *      (DE) = ADVANCED PAST NAME AND DELIMITER
3985X *      (HL) POINTS TO '00' BYTE OF DESTINATION
3986X *      (A) = DELIMITER
3987X *      'C' SET IF ERROR
3988X *      USES      ALL
3989X
3990X
061.164 006 022 3991X $CPF MVI      B,FB.NAML+1      SET MAX LENGTH
061.166 032 3992X $CPF1 LDAX     D
061.167 247 3993X      ANA      A
061.170 312 223 061 3994X      JZ      $CPF2      END
061.173 023 3995X      INX      D
061.174 376 054 3996X      CPI      ','
061.176 312 223 061 3997X      JE      $CPF2
061.201 376 075 3998X      CPI      '='

```

```
061.203 312 223 061 3999X JE $CPF2
061.206 376 040 4000X CFI ' '
061.210 312 223 061 4001X JE $CPF2 IS BLANK
061.213 167 4002X MOV M,A COPY
061.214 043 4003X INX H
061.215 005 4004X DCR B
061.216 302 166 061 4005X JNZ $CPF1 IF MORE GO TO
061.221 067 4006X STC OVERFLOW OF AREA
061.222 311 4007X RET
4008X
4009X * DONE.
4010X
061.223 066 000 4011X $CPF2 MVI M,0 TERMINATE
061.225 311 4012X RET
061.226 4013 XTEXT INDL
```

```
4015X ** $INDL - INDEXED LOAD.
4016X *
4017X * $INDL LOADS DE WITH THE TWO BYTES AT (HL)+DISPLACEMENT
4018X *
4019X * THIS ACTS AS AN INDEXED FULL WORD LOAD.
4020X *
4021X * (DE) = ( (HL) + DISPLACEMENT )
4022X *
4023X * ENTRY ((RET)) = DISPLACEMENT (FULL WORD)
4024X * (HL) = TABLE ADDRESS
4025X * EXIT TO (RET+2)
4026X * USES A,F,D,E
4027X
4028X
030.234 4029X $INDL EQU 30234A IN H17 ROM
061.226 4030 XTEXT MOVE
```

```
4032X ** $MOVE - MOVE DATA
4033X *
4034X * $MOVE MOVES A BLOCK OF BYTES TO A NEW MEMORY ADDRESS.
4035X * IF THE MOVE IS TO A LOWER ADDRESS, THE BYTES ARE MOVED FROM
4036X * FIRST TO LAST.
4037X *
4038X * IF THE MOVE IS TO A HIGHER ADDRESS, THE BYTES ARE MOVED FROM
4039X * LAST TO FIRST.
4040X *
4041X * THIS IS DONE SO THAT AN OVERLAPED MOVE WILL NOT 'RIPPLE'.
4042X *
4043X * ENTRY (BC) = COUNT
4044X * (DE) = FROM
4045X * (HL) = TO
4046X * EXIT MOVED
4047X * (DE) = ADDRESS OF NEXT FROM BYTE
4048X * (HL) = ADDRESS OF NEXT *TO* BYTE
```

```
4049X *      'C' CLEAR
4050X *      USES  ALL
4051X
4052X
030.252      4053X $MOVE EQU 30252A      IN H17 ROM
061.226      4054      XTEXT CRLF
```

```
4056X **      $CRLF - TYPE CARRIAGE RETURN/ LINE FEED
4057X *
4058X *      $CRLF IS USED TO GENERATE PADDED CRLF'S.
4059X *
4060X *      ENTRY  NONE
4061X *      EXIT   (A) = 0
4062X *      USES  A,F
4063X
4064X
061.226 076.012 4065X $CRLF MVI A,NL
061.230 377.002 4066X      DB SYSCALL,.SCOUT
061.232 257      4067X      XRA A
061.233 311      4068X      RET
061.234      4069      XTEXT DADA
```

```
4071X **      $DADA - PERFORM (H,L) = (H,L) + (0,A)
4072X *
4073X *      ENTRY  (H,L) = BEFORE VALUE
4074X *      (A) = BEFORE VALUE
4075X *      EXIT   (H,L) = (H,L) + (0,A)
4076X *      'C' SET IF OVERFLOW
4077X *      USES  F,H,L
4078X
4079X
030.072      4080X $DADA EQU 30072A      IN H17 ROM
061.234      4081      XTEXT UOD
```

```
4083X **      $UOD - UNPACK OCTAL DIGITS.
4084X *
4085X *      UOD CONVERTS A SINGLE BYTE INTO 3 OCTAL DIGITS., ZERO FILL
4086X *
4087X *      ENTRY  (A) = BYTE VALUE
4088X *      (H,L) = ADDRESS OF 3 BYTE AREA FOR DIGITS
4089X *      EXIT   (H,L) = (H,L)+3
4090X *      USES  A,H,L
4091X
4092X
061.234 305      4093X $UOD PUSH B
061.235 006 003 4094X      MVI B,3      (B) = LOOP COUNT
061.237 247      4095X      ANA A      CLEAR CARRY
```

COMMON DECKS.

\$UOD

15:21:42 16-MAY-80

```

061.240 027      4096X
061.241 027      4097X UOD1  RAL
061.242 027      4098X      RAL
061.243 365      4099X      RAL
061.244 346 007   4100X      PUSH  PSW      SAVE VALUE
061.246 306 060   4101X      ANI    7
061.250 167      4102X      ADI    '0'
061.251 043      4103X      MOV    M+A      STORE DIGIT
061.252 361      4104X      INX    H
061.253 005      4105X      POP    PSW      RESTORE VALUE
061.254 302 240 061 4106X      DCR    B
061.257 301      4107X      JNZ    UOD1     IF MORE TO GO
061.260 311      4108X      POP    B      RESTORE (B,C)
061.261          4109X      RET
          4110      XTEXT  DU66      EXIT

```

```

          4112X **      $DU66 - UNSIGNED 16 / 16 DIVIDE.
          4113X *
          4114X *      (HL) = (BC)/(DE)
          4115X *
          4116X *      ENTRY (BC), (DE) PRESET
          4117X *      EXIT (HL) = RESULT
          4118X *      (DE) = REMAINDER
          4119X *      USES ALL
          4120X
          4121X
030.106 4122X $DU66 EQU 30106A IN H17 ROM
061.261 4123      XTEXT MU66

```

```

          4125X **      $MU66 - UNSIGNED 16X16 MULTIPLY.
          4126X *
          4127X *      ENTRY (BC) = MULTIPLICAND
          4128X *      (DE) = MULTIPLIER
          4129X *      EXIT (HL) = RESULT
          4130X *      'Z' SET IF NOT OVERFLOW
          4131X *      USES ALL
          4132X
          4133X
030.337 4134X $MU66 EQU 30337A IN H17 ROM
061.261 4135      XTEXT TBL5

```

```

4137X **      $TBLS - TABLE SEARCH
4138X *
4139X *      TABLE FORMAT
4140X *
4141X *      DB      KEY1,VAL1,
4142X *      .      .
4143X *      .      .
4144X *      DB      KEYN,VALN
4145X *      DB      0
4146X *
4147X *      ENTRY   (A) = PATTERN
4148X *              (H,L) = TABLE FWA
4149X *      EXIT    (A) = PATTERN IF FOUND
4150X *              'Z' SET IF FOUND
4151X *              'Z' CLEAR IF NOT FOUND OR PATTERN=0      /78.10.GC/
4152X *      USES    A,F,H,L
4153X *
4154X *
061.261 305      4155X $TBLS  PUSH  B
061.262 376 000  4156X      CPI    0      /78.10.GC/
061.264 312 306 061 4157X      JZ    TBL2    /78.10.GC/
061.267 107      4158X      MOV    B,A
061.270 176      4159X TBL1  MOV    A,M      (A) = CHARACTER
061.271 043      4160X      INX    H
061.272 270      4161X      CMP    B
061.273 312 310 061 4162X      JZ    TBL3      IF MATCH
061.276 247      4163X      ANA    A
061.277 043      4164X      INX    H      SKIP FAST
061.300 302 270 061 4165X      JNZ    TBL1      IF NOT END OF TABLE
061.303 053      4166X      DCX    H
061.304 053      4167X      DCX    H
061.305 257      4168X      XRA    A      SET TO ZERO FOR OLD USERS      /78.10.GC/
061.306 376 001  4169X TBL2  CPI    1      CLEAR ZERO      /78.10.GC/
4170X *
4171X *      DONE
4172X *
061.310 301      4173X TBL3  POP    B
061.311 311      4174X      RET
061.312          4175      XTEXT  TBRA

```

```

4177X **      $TBRA - BRANCH RELATIVE THROUGH TABLE.
4178X *
4179X *      $TBRA USES THE SUPPLIED INDEX TO SELECT A BYTE FROM THE
4180X *      JUMP TABLE. THE CONTENTS OF THIS BYTE ARE ADDED TO THE
4181X *      ADDRESS OF THE BYTE, YEILDING THE PROCESSOR ADDRESS.
4182X *
4183X *      CALL    $TBRA
4184X *      DB      LAB1-*      INDEX = 0 FOR LAB1
4185X *      DB      LAB2-*      INDEX = 1 FOR LAB2
4186X *      DB      LABN-*      INDEX = N-1 FOR LABN
4187X *
4188X *      ENTRY   (A) = INDEX
4189X *              (RET) = TABLE FWA

```



```

4190X *      EXIT    TO COMPUTED ADDRESS
4191X *      USES    F,H,L
4192X
4193X
031.076      4194X $TBRA EQU    31076A      IN H17 ROM
061.312      4195      XTEXT  TYPT2

```

```

4197X **      $TYPTX - TYPE TEXT.
4198X *
4199X *      $TYPTX IS CALLED TO TYPE A BLOCK OF TEXT ON THE SYSTEM CONSOLE.
4200X *
4201X *      IMBEDDED ZERO BYTES INDICATE A CARRIAGE RETURN LINE FEED,
4202X *      A BYTE WITH THE 200Q BIT SET IS THE LAST BYTE IN THE MESSAGE.
4203X *
4204X *      ENTRY    (RET) = TEXT
4205X *      EXIT     TO (RET+LENGTH)
4206X *      USES     A,F
4207X
4208X
031.136      4209X $TYPTX EQU    31136A      IN H17 ROM
4210X
031.144      4211X $TYPTX EQU    31144A      IN H17 ROM
061.312      4212      XTEXT  FOPE

```

```

4214X **      $FOPEX - OPEN FILE BLOCK FOR I/O
4215X *
4216X *      $FOPEX IS CALLED BEFORE ANY I/O IS DONE VIA A
4217X *      FILE BLOCK. $FOPEX SETS UP THE FILE BLOCK, AND OPENS
4218X *      THE FILE VIA *HDOS*.
4219X *
4220X *      ENTRY    (DE) = ADDRESS OF DEFAULT BLOCK
4221X *              (HL) = ADDRESS OF FILE BLOCK
4222X *      EXIT     TO $FERROR IF ERROR
4223X *              TO CALLER IF OK
4224X *      USES     A,F,B,C,D,E
4225X
4226X
061.312 315 337 061 4227X $FOPER CALL    $FOPER.
061.315 320      4228X      RNC
061.316 303 112 064 4229X      JMP     $FERROR      IN ERROR
4230X
061.321 315 342 061 4231X $FOPEW CALL    $FOPEW.
061.324 320      4232X      RNC
061.325 303 112 064 4233X      JMP     $FERROR      IN ERROR
4234X
061.330 315 345 061 4235X $FOPEU CALL    $FOPEU.
061.333 320      4236X      RNC
061.334 303 112 064 4237X      JMP     $FERROR      IN ERROR
4238X
4239X

```

*FOPE

061.337	076 002	4240X	\$FOPER.	MVI	A,FT.0R	FILE TYPE OF OPEN FOR READ
061.341	001	4241X		DB	0010	LXI,B TO SKIP NEXT MVI
061.342	076 004	4242X	\$FOPEW.	MVI	A,FT.0W	OPEN FOR WRITE
061.344	001	4243X		DB	0010	LXI,B TO SKIP NEXT MIV
061.345	076 006	4244X	\$FOPEU.	MVI	A,FT.0R+FT.0W	
		4245X				
		4246X	*		(A) = FILE FLAGS	
		4247X				
061.347	345	4248X		PUSH	H	SAVE FILE BLOCK ADDRESS
061.350	365	4249X		PUSH	PSW	SAVE NEW FLAGS
000.000		4250X		ERRNZ	FB.CHA	
061.351	106	4251X		MOV	B,M	(B) = CHANNEL NUMBER
061.352	305	4252X		PUSH	B	SAVE HANNEL NUMBER
000.000		4253X		ERRNZ	FB.FLG-FB.CHA-1	
061.353	043	4254X		INX	H	
061.354	117	4255X		MOV	C,A	(C) = NEW FILE FLAGS
061.355	176	4256X		MOV	A,M	(A) = CURRENT TYPE
061.356	247	4257X		ANA	A	
061.357	171	4258X		MOV	A,C	(A) = NEW FLAGS TO BE SET
061.360	312 372 061	4259X		JZ	\$FOPE1	NOT ALREADY OPEN
		4260X				
		4261X	*		ALREADY OPEN. SQUACK	
		4262X				
061.363	301	4263X		POP	B	RESTORE (BC)
061.364	361	4264X		POP	PSW	DISCARD NEW FLAGS
061.365	341	4265X		POP	H	(HL) = FB ADDRESS
061.366	076 031	4266X		MVI	A,EC.FA0	FILE ALREADY OPEN
061.370	067	4267X		STC		
061.371	311	4268X		RET		
		4269X				
000.000		4270X		ERRNZ	FB.FWA-FB.FLG-1	
061.372	043	4271X	\$FOPE1	INX	H	(HL) = \$FB.FWA
061.373	116	4272X		MOV	C,M	
061.374	043	4273X		INX	H	
061.375	106	4274X		MOV	B,M	(BC) = FB.FWA
061.376	043	4275X		INX	H	
000.000		4276X		ERRNZ	FB.PTR-FB.FWA-2	
061.377	161	4277X		MOV	M,C	SET FB.PTR = FB.FWA
062.000	043	4278X		INX	H	
062.001	160	4279X		MOV	M,B	
062.002	043	4280X		INX	H	
000.000		4281X		ERRNZ	FB.LIM-FB.PTR-2	
062.003	161	4282X		MOV	M,C	SET FB.LIM = FB.FWA
062.004	043	4283X		INX	H	
062.005	160	4284X		MOV	M,B	
062.006	043	4285X		INX	H	
000.000		4286X		ERRNZ	FB.NAM-FB.LIM-4	
062.007	043	4287X		INX	H	
062.010	043	4288X		INX	H	(HL) = \$FB.NAM
		4289X				
		4290X	*		FILE BLOCK POINTERS SETUP. OPEN FILE	
		4291X				
062.011	345	4292X		PUSH	H	SAVE NEW ADDRESS FOR NAME
062.012	041 043 062	4293X		LXI	H,\$FOPEB	
062.015	247	4294X		ANA	A	
062.016	312 025 062	4295X		JZ	\$FOPE2	/78.10.GC/

\$FOPE

```

000.000      4296X      ERRNZ      .EXIT
062.021 315 261 061 4297X      CALL      $TBLS      FIND CODE
062.024 176      4298X      MOV      A,M
062.025 062 033 062 4299X $FOPE2 STA      $FOPEA      SET SYSCALL CODE
062.030 341      4300X      POP      H      (HL) = $FB.NAM
062.031 361      4301X      POP      PSW      (A) = CHANNEL NUMBER
062.032 377 000      4302X      DB      SYSCALL,.EXIT
062.033      4303X $FOPEA EQU      *-1      SYSCALL CODE
062.034 321      4304X      POP      D      (D) = NEW FLAG
062.035 341      4305X      POP      H      (HL) = FILE BLOCK ADDRESS
062.036 330      4306X      RC      EXIT IF ERROR
062.037 043      4307X      INX      H
000.000      4308X      ERRNZ      FB.FLG-1
062.040 162      4309X      MOV      M,D      SET NEW FLAGS
062.041 053      4310X      DCX      H      RESTORE (HL)
062.042 311      4311X      RET
062.043 002 042      4312X
062.045 004 043      4313X $FOPEB DB      FT.OR,.OPENR      TABLE OF SYSCALL CODES
062.047 006 044      4314X      DB      FT.OW,.OPENW
062.051 000      4315X      DB      FT.OR+FT.OW,.OPENU
062.052      4316X      DB      0      SHOULD NOT OCCUR
062.052      4317      XTEXT      FCLO

```

```

4319X **      $FCLO - CLOSE FILE BLOCK.
4320X *
4321X *      $FCLO IS CALLED TO TERMINATE PROCESSING THROUGH A FILE
4322X *      BLOCK.
4323X *
4324X *      ENTRY      (HL) = FILE BLOCK ADDRESS
4325X *      EXIT      TO $FERROR IF ERROR
4326X *      TO CALLER IF OK
4327X *      USES      A,F,B,C,D,E
4328X
4329X

```

```

062.052 315 061 062 4330X $FCLO CALL      $FCLO.
062.055 320      4331X      RNC      NO ERROR
062.056 303 112 064 4332X      JMP      $FERROR
062.061 345      4333X
062.062 043      4334X $FCLO. PUSH      H      SAVE FILE BLOCK ADDRESS
062.063 176      4335X      ERRNZ      FB.FLG-1
062.064 066 000      4336X      INX      H      (HL) = $FB.FLG
062.065 247      4337X      MOV      A,M
062.067 312 155 062 4338X      MVI      M,0      CLEAR FLAG
062.072 346 004      4339X      ANA      A
062.074 312 147 062 4340X      JZ      $FCLO4      FILE NOT OPEN
062.074      4341X      ANI      FT.OW
062.074      4342X      JZ      $FCLO3      NO WRITING, NO FLUSHING NEEDED
062.074      4343X
062.074      4344X *      WAS OPEN FOR WRITE. SEE IF NEED FLUSH THE LAST SECTOR
062.074      4345X
062.077 315 234 030 4346X      CALL      $INDL
062.102 003 000      4347X      DW      FB.PTR-FB.FLG
062.104 325      4348X      PUSH      D      SAVE (FB.PTR)

```

```
062.105 315 234 030 4349X CALL $INDL (DE) = (FB.FWA)
062.110 001 000 4350X DW FB.FWA-FB.FLG
062.112 341 4351X POP H (HL) = (FB.PTR)
062.113 175 4352X MOV A,L
062.114 223 4353X SUB E
062.115 117 4354X MOV C,A
062.116 174 4355X MOV A,H
062.117 232 4356X SBB D
062.120 107 4357X MOV B,A (BC) = AMOUNT IN BLOCK
062.121 261 4358X ORA C
062.122 312 147 062 4359X JZ $FCLO3 NONE TO FLUSH
4360X
4361X * NEED TO FLUSH BUFFER
4362X *
4363X * (BC) = DATA AMOUNT
4364X * (DE) = FWA
4365X * (HL) = LWA+1
4366X
062.125 171 4367X MOV A,C
062.126 247 4368X ANA A
062.127 312 142 062 4369X JZ $FCLO2 DONT HAVE PARTIAL SECTOR
4370X
4371X * ZERO FILL PARTIAL SECTOR
4372X
062.132 066 000 4373X $FCLO1 MVI M,0
062.134 043 4374X INX H
062.135 014 4375X INR C
062.136 302 132 062 4376X JNZ $FCLO1
062.141 004 4377X INR B COUNT ANOTHER FULL SECTOR
062.142 341 4378X $FCLO2 POP H (HL) = FB.FWA
062.143 176 4379X MOV A,M (A) = CHANNEL NUMBER
000.000 4380X ERNZ FB.CHA
062.144 345 4381X PUSH H
062.145 377 005 4382X DB SYSCALL,.WRITE FLUSH
4383X
4384X * READY TO CLOSE FILE.
4385X *
4386X * 'C' SET IF ERROR
4387X * (A) = ERROR CODE
4388X
062.147 341 4389X $FCLO3 POP H (HL) = FILE BLOCK ADDRESS
062.150 330 4390X RC ERROR
000.000 4391X ERNZ FB.CHA
062.151 176 4392X MOV A,M (A) = CHANNEL NUMBER
062.152 345 4393X PUSH H
062.153 377 046 4394X DB SYSCALL,.CLOSE CLOSE CHANNEL
062.155 341 4395X $FCLO4 POP H (HL) = FILE BLOCK ADDRESS
062.156 311 4396X RET
062.157 4397X XTEXT FCLEAR
```

```

4399X **      $FCLEAR - CLEAR FILE BLOCK.
4400X *
4401X *      $FCLEAR CLEARS OUT A FILE BLOCK BY SETTING THE POINTERS TO
4402X *      EMPTY, AND CLEARING ANY ERROR OR EOF FLAGS.
4403X *
4404X *      THE DISK (OR WHATEVER) FILE IS NOT POSITIONED, READ, WRITEN
4405X *      OPENED OR CLOSED.
4406X *
4407X *      ENTRY (HL) = FB ADDRESS
4408X *      EXIT  NONE
4409X *      USES  A,F,B,C
4410X
4411X
062.157      4412X $FCLEAR EQU *
062.157 345   4413X PUSH H SAVE FILE BLOCK ADDRESS
000.000      4414X ERRNZ FB.FLG-FB.CHA-1
062.160 043   4415X INX H
000.000      4416X ERRNZ FB.FWA-FB.FLG-1
062.161 043   4417X INX H (HL) = $FB.FWA
062.162 116   4418X MOV C,M
062.163 043   4419X INX H
062.164 106   4420X MOV B,M (BC) = FB.FWA
062.165 043   4421X INX H
000.000      4422X ERRNZ FB.PTR-FB.FWA-2
062.166 161   4423X MOV M,C SET FB.PTR = FB.FWA
062.167 043   4424X INX H
062.170 160   4425X MOV M,B
062.171 043   4426X INX H
000.000      4427X ERRNZ FB.LIM-FB.PTR-2
062.172 161   4428X MOV M,C SET FB.LIM = FB.FWA
062.173 043   4429X INX H
062.174 160   4430X MOV M,B
062.175 341   4431X POP H (HL) = FB.FWA
062.176 311   4432X RET
062.177      4433X XTEXT $REAL

```

```

4435X **      $FREAL - READ BYTES FROM FILE BUFFER.
4436X *
4437X *      $FREAL IS CALLED TO READ A NUMBER OF BYTES FROM A FILE BUFFER.
4438X *
4439X *      ENTRY (BC) = BYTE COUNT
4440X *      (DE) = FWA FOR BYTES
4441X *      (HL) = ADDRESS OF FILE BUFFER
4442X *      EXIT  TO *FERROR* IF ERROR
4443X *      TO CALLER IF OK
4444X *      (BC) = UNREAD BYTE COUNT (ONLY IF EOF)
4445X *      (DE) = ADDRESS OF FIRST UNUSED BYTE
4446X *      'C' SET IF EOF DURING READ
4447X *      USES  A,F,B,C,D,E
4448X
4449X
062.177 315 212 062 4450X $FREAL CALL $FREAL.
062.202 320      4451X RNC RETURN IF OK

```

```

062.203 376 001 4452X CPI EC.EOF
062.205 302 112 064 4453X JNE $FERROR ERROR IS NOT EOF
062.210 067 4454X STC
062.211 311 4455X RET ERROR IS SIMPLY EOF
4456X
4457X
062.212 4458X $FREAL EQU *
062.212 013 4459X DCX B (BC) = COUNT NOT ENCLUDING 00 BYTE
062.213 257 4460X XRA A
062.214 062 111 064 4461X STA EOFLG CLEAR EOF FLAG
062.217 345 4462X PUSH H
062.220 315 335 063 4463X CALL CBT COPY BUFFER POINTERS TO TEMP CELLS
4464X
4465X * COPY DATA FROM BUFFER TO TARGET
4466X
062.223 325 4467X $REAL2 PUSH D SAVE TARGET ADDRESS
062.224 072 100 064 4468X LDA T.FLG
062.227 346 002 4469X ANI FT.0R
062.231 076 011 4470X MVI A,EC.FNO
062.233 067 4471X STC ASSUME FILE NOT OPEN
062.234 312 370 062 4472X JZ $REAL8 ERROR
062.237 170 4473X MOV A,B
062.240 261 4474X ORA C
062.241 312 370 062 4475X JZ $REAL8 ALL DONE
4476X
4477X * COMPUTE MIN( DATA IN BUFFER, DATA REQUESTED)
4478X
062.244 052 103 064 4479X $REAL3 LHLD T.PTR
062.247 353 4480X XCHG (DE) = (FB.PTR) = ADDRESS OF DATA
062.250 052 105 064 4481X LHLD T.LIM (HL) = LIMIT ADDRESS
062.253 175 4482X MOV A,L
062.254 223 4483X SUB E
062.255 157 4484X MOV L,A
062.256 174 4485X MOV A,H
062.257 232 4486X SBB D
062.260 147 4487X MOV H,A (HL) = NUMBER OF BYTES IN BUFFER
062.261 171 4488X MOV A,C
062.262 225 4489X SUB L COMPARE TO REQUESTED COUNT
062.263 170 4490X MOV A,B
062.264 234 4491X SBB H
062.265 322 272 062 4492X JNC $REAL4 LESS THAN REQUESTED COUNT
062.270 140 4493X MOV H,B
062.271 151 4494X MOV L,C DONT TRANSFER MORE THAN LIMIT
062.272 174 4495X $REAL4 MOV A,H
062.273 265 4496X ORA L
062.274 302 310 062 4497X JNZ $REAL6 SOME IN BUFFER
4498X
4499X * BUFFER IS EMPTY, RE-FILL IT
4500X
062.277 315 015 064 4501X CALL $FFB FILL FILE BUFFER
062.302 332 370 062 4502X JC $REAL8 ERROR CONDITION
062.305 303 244 062 4503X JMP $REAL3 COUNT THE DATA
4504X
4505X * GOT THE DATA, MOVE IT FROM BUFFER TO TARGET
4506X *
4507X * (BC) = LIMIT COUNT

```

```

4508X *      (DE) = FROM
4509X *      (HL) = COUNT
4510X *      ((SP)) = TO
4511X
062.310 171 4512X $REAL6 MOV A,C
062.311 225 4513X SUB L
062.312 117 4514X MOV C,A
062.313 170 4515X MOV A,B
062.314 234 4516X SBB H
062.315 107 4517X MOV B,A      REMOVE BYTES ABOUT TO BE MOVED FROM REQUEST COUNT
062.316 305 4518X PUSH B
062.317 343 4519X XTHL      (HL) = REMAINING REQUEST COUNT
062.320 301 4520X POP B      (BC) = COUNT FOR THIS COPY
062.321 343 4521X XTHL      (HL) = TARGET ADDR, ((SP)) = REMAINING REQ. COUNT
062.322 032 4522X $REAL7 LDAX D
062.323 023 4523X INX D
062.324 167 4524X MOV M,A
062.325 043 4525X INX H
062.326 247 4526X ANA A      SEE IF 00 BYTE
062.327 302 336 062 4527X JNZ $REL7.3      NOT 00
4528X
4529X *      IS 00 BYTE, IGNORE IT
4530X
062.332 343 4531X XTHL
062.333 043 4532X INX H      ADD ONE TO UNREQUESTED COUNT
062.334 343 4533X XTHL
062.335 053 4534X DCX H      BACKSPACE OVER CHARACTER
062.336 013 4535X $REL7.3 DCX B
062.337 376 012 4536X CPI NL
062.341 312 361 062 4537X JE $REL7.5      IS END OF LINE
062.344 170 4538X MOV A,B
062.345 261 4539X ORA C
062.346 302 322 062 4540X JNZ $REAL7      MORE TO GO
062.351 353 4541X XCHG
062.352 042 103 064 4542X SHLD T,PTR      UPDATE POINTER
062.355 301 4543X POP B      (BC) = REMAINING COUNT
062.356 303 223 062 4544X JMP $REAL2      SEE IF MORE IN BUFFER
4545X
4546X *      END OF CODED LINE
4547X
062.361 353 4548X $REL7.5 XCHG
062.362 033 4549X DCX D      BACK OVER NL CHARACTER
062.363 042 103 064 4550X SHLD T,PTR      UPDATE POINTER
062.366 301 4551X POP B      (BC) = REMAINING COUNT
062.367 325 4552X PUSH D      SAVE TARGET LWA
4553X
4554X *      READ COMPLETE.
4555X *
4556X *      (PSW) = COMPLETION FLAGS
4557X
062.370 321 4558X $REAL8 POP D      RESTORE TARGET ADDRESS
062.371 365 4559X PUSH PSW      SAVE RETURN CODE
062.372 257 4560X XRA A
062.373 022 4561X STAX D      FLAG END OF LINE
062.374 361 4562X POP PSW      RESTORE RESULT FLAGS
062.375 023 4563X INX D      POINT TO NEXT FREE

```

```
062.376 341 4564X $REAL9 POP H
062.377 303.363.063 4565X JMP CTR COPY TEMP POINTERS BACK TO BLOCK, EXIT
063.002 4566 XTEXT FWRIL
```

```
4568X ** $FWRIL - WRITE LINE FROM FILE BUFFER.
4569X *
4570X * $FWRIL IS CALLED TO WRITE A LINE TO A FILE BUFFER.
4571X *
4572X * ENTRY (DE) = FWA FOR BYTES
4573X * (HL) = ADDRESS OF FILE BUFFER
4574X * EXIT TO *FERROR* IF ERROR
4575X * TO CALLER IF OK
4576X * (DE) = ADDRESS OF FIRST UNWRITTEN BYTE
4577X * USES A,F,B,C,D,E
4578X
4579X
```

```
063.002 315.011.063 4580X $FWRIL CALL $FWRIL
063.005 320 4581X RNC RETURN IF OK
063.006 303.112.064 4582X JMP $FERROR ERROR
```

```
4583X
4584X * SCAN FOR END OF LINE
4585X
```

```
063.011 325 4586X $FWRIL PUSH D SAVE LINE POINTER
063.012 001.377.377 4587X LXI B,-1 (BC) = COUNT
063.015 032 4588X $FWRIL1 LDAX D
063.016 023 4589X INX D
063.017 003 4590X INX B
063.020 247 4591X ANA A
063.021 302.015.063 4592X JNZ $FWRIL1 MORE TO GO
063.024 321 4593X POP D
063.025 315.047.063 4594X CALL $FWRIB WRITE BYTES
063.030 330 4595X RC ERROR
```

```
4596X
4597X * WRITE 'NL' CHARACTER
```

```
4598X
4599X
4600X
063.031 023 4601X INX D
063.032 325 4602X PUSH D
063.033 001.001.000 4603X LXI B,1
063.036 021.046.063 4604X LXI D,$FWRILA
063.041 315.047.063 4605X CALL $FWRIB
063.044 321 4606X POP D
063.045 311 4607X RET
4608X
063.046 012 4609X $FWRILA DB NL
063.047 4608 XTEXT FWRIB
```



```

4610X **      $FWRIB - WRITE BYTES FROM FILE BUFFER.
4611X *
4612X *      $FWRIB IS CALLED TO WRITE A NUMBER OF BYTES FROM A FILE BUFFER.
4613X *
4614X *      ENTRY (BC) = BYTE COUNT
4615X *              (DE) = FWA FOR BYTES
4616X *              (HL) = ADDRESS OF FILE BUFFER
4617X *      EXIT TO *FERROR* IF ERROR
4618X *              TO CALLER IF OK
4619X *              (DE) = ADDRESS OF FIRST UNWRITTEN BYTE
4620X *      USES A,F,B,C,D,E
4621X
4622X
063.047 315 056 063 4623X $FWRIB CALL $FWRIB.
063.052 320 4624X RNC RETURN IF OK
063.053 303 112 064 4625X JMP $FERROR ERROR
4626X
4627X
063.056 4628X $FWRIB. EQU *
063.056 345 4629X PUSH H
063.057 315 335 063 4630X CALL CBT COPY BUFFER POINTERS TO TEMP CELLS
4631X
4632X *      COPY DATA FROM USER AREA TO BUFFER
4633X
063.062 325 4634X $WRIB2 PUSH D SAVE AREA ADDRESS
063.063 072 100 064 4635X LDA T,FLG
063.066 346 004 4636X ANI FT,OW SEE IF OPEN FOR WRITE
063.070 312 224 063 4637X JZ $WRIB8 FILE NOT OPEN FOR WRITE
063.073 170 4638X MOV A,B
063.074 261 4639X ORA C
063.075 312 224 063 4640X JZ $WRIB8 ALL DONE
4641X
4642X *      COMPUTE MINC ROOM IN BUFFER, WRITE COUNT REQUESTED
4643X
063.100 052 103 064 4644X $WRIB3 LHL D T,PTR
063.103 353 4645X XCHG (DE) = (FB,PTR) = ADDRESS OF ROOM
063.104 052 107 064 4646X LHL D T,LWA (HL) = LIMIT ADDRESS
063.107 175 4647X MOV A,L
063.110 223 4648X SUB E
063.111 157 4649X MOV L,A
063.112 174 4650X MOV A,H
063.113 232 4651X SBB D
063.114 147 4652X MOV H,A (HL) = BYTES OF ROOM IN BUFFER
063.115 171 4653X MOV A,C COMPARE REQUESTED COUNT TO BUFFER ROOM
063.116 225 4654X SUB L
063.117 170 4655X MOV A,B
063.120 234 4656X SBB H
063.121 322 126 063 4657X JNC $WRIB4 MORE REQUESTED THEN ROOM
063.124 140 4658X MOV H,B
063.125 151 4659X MOV L,C USE REQUESTED COUNT
063.126 174 4660X $WRIB4 MOV A,H
063.127 265 4661X ORA L
063.130 302 170 063 4662X JNZ $WRIB6 SOME ROOM IN BUFFER
4663X
4664X *      BUFFER IS FULL, EMPTY IT
4665X

```

```

063.133 305      4666X      PUSH      B      SAVE COUNT
063.134 052 101 064 4667X      LHLD      T,FWA
063.137 042 103 064 4668X      SHLD      T,PTR      CLEAR REMOVAL POINTER
063.142 353      4669X      XCHG
063.143 052 107 064 4670X      LHLD      T,LWA
063.146 175      4671X      MOV       A,L
063.147 223      4672X      SUB       E
063.150 117      4673X      MOV       C,A
063.151 174      4674X      MOV       A,H
063.152 232      4675X      SBB      D
063.153 107      4676X      MOV       B,A      (BC) = DATA IN BUFFER
063.154 072 077 064 4677X      LDA       T,CHA
063.157 377 005      4678X      DB       SYSCALL,,WRITE WRITE BUFFER
063.161 301      4679X      POP       B      (BC) = DESIRED COUNT
063.162 322 100 063 4680X      JNC      $WRIB3      GOT THE DATA
063.165 303 224 063 4681X
063.165 303 224 063 4682X *      ERROR ON WRITE.
063.165 303 224 063 4683X
063.165 303 224 063 4684X      JMP      $WRIB8      HAVE ERROR
063.165 303 224 063 4685X
063.165 303 224 063 4686X *      GOT THE DATA. MOVE IT FROM BUFFER TO TARGET
063.165 303 224 063 4687X *
063.165 303 224 063 4688X *      (BC) = REQUEST COUNT
063.165 303 224 063 4689X *      (DE) = TO
063.165 303 224 063 4690X *      (HL) = COUNT
063.165 303 224 063 4691X *      ((SP)) = FROM
063.165 303 224 063 4692X
063.170 171      4693X $WRIB6 MOV      A,C
063.171 225      4694X      SUB      L
063.172 117      4695X      MOV      C,A
063.173 170      4696X      MOV      A,B
063.174 234      4697X      SBB      H
063.175 107      4698X      MOV      B,A      REMOVE BYTES ABOUT TO BE MOVED FROM REQUEST COUNT
063.176 305      4699X      PUSH     B
063.177 343      4700X      XTHL
063.200 301      4701X      POP      B      (HL) = REMAINING REQUEST COUNT
063.201 343      4702X      XTHL      (BC) = COUNT FOR THIS COPY
063.202 176      4703X $WRIB7 MOV      A,M      (HL) = TARGET ADDR, ((SP)) = REMAINING REQ. COUNT
063.203 022      4704X      STAX     D
063.204 023      4705X      INX      D
063.205 043      4706X      INX      H
063.206 013      4707X      DCX     B
063.207 170      4708X      MOV      A,B
063.210 261      4709X      ORA      C
063.211 302 202 063 4710X      JNZ     $WRIB7      MORE TO GO
063.214 353      4711X      XCHG
063.215 042 103 064 4712X      SHLD     T,PTR      UPDATE POINTER
063.220 301      4713X      POP      B      (BC) = REMAINING COUNT
063.221 303 062 063 4714X      JMP     $WRIB2      SEE IF MORE IN BUFFER
063.221 303 062 063 4715X
063.221 303 062 063 4716X *      WRITE COMPLETE.
063.221 303 062 063 4717X *
063.221 303 062 063 4718X *      (PSW) = COMPLETION FLAGS
063.221 303 062 063 4719X
063.224 321      4720X $WRIB8 POP      D      RESTORE TARGET ADDRESS
063.225 341      4721X      POP      H

```

063,226 303 363 063 4722X JMP CTR COPY TEMP POINTERS BACK TO BLOCK, EXIT

4724X ** \$FWBRK - BREAKOUTPUT /80,02,GC/

4725X *

4726X * \$FWBRK empties the specified buffer by filling it with NULLs
4727X * and then writing it. Note this is used to insure that block
4728X * mode I/O is output if it is not really a serial device (es.
4729X * writing to AT: from *EDIT*.

4730X *

4731X *

4732X * ENTRY: HL = FILE BLOCK POINTER

4733X *

4734X * EXIT: HL = FILE BLOCK POINTER

4735X * TO \$FERROR IF ERROR

4736X *

4737X * USES: PSW,BC,DE

4738X *

4739X *

063,231 315 240 063 4740X \$FWBRK CALL \$FWBRK,

063,234 320 4741X RNC NO ERROR

4742X *

063,235 303 112 064 4743X JMP \$FERROR

4744X *

063,240 345 4745X \$FWBRK, PUSH H

063,241 315 335 063 4746X CALL CBT COPY BUFFER TO TEMPORARY

063,244 315 254 063 4747X CALL \$FWBRK1

063,247 341 4748X POP H

063,250 315 363 063 4749X CALL CTR COPY TEMPORARY TO BUFFER

063,253 311 4750X RET

4751X *

063,254 052 107 064 4752X \$FWBRK1 LHLD T,LWA

063,257 353 4753X XCHG DE = BUFFER LWA

063,260 052 103 064 4754X LHLD T,PTR HL = BUFFER PTR

063,263 173 4755X MOV A,E

063,264 225 4756X SUB L

063,265 117 4757X MOV C,A

063,266 172 4758X MOV A,D

063,267 234 4759X SBB H

063,270 107 4760X MOV B,A BC = DE - HL

063,271 261 4761X ORA C

063,272 310 4762X RZ THE BUFFER IS ALREADY FLUSHED

4763X *

4764X * FILL THE BUFFER WITH NULLS

4765X *

063,273 170 4766X \$FWBRK2 MOV A,B

063,274 261 4767X ORA C

063,275 312 307 063 4768X JZ \$FWBRK3 NO MORE LEFT TO FILL

4769X *

063,300 066 000 4770X MVI M,0

063,302 043 4771X INX H

063,303 013 4772X DCX B

063,304 303 273 063 4773X JMP \$FWBRK2

4774X *

```

063.307 052 101 064 4775X FWBRK3 LHLD T,FWA
063.312 042 103 064 4776X SHLD T,PTR
063.315 353 4777X XCHG
063.316 052 107 064 4778X LHLD T,LWA DE = BUFFER FWA
063.321 175 4779X MOV HL = BUFFER LWA
063.322 223 4780X SUB E
063.323 117 4781X MOV C,A
063.324 174 4782X MOV A,H
063.325 232 4783X SBB D
063.326 107 4784X MOV B,A BC = HL - DE ( BC = COUNT )
063.327 072 077 064 4785X LDA T,CHA
063.332 377 005 4786X DB SYSCALL,,WRITE
063.334 311 4787X RET
063.335 4788 XTEXT FUTIL

```

```

4790X ** $FUTIL - UTILITY ROUTINES FOR FILE BLOCK ROUTINES.
4791X
4792X ** CBT - COPY BLOCK POINTERS TO TEMP CELLS.
4793X *
4794X * ENTRY (HL) = FILE BLOCK FWA
4795X * EXIT NONE
4796X * USES A,F,H,L
4797X
063.335 325 4798X CBT PUSH D
063.336 305 4799X PUSH B SAVE REGISTERS
000.000 4800X ERNZ TLEN-10 ASSUME 10 BYTES TO MOVE
063.337 021 077 064 4801X LXI D,T,CHA (DE) = TARGET FOR MOVE
063.342 006 005 4802X MVI B,10/2
063.344 176 4803X CBT1 MOV A,M COPY FILE BUFFER INTO WORK AREA
063.345 022 4804X STAX D
063.346 043 4805X INX H
063.347 023 4806X INX D
063.350 176 4807X MOV A,M
063.351 022 4808X STAX D
063.352 043 4809X INX H
063.353 023 4810X INX D
063.354 005 4811X DCR B
063.355 302 344 063 4812X JNZ CBT1 MORE TO GO
063.360 301 4813X POP B
063.361 321 4814X POP D (DE) = DATA TARGET ADDRESS
063.362 311 4815X RET
4816X
4817X
4818X ** CTB - COPY TEMP CELLS BACK TO FILE BLOCK.
4819X *
4820X * ENTRY (HL) = FILE BLOCK ADDRESS
4821X * EXIT NONE
4822X * USES NONE
4823X
063.363 365 4824X CTB PUSH PSW
063.364 325 4825X PUSH D
063.365 305 4826X PUSH B
063.366 345 4827X PUSH H SAVE REGISTERS

```

063.367	006 004	4828X	MOI	B,B/2	
063.371	021 077 064	4829X	LXI	D,T,CHA	
063.374	032	4830X	LDAX	D	
063.375	167	4831X	MOV	M,A	
063.378	023	4832X	INX	D	
063.377	043	4833X	INX	H	
064.000	032	4834X	LDAX	D	
064.001	167	4835X	MOV	M,A	
064.002	023	4836X	INX	D	
064.003	043	4837X	INX	H	
064.004	005	4838X	DCR	B	
064.005	302 374 063	4839X	JNZ	CTB1	RESTORE FILE BUFFER VALUES
064.010	341	4840X	POP	H	
064.011	301	4841X	POP	B	
064.012	321	4842X	POP	D	
064.013	361	4843X	POP	PSW	
064.014	311	4844X	RET		

4846X ** \$FFB - FILE FILE BUFFER.
4847X *
4848X * \$FFB FILLS THE FILE BUFFER BY READING FROM THE FILE.
4849X *

4850X * ENTRY NONE
4851X * EXIT %C SET IF READ INCOMPLETE
4852X * (A) = ERROR CODE
4853X * %C CLEAR IF READ COMPLETE
4854X * DATA IN BUFFER
4855X * USES A,F,D,E,H,L

064.015	072 111 064	4858X	\$FFB	LDA	EOFFLG
064.020	037	4859X		RAR	
064.021	330	4860X		RC	EOF

4861X
4862X * CAN READ MORE. DO SO
4863X

064.022	305	4864X	PUSH	B	SAVE COUNT
064.023	052 101 064	4865X	LHLD	T,FWA	
064.026	042 103 064	4866X	SHLD	T,PTR	CLEAR REMOVAL POINTER
064.031	353	4867X	XCHG		
064.032	052 107 064	4868X	LHLD	T,LWA	
064.035	042 105 064	4869X	SHLD	T,LIM	SET DATA LIMIT
064.040	175	4870X	MOV	A,L	
064.041	223	4871X	SUB	E	
064.042	117	4872X	MOV	C,A	
064.043	174	4873X	MOV	A,H	
064.044	232	4874X	SBR	D	
064.045	107	4875X	MOV	B,A	(BC) = ROOM IN BUFFER
064.046	072 077 064	4876X	LDA	T,CHA	
064.051	377 004	4877X	DB	SYSCALL,.READ	READ BUFFER
064.053	120	4878X	MOV	D,B	(D) = SECTORS UNREAD
064.054	301	4879X	POP	B	(BC) = DESIRED COUNT
064.055	320	4880X	RNC		GOT THE DATA

```
4881X
4882X *      ERROR ON READ, SEE IF EOF
4883X
4884X      RAL
4885X      STA      EOFLG      SET EOF, WE HOPE
4886X      CPI      EC.EOF*2+1
4887X      RAR
4888X      RNE
4889X      LDA      T.LIM+1      IS NOT EOF, RETURN NOW!
4890X      SUB      D
4891X      STA      T.LIM+1      SET AMOUNT OF DATA WE DID GET
4892X      ANA      A
4893X      RET
4894X
4895X
```

```
4896X **      TEMP CELLS TO HOLD FILE BLOCK POINTERS DURING I/O
4897X
4898X      ERRNZ      FB.CHA
4899X T.CHA      DB      0      CHANNEL NUMBER
4900X      ERRNZ      *-T.CHA-FB.FLG
4901X T.FLG      DB      0      FLAG BYTE
4902X      ERRNZ      *-T.CHA-FB.FWA
4903X T.FWA      DW      0
4904X      ERRNZ      *-T.CHA-FB.PTR
4905X T.PTR      DW      0
4906X      ERRNZ      *-T.CHA-FB.LIM
4907X T.LIM      DW      0
4908X      ERRNZ      *-T.CHA-FB.LWA
4909X T.LWA      DW      0
4910X T.LEN      EQU      *-T.CHA      LENGTH OF TEMP CELLS
4911X
4912X EOFLG      DB      0
4913      XTEXT      FERROR
```

```
4915X **      $FERROR - PROCESS FILE ERRORS.
4916X *
4917X *      $FERROR IS CALLED TO COMPLAIN ABOUT AN ERROR ENCOUNTERED
4918X *      WHEN PROCESSING FILES.
4919X *
4920X *      ENTRY      (A) = ERROR CODE
4921X *      (HL) = ADDRESS OF FILE NAME - FB.NAM
4922X *      EXIT      TO RESTART
4923X *      USES      ALL
4924X
4925X
4926X $FERROR PUSH      PSW      SAVE CODE
4927X      CALL      $TYPTX
4928X      DB      NL,BELL,'ERROR ON FILE','+2000
4929X      LXI      D,FB.NAM
4930X      DAD      D
4931X
4932X *      PRINT FILE NAME
4933X
```

064.142	176	4934X	\$FERR1	MOV	A,M	
064.143	043	4935X		INX	H	ADVANCE MESSAGE
064.144	247	4936X		ANA	A	
064.145	312 156 064	4937X		JZ	\$FERR2	
064.150	315 136 061	4938X		CALL	\$WCHAR	
064.153	303 142 064	4939X		JMP	\$FERR1	
		4940X				
		4941X	*			TYPE ERROR MESSAGE
		4942X				
064.156	315 136 031	4943X	\$FERR2	CALL	\$TYPTX	
064.161	040 055 240	4944X		DB	'-',','+2000	
064.164	046 012	4945X		MVI	H,NL	
064.166	361	4946X		POP	PSW	(A) = CODE
064.167	377 057	4947X		DB	SYSCALL,.ERROR	
064.171	303 340 042	4948X		JMP	RESTART	EXIT

```

4951 **      OPCTAB - OP CODE TABLE.
4952 *
4953 *      OPCTAB CONTAINS AN ENTRY FOR EACH OP CODE.
4954 *      THE TABLE IS SEARCHED SERIALY, SO THE MOST HEAVILY
4955 *      USED OP CODES SHOULD BE PLACED TOWARDS THE FRONT.
4956 *
4957 *      THE TABLE FORMAT IS:
4958 *
4959 *      DB      'OPCODE'      CHARACTERS 1 - N-1
4960 *      'E'+80H      LAST CHARACTER HAS HIGH BIT SET
4961 *      F            =1 IF TO ASSEMBLE REGARDLESS OF *IF*
4962 *      F            =1 IF NOT TO AUTOMATICALLY DEFINE LABEL
4963 *      I I I I I    = OP CODE TYPE INDEX
4964 *      CODE         OP CODE, IF MACHINE OP
4965 *                  IF PSEUDO OP, =0 IF IN GROUP 1
4966
000.200      4967 OF.CE EQU      200Q      CONDITIONAL ASSEMBLY EXCEPTION
000.100      4968 OF.LD EQU      100Q      DEFER LABEL DEFINITION
4969
4970
064.174      4971 OPCTAB EQU      *
064.174      4972 DB      'AC', 'I'+80H,1,316Q
064.201      4973 DB      'AD', 'C'+80H,3,210Q
064.206      4974 DB      'AD', 'D'+80H,3,200Q
064.213      4975 DB      'AD', 'I'+80H,1,306Q
064.220      4976 DB      'AN', 'A'+80H,3,240Q
064.225      4977 DB      'AN', 'I'+80H,1,346Q
064.232      4978 DB      'CAL', 'L'+80H,2,315Q
064.240      4979 DB      'C', 'C'+80H,2,334Q
064.244      4980 DB      'C', 'M'+80H,2,374Q
064.250      4981 DB      'CM', 'A'+80H,0,057Q
064.255      4982 DB      'CM', 'C'+80H,0,077Q
064.262      4983 DB      'CM', 'P'+80H,3,270Q
064.267      4984 DB      'CN', 'C'+80H,2,324Q
064.274      4985 DB      'CN', 'Z'+80H,2,304Q
064.301      4986 DB      'C', 'P'+80H,2,364Q
064.305      4987 DB      'CP', 'E'+80H,2,354Q
064.312      4988 DB      'CP', 'I'+80H,1,376Q
064.317      4989 DB      'CP', 'O'+80H,2,344Q
064.324      4990 DB      'C', 'Z'+80H,2,314Q
064.330      4991 DB      'DA', 'A'+80H,0,047Q
064.335      4992 DB      'DA', 'D'+80H,7,011Q
064.342      4993 DB      'D', 'B'+80H,13,1
064.346      4994 DB      'DC', 'R'+80H,4,005Q
064.353      4995 DB      'DC', 'X'+80H,7,013Q
064.360      4996 DB      'D', 'I'+80H,0,363Q
064.364      4997 DB      'D', 'S'+80H,14,0
064.370      4998 DB      'D', 'W'+80H,15,1
064.374      4999 DB      'E', 'I'+80H,0,373Q
065.000      5000 DB      'EJEC', 'T'+80H,0F.LD+16,0
065.007      5001 DB      'ELS', 'E'+80H,0F.LD+17+0F.CE,0
065.015      5002 DB      'EN', 'D'+80H,0F.LD+18+0F.CE,1
065.022      5003 DB      'ENDI', 'F'+80H,0F.LD+19+0F.CE,0
065.031      5004 DB      'EQ', 'U'+80H,0F.LD+20,0
065.036      5005 DB      'ERRM', 'I'+80H,0F.LD+21,3
065.045      5006 DB      'ERRN', 'Z'+80H,0F.LD+21,1

```


065.054	105	122	122	5007	DB	'ERRP', 'L'+80H, 0F, LD+21, 2
065.063	105	122	122	5008	DB	'ERRZ', 'R'+80H, 0F, LD+21, 0
065.072	110	114	324	5009	DB	'HL', 'T'+80H, 0, 166Q
065.077	111	306	126	5010	DB	'I', 'F'+80H, 0F, LD+22, 0
065.103	111	316	001	5011	DB	'I', 'N'+80H, 1, 333Q
065.107	111	116	322	5012	DB	'IN', 'R'+80H, 4, 004Q
065.114	111	116	330	5013	DB	'IN', 'X'+80H, 7, 003Q
065.121	112	303	002	5014	DB	'J', 'C'+80H, 2, 332Q
065.125	112	305	002	5015	DB	'J', 'E'+80H, 2, 312Q
065.131	112	315	002	5016	DB	'J', 'M'+80H, 2, 372Q
065.135	112	115	320	5017	DB	'JM', 'P'+80H, 2, 303Q
065.142	112	116	303	5018	DB	'JN', 'C'+80H, 2, 322Q
065.147	112	116	305	5019	DB	'JN', 'E'+80H, 2, 302Q
065.154	112	116	332	5020	DB	'JN', 'Z'+80H, 2, 302Q
065.161	112	320	002	5021	DB	'J', 'P'+80H, 2, 362Q
065.165	112	120	305	5022	DB	'JP', 'E'+80H, 2, 352Q
065.172	112	120	317	5023	DB	'JP', 'D'+80H, 2, 342Q
065.177	112	332	002	5024	DB	'J', 'Z'+80H, 2, 312Q
065.203	114	104	301	5025	DB	'LD', 'A'+80H, 2, 072Q
065.210	114	104	101	5026	DB	'LDA', 'X'+80H, 9, 012Q
065.216	114	110	114	5027	DB	'LHL', 'D'+80H, 2, 052Q
065.224	114	117	306	5028	DB	'LO', 'F'+80H, 0F, LD+23, 0
065.231	114	117	316	5029	DB	'LO', 'N'+80H, 0F, LD+24, 0
065.236	114	130	311	5030	DB	'LX', 'I'+80H, 11, 001Q
065.243	115	117	326	5031	DB	'MO', 'O'+80H, 12, 100Q
065.250	115	126	311	5032	DB	'MV', 'I'+80H, 8, 006Q
065.255	116	117	320	5033	DB	'NO', 'P'+80H, 0, 000Q
065.262	117	122	301	5034	DB	'OR', 'A'+80H, 3, 260Q
065.267	117	122	307	5035	DB	'OR', 'G'+80H, 0F, LD+25, 0
065.274	117	122	311	5036	DB	'OR', 'I'+80H, 1, 366Q
065.301	117	125	324	5037	DB	'OU', 'T'+80H, 1, 323Q
065.306	120	103	110	5038	DB	'PCH', 'L'+80H, 0, 351Q
065.314	120	117	320	5039	DB	'PO', 'P'+80H, 6, 301Q
065.321	120	125	123	5040	DB	'PUS', 'H'+80H, 6, 305Q
065.327	122	101	314	5041	DB	'RA', 'L'+80H, 0, 027Q
065.334	122	101	322	5042	DB	'RA', 'R'+80H, 0, 037Q
065.341	122	303	000	5043	DB	'R', 'C'+80H, 0, 330Q
065.345	122	305	000	5044	DB	'R', 'E'+80H, 0, 310Q
065.351	122	105	324	5045	DB	'RE', 'T'+80H, 0, 311Q
065.356	122	114	303	5046	DB	'RL', 'C'+80H, 0, 007Q
065.363	122	315	000	5047	DB	'R', 'M'+80H, 0, 370Q
065.367	122	116	303	5048	DB	'RN', 'C'+80H, 0, 320Q
065.374	122	116	305	5049	DB	'RN', 'E'+80H, 0, 300Q
066.001	122	116	332	5050	DB	'RN', 'Z'+80H, 0, 300Q
066.006	122	320	000	5051	DB	'R', 'P'+80H, 0, 360Q
066.012	122	120	305	5052	DB	'RP', 'E'+80H, 0, 350Q
066.017	122	120	317	5053	DB	'RP', 'D'+80H, 0, 340Q
066.024	122	122	303	5054	DB	'RR', 'C'+80H, 0, 017Q
066.031	122	123	324	5055	DB	'RS', 'T'+80H, 10, 307Q
066.036	122	332	000	5056	DB	'R', 'Z'+80H, 0, 310Q
066.042	123	102	302	5057	DB	'SB', 'B'+80H, 3, 230Q
066.047	123	102	311	5058	DB	'SB', 'I'+80H, 1, 336Q
066.054	123	103	101	5059	DB	'SCAL', 'L'+80H, 1, 377Q
066.063	123	105	324	5060	DB	'SE', 'T'+80H, 0F, LD+26, 0
066.070	123	110	114	5061	DB	'SHL', 'D'+80H, 2, 042Q
066.076	123	120	101	5062	DB	'SPAC', 'E'+80H, 0F, LD+27, 0

066.105	123 120 110	5063	DB	'SPH', 'L'+80H,0,3710
066.113	123 124 301	5064	DB	'ST', 'A'+80H,2,0620
066.120	123 124 101	5065	DB	'STA', 'X'+80H,9,0020
066.126	123 124 303	5066	DB	'ST', 'C'+80H,0,0670
066.133	123 124 314	5067	DB	'ST', 'L'+80H,0F.LD+28,0
066.140	123 125 302	5068	DB	'SU', 'B'+80H,3,2200
066.145	123 125 311	5069	DB	'SU', 'I'+80H,1,3260
066.152	124 111 124	5070	DB	'TITL', 'E'+80H,0F.LD+29,0
066.161	130 103 110	5071	DB	'XCH', 'G'+80H,0,3530
066.167	130 122 301	5072	DB	'XR', 'A'+80H,3,2500
066.174	130 122 311	5073	DB	'XR', 'I'+80H,1,3560
066.201	130 124 110	5074	DB	'XTH', 'L'+80H,0,3430
066.207	103 117 104	5075	DB	'COD', 'E'+80H,0F.LD+30,0
066.215	105 116 103	5076	DB	'ENCL', 'U'+80H,0F.LD+31,0
066.224	130 124 105	5077	DB	'XTEX', 'T'+80H,0F.LD+31,0
		5078		
066.233	000	5079	DB	0 END OF TABLE

5082 ** THE FOLLOWING AREAS ARE ASSEMBLY AREAS FOR LISTING LINES:

5084 ** HEADING LINE:

5085 *

5086

5087

066.234 5088 HEADING EQU * START OF PAGE HEADING

066.234 040 040 040 5089 TTLTXT DB /

000.062 5090 TTXTL EQU *-TTLTXT LENGTH

066.316 011 110 105 5091 DB / HEATH ASM #104.05.00'

066.343 012 5092 DB NL NEW LINE /78.10.GC/

5093

066.344 040 040 040 5094 STLTXT DB /

000.063 5095 STXTL EQU *-STLTXT LENGTH

067.027 011 5096 DB TAB

067.030 060 066 055 5097 HEADC DB '06-DEC-79'

000.011 5098 HEADCL EQU *-HEADC

067.041 040 040 040 5099 DB / Page /

067.052 000 5100 HEADA DB 0

067.053 000 5101 HEADR DB 0

067.054 012 012 5102 DB NL,NL 2 BLANK LINES

000.222 5103 HEADLEN EQU *-HEADING HEADER LENGTH

5105 ** LISTING LINE WORK AREA

5106

5107

067.056 5108 DSPLIN DS 0

067.056 040 040 040 5109 DB / ERROR FLAGS

067.061 040 040 040 5110 DSPLNA DB / BANK NUMBER

067.064 056 5111 DB /

067.065 040 040 040 5112 DB / BANK ADDRESS

067.070 040 040 5113 DB /

067.072 040 040 040 5114 DSPLNB DB / BYTE 1

067.075 040 5115 DB /

067.076 040 040 040 5116 DB / BYTE 2

067.101 040 5117 DB /

067.102 040 040 040 5118 DSPLNC DB / BYTE 3

067.105 040 5119 DB /

067.106 5120 DSPLIM EQU * LIMIT FOR OCTAL BYTES

067.106 060 060 060 5121 DSPLND DB '00000' LINE NUMBER /80.02.GC/

067.113 130 5122 DSPLNE DB 'X' XTEXT FLAG /80.02.GC/

067.114 040 040 5123 DB / /80.02.GC/

000.040 5124 DSPLN EQU *-DSPLIN LENGTH OF HEADER

067.116 000 5125 DB 0 TERMINATES DSPLIN FOR \$DTB

067.117 200 5126 DB 2000 TERMINATES DSPLIN FOR .PRINT

067.120	000 000	5129	OBFPTR	DW	0	BYTE DECODE POINTER
		5130				
067.122	000	5131	PASS	DB	0	PASS NUMBER
067.123	000 000	5132	ERRCNT	DW	0	NUMBER OF ERRORS IN PASS
067.125	000	5133	ERRSHD	DB	0	<0 IF TO TYPE ERRORS ON CONSOLE
067.126	000 000	5134	STATNO	DW	0	STATEMENT NUMBER
067.130	000	5135	LSTCTL	DB	0	LISTING CONTROL OPTIONS
067.131	000	5136	LSTCTLS	DB	0	FORCED SET LISTING CONTROL
067.132	000	5137	LSTCTLG	DB	0	FORCED CLEAR LISTING CONTROL (INVERTED MASK)
067.133	000	5138	ENDFLG	DB	0	NON-ZERO IF END STATEMENT READ
067.134	000 000	5139	ORG	DW	0	ORIGIN POINTER
067.136	000 000	5140	SORG	DW	0	VALUE OF *ORG* AT BEGINNING OF STATEMENT
067.140	000	5141	ERRFLG	DB	0	ERROR FLAGS FOR THIS STATEMENT
067.141	000	5142	GRPFLG	DB	0	<0 IF HAVE ASSEMBLED 2ND GROUP INSTRUCTIONS
067.142	000	5143	XTXFLG	DB	0	<0 IF READING FROM XTEXT
067.143	000	5144	XTXLINE	DB	0	<0 IF CURRENT LINE FROM XTEXT
		5145				
		5146	*			CODE GENERATION FLAG
		5147				
067.144	000	5148	FTFLAG	DB	0	FILE TYPE (FT.PIC, FT.ABS)
067.145	000	5149	RELFLG	DB	0	ST.REL IF RELOCATABLE ASSEMBLY
067.146	000	5150	CODEFLG	DB	0	<0 IF 'CODE' PSEUDO ENCOUNTERED THIS PASS
		5151				
067.147	000	5152	LARGE	DB	0	<0 IF TO SWAP OVERLAY
		5153				
		5154				
		5155	*			BINARY OUTPUT MANAGEMENT
		5156				
067.150		5157	BINFNAM	DS	FB.NAML	BINARY FILE NAME (=0 IF NONE)
067.171	000	5158	BINCSN	DB	0	CURRENT SECTOR NUMBER IN BINBUF
067.172	000	5159	BINSKW	DB	0	BYTES OF HEADER ON FRONT OF BINARY FILE
		5160				
067.173		5161	ABSHDR	DS	0	HEADER FOR ABS BINARY FILE
067.173	377 000	5162		DB	3770,FT.ABS	
067.175	377 377	5163	ABSFVA	DW	-1	LOWEST ADDRESS GENERATED (=0 IF PIC)
067.177	000 000	5164	ABSLN	DW	0	LENGTH
067.201	200 042	5165	ABSENT	DW	USERFWA	ENTRY POINT
		5166				
067.203	000 000	5167	ABSLWA	DW	0	MAX ADDRESS GENERATED
		5168				
067.205		5169	PICHDR	DS	0	HEADER FOR PIC BINARY FILE
067.205	377 001	5170		DB	3770,FT.PIC	
067.207	000 000	5171	PICLEN	DW	0	LENGTH OF ENTIRE THING
067.211	000 000	5172	PICPTR	DW	0	POINTER TO REL TABLE
		5174	**			LISTING FORMAT AND CONTROL FLAGS
		5175				
067.213	000	5176	WIDE	DB	0	<0 IF WIDE SWITCH
067.214	074	5177	PAGEDP	DB	60	DEPTH OF PAGE
067.215	000	5178	FORMDP	DB	0	FORM DEPTH (ONLY IF PRINTER WONT TAKE FORMFEED)
		5179				
		5180				
		5181	**			DYNAMIC TABLE ALLOCATION
		5182				

067,216	003 078	5183	SYMFWA	DW	SYMTAB	FWA
067,220	003 076	5184	SYMPTR	DW	SYMTAB	LWA+ (SMALL SLOP FACTOR)
067,222	000 000	5185	RELLWA	DW	0	REL TABLE END
067,224	000 000	5186	RELPTR	DW	0	REL TAB ACTIVE POINTER (IT GROWS DOWN)

5188 ** FILE BUFFERS

		5189				
067,226		5190	LISIFB	DS	0	LISTING FILE BLOCK
067,226	001	5191		DB	CN.LST	LISTING CHANNEL
067,227	000	5192		DB	0	FLAG
067,230	237 071	5193		DW	LISTBUF	
067,232	237 071	5194		DW	LISTBUF	
067,234	237 071	5195		DW	LISTBUF	
067,236	237 073	5196		DW	LISTBUF+LISTBFL	
067,240		5197		DS	FB.NAML	LISTING FILE NAME

		5198				
067,261		5199	SORCFB	DS	0	SOURCE FILE BLOCK
067,261	002	5200		DB	CN.SOU	SOURCE CHANNEL
067,262	000	5201		DB	0	FLAG
067,263	237 073	5202		DW	SORCBUF	
067,265	237 073	5203		DW	SORCBUF	
067,267	237 073	5204		DW	SORCBUF	
067,271	237 074	5205		DW	SORCBUF+SORCBFL	
067,273		5206		DS	FB.NAML	LISTING FILE NAME

		5207				
067,314		5208	XTXFB	DS	0	XTEXT FILE BLOCK
067,314	003	5209		DB	CN.XTX	XTEXT CHANNEL
067,315	000	5210		DB	0	FLAG
067,316	237 074	5211		DW	XTXBUF	
067,320	237 074	5212		DW	XTXBUF	
067,322	237 074	5213		DW	XTXBUF	
067,324	237 075	5214		DW	XTXBUF+XTXBFL	
067,326		5215		DS	FB.NAML	XTEXT FILE NAME

5216

5218 ** CNDFLG - CONDITIONAL ASSEMBLY FLAG.

		5219	*			
		5220	*		=000 - NO CONDITIONS	
		5221	*		=200 - AM ASSEMBLING	
		5222	*		=201 - AM SKIPPING	
		5223				
067,347	000	5224	CNDFLG	DB	0	CONDITIONAL ASSEMBLY FLAG
		5225				
067,350	001	5226	EJEFLG	DB	1	NON-ZERO IF TO EJECT
067,351	000	5227	LINCNT	DB	0	LINES PER LAGE
067,352	000	5228	PAGNUM	DB	0	PAGE NUMBER

5229
5230 * RELOCATION FLAGS

		5231				
067,353	000	5232	EXPREL	DB	0	=ST.REL IF EXPRESSION IS RELOCATABLE
067,354	000	5233	TOKREL	DB	0	=ST.REL IF TOKEN IS RELOCATABLE

5234
5235

ASM - HDOS RESIDENT ASSEMBLER
DATA AND WORK AREAS

HEATH HBASM V1.4 01/20/78
15:23:09 16-MAY-80

PAGE 110

067.355

5236 PATCH DS

64

PATCH AREA

```

5239 **      PRS - PRESET ASSEMBLER.
5240 *
5241 *      PRS IS THE INITIAL ENTRY POINT FOR THIS PROGRAM.
5242 *
5243 *      IT GETS THE COMMAND LINE (IF IT WASNT PASSED ON THE STACK)
5244 *      CRACKS THE FILE NAMES AND SWITCHES, AND SETS UP THE ASSEMBLER.
5245 *
5246 *      *****
5247 *      * N O T E *      THIS CODE IS OVERLAID DURING ASSEMBLY BY BUFFERS AND
5248 *      *****      WORKAREAS. IT MAY NOT BE RE-ENTERED AFTER THE INITIAL TIME.
5249 *
5250 *
5251 *      PRS PERFORMS 2 TASKS:
5252 *
5253 *      1) GET COMMAND LINE, CRACK SWITCHES, AND OPEN FILES:
5254 *          BINARY FILE
5255 *          LISTING FILE
5256 *          SOURCE FILE
5257 *      2) SETUP DYNAMIC TABLES
5258 *          SYMBOL TABLE
5259 *          RELOCATION TABLE
5260 *
5261 *      PRS IS THE ENTRY POINT FOR THIS ASSEMBLER. IF THE STACK IS NON-EMPTY
5262 *      IT IS ASSUMED TO CONTAIN THE COMMAND LINE. (1ST CHARACTER PUSHED
5263 *      ON LAST)
5264 *
5265 *      FROM THEN ON, STACK DISCIPLINE IS * * NOT MAINTAINED * *
5266 *      FOR THIS ROUTINE. ITS SUBROUTINES MAY VECTOR BACK TO IT FOR EXCEPTIONAL
5267 *      CASES, WITH THE STACK UNCLEAR. THE STACK IS KEPT 'EMPTY' ((SP) = STACK)
5268 *      WHILE IN PRS. PRS EXIT TO 'ASM' VIA A JUMP.
5269 *      ENTRY FROM SYSTEM
5270 *      EXIT TO HBASH
5271 *      USES ALL
5272 *
5273 *
5274 PRS      EQU      *
5275 *
5276 *      CHECK THE HDOS VERSION
5277 *
5278 DB      SYSCALL,,VERS      /79.12.GC/
5279 JC      PRSERR1      PROBABLY NO VERSION SYSTEM CALL /79.12.GC/
5280 CPI     VERS      /79.12.GC/
5281 JNZ     PRSERR1      NOT THE CORRECT VERSION OF HDOS /79.12.GC/
5282 *
5283 *      SEE IF A COMMAND IS ON THE STACK
5284 *
5285 LXI     H,RHEML
5286 DB      SYSCALL,,SETTP      SET LIMIT (TEMPORARILY, UNTIL *BDT*)
5287 JC      PRSERR      NOT ENOUGH MEMORY
5288 LXI     H,CCHIT
5289 MVI     A,CTLG
5290 DB      SYSCALL,,CTLG      SETUP CTL-C PROCESSING
5291 LXI     H,0
5292 DAD     SP      (HL) = STACK
5293 XCHG      (DE) = STACK VALUE
5294 MVI     A,*STACK

```

```

070.115 223 5295 SUB E
070.116 117 5296 MOV C,A
070.117 076 042 5297 MVI A,STACK/256
070.121 232 5298 SBB D
070.122 107 5299 MOV B,A (BC) = BYTES ON STACK
070.123 261 5300 ORA C
070.124 312 142 070 5301 JZ PRS1 READ COMMAND LINE
070.127 041 237 075 5302 LXI H,LINE
070.132 315 252 030 5303 CALL $MOVE MOVE IN LINE
070.135 046 000 5304 MVI H,0 GUARANTEE TERMINATOR
070.137 303 252 070 5305 JMP PRS3 CRACK LINE
5306
5307 * ANNOUNCE PRODUCT
5308
070.142 315 136 031 5309 PRS1 CALL $TYPTX
070.145 012 110 104 5310 DB NL,'HDOS Assembler Issue #104.05.00.',ENL /78.12.6C/
5311
5312 * GET THE CURRENT DATE
5313
070.207 315 141 061 5314 CALL $MOVE /79.12.6C/
070.212 011 000 5315 DW HEADCL /79.12.6C/
070.214 277 040 5316 DW S,DATE /79.12.6C/
070.216 030 067 5317 DW HEADC /79.12.6C/
5318
5319 * READ COMMAND LINE
5320
070.220 377 056 5321 PRS2 DB SYSCALL,'CLEARA CLEAR ALL CHANNELS'
070.222 257 5322 XRA A
070.223 062 227 067 5323 STA LISTFB+FB.FLG
070.226 062 262 067 5324 STA SORCFB+FB.FLG CLEAR FILE BUFFERS
070.231 061 200 042 5325 LXI SP,STACK CLEAN STACK
070.234 315 136 031 5326 CALL $TYPTX
070.237 012 252 5327 DB NL,'*'+2000
070.241 041 237 075 5328 LXI H,LINE
070.244 315 063 061 5329 CALL $RTL, READ LINE /78.10.6C/
070.247 332 325 042 5330 JC EXIT CTL-D STRUCK
5331
5332 * HAVE COMMAND LINE. DECODE SWITCHES
5333
070.252 315 004 072 5334 PRS3 CALL SDV SET DEFAULT VALUES
070.255 021 063 072 5335 LXI D,SWITAB
070.260 041 237 075 5336 LXI H,LINE
070.263 315 164 074 5337 CALL $DRS DECODE AND REMOVE SWITCHES
070.266 332 222 071 5338 JC SW.ERR SWITCH ERROR
5339
5340 * HAVE CRACKED SWITCHES FROM COMMAND LINE. NOW DECODE FILE NAMES
5341
070.271 257 5342 XRA A
070.272 062 150 067 5343 STA BINFNAM CLEAR BINARY FILE NAME
070.275 062 240 067 5344 STA LISTFB+FB.NAM CLEAR LISTING FILE NAME
070.300 041 237 075 5345 LXI H,LINE
070.303 176 5346 PRS4 MOV A,M CHECK FOR '='
070.304 376 075 5347 CPI '='
070.306 043 5348 INX H
070.307 312 324 070 5349 JE PRS5 GOT '='
070.312 247 5350 ANA A

```



```

070.313 302 303 070 5351 JNZ PRS4 MORE TO CHECK
070.316 021 237 075 5352 LXI D,LINE NO LISTING OR BINARY
070.321 303 371 070 5353 JMP PRS7
5354
5355 * HAVE '='. HAS SPECIFIED LISTING AND/OR BINARY
5356
070.324 021 237 075 5357 PRS5 LXI D,LINE
070.327 041 150 067 5358 LXI H,BINFNAM
070.332 315 164 061 5359 CALL $CPF COPY FILE NAME
070.335 332 143 071 5360 JC PRS10 FORMAT ERROR
070.340 376 054 5361 CFI ','
070.342 302 356 070 5362 JNE PRS6 NOT LISTING FILE
070.345 041 240 067 5363 LXI H,LISTFB+FB.NAM
070.350 315 164 061 5364 CALL $CPF COPY FILE NAME
070.353 332 143 071 5365 JC PRS10 FNAME ERROR
070.356 376 075 5366 PRS6 CPI '='
070.360 302 174 071 5367 JNE PRS11 FORMAT ERROR
070.363 315 261 071 5368 CALL OOF OPEN OUTPUT FILES
070.366 332 220 070 5369 JC PRS2 ERROR
5370
5371 * CRACK SOURCE FILE LIST.
5372
070.371 041 273 067 5373 PRS7 LXI H,SORCFB+FB.NAM
070.374 315 164 061 5374 CALL $CPF COPY FILE NAME
070.377 041 273 067 5375 LXI H,SORCFB+FB.NAM
071.002 021 377 074 5376 LXI D,DEFAULTI
071.005 001 074 070 5377 LXI B,EXPWRK
071.010 377 053 5378 DB SYSCALL,.DECODE GET DEVICE INFO ABOUT INPUT FILE
071.012 072 074 070 5379 LDA EXPWRK+0 (A) = DEVICE CODE
071.015 346 001 5380 ANI DT.DD
071.017 312 061 071 5381 JZ PRS9 NOT DIRECTORY DEVICE
071.022 041 261 067 5382 LXI H,SORCFB
071.025 021 377 074 5383 LXI D,DEFAULTI (DE) = INPUT DEFAULT POINTER
071.030 315 337 061 5384 CALL $FOPER.
071.033 322 044 071 5385 JNC PRS8 ALL OK
071.036 315 344 071 5386 CALL PFE PRESET FILE ERROR
071.041 303 220 070 5387 JMP PRS2 RE-TRY
5388
5389 * SETUP LINCNT SO THE FIRST PAGE TITLE DOES NOT EJECT ANY PAPER.
5390 * (THIS HAS NO EFFECT IF AM USING FORMFEEDS TO EJECT PAPER)
5391
071.044 072 214 067 5392 PRS8 LDA PAGEDP
071.047 041 215 067 5393 LXI H,FORMDP
071.052 226 5394 SUB M (A) = FUNNY LINECNT TO CREATE 0 SKIP COUNT
071.053 062 351 067 5395 STA LINCNT IN *FNFX*
071.056 303 200 042 5396 JMP START START ASSEMBLY
5397
5398 * INPUT FILE NOT ON DIRECTORY DEVICE
5399
071.061 315 136 031 5400 PRS9 CALL $TYPTX
071.064 007 123 157 5401 DB BELL, 'Source File Must be on SY0:, SY1:, or SY2:, 2000
071.140 303 220 070 5402 JMP PRS2 TRY AGAIN
5403
5404 * ERROR IN FILE NAME
5405
071.143 315 136 031 5406 PRS10 CALL $TYPTX

```

071.146	007 111 154	5407	DB	BELL,'Illegal File Name',2000	
071.171	303 220 070	5408	JMP	PRS2	TRY AGAIN
		5409			
		5410	*	ILLEGAL SYNTAX	
		5411			
071.174	315 136 031	5412	PRS11	CALL	\$TYPTX
071.177	007 111 154	5413	DB	BELL,'Illegal Syntax',2000	
071.217	303 220 070	5414	JMP	PRS2	
		5415			
		5416	*	SWITCH ERROR	
		5417			
071.222	315 136 031	5418	SW.ERR	CALL	\$TYPTX
071.225	007 111 154	5419	DB	BELL,'Illegal Switch',ENL	
071.245	303 220 070	5420	JMP	PRS2	
		5421			
071.250	076 050	5422	PRSEERR1	MVI	A,EC.NCV
		5423			NOT CORRECT VERSION OF HDOS
071.252	046 012	5424	PRSEERR	MVI	H,NL
071.254	377 057	5425	DB	SYSCALL,.ERROR	
071.256	257	5426	XRA	A	
071.257	377 000	5427	DB	SYSCALL,.EXIT	
		5428			

```

5432 **      DOF - OPEN OUTPUT FILES.
5433 *
5434 *      DOF IS CALLED TO OPEN THE BINARY AND LISTING FILES,
5435 *      TO THEIR RESPECTIVE CHANNELS.
5436 *
5437 *      ENTRY  BINFNAM = BINARY FILE NAME (=0 IF NONE)
5438 *      LISTFB+FB.NAM = LISTING FILE NAME (=0 IF NONE)
5439 *      EXIT   'C' CLEAR IF OK
5440 *      'C' SET IF ERROR
5441 *      ERROR IS MESSAGED BY DOF
5442 *      USES   A,F
5443
5444
071.261 305 5445 DOF PUSH B          SAVE REGISTERS
071.262 325 5446 PUSH D
071.263 345 5447 PUSH H
071.264 041 150 067 5448 LXI H,BINFNAM
071.267 176 5449 MOV A,M
071.270 247 5450 ANA A
071.271 312 311 071 5451 JZ DOF1          NO BINARY
5452
5453 *      OPEN BINARY FILE
5454
071.274 021 363 074 5455 LXI D,DEFLT B
071.277 076 000 5456 MVI A,CN,BIN
071.301 377 043 5457 DB SYSCALL,OPENW
071.303 041 136 067 5458 LXI H,BINFNAM-FB.NAM
071.306 332 334 071 5459 JC DOF3          ERROR
5460
5461 *      OPEN LISTING FILE
5462
071.311 041 226 067 5463 DOF1 LXI H,LISTFB
071.314 072 240 067 5464 LDA LISTFB+FB.NAM
071.317 247 5465 ANA A
071.320 312 340 071 5466 JZ DOF4          NONE TO OPEN
071.323 021 371 074 5467 LXI D,DEFLT L
071.326 315 342 061 5468 CALL $FOPEW,      OPEN FOR WRITE
071.331 322 340 071 5469 JNC DOF4          NO ERROR
5470
5471 *      ERROR IN FILE
5472
071.334 315 344 071 5473 DOF3 CALL PFE          PRESET FILE ERROR
071.337 067 5474 STC
071.340 341 5475 DOF4 POP H
071.341 321 5476 POP D
071.342 301 5477 POP B
071.343 311 5478 RET

```

```

5480 **      PFE - PRESET FILE ERROR.
5481 *
5482 *      PFE IS CALLED TO PRINT AN ERROR MESSAGE.
5483 *
5484 *      ENTRY      (A) = CODE
5485 *                  (HL) = FILE BLOCK ADDRESS (ONLY USED TO GET FB.NAM)
5486 *      EXIT      NONE
5487 *      USES      ALL
5488
5489
071.344 365 5490 PFE      PUSH      PSW          SAVE CODE
071.345 315 136 031 5491      CALL     $TYPTX
071.350 007 105 162 5492      DB       BELL,'Error On File','+200Q
071.367 001 012 000 5493      LXI       B,FB.NAM
071.372 011 5494      DAD       B
071.373 315 051 061 5495      CALL     $TYPLZ          TYPE FNAME
071.376 361 5496      POP      PSW
071.377 046 012 5497      MVI      H,NL
072.001 377 057 5498      DB       SYSCALL,'ERROR PRINT ERROR MEANING
072.003 311 5499      RET

```

```

5501 **      SDV - SET DEFAULT SWITCH VALUES.
5502 *
5503 *      SDV IS CALLED BY PRS TO SET ALL SWITCH FLAGS TO THEIR DEFAULT
5504 *      VALUES. THEIR VALUES CANNOT BE SIMPLY ASSEMBLED IN, BECUASE
5505 *      AN INCORRECT COMMAND LINE SWITCH MAY CHANGE THEM, BEFORE
5506 *      THE ERROR IS DETECTED. SDV RESETS THEM FOR THE
5507 *      NEXT TRY.
5508 *
5509 *      ENTRY      NONE
5510 *      EXIT      NONE
5511 *      USES      A,F,H,L
5512
5513
072.004 076 074 5514 SDV      MVI      A,60
072.006 062 214 067 5515      STA      PAGEDP          SET PAGE DEPTH
072.011 257 5516      XRA      A
072.012 062 215 067 5517      STA      FORMDP          USE PAGE FORM CONTROL
072.015 062 213 067 5518      STA      WIDE           CLEAR /WIDE SWITCH
072.020 062 131 067 5519      STA      LSTCTL5
072.023 057 5520      CMA
072.024 062 132 067 5521      STA      LSTCTL6
072.027 315 141 061 5522      CALL     $MOVE1
072.032 022 000 041 5523      DW       18,SDVA,DEFALTB
000.000 5524      ERNZ      DEFALTL-DEFALTB-6
000.000 5525      ERNZ      DEFALTI-DEFALTI-6
072.040 311 5526      RET
5527
072.041 123 131 060 5528 SDVA      DB       'SYOABS'          DEFAULT FOR BINARY
072.047 123 131 060 5529      DB       'SYOLST'         DEFAULT FOR LISTING
072.055 123 131 060 5530      DB       'SYOASM'         DEFAULT FOR INPUT

```

```

5533 ** SWITCH TABLE.
5534 *
5535 * THIS TABLE CONTAINS DESCRIPTIONS FOR COMMAND SWITCHES.
5536 * SEE '$DRS' FOR A DESCRIPTION OF IT'S FORMAT.
5537
5538
072.063 5539 SWITAB DS 0
5540
5541 * /PAGE:NN
5542
072.063 120 301 307 5543 DB 'P','A'+2000,'B'+2000,'E'+2000,2000
072.070 143 072 5544 DW SW.PAG
5545
5546 * /FORM:NN
5547
072.072 106 317 322 5548 DB 'F','O'+2000,'R'+2000,'M'+2000,2000
072.077 217 072 5549 DW SW.FOR
5550
5551 * /WIDE
5552
072.101 127 311 304 5553 DB 'W','I'+2000,'D'+2000,'E'+2000,2000
072.106 275 072 5554 DW SW.WID
5555
5556 * /LON:CCC
5557
072.110 114 117 116 5558 DB 'LON',2000
072.114 131 073 5559 DW SW.LON
5560
5561 * /LOF:CCC
5562
072.116 114 117 106 5563 DB 'LOF',2000
072.122 206 073 5564 DW SW.LOF
5565
5566 * /LARGE
5567
072.124 114 101 322 5568 DB 'LA','R'+2000,'G'+2000,'E'+2000,2000
072.132 361 072 5569 DW SW.LAR
5570
5571 * /ERR
5572
072.134 105 122 122 5573 DB 'ERR',2000
072.140 046 073 5574 DW SW.ERS
5575
072.142 000 5576 DB 0 END OF TABLE

5578 ** SW.PAG - /PAGE:NN
5579
072.143 315 107 074 5580 SW.PAG CALL $DNS, DECODE NUMERIC SWITCHES
072.146 332 157 072 5581 JC SW.PAG1 ERROR
072.151 173 5582 MOV A,E (A) = VALUE
072.152 247 5583 ANA A
072.153 062 214 067 5584 STA PAGEDF
072.156 300 5585 RNZ :0 IS ILLEGAL

```

```

072.157 315 136 031 5586 SW.PAG1 CALL $TYPTX
072.162 042 057 120 5587 DB '/PAGE:' Value is No Good', ' +200Q
072.214 303 222 071 5588 JMP SW.ERR

```

```

5590 ** SW.FOR - /FORM:NN
5591
072.217 315 107 074 5592 SW.FOR CALL $DNS. DECODE DECIMAL SWITCH
072.222 332 233 072 5593 JC SW.FOR1
072.225 173 5594 MOV A,E (A) = VALUE
072.226 062 215 067 5595 STA FORMDP
072.231 247 5596 ANA A
072.232 300 5597 RNZ VALUE OF 0 ILLEGAL
072.233 315 136 031 5598 SW.FOR1 CALL $TYPTX
072.236 042 057 106 5599 DB '/FORM:' Value is No Good -', ' +200Q
072.272 303 222 071 5600 JMP SW.ERR

```

```

5602 ** SW.WID - /WIDE
5603
072.275 312 306 072 5604 SW.WID JE SW.WID1 NO VALUE ALLOWED
072.300 076 001 5605 MVI A,1
072.302 062 213 067 5606 STA WIDE
072.305 311 5607 RET
5608
072.306 315 136 031 5609 SW.WID1 CALL $TYPTX
072.311 111 155 160 5610 DB 'Improper Format For "/WIDE" Switch -', ' +200Q
072.356 303 222 071 5611 JMP SW.ERR

```

```

5613 ** SW.LAR - /LARGE
5614
072.361 312 372 072 5615 SW.LAR JE SW.LAR1 NO VALUE ALLOWED
072.364 076 001 5616 MVI A,1
072.366 062 147 047 5617 STA LARGE
072.371 311 5618 RET
5619
072.372 315 136 031 5620 SW.LAR1 CALL $TYPTX
072.375 111 155 160 5621 DB 'Improper Format for "/LARGE" Switch -', ' +200Q
073.043 303 222 071 5622 JMP SW.ERR

```

```

5624 ** SW.ERS - /ERR
5625
073.046 312 057 073 5626 SW.ERS JE SW.ERS1 NO VALUE ALLOWED
073.051 076 001 5627 MVI A,1
073.053 062 125 067 5628 STA ERRSHO
073.056 311 5629 RET
5630
073.057 315 136 031 5631 SW.ERS1 CALL $TYPTX
073.062 111 155 160 5632 DB 'Improper Format for "/ERR" Switch -', ' +200Q
073.126 303 222 071 5633 JMP SW.ERR

```

```

5635 ** SW.LON - /LON:CCC
5636
073.131 315 264 073 5637 SW.LON CALL DLS DECODE LISTING SWITCHES
073.134 332 145 073 5638 JC SW.LON1
073.137 041 131 067 5639 LXI H,LSTCTL5
073.142 266 5640 ORA M
073.143 167 5641 MOV M,A SET SWITCHES
073.144 311 5642 RET
5643
073.145 315 136 031 5644 SW.LON1 CALL $TYPTX
073.150 042 057 114 5645 DB **/LON: Value is No Good -', '+2000
073.203 303 222 071 5646 JMP SW.ERR

```

```

5648 ** SW.LOF - /LOF:CCC
5649
073.206 315 264 073 5650 SW.LOF CALL DLS DECODE LISTING SWITCHES
073.211 332 223 073 5651 JC SW.LOF1
073.214 041 132 067 5652 LXI H,LSTCTL5
073.217 057 5653 CMA
073.220 246 5654 ANA M
073.221 167 5655 MOV M,A SET 0 BITS FOR SPECIFIED OPTIONS
073.222 311 5656 RET
5657
073.223 315 136 031 5658 SW.LOF1 CALL $TYPTX
073.226 042 057 114 5659 DB **/LOF: Value is No Good -', '+2000
073.261 303 222 071 5660 JMP SW.ERR

```

```

5662 ** DLS - DECODE LISTING SWITCHES.
5663 *
5664 * DLS IS CALLED TO DECODE THE SPECIFIED LIST SUBOPTIONS FOR THE /LON
5665 * AND THE /LOF SWITCHES.
5666 *
5667 * THE OPTIONS ARE ANALYZED, AND REPLACED WITH BLANKS
5668 *
5669 * ENTRY (HL) = ADDRESS OF 'CCC'
5670 * EXIT 'C' CLEAR IF OK
5671 * (A) = BITS SET FOR EACH SPECIFIED OPTION
5672 * 'C' SET IF ERROR
5673 * USES ALL
5674
073.264 176 5676 DLS MOV A,M (A) = SUPPOSED ':'
073.265 376 072 5677 CPI ':' CHECK UP ON HIM
073.267 067 5678 STC
073.270 300 5679 RNE NOT ':'
073.271 353 5680 XCHG (DE) = ADDRESS
073.272 006 000 5681 MVI B,0
5682
5683 * DECODE NEXT SWITCH
5684
073.274 076 040 5685 DLS1 MVI A,' '

```

073.276	022	5686	STAX	D	CLEAR THAT ONE
073.277	023	5687	INX	D	
073.300	032	5688	LDAX	D	
073.301	376 040	5689	CPI	' '	
073.303	312 274 073	5690	JE	DLS1	SKIP
073.306	376 057	5691	CPI	' '	
073.310	312 342 073	5692	JE	DLS2	DONE
073.313	376 054	5693	CPI	' '	
073.315	312 342 073	5694	JE	DLS2	DONE
073.320	247	5695	ANA	A	
073.321	312 342 073	5696	JZ	DLS2	DONE
		5697			
		5698	*		MUST BE A SUBOPTION
		5699			
073.324	041 342 045	5700	LXI	H,LSTA	
073.327	315 261 061	5701	CALL	\$TBLS	
073.332	067	5702	STC		
073.333	300	5703	RNZ		NOT A GOOD OPTION
073.334	176	5704	MOV	A,M	
073.335	260	5705	ORA	B	SET FLAGS
073.336	107	5706	MOV	B,A	
073.337	303 274 073	5707	JMP	DLS1	GET ANOTHER
		5708			
		5709	*		ALL DONE
		5710			
073.342	170	5711	DLS2	MOV	A,B
073.343	247	5712	ANA	A	
073.344	067	5713	STC		
073.345	310	5714	RZ		NONE FOUND: ERROR
073.346	247	5715	ANA	A	CLEAR CARRY
073.347	311	5716	RET		RETURN WITH OK

073.350

5719

XTEXT CVD

5721X ** \$CVD - CHECK FOR VALID DIGIT.
5722X *
5723X * CVD EXAMINES A DIGIT TO SEE IF IT IS A VALID DECIMAL DIGIT.
5724X *
5725X * ENTRY (HL) = ADDRESS OF CHARACTER
5726X * EXIT 'C' SET IF ILLEGAL
5727X * (A) = VALUE
5728X * USES A,F

5729X

5730X

073.350 176
073.351 326 080
073.353 330
073.354 376 012
073.356 077
073.357 311
073.360

5731X \$CVD MOV A,M (A) = CHARACTER
5732X \$CVD SUI '0'
5733X RC ILLEGAL
5734X CFI 9+1
5735X CMC
5736X RET
5737 XTEXT MUB6

5739X ** \$MUB6 - MULTIPLY 8X16 UNSIGNED.
5740X *
5741X * \$MUB6 MULTIPLIES A 16 BIT VALUE BY A 8
5742X * BIT VALUE.
5743X *
5744X * ENTRY (A) = MULTIPLIER
5745X * (DE) = MULTIPLICAND
5746X * EXIT (HL) = RESULT
5747X * 'Z' SET IF NOT OVERFLOW
5748X * USES A,F,H,L
5749X

5750X

031.007
073.360

5751X \$MUB6 EQU 31007A IN H17 ROM
5752 XTEXT DNV

5754X ** \$DNV - DECODE NUMERIC VALUE.
5755X *
5756X * \$DNV DECODES A NUMERIC VALUE (IN THE FORM OF AN ASCII STRING)
5757X * INTO A BINARY NUMBER. THE MAXIMUM MAGNITUDE IS
5758X * 855350.
5759X *
5760X * THE NUMBER MAY CONTAIN A POSTRADIX OF 'B' (BINARY)
5761X * 'O' OR 'Q' (OCTAL) OR 'D' (DECIMAL)
5762X *
5763X * ENTRY (HL) = ADDRESS OF FIRST BYTE OF NUMBER
5764X * (A) = DEFAULT BASE ('2' FOR BINARY, '10' FOR DECIMAL, ETC.)
5765X * EXIT 'C' CLEAR IF OK

```

5766X *      (HL) ADVANCED PAST NUMBER (AND POSTRADIX)
5767X *      (DE) = VALUE
5768X *      C SET IF ERROR
5769X *      USES ALL
5770X
5771X
073.360 062 075 074 5772X $DNV STA $DNVA SET DEFAULT BASE
073.363 104 5773X MOV B,H
073.364 115 5774X MOV C,L (BC) = TEXT ADDRESS
5775X
5776X *      SCAN FOR POSTRADIX
5777X
073.365 176 5778X $DNV1 MOV A,M
073.366 315 351 073 5779X CALL $CVD, CHECK FOR VALID DECIMAL DIGIT
073.371 043 5780X INX H
073.372 322 365 073 5781X JNC $DNV1 MORE TO GO
073.375 053 5782X DCX H REMOVE EXTRA INCREMENT
073.376 171 5783X MOV A,C
073.377 275 5784X CMP L SEE IF THERE WERE ANY NUMBERS
074.000 067 5785X STC ASSUME NOT
074.001 310 5786X RE ERROR
5787X
5788X *      OUT OF NUMBERS. SEE IF POSTRADIX FOLLOWS
5789X
074.002 176 5790X MOV A,M (A) = PROPOSED POSTRADIX
074.003 345 5791X PUSH H SAVE END ADDRESS
074.004 041 076 074 5792X LXI H,$DNVB
074.007 247 5793X ANA A
074.010 312 030 074 5794X JZ $DNV2 NO POSTRADIX
074.013 315 261 061 5795X CALL $TBLS
074.016 176 5796X MOV A,M
074.017 302 030 074 5797X JNE $DNV2 NOT POSTRADIX
074.022 341 5798X POP H
074.023 043 5799X INX H SKIP POSTRADIX
074.024 345 5800X PUSH H
074.025 062 075 074 5801X STA $DNVA SET NEW POSTRADIX
074.030 021 000 000 5802X $DNV2 LXI D,0 (DE) = ACCUMULATOR
5803X
5804X *      BUILD NUMBER
5805X
074.033 072 075 074 5806X $DNV3 LDA $DNVA (A) = BASE
074.036 365 5807X PUSH PSW SAVE BASE
074.037 315 007 031 5808X CALL $MUS6 MULTIPLY
074.042 321 5809X POP D (D) = BASE
074.043 332 073 074 5810X JC $DNV4 OVERFLOW
074.046 012 5811X LDAX B (A) = DIGIT
074.047 326 060 5812X SUI '0'
074.051 003 5813X INX B
074.052 272 5814X CMP D COMPARE TO BASE
074.053 077 5815X CMC
074.054 332 073 074 5816X JC $DNV4 TOO LARGE A DIGIT
074.057 315 101 030 5817X CALL $DADA, ADD TO VALUE
074.062 353 5818X XCHG (DE) = VALUE
074.063 012 5819X LDAX B
074.064 315 351 073 5820X CALL $CVD,
074.067 322 033 074 5821X JNC $DNV3 MORE TO GO

```

\$DNV

15:23:38 16-MAY-80

074.072	247	5822X	ANA	A	CLEAR CARRY
074.073	341	5823X \$DNV4	POP	H	RESTORE POINTER
074.074	311	5824X	RET		EXIT
		5825X			
074.075	000	5826X \$DNV4	DB	0	DEFAULT BASE
074.076	102 002	5827X \$DNV8	DB	'B',2	POSTRADIX TABLE
074.100	117 010	5828X	DB	'0',8	
074.102	121 010	5829X	DB	'Q',8	
074.104	104 012	5830X	DB	'D',10	
074.106	000	5831X	DB	0	
074.107		5832	XTEXT	DNS	
		5834X **			\$DNS - DECODE NUMERIC SWITCH.
		5835X *			
		5836X *			\$DNS DECODES A NUMERIC SWITCH OF THE FORM:
		5837X *			
		5838X *			;NNN
		5839X *			
		5840X *			A POSTRADIX OF D, Q, O, OR B IS ALLOWED, IF THE VALUE
		5841X *			IS SYNTACTICALLY VALID, IT IS REPLACED WITH BLANKS.
		5842X *			
		5843X *	ENTRY	(HL) = ADDRESS IF ':'	
		5844X *		(A) = DEFAULT BASE (2, 8 OR 10)	
		5845X *	EXIT	'C' CLEAR IF OK	
		5846X *		(HL) ADVANCED PAST VALUE	
		5847X *		VALUE BLANKED	
		5848X *		(DE) = VALUE	
		5849X *		'C' SET IF ERROR	
		5850X *	USES	ALL	
		5851X			
		5852X			
074.107	076 012	5853X \$DNS.	MVI	A,10	BASE 10 DEFAULT
074.111	107	5854X \$DNS	MOV	B,A	(B) = DEFAULT BASE
074.112	176	5855X	MOV	A,M	
074.113	376 072	5856X	CPI	':'	
074.115	067	5857X	STC		
074.116	300	5858X	RNE		NOT ':'
074.117	345	5859X	PUSH	H	SAVE ADDRESS OF SWITCH START
074.120	043	5860X	INX	H	
074.121	170	5861X	MOV	A,B	
074.122	315 360 073	5862X	CALL	\$DNV	DECODE NUMERIC VALUE
074.125	301	5863X	POP	B	(BC) = ADDRESS OF ':'
074.126	330	5864X	RC		ERROR
074.127	076 040	5865X \$DNS1	MVI	A,' '	
074.131	002	5866X	STAX	B	BLANK LINE
074.132	003	5867X	INX	B	INCREMENT ADDRESS
074.133	175	5868X	MOV	A,L	
074.134	271	5869X	CMF	C	
074.135	302 127 074	5870X	JNE	\$DNS1	
074.140	170	5871X	MOV	A,B	
074.141	274	5872X	CMF	H	SEE IF IN RIGHT BANK
074.142	302 127 074	5873X	JNE	\$DNS1	
074.145	311	5874X	RET		RETURN WITH 'C' CLEAR AND VALUE

074.146

5875

XTEXT SOB

```

5877X **      $SOB - SKIP OVER BLANKS.
5878X *
5879X *      $SOB IS CALLED TO SKIP AN ARBITRARILY LONG STRING OF BLANKS AND TABS.
5880X *
5881X *      ENTRY (HL) = FWA OF (POSSIBLE) BLANK STRING
5882X *      EXIT (HL) = LWA+1 OF BLANK STRING (UNCHANGED IF NO BLANKS)
5883X *      (A) = FIRST NON-BLANK, NON-TAB CHARACTER EEN
5884X *      USES A,F,H,L
5885X
5886X
074.146 053 5887X $SOB DCX H PRE-DECREMENT
074.147 043 5888X $SOB1 INX H
074.150 176 5889X MOV A,M
074.151 376 040 5890X CPI '/'
074.153 312 147 074 5891X JE $SOB1 GOT BLANK
074.156 376 011 5892X CPI TAB
074.160 312 147 074 5893X JE $SOB1 GOT TAB
074.163 311 5894X RET
074.164 5895X XTEXT DRS

```

```

5897X **      $DRS - DECODE AND REMOVE SWITCHES.
5898X *
5899X *      $DRS IS CALLED TO DECODE COMMAND SWITCHES FROM A LINE
5900X *      OF TEXT. SWITCHES TAKE THE FORM:
5901X *
5902X *      /XXXXX
5903X *
5904X *      AFTER A SWITCH HAS BEEN LOCATED, IT (AND THE PRECEDING '/')
5905X *      ARE REPLACED WITH BLANKS.
5906X *
5907X *      VALID SWITCH DESCRIPTIONS ARE ENCODED INTO A TABLE
5908X *      SUPPLIED BY THE CALLER, IN THE FORMAT:
5909X *
5910X *      DB 'X...X' REQUIRED SWITCH CHARACTERS
5911X *      DB 'C'+2000,...,'C'+2000 OPTIONAL CHARACTERS
5912X *      DB 2000 END OF CHARACTERS
5913X *      DW ADDR PROCESSOR ADDRESS (CALLED WHEN SWITCH DETECTED)
5914X *
5915X *      DB 'Y...Y' NEXT SWITCH
5916X *
5917X *
5918X *
5919X *
5920X *      DB 0 FLAGS END OF TABLE
5921X *
5922X *      SWITCHES MUST BE FOLLOWED BY A ':', A '/' (ANOTHER SWITCH)
5923X *      A ',', OR A 00 BYTE.
5924X *

```

```

5925X *      UPON DETECTION OF A VALID SWITCH, $DRS CALLS THE USER PROCESS
5926X *      ROUTINE. UPON ENTRY,
5927X *      (HL) = ADDRESS OF THE FIRST BYTE FOLLOWING THE SWITCH
5928X *      'Z' CLEAR IF CHARACTER = '/', ',', OR 00
5929X *      'Z' SET IF CHARACTER = '?'
5930X *
5931X *      THE USER ROUTINE CAN DECODE SWITCH SUB-OPTIONS, IF DESIRED.
5932X *      THE USER ROUTINE MAY USE ALL REGISTERS.
5933X *
5934X *      ENTRY (DE) = SWITCH TABLE FWA
5935X *      (HL) = LINE FWA
5936X *      EXIT 'C' CLEAR IF OK
5937X *      'C' SET IF ERROR
5938X *      (HL) = ADDRESS OF START OF BAD SWITCH
5939X *      (A) = ERROR CODE
5940X *      USES ALL
5941X
5942X
074.164      5943X $DRS EQU *
5944X
5945X *      LOOK FOR SWITCHES
5946X
074.164 176      5947X $DRS1 MOV A,M
074.165 247      5948X ANA A
074.166 310      5949X RZ                      END OF LINE
074.167 043      5950X INX H
074.170 376 057  5951X CPI '/'
074.172 302 164 074 5952X JNE $DRS1          NOT A SWITCH
074.175 042 361 074 5953X SHLD $DRSB      ($DRSB) = SWITCH FWA (AFTER '/')
5954X
5955X *      GOT A SWITCH. LOOK FOR A MATCH IN THE CALLER'S TABLE
5956X
074.200 325      5957X PUSH D          SAVE TABLE FWA
074.201 052 361 074 5958X $DRS2 LHLD $DRSB      (HL) = SWITCH FWA
074.204 032      5959X $DRS3 LDAX D          (A) = TABLE ENTRY
074.205 346 177  5960X ANI 1770
074.207 312 257 074 5961X JZ $DRS6          GOT A MATCH
074.212 276      5962X CMP M
074.213 302 223 074 5963X JNE $DRS4          NO MATCH
074.216 023      5964X INX D
074.217 043      5965X INX H
074.220 303 204 074 5966X JMP $DRS3          SEE IF MORE MATCH
5967X
5968X *      HAVE MIS-MATCH. SEE IF THE MISSING CHARACTER IS SIGNIFICANT
5969X
074.223 176      5970X $DRS4 MOV A,M          (A) = LINE CHARACTER WE COULDN'T MATCH
074.224 315 330 074 5971X CALL $DRS15      SEE IF OK TERMINATOR
074.227 302 237 074 5972X JNE $DRS4.5      NO MATCH ON THIS SWITCH
074.232 032      5973X LDAX D          (A) = NEXT CHARACTER IN SWITCH PATTERN
074.233 247      5974X ANA A
074.234 372 257 074 5975X JM $DRS6          HAVE SUFFICIENT MATCH
074.237 315 343 074 5976X $DRS4.5 CALL $DRS20      SKIP TABLE ENTRY
074.242 032      5977X LDAX D
074.243 247      5978X ANA A
074.244 302 201 074 5979X JNZ $DRS2          MORE SWITCHES IN TABLE TO CHECK
5980X

```

```

5981X *      BAD SWITCH
5982X
074.247 321 5983X $DRS5 POP D RESTORE STACK
074.250 052 361 074 5984X LHL D $DRSB POINT TO BAD SWITCH
074.253 067 5985X STC
074.254 076 032 5986X MVI A,EC.IS ILLEGAL SWITCH
074.256 311 5987X RET
5988X
5989X *      HAVE SWITCH. CHECK IT'S FOLLOWING CHARACTER
5990X
074.257 315 146 074 5991X $DRS6 CALL $SOB SKIP OVER BLANKS
074.262 176 5992X MOV A,M
074.263 315 330 074 5993X CALL $DRS15 CHECK CHARACTER
074.266 302 247 074 5994X JNE $DRS5 IN ERROR
074.271 315 343 074 5995X CALL $DRS20 GET PROCESSOR ADDRESS
074.274 021 306 074 5996X LXI D,$DRS7
074.277 345 5997X PUSH H SAVE (HL)
074.300 325 5998X PUSH D SET RETURN ADDRESS FOR TABLE CODE
074.301 305 5999X PUSH B SAVE PROCESSOR ADDRESS
074.302 176 6000X MOV A,M (A) = NEXT CHARACTER
074.303 376 072 6001X CFI ' ' SET CONDITION CODES
074.305 311 6002X RET CALL USER PROCESS
6003X
6004X *      USER PROCESS RETURNS HERE
6005X
074.306 321 6006X $DRS7 POP D (DE) = LAST CHARACTER OF SWITCH+1
074.307 052 361 074 6007X LHL D $DRSB (HL) = FIRST CHARACTER OF SWITCH AFTER /
074.312 053 6008X DCX H (HL) = ADDRESS OF '/'
6009X
6010X *      REPLACE SWITCH WITH BLANKS
6011X
074.313 066 040 6012X $DRS8 MVI M,' '
074.315 043 6013X INX H
074.316 315 216 030 6014X CALL $CDEHL
074.321 302 313 074 6015X JNE $DRS8 NOT THERE YET
074.324 321 6016X POP D (DE) = SWITCH TABLE FWA
074.325 303 164 074 6017X JMP $DRS1 LOOK FOR MORE SWITCHES

6019X **      $DRS15 - CHECK FOR VALID DELIMITER CHARACTER.
6020X *
6021X *      $DRS15 CHECKS THE NEXT TEXT CHARACTER TO SEE IF IT IS
6022X *
6023X *      00, '/', ',', '.', '!'
6024X *
6025X *      ENTRY (A) = CHARACTER
6026X *      EXIT 'Z' SET IFF CHARACTER IS ONE OF THE ABOVE
6027X *      USES F
6028X
074.330 247 6029X $DRS15 ANA A
074.331 310 6030X RZ IS 00
074.332 376 057 6031X CPI '/'
074.334 310 6032X RE
074.335 376 054 6033X CPI ','
074.337 310 6034X RE
074.340 376 072 6035X CPI '!'

```

```

074.342 311      6036X      RET
                                6038X **      $DRS20 - GET PROCESSOR ADDRESS.
                                6039X *
                                6040X *      $DRS20 IS CALLED TO GET THE PROCESSOR ADDRESS FIELD OUT OF
                                6041X *      AN ENTRY IN THE SWITCH TABLE. THE CALLER SUPPLIES A POINTER
                                6042X *      TO SOMEWHERE IN THE TEXT PART OF THE SWITCH DESCRIPTION;
                                6043X *      $DRS20 ADVANCES THE POINTER TO THE PROCESSOR ADDRESS.
                                6044X *
                                6045X *      ENTRY (DE) = POINTER TO TEXT PART OF SWITCH ENTRY
                                6046X *      EXIT (DE) = POINTER TO 1ST BYTE OF NEXT SWITCH TABLE ENTRY
                                6047X *      (BC) = PROCESSOR ADDRESS FROM TABLE
                                6048X *      USES A,F,B,C,D,E
                                6049X
                                6050X
074.343 032      6051X $DRS20 LDAX D
074.344 023      6052X INX D
074.345 376 200  6053X CPI 2000
074.347 302 343 074 6054X JNE $DRS20
074.352 032      6055X LDAX D (A) = LOW BYTE OF PROCESSOR ADDRESS
074.353 117      6056X MOV C,A
074.354 023      6057X INX D
074.355 032      6058X LDAX D
074.356 107      6059X MOV B,A (BC) = PROCESSOR ADDRESS
074.357 023      6060X INX D
074.360 311      6061X RET
                                6062X
074.361 000 000  6063X $DRSE DW 0 POINTER TO SWITCH BEING PROCESSED
                                6064
074.363          6065 DEFALTB DS 6 DEFAULTS FOR BINARY FILE NAME
074.371          6066 DEFALTL DS 6 DEFAULTS FOR LISTING FILE NAME
074.377          6067 DEFALTI DS 6 DEFAULTS FIR INPUT FILE NAME
                                6068
075.005          6069 MEML EQU * LWA LOADED MEMORY
                                6070

```

```

070.055      6073
              6074      ORG      PRS      THESE BUFFERS OVERLAY PRS
              6075
              6076      **      STATEMENT UNPACK FIELDS, SETUP BY *UNL*.
              6077
070.055      6078      DS      1      SMASHED IF NO LABEL
070.056      6079      LABEL   DS      8      LABEL FIELD
070.066      6080      DS      1      SMASHED IF NO OPCODE
070.067      6081      OPCODE  DS      5      OPCODE VALUE
070.074      6082      EXPWRK  DS      99     EXPRESSION WORK-AREA      /80.02.GC/
000.000      6083      ERRNZ   *-EXPWRK+1-LINEMAX      /80.02.GC/
070.074      6084      XTEXTB  EQU     EXPWRK      XTEXTB SCRATCH BUFFER
              6085
070.237      6086      BINBFR  DS      256     BINARY BUFFER
002.000      6087      LISTBFL EQU     512     LISTING BUFFER SIZE
001.000      6088      SORCBFL EQU     256     SOURCE BUFFER SIZE
001.000      6089      TXTBFL  EQU     256     XTEXT BUFFER SIZE
              6090
071.237      6091      LISTBUF  DS      LISTBFL
073.237      6092      SORCBUF  DS      SORCBFL
074.237      6093      XTxBUF  DS      TXTBFL
              6094
              6095
076.003      6096      *      BUFFERS USED BY PRESET
              6097
377.146      6098      ERRPL   MEML-*      MUST NOT OVERLAY PRESET CODE
075.237      6099      LINE    DS      100     LINE BUFFER
000.144      6100      LINEMAX EQU     *-LINE      MAX LENGTH
              6101
076.003      6102      RMEML   EQU     *      MEM LIMIT WHEN RUNNING *PRS*
              6103
076.003      6104      SYMTAB  EQU     *      START OF SYMBOL TABLE
              6105
              6106
076.003      6107      END
ASSEMBLY COMPLETE
6107 STATEMENTS
0 ERRORS DETECTED
8916 BYTES FREE

```


PAGE 129

[illegible]

XREF 01:1
 PAGE 130

[illegible]

CROSS REFERENCE TABLE

PAGE 131

.CLEAR	000056	213L	5321				
.CLOSE	000046	205L	604	4394			
.CLRCO	000007	189L					
.CONSL	000006	188L	3708				
.CTLG	000041	200L	5290				
.DECODE	000053	210L	5378				
.DELET	000050	207L					
.DISMT	000061	216L					
.DMNMS	000203	227L					
.DMOUN	000201	225L					
.ERROR	000057	214L	2432	4947	5425	5498	
.EXIT	000000	182L	617	634	4296	4302	5427
.LINK	000040	189L					
.LOADD	000062	217L					
.LOADQ	000010	190L					
.MONMS	000202	226L					
.MOUNT	000200	224L					
.NAME	000054	211L	1517				
.OFENC	000045	204L					
.OPENR	000042	201L	4313				
.OPENU	000044	203L	4315				
.OPENW	000043	202L	4314	5457			
.POSIT	000047	206L	2341	3343	3626		
.PRINT	000003	185L	614	2763			
.READ	000004	186L	2352	4877			
.RENAM	000051	208L					
.RESET	000204	228L					
.SCIN	000001	183L	3903				
.SCOUT	000002	184L	3907	4066			
.SETTP	000052	209L	2429	5286			
.SYSRES	000012	192L					
.VERS	000011	191L	5278				
.WRITE	000005	187L	2347	3645	4382	4678	4786
.ABS.COD	000010	451L	567	1480	1483		
.ABS.ENT	000006	449L					
.ABS.ID	000000	445L					
.ABS.LDA	000002	447L					
.ABS.LEN	000004	448L					
.ABSENT	067201	1333	5165L				
.ABSFWA	067175	1456	1476	2311	2373	2382	5163L
.ABSHDR	067173	1483	5161L				
.ABSLN	067177	1479	5164L				
.ABSLWA	067203	1474	2384	2391	5167L		
.ABV	052070	594	598	2302L	3216		
.ABVQ	052206	2342	2349L				
.ABV00	052230	2353	2356L				
.ABV1	052231	2325	2360L				
.ABV2	052244	2309	2370L				
.ABV3	052275	2381	2383L				
.AIO.CGN	041047	410L					
.AIO.CHA	041116	425L					
.AIO.CNT	041111	421L					
.AIO.CSI	041050	411L					
.AIO.DDA	041041	406E					
.AIO.DES	041055	415L					
.AIO.DEV	041057	416L					
.AIO.DIR	041062	419L					
.AIO.DTA	041053	414L					

```

XREF V1.1

```

PAGE 132

[illegible]

CROSS REFERENCE TABLE

PAGE 133

CNDFLG	067347	661	712	748	1140	1156	1169	5224L
CO.FLG	000001	305E	3707					
CODE	046164	797	1414E					
CODE0	046172	1415	1418L					
CODE1	046206	1422	1427L					
CODE2	046231	933	1430	1434	1443L			
CODE2.5	046273	1454	1460L					
CODE3	046341	1468	1490L					
CODEFLG	067146	665	928	1450	5150L			
COL	053100	1067	2525L	2771				
COL1	053123	2536	2542L					
COL2.5	053153	2555	2557L					
COL3	053205	2538	2569L					
CR	000015	76E						
CS.FLG	000200	306E						
CSL.CHR	000001	283E						
CSL.ECH	000200	281E						
CSL.WRP	000002	282E						
CT.ALPH	000200	53E	1984	3517				
CTB	063363	4565	4722	4749	4824L			
CTB1	063374	4830L	4839					
CTLA	000001	91E						
CTLB	000002	92E						
CTLC	000003	93E	5289					
CTLD	000004	94E	3869					
CTLO	000017	95E						
CTLP	000020	96E						
CTLQ	000021	97E						
CTLS	000023	98E						
CTLZ	000032	99E						
CTP.2SB	000010	291E						
CTP.BKM	000002	292E						
CTP.BKS	000200	288E						
CTP.MLI	000040	289E						
CTP.MLO	000020	290E						
CTP.TAB	000001	293E						
CUS	053212	2588L	3240	3315	3388			
D.CON	040110	243L						
D.RAM	040240	246L						
D.VEC	040130	245L						
DB	044116	780	942E					
DB1	044121	944L	988					
DB2	044131	956L	957					
DB3	044157	950	968L					
DB4	044172	960	962	964	976L			
DB5	044175	979L	981					
DB6	044206	972	985L					
DEF	053255	1377	1403	2639L	2695			
DEF0	053271	2641	2646L					
DEF1	053327	2656	2658	2665L				
DEFALTB	074363	5455	5523	5524	6065L			
DEFALTI	074377	5376	5383	5525	6067L			
DEFALTL	074371	5467	5524	5525	6066L			
DEV.DDA	000004	141L						
DEV.DVG	000016	153L						
DEV.DVL	000014	152L						
DEV.FLG	000006	142L						
DEV.JMP	000003	140L						

DEV.MNU	000011	149L		
DEV.MUM	000010	148L		
DEV.NAM	000000	132L		
DEV.RES	000002	136L		
DEV.SPG	000007	147L		
DEV.UNT	000012	150L		
DEVELEN	000000	155E		
DF CLR	000376	108E		
D: CMP	000377	107E		
DHD	053232	1904	2611L	
DHD1	053253	2614	2621L	
DIR.ALD	000025	123L		
DIR.CLU	000015	116L		
DIR.CRD	000023	122L		
DIR.EXT	000010	111L		
DIR.FGN	000020	119L		
DIR.FLG	000016	117L		
DIR.LGN	000021	120L		
DIR.LSI	000022	121L		
DIR.NAM	000000	110L		
DIR.PRO	000013	112L		
DIR.VER	000014	113L		
DIRELEN	000027	125E	419	557
DIRIDL	000015	114E		
DLLH	053335	718	1313	2683L
DLL	053370	894	2707L	3202
DLL0	054011	2715	2720L	
DLL1	054030	2723	2730L	
DLL2	054052	2738L	2744	2749
DLL2.5	054074	2742	2753L	
DLL2.7	054112	2755	2762L	
DLL3	054130	2726	2758	2770E
DLL4	054167	2775	2788L	
DLLB	054204	2736	2794E	
DLS	073264	5637	5650	5676L
DLS1	073274	5685L	5690	5707
DLS2	073342	5692	5694	5696
DNT	047346	1758E	2149	2165
DNT1	050014	1775	1780L	
DNT10	050256	1901L	1927	
DNT11	050330	1903	1929L	
DNT12	050352	1905	1926	1943L
DNT13	050353	1771	1779	1799
DNT14	050362	1935	1952L	1947L
DNT15	050377	1938	1957	1963L
DNT2	050017	1765	1785L	
DNT3	050061	1795	1811L	
DNT4	050070	1815L	1823	
DNT5	050110	1817	1827L	
DNT5.5	050135	1838	1842L	
DNT5.7	050154	1847	1850E	
DNT6	050163	1797	1860L	
DNT7	050170	1862L	1870	
DNT8	050210	1864	1874E	
DNT9	050233	1881	1888L	
DNTA	051004	1811	1860	1899
DNTB	050232	1884E		1968L
DNTC	050333	1878	1932E	

PAGE 135

UNITD	050332	1876	1912	1931E						
DR.IM	000001	137E								
DR.FR	000002	138E								
DRS	054225	1585	1594	1606	1617	1631	1645	1714	1722	2830L
DRS1	054252	2849L	2854							
DRS2	054266	2852	2855L							
DRS3	054270	2843	2860L							
DRSA	054276	1586	1595	1646	1715	1723	2835	2867E	2893	
DRSB	054317	1607	1632	2878E						
DRSC	054330	1618	2885E	2892						
DS	044220	781	998E							
DSPLEN	000040	2779	3285	5124E						
DSPLIM	067106	3190	5120E							
DSPLIN	067056	2735	2762	2780	2788	3284	5108L	5124		
DSPLNA	067061	3427	5110L							
DSPLNB	067072	3291	5114L							
DSPLNC	067102	5118L								
DSPLND	067106	3453	5121L							
DSPLNE	067113	2718	5122L							
DT.CR	000002	144E								
DT.CW	000004	145E								
DT.DD	000001	143E	5380							
DV.EL	000000	133E								
DV.NU	000001	134E								
DW	044240	782	1012E							
DW1	044243	1017L	1026							
EBB	054341	969	1048	1054	1565	1653	1683	2907L		
EBB1	054365	2910	2922L							
EC.CNA	000004	487L								
EC.DDA	000027	506L								
EC.DIF	000017	498L								
EC.DIW	000035	512L								
EC.DNI	000045	520L								
EC.DNR	000046	521L								
EC.DNS	000005	488L								
EC.DSC	000047	522L								
EC.EOF	000001	484L	2354	3628	4452	4886				
EC.EOM	000002	485L								
EC.FAD	000031	508L	4266							
EC.FAP	000026	505L								
EC.FL	000030	507L								
EC.FNF	000014	495L								
EC.FND	000011	492L	4470							
EC.FNR	000034	511L								
EC.FOD	000043	518L								
EC.FUC	000013	494L								
EC.ICN	000016	497L								
EC.IDN	000006	489L								
EC.IFC	000020	499L								
EC.IFN	000007	490L								
EC.ILC	000003	486L								
EC.ILO	000040	515L								
EC.ILR	000012	493L								
EC.ILV	000037	514L								
EC.IOI	000052	525L								
EC.IS	000032	509L	5986							
EC.NCV	000050	523L	5422							
EC.NEM	000021	500L								

CROSS REFERENCE TABLE

EC.NOS	000051	524L							
EC.NPM	000044	519L							
EC.NRD	000010	491L							
EC.NVM	000042	517L							
EC.OTL	000053	526L							
EC.RF	000022	501L							
EC.UNA	000036	513L							
EC.UND	000015	496L							
EC.UUN	000033	510L							
EC.VPM	000041	516L							
EC.WF	000023	502L							
EC.WP	000025	504L							
EC.WPV	000024	503L							
EJECT	044273	783	1034L	1057	1066				
EJEFLG	067350	1037	2529	2533	5226L				
ELSE	045151	784	1156L						
END	046007	785	1322L						
END.	046072	1352E	3393						
END1	046047	1329	1340L						
END2	046067	1342	1347L						
END3	046072	1336	1345	1354L					
ENDFLG	067133	662	907	1355	5138L				
ENDIF	045167	786	1169L						
ENL	000212	89E	3272	3637	5310	5419			
EOFFLG	064111	4461	4858	4885	4912L				
EPO	054372	999	1138	1307	1330	1369	2944L		
EPO2	055045	2953	2971L						
EPO3	055066	2967	2975	2978L					
EPO4	055067	2960	2979L						
EPOA	055075	2946	2965	2972	2984L				
EQU	046102	787	1368L						
EQU1	046126	1372	1382L						
ERR.A	000010	28E	856	1436	1520	1948	2198	2513	2798
ERR.D	000004	27E	1389	2662	2797				3106
ERR.F	000040	30E	2643	2800	3505	3520	3582		
ERR.O	000100	31E	703	866	2801				
ERR.Q	044026	865L	1306	1325	1423	1507			
ERR.P	000200	32E	1192	1849	2802				
ERR.R	000002	26E	1672	2271	2796	2861	2923		
ERR.U	000001	25E	1539	1840	2795	2958	2977		
ERR.V	000020	29E	1688	1964	2799	2917			
ERR1	045223	1190	1195L						
ERRCNT	067123	658	2732	2734	3248	5132L			
ERRFLG	067140	896	902	2720	2957	2980	3362	3364	5141L
ERRMI	045246	1199	1216L						
ERRNZ	045236	1197	1206L						
ERRPL	045242	1198	1211L						
ERRSHD	067125	2756	5133L	5628					
ERRXX	045204	788	1187L						
ERRZR	045232	1196	1201L						
ESC	000033	87E							
EVL	051215	1017	1187	1401	1576	1705	2134L	2907	2950
EVL1	051240	2142	2145L						
EVL10	051345	2189	2219E						
EVL11	051354	2221	2226L						
EVL12	051373	2222	2237L						
EVL13	052017	2223	2252L						
EVL14	052026	2224	2256L						


```

XREF V1.1

```

PAGE 137

EVL15	052035	2254	2261L															
EVL16	052047	2234	2249	2270L														
EVL2	051254	2152	2157L															
EVL20	052054	2157	2184	2282L														
EVL3	051257	2154	2158L															
EVL4	051267	2165L	2192															
EVL5	051300	2150	2151	2166	2167	2168	2169	2175L										
EVL6	051312	2179	2184L															
EVL7	051315	2181	2185L															
EVL7.5	051331	2177	2178	2180	2182	2196L												
EVL8	051332	2153	2155	2170	2197L													
EVL9	051336	2171	2202L															
EVL A	051342	2145	2206E															
EXIT	042325	576	616L	625	3638	5330												
EXPREL	067353	2137	2161	2228	2244	2262	2646	2692	2908	3307	5232L							
EXPWRK	070074	743	3530	5377	5379	6082L	6083	6084										
FB.CHA	000000	469L	4250	4253	4380	4391	4414	4898										
FB.FLG	000001	470L	2753	4253	4270	4308	4335	4347	4350	4414	4416	4900	5323					
		5324																
FB.FWA	000002	471L	4270	4276	4350	4416	4422	4902										
FB.LIM	000006	473L	4281	4286	4427	4906												
FB.LWA	000010	474L	4908															
FB.NAM	000012	475L	476	1511	3650	4286	4929	5344	5363	5373	5375	5458	5464					
		5493																
FB.NAML	000021	476E	3991	5157	5197	5206	5215											
FB.PTR	000004	472L	4276	4281	4347	4422	4427	4904										
FBNEL	000033	477E																
FF	000014	90E	3031															
FLGERA	044017	855L	1087	1120	1139	1143	1159	1173	1281	1308								
FNFP	055104	609	2543	2996L														
FNPF1	055132	2998	3010L															
FNPF2	055145	3015L	3022															
FNPF3	055167	3006	3014	3026L														
FNPA	055200	3003	3031L															
FNPB	055201	3016	3032L															
FORMDP	067215	2996	5178L	5393	5517	5595												
FT.ABS	000000	435E	568	932	1305	1327	1429	1433	1452	5162								
FT.BAC	000003	438E																
FT.ID	000001	537E																
FT.OR	000002	538E	4240	4244	4313	4315	4469											
FT.OU	000010	540E																
FT.OW	000004	539E	4242	4244	4314	4315	4341	4636										
FT.PIC	000001	436E	584	1433	1445	1467	3045	5170										
FT.REL	000002	437E																
FTFLAG	067144	583	666	1303	1326	1444	2370	3044	5148L									
FWBRK2	063273	4766L	4773															
FWBRK3	063307	4768	4775L															
GCP	044076	733	928L															
GRFFLG	067141	664	730	1418	5142L													
GRT	055202	1347	3044L															
GRT1	055216	3052L	3064															
GRT2	055242	3053	3066L															
GSC	055252	956	980	1099	1770	1774	1778	3084L										
GSC3	055274	3095	3100L															
GSC4	055276	3087	3105L															
HBASM1	042263	585	599L															
HBASM2	042301	601	608L															
HEAD A	067052	2558	5100L															

XREF V1.1
PAGE 138

[illegible]

```

XREF V1.1
PAGE 139

```

LST.L	000001	36E	667	1289	2486														
LSTA	045342	1279	1288E	5700															
LSTCTL	067130	668	677	682	754	1283	2484	3196	5135L										
LSTCTLC	067132	680	1237	5137L	5521	5652													
LSTCTLS	067131	676	680	1256	5136L	5519	5639												
LVT	055310	700	3129E																
LVT.	055306	2974	3127L	3381															
LVT0	055313	3131L	3159																
LVT1	055317	3138L	3144																
LVT2	055334	3140	3151L	3154															
LVTa	055344	3130	3157L																
LXI	047303	774	1701L																
MEML	075005	570	6069E	6098															
MEMOVR	057036	3316	3391L																
MI,INXH	000043	69E	3127																
MI.JMP	000303	66E																	
MI.LDA	000072	67E																	
MI.PSHD	000325	70E																	
MI.RET	000311	68E																	
MOD	000101	21E																	
MOV	047322	725	1714L																
MVI	047206	771	1645L																
NL	000012	88E	89	1078	3032	3267	3267	3268	3269	3270	3392	3392	3392						
		3392	3634	3635	3873	4065	4536	4607	4928	4945	5092	5102	5102	5310					
		5327	5424	5497															
NUL2	000000	79E																	
NULL	000200	78E																	
MULTITL	045115	653	655	1128L															
ORB	055351	814	828	831	842	971	979	1020	1022	1335	3061	3063	3067						
		3069	3181L																
ORB1	055373	3189L	3206																
ORB2	056030	3192	3210L																
ORB3	056041	3188	3199	3215L															
ORBA	056031	3185	3211E	3215															
ORBPTR	047120	3189	3214	3292	5129L														
OF,CE	000200	709	4967E	5001	5002	5003													
OF,LD	000100	716	4968E	5000	5001	5002	5003	5004	5005	5006	5007	5008	5010						
		5028	5029	5035	5060	5062	5067	5070	5075	5076	5077								
OFF	071261	5368	5445L																
OFF1	071311	5451	5463L																
OFF3	071334	5459	5473L																
OFF4	071340	5466	5469	5475L															
OFCDDE	070067	698	3524	3550	6081L														
OFCTAB	064174	697	4971E																
ORB	047134	589	592	594	672	684	1001	1003	1311	1343	1458	2284	2313						
		2375	2689	3182	3184	3317	5139L												
OVL.IN	000001	352E																	
OVL.NUM	000014	354E																	
OVL.RES	000002	353E																	
OVL.UCS	000200	355E																	
PAGEDP	047214	1060	3010	3026	5177L	5392	5515	5584											
PAGNUM	047352	663	2544	5228L															
PAS	056051	608	3234L																
PAS0	056101	3243	3245L																
PAS2	056146	3258	3261L																
PASA	056163	613	3262	3267L	3271														
PASAL	000104	3261	3271E																
PASB	056165	3237	3268L																

```
..XREF Vi.i
```

PAGE 140

[illegible]

[illegible]

PAGE 142

SST1	057071	3390	3395L																	
SST2	057073	3400L	3405																	
ST.DBL	000200	48E	1385	2659																
ST.EQU	000002	45E	1376																	
ST.LAB	000001	44E	2690																	
ST.REL	000100	47E	1445	1843	2231															
ST.SET	000003	46E	1402	1402																
ST.UND	000000	43E	1376	1836	2655															
STACK	042200	254E	5294	5297	5325															
STACKL	001032	252E																		
START	042200	573E	5396																	
STATNO	067126	659	689	691	2945	2948	3234	3450	5134L											
STL	045073	795	1119L																	
STL	045104	656	1119	1123L																
STLTXT	066344	112A	5094L	5095																
STXTL	000063	1123	5095E																	
SW.ERR	071222	5338	5418L	5588	5600	5611	5622	5633	5646	5660										
SW.ERS	073046	5574	5626L																	
SW.ERS1	073057	5626	5631L																	
SW.FOR	072217	5549	5592L																	
SW.FOR1	072233	5593	5598L																	
SW.LAR	072361	5569	5615L																	
SW.LAR1	072372	5615	5620L																	
SW.LOF	073206	5564	5650L																	
SW.LQF1	073223	5651	5658L																	
SW.LON	073131	5559	5637L																	
SW.LON1	073145	5638	5644L																	
SW.PAG	072143	5544	5580L																	
SW.PAG1	072157	5581	5586L																	
SW.WID	072275	5554	5604L																	
SW.WID1	072306	5604	5609L																	
SWITAB	072063	5335	5539L																	
SYDD	040130	244E																		
SYMFWA	067216	2436	2973	3380	5183L															
SYMFTK	067220	2589	3409	5184L																
SYMTAB	076003	2423	5183	5184	6104E															
SYSCALL	000377	175E	604	614	617	634	1517	2341	2347	2352	2429	2432	2763							
		3343	3626	3645	3708	3903	3907	4066	4302	4382	4394	4678	4786	4877						
		4947	5278	5286	5290	5321	5378	5425	5427	5457	5498									
T.CHA	064077	4677	4785	4801	4829	4876	4899L	4900	4902	4904	4906	4908	4910							
T.FLG	064100	4468	4635	4901L																
T.FWA	064101	4667	4775	4865	4903L															
T.LIM	064105	4481	4869	4889	4891	4907L														
T.LWA	064107	4646	4670	4752	4778	4868	4909L													
T.PTR	064103	4479	4542	4550	4644	4668	4712	4754	4774	4866	4905L									
TAB	000011	86E	2466	3573	3600	5096	5892													
TBL1	061270	4159L	4165																	
TBL2	061306	4157	4169L																	
TBL3	061310	4162	4173L																	
THR	047117	765	1576L																	
TITLE	045020	796	1086L																	
TITLE	045031	654	1086	1090L																
TLEN	000012	4800	4910E																	
TOKREL	067354	1762	1844	2160	2229	2245	2263	2283	5233L											
TTL1	045036	1092L	1125																	
TTL2	045045	1096L	1101																	
TTL3	045061	1105L	1108																	
TTL4	045071	1094	1097	1111L																

CROSS REFERENCE TABLE

PAGE 143

TTLTXT	066234	1090	5089L	5090			
TTXTL	000062	1091	5090E				
UNL	057145	692	3448E	3472			
UNL0	057250	3465	3476	3483L			
UNL0.3	057277	3496L	3507				
UNL0.5	057305	3495	3501L				
UNL0.7	057323	3503	3511L				
UNL0.9	057351	3514	3518	3524L			
UNL00	057225	3462	3474L				
UNL1	057364	3531L	3536				
UNL10	060060	3597L					
UNL2.5	060002	3547L					
UNL3	060015	3492	3526	3566L	3577		
UNL5	060047	3569	3572	3574	3586L		
UNL9	060057	3596L	3599	3601			
UNLA	060074	3481	3604L	3605			
UNLAL	000027	3481	3605E				
UNT.DIS	000005	164L					
UNT.FLG	000000	161L					
UNT.GRT	000001	162L					
UNT.GTS	000003	163L					
UNT.SIZ	000007	166E					
UDD1	061240	4097L	4107				
UOL	057116	943	1013	1322	3423L		
UOL	057121	888	3424L				
USERFWA	042200	255E	567	569	570	671	5165
VER	000002	19E					
VERS	000026	173E	5280				
WBR	060123	602	2330	3621L			
WBR1	060126	3622L					
WBR2	060317	3627	3642L				
WIDE	067213	5176L	5518	5606			
XTEXT	046357	798	1505L				
XTEXT0	047021	1518	1523L				
XTEXT1	047064	1528	1534	1544L			
XTEXTA	047074	1513	1524	1548L			
XTEXTB	070074	1514	6084E				
XTEXTC	047102	1532	1549L				
XTEXTD	047105	1524	1550L				
TXBFL	001000	5214	6089E	6093			
TXBUF	074237	5211	5212	5213	5214	6093L	
TXFB	067314	1511	1526	3463	5208L		
TXFLB	067142	1323	1505	1545	3459	3471	5143L
TXLINE	067143	2488	2713	3460	5144L		

10064 BYTES FREE

