HP71 HP-IL Module
Internal Design Specification Update

This document provides updated pages for the Internal Design Specification of the HP71 HP-IL module. It is applicable to the version 1.0 preliminary IDS dated January 1984.

The updated pages come from the IDS release, HP Part No. 82401-90023, dated March 1984.

Note that the actual HP-IL ROM 1B, dated August 1984, exhibits some differences versus this IDS update in the Display Driver module, the changes are hand-written on the updated pages (see below).

Replace the following pages:

- BASIC ROUTINES: pages 3 to 7,
- ENTER execution: pages 61 to 65, and pages 69 to 79 with new pages 69 to 80 (one additional page),
- Basic Interface: pages 14 to 18 with new pages 14 to 19 (one additional page),
- Display Driver: all pages 1 to 20 (note the manual corrections on pages 3, 12 and 20, and that all adresses after F36A4 are wrong),
- BASIC UTILITIES: page 7,
- POLL HANDLERS: pages 42 to 45,
- Symbolic Assignments: page 1, and pages 5 to 7.

J-F Garnier, September 2006

Hewlett-Packard -- Portable Computer Division

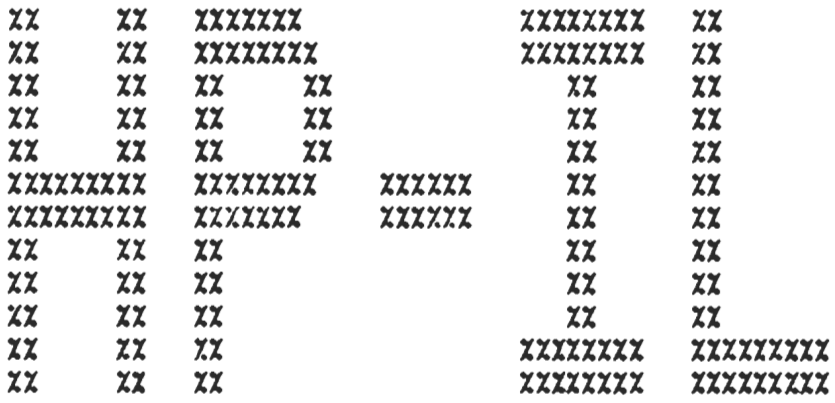
Corvallis, Oregon

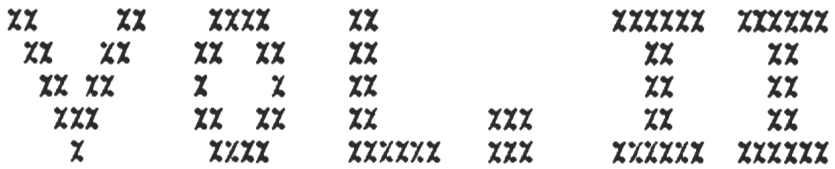
HP-71 HP-IL Module

Internal Design Specification

VOLUME II

Source Listings





HP Part No. 82401-90023

ROM Release 1B -- March 1984

```
107                 * Carry is CLEAR from the D1=D1+ 5 above...TRESD1 doesn't
108                 * affect the carry
109                 *
110 F0FE4 8C00 Tresd1  GOLONG =TRESD1       Restore D1, return "handled"
          00
111                 *-
112                 *-
113                 *
114                 * Not assigned or error...return, carry clear, XM=1
115                 *
116 F0FEA 1B00 PRTIS0  D0=(5) =FUNCD0
          000
117 F0FF1 146           C=DATO A
118 F0FF4 0A            ST=C               Restore status bits from FUNCD0
119 F0FF6 7AEF PRTIS1  GOSUB  Tresd1       Restore D1 from FUNCD1
120 F0FFA 21            P=     1
121 F0FFC 0D            P=P-1              Clear carry, P=0
122 F0FFE 00            RTNSXM             Return, not handled
123                 *-
124                 *-
125 F1000 1B00 PRTIS2  D0=(5) =FUNCD0      Save status bits in FUNCD0
          000
126 F1007 0B            CSTEX
127 F1009 15C2          DATO=C 3
128 F100D 0B            CSTEX
129 F100F 846           ST=0   SaveIt      Initially say don't save it
130 F1012 859           ST=1   MeTalk      Set up MeTalk status bit...
131 F1015 B44           A=A+1  S           ...MeTalk = 1 if A[S]=F
132 F1018 450           GOC    PRTIS,      ...MeTalk = 0 if A[S]=0
133 F101B 849           ST=0   MeTalk
134 F101E D7    PRTIS,  D=C    A           Put device specifier in D[A]
135 F1020 94A           ?C=0   S           Did CHKASN say to find it?
136 F1023 50            GOYES  PRTIS"      No...don't need to save it
137 F1025 856           ST=1   SaveIt      Yes...need to save address
138 F1028 7000 PRTIS"  GOSUB  =START       Set up the device
139 F102C 4DB           GOC    PRTIS0      Error...can't handle the poll
140                 *
141                 * Now address listener, make me talker (conditionally)
142                 *
143 F102F 96B           ?D=0   B           Is this "LOOP"?
144 F1032 61            GOYES  PRTS01      Yes...don't change addressing
145 F1034 879           ?ST=1  MeTalk      Should I be addressed as talker?
146 F1037 A0            GOYES  PRTIS@      Yes...set it up
147 F1039 7000          GOSUB  =ULYL       No...send UNL, LAD n
148 F103D 6700          GOTO   PRTS00      (Check errors at PRTS00)
149                 *-
150                 *-
151 F1041 7616 PRTIS@  GOSUB  Mtyl         Address device as listener
152 F1045 44A  PRTS00  GOC    PRTIS0      HPIL error...don't handle it
153 F1048 866  PRTS01  ?ST=0  SaveIt      Do I need to write it out?
154 F104B 90            GOYES  PRTIS4      No...continue
155 F104D ABB           C=D    X           Yes...copy address from D[X]
156 F1050 15D2          DAT1=C 3           Write out the device address @ D1
157 F1054 729F PRTIS4  GOSUB  PRTIS0       Restore caller's status, D1 (XM=1)
158 F1058 821           XM=0               Clear XM
```

```
159             *
160 F105B 7000          GOSUB  PRTIS5        Get my current address...
161 F105F 07    PRTIS5  C=RSTK               ...pop it off...
162 F1061 DA            A=C    A             ...move it to A[A]...
163 F1063 3402          LC(5)  (PRASCI)-(PRTIS5)  ...Offset of part 2 routine
          000
164 F106A CA            A=A+C  A             (Address of part 2 routine in A)
165 F106C 03            RTNCC                Done, handled
166             *-
167             *-
168 F106E AF2   PREND2  C=0    W
169 F1071 7CE5          GOSUB  Saveit        Deallocate any buffers
170 F1075 583           GONC   PREND3        Go always
171             *-
172             *-
173 F1078 0             CON(1) =FIXSPC       2 nibbles available here
174 F1079               BSS    2-1
175             ********************************************************************
176             ********************************************************************
177             **
178             ** Name:       PRASCI - Send ASCII characters to the loop
179             **
180             ** Category:   PILI/O
181             **
182             ** Purpose:
183             **      Send the ASCII characters to the loop (already set up)
184             **
185             ** Entry:
186             **      MBOX^ points to the desired mailbox
187             **      A[A] contains the length of the string in bytes
188             **      D[A] is the start address of the string
189             **
190             ** Exit:
191             **      If loop error, jumps to ERRORX
192             **      P=0
193             **      D1 positioned following last character sent
194             **
195             ** Calls:      GETMBX,WRITIT,TSAVD0,TRESD0,<ERRORX>
196             **
197             ** Uses.......
198             **  Inclusive: A[A],C,D1,P,FUNCD0,ST[8,3:0]
199             **
200             ** Stk lvls:   3 (pushed D0;WRITIT)(pushed D0;TRESD0)
201             **
202             ** History:
203             **
204             **    Date      Programmer             Modification
205             **  --------    ----------    ---------------------------------
206             **  01/27/84       NZ         Moved PRASER to pack 9 nibbles
207             **  12/15/82       NZ         Updated documentation
208             **  01/27/83       NZ         Modified entry, exit save method,
209             **                            added exit condition on D1
210             **
211             ********************************************************************
212             ********************************************************************
```

```
213 F107A D300          REL(5) =PREND        Address of the final part
          0
214 F107F 09    =PRASCI C=ST
215 F1081 136           CDOEX                ST into D0, D0 value into C[A]
216 F1084 7116          GOSUB  TsavD0        Save status in FUNCD0
217 F1088 06            RSTK=C               Save D0 on RSTK
218 F108A 8E00          GOSUBL =GETMBX       Get the mailbox address
          00
219 F1090 DB            C=D    A
220 F1092 135           D1=C                 Set D1 to the start of the buffer
221             *
222             * Now D1-->buffer, A[A] is length in bytes, D0-->mailbox
223             * Loop is addressed (Talker and Listener(s))
224             *
225 F1095 840           ST=0   =LoopOK       Do not abort with one ATTN hit
226 F1098 8E00          GOSUBL =WRITIT       Transfer the data to the loop
          00
227 F109E 435           GOC    PRASER        Error if carry set
228 F10A1 7AF5          GOSUB  TresD0        Get status back to D0
229 F10A5 07            C=RSTK               Get old D0 from RSTK
230 F10A7 136           CDOEX                Now D0 restored, ST in C[X]
231 F10AA 0A            ST=C                 Restore the status bits
232 F10AC 01            RTN
233             *-
234             *-
235 F10AE 8E00 PREND3   GOSUBL =UTLEND       Unaddress all talkers, listeners
          00
236 F10B4 03   PREND4   RTNCC
237             *-
238             *-
239 F10B6 0             CON(1) =FIXSPC       1 nibble available here
240             ********************************************************************
241             ********************************************************************
242             **
243             ** Name:       PREND - Clean up the loop after PRINT/OUTPUT
244             **
245             ** Category:   LOCAL
246             **
247             ** Purpose:
248             **      Clean up the loop after a PRINT/OUTPUT sequence
249             **
250             ** Entry:
251             **      Device(s) are addressed as listener(s)
252             **      MBOX^ points to the mailbox used
253             **
254             ** Exit:
255             **      D0 points to the mailbox used
256             **      Carry clear (P may be non-zero)
257             **
258             ** Calls:      D1=SR0,SAVEIT,UTLEND
259             **
260             ** Uses.......
261             **  Inclusive: A,B,C,D,R2,R3,D0,D1,P,ST[3:0]
262             **
263             ** Stk lvls:   4 (UTLEND)(SAVEIT)
```

```
264             **
265             ** History:
266             **
267             **    Date     Programmer            Modification
268             **  --------   ----------   --------------------------------
269             **  01/27/84      NZ        Rewrote to fix bug with PRINT not
270             **                          unaddressing the loop (checked
271             **                          LOOP by looking at STMTR1!)
272             **  11/29/83      NZ        Updated documentation
273             **  12/15/82      NZ        Added documentation
274             **
275             ********************************************************************
276             ********************************************************************
277 F10B7      =PREND
278             *
279             * If device code equals OUTPTt, then need to deallocate the
280             * buffer!
281             *
282 F10B7 7CC5         GOSUB  D1=SR0        Device code
283 F10BB 14F          C=DAT1 B             Read in 1 nib
284 F10BE 80D0         P=C    0             Copy device code to P
285 F10C2 1D00         D1=(2) (=STMTR1)+2   Point to device spec
286 F10C6 880          ?P#    =PRINTt       Is this PRINT?
287 F10C9 80           GOYES  PREND1        No...D1 is OK
288 F10CB 1E00         D1=(4) =IS-PRT       Yes...look at IS-PRT
          00
289 F10D1 14F  PREND1  C=DAT1 B
290 F10D4 96A          ?C=0   B             NULL or LOOP?
291 F10D7 DD           GOYES  PREND4        Yes...exit cleanly
292             *
293 F10D9 880          ?P#    =OUTPTt       If OUTPUT, deallocate any buffers
294 F10DC 2D           GOYES  PREND3        Not output...Unaddress talk,listen
295 F10DE 6F8F         GOTO   PREND2        Could be GONC, but leaves a nib
296             ********************************************************************
297             ********************************************************************
298             **
299             ** Name:       OUTPUT - Execute the OUTPUT statement
300             **
301             ** Category:   STEXEC
302             **
303             ** Purpose:
304             **      Send output to the specified device(s)
305             **
306             ** Entry:
307             **      D0 at tokenized device specifier
308             **
309             ** Exit:
310             **      Through mainframe PRINT*
311             **
312             ** Calls:      GETDID,SAVEIT,TRESD0,<PRINT*>,<ERRORX>
313             **
314             ** Uses.......
315             **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,FUNCxx,STMTD1[3:0],STMTR1,
316             **             ST[11:0],all RAM that EXPEXC is permitted to use
317             **
```

```
                ** Stk lvls:   7 (GETDID)
                **
                ** History:
                **
                **    Date     Programmer             Modification
                **  --------   ----------   --------------------------------
                **  11/29/83      NZ        Updated documentation
                **  03/15/83      NZ        Replaced GETMUL with GETDID
                **  12/15/82      NZ        Wrote code and documentation
                **
                ********************************************************************
                ********************************************************************
F10E2 0000 0          REL(5) =OUTPd        OUTPUT decompile
F10E7 0000 0          REL(5) =OUTPp        OUTPUT parse
F10EC 8E00 00 =OUTPUT GOSUBL =GETDID       Get device specifier
F10F2 414  PRASER     GOC    OUTPer        Error with device or loop
F10F5 1F00 000        D1=(5) (=STMTR1)+2   (This is where I save the 7 nibs)
F10FC AF0             A=0    W             Clear position, length
F10FF 159A            DAT1=A 11            (STMTR1)+9 is position, width
F1103 7A55            GOSUB  Saveit        Save the source @ D1
F1107 7495            GOSUB  TresdO        Restore the PC (saved by GETDID)
F110B 1F00 000        D1=(5) =EOLLEN       Point to EOL length, EOL string
F1112 15F6            C=DAT1 7             Read EOLLEN, EOL string
F1116 1E00 00         D1=(4) (=STMTRO)+11  Position to CKINFO location
F111C 15D6            DAT1=C 7             Write it out EOL info out
F1120 1CB             D1=D1- 12            Position to MLFFLG
                *****
                *
                *     LC(2)  (=OUTPTt)*16+#F Set MLFFLG="F", type=OUTPTt
F1123 31F             NIBHEX 31F
F1126 0               CON(1) =OUTPTt
                *
                *****
F1127 14D             DAT1=C B             Write the info out to MLFFLG
                *
                * Now have written the info needed for the hPRTCL handler to
                * do its job
                *
F112A 161             D0=D0+ 2             Skip the t@ used to stop GETDID
F112D 8D00 000        GOVLNG =PRINT*       Now continue with PRINT handler
                *-
                *-
F1134 60D4 OUTPer     GOTO   Errorx
                ********************************************************************
                ********************************************************************
                **
                ** Name:       PRNTIS - Reassign HPIL PRINT device
                ** Name:       DISPIS - Reassign HPIL DISPLAY device
```

```
2491 F289F 8E00          GOSUBL =DTOH          Result in C[A]
           00
2492 F28A5 DA            A=C    A
2493 F28A7 855  ENTSTr   ST=1   sCOUNT         Count input chars.
2494 F28AA 137  ENTST2   CD1EX                 Save D1 (=xqt addr) in R0.
2495 F28AD 108           R0=C
2496 F28B0 8E00          GOSUBL =RESTD1        Restore stack pointer from STMTD1.
           00
2497 F28B6 8A8           ?A=0   A              Is the input count zero?
2498 F28B9 61            GOYES  ENTST3         Yes...skip the read phase.
2499 F28BB 783A          GOSUB  REDCHR         No...input chars.
2500 F28BF 491           GOC    REDCer
2501 F28C2 1B00          DO=(5) =STMTD1        Write the stack pointer to STMTD1.
           000
2502 F28C9 137           CD1EX
2503 F28CC 144           DAT0=C A
2504 F28CF D1   ENTST3   B=0    A              Zero counter to start again.
2505 F28D1 118           C=R0                  Restore D1 (=xqt addr).
2506 F28D4 135           D1=C
2507 F28D7 03            RTNCC
2508             *-
2509             *-
2510 F28D9 874  REDCer   ?ST=1  MemErr         Insufficient memory?
2511 F28DC 60            GOYES  MEMerr         Yes...go to MEMERR
2512 F28DE 6C4A          GOTO   ENTRex         No...set up the error, exit
2513             *-
2514             *-
2515 F28E2 8D00 MEMerr   GOVLNG =MEMERR        Say "Insufficient memory"
           000
2516             *-
2517             *-
2518 F28E9 8D00 D1fstk   GOVLNG =D1FSTK
           000
2519             *-
2520             *-
2521 F28F0 AC9  Fndmbb   C=B    S
2522 F28F3 8C00 Fndmbx   GOLONG =FNDMBX        Find the mailbox
           00
2523             *-
2524             *-
2525 F28F9 8C00 =getdev  GOLONG =GETDev
           00
```

```
2526                    STITLE
2527                ************************************************************
2528                ************************************************************
2529                **
2530                ** Name:        CKmode - Check if the mailbox is controller
2531                **
2532                ** Category:    PILUTL
2533                **
2534                ** Purpose:
2535                **      Check if the mailbox is the loop controller.  If it is
2536                **      not, take a direct error exit.
2537                **
2538                ** Entry:
2539                **      D0 points to the selected mailbox
2540                **
2541                ** Exit:
2542                **      Carry clear
2543                **      Direct exit to error routine if not loop controller
2544                **
2545                ** Calls:       GETDev
2546                **
2547                ** Uses:        ST[3:0]
2548                **
2549                ** Stk lvls:    2 (GETDev)
2550                **
2551                ** History:
2552                **
2553                **    Date      Programmer              Modification
2554                **  --------    ----------    --------------------------------
2555                **  12/19/83       NZ         Updated documentation
2556                **                 SC         Wrote routine
2557                **
2558                ************************************************************
2559                ************************************************************
2560 F28FF 76FF =CKmode GOSUB  getdev         Check if controller
2561 F2903 500         RTNNC                  Controller...return, carry clear
2562 F2906 300         LC(1)  =eBADMD         Not controller...error exit
2563 F2909 20          P=     =ePIL
2564 F290B 6F1A        GOTO   ENTRex          "Invalid Mode"
```

```
2565                    STITLE REQUEST execute
2566                ******************************************************************
2567                ******************************************************************
2568                **
2569                ** Name:        REQST - Execute the REQUEST statement
2570                **
2571                ** Category:    STEXEC
2572                **
2573                ** Purpose:
2574                **      Set up HPIL response to serial poll:
2575                **        If bit 6 if the status byte is set, loop SRQ will be
2576                **        set when the I/O CPU is in device mode.
2577                **      If the I/O CPU is the controller, it will remember the
2578                **      response value for serial poll when it becomes a device.
2579                **
2580                ** Entry:
2581                **      D0 is the PC
2582                **
2583                ** Exit:
2584                **      Through NXTSTM if no error, BSERR if error
2585                **
2586                ** Calls:       GLOOPW,GETARG,PUTE,<NXTSTM>
2587                **
2588                ** Uses.......
2589                **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,STMTD0,ST[11:0],FUNCxx,
2590                **             All RAM EXPEXC is permitted to use
2591                **
2592                ** Stk lvls:   7 (GLOOPW)(GETARG)
2593                **
2594                ** History:
2595                **
2596                **    Date      Programmer              Modification
2597                **  --------    ----------    --------------------------------
2598                **  12/20/83       NZ         Packed, changed call to GETARG to
2599                **                            call GLOOPW first to save a stack
2600                **                            level
2601                **  12/19/83       NZ         Added documentation
2602                **                 SC         Wrote routine
2603                **
2604                ******************************************************************
2605                ******************************************************************
2606 F290F 0000          REL(5) =REQSTd
           0
2607 F2914 0000          REL(5) =REQSTp
           0
2608 F2919 7000 =REQST   GOSUB  =GLOOPW        Get loop number
2609 F291D 7620          GOSUB  GETARG         Get argument
2610 F2921 3500          LC(6)  =mSETS1        Set status length=1 byte
           0000
2611 F2929 7ACB          GOSUB  pute
2612 F292D D9            C=B    A
2613 F292F F2            CSL    A
2614 F2931 F2            CSL    A
2615 F2933 3100          LC(2)  =mSETST        Load low 2 nibs of SET STATUS msg
2616 F2937 24            P=     4
```

```
2617 F2939 3100          LC(2)  =mSTS@4
2618 F293D 76BB          GOSUB  pute          Set status value to B[B] value
2619 F2941 8C54 RQSTRT   GOLONG ENTRTN
           6F
2620               ****************************************************************
2621               ****************************************************************
2622               **
2623               ** Name:       GETARG - Get an argument from memory
2624               **
2625               ** Category:   LOCAL
2626               **
2627               ** Purpose:
2628               **      Get an argument which follows an (optional) loop #
2629               **      (Assumes GLOOP# has been called just before this)
2630               **
2631               ** Entry:
2632               **      All exit conditions of GLOOP#
2633               **      D0 is the PC
2634               **
2635               ** Exit:
2636               **      D0 points to the mailbox
2637               **      B[B] is the value of the argument
2638               **      Carry clear
2639               **      P=0
2640               **
2641               ** Calls:      SAVED0,FNDCHK,SWAPD0,GTYPR+,RESTD0
2642               **
2643               ** Uses.......
2644               **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,STMTD0,ST[11:0],FUNCxx,
2645               **             All RAM EXPEXC is permitted to use
2646               **
2647               ** Stk lvls:   6 (GTYPR+)
2648               **
2649               ** History:
2650               **
2651               **    Date     Programmer            Modification
2652               **  --------   ----------   --------------------------------
2653               **  02/22/84      MZ        Changed GOSUB FNDCHK to GOSUBL
2654               **  12/20/83      MZ        Installed fix for SR #0039-1075(1)
2655               **                          The fix involves moving the call
2656               **                          to GLOOP# to the calling routine
2657               **                          to save one RSTK level, then calling
2658               **                          GETARG
2659               **  12/19/83      MZ        Added documentation
2660               **                SC        Wrote routine
2661               **
2662               ****************************************************************
2663               ****************************************************************
2664 F2947 8E00 GETARG   GOSUBL =SAVED0       Save D0 in STMTD0 for use later
           00
2665 F294D 8E00          GOSUBL =FNDCHK       Find the mailbox
           00
2666 F2953 4D5           GOC    ErrorX        Error...exit
2667 F2956 8E00          GOSUBL =SWAPD0       Save mailbox addr in STMTD0, get PC
           00
```

```
2668 F295C 161          DO=DO+ 2            Skip the leading <tCOMMA>
2669 F295F 8E00         GOSUBL =GTYPR+      Get the status byte
           00
2670 F2965 4B4          GOC    ErrorX       Error...exit
2671 F2968 8C00         GOLONG =RESTDO      Restore mailbox pointer
           00
2672              *-
2673              *-
2674 F296E 7000 ENABfx  GOSUB  =GLOOPW      Get loop number
2675 F2972 71DF         GOSUB  GETARG       Get argument
2676 F2976 6C60         GOTO   ENABL1       Continue with enable code
2677              *-
2678              *-
2679 F297A 0            CON(1) =FIXSPC      1 nibble available here
2680 F297B              BSS    1-1
2681              ********************************************************************
2682              ********************************************************************
2683              **
2684              ** Name:       CKLOPW - Read and check loop # for range
2685              ** Name:       GETLOP - Check loop # for range, put into C[S]
2686              **
2687              ** Category:   LOCAL
2688              **
2689              ** Purpose:
2690              **      Get loop number from memory, if there.  If not there,
2691              **      return loop # 1.  If there, verify that the loop # is
2692              **      in the range 1 <= l <= 3
2693              **
2694              ** Entry:
2695              **      P=0,HEXMODE
2696              **      CKLOPW:D0 points to the loop # expression, if any
2697              **      GETLOP:B[A] is the loop # (in HEX)
2698              **
2699              ** Exit:
2700              **      Carry set
2701              **      C[S] is the loop # - 1
2702              **      If an error is detected, takes a direct exit to BSERR
2703              **
2704              ** Uses:
2705              **      CKLOPW:A,B,C,D,R0-R4,D0,D1,P,FUNCxx,ST[11:0],all RAM
2706              **             EXPEXC is permitted to use
2707              **      GETLOP:A[A],C[W]
2708              **
2709              ** Stk lvls:
2710              **      CKLOPW:6 (GTYPR+)
2711              **      GETLOP:0
2712              **
2713              ** History:
2714              **
2715              **    Date     Programmer          Modification
2716              **  --------   ----------   ------------------------------
2717              **  12/19/83      NZ        Updated documentation
2718              **  03/19/83      NZ        Modified routine
2719              **                SC        Wrote routine
2720              **
```

```
2830                    STITLE PASS CONTROL execute
2831                ******************************************************************
2832                ******************************************************************
2833                **
2834                ** Name:        PASS - Execute the PASS CONTROL statement
2835                **
2836                ** Category:    STEXEC
2837                **
2838                ** Purpose:
2839                **      Execute the PASS CONTROL statement (device specifier
2840                **      is optional)
2841                **
2842                ** Entry:
2843                **      D0 is the PC
2844                **
2845                ** Exit:
2846                **      Through NXTSTM if OK, through BSERR if error
2847                **
2848                ** Calls:       GETDID,START,CKmode,UNLPUT,TALK,PUTE,PUTGF
2849                **
2850                ** Uses.......
2851                **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,STMTD1[3:0],STMTR1,ST[11:0],
2852                **             FUNCxx,All RAM EXPEXC is permitted to use
2853                **
2854                ** Stk lvls:   7 (GETDID)
2855                **
2856                ** History:
2857                **
2858                **    Date      Programmer              Modification
2859                **  --------    ----------    ---------------------------------
2860                **  12/20/83       NZ         Packed 5 nibbles for future use
2861                **  12/19/83       NZ         Added documentation
2862                **                 SC         Wrote routine
2863                **
2864                ******************************************************************
2865                ******************************************************************
2866 F29F4 0000            REL(5) =PASSd
           0
2867 F29F9 0000            REL(5) =PASSp
           0
2868 F29FE 14A  =PASS      A=DAT0 B
2869 F2A01 3100            LC(2)  =tCOMMA
2870 F2A05 D3              D=0    A                Preset device to "LOOP"
2871 F2A07 962             ?A=C   B                Is there a device specifier?
2872 F2A0A B0              GOYES  PASS20           No...use "LOOP"
2873 F2A0C 8E00            GOSUBL =GETDID          Yes...get the device specifier
           00
2874 F2A12 4F3             GOC    Errorx           Error
2875 F2A15 8E00 PASS20     GOSUBL =START           Find and set up the loop
           00
2876 F2A1B 463             GOC    Errorx           Error
2877 F2A1E 7DDE            GOSUB  CKmode           Make sure I'm the loop controller
2878 F2A22 96B             ?D=0   B                Is this either "LOOP" or (nothing)?
2879 F2A25 41              GOYES  PASS30           Yes...just send TCT
2880 F2A27 8E00            GOSUBL =UNLPUT          No...unaddress all listeners
```

```
            00
2881 F2A2D 442          GOC     Errorx          Error
2882 F2A30 8E00         GOSUBL  =TALK           Make the device the talker
            00
2883 F2A36 4B1          GOC     Errorx          Error...set up code, goto BSERR
2884 F2A39 3100 PASS30  LC(2)   =mTCT@4         Send TCT
2885 F2A3D 8E00         GOSUBL  =PUTGF-         Get back response from mailbox
            00
2886 F2A43 4E0          GOC     Errorx          Error
2887 F2A46 890          ?P=     =pACK           Is it an "ACKNOWLEDGE" frame?
2888 F2A49 A6           GOYES   CNTR35          Yes...OK
2889 F2A4B 20           P=      0
2890 F2A4D 300          LC(1)   =eNORDY         No...Device Not Ready error
2891 F2A50 20           P=      =ePIL
2892 F2A52 6000 Errorx  GOTO    =eRRORX
2893            *-
2894            *-
2895 F2A56 8C00 Chksts  GOLONG  =CHKSTS
            00
2896            *-
2897            *-
2898 F2A5C 3300 SETIM0  LC(4)   =mSETIM         Set interrupt mask to zero
            00
2899 F2A62 6D7A         GOTO    putc
2900            *-
2901            *-
2902 F2A66 0            CON(1)  =FIXSPC         3 nibbles available here
2903 F2A67              BSS     3-1
```

```
2904                    STITLE CONTROL ON/OFF execute
2905                ****************************************************************
2906                ****************************************************************
2907                **
2908                ** Name:        CONTRL - Execute the CONTROL ON/OFF statements
2909                **
2910                ** Category:    STEXEC
2911                **
2912                ** Purpose:
2913                **      Execute the CONTROL ON/OFF statements (take or give up
2914                **      control on a loop)
2915                **
2916                ** Entry:
2917                **      D0 is the PC
2918                **
2919                ** Exit:
2920                **      Through NXTSTM if no error, through BSERR if error
2921                **
2922                ** Calls:       CKLOP#,FNDMBD,CHKSTS,PUTE,FNDCH-,PUTC,<NXTSTM>,
2923                **              <REST10>
2924                **
2925                ** Uses.......
2926                **  Inclusive: A,B,C,D,R0-R4,D0,D1,P,STMTD0,ST[11:0],FUNCxx,
2927                **              All RAM EXPEXC is permitted to use
2928                **
2929                ** Stk lvls:    7 (CKLOP#)
2930                **
2931                ** History:
2932                **
2933                **    Date      Programmer              Modification
2934                **  --------    ----------    --------------------------------
2935                **  12/19/83       NZ         Added documentation
2936                **                 SC         Wrote routine
2937                **
2938                ****************************************************************
2939                ****************************************************************
2940 F2A69 0000         REL(5) =CNTRLd
           0
2941 F2A6E 0000         REL(5) =CNTRLp
           0
2942 F2A73 161  =CONTRL D0=D0+ 2              Skip the tON/tOFF token for now
2943 F2A76 710F         GOSUB  CKLOP#         Get the loop # from memory
2944 F2A7A 1F00         D1=(5) =PCADDR        (C[S] is the loop #)
           000
2945 F2A81 143          A=DAT1 A
2946 F2A84 131          D1=A                  Set D1 to the current PCADDR
2947 F2A87 177          D1=D1+ 2+6            Skip the line length, CONTROL token
2948 F2A8A 14B          A=DAT1 B              Read the tON/tOFF token
2949 F2A8D 3100         LC(2)  =tON
2950 F2A91 962          ?A=C   B              Is this CONTROL ON?
2951 F2A94 32           GOYES  CNTR40         Yes...set the controller flag
2952                *
2953                * CONTROL OFF if here
2954                *
2955 F2A96 8E00         GOSUBL =FNDMBD        Clear DISPLAY OK bits
```

```
           00
2956 F2A9C 45B          GOC     Errorx          Error
2957 F2A9F 73BF         GOSUB   Chksts          Check if reset, get status
2958 F2AA3 4EA          GOC     Errorx          Error
2959 F2AA6 3300         LC(4)   =nCLRCA         Clear Controller Active state
           00
2960 F2AAC 703A         GOSUB   putc
2961 F2AB0 41A          GOC     Errorx
2962 F2AB3 6D8E CNTR35  GOTO    RQSTRT          Goto NXTSTM
2963            *-
2964            *-
2965 F2AB7 AC5  CNTR40  B=C     S               Save mailbox in B[S] for REST10
2966 F2ABA 8E00         GOSUBL  =FNDCH-         Find and check the mailbox
           00
2967 F2AC0 419          GOC     Errorx
2968 F2AC3 3300         LC(4)   =nSETCA         Set Controller Active state
           00
2969 F2AC9 731A         GOSUB   putc
2970 F2ACD 448          GOC     Errorx
2971 F2AD0 AC9          C=B     S               Restore mailbox # from B[S]
2972 F2AD3 8C00         GOLONG  =REST10         Restore IO (readdress, etc)
           00
2973            *-
2974            *-
2975 F2AD9 0            CON(1)  =FIXSPC         4 nibbles available here
2976 F2ADA              BSS     4-1
```

```
2977                    STITLE Zero program poll handler
2978                ********************************************************************
2979                ********************************************************************
2980                **
2981                ** Name:        hZERPG - Handler for the ZERO program poll
2982                **
2983                ** Category:    POLL
2984                **
2985                ** Purpose:
2986                **      Handle the ZERO program poll (set interrupt mask=0)
2987                **
2988                ** Entry:
2989                **      None
2990                **
2991                ** Exit:
2992                **      XM=1, carry clear
2993                **
2994                ** Calls:       SAVSTS,FNDMBX,CHKSTS,PUTC,RESSTS
2995                **
2996                ** Uses.......
2997                **  Inclusive: A,B[S],C,D0,P,SNAPBF[37:0]
2998                **
2999                ** Stk lvls:   1 (SAVSTS) (SAVSTS saves the levels in SNAPBF)
3000                **
3001                ** History:
3002                **
3003                **    Date      Programmer              Modification
3004                **  --------    ----------    --------------------------------
3005                **  02/22/84       NZ         Changed call to FNDCHK into two
3006                **                            calls (FNDMBX,CHKSTS) to fix a
3007                **                            bug with mailboxes in manual mode
3008                **                            interrupting the mailbox checking
3009                **  12/19/83       NZ         Added documentation
3010                **                 SC         Wrote routine
3011                **
3012                ********************************************************************
3013                ********************************************************************
3014 F2ADD 8E00 =hZERPG GOSUBL =SAVSTS       Save 5 RSTK levels & status bits
           00
3015 F2AE3 AC1          B=0    S            Counter for which loop is next
3016 F2AE6 760E ZERP10  GOSUB  Fndmbb       Find that mailbox
3017 F2AEA 431          GOC    ZERP30       Not found...exit
3018 F2AED 756F         GOSUB  Chksts       Check it
3019 F2AF1 460          GOC    ZERP20       Error...try next mailbox
3020 F2AF4 746F         GOSUB  SETIM0       Set interrupt mask to 0
3021 F2AF8 B45  ZERP20  B=B+1  S            Go to next loop
3022 F2AFB 5AE          GONC   ZERP10       Go "always" (if fall through, done)
3023                *-
3024                *-
3025 F2AFE 8E00 ZERP30  GOSUBL =RESSTS       Restore RSTK levels, D[A],ST[11:0]
           00
3026 F2B04 00           RTNSXM              Return, XM=1, Carry clear
3027                *-
3028                *-
3029 F2B06 0            CON(1) =FIXSPC      4 nibbles available here
```

```
   3030 F2B07              BSS    4-1
```

```
3031                    STITLE Exception poll handler
3032                ****************************************************************
3033                ****************************************************************
3034                **
3035                ** Name:        hEXCPT - Exception poll handler
3036                **
3037                ** Category:    POLL
3038                **
3039                ** Purpose:
3040                **      Handle the exception poll (check for EOL branch due)
3041                **
3042                ** Entry:
3043                **      None
3044                **
3045                ** Exit:
3046                **      If not ON INTR: XM=1, carry clear
3047                **      If ON INTR pending and due: exits through ONTIMR!
3048                **
3049                ** Calls:       FNDMBX,CHKSTS,PUTC,<ONTIMR>
3050                **
3051                ** Uses.......
3052                **  Inclusive: A,B,C,D0,D1,P,ST[11:0] (also what ONTIMR uses)
3053                **
3054                ** Stk lvls:    3 (CHKSTS)
3055                **
3056                ** History:
3057                **
3058                **    Date      Programmer              Modification
3059                **  --------    ----------    --------------------------------
3060                **  02/22/84       NZ         Split call to FNDCHK into two calls
3061                **                            (FNDMBX,CHKSTS) to fix a bug with
3062                **                            multiple loops, one in manual mode
3063                **  12/19/83       NZ         Added documentation
3064                **                 SC         Wrote routine
3065                **
3066                ****************************************************************
3067                ****************************************************************
3068 F2B0A AC1    =hEXCPT B=0     S               Initialize loop counter to first
3069 F2B0D 7FDD   EXPT10  GOSUB   Fndmbb          Find the current mailbox
3070 F2B11 482           GOC     RtnSXM          If mailbox not found, done
3071 F2B14 7E3F          GOSUB   Chksts          Check it
3072 F2B18 490           GOC     EXPT15          Error...go to next one
3073                *
3074                * FNDCHK returns with status in C[X]
3075                *
3076 F2B1B 0B            CSTEX
3077 F2B1D 870           ?ST=1   =sINTR          Interrupt pending?
3078 F2B20 80            GOYES   EXPT20          Yes...see if ON INTR branch defined
3079 F2B22 B45   EXPT15  B=B+1   S               No...check next loop
3080 F2B25 57E           GONC    EXPT10          Go "always" (if fall thru, OK)
3081                *-
3082                *-
3083                *
3084                * Interrupt pending on mailbox, see if ON INTR branch exists
3085                *
```

```
3086 F2B28 1F00 EXPT20  D1=(5) =ONINTR
           000
3087 F2B2F 147          C=DAT1 A
3088 F2B32 8AE          ?C#0   A            Is the ON INTR address zero?
3089 F2B35 B0           GOYES  EXPT40       No...see if program running
3090               *
3091               * Interrupt pending, but ONINTR=0, set Except and exit for now
3092               *
3093 F2B37 850  EXPT30  ST=1   =Except
3094 F2B3A 21   RtnSXM  P=     1            Clear carry and set XM
3095 F2B3C 0D           P=P-1
3096 F2B3E 00           RTNSXM
3097               *-
3098               *-
3099               *
3100               * Interrupt pending and ONINTR#0, check if program running
3101               *
3102 F2B40 86D  EXPT40  ?ST=0  13           Running?
3103 F2B43 4F           GOYES  EXPT30       No...set Except and keep waiting
3104               *
3105               * See if the ATTN key pressed
3106               *
3107 F2B45 8E00         GOSUBL =CK=ATn      Check if ATTN key has been pressed
           00
3108 F2B4B 5BE          GONC   EXPT30       Yes...wait for next time around
3109               *
3110               * Interrupt pending, ONINTR#0, Running; check if at end of line
3111               *
3112 F2B4E 07           C=RSTK              Current PC is on third RSTK level
3113 F2B50 D5           B=C    A              save first RSTK level in B[A]
3114 F2B52 07           C=RSTK              Pop off the second RSTK level
3115 F2B54 DA           A=C    A              save it in A[A]
3116 F2B56 07           C=RSTK              Pop off the third RSTK level
3117 F2B58 06           RSTK=C                and push it back on
3118 F2B5A DE           ACEX   A            Get the second RSTK level from A[A]
3119 F2B5C 06           RSTK=C                and push it back on
3120 F2B5E DD           BCEX   A            Get the first RSTK level from B[A]
3121 F2B60 06           RSTK=C                and put it back on
3122               *
3123               * Now check if the PC is at an EOL
3124               *
3125 F2B62 131          D1=A                Set D1 to the current PC
3126 F2B65 14B          A=DAT1 B
3127 F2B68 3100         LC(2)  =tEOL        Check if it points to an EOL
3128 F2B6C 966          ?A#C   B            Is it at EOL?
3129 F2B6F 8C           GOYES  EXPT30       No...set Except, wait for next time
3130               *
3131               * We are going to do an end-of-line branch
3132               *
3133 F2B71 137          CD1EX               Save PC on stack
3134 F2B74 06           RSTK=C
3135               *
3136 F2B76 72EE         GOSUB  SETIM0       Set IM=0 to clear interrupt pending
3137 F2B7A 1F00         D1=(5) =ONINTR
           000
```

```
3138 F2B81 147          C=DAT1 A            Read the ONINTR address again
3139 F2B84 08           CLRST               Clear ON ERROR & ON TIMER flags
3140 F2B86 850          ST=1   =sEXTGS      Set external flag
3141 F2B89 8D00         GOVLNG =ONTIMR      Take the jump
           000
3142            *-
3143            *-
3144 F2B90 0            CON(1) =FIXSPC      8 nibbles available here
3145 F2B91              BSS    8-1
```

```
3146                    $TITLE Key definition poll handler
3147                ****************************************************************
3148                ****************************************************************
3149                **
3150                ** Name:        hKYDF - Handler for the keydef poll
3151                **
3152                ** Category:    POLL
3153                **
3154                ** Purpose:
3155                **      Handle the key def poll for HPIL key (#FF)
3156                **
3157                ** Entry:
3158                **      P=0
3159                **      R0[6:5] is the key number
3160                **
3161                ** Exit:
3162                **      If HPIL data and remote then define a colon-def key
3163                **      to execute the statement
3164                **
3165                ** Calls:       ASRC5,FNDMBX,CHKSTS,GETHSS,D1MSTK,CHKSTK,RDST30,
3166                **              STRPcr,D1=AVE,I/OALL
3167                **
3168                ** Uses.......
3169                **  Inclusive: A (If not handled)
3170                **  Inclusive: A,B,C,D,R0,R1,R2,D0,D1,P (If handled)
3171                **
3172                ** Stk lvls:   4 (RDST30)    (If handled...if not, 1)
3173                **
3174                ** History:
3175                **
3176                **    Date      Programmer            Modification
3177                **  --------    ----------    --------------------------------
3178                **  02/22/84        NZ        Split call to FNDCHK into two calls
3179                **                            to fix a bug with multiple loops
3180                **                            with one in manual mode, changed
3181                **                            call to RDST35 to RDST30
3182                **  01/10/84        NZ        Changed size checking to always
3183                **                            get the first 255 characters from
3184                **                            the loop, if more than 255 received
3185                **  12/21/83        NZ        Added code to force valid size
3186                **                            (<4096 nibs) for key def...check
3187                **                            is done BEFORE call to I/OALL!
3188                **  12/19/83        NZ        Added documentation
3189                **  04/01/82        SC        Wrote routine
3190                **
3191                ****************************************************************
3192                ****************************************************************
3193 F2B98 110     =hKYDF  A=R0                  Recall key number...
3194 F2B9B 8E00            GOSUBL =ASRC5         ...from A[6:5]
            00
3195 F2BA1 B64             A=A+1  B
3196 F2BA4 559             GONC   RtnSXM         Not HPIL key...don't handle it
3197                *
3198                * Find out which mailbox has data available
3199                *
```

```
3200 F2BA7 AC1          B=0    S            Start from mailbox #1
3201 F2BAA 724D DFKY10  GOSUB  FndMbb       Find loop B[S] (Sets Device bit)
3202 F2BAE 464          GOC    NoKYDF       No mailbox has data available
3203 F2BB1 71AE         GOSUB  Chksts       Check that mailbox
3204 F2BB5 401          GOC    DFKY20       Error...try next one
3205 F2BB8 D5           B=C    A            Save status bits in B[X]
3206 F2BBA 7000         GOSUB  =GETHSS      Read mailbox's handshake bits
3207 F2BBE 463          GOC    NoKYDF       If abort, exit
3208 F2BC1 870          ?ST=1  =hsRQSR      Is this mailbox requesting service?
3209 F2BC4 80           GOYES  DFKY30       Yes...see if it has data available
3210            *
3211            * Continue on to next mailbox...this one not requesting service
3212            *
3213 F2BC6 B45  DFKY20  B=B+1  S
3214 F2BC9 50E          GONC   DFKY10       Go always
3215            *-
3216            *-
3217 F2BCC      DFKY30
3218            *
3219            * Status bits are in B[X]
3220            *
3221 F2BCC D9           C=B    A            Recall status bits
3222 F2BCE 0B           CSTEX
3223 F2BD0 860          ?ST=0  =sDATAV      Is data available?
3224 F2BD3 3F           GOYES  DFKY20       No...try next mailbox
3225            *
3226            * Read the data from the mailbox and save it on math stack
3227            *
3228 F2BD5 777A         GOSUB  D1=stk       Set D1 to the top of stack
3229 F2BD9 08           CLRST
3230 F2BDB 7749         GOSUB  CHKSTK       See if room left on stack for string
3231 F2BDF 2E           P=     14
3232 F2BE1 31A0         LCHEX  0A           Put <Lf> in B[15:14] (Term. char)
3233 F2BE5 AF5          B=C    W
3234 F2BE8 D8           B=A    A            Put memory limit into B[A]
3235 F2BEA AF0          A=0    W            Clear count, flag -23 indicator
3236 F2BED CC           A=A-1  A            Set count="FFFFF"
3237 F2BEF 8E5A         GOSUBL RDST30       Read the data from the loop
         7F
3238 F2BF5 454  NoKYDF  GOC    NOKYDF       Return no key def if error
3239 F2BF8 7C3A         GOSUB  STRPcr       Strip off trailing <Cr>, if any
3240 F2BFC 133          AD1EX               A[A] is address of top of stack
3241 F2BFF 8E00         GOSUBL =D1=AVE
         00
3242 F2C05 AF2          C=0    W            Clear C[5] for below
3243 F2C08 147          C=DAT1 A            C[A] is bottom of stack
3244 F2C0B E2           C=C-A  A            C[A] is string length in nibbles
3245 F2C0D 81E          CSRB                Convert length to bytes (temp)
3246 F2C10 AF5          B=C    W            Put length into B[A] (for I/OALL)
3247 F2C13 AF2          C=0    W            Truncate string to 255 chars max
3248 F2C16 A6E          C=C-1  B
3249 F2C19 8BD          ?B<=C  A            Is the length currently <=255?
3250 F2C1C 40           GOYES  DFKY40       Yes...leave it as is
3251 F2C1E D5           B=C    A            No...set it to 255.
3252 F2C20 C5   DFKY40  B=B+B  A            Convert back to nibbles
```

```
3253 F2C22 07          C=RSTK              Save 1 level...I/OALL uses three
3254 F2C24 10A         R2=C                  RSTK levels if buffer shrinks
3255 F2C27 3200        LC(3)  =bSTMXQ      Load HPIL stmt execute buffer ID
          0
3256 F2C2C 8F00        GOSBVL =I/OALL      Allocate the buffer
          000
3257 F2C33 11A         C=R2
3258 F2C36 06          RSTK=C              Restore the RSTK level from R2
3259 F2C38 470         GOC    DFKY50       Go if OK
3260             *
3261             * Not enough memory to create the stmt execute buffer
3262             *
3263             * If fail to return a key definition, set SO=0
3264             * XM is zero already
3265             *
3266 F2C3B 840  NOKYDF  ST=0   0            No key definition
3267 F2C3E 03          RTNCC               Handled, no error
3268             *-
3269             *-
3270             *
3271             * Save the input string in the buffer
3272             * D0 points to the buffer header
3273             * D1 points to the buffer start
3274             *
3275 F2C40 137  DFKY50  CD1EX              C[A] is the buffer start address
3276 F2C43 162          D0=D0+ 3            D0 points to the buffer length
3277 F2C46 142          A=DAT0 A            Read the buffer length
3278 F2C49 1F00        D1=(5) =DEFADR
          000
3279 F2C50 81C         ASRB                A[X] is the string length in bytes
3280 F2C53 149         DAT1=A B            Write out the length to the buffer
3281 F2C56 171         D1=D1+ 2            Move to key type
3282 F2C59 BF2         CSL    W            Put buffer start address in C[5:1]
3283 F2C5C 306         LC(1)  6            Type 6: colon def key
3284 F2C5F 15D5        DAT1=C 6            Write type, buffer start address
3285 F2C63 79E9        GOSUB  D1=stk       Set D1 to start of input string
3286 F2C67 162         D0=D0+ 3            Set D0 past the buffer header
3287             *
3288 F2C6A A6C  DFKY60  A=A-1  B            Are all characters written yet?
3289 F2C6D 411         GOC    DFKY70       Yes...exit
3290 F2C70 1C1         D1=D1- 2            No...read next character
3291 F2C73 14F         C=DAT1 B
3292 F2C76 14C         DAT0=C B            Write the character to buffer
3293 F2C79 161         D0=D0+ 2
3294 F2C7C 5DE         GONC   DFKY60       Go always
3295             *-
3296             *-
3297 F2C7F 850  DFKY70  ST=1   0            Do have a key definition
3298 F2C82 03          RTNCC               No error
3299             *-
3300             *-
3301 F2C84 0           CON(1) =FIXSPC      18 nibbles available here
3302 F2C85             BSS    18-1
3303 F2C96             END
```

```
                        STITLE Main loop poll handler
                    ******************************************************************
                    ******************************************************************
                    **
                    ** Name:       PILMLP - HPIL handler for main loop
                    **
                    ** Category:   POLL
                    **
                    ** Purpose:
                    **      Main loop handler code - if display is not offed,
                    **      set ST[LoopOK] true
                    **
                    ** Entry:
                    **      P=0,HEXMODE
                    **
                    ** Exit:
                    **      Carry clear,XM=1
                    **
                    ** Calls:      D1=DST
                    **
                    ** Uses.......
                    **  Inclusive: C[XS],D1,P
                    **
                    ** Stk lvls:   1 (D1=DST)
                    **
                    ** History:
                    **
                    **    Date      Programmer              Modification
                    **  --------    ----------    --------------------------------
                    **  12/21/82       NZ         Updated documentation
                    **  01/17/83       NZ         Changed Search from 4 to 5 (START
                    **                            is now using ST[4] also)
                    **
                    ******************************************************************
                    ******************************************************************
F30E7 1F00 000      =PILMLP D1=(5) =LOOPST      First check if loop is "OFFED"
F30EE 1572          C=DAT1 XS
F30F2 0B            CSTEX
F30F4 870           ?ST=1  =Offed       Is it offed?
F30F7 20            GOYES  PILM05       Set carry if yes
F30F9 0B    PILM05  CSTEX
F30FB 451           GOC    PILMRC       If offed, just return
                    *
                    * Not OFFED by OFFIO...set loop OK true here
                    *
F30FE 7881          GOSUB  D1=DST
F3102 1572          C=DAT1 XS
F3106 0B            CSTEX
F3108 850           ST=1   =LoopOK      Set Loop OK flag true again
F310B 0B            CSTEX
F310D 1552          DAT1=C XS           Write out the statuses
F3111 605E  PILMRC  GOTO   RTNCCX       Return w/carry clear, XM=1
                    *-
                    *-
```

```
    518 F3115 0           CON(1) =FIXSPC      4 nibbles available here
    519 F3116             BSS    4-1
```

```
                        STITLE Service Request Handler
                ****************************************************************
                ****************************************************************
                **
                ** Name:        PILSRQ - HPIL service request handler
                **
                ** Category:    POLL
                **
                ** Purpose:
                **      HPIL service request poll handler - determine SRQ
                **      source, process SRQ
                **
                ** Entry:
                **      P=0,HEXMODE
                **
                ** Exit:
                **      Carry clear,P=0,XM=1
                **
                ** Calls:       SAVSTS,FNDMBX,GETHSS,CHKSTS,PUTCN,GETST-,SFLAG?,
                **              RESSTS
                **
                ** Uses.......
                **  Exclusive:      B[A],C[W],              D1,P
                **  Inclusive: A[W],B[A],C[W],D[15,5],D0,D1,P,SNAPBF[37:0]
                **
                ** Stk lvls:    1 (SAVSTS,RESSTS save all except call to SAVSTS)
                **
                ** NOTE: Must NOT use many RSTK levels OR any status bits
                **
                ** Algorithm:
                **      Check if mailbox SRQ...if not, return
                **      Find which mailbox is requesting service
                **      Check if interrupt pending...if pending, set exception
                **      Check if data available and remote mode and "dormant":
                **        if so, set up HPIL external key
                **      If not interrupt and not (data available and remote)
                **        then continue checking with next loop
                **
                ** History:
                **
                **    Date      Programmer              Modification
                **   --------   ----------    --------------------------------
                **   02/22/84      NZ         Added check for carry from CHKSTS
                **                            (also changed from CHKSET to CHKSTS
                **                            at REQSER to check for manual mode)
                **   10/20/83      NZ         Implemented ER #39-10744 (if the
                **                            first loop requesting service
                **                            does not have anything to do, try
                **                            any other loops for SRQ)
                **   12/21/82      NZ         Updated documentation
                **
                ****************************************************************
                ****************************************************************
F3119 80E   =PILSRQ SREQ?                   First check this is HPIL
F311C 834           ?SR=0
```

```
575 F311F 2F           GOYES  PILMRC       No request pending...exit
576 F3121 824          SR=0
577 F3124 0B           CSTEX
578 F3126 860          ?ST=0  =sMBXsr      Mailbox SRQ?
579 F3129 20           GOYES  PILS00       Set carry if not HPIL
580 F312B 0B   PILS00  CSTEX
581 F312D 43E          GOC    PILMRC       Not HPIL...exit
582            *
583            * This is an HPIL SRQ...service it
584            *
585 F3130 7623         GOSUB  SAVSTS       Save status, 5 levels, D[A]
586 F3134 1F00         D1=(5) =MBOX^
           000
587 F313B 147          C=DAT1 A            Save old MBOX^ value in B[3:1]
588 F313E F2           CSL    A
589 F3140 D5           B=C    A            Mbox value in B[3:1], # in B[0]
590            *                           Set up for mbox #1
591 F3142 816  PILS20  CSRC                Shift mailbox number into C[S]
592 F3145 8E00         GOSUBL =FNDMBX      Look for the mailbox
           00
593 F314B 4D6          GOC    PILS50       Not found...done
594 F314E 7D70         GOSUB  GETHSS       Read handshake nibbles (2)
595 F3152 870          ?ST=1  =hsRQSR      Requesting service?
596 F3155 90           GOYES  REQSER       Yes...see what it is
597 F3157 E5   PILS23  B=B+1  A            No...try next mailbox
598 F3159 D9           C=B    A
599 F315B 56E          GONC   PILS20       Go always (if more than 16, no)
600            *-
601            *-
602            *
603            * Mailbox requesting service pointed to by D0
604            *
605 F315E 8E00 REQSER  GOSUBL =CHKSTS      Check this loop for reset,man mode
           00
606 F3164 42F          GOC    PILS23       Error...try next one
607 F3167 3300         LC(4)  =mSTSTC      Request status & clear SRQ
           00
608 F316D 8E00         GOSUBL =PUTCN
           00
609 F3173 8E00         GOSUBL =GETST-      Read the mailbox's status
           00
610 F3179 5B0          GONC   REQS10       (OK)
611 F317C 890          ?P=    =eABORT      Error from ATTN key hit?
612 F317F A3           GOYES  PILS50       Yes...exit routine NOW
613 F3181 F6           CSR    A            No...status is in C[3:1]
614 F3183 20           P=     0            (P was =ePIL)
615 F3185 0B   REQS10  CSTEX
616            *
617            * Check if there is an interrupt pending
618            *
619 F3187 860          ?ST=0  =sINTR       Interrupt pending?
620 F318A 80           GOYES  REQS30       No...check if data is available
621 F318C 850          ST=1   =Except      Yes...set exception flag and exit
622 F318F 57C          GONC   PILS23       Go always...check next for remote ke
623            *-
```

```
                *-
F3192           REQS30
                *
                * Check if there is data available
                *
F3192 860               ?ST=0  =sDATAV      Data available?
F3195 2C                GOYES  PILS23       No...try next mailbox
                *
                * Data is available...check if I/O CPU is in remote mode
                *
F3197 860               ?ST=0  =sRMOTE      Remote mode?
F319A DB                GOYES  PILS23       No...ignore the data, try next mbox
                *
                * Data available, remote mode...check if the HP-71 is dormant
                *
F319C 3100              LC(2)  =flDORM
F31A0 8E00              GOSUBL =sFLAG?      Check the dormant flag
      00
F31A6 508               GONC   PILS23       Not dormant...try next mailbox
                *
                * Data available, remote mode, dormant...generate special key
                *
F31A9 1F00              D1=(5) =KEYPTR
      000
F31B0 321F              LCHEX  FF1
      F
F31B5 15D2              DAT1=C 3            Set to one key, keycode = "FF"
                *
                * Restore MBOX^ value, restore status, RSTK, D[A], and exit
                *
F31B9 D9        PILS50  C=B    A
F31BB F6                CSR    A            Get mailbox # back to C[X]
F31BD 1F00              D1=(5) =MBOX^
      000
F31C4 15D2              DAT1=C 3            Restore the mailbox address
F31C8 72C2              GOSUB  RESSTS       Restore status, 5 levels, D[A]
F31CC 00                RTNSXM              Exit with carry clear, XM=1
                *-
                *-
F31CE 0                 CON(1) =FIXSPC      1 nibble available here
F31CF                   BSS    1-1
                ********************************************************************
                ********************************************************************
                **
                ** Name:       GETHSS - Get 2 handshake nibbles from I/O CPU
                **
                ** Category:   PILI/O
                **
                ** Purpose:
                **      Read the two handshake nibbles from I/O CPU to the HP-71
                **      and put into ST[7:0]
                **
                ** Entry:
                **      D0 points to HPIL mailbox
                **
```

```
675                ** Exit:
676                **      The two handshake nibbles from I/O CPU are in ST[7:0]
677                **      Carry clear
678                **
679                ** Calls:      None
680                **
681                ** Uses:
682                **  Inclusive: ST[7:0]
683                **
684                ** Stk lvls:   0
685                **
686                ** History:
687                **
688                **    Date     Programmer          Modification
689                **  --------   ----------   --------------------------------
690                **  09/29/83      NZ        Updated documentation
691                **  04/01/83      SC        Wrote routine
692                **
693                ********************************************************************
694                ********************************************************************
695 F31CF 0B    =GETHSS CSTEX                  Save C[X] in ST, put ST in C[X]
696 F31D1 160          DO=DO+ =oINHS
697 F31D4 14E          C=DATO B               Read two nibbles of handshake
698 F31D7 180          DO=DO- =oINHS
699 F31DA 0B           CSTEX                  Put back into ST, restore C[X]
700 F31DC 01           RTN                    Return, carry clear
```

```
 1            *
 2            *       N   N  ZZZZZ   &     DDDD    SSS   PPPP
 3            *       N   N      Z  & &     D  D  S   S  P   P
 4            *       NN  N     Z   & &     D  D  S      P   P
 5            *       N N N    Z     &      D  D   SSS   PPPP
 6            *       N  NN   Z     & & &   D  D      S  P
 7            *       N   N  Z      &  &    D  D  S   S  P
 8            *       N   N  ZZZZZ   && &  DDDD    SSS   P
 9            *
10            *
11                    TITLE  Display driver <840301.1344>
12 F3637              ABS    #F3637        TIXHP6 address (fixed)
13           ************************************************************
14           ************************************************************
15           **
16           ** Name:       BDISPJ - HPIL Character-oriented display routine
17           **
18           ** Category:   PILI/O
19           **
20           ** Purpose:
21           **      Routine to display characters on HPIL devices
22           **
23           ** Entry:
24           **      A[B] is a data byte
25           **      HEX mode
26           **
27           ** Exit:
28           **      A[B] is the data byte from entry
29           **      Display status bits restored
30           **      HEX mode, carry clear
31           **
32           ** Calls:      CHKASN,SETLP,FNDMBX,START,GTYPE,MTYL,FINDA,
33           **             GETMBX,WRITIT,SENDIT,SENDI+,PUTD,PUTX,END,
34           **             MOVCUR,MOVCU+,DO=CUR,DO@CUR,Clear?,SendBf,
35           **             BLANKC,LCleft,DSPCL?
36           **
37           ** Uses.......
38           **  Exclusive: A[15:2],B[W],C[W],D[A],           D0,D1,P,(ST)
39           **  Inclusive: A[15:2],B[W],C[W],D[15:13],D[5:0],D0,D1,P,(ST)
40           **
41           ** Stk lvls:   4 (START)
42           **
43           ** NOTE:
44           **      Does not alter A[B], returns (DSPSTA+3) in STatus bits
45           **
46           ** History:
47           **
48           **    Date     Programmer          Modification
49           **  --------   ----------   ------------------------------
50           **  02/24/84      NZ        Reworked and packed to fix bug
51           **                          with (non-HP82163 device, insert
52           **                          mode, protected field following,
53           **                          and delete through end of line)
54           **  09/28/83      NZ        Updated documentation
55           **  06/24/83      NZ        Fixed bug of losing <Cr> if DISP
```

```
                **                              device is a printer device
                **  05/18/83      NZ            Changed return from GTYPE to
                **                              match new exit conditions of same
                **  04/14/83      NZ            Added check to ignore NULL char
                **  02/16/83      NZ            Removed Talker code (doesn't work
                **                              with multiple loop displays)
                **  12/09/82      NZ            Added documentation
                **
                ****************************************************************
                ****************************************************************
                Esc     EQU     #1B             <Escape>
                Bs      EQU     #08             <Backspace>
                *
                RepCur  EQU     0               Status bit...Replace the cursor
                *
                Delete  EQU     4               Status bit...Delete character
                CurLft  EQU     5               Status bit...Cursor direction
                SetCur  EQU     6               Status bit...Set vs move cursor
                Protec  EQU     SetCur          Status bit...Hit protected char?
                *
                *-
                *-
F3637           =BDISPJ
F3637 1B00          DO=(5) =IS-DSP      IS assignment
      000
F363E 15E6          C=DATO 7            Read it in...
F3642 7000          GOSUB  =CHKASN      Check if assigned...
F3646 551           GONC   DISP00       Assigned
F3649 6A63 DISPoF   GOTO   DISPOF       This is NOT assigned...return
                *-
                *-
                *
                * Now get back the correct loop for the display
                *
F364D 7000 DISP02   GOSUB  =SETLP       SETUP sets C[S] to current mbox
F3651 7000          GOSUB  =FNDMBX      FNDMBX sets MBOX^ to current mbox
F3655 4D2           GOC    DISPNS       If carry, not found...not set up?
F3658 67C0          GOTO   DISPOK
                *-
                *-
F365C 1900 DISP00   DO=(2) =DSPSET      Status nibble for display
F3660 0B            CSTEX
F3662 1562          C=DATO XS           Read in status...
F3666 0B            CSTEX
F3668 860           ?ST=0  =LoopOK
F366B ED            GOYES  DISPoF       Loop has been offed...exit now
F366D D7            D=C    A            Put address in D[A] for START
F366F 94E           ?C#0   S
F3672 11            GOYES  DISPNS       ...not current...set it up
F3674 870           ?ST=1  =DispOK      Currently set up?
F3677 6D            GOYES  DISP02       Yes...check if mailbox is there
                *
                * Display is NOT set up...check if this is a new assignment
                *
                * (New assignments have BOTH ST(=H82163) and ST(=Printr) true)
```

```
110              *
111 F3679 860            ?ST=0  =H82163       Not HP82163A?
112 F367C A0             GOYES  DISPN0        No...this is NOT a new assignment
113 F367E 860            ?ST=0  =Printr       Not printer?
114 F3681 50             GOYES  DISPN0        No...this is NOT a new assignment
115 F3683 850  DISPNS     ST=1   =DispOK       Reuse this status as a flag
116              *
117              * If ST(DispOK)=1, then need to check accessory ID here
118              *
119 F3686        DISPN0
120              *
121              * Loop is NOT set up for DISPLAY IS
122              *
123              * Save character on RSTK before calls to START, GTYPE, etc
124              *
125 F3686 D6             C=A    A
126 F3688 06             RSTK=C               Push the character
127              *
128              * Call START, with device specifier in D[A]...
129              *
130 F368A 8E00           GOSUBL =START        Set up Loop
          00
131 F3690 4E3            GOC    DISPN.        Error
132 F3693 860            ?ST=0  =DispOK       Are the status bits OK already?
133 F3696 33             GOYES  DISPn4        Yes...continue
134              *
135              * Get the accessory ID of the device in A[B]
136              *
137 F3698 8E00           GOSUBL =GTYPE        Returns Acc Id in A[B]
          00
138 F369E 403            GOC    DISPN.        Error if carry
139              *
140              * If no response, then A[B] is zeroed by GTYPE
141              *
142              * Now set DSPSET true, set up other bits of DSPSET using B[B],
143              * then restore all and return
144              *
145 F36A1 840            ST=0   =H82163       Preclear these statuses
146 F36A4 840            ST=0   =Printr
147 F36A7 80D1           P=C    1             Copy class nibble into P
148 F36AB 891            ?P=    1             Mass storage class?
149 F36AE 84             GOYES  DISPN1        Yes...error
150 F36B0 882            ?P#    2             Printer class device?
151 F36B3 80             GOYES  DISPn3        No...check if HP82163A
152              *
153              * Printer class device
154              *
155 F36B5 850            ST=1   =Printr
156 F36B8 501            GONC   DISPn4        Go always
157              *-
158              *-
159 F36BB 20   DISPn3    P=     0
160 F36BD 3103           LCHEX  30            HP82163A accessory id
161 F36C1 966            ?A#C   B
162 F36C4 50             GOYES  DISPn4        Not an HP82163A
```

All addresses below are wrong.

← C=A A

```
163 F36C6 850          ST=1   =H82163
164 F36C9      DISPn4
165            *
166            * Now set up the display as a listener (Acc ID in A[B])
167            *
168 F36C9 8E00         GOSUBL =MTYL          (Character is on RSTK)
          00
169 F36CF 462  DISPN.  GOC    DISPN1         Error
170 F36D2 850          ST=1   =DispOK        Display set up
171 F36D5 1A00         DO=(4) =DSPSET
          00
172 F36DB 0B           CSTEX
173 F36DD 1542         DATO=C XS             Write it back out
174 F36E1 0B           CSTEX
175 F36E3 1900         DO=(2) =IS-DSP
176 F36E7 DB           C=D    A
177 F36E9 15C2         DATO=C 3              Write out the address
178 F36ED 07           C=RSTK
179 F36EF DA           A=C    A              Restore the character to A[B]
180 F36F1 20           P=     0
181 F36F3 5C2          GONC   DISPOK         Go always
182            *-
183            *-
184            *
185            * If here, had a loop error...clear DISPLAY IS
186            *
187 F36F6 07   DISPN1  C=RSTK
188 F36F8 DA           A=C    A              Restore character from RSTK
189 F36FA 1A00         DO=(4) =IS-DSP        DISPLAY IS assignment
          00
190 F3700 146          C=DATO A              Read code nibble into C[3]
191 F3703 F6           CSR    A              (code into C[XS])
192 F3705 0B           CSTEX
193 F3707 85B          ST=1   11             Set "OFF"ed flag
194 F370A 0B           CSTEX
195 F370C F2           CSL    A              Back to C[3]
196 F370E AB2          C=0    X
197 F3711 A3E          C=C-1  X              C[X]=FFF
198 F3714 15C3         DATO=C 4              OFF the display
199 F3718 6052         GOTO   DISPEX         Done
200            *-
201            *-
202 F371C 6792 DISPOx  GOTO   DISPOX         Done, don't check carry
203            *-
204            *-
205            *
206            * Loop is set up now
207            *
208 F3720      DISPOK
209            *
210            * First ensure that not in an escape sequence
211            *
212 F3720 1B00         DO=(5) =ESCSTA        Escape status
          000
213 F3727 15E0         C=DATO 1              Read it...
```

```
F372B A0E          C=C-1  P            ...decrement it...
F372E 484          GOC    DISPnE       Not escape
                  *
                  * This is in an escape sequence...what do I do?
                  *
                  *
                  * Check if printer...if so, return
                  *
F3731 870          ?ST=1  =Printr
F3734 8E           GOYES  DISP0x       Exit, restore all levels
                  *
                  * Not a printer...continue
                  *
F3736 90A          ?C=0   P            Is it "escape"?
F3739 90           GOYES  DISP1        Yes...check further
F373B 846  Dspsn0  ST=0   SetCur       No...send the character without
F373E 6552         GOTO   DspSn0         repositioning the cursor
                  *-
                  *-
                  *
                  * Escape mode
                  *
F3742 844  DISP1   ST=0   Delete       Assume NOT a delete until proven
                  *                    otherwise
F3745 8F00         GOSBVL =FINDA       A[B] is value
      000
F374C 34           CON(2) \C\          Right arrow
F374E 6C0          REL(3) RArrow
F3751 44           CON(2) \D\          Left arrow
F3753 7D0          REL(3) LArrow
F3756 05           CON(2) \P\          Delete character
F3758 890          REL(3) DelChr
F375B F4           CON(2) \O\          Delete character with wrap
F375D 390          REL(3) DelChr
F3760 E4           CON(2) \N\          Insert char with wrap
F3762 990          REL(3) InsChr
F3765 B4           CON(2) \K\          Delete through end of line
F3767 011          REL(3) DelLin
F376A 30           CON(2) 3            Cursor far right
F376C 4C0          REL(3) FarRt
F376F 40           CON(2) 4            Cursor far left
F3771 001          REL(3) FarLft
F3774 00           CON(2) 0            Others...
F3776 68F1         GOTO   EscSnd       Send <Esc> <character> & return
                  *-
                  *-
                  *
                  * If <Lf>: Send it immediately, independent of current mode
                  * If <Cr>: If (not Printr): send immediately (Don't set cursor)
                  *                    else: transmit buffer, then <Cr>
                  * If chr$(0): Ignore it entirely if not in escape sequence
                  * If <anything else> and <Printr>: return without action
                  *
F377A 968  DISPnE  ?A=0   B            Is A[B]=0?
F377D F9           GOYES  DISP0x       Yes...exit.
```

```
268 F377F 31A0          LCHEX  0A            <Lf>
269 F3783 962           ?A=C   B
270 F3786 5B            GOYES  DspsnO        Send it
271 F3788 30D           LCHEX  D             Preload <Cr>
272 F378B 870           ?ST=1  =Printr
273 F378E B0            GOYES  DISP.1        Check further in printer code
274 F3790 962           ?A=C   B             Is it a <Cr>?
275 F3793 8A            GOYES  DspsnO        Yes...don't reposition the cursor
276 F3795 6251          GOTO   DISP2         No...process the character
277             *-
278             *-
279 F3799 966 DISP.1    ?A#C   B             Is it a <Cr>?
280 F379C 08            GOYES  DISPOx        No...Exit, no action
281 F379E 77C3          GOSUB  Clear?        Is the clear flag set?
282 F37A2 489           GOC    DspsnO        Yes...send only the <Cr>
283             *
284             * This is a printer, and I got a <Cr>...
285             * need to send whole buffer
286             *
287 F37A5 1900          D0=(2) (=DSPBFS)-2   (Clear? leaves D0 @ DSPSTA+3)
288 F37A9 31F5          LC(2)  95
289 F37AD 161 DISP.2    D0=D0+ 2
290 F37B0 14A           A=DAT0 B
291 F37B3 968           ?A=0   B
292 F37B6 80            GOYES  DISP.3        End of buffer (Logical)
293 F37B8 A6E           C=C-1  B             End of buffer (Physical)?
294 F37BB 51F           GONC   DISP.2        No...try next character
295             *
296             * Now D0 points to first "non-character"
297             *
298 F37BE AF0 DISP.3    A=0    W             Clear for ASRB below
299 F37C1 132           ADOEX
300 F37C4 3400          LC(5)  =DSPBFS
          000
301 F37CB 135           D1=C                 Set D1 @ DSPBFS also
302 F37CE EA            A=A-C  A
303 F37D0 81C           ASRB                 Now A[A] is # of characters
304             *
305             * Set up for HPIL transfer
306             *
307 F37D3 7000          GOSUB  =GETMBX       Restore the HPIL mailbox to D0
308 F37D7 8E00          GOSUBL =WRITIT       Send the buffer
          00
309 F37DD 20            P=     0
310 F37DF 31D0          LCHEX  0D            Restore the <Cr>
311 F37E3 DA            A=C    A
312 F37E5 460           GOC    DISPEx        Exit if error
313 F37E8 7B93 DISP..   GOSUB  Putd          Send it to the printer
314 F37EC 6C71 DISPEx   GOTO   DISPEX
315             *-
316             *-
317             *
318             * Code to check if Insert or Delete
319             *
320 F37F0       DelChr
```

```
                *
                * Delete character (Either HP82163A or "other")
                *
F37F0 854               ST=1    Delete          This IS a delete
F37F3 7CC1              GOSUB   SendBf          Send to end of line
F37F7 6171              GOTO    DISPEX          Restore, etc.
                *-
                *-
F37FB           InsChr
                *
                * Insert character (Send Esc Q Esc N to turn on insert mode)
                *
                * (Esc Q is for HP82163A, as it does not understand Esc N)
                *
F37FB 7000              GOSUB   =GETMBX         Get back the mailbox first
F37FF 35B1              LCHEX   1B511B          Esc Q Esc
      15B1
F3807 8E00              GOSUBL  =PUTX
      00
F380D 4ED               GOC     DISPEx          Error if carry
F3810 6A2F              GOTO    Dspsn0          Now send the current char (N)
                *-
                *-
F3814           RArrow
                *
                * Right arrow
                *
F3814 7573              GOSUB   D0@CUR
F3818 14E               C=DAT0  B
F381B 96A               ?C=0    B
F381E F4                GOYES   DISPox          At end of buffer NOW
F3820 845               ST=0    CurLft
F3823 846       Arrow   ST=0    SetCur          This is NOT just a set, but MOVE
F3826 6981              GOTO    DISPMC          MOVCUR, DISPOX
                *-
                *-
F382A           LArrow
                *
                * Left arrow
                *
F382A 855               ST=1    CurLft
F382D 55F               GONC    Arrow           Go always (FINDA:RTNCC)
                *-
                *-
F3830           FarRt
                *
                * Cursor far right
                *
F3830 845               ST=0    CurLft          This is cursor RIGHT
F3833 7653      Farxx   GOSUB   D0@CUR          C[B] is current cursor value
F3837 DA                A=C     A               Save cursor value in A[B]
F3839 846               ST=0    SetCur          This is NOT just a SET, but MOVE
F383C 875       FarRt1  ?ST=1   CurLft          Is this LEFT?
F383F E0                GOYES   FarRt2          Yes...don't check for end
F3841 7843              GOSUB   D0@CUR          No...check if at end already
```

```
374 F3845 14E          C=DATO B
375 F3848 96A          ?C=0   B
376 F384B C0           GOYES  FarRt3        Already at far right of buffer
377 F384D 850  FarRt2  ST=1   RepCur        Reposition the cursor at new loc.
378 F3850 7C62         GOSUB  MOVCU+
379 F3854 57E          GONC   FarRt1        Moved it...move it again
380 F3857 20   FarRt3  P=     0             Force P back to zero
381 F3859 3130         LC(2)  3             Cursor far right
382 F385D 865          ?ST=0  CurLft        Is this RIGHT?
383 F3860 40           GOYES  FarEnd        Yes...exit
384 F3862 E6           C=C+1  A             No...(LEFT=4)
385 F3864 7D33 FarEnd  GOSUB  DO=CUR
386 F3868 148          DATO=A B             Restore the cursor value
387 F386B DA           A=C    A
388 F386D 6641 DISPox  GOTO   DISPOX        Finish it up
389            *-
390            *-
391 F3871      FarLft
392 F3871 855          ST=1   CurLft
393 F3874 5EB          GONC   Farxx         Go always
394            *-
395            *-
396            *
397            * Delete through end of line
398            *
399 F3877 8F00 DelLin  GOSBVL =SCNRT
          000
400            *
401            * Check if no protected fields after this...if none, send
402            * <Esc> J (Clear to end of screen)
403            *
404 F387E 4D0          GOC    DelL0         If carry, reached end of buffer
405 F3881 130          DO=A
406 F3884 14E          C=DATO B             Read in indicated character
407 F3887 96E          ?C#0   B
408 F388A 31           GOYES  DelL1         Protected field
409 F388C D4   DelL0   A=B    A
410 F388E 7DE2         GOSUB  GTPEsc        GETMBX, PUTEsc
411 F3892 415          GOC    DISPeX        Carry exit
412 F3895 31A4         LCASC  \J\
413 F3899 6E4F         GOTO   DISP..        Putd, DISPEX
414            *-
415            *-
416            *
417            * Delete to protected field
418            *
419 F389D 870  DelL1   ?ST=1  =H82163       Is the display an HP82163A?
420 F38A0 90           GOYES  DelL2          Yes...skip turning off insert mode
421 F38A2 7413         GOSUB  InsOff        No...turn off insert mode
422 F38A6 453          GOC    DelEx         Error...set up the char, exit
423 F38A9 AF2  DelL2   C=0    W
424 F38AC DB           C=D    A
425 F38AE EE           C=A-C  A
426 F38B0 81E          CSRB
427 F38B3 E6           C=C+1  A             Increment for current character
```

```
428                *
429                * Now C[A] is count of blanks to send
430                *
431 F38B5 DA           A=C    A              Copy count to A[A]
432 F38B7 D7           D=C    A              D[A]=count
433 F38B9 8E00         GOSUBL =BLANKC        Blanks (Clear the items)
          00
434 F38BF AF5          B=C    W              Copy to B[7:0]
435 F38C2 8E00         GOSUBL =SENDI+        Get mailbox, Send A[A] blanks
          00
436 F38C8 431          GOC    DelEx          If carry, abort
437                *
438                * Now back up to starting point
439                *
440 F38CB DB           C=D    A
441 F38CD DA           A=C    A              Count to A[A]
442 F38CF C4           A=A+A  A              Double count for <Esc> D
443 F38D1 73D1         GOSUB  SendBk         Send Esc D's (count in A[A])
444 F38D5 460          GOC    DelEx          Error...set up the char, exit
445 F38D8 72F2         GOSUB  InsOn          Turn on insert mode (if not HP82163)
446 F38DC 20   DelEx   P=     0
447 F38DE 31B4         LCASC  \K\            Restore original character (K)
448 F38E2 DA           A=C    A
449 F38E4 6480 DISPeX  GOTO   DISPEX         Done...exit
450                *-
451                *-
452 F38E8      DISP2
453                *
454                * Check if it is an <Esc>...if so, do NOTHING until next char
455                *
456 F38E8 31B1         LC(2)  Esc
457 F38EC 962          ?A=C   B              Is this an escape?
458 F38EF F0           GOYES  DISPoX         Yes...exit, no change
459                *
460                * Check if backspace - if so, do a backspace and return
461                *
462 F38F1 3180         LC(2)  Bs             <Bs>
463 F38F5 966          ?A#C   B              Is this a backspace?
464 F38F8 A0           GOYES  DISP25         No...check further
465                *
466                * This is a backspace
467                *
468 F38FA 6F2F         GOTO   LArrow         Carry MUST be clear for LArrow
469                *-
470                *-
471 F38FE 65B0 DISPoX  GOTO   DISPOX         Jump (GOYES out of range)
472                *-
473                *-
474 F3902 185  DISP25  DO=DO- 6              Move to DSPSTA from ESCSTA
475 F3905 15E2         C=DATO 3
476 F3909 0A           ST=C                  Restore user status for DSPCL?
477 F390B 8F00         GOSBVL =DSPCL?
          000
478 F3912 1A00         DO=(4) (=DSPSTA)+3    Restore display status for me
          00
```

```
479 F3918 14E          C=DATO B
480 F391B 1A00         DO=(4) =DSPSET       Point to the HPIL status nibble
          00
481 F3921 1562         C=DATO XS            Recall the HPIL status from RAM
482 F3925 0A           ST=C
483               *
484               * Check if cursor is at end of buffer
485               *
486 F3927 7A72         GOSUB  DO=CUR
487 F392B 14E          C=DATO B
488 F392E D5           B=C    A             Copy cursor value to B[B]
489 F3930 31F5         LC(2)  95
490 F3934 9E5          ?B<C   B             Reached physical end of buffer?
491 F3937 02           GOYES  DISP30        No...check if insert mode
492               *
493               * Cursor is at end of buffer...check if insert or replace mode
494               *
495 F3939 870          ?ST=1  =Insert
496 F393C 2C           GOYES  DISPoX        Exit, no error (no room)
497               *
498               * At end of buffer, not insert...send char, backspace
499               *
500 F393E 7000         GOSUB  =GETMBX       Get mailbox
501 F3942 3500         LCHEX  441B00
          B144
502 F394A AE6          C=A    B             (char)&<esc>&"D"
503 F394D 8E00         GOSUBL =PUTX         Send it
          00
504 F3953 6510         GOTO   DISPEX        Exit
505               *-
506               *-
507 F3957         DISP30
508               *
509               * Cursor is NOT at end of buffer...check if insert or replace
510               *
511 F3957 860          ?ST=0  =Insert       Insert mode?
512 F395A 73           GOYES  DspSnd        Not Insert...send the char
513               *
514               * Insert mode...call SendBf (It checks for HP82163A)
515               *
516 F395C 844          ST=0   Delete        This is NOT delete
517 F395F 7060         GOSUB  SendBf        Send to end of line
518 F3963 856          ST=1   SetCur        Set the cursor to new spot...
519 F3966 5F3          GONC   DspSn2        If OK, position it
520               *
521               * Following jump taken ONLY if entered through DISPEX
522               * (Packing technique)
523               *
524 F3969 5A4  DISPEX  GONC   DISPOX        If no carry, finish up
525 F396C 4C0          GOC    DspErr        Go always
526               *-
527               *-
528 F396F 7C02 EscSnd  GOSUB  GTPEsc        GETMBX, PUTEsc
529 F3973 846          ST=0   SetCur        ...DON'T set the cursor
530 F3976 512          GONC   DspSn1        Go unless interrupted
```

```
531 F3979 840  DspErr  ST=0   =LoopOK      Interrupted
532 F397C 840          ST=0   =DispOK      (If interrupted, display not OK)
533 F397F 1B00         DO=(5) =DSPSET      Rewrite display settings
          000
534 F3986 0B           CSTEX
535 F3988 1542         DATO=C XS
536 F398C 0B           CSTEX
537 F398E 452          GOC    DISPOX       Go always...exit
538             *-
539             *-
540 F3991       DspSnd
541             *
542             * Send the character and return
543             *
544 F3991 856          ST=1   SetCur       SET the cursor to next position
545 F3994 7000 DspSn0  GOSUB  =GETMBX      Find the mailbox...
546 F3998 D6   DspSn1  C=A    A            ...copy character to C[B]...
547 F399A 79E1         GOSUB  Putd         ...Send the character
548 F399E 4AD          GOC    DspErr       Interrupted
549 F39A1 866          ?ST=0  SetCur       Set the new cursor position?
550 F39A4 01           GOYES  DISPOX       No...exit
551 F39A6 7FB1 DspSn2  GOSUB  Clear?       Check if Clear is set
552 F39AA 490          GOC    DISPOX       Yes...exit (Don't move cursor)
553 F39AD 845          ST=0   CurLft       No...move the cursor RIGHT
554 F39B0 7901 DISPMC  GOSUB  MOVCUR
555 F39B4       DISPOX
556             *
557             * Now restore status bits and return
558             *
559 F39B4       DISPOF
560 F39B4 1B00         DO=(5) (=DSPSTA)+3  Display status bits
          000
561 F39BB 15E2         C=DATO 3
562 F39BF 0A           ST=C                Restore them
563 F39C1 03   RtnCC   RTNCC               Done...return, carry clear
564             ****************************************************************
565             ****************************************************************
566             **
567             ** Name:       SendBf - Insert/delete a char, send line if needed
568             **
569             ** Category:   LOCAL
570             **
571             ** Purpose:
572             **      Insert/delete a character, even if this is an HP82163A
573             **      display device
574             **
575             ** Entry:
576             **      ST(Insert):
577             **            if 1, insert (send from position through end)
578             **               (send character from A[B] first)
579             **      ST(Delete) is type:
580             **            if 1, delete (send from next char to end,
581             **               append blank)
582             **            if 0, insert (send char from A[B], then to end)
583             **
```

```
                ** Exit:
                **      A[B] is not changed from entry
                **      Carry clear:
                **        All OK (P=0)
                **      Carry set:
                **        Interrupted (P=error code)
                **
                ** Calls:      SCNRT,GETMBX,PUTEsc,PUTD,WRITIT,LCleft
                **
                ** Uses.......
                **  Exclusive: A[15:2],B[W],C[W],     D0,D1   ST[Protec]
                **  Inclusive: A[15:2],B[W],C[W],D[A],D0,D1,P,ST[Protec,3:0]
                **
                ** Stk lvls:   2 (WRITIT)(SCNRT)
                **
                ** History:
                **
                **    Date     Programmer            Modification
                **  --------   ----------   --------------------------------
                **  02/24/84      NZ        Rearranged code to allow common
                **                          subroutines for turning off and
                **                          on the insert cursor on display
                **                          device
                **  06/24/83      NZ        Packed code by no longer preserve
                **                          D1 in this routine
                **  06/02/83      NZ        Added code to do Esc N (Insert w/
                **                          wrap)
                **  12/09/82      NZ        Added documentation
                **
                ********************************************************************
                ********************************************************************
F39C3           SendBf
                *
                * Find first character NOT to send (Either EOB or protected)
                *
F39C3 8F00          GOSBVL =SCNRT       Scan right
      000
                *
                * SCNRT returns A[A]-->past unprotected item, carry set if end
                * of buffer, D[A] is pointer to first after current position,
                * B[B] contains the entry A[B]
                *
F39CA 5C0           GONC   NotEnd       If carry, at end of buffer
                *
                * If Insert and End of buffer, return (Do nothing)
                *
F39CD 860           ?ST=0  =Insert      Is it NOT insert?
F39D0 70            GOYES  NotEnd       Not insert...continue
F39D2 864           ?ST=0  Delete       Is it a delete?
F39D5 CE            GOYES  RtnCC        No...buffer is full, insert: exit
F39D7           NotEnd
                *
                * B[B] is the new character...saved here for now
                *
                * D[A] is first char after current position in buffer
```

?ST=1 Delete
GOYES NotEnd
GONC Sendex

```
                *
F39D7 AF2       C=0     W       Clear high bits for CSRB below
F39DA DB        C=D     A       Start of string in C[A]
F39DC 135       D1=C            Start of string in D1
F39DF 846       ST=0    Protec  Check if protected field
F39E2 130       D0=A
F39E5 14E       C=DAT0  B
F39E8 96A       ?C=0    B
F39EB 50        GOYES   NotPro  Not protected (EOB)
F39ED 856       ST=1    Protec
F39F0 137 NotPro CD1EX          Bring pointer back to C[A]...
F39F3 135       D1=C            ...And copy back to D1
F39F6 EE        C=A-C   A       # of nibbles to send
F39F8 81E       CSRB            C[A] is length to send (bytes)
F39FB DA        A=C     A       A[A] is length to send (bytes)
                *
                * Now D1 points past start of buffer, A[A] is a character count
                *
                * Get the mailbox address into D0 now...
                *
F39FD 7000      GOSUB   =GETMBX Alters only C,D0
                *
                * Now D0 points to the mailbox
                *
                * Check if Protec is set...if so, and in insert mode, and not
                * HP82163A, then send <Esc>R to turn OFF insert mode
                *
F3A01 870       ?ST=1   =H82163 HP82163A?
F3A04 62        GOYES   Send#   Yes...continue
F3A06 876       ?ST=1   Protec  Protected?
F3A09 A1        GOYES   Send-   Yes...continue
                *
                * Not HP82163A, not protected...just send the char (or delete
                * escape sequence)
                *
F3A0B D0        A=0     A
F3A0D 864       ?ST=0   Delete  Is this a delete?
F3A10 90        GOYES   Send+  No...just the character
F3A12 7D61      GOSUB   PUTEsc  Yes...send Esc...
F3A16 480       GOC     Sendex
F3A19 D9  Send+ C=B     A       Copy B[B] (the character)
F3A1B 7861      GOSUB   Putd    ...send the character
F3A1F D4  Sendex A=B    A       Restore the character from B[B]
F3A21 01        RTN             Exit, preserve carry
                *-
                *-
                *
                * This is not HP82163A, Protected
                *
F3A23 7391 Send- GOSUB  InsOff  Turn off insert mode, if on
F3A27 47F       GOC     Sendex  Error...restore, return
                *
                * Check if insert...if so, send the character in B[B] first
                * If not insert (if delete), skip first character in buffer
                *
```

```
693             * Check if this is the logical end of buffer
694             *
695 F3A2A 1C1  Send#   D1=D1- 2            Point to first character
696 F3A2D 14F          C=DAT1 B            Check the character for EOB
697 F3A30 171          D1=D1+ 2            Restore pointer to next char
698 F3A33 96E          ?C#0   B            End of line?
699 F3A36 40           GOYES  Send00       No...check if need to adjust
700 F3A38 D0           A=0    A            Yes...set =0
701 F3A3A 874  Send00  ?ST=1  Delete       Delete?
702 F3A3D A1           GOYES  SendNI       Yes...NOT insert
703             *
704             * This is an insert
705             *
706 F3A3F 1C1          D1=D1- 2            This is an insert...
707 F3A42 96A          ?C=0   B            Is it End of buffer?
708 F3A45 90           GOYES  Send02       Yes...skip this adjustment
709 F3A47 876          ?ST=1  Protec       Is it protected?
710 F3A4A 40           GOYES  Send02       Yes...leave A[A] unchanged
711 F3A4C E4           A=A+1  A            Increment count
712             *
713             * Now A[A] is corrected character count, D1 @ first char to
714             * be sent.
715             *
716 F3A4E D9   Send02  C=B    A            Read the character from B[B]
717 F3A50 7331         GOSUB  Putd         Send the character
718 F3A54 4AC          GOC    Sendex       Error...exit
719             *
720             * This is the entry point for a delete
721             *
722 F3A57      SendNI
723             *
724             * Now retransmit the line...
725             *
726 F3A57 8E00         GOSUBL =WRITIT      Send the data to the loop
          00
727             *
728             * If carry set, ATTN hit...return
729             *
730 F3A5D 41C          GOC    Sendex       Exit after restoring A[B]
731             *
732             * Done with transfer now...check if delete; if so, send blank
733             *
734 F3A60 864          ?ST=0  Delete
735 F3A63 D0           GOYES  SendLs       Insert...no trailing blank
736 F3A65 3102         LCASC  \ \          Delete...
737 F3A69 7A11         GOSUB  Putd         ...send a trailing blank
738 F3A6D 41B          GOC    Sendex       Exit if error
739             *
740             * Now D1 points to the "Next" character...subtract current
741             * position and divide by 2 to get # of bytes sent
742             *
743 F3A70 866  SendLs  ?ST=0  Protec       Is this NOT protected field?
744 F3A73 90           GOYES  SendL1       Not protected...back up
745 F3A75 7551         GOSUB  InsOn        Turn insert mode back on
746 F3A79 45A          GOC    Sendex
```

```
747 F3A7C AF0  SendL1  A=0    W            Clear high bits for ASRB below
748 F3A7F 133          AD1EX
749 F3A82 3400         LC(5)  =DSPBFS
          000
750 F3A89 EA           A=A-C  A
751 F3A8B 81C          ASRB                A[A] is # bytes from buffer start
752 F3A8E 1F00         D1=(5) =CURSOR
          000
753 F3A95 D2           C=0    A
754 F3A97 14F          C=DAT1 B            Read the cursor...
755 F3A9A EA           A=A-C  A            Now A[A] is # backspaces to send
756 F3A9C C4           A=A+A  A            Double for <Esc> D
757 F3A9E DC           ABEX   A            Now character in A[B], # in B[A]
758 F3AA0 814          ASRC
759 F3AA3 814          ASRC                Save character in A[15:14]
760 F3AA6 D4           A=B    A            Count back to A[A]
761 F3AA8 7201 SendBk  GOSUB  LCleft       Load C with Esc D Esc D
762 F3AAC AF5          B=C    W
763 F3AAF 8E00         GOSUBL =SENDIT      Send the sequence
          00
764 F3AB5 810          ASLC
765 F3AB8 810          ASLC                Restore A[B] from A[15:14]
766 F3ABB 01           RTN                 Don't alter carry
767             ****************************************************************
768             ****************************************************************
769             **
770             ** Name:       MOVCUR - Move the cursor right/left
771             ** Name:       MOVCU+ - Move the cursor permanently (no restore)
772             **
773             ** Category:   LOCAL
774             **
775             ** Purpose:
776             **      Move the cursor in the direction specified by CurLft
777             **      status bit (Similar to mainframe routine by same name)
778             **
779             ** Entry:
780             **      CurLft set to move left, clear to move right
781             **      P=0
782             **
783             ** Exit:
784             **      Contents of A[A] restored upon exit
785             **      Carry set if no move
786             **      Carry clear if moved, cursor positioned on display
787             **      Clears ST(=LoopOK) if interrupted
788             **
789             ** Calls:      D0=CUR,MOVC60,GETMSK,SENDI+,LCleft
790             **
791             ** Uses.......
792             **  Exclusive: A[15:5],B[W],C[W],D[A],   P
793             **  Inclusive: A[15:5],B[W],C[W],D[A],D0,P,ST[3:0]
794             **
795             ** Stk lvls:   2 (SENDI+)
796             **
797             ** NOTE: Does not alter A[A]
798             **
```

```
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  02/24/84      NZ        Moved MOVC60 to inline code (at
                **                          MOVC10)
                **  12/09/82      NZ        Added documentation
                **
                ********************************************************************
                ********************************************************************
F3ABD 840   MOVCUR  ST=0    RepCur          Do NOT replace cursor
F3AC0 71E0  MOVCU+  GOSUB   DO=CUR
F3AC4 14E           C=DATO  B
F3AC7 D7            D=C     A               Save original value in D[B]
F3AC9 D8            B=A     A               Save original character in B[A]
F3ACB 14A   MOVC10  A=DATO  B
F3ACE CC            A=A-1   A               Assume LEFT first
F3AD0 875           ?ST=1   CurLft          Is it LEFT?
F3AD3 60            GOYES   MOVC12          Yes...good choice
F3AD5 E4            A=A+1   A               No...undo LEFT,
F3AD7 E4            A=A+1   A                 do RIGHT
F3AD9 31F5  MOVC12  LC(2)   95
F3ADD 9E6           ?A>C    B               Would this be past end of display?
F3AE0 C6            GOYES   MOVC50          Yes, then restore original value
F3AE2 148           DATO=A  B               No, then update cursor position
F3AE5 D4            A=B     A               Save original char in A[B]
F3AE7 8F00          GOSBVL  =GETMSK         Get bit map (Alters B[A],C,D0,P)
      000
F3AEE D8            B=A     A               Resave original char in B[B]
F3AF0 15A0          A=DATO  1               Read mask nibble
F3AF4 0E06          A=A&C   P
F3AF8 79A0          GOSUB   DO=CUR
F3AFC 90C           ?A#0    P               Is it protected?
F3AFF CC            GOYES   MOVC10          Yes, then keep looking
                *
                * Now calculate how far to move cursor, and which direction...
                * ...and restore cursor value
                *
F3B01 D0            A=0     A               Clear high nibbles of A[A]
F3B03 14A           A=DATO  B               Read in cursor position
F3B06 DB            C=D     A
F3B08 870           ?ST=1   RepCur          Replace the cursor?
F3B0B 50            GOYES   MOVC15          Yes...don't restore it
F3B0D 14C           DATO=C  B               Restore original cursor position
F3B10 B6A   MOVC15  A=A-C   B               Offset (Bytes) in A[B]
F3B13 37B1          LCHEX   431B431B        Right arrows
      34B1
      34
F3B1D 590           GONC    MOVC20          If carry, left arrow
                *
                * Left arrows needed
                *
F3B20 7A80  MOVC17  GOSUB   LCleft          Left arrows
F3B24 BE8           A=-A    B
F3B27 AFD   MOVC20  BCEX    W               Move arrows to B[W], char to C[B]
```

```
851 F3B2A D7          D=C    A            Save char in D[B]
852 F3B2C 866         ?ST=0  SetCur       Is this a move or a set?
853 F3B2F 90          GOYES  MOVC30       No...MOVE that # of chars
854             *
855             * This is a set cursor...if next char is the destination, exit
856             *
857 F3B31 CC          A=A-1  A
858 F3B33 8A8         ?A=0   A
859 F3B36 21          GOYES  MOVC45       Exit w/o sending any(char in C,D)
860             *
861             * Must MOVE the cursor...send <Esc> C|D (A is # moves)
862             *
863 F3B38 C4  MOVC30  A=A+A  A            Double for <Esc>
864 F3B3A 8E00        GOSUBL =SENDI+      Get mailbox, send left arrows
        00
865 F3B40 550         GONC   MOVC40       No interrupt...ok
866 F3B43 840         ST=0   =LoopOK      Interrupt...clear =LoopOK
867 F3B46 DB  MOVC40  C=D    A
868 F3B48 DA  MOVC45  A=C    A            Restore the original character...
869 F3B4A 03          RTNCC               ...and return
870             *-
871             *-
872 F3B4C     MOVC50
873 F3B4C 866         ?ST=0  SetCur       Is it NOT SetCur?
874 F3B4F 11          GOYES  MOVC55       Not SetCur...OK to not move
875             *
876             * SetCur...need to take action if unable to move right
877             *
878             * First restore the cursor
879             *
880 F3B51 DB          C=D    A
881 F3B53 14C         DATO=C B            DO is still at cursor...
882 F3B56 D0          A=0    A
883 F3B58 A6C         A=A-1  B
884 F3B5B CC          A=A-1  A            A[B]=FE (A=-A B will make this 2)
885             *
886             * Go move the cursor left 1 position (since this is SetCur,
887             * MOVC17 reduces the count by one, therefore A[B] is now -2)
888             *
889 F3B5D 52C         GONC   MOVC17       Go always
890             *-
891             *-
892 F3B60      MOVC55
893 F3B60 DB          C=D    A            C(B)=Original cursor
894 F3B62 14C         DATO=C B            Restore original cursor
895 F3B65 D4          A=B    A            Restore original char from B[B]
896 F3B67 02          RTNSC
897             ****************************************************************
898             ****************************************************************
899             **
900             ** Name:       Clear? - Check if the clear bit is set in DSPSTA
901             **
902             ** Category:   LOCAL
903             **
904             ** Purpose:
```

```
905                **      Set/clear carry if clear bit in DSPSTA is set/clear
906                **
907                ** Entry:
908                **      None
909                **
910                ** Exit:
911                **      Carry set if ST[Clear] is set, else clear
912                **      DO @ DSPSTA+3
913                **
914                ** Calls:      None
915                **
916                ** Uses.......
917                **  Inclusive: C[X],DO
918                **
919                ** Stk lvls:   0
920                **
921                ** History:
922                **
923                **    Date      Programmer              Modification
924                **  --------    ----------    ---------------------------------
925                **  09/28/83      NZ          Added documentation
926                **
927                ********************************************************************
928                ********************************************************************
929 F3B69 1B00 Clear?  DO=(5) (=DSPSTA)+3   Point to status
           000
930 F3B70 15E2         C=DATO 3            Read in 3 nibbles of status
931 F3B74 0B           CSTEX               Now check if CLEAR is set...
932 F3B76 870          ?ST=1  =Clear
933 F3B79 20           GOYES  Clear1       Set/clear carry...
934 F3B7B 0B   Clear1  CSTEX               (Restore my status)
935                    *
936                    * If carry set, then =Clear is set
937                    *
938 F3B7D 01           RTN                 Return, carry unchanged
939                    *-
940                    *-
941 F3B7F 7000 GTPEsc  GOSUB  =GETMBX       Get the mailbox
942 F3B83 31B1 PUTEsc  LC(2)  Esc           Send an Escape
943 F3B87 8C00 Putd    GOLONG =PUTD
           00
944                ********************************************************************
945                ********************************************************************
946                **
947                ** Name:       DO@CUR - Set DO to the current cursor position
948                **
949                ** Category:   LOCAL
950                **
951                ** Purpose:
952                **      Set DO to the cursor position in the display
953                **
954                ** Entry:
955                **      None
956                **
957                ** Exit:
```

```
                **      DO at cursor position
                **      Carry clear
                **      C[A] is cursor value (from =CURSOR)
                **
                ** Calls:      DO=CUR
                **
                ** Uses.......
                **  Inclusive: C[A],DO
                **
                ** Stk lvls:   1 (DO=CUR)
                **
                ** History:
                **
                **    Date     Programmer          Modification
                **  --------   ----------   --------------------------------
                **  02/18/83      NZ        Added DO=CUR call, renamed to
                **                          DO@CUR
                **  12/09/82      NZ        Added documentation
                **
                ********************************************************************
                ********************************************************************
F3B8D 7410 DO@CUR  GOSUB  DO=CUR        Leaves DO pointing to cursor loc
F3B91 D2           C=0    A
F3B93 14E          C=DATO B
F3B96 161          DO=DO+ 2             (=CURSOR)-(=DSPBFS)
F3B99 132          ADOEX                Save A[A] in DO, set A[A] to DSPBFS
F3B9C CA           A=C+A  A
F3B9E CA           A=C+A  A
F3BA0 132          ADOEX                Restore A[A], set DO to cursor
F3BA3 03   Rtncc   RTNCC
                *-
                *-
F3BA5 1B00 DO=CUR  DO=(5) =CURSOR
      000
F3BAC 01           RTN
                *-
                *-
F3BAE 37B1 LCleft  LCHEX  441B441B      Esc D Esc D
      44B1
      44
F3BB8 01           RTN
                *-
                *-
F3BBA 860  InsOff  ?ST=0  =Insert       Insert mode?
F3BBD 6E           GOYES  Rtncc         No...leave alone
                *
                * This is not HP82163A, protected, insert mode...temporarily
                * disable insert mode
                *
F3BBF 7CBF         GOSUB  GTPEsc        Get mailbox, Put Esc...
F3BC3 400          RTNC                 (Error)
F3BC6 3125         LCASC  \R\           ...R
F3BCA 6CBF         GOTO   Putd
                *-
                *-
```

```
1010 F3BCE 870  InsOn   ?ST=1   =H82163      Is this an HP82163A?
1011 F3BD1 2D           GOYES   Rtncc        Yes...exit
1012 F3BD3 860          ?ST=0   =Insert      Am I in insert mode?
1013 F3BD6 DC           GOYES   Rtncc        No...exit
1014 F3BD8 77AF         GOSUB   PUTEsc       Send <Esc>N to turn insert on
1015 F3BDC 400          RTNC
1016 F3BDF 31E4         LCASC   \N\
1017 F3BE3 63AF         GOTO    Putd
1018            *-
1019            *-
1020 F3BE7 0            CON(1)  =FIXSPC      11 nibbles available here
1021 F3BE8              BSS     11-1
1022 F3BF7              END
```

```
                *
F3CEC AC2       C=0   S          Preclear "FIND" flag
F3CEF 20        P=    0
                *
                * Check if this is OK as is...
                *
F3CF1 96A       ?C=0  B          Is it LOOP or NULL?
F3CF4 00        RTNYES           Yes..."not" set up
F3CF6 B26       C=C+1 XS
F3CF9 A2E       C=C-1 XS
F3CFC 500       RTNNC            This is OK as is
F3CFF B36       C=C+1 X
F3D02 A3E       C=C-1 X
F3D05 4F0       GOC   CHKAS0
F3D08 02        RTNSC            This is NOT a HPIL assignment!
                *-
                *-
F3D0A 0         CON(1) =FIXSPC   11 nibbles available here
F3D0B           BSS   11-1
                *-
                *-
F3D15           CHKAS0
                *
                * Check if this is not assigned (nibble 3="F")
                *
F3D15 23        P=    3
F3D17 B06       C=C+1 P
F3D1A A0E       C=C-1 P          Alter carry only...not value!
F3D1D 20        P=    0          Reset P to 0!
F3D1F 400       RTNC             Not defined...return!
F3D22 BF6       CSR   W          Now code nibble in C[XS]
F3D25 92E       ?C#0  XS
F3D28 60        GOYES CHKAS1     This is not an address...
F3D2A 6470      GOTO  CHKAS9     This is an address!
                *-
                *-
                *
                * If here, have either iobuffer, type, or "OFF"ed assignment
                *
F3D2E BF6  CHKAS1 CSR   W        C[1] is the code nibble!
F3D31 80D1        P=C   1        Copy C[1] into P
F3D35 80CF        C=P   15       Use C[S] to test it
                *
                * If C[S] is >=8, then "OFF"ed (RTNSC)
                *
F3D39 A46         C=C+C S
F3D3C AC2         C=0   S        Clear it again!
F3D3F 20          P=    0
F3D41 400         RTNC           If carry, "OFF"ed!
                *
                * Now either iobuffer or type
                *
~~F3D44 8081~~    ~~Def  4~~
~~F3D48 830~~     ~~?P=  Saguer~~  ~~Is this a single entry buffer?~~
~~F3D4B 71~~      ~~GOYES CHKAS2~~  ~~Yes...process it!~~
```

```
1170 F56A2 133  hCPY3+  AD1EX
1171 F56A5 131          D1=A                Copy D1==>A
1172 F56A8 CA           A=A+C  A            Add offset to data start
1173 F56AA 8E00         GOSUBL =ASLC4       Rotate into A[8:4]
           00
1174               *
1175               * Now get the file type (from the source)
1176               *
1177 F56B0 1CF          D1=D1- (oFLENh)-(oFTYPh) Move to file type
1178 F56B3 1583         A=DAT1 4            Read it
1179               *
1180               * check if BASIC file...if so, set flag "BASIC"
1181               *
1182               Basic   EQU    0
1183 F56B7 840          ST=0   Basic
1184 F56BA 3341         LC(4)  =fBASIC
           2E
1185 F56C0 23           P=     3
1186 F56C2 916          ?A#C   WP
1187 F56C5 50           GOYES  hCPY3f
1188 F56C7 850          ST=1   Basic        Set Basic flag
1189 F56CA         hCPY3f
1190               *
1191               * Rotate file type into A[9:6], file start into A[14:10]
1192               *
1193 F56CA 8E00         GOSUBL =ASLC6
           00
1194 F56D0 119          C=R1                Read back the length...
1195 F56D3 E6           C=C+1  A            ...add 1 to round UP...
1196 F56D5 81E          CSRB                ...convert to bytes!
1197 F56D8 DA           A=C    A            (NOT WP: nibble 5 is always zero)
1198 F56DA 101          R1=A                Now size, type, and start are set
1199 F56DD 860          ?ST=0  Basic        Is this NOT a BASIC file?
1200 F56E0 52           GOYES  hCPY3g       Not BASIC...continue
1201               *
1202               * This is a BASIC file...chain it first!
1203               *
1204 F56E2 8E00         GOSUBL =ASRC10      File start ==> A[A]
           00
1205 F56E8 20           P=     0
1206 F56EA D2           C=0    A
1207 F56EC 3113 3152    LC(2)  (=oFLENh)+(oBSsod) +(IFLENH)
1208 F56F0 EA           A=A-C  A
1209               *
1210               * Now A[A] is the start of the file header
1211               *
1212 F56F2 11A          C=R2
1213 F56F5 10B          R3=C                Save R2 in R3 for now...
1214 F56F8 8F00         GOSBVL =CHAIN-      Chain the file
           000
1215 F56FF 11B          C=R3
1216 F5702 10A          R2=C                Restore R2 from R3!
1217 F5705 6A50 hCPY3g  GOTO   hCPY39       Get the destination name, do it!
1218               *-
1219               *-
```

```
2099            **      Source, destination info on SAVSTK (under POLLSV)
2100            **
2101            ** Exit:
2102            **      P=0
2103            **      Carry set: Error...error # in C[3:0]
2104            **      Carry clear:
2105            **        XM=0: handled
2106            **        XM=1: not handled
2107            **
2108            ** Calls:       CKBITL,hRNMsb,FINDF+,FINDFx,SAVDIR,D1=SCR,hPUTDR,
2109            **              ENDTAP
2110            **
2111            ** hRNMsb calls RDINFO
2112            **
2113            ** Uses.......
2114            **  Inclusive: A-D,R0,R1,R3,D0,D1,P,ST[8,5:0],SCRTCH
2115            **
2116            ** Stk lvls:   6 (FINDF+)
2117            **
2118            ** History:
2119            **
2120            **    Date      Programmer              Modification
2121            **  --------    ----------    ---------------------------------
2122            **  06/02/83       NZ         Rewrote parts to pack code and
2123            **                            share routines with PURGE, SECURE
2124            **  01/13/83       NZ         Fixed bug in hRNMsb (setup for
2125            **                              FINDFx was incorrect)
2126            **                            Changed very first part of hRENAM
2127            **  01/12/83       NZ         Updated documentation
2128            **
2129            ************************************************************
2130            ************************************************************
2131 F5D6E 721A =hRENAM GOSUB  CKBITL
2132 F5D72 500         RTNNC                 Not HPIL HP82161...returnCC, XM=1
2133            *
2134            * Source or destination is HPIL (D[A] is address)
2135            *
2136            * A[W] is first 8 chars of source name, R0[3:0] is last 2 char
2137            * D[X] is HPIL address, D[S] is "8"
2138            *
2139 F5D75 7470         GOSUB  hRNMsd
2140 F5D79 70FE         GOSUB  Findf+         Find the destination file
2141            *
2142            * If found, error (File exists already)
2143            *
2144 F5D7D 5C1          GONC   hRNMfx         Error...file exists already
2145            *
2146            * Check if error is "file not found" or something else
2147            *
2148 F5D80 880          ?PW    =eTAPE         Is it tape error?
2149 F5D83 E1           GOYES  hRNMER         No..."real" error
2150 F5D85 80D0         P=C    0
2151 F5D89 890          ?P=    =eNFILE        Is it "No file" (Not found)?
2152 F5D8C D1           GOYES  hRNM30
2153 F5D8E 501          GONC   hRNMeT         Go always - tape error
```

```
2154                *-
2155                *-
2156 F5D91 0           CON(1) =FIXSPC      9 nibbles available here
2157 F5D92             BSS    9-1
2158                *-
2159                *-
2160 F5D9A 20   hRNMfx  P=     0
2161 F5D9C 300         LC(1)  =eEFILE      File already exists
2162 F5D9F 20   hRNMeT  P=     =eTAPE
2163 F5DA1 651F hRNMER  GOTO   Error        Set up error code, RTNSC
2164                *-
2165                *-
2166 F5DA5 67F9 hRNMXM  GOTO   hCPYxm       Carry clear, XM=1
2167                *-
2168                *-
2169                *
2170                * Destination file not found...continue
2171                *
2172 F5DA9 843  hRNM30  ST=0   =sDEST
2173 F5DAC 7040        GOSUB  hRNMsb
2174 F5DB0 120         AROEX                Put first 8 chars in R0...
2175 F5DB3 101         R1=A                 ...and last 2 chars in R1
2176 F5DB6 8E00        GOSUBL =FINDFx       Find the source file
           00
2177 F5DBC 44E         GOC    hRNMER        Error
2178                *
2179                * Now the B[3:0] is the directory pointer for the file
2180                *
2181 F5DBF 72EE        GOSUB  SAVDIR        Save directory info, get file type
2182                *                       (Ignore carry from FTYPF#)
2183                *
2184                * Now get the destination name back
2185                *
2186 F5DC3 7620         GOSUB  hRNMsd        Get back the destination info
2187                *
2188                * Now A[W] is the first 8 chars, R0 is the last 2 chars
2189                *
2190 F5DC7 8E00        GOSUBL =D1=SCR       Point D1 @ SCRTCH
           00
2191 F5DCD 1517        DAT1=A W             Write out first 8 chars of name
2192 F5DD1 17F         D1=D1+ 16            Position to last 2 chars location
2193 F5DD4 110         A=R0
2194 F5DD7 1593        DAT1=A 4             Write out last 2 chars of name
2195 F5DDB 722F        GOSUB  hPUTDR        Write directory entry from SCRTCH
2196 F5DDF 41C         GOC    hRNMER        Error
2197 F5DE2 724E        GOSUB  Endtap        End the tape conversation
2198 F5DE6 4AB         GOC    hRNMER        Error
2199 F5DE9 6599        GOTO   RtnXM0        Return, indicate "handled"
2200                *-
2201                *-
2202 F5DED 853  hRNMsd  ST=1   =sDEST        Set destination first
2203 F5DF0 DB   hRNMsb  C=D    A
2204 F5DF2 109         R1=C                 Save address in R1[A]
2205 F5DF5 706E        GOSUB  Rdinfo
2206 F5DF9 AFB         C=D    W             Save dest device & address in R1
```

```
2207 F5DFC 129          CR1EX               Restore old address, save new
2208 F5DFF D7           D=C    A            Restore address to D[A]
2209 F5E01 03           RTNCC               Carry clear
2210              ****************************************************************
2211              ****************************************************************
2212              **
2213              ** Name:       hFPROT - File protection handler (HPIL files)
2214              **
2215              ** Category:   POLL
2216              **
2217              ** Purpose:
2218              **      Execute the SECURE/PRIVATE command for an HPIL device
2219              **
2220              ** Entry:
2221              **      D[S] is the device type: if HPIL, then A[W] is first
2222              **      8 chars of filename, R0[3:0] is last 2 chars, D[X] is
2223              **      HPIL address of the device
2224              **      Destination info on SAVSTK (under POLLSV)
2225              **      (See detail also!)
2226              **
2227              ** Exit:
2228              **      Carry set: Error (C[3:0] is error number)
2229              **      Carry clear:
2230              **        XM=1: Not handled (not HPIL/not HP82161)
2231              **        XM=0: Handled (action taken)
2232              **
2233              ** Calls:      CKBITL,FINDF+,SAVDIR,CHKSEC,D1=S20,PT2BYT,
2234              **             hPUTDR,ENDTAP
2235              **
2236              ** Uses.......
2237              **  Inclusive: A-D,R0,R1,R3,D0,D1,P,ST[8,5:0],SCRTCH
2238              **
2239              ** Stk lvls:   6 (FINDF+)
2240              **
2241              ** Detail:
2242              **      ST(sPRIVT) set if PRIVATE, clear if SECURE
2243              **      ST(sUNSEC) set if UNSECURE, clear if SECURE
2244              **
2245              ** History:
2246              **
2247              **    Date     Programmer          Modification
2248              **  --------   ----------   ------------------------------
2249              **  06/02/83      NZ        Reworked to share much code with
2250              **                          PURGE and RENAME
2251              **  02/08/83      NZ        Changed to prevent PRIVATE on a
2252              **                            secure file (design change)
2253              **  01/12/83      NZ        Converted to single poll entry
2254              **  12/20/82      NZ        Added routine and documentation
2255              **
2256              ****************************************************************
2257              ****************************************************************
2258 F5E03 7D79 =hFPROT GOSUB  CKBITL        Check if this is HPIL & HP82161
2259 F5E07 500          RTNNC               No...set XM (not handled)
2260              *
2261              * This is an HPIL device
```

```
2262            *
2263 F5E0A 7F5E         GOSUB  Findf+        Save A in R0, R0>R1, START,FINDFx
2264 F5E0E 4E7          GOC    hSECer        Error
2265            *
2266            * Have found the file (D1 is at file type)
2267            *
2268 F5E11 709E         GOSUB  SAVDIR        Save dir info in R3, check type
2269 F5E15 460          GOC    hSEC15        Found type entry...continue
2270 F5E18 66D9 hSECft  GOTO   hCPYtp        Not found...error
2271            *-
2272            *-
2273            *
2274            * Found it...C[A], B[A] point to the entry, B[S] is position
2275            * of the type within the entry
2276            *
2277 F5E1C 7BCE hSEC15  GOSUB  CHKSEC        Check if secure(leaves P=entry #)
2278 F5E20 0B           CSTEX
2279 F5E22 80F0         CPEX   0
2280 F5E26 0B           CSTEX                Now ST[3:0] is the current pos
2281            sSEC    EQU    0             Bit for SECURE
2282            sPR     EQU    1             Bit for PRIVATE
2283 F5E28 860          ?ST=0  =sPRIVT       Is this PRIVATE statement?
2284 F5E2B E0           GOYES  hSEC20        No...must be secure
2285            *
2286            * PRIVATE statement
2287            *
2288 F5E2D 851          ST=1   sPR           Make it private!
2289 F5E30 860          ?ST=0  sSEC          Is it OK (NOT secure)?
2290 F5E33 41           GOYES  hSEC30        Yes...write it back out
2291 F5E35 6CAE         GOTO   hPURSC        No...file secure
2292            *-
2293            *-
2294 F5E39      hSEC20
2295            *
2296            * [UN]SECURE statement (need to determine which it is)
2297            *
2298 F5E39 860          ?ST=0  =sUNSEC       UNSECURE?
2299 F5E3C 80           GOYES  hSEC25        No...must be SECURE statement
2300            *
2301            * This is the UNSECURE statement
2302            *
2303 F5E3E 840          ST=0   sSEC          Clear the security bit
2304 F5E41 550          GONC   hSEC30        Go always
2305            *-
2306            *-
2307 F5E44      hSEC25
2308            *
2309            * This is the SECURE statement
2310            *
2311 F5E44 850          ST=1   sSEC
2312 F5E47      hSEC30
2313            *
2314            * Now ST[3:0] is the desired entry #
2315            *
2316 F5E47 0B           CSTEX
```

```
2317 F5E49 80F0          CPEX   0            Restore ST[3:0] from P
2318 F5E4D 0B            CSTEX
2319 F5E4F 80CF          C=P    15           Set C[S] to desired security
2320              *
2321              * Now C[S] is the desired type #, C[A] is the entry address
2322              *
2323 F5E53 135           D1=C
2324 F5E56 17E           D1=D1+ 15           Point to # types
2325 F5E59 1534          A=DAT1 S            Read it in...
2326 F5E5D 9CA           ?A<=C  S            ...is the type I want available?
2327 F5E60 8B            GOYES  hSECft       No...file type error
2328 F5E62 1C4           D1=D1- 5            Position to (type-2)
2329 F5E65 173  hSEC40   D1=D1+ 4            Go to next type
2330 F5E68 A4E           C=C-1  S            Done yet?
2331 F5E6B 59F           GONC   hSEC40       No...loop back
2332              *
2333              * Now D1 is at the desired file type
2334              *
2335 F5E6E 15F5          C=DAT1 6            Read type into C[5:2]
2336 F5E72 7D1E          GOSUB  D1=S20       Point to the type
2337 F5E76 8E00          GOSUBL =PT2BYT      Write the new file type
           00
2338              *
2339              * Now get the pointer back from R3 and write the entry
2340              *
2341 F5E7C 718E          GOSUB  hPUTDR       Write the entry from SCRTCH
2342 F5E80 4C0           GOC    hSECer       Error
2343 F5E83 71AD          GOSUB  Endtap       Clean up the loop
2344 F5E87 821           XM=0                Make sure XM=0 (handled)
2345 F5E8A 500           RTNNC               Return if no carry...done
2346              *
2347              * If fall through RTNNC, then error has occurred during ENDTAP
2348              *
2349 F5E8D 692E hSECer   GOTO   Error         Return, carry set
2350 F5E91               END
```

```
1            *
2            *
3            *       N   N  ZZZZZ   &      SSS   Y   Y  M   M
4            *       N   N      Z  & &    S   S  Y   Y  MM MM
5            *       NN  N     Z   & &    S       Y Y   M M M
6            *       N N N    Z     &      SSS     Y    M M M
7            *       N  NN   Z     & & &      S    Y    M   M
8            *       N   N  Z      &  &   S   S    Y    M   M
9            *       N   N  ZZZZZ   && &   SSS     Y    M   M
10           *
11                   TITLE  Symbolic Assignments <840301.1402>
12           *
13           * Status bit for ATTN key pressed (or other exception cause)
14           *
15           =Attn   EQU    12
16           *
17           * Other status bits
18           *
19           =sPRIVT EQU    11            Status for PRIVATE/SECURE stmt
20           =sUNSEC EQU    10            Status for [UN]Secure statement
21           =sOVERW EQU    8             Status for overwrite existing file
22           =sDevOK EQU    8             Status for device spec exec OK
23           =sSTK   EQU    7             Status for reading from stack
24           =CkTape EQU    5             Status to check for tape device
25           =sLoop? EQU    5             Status for allowing LOOP spec
26           =sReadd EQU    4             Status to force readdress the loop
27           =sFirst EQU    0             Status for first char in filespec
28           *
29           * Status bit corresponding to the bit I/O CPU sets if SREQ?
30           *
31           =sMBXsr EQU    1
32           *
33           * See NZ&PAR for parse status bits
34           *
35           *-------------------------------------------------------------
36           *
37           * Equates for P=, DDL/DDT
38           *
39           * DDL's
40           *
41           =Write0 EQU    0             Write to buffer 0
42           =Write1 EQU    1             Write to buffer 1
43           =Write  EQU    2             Write to tape
44           =SetBP  EQU    3             Set byte pointer
45           =Seek   EQU    4             Seek a record
46           =Format EQU    5             Format the medium
47           =PWrite EQU    6             Partial write mode
48           =Rewind EQU    7             Rewind
49           =CloseR EQU    8             Close record
50           =Xfr01L EQU    9             Transfer buffer 0-->1 (Listener)
51           =XchgL  EQU    10            Exchange buffers 0,1 (Listener)
52           =Verify EQU    11            Verify the medium
53           *
54           * DDT's
55           *
```

```
221             =mSDI   EQU    #A00000       Start Device Id
222             =mSAI   EQU    #B00000       Start Accessory Id
223             =mTCT   EQU    #C00000       Transfer ConTrol
224             =mTCT@4 EQU    (mTCT)/#10000 Transfer ConTrol (P=4)
225             =mSETTO EQU    #D00000       SET TimeOut
226             =mSTO@5 EQU    (mSETTO)/#100000 Set TimeOut (P=5)
227             =mSETFC EQU    #E00000       SET Frame Count
228             =mSFC@5 EQU    (mSETFC)/#100000 Set Frame Count @ nibble 5
229             *
230             * One-byte parameter class
231             *
232             =mSETDR EQU    #F30000       SET Device response
233             =mSETA1 EQU    #F30120       SET Accessory ID length (=1)
234             =mSETAI EQU    #F30321       SET Accessory ID value (=3)
235             =mSETS1 EQU    #F30140       SET Status length (=1)
236             =mSETST EQU    #F30041       SET Status value
237             =mSTS@4 EQU    #F3           SET Status value (at nibble 4)
238             =mSETD1 EQU    #F30610       SET Device ID length (=6)
239             =mSETDI EQU    #F30011       SET Device ID value (first byte)
240             =vDEVID EQU    \17PH\        Value of device ID (=HP71)
241             *
242             =mSETTM EQU    #F400         SET Terminator Mode
243             =mSETTC EQU    #F500         SET Terminator Character
244             =mSETIC EQU    #F600         SET # of IDY Timeouts
245             =mSETIT EQU    #F700         SET IDY Timeout (in mS)
246             =mCLRBF EQU    #F8           Clear data buffers (input&output)
247             =mSPTO  EQU    #F900         Set Serial Poll TimeOut
248             =mSETIM EQU    #FA00         Set interrupt mask
249             =mREADI EQU    #FB           Read interrupt cause
250             =mREADC EQU    #FC           Read last device dependent command
251             =CLRTSR EQU    #FD00         ...CLEAR terminate on SRQ mode
252             =SETTSR EQU    (CLRTSR)+1    ...SET terminate on SRQ mode
253             =mPULOP EQU    #FE           Power up the loop
254             =mSPDIS EQU    #FF00         Disable IDY serial poll
255             =mSPEN  EQU    (mSPDIS)+1    Enable IDY serial poll
256             *
257             * Non-parameter messages
258             *
259             =mNOP   EQU    #00           NO oPeration (check for HS)
260             =mRDADR EQU    #01           ReaD ADdRess table
261             =mSTATS EQU    #02           STATuS request to I/O CPU
262             =mSTSTC EQU    #0201         Request status, clear service reques
263             =mENDM  EQU    #03           END of Message
264             =mCSRQ  EQU    #04           Clear SRQ on loop
265             =mSSRQ  EQU    #05           Set SRQ on loop
266             =mERSTS EQU    #06           Request ERror STatuS
267             =mAUTOE EQU    #07           Enter AUTO End mode
268             =mMANUL EQU    #08           Go to manual mode
269             =mSCOPE EQU    #0801         Go into MANUAL mode, retransmit
270             =mAUTO  EQU    #09           Go to auto mode
271             =mUPDSC EQU    #0A00         Update System Controller bit(8/0)
272             =mRSTCA EQU    #0B           Reset current address
273             =mGETCA EQU    #0C           Read current address
274             =mINCCA EQU    #0D           Increment current address
275             =mMADDR EQU    #0E           Return "MY" address
```

```
276           =mCLRCA EQU    #0F00         Clear controller status
277           =mSETCA EQU    #0F01         (Set controller active)
278           =mTAKEC EQU    #0F03         Take control of the loop
279           *                            (2/0: if 2, then use IFC)
280           =mTAKEI EQU    (mTAKEC)~#90  Take control and send IFC frame
281           =mTAKEO EQU    (mTAKEC)~#10  Take control and send NOP frame
282           *
283           * Diagnostic class
284           *
285           =mRdMem EQU    #F00000       Read memory (add addr, RAM page)
286           =mWrMem EQU    #F10000       Write memory (add value~address)
287           =mTEST  EQU    #F2           I/O CPU self-test
288           *
289           *-------------------------------------------------------------
290           *
291           * RAM usage...
292           *
293           =SngDev EQU    4             Single device I/O buffer
294           *
295           * IS-xxx:
296           *       nib:   usage:
297           *       ---    ------
298           *       2-0:   If device address known, address, loop # here
299           *              If not known/assigned/iobuffer, FFF
300           *              If assigned, not HPIL, Fxx, xx<>FF
301           *
302           *         3:   If unassigned/not HPIL, F
303           *              If IO buffer for device ID/volume label, 4
304           *              If type specified, loop # + 1 (nib 3: 1,2,3)
305           *              If address specified, 0
306           *              If this assignment has been "OFF"ed, bit 3 is 1
307           *
308           *       6-4:   If type, nib 6: sequence #, nibs 5-4: Acc id
309           *              If address, 6-4: address, loop #
310           *              If IO buffer, 6-4: io buffer #
311           *              If unassigned (NOT "OFF"ed), FFF
312           *              If not HPIL and nib 3=F, not defined
313           *
314           *-------------------------------------------------------------
315           *
316           * Nibble "DSPSET"
317           *
318           =DispOK EQU    11            Display device is set up
319           =H82163 EQU    10            Display device is an HP82163A
320           =Printr EQU    9             Display device is a printer
321           =LoopOK EQU    8             Loop has not died while in disp
322           *
323           *-------------------------------------------------------------
324           *
325           * Nibble "LOOPST" (bits 8 and 9 are cleared when START is called)
326           *
327           =Offed  EQU    11            If set, USER specified OFF IO
328           =Device EQU    10            Last START found device mode
329           *
330           *-------------------------------------------------------------
```

```
331               *
332               * MBOX^: (3 nibbles)
333               *       Middle 3 digits of address of last mailbox used (ie if
334               *         mailbox was at address #20010 then MBOX^ is #001)
335               *
336               *-------------------------------------------------------------
337 00000                 END
```