```
;Ø4.15.8Ø 19:38 RJC: Changed CO to take character in a-reg.
;                    Changed repeat to take char in A-register
;                       and count in C-register.
;                    Expanded comments in beginning.
;                    Rearranged comments so that assembled 1.lst
;                       would fit in 8Ø columns.
;Ø4.11.8Ø 18:16 RJC: Added comment-tie RS232 DSR input to RS232
;                       receive data input.
;Ø4.Ø5.8Ø ØØ:Ø1 RJC: Recalculate cycles not counted by 8253 for
;                       baud rate calculation.
;Ø4.Ø4.8Ø 17:25 RJC: All functions tested.  Monitor "done".
;*****************************************************************
;*                                                               *
;*              Copyright 198Ø Raymond J. Clark                  *
;*                                                               *
;*      Permission is granted to reproduce this  monitor         *
;*      in  whole or in part, by any means, for any pur-         *
;*      pose other than sale, barter, or profit  of  any         *
;*      kind.                                                    *
;*                                                               *
;*      In return it is requested  that  this  copyright         *
;*      notice  be  included  with the portion copied in         *
;*      recognition of the time and effort  invested  in         *
;*      this software.                                           *
;*                                                               *
;*****************************************************************
;
;       ****            sccs-85 <= 1Ø24 byte monitor            ****
;                                                                   ****
;       **** Start up message:                                      ****
;       ****                                                        ****
;       ****            Mxxx.v                                      ****
;       ****            xxx     = last used address                 ****
;       ****            v       = version number which              ****
;       ****                      is incremented every              ****
;       ****                      time code is changed.             ****
;
;       **** Direct comments, suggestions, notes about              ****
;       **** bugs, etc. to:                                         ****
;       ****                                                        ****
;       ****    Ray Clark                                           ****
;       ****    School of Electrical Engineering                    ****
;       ****    Purdue University                                   ****
;       ****    West Lafayette, IN   479Ø7                          ****
;       ****                                                        ****
;       ****    UNIX login "clark"                                  ****
;       ****
;
;
;   On power up type a control-d.  8253 divisor calculated
;        automatically.
;
;   With 4 MHz crystal:
;              Works from 15Ø to 96ØØ baud.
;
;   With 3.579 MHz Color Burst crystal:
;              Works from 11Ø to 96ØØ baud.  Communicates at
;              192ØØ baud but tape load function will not work.
;
;
;   *** Commands:
;
;       L       load intel hex tape. Bias not implemented.
;               "answer" = G on exit if no errors detected.
;               If errors were found, answer = B
;       E       edit memory in hex.  See comments in code.
;       G       goto address.  Stack pointer reset.
;       D       dump memory.
;       P       punch intel hex tape.
;       (cr)    nop.  Does not change one byte "answer"
;       ?       print one byte message (answer) left by last
;               command.  This is cleared before the execution
;               of each command.
;
;       The "answer" is cleared before each command is run,
;       except ? and (cr) do not clear it.
;
;       The L, G, D, and P commands call an "ok" subroutine to
;       give you a last chance to abort the command.  A (cr) will
```

```
                        ;       go ahead and execute the command, anything else aborts.
                        ;
          Ø             BASE    equ     Ø       ;base address of monitor
       13ff             MEMTOP  equ     13ffh   ;top of memory
          f             WIDTH   equ     Øfh     ;controls the width of "dump" "punch"
                        ;                       ;commands:
                        ;                       ;           Øfh = 16 bytes, 53 columns
                        ;                       ;           Ø7h = 8 bytes, 29 columns

       1ØØ8             trap    equ     1ØØ8h  ;vectors for hardware interrupt lines
       1Ø1Ø             rst55   equ     1Ø1Øh
       1Ø18             rst65   equ     1Ø18h
       1Ø2Ø             rst75   equ     1Ø2Øh

         21             time1   equ     21h
         23             timctl  equ     23h

          1             sercon  equ     Ø1h
          Ø             serdat  equ     ØØh

          Ø             CRDLY   equ     ØØ      ;carriage return delay in 1ØmS units

                        ;===============================================================
                        ;
                        ; Reset, power-up restart, and RST Ø entry point

          Ø                     org     BASE+Ø

1Ø8       Ø 31 ff13             lxi     sp,STACKINIT    ; initialize stackpointer
1Ø9       3  Ø                  nop
11Ø       4 c3 6aØØ            jmp     entry

                        ;===============================================================
                        ;
                        ; RST 1 entry point

          8                     org     BASE+Ø8h                ; rst 1
                                db      Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
117       8  Ø          Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø

                        ;===============================================================
                        ;
                        ; RST 2 entry point

         1Ø                     org     BASE+1Øh                ; rst 2
                                db      Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
124      1Ø  Ø          Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø

                        ;===============================================================
                        ;
                        ; RST 3 entry point

         18                     org     BASE+18h                ; rst 3
                                db      Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
131      18  Ø          Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø

                        ;===============================================================
                        ;
                        ; RST 4 entry point

         2Ø                     org     BASE+2Øh                ; rst 4
                                db      Ø,Ø,Ø,Ø
138      2Ø  Ø          Ø  Ø  Ø  Ø

                        ;===============================================================
                        ;
                        ; TRAP entry point

         24                     org     BASE+24h                ; trap
145      24 c3  81Ø             jmp     trap
146      27  Ø                  nop

                        ;===============================================================
                        ;
                        ; RST 5 entry point

         28                     org     BASE+28h                ; rst 5
                                db      Ø,Ø,Ø,Ø
153      28  Ø          Ø  Ø  Ø  Ø
```

```
                        ;=================================================================
                        ;
                        ; RST 5.5 entry point

      2c                        org     BASE+2ch                ; rst 5.5
160   2c c3 1010                jmp     rst55
161   2f  0                     nop

                        ;=================================================================
                        ;
                        ; RST 6 entry point

      30                        org     BASE+30h                ; rst 6
                                db      0,0,0,0
168   30  0              0  0  0

                        ;=================================================================
                        ;
                        ; RST 6.5 entry point

      34                        org     BASE+34h                ; rst 6.5
175   34 c3 1810                jmp     rst65
176   37  0                     nop

                        ;=================================================================
                        ;
                        ; RST 7 entry point

      38                        org     BASE+38h                ; rst 7
                                db      0,0,0,0
183   38  0              0  0  0

                        ;=================================================================
                        ;
                        ; RST 7.5 entry point

      3c                        org     BASE+3ch
190   3c c3 2010                jmp     rst75
191   3f  0                     nop

                        ;=================================================================
                        ;=================================================================

      40                        org     BASE+40h

                        ; Jump table for monitor subroutines
199   40 c3 8a02                jmp     CI      ;char returned in A register
200   43 c3 9402                jmp     CICO    ;char returned in A register
201   46 c3 9702                jmp     CO      ;char passed in A register
202   49 c3 b302                jmp     crlf    ;prints (cr) (lf)
203   4c c3 ca02                jmp     ghw     ;word ret in h&l or cy=1 & bad char in A
204   4f c3 e102                jmp     ghb     ;byte ret in A or CY=1 & bad char in A
205   52 c3 f602                jmp     ghd     ;digit ret in A or CY=1 & bad char in A
206   55 c3 1003                jmp     msg     ;address of 0ffh terminated msg in d&e
207   58 c3 3f03                jmp     phw     ;word passed in h&l
208   5b c3 4a03                jmp     phb     ;byte passed in A
209   5e c3 5a03                jmp     phd     ;digit passed in A
210   61 c3 7d03                jmp     space   ;print space
211   64 c3 8503                jmp     sub16   ;(h&l) <- (h&l) - (d&e)
212   67 c3 9103                jmp     ucase   ;upper to lower case conversion
                        ;       jmp     cmp16   ;uncomment when cmp16 routine included

                        ; Power-up and Reset initialization
                        ;
                        ;Twiddle, twiddle little thumbs. . . .
                        ;
      6a                entry:  equ     $               ;
                        ;
                        ; hardware              mvi     a,10            ;
                        ; delay is       twiddle:call    d10ms           ;delay 10 mS
                        ; long                  dcr     a               ;
                        ; enough                jnz     twiddle         ;
                        ;
                        ; now initialize usart chip
                        ;
228   6a 3e 82                  mvi     a,082h  ;force usart to expect command word
229   6c d3  1                  out     sercon
230   6e 3e 40                  mvi     a,040h  ;now make usart to expect mode word
```

```
231   7Ø d3  1          out     sercon
                  ;
233   72 3e ce          mvi     a,Øceh   ;mode byte -
234   74 d3  1          out     sercon   ;  11 ØØ 11 1Ø
                  ;                      ;   |  |  | ------- X16 clock
                  ;                      ;   |  |  --------- 8 bits of data
                  ;                      ;   |  ------------ no parity
                  ;                      ;   --------------- 2 stop bits
                  ;
24Ø   76 3e 37          mvi     a,Ø37h   ;command byte -
241   78 d3  1          out     sercon   ;  Ø Ø 1 1 Ø 1 1 1
                  ;                      ;  | | | | | | | -- xmit enable
                  ;                      ;  | | | | | | ---- dtr/ = 1
                  ;                      ;  | | | | | ------ rcvr enable
                  ;                      ;  | | | | -------- norm op, (not break)
                  ;                      ;  | | | ---------- reset error flags
                  ;                      ;  | | ------------ rts/ = 1
                  ;                      ;  | -------------- 1 = internal reset
                  ;                      ;  ---------------- asyncronous mode
                  ;
                  ; Calculate baud rate assuming user types a control-d
                  ;
                  ; ***** DSR/ must be connected to RxD. These could be connected *
                  ;       on the RS232 side of the 1489 RS232 receiver.  On the   *
                  ;       SCCS-85 be sure to cut the trace on the bottom of the   *
                  ;       board connecting the RS232 dataset ready input to the   *
                  ;       +12v power supply.                                      *
                  ;
259   7a db  1    baud1:  in      sercon           ;wait for line to drop
26Ø   7c  7               rlc                      ;
261   7d d2 7aØØ          jnc     baud1            ;
                  ;
263   8Ø 3e 7Ø            mvi     a,Ø7Øh           ;set up timer to time next 4 bits
264   82 d3 23            out     timctl           ;
265   84 3e dd            mvi     a,-35    ; 8Ø85  ;34 cycles not counted by timer
                  ;       mvi     a,-41    ; 8Ø8Ø  ;4Ø cycles not counted by timer
267   86 d3 21            out     time1            ;  plus one for +1 after 1's comp
268   88 3e ff            mvi     a,Øffh           ;
269   8a d3 21            out     time1            ;
                  ;
                  ;       at baud1:    ;avg time out of loop after drop.
                  ;                    ;  loop 24 cycles long.............-   12
                  ;                    ;rlc...............................-    4
                  ;                    ;jnc    baud1...with cy=Ø 8Ø85/8Ø8Ø - 7/1Ø
                  ;                    ;mvi    a,Ø7Øh.....................-    7
                  ;                    ;out    timctl.....................-   1Ø
                  ;                    ;mvi    a,-35......................-    7
                  ;                    ;out    time1......................-   1Ø
                  ;                    ;mvi    a,Øffh.....................-    7
                  ;                    ;out    time1......................-   1Ø
                  ;
                  ;       at baud4:    ;avg time into loop since rose.
                  ;                    ;  loop 24 cycles long.............+   12
                  ;                    ;rlc...............................+    4
                  ;                    ;jnc    baud4   with cy=Ø 8Ø85/8Ø8Ø + 7/1Ø
                  ;                    ;mvi    a,4Øh......................+    7
                  ;                    ;out    timctl.....................+   1Ø
                  ;                    ;                                    ------
                  ;                    ;total cycles not counted by timer..-34/4Ø
                  ;                    ;                                   8Ø85/8Ø
                  ;
292   8c db  1    baud3:  in      sercon           ;wait for line to rise
293   8e  7               rlc                      ;
294   8f da 8cØØ          jc      baud3            ;
295   92 db  1    baud4:  in      sercon           ;wait for line to drop
296   94  7               rlc                      ;
297   95 d2 92ØØ          jnc     baud4            ;
                  ;
299   98 3e 4Ø            mvi     a,4Øh            ;counter latching command
3ØØ   9a d3 23            out     timctl           ;
3Ø1   9c 3e 7Ø            mvi     a,7Øh            ;set up timer to read lsb,msb
3Ø2   9e d3 23            out     timctl           ;
3Ø3   aØ db 21            in      time1            ;get lsb of count
3Ø4   a2 6f               mov     l,a              ;
3Ø5   a3 db 21            in      time1            ;get msb of count
3Ø6   a5 67               mov     h,a              ;
3Ø7   a6 7d               mov     a,l              ;compliment count - don't need to
3Ø8   a7 2f               cma                      ;  add one because taken into
3Ø9   a8 6f               mov     l,a              ;  account in initial load of
```

```
310   a9 7c              mov     a,h             ;  counter.
311   aa 2f              cma                     ;
312   ab 67              mov     h,a             ;
                  ;
314   ac  e  6           mvi     c,6             ;shift count right 6
315   ae b7      baud5:  ora     a               ;cy=Ø to come in left end of H
316   af 7c              mov     a,h             ;
317   bØ 1f              rar                     ;
318   b1 67              mov     h,a             ;
319   b2 7d              mov     a,l             ;
32Ø   b3 1f              rar                     ;
321   b4 6f              mov     l,a             ;
322   b5  d              dcr     c               ;
323   b6 c2 ae ØØ        jnz     baud5           ;
324   b9 ce  Ø           aci     Ø               ;round
325   bb 6f              mov     l,a             ;
326   bc 7c              mov     a,h             ;propagate possible round-up
327   bd ce  Ø           aci     Ø               ;  carry into H.
328   bf 67              mov     h,a             ;
                  ;
                  ; initialize timer chip to generate 16X baudrate for
                  ;
332   cØ 3e 76           mvi     a,76h           ;init timer 1 to divide by n
333   c2 d3 23           out     timctl          ;
334   c4 7d              mov     a,l             ;
335   c5 d3 21           out     time1           ;
336   c7 7c              mov     a,h             ;
337   c8 d3 21           out     time1           ;
                  ;
339   ca 11 ee Ø3        lxi     d,start         ;print startup message
34Ø   cd cd 1Ø Ø3        call    msg
                  ;
342   dØ db  Ø           in      serdat          ;eat garbage character
                  ;
                  ; COMMAND LEVEL - get character; jump to appropriate routine

      d2           comnd:  equ     $

349   d2 11 e9 Ø3        lxi     d,prmpt ;print command prompt
35Ø   d5 cd 1Ø Ø3        call    msg
                  ;
352   d8 cd 94 Ø2        call    CICO    ;
353   db cd 7d Ø3        call    space   ;
                  ;       ani     7fh     ;put in if ucase taken out
355   de cd 91 Ø3        call    ucase   ;convert low to up case & strips parity
                  ;
357   e1 fe  d           cpi     Ødh     ;special case, (cr) is nop that does not
358   e3 ca d2 ØØ        jz      comnd   ;  clear the answer
                  ;
36Ø   e6 11 d2 ØØ        lxi     d,comnd ;addr for pseudo call completed by pchl
361   e9 d5              push    d       ;
                  ;
363   ea fe 3f           cpi     '?'     ;special case '?', must not clear
364   ec ca  fØ1         jz      ask     ; answer first.
                  ;
366   ef 21 bØ Ø3        lxi     h,cmds  ;scan command table
367   f2 be      cmdnxt: cmp     m       ;
368   f3 ca  2Ø1         jz      cmdfnd  ;if matches go process
369   f6 23              inx     h       ;
37Ø   f7 23              inx     h       ;
371   f8 23              inx     h       ;
372   f9 46              mov     b,m     ;check for end of table
373   fa  5              dcr     b       ;
374   fb f2 f2 ØØ        jp      cmdnxt  ;not end...try next entry
                  ;
376    fe cd 6d Ø3 error:  call    prbad   ;print error message and return.  "comnd"
377   1Ø1 c9              ret             ;  is on stack as return addr for command
                  ;
      1Ø2          cmdfnd: equ     $
38Ø   1Ø2 f5              push    psw
381   1Ø3 3e 2Ø           mvi     a,' '   ;clear answer
382   1Ø5 32 ff 13        sta     answer  ;
383   1Ø8 f1              pop     psw
384   1Ø9 23              inx     h       ;get address
385   1Øa 5e              mov     e,m     ;
386   1Øb 23              inx     h       ;
387   1Øc 56              mov     d,m     ;
388   1Ød eb              xchg            ;
```

```
389  10e e9              pchl            ;
                    ;
                    ;****************** end of command level ************************
                    ;*********************************************************************
                    ;
                    ;       print   one byte note left by last command
                    ;
397  10f cd 7d03 ask:    call    space
398  112 3a ff13         lda     answer
399  115 cd 9702         call    CO
400  118 c9              ret
                    ;
                    ;****************** end of 20 questions *************************
                    ;*********************************************************************
                    ; GOTO routine - starts execution in memory location
408  119 cd ca02 goto:   call    ghw     ;get hex word
409  11c da fe00         jc      error   ;
410  11f cd 2203         call    okck    ;
411  122 d8              rc              ;
                    ;
413  123 31 ff13         lxi     sp,STACKINIT    ;initialize stack pointer
414  126 e9              pchl            ;jmp to location, return addr is on stack
                    ;
416  127 c3 fe00         jmp     error
                    ;
                    ;*************** end of goto routine ****************************
                    ;
                    ;*********************************************************************
                    ; Memory editor routine
                    ;
                    ; MEMED - Hexadecimal Memory Editor
                    ;
                    ;       1) Computer types (cr),(lf),"("
                    ;       2) User enters one of the following:
                    ;          a) Valid Hex  Word  (four hex  digits)  -  goto
                    ;             step 3
                    ;          b) "/" - Exit editor by doing a "rst 0"
                    ;          c) Anything else - type " what ?" and
                    ;             goto 1
                    ;       3) Computer types ") = xx ", where xx is the con-
                    ;          tents (in hex) of the memory byte addressed.
                    ;       4) The  user  now has  a number  of things  he can
                    ;          type:
                    ;          a) Valid Hex Byte (two hex digits) -  overwrite
                    ;             the  memory  location  addressed  with  this
                    ;             byte.   Then read  another  character  from
                    ;             the user and continue with 4b.
                    ;
                    ;          b) A non-hex character - do one of the  follow-
                    ;             ing:
                    ;
                    ;             i)   (cr) or " " - Address the next  sequen-
                    ;                  tial location and print the address and
                    ;                  contents like this:
                    ;
                    ;                          (cr),(lf),"(addr) = xx "
                    ;
                    ;             ii)  '.' - Re-display the same location like
                    ;                  this:
                    ;
                    ;                          (cr),(lf),"(addr) = xx "
                    ;
                    ;             iii) "-" - Address the next  previous  loca-
                    ;                  tion and print the address and contents
                    ;                  like this:
                    ;
                    ;                          (cr),(lf),"(addr) = xx "
                    ;
                    ;             iv)  "/" - Goto step 1 and read  a  new  ad-
                    ;                  dress
                    ;
                    ;             v)     Anything   else   -   Type   "   what
                    ;                  ?" and treat like a "."
                    ;
```

```
                        ;
                        ;          Note - If option "a" is not executed  then  memory
                        ;          is not altered.
                        ;
472  12a 11 d303 memed:  lxi     d,form2 ;print "cr, lf, ("
473  12d cd 1003         call    msg

475  130 cd ca02         call    ghw
476  133 d2 3f01         jnc     ok      ;get hex word into HL, jump if valid

478  136 fe 2f           cpi     '/'     ;bad char received - was it "/"
479  138 c8              rz              ;go back to command level if so

481  139 cd 6d03         call    prbad   ;print "what ?"
482  13c c3 2a01         jmp     memed   ;then try again

484  13f cd 9001 ok:     call    discon  ;display contents of location
485  142 cd 4801         call    edit    ;then begin editing
486  145 c3 2a01         jmp     memed   ;loupe if edit returns
                        ;
                        ;       end     memed
                        ;
                        ;
                        ; Get either a new hex byte to be written where HL points,
                        ; followed by another command, or just another command.
                        ;
494  148 cd e102 edit:   call    ghb     ;get the new hex byte if typed
495  14b da 5701         jc      next    ;jmp if other than hex byte received
496  14e 77              mov     m,a     ;else store it in memory
497  14f cd 7d03         call    space   ;space to reinforce that once two digits
                        ;                       ;  are entered, location is changed.
499  152 cd 9402         call    CICO    ;and get another char& echo it
500  155 e6 7f           ani     7fh     ;kill top bit
                        ;
502  157 fe  d   next:   cpi     0dh     ;carriage return?
503  159 c2 6001         jnz     e1
504  15c 23              inx     h
505  15d c3 7d01         jmp     pr      ;yes- print NEXT location
                        ;
507  160 fe 20   e1:     cpi     ' '     ;or blank
508  162 c2 6901         jnz     e2
509  165 23              inx     h
510  166 c3 7d01         jmp     pr      ;yes- do the same
                        ;
512  169 fe 2e   e2:     cpi     '.'     ; period?
513  16b ca 7d01         jz      pr      ;print current location
                        ;
515  16e fe 2d   e3:     cpi     '-'     ; dash?
516  170 c2 7701         jnz     e4
517  173 2b              dcx     h
518  174 c3 7d01         jmp     pr      ;yes - print previous location
                        ;
520  177 fe 2f   e4:     cpi     '/'     ;slash?
521  179 c8              rz              ;edit all done if so
                        ;
523  17a cd 6d03         call    prbad   ;if none of the above, print "what ?"
                        ;
525  17d cd 8301 pr:     call    dismem  ;display the new current memory location
526  180 c3 4801         jmp     edit    ;and loupe

                        ; Print CR, LF then an ( followed by the contents of HL in hex.

530  183 11 d303 dismem: lxi     d,form2 ;do cr,lf, "("
531  186 cd 1003         call    msg
532  189 cd 3f03         call    phw
533  18c cd 9001         call    discon
534  18f c9              ret
                        ;
                        ; **** discon ****
                        ;
                        ; print ') = ' followed by the contents of the memory loc.
                        ; pointed to by HL
                        ;
541  190 11 ce03 discon: lxi     d,form
542  193 cd 1003         call    msg
543  196 7e              mov     a,m     ;get contents of mem loc.
544  197 cd 4a03         call    phb     ;print it
545  19a cd 7d03         call    space
546  19d c9              ret
```

```
                        ;
                        ;
                        ;********************** end of memory editor ********************

                        ;****************************************************************
                        ; Hex-format loader

554   19c cd b201 loader: call    getrec  ;get the next record and process it

                        ; at this point the A register contains the length of the record
                        ; just processed - see if the record length was zero. If so then
                        ; the loader has found the end of the file and is finished
                        ;
560   1a1 b7              ora     a       ;compare the A reg. to Ø
561   1a2 3e 47           mvi     a,'G'   ;answer to question = Good
562   1a4 ca ae01         jz      done    ;jump to "done" if equal to Ø

                        ; the record was not the last - see if any errors were detected
565   1a7 7a              mov     a,d     ;
566   1a8 b7              ora     a       ;see if the "error" flag is non-zero.
567   1a9 ca 9e01         jz      loader  ;if not, go do next record
                        ;
569   1ac 3e 42           mvi     a,'B'   ;store "Bad" flag in answer to question
570   1ae 32 ff13 done:   sta     answer
571   1b1 c9              ret

                        ;********************** end of main program **********************
                        ;
                        ; 1st level of subroutines

577   1b2 cd cc01 getrec: call    fndmrk  ;find the record mark

579   1b5 cd e601         call    lghb    ;get the record length into the C reg.
580   1b8 4f              mov     c,a

582   1b9 cd e601         call    lghb    ;put the load address field into the HL
583   1bc 67              mov     h,a     ;pair
584   1bd cd e601         call    lghb    ;
585   1cØ 6f              mov     l,a     ;

587   1c1 cd e601         call    lghb    ;get the record-type byte, don't do
                                          ;anything with it

590   1c4 cd d901         call    data    ;put the next N=(C) bytes into memory
                                          ;starting where HL points

593   1c7 cd e601         call    lghb    ;read the checksum byte

595   1ca 79              mov     a,c     ;put the record length back into A reg.
596   1cb c9              ret             ;return from subroutine


                        ;***************** end of 2nd level of subroutines ***************
                        ;
                        ; 3rd level of subroutines


605   1cc cd 8a02 fndmrk: call    CI      ;get character from CRT
606   1cf e6 7f           ani     Ø7fh    ;strip off 8th bit
607   1d1 fe 3a           cpi     ':'     ;
608   1d3 c2 cc01         jnz     fndmrk  ;
                        ;               ;
610   1d6 16  Ø           mvi     d,Ø     ; clear D register
611   1d8 c9              ret             ;
                        ;
613   1d9 41        data: mov     b,c     ;copy C reg. to B
614   1da 78        loop: mov     a,b     ;get remaining byte count
615   1db b7              ora     a       ;get flags
616   1dc c8              rz              ;return from subr. if none left
617   1dd  5              dcr     b       ;else decrement b reg.

619   1de cd e601         call    lghb    ;get byte from data field
620   1e1 77              mov     m,a     ;store in memory
621   1e2 23              inx     h       ;bump pointer
622   1e3 c3 da01         jmp     loop    ;go back for next char.


                        ;*************** end of 3rd level ********************************
```

```
                    ;
                    ; 4th level of subroutines
                    ;
629  1e6 cd e102 lghb:   call    ghb     ; loader ghb - adds byte gotten to D-reg
630  1e9 f5              push    psw
631  1ea 82              add     d
632  1eb 57              mov     d,a
633  1ec f1              pop     psw
634  1ed c9              ret
                    ;
                    ;************* end of 4th level of subroutines ******************
                    ;
                    ;************* end of loader ************************************
                    ;
                    ;
                    ;****************************************************************
                    ;
                    ;       Common code for dump and punch routine.
                    ;         Must not destroy a register
                    ;
646  1ee 11 e203 d_p:    lxi     d,plo   ;prompt for lo limit
647  1f1 cd 1003         call    msg
648  1f4 cd ca02         call    ghw
649  1f7 da fe00         jc      error   ;jump if error
650  1fa 11 dd03         lxi     d,phi   ;prompt for hi limit
651  1fd cd 1003         call    msg
652  200 eb              xchg
653  201 cd ca02         call    ghw
654  204 eb              xchg
655  205 da fe00         jc      error
656  208 cd 2203         call    okck
657  20b d8              rc              ;return if aborted by okck call
                    ;
                    ; (h&l) = beginning address    (d&e) = ending address
                    ;
661  20c e5              push    h       ;
662  20d cd 8503         call    sub16   ;calc number of bytes to be processed
663  210 eb              xchg            ;
664  211 e1              pop     h       ;
665  212 1b              dcx     d       ;d&e = number of bytes
                    ;
                    ; Call routine originally requested
                    ;
669  213 fe 50           cpi     'P'
670  215 ca 4702         jz      punch
671  218 fe 44           cpi     'D'
                    ;       else    dump
                    ;
                    ; Dump routine
                    ;
676  21a cd b302 dump:   call    crlf    ;go to new line
677  21d cd 3f03         call    phw     ;print memory address
678  220 7d              mov     a,l     ;make locations with same lower 4 bits
679  221 e6  f           ani     WIDTH   ;  land in same columns in first line
680  223 4f              mov     c,a     ;  as in other lines
681  224 87              add     a       ;multiply by 3...
682  225 81              add     c       ;
683  226 4f              mov     c,a     ;move count to c
684  227 3e 20           mvi     a,' '   ;space over to appropriate column
685  229 cd 7303         call    repeat  ;print (c) (a) times

687  22c cd 7d03 dil:    call    space
688  22f 7e              mov     a,m     ;get byte
689  230 cd 4a03         call    phb     ;print it
690  233 23              inx     h       ;point to next byte
691  234 13              inx     d       ;decrement count of number of bytes left
692  235 7b              mov     a,e     ;
693  236 b2              ora     d       ;
694  237 c8              rz              ;return if zero left
695  238 7d              mov     a,l
696  239 e6  f           ani     WIDTH   ;print crlf on multiple of 16
697  23b c2 2c02         jnz     dil

699  23e cd b302         call    crlf    ;go to new line
700  241 cd 3f03         call    phw     ;print memory address
701  244 c3 2c02         jmp     dil     ;

                    ;****************** end of dump *********************************
                    ;
```

```
                        ;*****************************************************************
                        ;
                        ; *** Punch Hex Tape in INTEL format  ***
                        ;
                        ;preliminary processing done at d_p
                        ;
711  247 cd b302 punch:  call    crlf            ;
712  24a 3e 3a           mvi     a,':'           ;print record mark
713  24c cd 9702         call    CO              ;
714  24f 7b              mov     a,e             ;calc number of bytes in record
715  250 2f              cma                     ; & start accumulating check sum
716  251 e6  f           ani     WIDTH           ;
717  253 3c              inr     a               ;
718  254 cd 4a03         call    phb             ;print number of bytes in record
719  257 84              add     h               ;add load address to check sum
720  258 85              add     l               ;
721  259 47              mov     b,a             ;initialize checksum
722  25a cd 3f03         call    phw             ;load address
723  25d af              xra     a               ;
724  25e cd 4a03         call    phb             ;record type
                        ;
726  261 7e      pnxtbyt:mov     a,m             ;
727  262 cd 4a03         call    phb             ;
728  265 80              add     b               ;accumulate checksum
729  266 47              mov     b,a             ;
730  267 13              inx     d               ;
731  268 7b              mov     a,e             ;
732  269 b2              ora     d               ;
733  26a ca 7d02         jz      pdone           ;
734  26d 23              inx     h               ;
735  26e 7b              mov     a,e             ;test for end of record
736  26f e6  f           ani     WIDTH           ;
737  271 c2 6102         jnz     pnxtbyt         ;
                        ;
739  274 78              mov     a,b             ;end of record processing
740  275 2f              cma                     ;compliment checksum
741  276 3c              inr     a               ;
742  277 cd 4a03         call    phb             ;
743  27a c3 4702         jmp     punch           ;
                        ;
745  27d 78      pdone:  mov     a,b             ;compliment last checksum
746  27e 2f              cma                     ;
747  27f 3c              inr     a               ;
748  280 cd 4a03         call    phb             ;
749  283 11 c003         lxi     d,endrec        ;
750  286 cd 1003         call    msg             ;
751  289 c9              ret                     ;
                        ;                       ;
                        ;       end     punch   ;
                        ;
                        ;*****************************************************************
                        ;
                        ;       UTILITY ROUTINES - in alphabetical order
                        ;
                        ;*****************************************************************
                        ;
                        ; I/O routines
764  28a db  1   CI:     in      sercon
765  28c e6  2           ani     2
766  28e ca 8a02         jz      CI
767  291 db  0           in      serdat
768  293 c9              ret


                        ;;****** cmp16 ** 16 bit compare h&l and d&e *********************
                        ;;
                        ;;       if( h&l = d&e ) z=1, cy=0      *** crafty and very   ***
                        ;;       if( h&l > d&e ) z=0, cy=0      *** useful routine if ***
                        ;;       if( h&l < d&e ) z=0, cy=1      *** ever room         ***
                        ;;
                        ;cmp16:  push    h               ;save psw & h&l
                        ;        push    psw             ;
                        ;        mov     a,h             ;if h != d enough info found
                        ;        sub     d               ;
                        ;        jnz     cmp16e          ;
                        ;        mov     a,l             ;if h=d then compare lower bytes
                        ;        sub     e               ;
```

```
                    ;cmp16e: pop     h                ;
                    ;        mov     a,h              ;
                    ;        pop     h                ;
                    ;        ret                      ;
                    ;;                                ;
                    ;;       end     cmd16            ;
                    ;
                    ;
                    ; CICO - input char then echo it
                    ;
                    ;       ******* NOTE ! ******** NO RETURN ! ********
                    ;       *                                           *
                    ;       *       MUST BE IMMEDIATELY FOLLOWED        *
                    ;       *                  BY CO                    *
                    ;       *                                           *
                    ;       *********************************************
                    ;
801  294 cd 8a02 CICO:   call    CI       ;
                    ;
                    ;**** CO Console Output - destroys only flags...
                    ;
                    ;                              ...char passed in a register
806  297 f5       CO:     push    psw
807  298 db  1    c1:     in      sercon
808  29a  f               rrc
809  29b d2 9802          jnc     c1
810  29e f1               pop     psw
811  29f d3  0            out     serdat
812  2a1 fe  d            cpi     0dh              ;if cr then delay
813  2a3 c8               rz                       ;
814  2a4 f5               push    psw
815  2a5 3e  1            mvi     a,CRDLY+1        ;
816  2a7 3d       c2:     dcr     a                ;
817  2a8 ca b102          jz      c3               ;
818  2ab cd bc02          call    d10ms            ;delay 10mS
819  2ae c3 a702          jmp     c2               ;
820  2b1 f1       c3:     pop     psw
821  2b2 c9               ret

823  2b3 d5       crlf:   push    d
824  2b4 11 a503          lxi     d,mcrlf
825  2b7 cd 1003          call    msg
826  2ba d1               pop     d
827  2bb c9               ret
                    ;
                    ; d10ms - Delay 10 mS
                    ;
831  2bc e5       d10ms:  push    h        ;
832  2bd f5               push    psw      ;
833  2be 21  103          lxi     h,769    ;
834  2c1 7d       dtwid1: mov     a,1      ;    ;~0.01 seconds on a 4 MHz 8085      5
835  2c2 b4               ora     h        ;    ;                                    4
836  2c3 2b               dcx     h        ;    ;                                   10
837  2c4 c2 c102          jnz     dtwid1   ;    ;                 8085/8080        7/10
838  2c7 f1               pop     psw      ;    ;                 total           26/29
839  2c8 e1               pop     h        ;
840  2c9 c9               ret              ;
                    ;                     ;
                    ;       end     d10ms    ;
                    ;
                    ; GHW - Get Hex Word
                    ;
                    ;       Read 4 hex digits frm terminal & convert to 16 bit word
                    ;
                    ;       INPUT  : None
                    ;       OUTPUT : if (no non-hex charaters typed)
                    ;
                    ;                       (h&l) = hex word typed
                    ;                       (a)   = garbage
                    ;                       CY    = 0
                    ;                 else
                    ;                       (h&l) = garbage
                    ;                       (a)   = bad character as received from CO
                    ;                       CY    = 1
                    ;
                    ;       REGISTERS CHANGED: h, l, flags
                    ;
861  2ca c5       ghw:    push    b
862  2cb f5               push    psw
```

```
863  2cc cd e1Ø2          call    ghb             ; get first byte in a-register
864  2cf da deØ2          jc      ghwend          ; return if bad char
865  2d2 67               mov     h,a             ; move byte to final destination
866  2d3 cd e1Ø2          call    ghb             ; get second byte
867  2d6 da deØ2          jc      ghwend          ;
868  2d9 6f               mov     l,a             ;
869  2da c1               pop     b               ;
87Ø  2db 78               mov     a,b             ;
871  2dc c1               pop     b               ;
872  2dd c9               ret                     ;
873  2de c1       ghwend: pop     b               ;
874  2df c1               pop     b               ; do NOT restore a
875  2eØ c9               ret                     ;
                  ;                              ;
                  ;       end     ghw             ;
                  ;
                  ;
                  ; GHB - Get Hex Byte
                  ;
                  ;       Read 2 hex digits from terminal & convert to 8 bit word
                  ;
                  ;       INPUT  : None
                  ;       OUTPUT : if (no non-hex charaters typed)
                  ;
                  ;                               (a)   = Hex byte typed
                  ;                               CY    = Ø
                  ;                 else
                  ;                               (a)   = bad character as received from CO
                  ;                               CY    = 1
                  ;
                  ;       REGISTERS CHANGED: a, flags
                  ;
895  2e1 c5       ghb:    push    b               ; save b&c
896  2e2 cd f6Ø2          call    ghd             ; get first hex digit in a-reg
897  2e5 da f4Ø2          jc      ghbend          ; if bad char quit and pass back
898  2e8  7               rlc                     ; shift to upper half of byte
899  2e9  7               rlc                     ;   .
9ØØ  2ea  7               rlc                     ;   .
9Ø1  2eb  7               rlc                     ;   .
9Ø2  2ec 47               mov     b,a             ; save first digit
9Ø3  2ed cd f6Ø2          call    ghd             ; get second digit
9Ø4  2fØ da f4Ø2          jc      ghbend          ; bad char read, ret it to caller
                  ;                              ;
9Ø6  2f3 bØ               ora     b               ; combine first and second digits
                  ;                              ;
9Ø8  2f4 c1       ghbend: pop     b               ; restore original b&c
9Ø9  2f5 c9               ret                     ;
                  ;                              ;
                  ;       end     ghb             ;
                  ;
                  ;
                  ; GHD - Get Hex Digit
                  ;
                  ;       Read 1 hex digit from terminal & convert to 4 bit nibble
                  ;
                  ;       INPUT  : None
                  ;       OUTPUT : if (valid hex character typed)
                  ;
                  ;                               (a)   = Øxh, x = hex digit typed
                  ;                               CY    = Ø
                  ;                 else
                  ;                               (a)   = bad character as received from CO
                  ;                               CY    = 1
                  ;
                  ;       REGISTERS CHANGED: a, c, flags
                  ;
929  2f6 cd 94Ø2 ghd:    call    CICO            ; get character & echo
                  ;       ani     Ø7fh            ; put in if ucase taken out
931  2f9 cd 91Ø3          call    ucase           ; map lower to upper case and
                  ;                              ;   strip parity.
933  2fc fe 3Ø            cpi     'Ø'             ;
934  2fe d8               rc                      ;       non-hex character
935  2ff fe 3a            cpi     ':'             ; if  (a) =< '9'+1
936  3Ø1 da  dØ3          jc      ghd2            ;       'Ø'-'9' typed - convert
937  3Ø4 fe 41            cpi     'A'             ; if  (a) < 'A'
938  3Ø6 d8               rc                      ;       non-hex character
939  3Ø7 fe 47            cpi     'G'             ; if  (a) >= 'G'
94Ø  3Ø9 3f               cmc                     ;       .
941  3Øa d8               rc                      ;       non-hex character
```

```
942  30b d6  7          sui     07h             ; shift 'A'-'F' down
943  30d d6 30   ghd2:  sui     '0'             ; convert
944  30f c9             ret                     ;
                ;                               ;
                ;       end     ghd             ;
                ;
                ; Subroutine to print message pointed to by DE and
                ; terminated by 0FFh byte.
                ;       destroys no registers

952  310 f5     msg:    push    psw
953  311 d5             push    d
954  312 1a     loupe:  ldax    d       ;get char
955  313 fe ff          cpi     0ffh    ;end of string?
956  315 ca 1f03        jz      mdn     ;jump if so
957  318 cd 9702        call    CO      ;else print it
958  31b 13             inx     d       ;bump pointer
959  31c c3 1203        jmp     loupe   ;do it again
960  31f d1     mdn:    pop     d
961  320 f1             pop     psw
962  321 c9             ret
                ;
                ;
                ; routine to verify an entry
                ;
967  322 d5     okck:   push    d
968  323 f5             push    psw
969  324 11 d703        lxi     d,mok
970  327 cd 1003        call    msg
971  32a cd 8a02        call    CI
972  32d e6 7f          ani     07fh
973  32f fe  d          cpi     0dh
974  331 ca 3b03        jz      okckend
975  334 11 9d03        lxi     d,abort
976  337 cd 1003        call    msg
977  33a 37             stc
978  33b d1     okckend:pop     d
979  33c 7a             mov     a,d
980  33d d1             pop     d
981  33e c9             ret
                ;
                ;       end     okck
                ;
                ;
                ; PHW - Print Hex Word
                ;
                ;       Convert 16 bit word to ascii and print
                ;
                ;       INPUT  : (h&l) = word to be printed
                ;       OUTPUT : None
                ;
                ;       REGISTERS CHANGED: None
                ;
995  33f f5     phw:    push    psw             ; save a-register and flags
996  340 7c             mov     a,h             ;
997  341 cd 4a03        call    phb             ; print high-order byte
998  344 7d             mov     a,l             ;
999  345 cd 4a03        call    phb             ; print low-order byte
1000 348 f1             pop     psw             ; restore a-register and flags
1001 349 c9             ret                     ;
                ;                               ;
                ;       end     phw             ;
                ;
                ;
                ; PHB - Print Hex Byte
                ;
                ;       Convert 8 bit byte to ascii and print
                ;
                ;       INPUT  : (a) = Byte to be printed
                ;       OUTPUT : None
                ;
                ;       REGISTERS CHANGED: Flags
                ;
1015 34a c5     phb:    push    b               ; save b&c
1016 34b 47             mov     b,a             ; save lower nibble
1017 34c  f             rrc                     ; shift to lower half of byte
1018 34d  f             rrc                     ;   .
1019 34e  f             rrc                     ;   .
1020 34f  f             rrc                     ;   .
```

```
1021  35Ø cd 5a03          call    phd             ; print upper hex digit
1022  353 78               mov     a,b             ; get lower nibble
1023  354 cd 5a03          call    phd             ; ...and print
1024  357 78               mov     a,b             ; restore original byte to a
1025  358 c1               pop     b               ; restore b&c
1026  359 c9               ret                     ;
                           ;                       ;
                           ;       end     phb     ;
                           ;
                           ;
                           ; PHD - Print Hex Digit
                           ;
                           ;       Convert hex digit to ascii and print it
                           ;
                           ;       INPUT  : (a) = ?xh where x is the hex digit to be printed
                           ;                      the ? nibble is immaterial
                           ;       OUTPUT : None
                           ;
                           ;       REGISTERS CHANGED: flags
                           ;
1041  35a c5         phd:    push    b               ;save a&c
1042  35b 47               mov     b,a             ;
1043  35c e6  f            ani     Øfh             ; mask off lower nibble
1044  35e c6 3Ø            adi     'Ø'             ; convert 'Ø'-'9' to ascii
1045  36Ø fe 3a            cpi     '9'+1           ; if 'Ø'-'9'
1046  362 da 67Ø3          jc      phd1            ;       then done
1047  365 c6  7            adi     'A'-':'         ; convert 'A'-'F'
1048  367 cd 97Ø2 phd1:    call    CO              ; print digit
1049  36a 78               mov     a,b             ;restore registers
105Ø  36b c1               pop     b               ;
1051  36c c9               ret                     ;
                           ;                       ;
                           ;       end     phd     ;
                           ;
                           ;
                           ; ***** prbad - print ' WHAT ?' **** DESTROYS D&E ****
                           ;
1058  36d 11 a8Ø3 prbad:  lxi     d,bad   ;
1059  37Ø c3 1Ø03          jmp     msg     ;I know, very bad code...
                           ;
                           ;       end     prbad
                           ;
                           ;
                           ;Subroutine to print (a) (c) times
                           ;       uses a, c...(c) = Ø on exit
1066  373  c        repeat: inr     c       ;check for printing (c) Ø times
1067  374  d              dcr     c       ;
1068  375 c8         rep1:   rz
1069  376 cd 97Ø2          call    CO
107Ø  379  d               dcr     c
1071  37a c3 75Ø3          jmp     rep1
                           ;
                           ; ***** space ***** print space
                           ;
1075  37d f5         space:  push    psw
1076  37e 3e 2Ø            mvi     a,' '
1077  38Ø cd 97Ø2          call    CO
1078  383 f1               pop     psw
1079  384 c9               ret
                           ;
                           ; ***** sub16 ***** 16 bit subtract (h&l) <- (h&l) - (d&e)
                           ;
                           ;       if    (d&e)  < (h&l)    CY = 1
                           ;       if    (d&e) >= (h&l)    CY = Ø
                           ;
1086  385 d5         sub16:  push    d       ;
1087  386 f5               push    psw     ;
1088  387 7d               mov     a,l     ;
1089  388 93               sub     e       ;
1090  389 6f               mov     l,a     ;
1091  38a 7c               mov     a,h     ;
1092  38b 9a               sbb     d       ;
1093  38c 67               mov     h,a     ;
1094  38d d1               pop     d       ;
1095  38e 7a               mov     a,d     ;
1096  38f d1               pop     d       ;
1097  39Ø c9               ret             ;
                           ;
                           ; UCASE - subroutine which checks the A reg for a lower case
```

```
                        ; ASCII letter. If one present, it is converted to upper case.
                        ; If not present, nothing done.  Strips parity first.
1102  391 e6 7f         ucase:  ani     07fh    ;strip parity
1103  393 fe 61                 cpi     61h
1104  395 3f                    cmc
1105  396 d0                    rnc             ;don't convert if before 'a'
1106  397 fe 7b                 cpi     7bh
1107  399 d0                    rnc             ;don't convert if after 'z'
1108  39a d6 20                 sui     20h     ;convert lower to upper
1109  39c c9                    ret
                        ;
                        ; ROM constant allocation - alphabetical order
                        ;
                        abort:  db      ' ABORT !'
1113  39d 20            41 42 4f 52 54 20 21
                        mcrlf:  db      0dh,0ah,0ffh
1114  3a5  d             a ff
                        bad:    db      ' WHAT ?'
1115  3a8 20            57 48 41 54 20 3f
1116  3af ff                    db      0ffh
1117  3b0 4c            cmds:   db      'L'             ;command table
1118  3b1 9e  1                 dw      loader          ;
1119  3b3 45                    db      'E'             ;
1120  3b4 2a  1                 dw      memed           ;
1121  3b6 47                    db      'G'             ;
1122  3b7 19  1                 dw      goto            ;
1123  3b9 44                    db      'D'             ;
1124  3ba ee  1                 dw      d_p             ;common code for dump and punch
1125  3bc 50                    db      'P'             ;  commands.
1126  3bd ee  1                 dw      d_p             ;  .
1127  3bf  0                    db      0               ;end of table mark
                        endrec: db      0dh,0ah         ;end of record for punch
1128  3c0  d             a
                                  db      ':00000001FF'   ;
1129  3c2 3a            30 30 30 30 30 30 30 31 46 46
1130  3cd ff                    db      0ffh
                        form:   db      ') = '
1131  3ce 29            20 3d 20
1132  3d2 ff                    db      0ffh
                        form2:  db      0dh, 0ah
1133  3d3  d             a
1134  3d5 28                    db      '('
1135  3d6 ff                    db      0ffh
                        mok:    db      ' OK ?'
1136  3d7 20            4f 4b 20 3f
1137  3dc ff                    db      0ffh
                        phi:    db      ' TO '
1138  3dd 20            54 4f 20
1139  3e1 ff                    db      0ffh
                        plo:    db      ' FROM '
1140  3e2 20            46 52 4f 4d 20
1141  3e8 ff                    db      0ffh
                        prmpt:  db      0dh, 0ah
1142  3e9  d             a
                                  db      ' >'
1143  3eb 20            3e
1144  3ed ff                    db      0ffh
                        start:  db      0dh,0ah
1145  3ee  d             a
                                  db      'M3F8.6'
1146  3f0 4d            33 46 38 2e 36
                                  db      0dh,0ah,0ffh
1147  3f6  d             a ff

                        ;
                        ; RAM allocation if alphabetical order
                        ;
      13ff                        org     MEMTOP-0        ;MEMTOP - (# bytes alloc - 1)
                        ;
      13ff              STACKINIT         equ     $       ;initial stack pointer overlaps
                        ;                                 ;lowest byte allocated.
                        ;
1157 13ff  1            answer  db      1               ; answer to question
                        ;
                        ; At this point $ should = MEMTOP
                        ;
                                end

   0 BASE                      28a CI                      294 CICO
 297 CO                          0 CRDLY                   13ff MEMTOP
```

```
13ff STACKINIT
13ff answer
  7a baud1
  ae baud5
 2b1 c3
 3bØ cmds
 2bc d1Øms
 22c di1
 1ae done
 16Ø e1
 177 e4
  6a entry
 3ce form
 2e1 ghb
 3Ød ghd2
 119 goto
 1da loop
 31f mdn
 31Ø msg
 322 okck
 34a phb
 3dd phi
 261 pnxtbyt
 3e9 prmpt
 373 repeat
1Ø2Ø rst75
 37d space
  23 timctl
 391 ucase
   f WIDTH
 1Øf ask
  8c baud3
 298 c1
 1Ø2 cmdfnd
  d2 comnd
 1ee d_p
 19Ø discon
 2c1 dtwid1
 169 e2
 148 edit
  fe error
 3d3 form2
 2f4 ghbend
 2ca ghw
 1e6 lghb
 312 loupe
 12a memed
 157 next
 33b okckend
 35a phd
 33f phw
 17d pr
 247 punch
1Ø1Ø rst55
   1 sercon
 3ee start
  21 time1
 39d abort
 3a8 bad
  92 baud4
 2a7 c2
  f2 cmdnxt
 2b3 crlf
 1d9 data
 183 dismem
 21a dump
 16e e3
 3cØ endrec
 1cc fndmrk
 1b2 getrec
 2f6 ghd
 2de ghwend
 19e loader
 3a5 mcrlf
 3d7 mok
 13f ok
 27d pdone
 367 phd1
 3e2 plo
 36d prbad
 375 rep1
1Ø18 rst65
   Ø serdat
 385 sub16
1ØØ8 trap
```