```
  exec 4003

T/S START: 1000
T/S END: 1008
MEM START: 2000
LAST MEM ADDR: 24 7f
? fixfADDR? 2000
? list
0000  org 03fah
0010  jmp evcod
0100  org 0067h
0110 ************************************************************
0120 *
0130 * Module EVCOD
0140 *
0150 * Written by Bruce Cichowlas 8/25/76
0160 * Copyrisht Circle C - ARP Instruments 9/18/76
0170 *
0180 * EVCOD - evaluates preset code by table lookup
0190 *
0200 * C contains input code on entry
0210 * A has answer (-1 indicates invalid code)
0220 *
0230 ************************************************************
1000 evcod equ $
1010  push h
1020  lxi h,cdtab ;code table
1030 evco1 mov a,m
1040  inx h
1050  inx h
1060  cmp c
1070  jz evco2
1080  cpi -1  ;indicates end of table
1090  jnz evco1
1100  mvi a,-1 ;return invalid indication
1110  pop h
1120  ret
1130 evco2 dcx h  ;back up one
1140  mov a,m
1150  pop h
1160  ret
1170 cdtab dw 0101h
1180  dw 0202h
1190  dw 0304h
1200  dw 0408h
1210  dw 0510h
1220  dw 0620h
1230  dw 0040h
1240  dw 0703h
1250  dw 0806h
1260  dw 090ch
1270  dw 0a18h
1280  dw 0b30h
1290  dw 0c60h
1300  dw 0ffffh
9999 *END

?
```

```
  exec 4003
T/S START: 1200
T/S END: 1209
MEM START: 2000
LAST MEM ADDR: 24 ff
? fixfADDR? 2000
? list
0000  org 03fdh
0010  jmp presc
0100  org 0040h
0110 ************************************************************
0120 *
0130 * Module PRESC
0140 *
0150 * Written by Bruce Cichowlas 8/25/76
0160 * Copyright Circle C - ARP Instruments 9/15/76
0170 *
0180 * PRESC - checks the preset panel and sets the previous
0190 *  preset readings (prvpr), as well as the preset request
0200 *  word (prerq).  This routine should be invoked frequently,
0210 *  as it actually does the preset panel scanning.
0220 *
0230 *  No arguments
0240 *
0250 ************************************************************
1000 presc equ $
1010  push b
1020  lda ppadr    ;preset panel addr
1030  ora a        ;set flags
1040  jz pres1     ;none set
1050  mov b,a
1060  lda prvpr    ;load previous readings
1070  ora b
1080  sta prvpr
1090 pres2 pop b
1100  ret
1110 pres1 lda prvpr ;find out if any were set previously
1120  ora a
1130  jz pres2 ;none before
1140  mov b,a
1150  lda prerq
1160  ora b        ;include bits set in b
1170  sta prerq
1180  xra a
1190  sta prvpr  ;store 0 for the previous readings
1200  pop b
1210  ret
5000 ppadr equ 1800h
6000 *RAM
6001 prvpr equ 08ffh
6002 prerq equ 08feh

?
```

```
  exec 4003
exec 4003

T/S START: 1400
T/S END: 1419
MEM START: 2000
LAST MEM ADDR: 2c ff
? fixfADDR? 2000
? fi^^list
0000  org 03e8h
0001  jmp rdsld
0002  jmp clman
0003  jmp nmmsk
0004  jmp msknm
0005  jmp swpnl
0006  jmp clsw
0100  org 009bh
0110 ************************************************************
0120 *
0130 * Written by Bruce Cichowlas 9/3/76
0140 * Copyrisht Circle C - ARP Instruments 9/16/76
0150 *
0160 * CLSW - clears all switches.  No arsuments.
0170 *
0180 * SWPNL - sets the switches from the scratchpad.  Just
0190 *  one byte of them for now.  No arsuments.
0200 *
0210 * MSKNM - Converts a one bit mask to a number from zero
0220 *  to seven indicatins bit position, e.s. 01h becomes 0,
0230 *  80h becomes seven.  Input and output are from the A resister.
0240 *
0250 * NMMSK - Performs the inverse operation of MSKNM.
0260 *
0270 * CLMAN - Clears the manual lishts for the slidepots.  No
0280 *  arsuments
0290 *
0300 * RDSLD - Reads a slidepot.  On input, A contains
0310 *  the slidepot number, and on return it contains the
0320 *  slidepot's current value.
0330 *
0340 ************************************************************
1000 clsw equ $
1010  sta swclr
1020  ret
1030 swpnl equ $
1040  push h
1050  push d
1060  lhld prbas ;base of scratchpad
1070  lda slcnt   ;slide count
1080  mov e,a
1090  mvi d,0
1100  dad d       ;calculate address of switch values in the scratchpad
1110  mov a,m     ;set word
1120  cma         ;since switches use nesative losic
1130  sta setsw   ;set the switches
1140  pop d
1150  pop h
1160  ret
1170 msknm equ $
1180  push b
1190  mvi b,7
1200 mskn1 rlc
1210  jc mskn2
1220  dcr b
```

```
1230  jnz mskn1
1240 mskn2 mov a,b
1250  pop b
1260  ret
1270 nmmsk equ $
1280  push b
1290  mov b,a
1300  mvi a,1
1310  inr b
1320 nmms1 dcr b
1330  jz nmms2
1340  rlc
1350  jmp nmms1
1360 nmms2 pop b
1370  ret
1380 clman equ $
1390  push b
1400  push d
1410  push h
1420  lda slcnt  ;slide count
1430  dcr a      ;convert to offset
1440 clma1 push m
1450  lhld lsbas ;man/auto light copy base
1460  call wrdbt ;set wrdbt offset
1470  xra a      ;for reset
1480  call setwb ; adjust bit
1490  lxi h,lsdev ; light device
1500  mov e,b
1510  mvi d,0
1520  dad d
1530  mov m,a    ;store new value
1540  pop m
1550  dcr a
1560  jp clma1
1570  pop h
1580  pop d
1590  pop b
1600  ret
1610 rdsld equ $
1620  sta mxadr ;slidepot's MUX addr
1630  nop       ;wait for MUX to settle
1640  nop
1650  sta strtc ;start conversion
1660  mvi a,6
1670 rdsl1 dcr a ;delay
1680  jnz rdsl1
1690  lda adout   ;set a/d value
1700  ret
7000 *ROM parameter table
7001 prbas equ 0400h ;base of scratchpad
7002 slcnt equ 0402h ;slide pot count
7003 lsbas equ 0403h ;man/auto light copy base
8000 *Device addresses
8001 swclr equ 3003h ;store to this location clears the switch settings
8002 setsw equ 3004h ;store to this setting stores the complement of the
8003 *            ;input into the switches
8004 mxadr equ 3000h ;store to here sets the MUX slidepot address
8005 strtc equ 3002h ;store to here starts the conversion
8006 adout equ 3000h ;this is where the a/d output is put
8007 lsdev equ 3001h ;store to here sets the man/auto lights
9000 *Subroutines
9001 wrdbt equ 03e5h
9002 setwb equ 03e2h

?
```

```
  exec 4003

T/S START: 1600
T/S END: 1619
MEM START: 2000
LAST MEM ADDR: 2c ff
? fixfADDR? 2000
? list
0000  org 03dch
0001  jmp setwb
0002  jmp stman
0003  jmp setwb
0004  jmp wrdbt
0100  org 0105h
0110 ************************************************************
0113 *
0116 * Module WRDBT
0120 *
0130 * Written by Bruce Cichowlas 9/3/76
0140 * Copyright Circle C - ARP Instruments 9/16/76
0150 *
0160 * SETWB - Sets or resets a particular bit in a word.
0170 *
0180 * On entry:
0190 *  HL contains the base address of the table being accessed
0200 *  B - contains the desired word offset within the table
0210 *  C - contains the bit offset within the word (from bit 0)
0220 *  A - contains -1 if the bit is to be set, and zero if the
0230 *      bit is to be reset
0240 *
0250 * On exit:
0260 *  A - contains the resulting word
0270 *
0280 * GETWB - Gets the contents of a particular bit.  The parameters
0290 *  are basically the same as WRDBT with these exceptions:
0300 *
0310 * On entry, A is irrelevant
0320 *  On exit, A contains -1 if the bit is on, and 0 if the bit is off.
0330 *
0340 * WRDBT - Converts a bit offset to a word/ bit offset pair.
0350 *
0360 * On entry, A is the bit offset
0370 *
0380 * On exit:
0390 *  B - contains the resulting word offset.
0400 *  C - contains the remaining bit offset (<=7).
0410 *
0420 * STMAN - This call sets the synthesizer to manual operation as
0430 *  far as the slide pots are concerned.
0440 *
0450 *  No arguments
0460 *
0470 ************************************************************
1000  org 0105h
1010 setwb equ $
1020  push b
1030  push d
1040  push h
1050 *set word
1060  mov e,b
1070  mvi d,0
1080  dad d       ;HL now points to word
1090  mov a,c
1100  call mmmsk ;make up bit mask
```

```
1110  ana m       ;mask memory word
1120  mvi a,0     ;flags remain set
1130  Jz setw1    ;Jump if bit was off
1140  dcr a       ;will set a to -1
1150 setw1 pop h
1160  pop d
1170  pop b
1180  ret
1190 *
1200 wrdbt equ $
1210  mvi b,0
1220 wrdb1 cpi 8
1230  Jc wrdb2    ;b is correct
1240  sbi 8
1250  inr b
1260  Jmp wrdb1
1270 wrdb2 mov c,a
1280  ret
1290 *
1300 setwb equ $
1310  push b
1320  push d
1330  push h
1340 *set word
1350  mov e,b
1360  mvi d,0
1370  dad d       ;hl now points to desired word
1380  mov e,a     ;temporary storage
1390  mov a,c     ;make bit mask
1400  call nmmsk
1410  mov d,a     ;mask is now in d
1420  mov a,e     ;set back request
1430  ora a       ;set flag
1440  mov a,d
1450  Jm setw1    ;set
1460 *reset logic
1470  mov a,d
1480  cma
1490  ana m
1500 setw2 mov m,a
1510  pop h
1520  pop d
1530  pop b
1540  ret
1550 *set logic
1560 setw1 mov a,d
1570  ora m
1580  Jmp setw2
1590 *
1600 stman equ $
1610  push b
1620  push d
1630  push h
1640  lda slcnt   ;slide count
1650  dcr a
1660 stma1 push m ;save a
1670  lhld lsbas ;light base
1680  call wrdbt
1690  mvi a,-1    ;set
1700  call setwb
1710  lxi h,lsdev  ;light device
1720  mov e,b
1730  mvi d,0
1740  dad d
1750  mov m,a
1760  pop m
```

```
1770  dcr a
1780  jp stma1
1790  pop h
1800  pop d
1810  pop b
1820  ret
7000 *ROM parameter table
7001 slcnt equ 0402h ;slide pot count
7002 lsbas equ 0403h ;man/auto light copy base
8000 *Devices
8001 lsdev equ 3001h ;store to here sets man/auto lights
9000 *Subroutines
9001 nmmsk equ 03eeh

?
```

```
  exec 4003

T/S START: 1800
T/S END: 191fr^^^^^181f
MEM START: 2000
LAST MEM ADDR: 2f ff
? fixfADDR? 2000
? list
0000  org 03d9h
0001  jmp prern
0100  org 026ch
0110 ************************************************************
0115 *
0120 * Module PRERN
0125 *
0130 * Written by Bruce Cichowlas 9/3/76
0140 * Copyright Circle C - ARP Instruments 9/16/76
0150 *
0160 * PRERN - does all on the necessary preset activity.
0170 *  Works using the preset request word (prerq) generated by
0180 *  program "presc".
0190 *
0200 *  No arguments
0210 *
0220 ************************************************************
0230 prern equ $
0240  lda prerq  ;set the preset request word.
0245  ora a      ;set flags
0250  rz         ;return if there is nothing to do.
0260  ani 80h    ;see if the write bit is on.
0270  jz prer1   ;not on
0280  mvi a,1    ;indicates write operation is next
0290  sta wropn  ;RAM byte which is one if write operation is coming up.
0300  lda pecho  ;preset light "echo" word
0310  ori 80h    ;turn on "write" light
0320  sta pecho
0330  sta plght  ;also actually put it on the panel lights
0340 prer1 lda prerq ;set the request word again
0350  ani 7fh    ;ignore the "write" bit
0360  rz         ;if none of the other bits are on, we can exit
0370  push b
0380  mov c,a    ;in preparation for "evcod"
0390  sta pecho  ;note that this puts write light out if it was on
0400  sta plght
0410  xra a
0420  sta prerq   ;will have satisfied all requests
0430  call evcod ;answer is in a
0435  sta prstn
0440  pop b
0450  ora a      ;set flags
0460  jz manl    ;zero indicates manual operation
0470  jp prer2   ;valid combination
0480  xra a
0490  sta prerq  ;ignore this invalid request
0500  ret
0510 prer2 lda wropn ;see if this is supposed to be a write
0520  ora a
0530  jnz prewr  ;it is
0540 *READ LOGIC
0550  push b
0560  push d
0570  push h
0575  lda prstn
0580  mov c,a    ;put preset number in c
```

```
0590  lhld prbas ;hl->scratchpad
0600  lda preln  ;preset length
0610  mov b,a    ;b=preset length
0620  mov e,a
0630  mvi d,0    ;de=preset length
0640 prer3 dad d ;hl->next preset
0650  dcr c
0660  jnz prer3
0670  xchg       ;de->preset
0680  lhld prbas ;hl->scratchpad
0690 prer4 ldax d ;transfer preset
0700  mov m,a
0710  inx h
0720  inx d
0730  dcr b
0740  jnz prer4
0750  call clman ;clear manual lights
0760  call clsw  ;clear panel switches
0770  call swpnl ;move the current switch setting to the panel
0780 pops pop h
0790  pop d
0800  pop b
0810  ret
0820 *MANUAL LOGIC
0830 manl call stman  ;set manual lights on
0840  ret
0850 *WRITE LOGIC
0860 prewr push b
0861  push d
0862  push h
0863 *Copy switches into scratchpad
0864 *(Just set up for one byte worth for now)
0865  lhld prbas ;hl->scratchpad
0866  lda slcnt ;last word of scratchpad is switches
0867  mov e,a
0868  mvi d,0
0869  dad d       ;hl->switches in scratchpad
0870  lda swdev   ;set current switch setting
0871  mov m,a    ;copy into scratchpad
0885  lda prstn
0890  mov c,a    ;put preset number in c
0900  lhld prbas ;hl->scratchpad
0910  lda preln  ;preset length
0920  mov b,a   ;b=preset length
0930  mov e,a
0940  mvi d,0    ;de=preset length
0950 prer5 dad d  ;hl->next preset
0960  dcr c
0970  jnz prer5  ;not there yet
0975  xchg
0980  lhld prbas
0985  xchg       ;de->scratchpad
0990 prer6 ldax d ;move from scratch pad to preset storage
1000  mov m,a
1010  inx h
1020  inx d
1030  dcr b
1040  jnz prer6
1043  mvi a,0    ;turn off "wropn"
1046  sta wropn
1050  jmp pops   ;restore resisters and return
6000 *RAM
6001 prera equ 08feh ;preset request word
6002 wropn equ 08fdh ;write option word
6003 pecho equ 08fch ;preset echo word
6004 prstn equ 08fbh ;this preset number
```

```
7000 *ROM parameters
7001 prbas equ 0400h ;base of scratchpad (and presets)
7002 preln equ 0405h ;preset length
7003 slcnt equ 0402h ;slide pot count
8000 *Devices
8001 plsht equ 1c00h ;preset lights
8002 swdev equ 3001h  ;panel switches
9000 *Subroutines
9001 evcod equ 03fah
9002 clman equ 03ebh
9003 clsw equ 03f7h
9004 swpnl equ 03f4h
9005 stman equ 03dfh
9999 *END

?
```

```
  exec 4003

T/S START: 1a00
T/S END: 1a1c
MEM START: 2000
LAST MEM ADDR: 2e 7f
? fixfADDR? 2000
? list
0000  ors 03d6h
0001  jmp mn30p
0100  ors 031eh
0110 ************************************************************
0120 *
0130 * Module MN30P
0140 *
0150 * Written by Bruce Cichowlas 9/3/76
0160 * Copyright Circle C - ARP Instruments 9/16/76
0170 *
0180 * MN30P - the main controlling module for the 2430 system.
0190 *  This should be given control on initialization.
0200 *  (A transfer to the start of this should normally be placed
0210 *  at location zero.)
0220 *
0230 *  No arguments
0240 *
0250 ************************************************************
1000 mn30p equ $
1005  lxi m,0840h ;initialize stack pointer
1010  call mn30i  ;initialize the system
1020 mn301 lda slcnt  ;slidepot count
1030 -dcr a
1040 mn302 push m ;save count
1050  mov e,a
1060  mvi d,0 ;de=slide pot position offset (among other things)
1070  lhld lsbas ;man/auto copy status base
1080  call wrdbt ;make bc be a word/bit offset for a
1090  call setwb ;see if this one is manual or automatic
1100  ora a
1110  jz mn303   ;automatic
1111 *MANUAL SLIDEPOT PROCESSING
1112  mov a,e
1113  call rdsld
1117  lhld prbas ;hl->scratchpad
1118  dad d      ;point to scratchpad word
1119  mov b,a   ;b=slidepot
1120  mov a,m    ;scratchpad value
1121  sub b     ;find difference
1122  inr a      ;a=a-1
1123  cpi 5 ;if < 5, don't update
1124  jc mn306
1125  lxi h,shbas ;sample/hold base
1126  dad d
1127  mov m,b    ;update
1128  lhld prbas ;scratchpad base
1129  dad d
1130  mov m,b    ;update scratchpad, too
1131 mn306 jmp mn304
1132 *AUTOMATIC SLIDEPOT PROCESSING
1139 mn303 mov a,e ;restore a from e
1140  call rdsld ;read that slidepot
1150  lhld prbas ;hl->scratchpad
1160  dad d      ;hl->this slidepot setting
1170  mov b,a    ;b=actual slidepot setting
1180  cmp m      ;compare with setting on scratchpad (currently this
```

```
1190 *            ;calls for an exact match.  We could make it require
1200 *            ;a less precise match, if this proves to be desirable)
1210  Jz mn305    ;the same
1211 *activity if not the same
1212  mov a,m
1213  lxi h,shbas
1214  dad d       ;point to this one
1215  mov m,a
1216  Jmp mn304
1220 mn305 mov a,e ;set to manual logic
1230  call wrdbt
1240  mvi a,-1    ;to indicate set
1250  lhld lsbas  ;man/auto light base
1260  call setwb
1270  lxi h,lsdev  ;now actually change lights
1280  push d      ;don't clobber d
1290  mov e,b
1300  mvi d,0
1310  dad d       ;hl->correct light word
1320  mov m,a     ;change the lights
1330  pop d
1340 mn304 call presc  ;check preset panel for activity
1350  call prern ;process any activity
1360  pop m       ;set back slide counter that we stored at the
1370 *            ;start of this loop
1380  dcr a
1390  Jp mn302
1400  Jmp mn301  ;start again
1410 *
1420 mn30i equ $ ;initialization routine
1430  xra a
1440  sta wropn
1450  sta prera
1455  sta pecho
1458  mvi a,40h   ;start with manual operation
1460  sta prvpr
1470  ret
6000 *RAM
6001 prvpr equ 08ffh ;previous preset readins
6002 prera equ 08feh ;preset request word
6003 wropn equ 08fdh ;write option word
6004 pecho equ 08fch ;preset echo word
7000 *ROM parameters
7001 prbas equ 0400h ;base of scratchpad
7002 slcnt equ 0402h ;slide pot count
7003 lsbas equ 0403h ;man/auto light copy base
8000 *Devices
8001 shbas equ 3400h ;base of sample/hold locations
8002 lsdev equ 3001h ;lights to indicate man/auto operation
9000 *Subroutines
9001 wrdbt equ 03e5h
9002 setwb equ 03dch
9003 rdsld equ 03e8h
9004 presc equ 03fdh
9005 prern equ 03d9h
9006 setwb equ 03e2h
9999 *END

?
```

```
  exec 4003
T/S START: 1c00
T/S END: 1c06
MEM START: 2000
ERROR AT T/S: 1c 00
LAST MEM ADDR: 23 7f
? fixfADDR? 2000
? list
0000 ************************************************************
0001 *
0002 * Module R30A
0003 *
0004 * Written by Bruce Cichowlas 9/16/76
0005 * Copyrisht Circle C - ARP Instruments 9/16/76
0010 *
0020 * R30A - This module just contains a table of the ROM parameters
0030 *  (essentially the "SYSGEN" parameters).  These can be chansed
0040 *  as appropriate and reloaded by re-assemblins this module.
0050 *
0060 ************************************************************
1000  ors 0400h ;ROM parameter table location
1010 prbas dw 0c00h ;base of scratchpad
1020 slcnt dw 6 ;slide pot count (actually just one byte)
1030  ors $-1  ;due to db bus
1040 lsbas dw 08c0h ;man/auto lisht copy base
1050 preln dw 7  ;lensth of each preset record (one byte)
1060  ors $-1
9999 *END
?
```