```
                        * These routines are responsible for maintaining the time-of-day clock,
                        *       and for transmitting it to and receiving it from the other CPU's in
                        *       COED.


                        * The time is transmitted using the "Multi-processor Interrupt" as a single
                        *       49 bit character, as if the interrupt line were an asynchronous commun-
                        *       ications line.



                        * Multi-processor Interrupt

                        * This routine uses R12 to count the data bits received.  It
                        *       expects R12 to be zeroed during each "bit time".
                        *       If R12 becomes to large, indicating some CPU is improperly trans-
                        *       mitting, the interrupt is disabled using "RIE,3".  On each data bit,
                        *       it sets the timer counting "bit times", CKBT, to the middle of
                        *       its period, -2.

1002  0302+  72CF    E3$TIM ABRB,R12,B15    TIM1        count the data bit
      0303+  0305+
1003  0304+  2500           CIR                          then quit if we sent it
1004  0305+  66CD    TIM1   TBR,R12,B13                  did we get 4 interrupts this "bit time"?
1005  0306+  AB82           HCR,$+2
1006  0307+  2743           RIE,3                        yes, turn off the interrupt

1008  0308+  E610           STM,R1          CKBT        Save R1 in CKBT
      0309+  0532+
1009  030A+  671E           LDC,R1          -2          set the "bit timer" to -2
100A  030B+  B610           IRM,R1          CKBT        and reload R1
      030C+  0532+
100B  030D+  2500           CIR                         finished
```

```
1013                    * This is the actual real-time, level 6 clock interrupt.
1014                    *       Note that it clobbers the overflow register.



1018                    * This routine was written to minimized code and average execution time.
1019                    *       That is, the worst-case execution of this interrupt is shocking, but
1020                    *       the average is dominated by the simplest, fastest cases.



1022            0001    GRAPH   EQU             1           1 for bargraph in R8, -1 to leave out

1024            030E+   E6$CLK  EQU             $
1026    030E+   E510            LDM,R1          CJCB
        030F+   0009
1027    0311+   E911            CRI,R1          B$JCB
        0312+   0F1DX
1028    0313+   A903            BZS,$+3
1029    0314+   8EF0            ABMM,B15        BARGPA
        0315+   0534+
1031    0316+   84F0            ABMB,B15        CKBT,CLK3P2 branch if not at end of a "bit time"
        0317+   0532+
        0318+   03A1+
1032    0319+   ED1C            LDC,R1          -3          "bit time" is up, so reset it
        031A+   FFFD
1033    031B+   E610            STM,R1          CKBT
        031C+   0532+
1034    031D+   6F4C            TTR,R4,R12
1035    031E+   6CCC            ZRR,R12                     reset count of bits during this time
1036    031F+   E7C6            BRU*            TIMPST      go do post-processing on the current state
        0320+   0531+



1038                    * The several states required to transmit and receive the time-of-day are
1039                    *       implemented with an abstract machine having 5 different instructions
1040                    *       and two addresses.  Each "instruction" is begun at the end of one
1041                    *       "bit time", and finished immediately, or at the start of a following
1042                    *       "bit time".  The two "addresses" specify "locations" in the state table,
1043                    *       TMSTBL.  They are called "0" and "1" exits, and are usually choosen
1044                    *       according to the state of the last data bit received.

1046                    *       The code for each "instruction" has two entries:  one is labeled
1047                    *       CLKOn0 and the other CLKOn5, where n=the "instruction" number.  The
1048                    *       first is used when starting an "instruction" and the second when
1049                    *       continuing one during successive bit times.

1051                    * The following code implements the instructions themselves:
```

```
1053                    * Transmit a PREamble--Instruction number 0

1055  0320+  ED70  CLK000 LDI,R7          ^B9|^B11    forget about previous interrupts
      0321+  0050
1056  0322+  C170        ETMM,R7          TIMAST
      0323+  052D+
1057  0324+  E514        LDM,R1,R4        CLKOLD      GET THE DESIRED PREAMBLE LENGTH
      0325+  0336+
1058  0326+  E610        STM,R1           CKET
      0327+  0532+
1060                      EXT              CHRONO      Cause chrono routines to load
1061  0328+  E710        BLM,R1           CHRONO      Get time from Chronolog
      0329+  002BX
1063  032A+  2643        SIE,3                        start listening to RMI's
1064  032B+  E700        JMP              CLK300      after setting it, wait for it to end
      032C+  03A1+



1068  032D+  7D44  CLK005 TRRB,R4,R4       CLK201      take "1" exit if we got any bits
      032E+  0382+
1069  032F+  E542        LDM,R4           CLK$2       get ticks until end of current even minute
      0330+  052C+
1070  0331+  6047        ABR,R4,R7
1071  0332+  A700        JNS,R4           CLK200      take "0" exit if it won't end for 256 ticks
      0333+  0381+
1072  0334+  E700        JMP              CLK300      otherwise, wait a while.
      0335+  03A1+



1076                    * Table of PREamble lengths in 5 millisecond ticks.

1078          0000  A     SET              (MODAC20&#FF)*3 priority in bits for below
1079                *     It essentially encodes the reliability of this CPU's time-keeping
1080                *     (as well as the existence of a CHRONO-LOG).

1082                *     Any transmitted preamble must be at least 1 second long to permit
1083                *     the CHRONO-LOG interrupt to cycle and at least 49 bit times to be
1084                *     certain the bus is quiet.  Any preamble is limited to the number of 5
1085                *     msec. ticks which will fit in one 16 bit word.

1087                *     For every possible value of A, there are four different, associated
1088                *     preambles.  Three of those preambles depend upon A.  Thus, if N is
1089                *     the number of CPU's on the RMI bus, then there are 3N+1 different
1090                *     preamble lengths.  NOTICE!!!  These values must be distinct!!!!

1092                *     The following are adjusted to the assumption that there are only a few
1093                *     such values.

1095  0336+  A240  CLK008 DFC              -120*200    Calm PREamble=2 minutes
1096  0337+  FF37        DFC              -201-A      Master PREamble=1 second
1097  0338+  FEF0        DFC              -261+A      Rebel PREamble=little bit to wait until
1098                *                                     lower priorities have had their say
1099  0339+  FF23        DFC              -221-A      Slave PREamble=little bit before lower
1100                *                                     priorites will speak
```

```
1102                    * RECeive the time--Instruction number 1
1103                    *       This "instruction" receives all 49 bits of the time-of-day, counting
1104                    *       the trailing stop bit.
1105                    *       NOTE, this instruction must have a non-zero augment.

1107            033A+   CLKO1C  EQU             $
1108    033A+   84F0            CLKO15  ABMB,R15        TMBCNT,CLK28D "WAIT" if it's not last data bit
        033B+   052F+
        033C+   039D+
1109    033D+   704F            ABRB,R4,B15     CLK2C1      take "1" exit if stop bit was bad
        033E+   0382+
1110    033F+   8280            OBMM,B8         TIMAST      Assume the time not from a CHRONO-LOG
        0340+   052D+
1111    0341+   7699            TBRB,R9,B9      CLKO1C
        0342+   0345+
1112    0343+   8180            ZBMM,B8         TIMAST      Change that if wrong
        0344+   052D+
1113    0345+   6299    CLKO1C  ZBR,R9,B9
1114    0346+   7DAA            TRRB,R10,R12    CLKO1?      branch if more than YEAR and MONTH received
        0347+   034F+
1115    0348+   AE49            MUR,R4,R9                   if no more, set for later "1" exit
1116    0349+   E590            LDM,R9          CLK$0       don't change anything but the year
        034A+   052A+
1117    034B+   CDB0            LDMD,R10        CLK$1
        034C+   052B+
1118    034D+   0C99            MLR,R9,R9
1119    034E+   6E94            ORR,R9,R4
1120    034F+   E690    CLKO17  STM,R9          CLK$0       set the new time-of-day
        0350+   052A+
1121    0351+   CEB0            STMD,R10        CLK$1
        0352+   052B+
1122    0353+   F72F            JMP             CLK2C1      then take the "0" or "1" exit



1124                    * TRANsmit the time--Instruction number 2
1125                    *       If the augment of the abstract machine instruction is 1, and this
1126                    *       CPU is a "SLAVE", then zero is used as the 2nd word of the time, to
1127                    *       indicate that only the year is valid.
1128                    *       The "1" exit is taken whenever some other CPU interrupts this CPU's
1129                    *       transmission.  Otherwise, the instruction pauses to output all 48
1130                    *       bits of the time, the one stop bit, and then takes the "0" exit.

1132                    * This instruction must always be preceeded by a quiet PREamble.

1134    0354+   E590    CLKO2C  LDM,R9          CLK$C       get our own current time
        0355+   052A+
1135                    *       TBMB,B11        TIMAST,$+4
1136                    *       OBR,R9,B9                   Mark the source of our time
1137    0356+   CDB0            LDMD,R10        CLK$1
        0357+   052B+
1138    0358+   E8B0            ADI,R11         49*3        make it correct for when it will be received
        0359+   0093
1139                    *                                   Note that the only reason we can do such a
1140                    *                                   simple add (without upping minute count) is
1141                    *                                   because all TRAN instructions are preceeded
```

```
1142                    *                                       by PRE's which don't end near the end of an
1143                    *                                       even minute.
1144  035A+ C640+       TRMB,R4         TIMAST,$+4
      035B+ 052D+
      035C+ 035E+
1145  035D+ F709        HOP,CLK026
1146  035E+ 6CAA        ZRR,R10                                 don't send the minutes if augment says so
1147  035F+ F707        JMP             CLK024                  and take the "wait" exit this time


1151  0360+ 6391  CLK024 OBR,R9,B1                             entry for CLK035

1153  0361+ 7D44  CLK025 TRRB,R4,F4     CLK251                  take the "1" exit if we were interrupted
      0362+ 0382+
1154  0363+ 8FF0        ABMM,B15        TMBCNT                  have we sent all 49 bits, including stop?
      0364+ 052F+
1155  0365+ A91D        JZS             CLK251                  yes, exit


1159  0366+ 674F  CLK026 GMR,R4,B15
1160  0367+ 7690        TERB,R9,B0      CLK268                  are we sending a zero?--no, simply WAIT
      0368+ 039D+
1161  0369+ 68C4        ADR,R12,R4                              yes, set to ignore our own interrupt
1162  036A+ 0108        RMI,8                                   and then cause an interrupt
1163  036B+ F732        JMP             CLK267                  then wait


1165                    * REQuest the time--Instruction number 3
1166                    *       Transmitts a request for the time.  A request consists of a single
1167                    *       RMI followed by 48 silences.  That is, it's a weird time-of-day
1168                    *       without a stop bit.  It must always be preceeded by a preamble
1169                    *       to see that things are quiet.  The 0 exit is taken unless some
1170                    *       other CPU is talk on the interrupt line.

1172          0354+ CLK031 EQU              CLK020                  this is cheating to save code.
1173          0360+ CLK035 EQU              CLK024


1175                    * Set SLAVE/MASTER bit--Instruction number 4
1176                    *       The bit is set with the augment in the "instruction", using unit's bit
1177                    *       to set MASTER/SLAVE.  The 2's bit is "OR'ed" into bit 14 to kill the
1178                    *       CHRONO-LOG.

1180  036C+ E550  CLK040 LDM,R5           TIMAST                  Get the old bits
      036D+ 052D+
1181  036E+ 625A        ZBR,R5,B10
1182  036F+ 615B        SBR,R5,B11                              (This may carry out of B9--set in CLK2001+?)
1183  0370+ EA50        ETI,R5          ^B11|^B15
      0371+ 0011
```

```
1184   0372+  6350          ORR,R5,B13                Clear forcing, and old aliveness
1185   0373+  6B54          ORR,R5,R4                 Set the new bits
1186   0374+  F650          STM,R5          TIMAST
       0375+  052D+
1187   0376+  F700          JMP             CLK200    then take the "0" exit



1189                 * Test "CHRONO-LOG-was-here" bit--Instruction number 5
1190                 *      This instruction immediately takes either the "0" exit or the "1" exit.
1191                 *      If the augment is 3, then the "0" exit is taken if the CHRONO-LOG
1192                 *                          was alive, but is now dead.
1193                 *      If the augment is 1, then the "0" exit is taken if the CHRONO-LOG
1194                 *                          is now dead.

1196   0377+  2D44    CLK050 LLS,R4,4
1197   0378+  C640          TRMB,R4         TIMAST,CLK201 take "1" exit if CHRONO-LOG as expected
       0379+  052D+
       037A+  0382+
1198   037B+  F706          JMP             CLK210    Else, take "0" exit



1200                 * Test the relative merit of our time--Instruction number 6
1201                 *      This instruction should only be used after receiving the time.  The "0"
1202                 *      exit is taken just in case our CHRONO-LOG is good and the transmitter's
1203                 *      isn't.

1205   037C+  ED40    CLK060 LDI,R4          ^B8|^B11
       037D+  0099
1206   037E+  E240          ETM,R4          TIMAST
       037F+  052D+
1207   0380+  F702          JMP             CLK201
```

```
1209                    * These are the "exits" to the "instructions".

1212                    * "0" branch

1214  0381+  6C44      CLK200  ZRR,R4

1217                    * "0/1" branch

1219  0382+  E550      CLK201  LDM,R5          TMSTAT      get the current, old "instruction"
      0383+  0530+
1220  0384+  7D44              TRRB,R4,R4      $+3
      0385+  0387+
1221  0386+  2955              RLS,R5,5
1222  0387+  EA50              ETC,R5          ~#1F        pick "0" or "1" branch depending on R4
      0388+  FFE3
1223  0389+  E545              LDM,R4,R5       ABSMEM      get the new "instruction"
      038A+  045D+
1224  038B+  E640              STM,R4          TMSTAT      and save it
      038C+  0530+
1225  038D+  2949              RLS,R4,9
1226  038E+  6D54              TRR,R5,R4
1227  038F+  2945              RLS,R4,5                    separate augment and opcode
1228  0390+  EA50              ETC,R5          ~#1E        pick "instruction" number times 2
      0391+  FFE1
1229  0392+  A465              LFM,R6,R5       TMUTBL      get addresses of code for new "instruction"
      0393+  044F+
1230  0394+  E670              STM,R7          TIMPST      set post-processing address
      0395+  0531+

1234  0396+  ED70              LDI,R7          -49
      0397+  FFCF
1235  0398+  E670              STM,R7          TMBCNT      and the total bit counter
      0399+  052F+
1236  039A+  8290              OBMM,B9         TIMAST      kill CHRONO-LOG interrupts
      039B+  052D+
1237  039C+  FF06              BRX,R6                      then start the "instruction" with R4=augment

1239                    * "WAIT" branch for TRAN and REC
1240                    *       Before waiting here, we shift the data being sent or received.
1241                    *       1's are shifted into the bottom just in case R4=0.

1243  039D+  2C91      CLK280  LLG,R5,1
1244  039E+  7D44              TRRB,R4,R4      $+3
      039F+  03A1+
1245  03A0+  63BF              OBR,R11,B15
```

```
1247                    * On every 5 millisecond tick, the time-of-day clock must be incremented.

1249  03A1+ 84F0  CLK300 ABMB,B15      CLK$2,CLK400 count a tick, is it last in an even minute?
      03A2+ 052C+
      03A3+ 03C1+
1250  03A4+ 677F         LDC,R7        -1           yes, do we reset it this time?
1251  03A5+ E670         STM,R7        CLK$2
      03A6+ 052C+
1252  03A7+ 841D         ABMB,B1       TIMSLW,CLK400 No, wait until later
      03A8+ 052E+
      03A9+ 03C1+
1253  03AA+ ED70         LD1,R7        -60*200*2
      03AB+ A240
1254  03AC+ E670         STM,R7        CLK$2        Yes, reset it
      03AD+ 052C+
1255  03AE+ 84F0         ABMB,B15      CLK$1,CLK400 and count the even minute
      03AF+ 052B+
      03B0+ 03C1+
1256  03B1+ E550         LDM,R5        CLK$0
      03B2+ 052A+
1257  03B3+ E850         ADC,R5        #13+1        also count the month at end-of-month
      03B4+ 0014
1258  03B5+ 765B         TBRB,R5,B11   CLK305
      03B6+ 03B9+
1259  03B7+ E85D         ADI,R5        #100-12      and the year if New-year's
      03B8+ FFF4
1260  03B9+ E950  CLK305 SUC,R5        #13
      03BA+ 0013
1261  03BB+ E650         STM,R5        CLK$1
      03BC+ 052A+
1262  03BD+ E740         BLM,R4        CLKCN4       if new month, figure even minutes until end
      03BE+ 0429+
1263  03BF+ E670         STM,R7        CLK$1        and set it
      03C0+ 052D+



1265                    * On every "system tick", we must do some housekeeping.
1266          03C1+ CLK400 EQU          $
1269  03C1+ 8400         ABMB,14-[FBIT]SY$TIK TICK$,CLK900 is this a "system tick"?
      03C2+ 0542+
      03C3+ 0414+
1270  03C4+ 0C2F         MLR,R2,R0                  yes, set to display memory
1271  03C5+ 2830         MBR,R3,R0
1272  03C6+ BEF2         LDAM,R15,R2                Display bottom 64 K in R15
1273  03C7+ 6837         ABR,R3,B7
1274  03C8+ BEE2         LDAM,R14,R2                Display 64K-128K in R14
1275                    * The VIEWJCB logic uses LDVM rather than LDM to preclude stalls if
1276                    * VIEWJCB contains a bad value.
1277  03C9+ E520         LDM,R2        VIEWJCB
      03CA+ 0533+
1278  03CB+ 653F         LDC,R3        1            System map image
```

```
1279   03CC+  ES20          ADC,R2          JIMET
       03CD+  8018
1280   03CE+  BE23          LDVM,R2,R2                  Get met
1281   03CF+  A90C          HZS,CLK402                  If not in a map now
1282   03D0+  ES20          ADC,R2          MMIAF
       03D1+  0003
1283   03D2+  BE13          LDVM,R1,R2                  Get Map Image page
1284   03D3+  A908          HZS,CLK402
1285   03D4+  BED1          LDVM,R13,R0                 Load virtual display to R13
1286   03D5+  AA86          HOR,CLK402                  If successful load
1287   03D6+  67D7          GMR,R13,B7                  No, flash the lights
1288   03D7+  8600          TBMB,B0         NDOG,CLK402
       03D8+  0545+
       03D9+  03DB+
1289   03DA+  08DD          MBR,R13,R13
1290   03DB+  84F0    CLK402 ABMB,B15       JCLOCK+1,CLK401 count accounting each system tick
       03DC+  FE55
       03DD+  03E0+
1291   03DE+  80F0          ABMM,B15        JCLOCK      accounting is double precision no.of ticks
       03DF+  FE54
1292   03E0+  84F0    CLK401 ABMB,B15       QUANTB,CLK410 reduce the quantum for current user
       03E1+  0529+
       03E2+  03E8+
1293   03E3+  8230          OBMM,CXS        CPUSW       and signal a change if he is finished
       03E4+  000D
1294   03E5+  8208          OBMM,MM         CPUSW       Keep memory manager on its toes
       03E6+  200D
1295   03E7+  268F          SIR,TSKLV
```

```
1297                 * Process the watch-dog queues.
1298          03E8+  CLK410 EQU             $
1301   03E8+  8420          ABMB,14-[FBIT](SD$TIK/SY$TIK) SDOG$,CLK900
       03E9+  0544+
       03EA+  0414+
1303                 * The following bargraph code was written assuming we come here every
1304                 * 80 ms.  The thing is hand coded and non-critical, so I didn't try
1305                 * real hard to parameterise it.  Vern would turn blue if he saw this.
1306   03EB+  6C11          ZRR,R1
1307   03EC+  B010          IRM,R1          BARGRA      Get count of number of utilised ticks
       03ED+  0534+
1308   03EE+  678F          GMR,R8,B15                  Assume all used
1309   03EF+  761B          TBRB,R1,B11     CLK403      If correct
       03F0+  03F5+
1310   03F1+  ED30          LDI,R3          #6780       GMR,R8,x
       03F2+  6780
1311   03F3+  6313          EXR,R1,R3
1312   03F4+  2D81          LLS,R8,1
1313          03F5+  CLK403 EQU             $
1315                 * The "short" watchdog has rolled over, so it may have died.
1316   03F5+  ED30          LDI,R3          SDOGG
       03F6+  0535+
1317   03F7+  E720          BLM,R2          EDOGS       bury any dead "short" dogs
       03F8+  041E+

1319                 * Check the "normal" dogs.
```

```
1321  03F9+  8410          ABMB,14-[FBIT](ND$TIK/SD$TIK) NDOG,CLK418 has normal watchdog rolled?
      03FA+  0545+
      03FB+  040D+
1322  03FC+  E720          BLM,R2          BDOGS       yes, bury them
      03FD+  041E+

1325  03FE+  8410          ABMB,14-[FBIT](OD$TIK/ND$TIK) ODOG,CLK418
      03FF+  0546+
      0400+  040D+
1326  0401+  E720          BLM,R2          BDOGS
      0402+  041E+

1329  0403+  8410          ABMB,14-[FBIT](MD$TIK/OD$TIK) MDOG,CLK418
      0404+  0547+
      0405+  040D+
1330  0406+  E720          BLM,R2          BDOGS
      0407+  041E+

1333  0408+  8430          ABMB,14-[FBIT](LD$TIK/MD$TIK) LDOG,CLK418
      0409+  0548+
      040A+  040D+
1334  040B+  E720          BLM,R2          BDOGS
      040C+  041E+

1336                       * Check the "timer" dogs.
1337  040D+  84B0    CLK418 ABMB,[FBIT]SD$TIK TDOG$,CLK900
      040E+  0543+
      040F+  0414+
1338  0410+  ED30          LDI,R3          TDOG0
      0411+  053F+
1339  0412+  E720          BLM,R2          BDOGS       bury the dead ones
      0413+  041F+

1341                       * Finished with Level 6 clock interrupt.

1343  0414+  2500    CLK900 CIR                         finished
```

```
1345                    * Bury some dead watchdogs for the clock routine.
1346                    *       Must be called with level SILOC or higher active.
1347                    * On entry:  R2=return address,  R3=address of pair of queue pointers.
1348                    *       NOTE:  The entry is :BDOGS:, not :BDOGS1:.
1349                    * On exit:  R2 unchanged,  R3=old R3+2,  bit LOCK in interrupt active register
1350                    *       reset, and everything else zapped.

1352  0415+ E740  BDOGS1 BLM,R4          DQ$DOG
      0416+ 04FC+
1353  0417+ ED70         LDI,R7          DDOGS$      queue a dead doc on deac-dog queue
      0418+ 053A+
1354  0419+ E740         BLM,R4          QDOG
      041A+ 051C+
1355  041B+ 8210         OEMM,CLOCK      CPUSW       flag the SCHEDULER
      041C+ 0000
1356  041D+ 268F         SIR,TSKLV


1360  041E+ E563  BDOGS  LDM,R6,R3       1           get list of now dead dogs
      041F+ 0001
1361  0420+ 7D16         TRRB,R1,R6      BDOGS1      go bury too dead one
      0421+ 0415+
1362  0422+ FD73         LDY,R7,R3                   if no dead dogs, move dying ones down
1363  0423+ 6D17         TRRZB,R1,R7     BDOGS9      don't even do that if nothing at all queued
      0424+ A963
1364  0425+ 90F8         ABSM,B15,DTQBAK             to move queue, must adjust 1st back pointer
1365  0426+ EE73         STXD,R6,R3
1366  0427+ 623E  BDOGS9 ABR,R3,B14                  advance R3 to next pair of queues
1367  0428+ FF02         BRX,R2                      exit
```

```
1369                    * Computes the number of even minutes in the current month.

1371                    * On entry: R4=mark word,  R5=current year/month in standard internal format,
1372                    * On exit:  R5=0,  R6=-1,  R7=minus the number of even minutes in current month.

1374   0429+  ED60   CLKCN$ LDI,R6          #3BB        these are magic numbers that happen to work
       042A+  03BB
1375   042B+  ED70          LDI,R7          #EECC       they encode the number of days in any month
       042C+  EECC
1376   042D+  7657          TBRB,R5,B7      CLKCN1
       042E+  0432+
1377   042F+  7656          TBRB,R5,B6      CLKCN1      branch if this is not a leap year
       0430+  0432+
1378   0431+  637B          OBR,R7,B11                  but it it is, set february to have 29 days


1381   0432+  0C55   CLKCN1 ETC,R5           ~#FF        pick out the number of the current month
1382   0433+  2862          RLD,R6,2
1383   0434+  715F          SBRB,R5,B15     $-1         and from that, pick 2-bit number of days
       0435+  0433+
1384   0436+  EA70          ETC,R7          ~3          clean the 2 bits
       0437+  FFFC
1385   0438+  E870          ADC,R7          28          the number is the offset from 28
       0439+  001C
1386   043A+  ED6F          LDI,R6          -24*60/2
       043B+  FD38
1387   043C+  2066          MPR,R6,R6                   then multiply by number of even minutes/day
1388   043D+  FF04          BRX,R4                      finally, we're finished
```

```
1390                    * Change the state of the time-of-day transmitting/receiving system.

1392                    * Entry:  R7+1=return,  (R7)=argument, one of the values ABSINI, ABSREQ, ABSYER,
1393                    *       or what ever.  These values cause the abstract machine to start
1394                    *       INItializing (after a power failure), REQuesting the time, sending the
1395                    *       YEaR, REBeling and sending our own time, or starting the CHRONO-LOG,
1396                    *       or what ever.  (The table of permisible values is below.)
1397                    * Exit:  Interrupt Level CLKLVL reset,  registers R1-R5 unchanged.

1399  043E+  2606      CLK$OP SIA,CLKLVL                      lock-out the clock interrupt
1400  043F+  FD67             LDX,R6,R7                       get argument
1401  0440+  E660             STM,R6          TMSTAT          set to start desired next instruction
      0441+  0530+
1402  0442+  ED60             ORI,R6          #FFF0
      0443+  FFF0
1403  0444+  C160             ETMM,R6         TIMAST          zero any desired bits in TIMAST, bits 13-15
      0445+  052D+
1404  0446+  ED60             LDI,R6          CLK2MK          and force the current instruction to end
      0447+  0381+
1405  0448+  E660             STM,R6          TIMPST
      0449+  0531+
1406  044A+  2706             RIA,CLKLVL
1407  044B+  2643             SIE,3
1408  044C+  2683             SIR,3                           make things happen pretty soon
1409  044D+  E707             BKU,R7          1               exit
      044F+  0001

1411                    * Arguments for CLK$OP.
1412                    *       Each is a value consisting of an abstract machine instruction with only
1413                    *       the "0" exit present, merged with the bottom 4 bits of TIMAST.
1414                    *       These bits are used to zero the corresponding bits.

1416           0007     ABSINI EQU              ABS00^5+7       Initialize after a power failure
1417           0220     ABSKIL EQU              ABS30^5         kill the CHRONO-LOG
1418           0240     ABSREQ EQU              ABS31^5         Request the correct time
1419           0280     ABSREB EQU              ABS40^5         SET-DATE, so Rebel and send our time
1420           02E0     ABSYER EQU              ABS50^5         SET-YEAR, so send it
1421           0326     ABSCLK EQU              ABS60^5+6       start believing our CHRONO-LOG clock
```

```
1423                    * This table converts the abstract machine's op-codes into pointers to
1424                    *       MODCOMP code to simulate both the pre- and post-processing required
1425                    *       by the abstract machine.

1427  0044F+  0320+  TMOTBL DFC             CLK000,CLK005 send PREamble
      0450+  0320+
1428  0451+  033A+         DFC             CLK010,CLK015 RECeive the time-of-day
      0452+  033A+
1429  0453+  0354+         DFC             CLK020,CLK025 TRANsmit the time-of-day
      0454+  0361+
1430  0455+  0354+         DFC             CLK030,CLK035 transmit a REQuest for the time-of-day
      0456+  0360+
1431  0457+  036C+         DFC             CLK040,0      set Master/Slave & CHRONO-LOG Enable bits
      0458+  0000
1432  0459+  0377+         DFC             CLK050,0      test for expected CHRONO-LOG interrupt
      045A+  0000
1433  045B+  037C+         DFC             CLK060,0      test for source of received time
      045C+  0000



1435                    * Abstract machine instruction definitions
1436                    APRE    MAC,,,[LNSR]                    send a preamble
1437                            AMAC,%%1,0,%%2,%%3
1438                            EMP
1439                    ARED    MAC,,,[LNSR]                    read the time-of-day
1440                            AMAC,1,1,%%1,%%2
1441                            EMP
1442                    AWIT    MAC,,,[LNSR]                    write the time-of-day
1443                            AMAC,0,2,%%1,%%2
1444                            EMP
1445                    AYER    MAC,,,[LNSR]                    write the year
1446                            AMAC,1,2,%%1,%%2
1447                            EMP
1448                    AREQ    MAC,,,[LNSR]                    request the time-of-day
1449                            AMAC,0,3,%%1,%%2
1450                            EMP
1451                    AMAS    MAC,,,[LNSR]                    set MASTER
1452                            AMAC,0,4,%%1,0
1453                            EMP
1454                    ASLAV   MAC,,,[LNSR]                    set SLAVE
1455                            AMAC,1,4,%%1,0
1456                            EMP
1457                    ACKIL   MAC,,,[LNSR]                    kill the CHRONO-LOG
1458                            AMAC,2,4,%%1,0
1459                            EMP
1460                    ACDED   MAC,,,[LNSR]                    take 0 exit if CHRONO-LOG dead
1461                            AMAC,1,5,%%1,%%2
1462                            EMP
1463                    ACDIE   MAC,,,[LNSR]                    take 0 exit if CHRONO-LOG dying
1464                            AMAC,3,5,%%1,%%2
1465                            EMP
1466                    ACHRO   MAC,,,[LNSR]                    Test the source of the time
1467                            AMAC,0,6,%%1,%%2
1468                            EMP

1470                    AMAC    DFF             2,4,5,5
```

```
1472                    * This is the state table for the routines sending and receiving time-of-day
1473                    *       among the CPU's of COED.  Another way of viewing it is as the memory
1474                    *       of the abstract machine.


1477          045D+     ABSMEM  EQU             $
1478    0000                    SPORG           0               start of the machine's memory.

1480                    * Initial instruction upon start-up of the abstract machine.
1481    045D+   4120    ABS00   APRE,1,$+1,$                    check that bus is quiet
1482    045E+   0C42            AREQ,$+1,$-1                    request the network's time
1483    045F+   C064    ABS01   APRE,3,$+1,ABS02                wait for it to arrive
1484    0460+   5455            ACDED,$-1,ABS41                 rebel if CHRONO-LOG good and it doesn't
1485    0461+   44A2    ABS02   ARED,$+1,ABS01                  read a time
1486    0462+   1AAC            ACHRO,ABS41,ABS20               rebel if we have CHRONO, but he doesn't

1488                    * This is the main Master loop.
1489    0463+   1EE0    ABS10   AMAS,$+1                        set MASTER
1490    0464+   C10B    ABS11   APRE,0,$+1,ABS18                send a Calm PREamble
1491    0465+   D669            ACDIE,ABS32,$+1                 become SLAVE if CHRONO-LOG just died
1492    0466+   0ECA            AWIT,ABS10,$+1                  transmit our time
1493    0467+   410A    ABS15   APRE,1,$-2,$                    Master PREamble and retry if interrupted

1495    0468+   458A    ABS18   ARED,ABS20,ABS15                Try to receive, transmit again if bad

1497                    * This is the main slave loop.
1498    0469+   51A0    ABS20   ASLAV,$+1                       set SLAVE
1499    046A+   C1D0            APRE,0,$+1,ABS29                send a long, calm preamble
1500    046B+   C1FA    ABS21   APRE,3,$+1,ABS29                then wait a little more
1501    046C+   0ECE            AWIT,ABS10,$-1                  send our time if nobody sends theirs

1503    046D+   458E    ABS29   ARED,ABS20,ABS21                receive time if anybody sends it


1506                    * These are the special entries.

1508    046E+   9240    ABS30   ACKIL,$+1                       kill the CHRONO-LOG
1509    046F+   4272    ABS31   APRE,1,$+1,$                    request the time, so send short PREamble
1510    0470+   0DD2    ABS32   AREQ,ABS21,$-1                  and then formally send the request

1512    0471+   92A0    ABS40   ACKIL,$+1                       "SET-DATE", so kill CHRONO-LOG
1513    0472+   82D5    ABS41   APRE,2,$+1,$                    send Rebel PREamble
1514    0473+   08D5    ABS42   AWIT,ABS10,$-1                  and then our time-of-day

1516    0474+   8317    ABS50   APRE,2,$+1,$                    "SET-YEAR", so send Rebel PREamble
1517    0475+   4BF7            AYER,ABS11,$-1                  and then our year.

1519    0476+   8359    ABS60   APRE,2,$+1,$                    start believing our CHRONO-LOG clock
1520    0477+   5676            ACDED,ABS32,ABS42               request or send time, depending.

1523    0478+                   RTORG                           end of the machine's memory
```

478
45D
1B
= 16+11=27

```
1525                        * Preset the clock's register block
1526   00A1+                        ABS
1527   00B2+                        ORG     GRBSAV+(RBSCLK-1)*15
1528   00B8+                        RES     7       R1-R7 = don't care
1529   00B7+  FA00                 DFC     -1      R8 = Bargraph (-1=complete utilization)
1530   00B8+                        RES     3       R9-R11=don't care
1531   00BB+  0000                 DFC     0       R12 = bit counter
1532   00BC+                        RES     1       R13 = Virtual memory display
1533   00BD+                        RES     2       R14-R15 = absolute memory display
1534   0478+                        REL
```