Chapter 4
Time-of-Day Counting

Time-of-Day Counting with the Am9513

Time-of-day (TOD) counting in the Am9513 is controlled by Master Mode register bits MM0 and MM1. When these bits are set to 01, 10 or 11, logic on Counters 1 and 2 is enabled to cause the counters to roll over at the counts required for Time-of-Day accumulation.

Figure 4-1 shows the format of the 24 hour clock. The two high order decades of Counter 2 contain the hours digits and can hold a maximum count of 23. The two low order Counter 2 decades indicate minutes and will hold values up to 59. The three most significant Counter 1 decades count up to 59.9 and are used to accumulate seconds and tenths of seconds. The low order Counter 1 decade is used to prescale the input frequency to generate tenth-of-second periods to the next higher counter decade.

Figure 4-2 shows the Counter 1 input frequencies supported. Note that the setting of Master Mode register bits MM0 and MM1 selects the optional prescaling factor used in Counter 1's least significant decade. The 50Hz (MM0 = 0, MM1 = 1) option and 60Hz (MM0 = 1, MM1 = 0) option permit AC powerline frequency sources to be used. When the 100Hz (MM0 = 1, MM1 = 1) option is used, the least significant decade of Counter 1 accumulates time in hundredths of a second (tens of milliseconds). Many convenient frequency sources, including the on-chip oscillator, may be used to drive the TOD clock.

Generating Time-of-Day Reference Frequencies

Bits CM11-CM8 in Counter Mode register 1 select the source input used to drive the Counter 1 Time-of-Day circuitry. The variety of inputs offered provides a number of easily implemented frequency sources.

If a 1MHz crystal is connected to the X1 and X2 oscillator inputs, and the BCD prescaling mode is selected (MM15 = 1), a 100Hz signal will be generated on internal signal F5. This F5 source can be selected as the count source for Counter 1 by setting Counter 1 Mode register bits CM11-CM8 to 1111.

As shown in Figure 4-3, crystals between 1 and 10MHz, in 1MHz increments, may also be used. The frequency prescaler is used with BCD scaling mode (MM15 = 1) to divide the crystal's frequency down to a multiple of 100Hz on F5. This F5 signal is then selected by Master Mode bits MM7-MM4 to drive the FOUT pin.

The FOUT scaler, programmed by bits MM11-MM8, is set to generate a 100Hz FOUT signal. The FOUT pin is externally strapped to one of SRC1-SRC5 or GATE1-GATE5. Counter 1 Mode register bits CM11-CM8 are then used to select the driven input pin.

If a spare counter is available, it can be used to prescale almost any crystal input frequency to produce an output rate appropriate for Counter 1's Time-of-Day input. Counter 5 is the easiest to use for this prescaling since its output appears as the TCN−1 input to Counter 1. This means no external strapping is required for the prescaled signal; all that is required is to set the Counter 1 Mode register bits CM11-CM8 to 0000 to select the TCN−1 input.

One readily available, low-cost crystal useful in this configuration is the 3.579545MHz TV color burst crystal. When divided by 59659, a 60Hz output is generated. Figure 4-4 shows a sample configuration using this crystal. Counter 5's Mode register should be set to use F1 as the source, to count down repetitively, to reload from the Load register and to disable special gating. The Gating Control Field should be set to "No Gating." Since internal concatenation is used, the Counter 5 Output Control Field setting is not relevant.

Note that in BCD mode the maximum counter value is 10000 decimal. If Counter 5 is to divide by 59659 decimal, the counter must operate in binary mode to get a range of 65535. Accordingly, binary counting should be selected in Counter 5's Mode register, and the Load register should be loaded with the binary representation of 59659 decimal (E90B hex). Figure 4-4 shows the required Counter 5 Load and Mode register configurations.

Initializing to Current Time-of-Day

The Time-of-Day circuitry requires a special initialization sequence. The following steps MUST be performed in the order given.

The first step is to set the Master Mode register and then the Counter Mode registers to the desired values. The Master Mode bits controlling Time-of-Day are MM0 and MM1; these have been discussed earlier. The user may also set Master Mode bits MM2 and MM3 at this time if the alarm feature is to be used.

The Counter 1 Mode register should be set to select the desired source input. For most real-time applications, the Gating Control field will be set for "No Gating," although edge and level gating

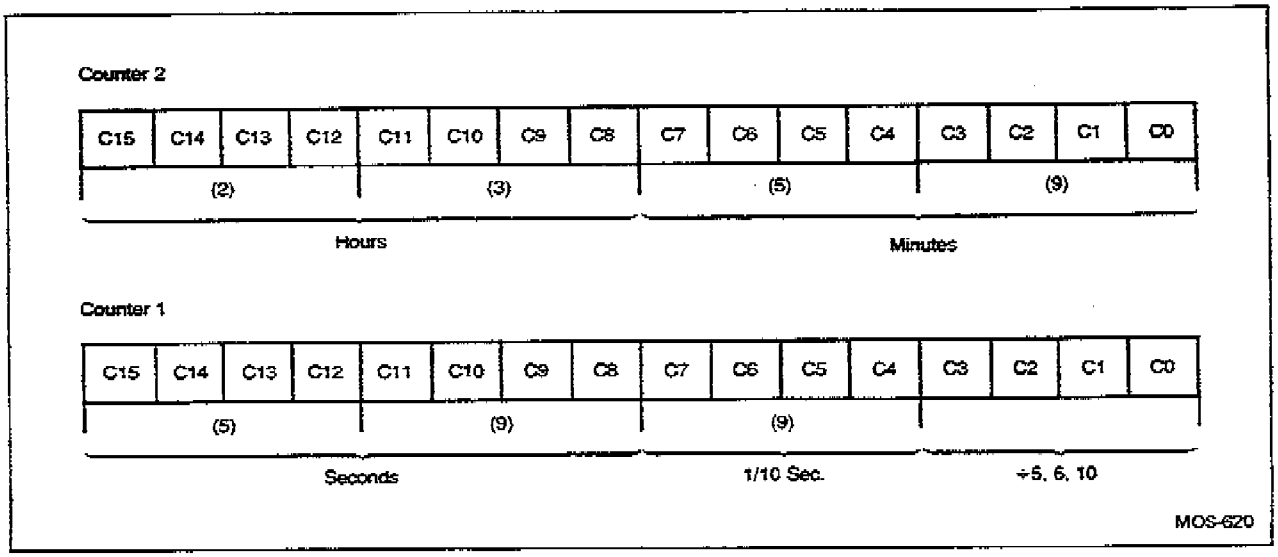

Figure 4-1. Time-of-Day Configuration

MM1	MM0	Configuration
0	0	TOD disabled
0	1	TOD, 50Hz input
1	0	TOD, 60Hz input
1	1	TOD, 100Hz input

Figure 4-2. Time-of-Day Control Options

can be used in Time-of-Day mode. For general purpose time-keeping, the Count Control field should be set to disable special gate; reload from Load; count up; BCD counting; and count repetitively. To generate special functions, the user may select other settings for these bits with the exception of the BCD bit. Time-of-Day counting requires that the counter be set for BCD operation. The Output Control field setting does not affect the Time-of-Day operation since the Counter 1 output will be internally concatenated to the Counter 2 input.

Counter 2's Mode register should be set to the same as Counter 1's with the exception that the Count Source Selection field for Counter 2 should be set to active-high counting from TCN−1, to enable internal concatenation.

The second step is to initialize Counters 1 and 2 to a value of all zeros. This is done by setting Load registers 1 and 2 to 0 and transferring their contents into Counters 1 and 2. Loading the counter with zeros conditions the Time-of-Day count circuitry and must follow the setting of the Master Mode register and Counter Mode registers. If auto-sequencing is being used, each counter's Load register can be set to zero after the Counter's Mode register is loaded. It is important, however, that the Master Mode register be loaded first, and that the Counter Mode registers be loaded before zeros are transferred into the counter. Note that the Master Mode register contents may be changed after this step, providing MM0 and MM1 are not altered; if these are changed, the Time-of-Day circuitry must be reconditioned.

Step 3 of the initialization process involves setting Load registers 1 and 2 to the current time and transferring this into the counters. The format used for the time is that given in Figure 4-1. The user must ensure that the time loaded does not set any of the decades to an illogical value. In particular, no decades should be set to A (hex) through F (hex); the high order decade of Counter 1 and the next-to-least-significant decade in Counter 2 should be 0 through 5; the two high order decades of Counter 2 should be 00 through 23 (BCD); and the low order decade of Counter 1 should be 0. Note also that the Am9513 uses a 24 hour clock, so PM times on a 12 hour clock must have a 12 hour bias added.

The fourth step of the initialization process is to set Load registers 1 and 2 to all 0s. Counter 1 generates a TC and reloads itself from the Load register on the count source edge after reaching 59.99 seconds (100Hz input assumed). Similarly Counter 2 generates a TC on the count source edge after 23:59. By setting the Load registers to 0, the user ensures that the counter will "roll over" to the correct time at TC. For example, a time of 23:59:59.99 will roll over to 00:00:00.00 at midnight.

The final initialization step is to start the counters by writing the "Arm Counters 1 and 2" command (FF23 hex) to the Am9513's Command register. In most Time-of-Day applications, no significant error will be introduced by the delay from the entry of the current time into the counters to the Arming of the counters. This is partly due to the maximum resolution of 10 milliseconds in the Time-of-Day circuitry. In high precision applications, which may use a frequency prescaler for added precision, the user may wish to load a time somewhat later than the current time and delay arming the counter until the current time matches the loaded time.

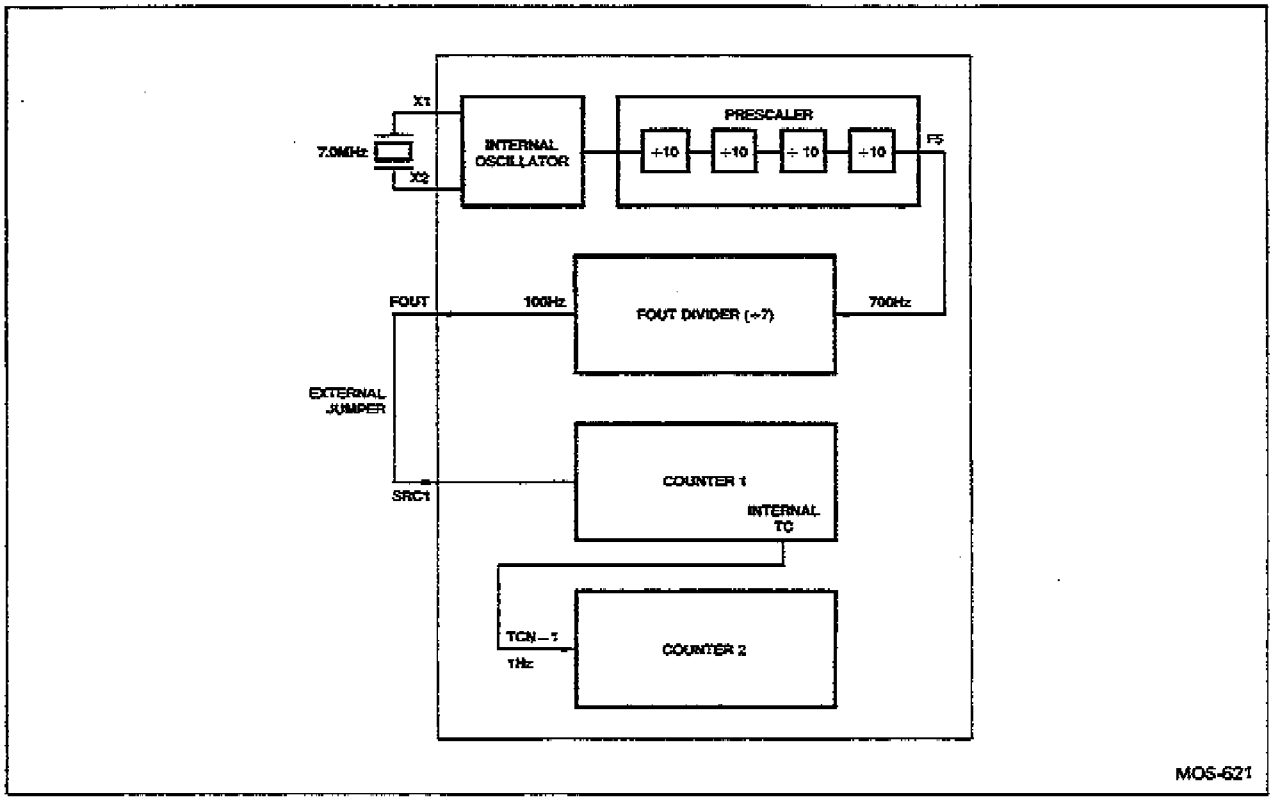

Figure 4-3. Time-of-Day Counting with a 7MHz Crystal

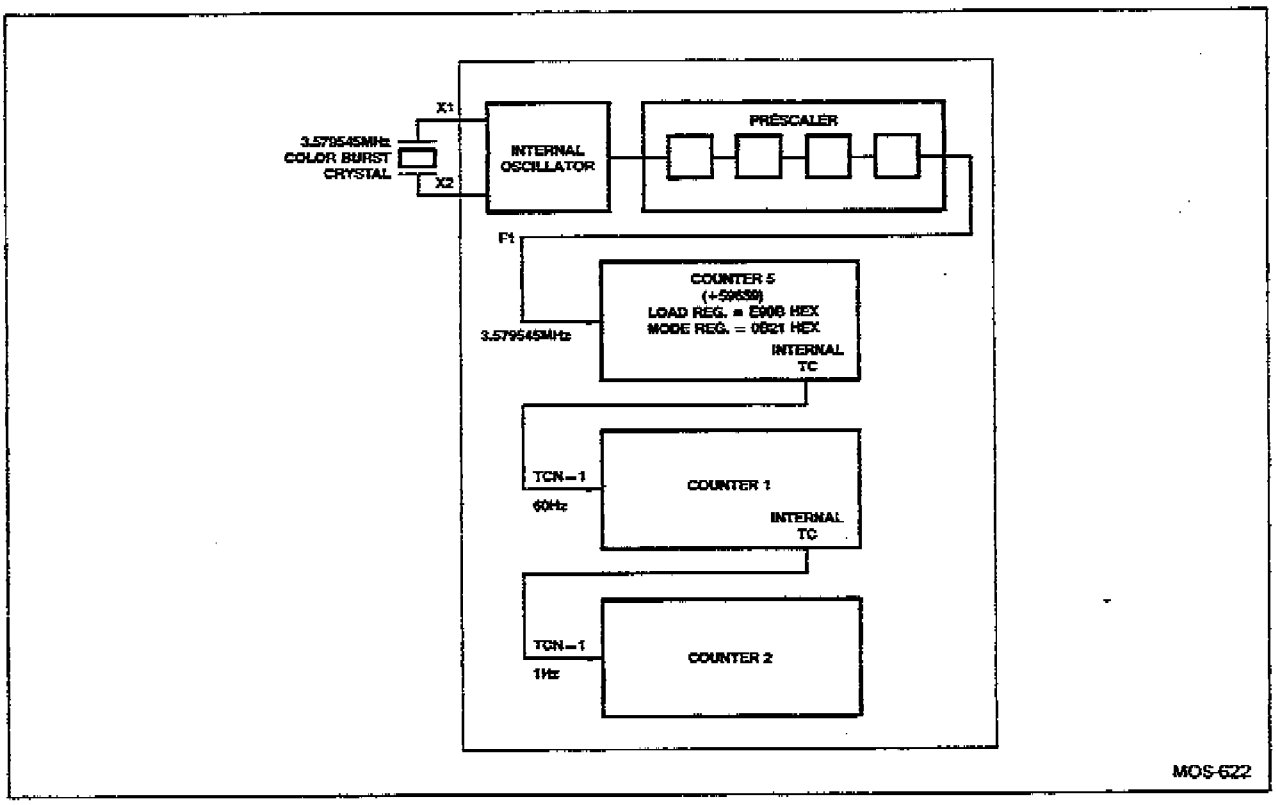

Figure 4-4. Time-of-Day Counting with a Color Burst Crystal

Reading the Current Time

The user may read the current time from the Am9513 by issuing the "Save Counters 1 and 2" command (FFA3 hex) to the Am9513. This causes the contents of Counters 1 and 2 to be transferred into Hold registers 1 and 2. If the user were to now read the Hold register contents, an incorrect time may be read. This is because although Counters 1 and 2 are both synchronous counters, the TC output from Counter 1 ripples through to increment Counter 2. If the save operation occurred just after Counter 1 incremented but before Counter 2 incremented, the value stored in Hold register 2 would be in error. (This consideration should not be confused with the somewhat different points discussed in Appendix A). The user can easily protect against this by examining the contents of Hold register 1. If they are 0, a TC may have just occurred and Hold register 2 may have an erroneous value. Accordingly, if Hold register 1 contains 0, the "Save Counter 2" command (FFA2 hex) should be executed to resave Counter 2. By the time this test for 0 is performed, and Counter 2 is resaved, any rippling carry will have propagated through Counter 2.

Setting the Alarm Time

Master Mode register bits MM2 and MM3 control the Comparators associated with Counters 1 and 2. When a Comparator is enabled, its output is substituted for the normal counter output on the associated OUT1 and OUT2 pins. The polarity definition for the Comparator output will depend on the active-high or active-low definition as programmed in the appropriate Counter Mode register. Once the Comparator output is true, it will remain so until the count changes and the comparison therefore goes false.

The two Comparators can always be used individually in any operating mode. One special case occurs when the time-of-day option is involved and both Comparators are enabled. The operation of Comparator 2 will then be conditioned by Comparator 1 so that a full 32-bit comparison must be true in order to generate a true signal on OUT2. OUT1 will continue, as usual, to reflect the state of the 16-bit comparison between Alarm register 1 and Counter 1.

In some systems, the Alarm output on pin OUT2 might be used to generate an interrupt request to the host CPU. When changing the Alarm register values on a system such as this, the Alarm interrupt should be ignored by the CPU while the Alarm registers are being reloaded, since spurious comparisons may be generated during the reloading process.

Other Time-of-Day Variations

So far this discussion has assumed that Counters 1 and 2 are operated together. The user may, however, elect to drive Counter 2 from a source other than the Counter 1 output, to permit independent accumulation of seconds (Counter 1) and hours/minutes (Counter 2). Note that although Counters 1 and 2 may be independently operated, the Time-of-Day enable bits (MM0 and MM1) are such that either both or neither of the two counters are in Time-of-Day mode; the user cannot set only one counter for Time-of-Day counting.

Separating the two counters would permit Counter 2 to keep track of hours and minutes in real-time, and Counter 1 to time events up to 60 seconds long. The TC output from Counter 1 might perhaps be used to interrupt the CPU. Note that in normal, concatenated Time-of-Day operation, the Counter 1 TC may be used to generate interrupts every minute, a feature useful in real-time

scheduling applications. Other adjustments can be made to enhance the usefulness of Counter 1. For example, with a 100Hz input rate specified for Counter 1, a 1kHz input would provide interrupts at six-second intervals. Similarly, a 6kHz input would provide interrupts every second, etc.

Counting Down in Time-of-Day Mode

The Am9513 allows the user to count down in Time-of-Day mode by setting bit CM3 in the Counter 1 and 2 Mode registers. A few other changes are necessary from counting-up to ensure correct operation.

When initializing the counters for count up operation, they were first loaded with zeros to condition the time-of-day circuitry. To condition the Time-of-Day circuitry for count down applications, instead of zeros, Counter 1 should be loaded with 59.94, 59.95 or 59.99 for 50Hz, 60Hz, or 100Hz input frequencies respectively, and Counter 2 should be loaded with 23.59. The current time should then be loaded into the counters and the Load registers should be set to all zeros to ensure proper roll over.

In counting up, the leading edge of the low order counter's TC is used as a carry to increment the high order counter. When counting down a borrow, rather than a carry signal is desired. Hence, the trailing edge of TC should be used to decrement Counter 2. When internal concatenation is used (Counter 2 Mode register bits CM11-CM8 = 0000) Counter 2 should be set to count on the falling source edge (Counter 2 Mode register bit CM12 = 1). Figure 4-5 shows carry/borrow propagation for up and down time-of-day counting.

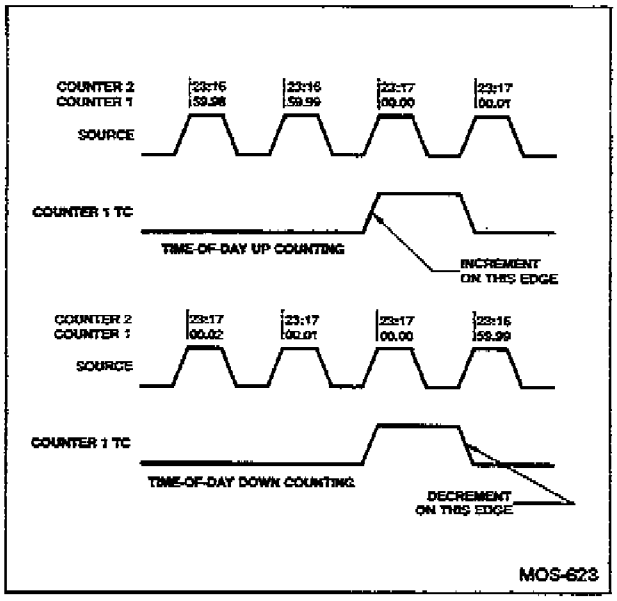

Figure 4-5. Timing Waveforms for Time-of-Day Counting

As with counting up, an incorrect value can be read from the counters if a Save operation is performed while a carry/borrow is rippling between Counters 1 and 2. In count up mode, erroneous readings are prevented by resaving Counter 2 when the value read from Counter 1 was 0. In count down, Counter 2 should be resaved if the value read from Counter 1 indicates a borrow was just generated. This value depends on the input scaler used. For 50Hz, it is 59.94; for 60Hz it is 59.95; for 100Hz it is 59.99. For general-purpose software routines the user can mask out the last decade and resave Counter 2 whenever the upper 3 decades read from Counter 1 are 59.9. As with counting up, the purpose of the resave operation is to ensure that the rippling carry/borrow signal has propagated through Counter 2 and Counter 2's contents have settled.

Accelerated Time

In some applications, such as accelerated life cycle analysis or non-real-time simulations, it is useful to have Time-of-Day accumulations occur at accelerated or decelerated rates. The Time-of-Day circuitry on the Am9513 is capable of operating at the full clock rate permissible. For example, with a 6MHz input rate to Counter 1 and with the 60Hz, Time-of-Day prescaler selected (Master Mode register bits MM1-MM0 = 10), Time-of-Day accumulation would proceed 10^5 times faster than real time.

Am8080A/8085A TIME-OF-DAY SOFTWARE

Figure 4-6 provides sample Am8080A/8085A software listings of general-purpose Time-of-Day routines useful in setting and reading the current time and setting the Alarm time. Each of the basic operations is coded as a subroutine. The user should save those Am8085A registers containing valuable data prior to calling these routines, since some registers are overwritten.

Subroutine SETTIME initializes the Master Mode register and Counter Mode registers 1 and 2 and then sets the counters to the current time. The Master Mode register is set to D00F (hex), resulting in: BCD prescaling; Data Pointer autoincrement disabled; 8-bit bus interface; FOUT off; Comparators 1 and 2 enabled; and Time-of-Day enabled for a 100Hz input. This Master Mode register configuration is representative of a system using a 1MHz crystal on X1 and X2; since BCD prescaling is used, F5 can drive counter 1 with the required 100Hz input signal.

Each Counter Mode register is set for BCD repetitive up counting, with no gating and the special gate disabled. Reloads are set to occur from the Load register only. Counter 1 uses F5 as a count source and Counter 2 uses internal (TCN−1) concatenation. Counter 1's output is set to the high-impedance state. If the comparators had been disabled, a TC output could have been selected for Counter 1 to generate interrupts every minute. (Recall that when the comparators are enabled, the normal TC or toggle output is disabled.) Accordingly, in this sample configuration OUT1, if enabled, would reflect the output of Comparator 1. Counter 2's output is set to active-high TC, but since both Comparators 1 and 2 are enabled in Time-of-Day mode, the output will indicate when a 32-bit match occurs.

Following Counter Mode register initialization, Counters 1 and 2 are loaded with zero to condition the Time-of-Day count circuitry. The current time, stored in variable TIME is then loaded into the counters. Note the data format used for TIME is hours, minutes, seconds and tenths-of-seconds. Finally, the Load registers are reset to zero and the counter is armed.

The second subroutine, ALARMSET, sets the Counter 1 and 2 Alarm registers to the Alarm time stored in variable ALARM. The counter outputs may generate spurious compares while the Alarm registers are being loaded; the subroutine assumes the desired Alarm time has been stored in variable ALARM with the same format used for variable TIME. The subroutine also assumes that Master Mode register bits MM2 and MM3 have been previously set to '1' to enable the comparator circuitry.

Subroutine READTIME reads the time from Counters 1 and 2 and stores it in variable TIME. As discussed previously, special precautions are required when reading the time to avoid saving erroneous values generated by the ripple carry. The subroutine provides the required safeguards.

```
 1:   0100                        ORG 100H
 2:                               ;
 3:                               ;EQU'S FOR AM9513 TIME-OF-DAY
 4:                               ;
 5:   0012 =          CMDPRT  EQU     12H     ;COMMAND PORT
 6:   0010 =          DATAPRT EQU     10H     ;DATA PORT
 7:                               ;
 8:                               ;
 9:                               ;
10:                               ;SUBROUTINE SETTIME SETS CURRENT
11:                               ;TIME USING DATA STORED IN VARIABLE TIME
12:                    SETTIME:
13:                               ;SET MASTER MODE REGISTER
14:                               ;
15:   0100 3E17                   MVI A,017H      ;SET DATA POINTER TO
16:   0102 D312                   OUT CMDPRT      ;MASTER MODE REGISTER
17:   0104 3E0F                   MVI A,0FH       ;LOWER BYTE
18:   0106 D310                   OUT DATAPRT
19:   0108 3ED0                   MVI A,0D0H      ;UPPER BYTE
20:   010A D310                   OUT DATAPRT
21:                               ;
22:                               ;SET COUNTER 1 MODE REG.
23:                               ;
24:   010C 3E01                   MVI A,01H       ;SET DATA POINTER TO
25:   010E D312                   OUT CMDPRT      ;COUNTER 1 MODE REG
26:   0110 3E3C                   MVI A,03CH      ;LOWER BYTE
27:   0112 D310                   OUT DATAPRT
28:   0114 3E0F                   MVI A,0FH       ;UPPER BYTE
29:   0116 D310                   OUT DATAPRT
30:                               ;
31:                               ;SET COUNTER 2 MODE REG.
32:                               ;
33:   0118 3E02                   MVI A,02H       ;SET DATA POINTER TO
34:   011A D312                   OUT CMDPRT      ;COUNTER 2 MODE REG
35:   011C 3E39                   MVI A,039H      ;LOWER BYTE
36:   011E D310                   OUT DATAPRT
37:   0120 3E00                   MVI A,0H        ;UPPER BYTE
38:   0122 D310                   OUT DATAPRT
39:                               ;
40:                               ;CLEAR COUNTERS 1 AND 2
41:                               ;
42:   0124 3E09                   MVI A,09H       ;SET DATA POINTER TO
43:   0126 D312                   OUT CMDPRT      ;LOAD REG 1
44:   0128 3E00                   MVI A,0H
45:   012A D310                   OUT DATAPRT     ;LOWER BYTE
46:   012C D310                   OUT DATAPRT     ;UPPER BYTE
47:                               ;
48:   012E 3E0A                   MVI A,0AH       ;SET DATA POINTER TO
49:   0130 D312                   OUT CMDPRT      ;LOAD REG 2
50:   0132 3E00                   MVI A,0H
51:   0134 D310                   OUT DATAPRT     ;LOWER BYTE
52:   0136 D310                   OUT DATAPRT     ;UPPER BYTE
53:                               ;
54:   0138 3E43                   MVI A,043H      ;TRANSFER CONTENETS OF LOAD
55:   013A D312                   OUT CMDPRT      ;REGS 1 AND 2 INTO COUNTERS
56:                               ;
57:                               ;SET COUNTERS 1 AND 2 TO CURRENT TIME
58:                               ;
```

Figure 4-6. Sample 8080/8085 Time-of-Day Subroutines

```
 59:   013C 3E09              MVI A,09H       ;SET DATA POINTER TO
 60:   013E D312              OUT CMDPRT      ;COUNTER 1 LOAD REG.
 61:   0140 3A7801            LDA TIME+3      ;TENTHS-OF-SECONDS
 62:   0143 D310              OUT DATAPRT
 63:   0145 3A7701            LDA TIME+2      ;SECONDS
 64:   0148 D310              OUT DATAPRT
 65:                         ;
 66:   014A 3E0A              MVI A,0AH       ;SET DATA POINTER TO
 67:   014C D312              OUT CMDPRT      ;COUNTER 2 LOAD REG
 68:   014E 3A7601            LDA TIME+1      ;MINUTES
 69:   0151 D310              OUT DATAPRT
 70:   0153 3A7501            LDA TIME        ;HOURS
 71:   0156 D310              OUT DATAPRT
 72:                         ;
 73:   0158 3E43              MVI A,043H      ;TRANSFER CONTENTS OF LOAD
 74:   015A D312              OUT CMDPRT      ;REGS 1 AND 2 INTO COUNTERS
 75:                         ;
 76:                         ;SET LOAD REG 1 AND 2 TO 0
 77:                         ;
 78:   015C 3E09              MVI A,09H       ;SET DATA POINTER TO
 79:   015E D312              OUT CMDPRT      ;LOAD REG 1
 80:   0160 3E00              MVI A,0H
 81:   0162 D310              OUT DATAPRT     ;LOWER BYTE
 82:   0164 D310              OUT DATAPRT     ;UPPER BYTE
 83:                         ;
 84:   0166 3E0A              MVI A,0AH       ;SET DATA POINTER TO
 85:   0168 D312              OUT CMDPRT      ;LOAD REG 2
 86:   016A 3E00              MVI A,0H
 87:   016C D310              OUT DATAPRT     ;LOWER BYTE
 88:   016E D310              OUT DATAPRT     ;UPPER BYTE
 89:                         ;
 90:                         ;ARM COUNTERS 1 AND 2 TO START COUNT
 91:                         ;
 92:   0170 3E23              MVI A,023H      ;ARM 1 AND 2 COMMAND
 93:   0172 D312              OUT CMDPRT
 94:                         ;
 95:   0174 C9                RET
 96:                         ;
 97:                         ;
 98:   0175 00000000  TIME:   DB      0,0,0,0 ;CURRENT TIME, IN FORMAT
 99:                                         ;HOURS, MINUTES, SECONDS,
100:                                         ;TENTHS-OF-SECONDS. NOTE LOWER
101:                                         ;DECADE OF TENTHS-OF-SECONDS DIGIT
102:                                         ;MUST BE 0.
103:   0179 00000000  ALARM:  DB      0,0,0,0 ;ALARM TIME, SAME FORMAT
104:                                         ;AS TIME.
105:                         ;
106:                         ;
107:                 ALARMSET:               ;SETS ALARM TO VALUE
108:                                         ;STORED IN VARIABLE ALARM
109:                         ;
110:   017D 3E07              MVI A,07H       ;SET DATA POINTER TO
111:   017F D312              OUT CMDPRT      ;ALARM REG 1
112:   0181 3A7C01            LDA ALARM+3     ;TENTHS-OF-SECONDS
113:   0184 D310              OUT DATAPRT
114:   0186 3A7B01            LDA ALARM+2     ;SECONDS
115:   0189 D310              OUT DATAPRT
116:                         ;
117:   018B 3E0F              MVI A,0FH       ;SET DATA POINTER TO
```

Figure 4-6. Sample 8080/8085 Time-of-Day Subroutines (Cont.)

```
118:    018D D312              OUT CMDPRT       ;ALARM REG 2
119:    018F 3A7A01            LDA ALARM+1      ;MINUTES
120:    0192 D310              OUT DATAPRT
121:    0194 3A7901            LDA ALARM        ;HOURS
122:    0197 D310              OUT DATAPRT
123:    0199 C9                RET
124:                           ;
125:                           ;
126:                           ;
127:                 READTIME:       ;READS CURRENT TIME AND
128:                                 ;STORES IT IN VARIABLE TIME
129:                           ;
130:    019A 3EA3              MVI A,0A3H       ;SAVE COUNTERS 1 AND 2
131:    019C D312              OUT CMDPRT
132:                           ;
133:    019E 3E11              MVI A,011H       ;SET DATA POINTER TO
134:    01A0 D312              OUT CMDPRT       ;COUNTER 1 HOLD REG.
135:    01A2 DB10              IN DATAPRT       ;TENTHS-OF-SECONDS
136:    01A4 47                MOV B,A          ;SAVE IN B REG FOR LATER
137:    01A5 327801            STA TIME+3
138:    01A8 DB10              IN DATAPRT       ;SECONDS
139:    01AA 327701            STA TIME+2
140:    01AD A7                ANA A            ;TEST FOR 0
141:    01AE C2BA01            JNZ CTR2IN
142:    01B1 78                MOV A,B          ;GET TENTHS-OF-SECONDS
143:    01B2 A7                ANA A            ;TEST FOR 0
144:    01B3 C2BA01            JNZ CTR2IN
145:                           ;
146:                           ;COUNTER 1 HAD 0, SAVE CTR 2 AGAIN
147:                           ;
148:    01B6 3EA1              MVI A,0A1H       ;SAVE CTR 2 COMMAND
149:    01B8 D312              OUT CMDPRT
150:                           ;
151:    01BA 3E12        CTR2IN: MVI A,12H       ;SET DATA POINTER TO
152:    01BC D312              OUT CMDPRT       ;COUNTER 2 HOLD REG
153:    01BE DB10              IN DATAPRT       ;MINUTES
154:    01C0 327601            STA TIME+1
155:    01C3 DB10              IN DATAPRT       ;HOURS
156:    01C5 327501            STA TIME
157:                           ;
158:    01C8 C9                RET
159:                           ;
160:    01C9                   END
```

Figure 4-6. Sample 8080/8085 Time-of-Day Subroutines (Cont.)

A COOKBOOK APPROACH TO TIME-OF-DAY COUNTING

The following steps are given as a simple, easy-to-use guide to the operation of the Time-of-Day Software. At the end of each step is a reference to the line number appearing along the left side of the program listing in Figure 4-6. Note that these programs have been coded to maximize clarity. Use of automatic Data Pointer sequencing and special programming tricks would likely result in more compact code. The program assumes that the Am9513's Command port is located at address 12 (hex) and that the Data port is at address 10 (hex).

SETTING THE CURRENT TIME

Subroutine SETTIME in the software listing sets the current time and configures the Master Mode register and Counter Mode register 1 and 2.

1. Set the Master Mode register as follows:

   a) Set the Data Pointer register to point to the Master Mode register by writing 17 (hex) to the Am9513 Command port (Lines 15-16).

b) Write XXXX AA11 (binary) to the Am9513 Data port to set the low order byte of the Master Mode register (Lines 17-18).

X = don't care bit
A = set to '1' for Alarm function; set to '0' if Alarm will not be used.

c) Write 110X XXXX (binary) to the Am9513 Data port to set the high order byte of the Master Mode register (Lines 19-20).

2. Set Counter 1's Mode register as follows:
   a) Set the Data Pointer to the Counter 1 Mode register by writing 01 (hex) to the Command port (Lines 24-25).
   b) Write 3C (hex) to the Data port. This is the lower byte of the Counter 1 Mode register (Lines 26-27).
   c) Write 0F (hex) to the Data port. This is the upper byte of the Counter 1 Mode register (Lines 28-29).
3. Set Counter 2's Mode register as follows:
   a) Set the Data Pointer to Counter 2's Mode register by writing 02 (hex) to the Command port (Lines 33-34).
   b) Set the low byte of Counter 2's Mode register by writing 39 (hex) to the Am9513 Data port (Lines 35-36).
   c) Write 00 (hex) to the Data port to set the upper byte (Lines 37-38).
4. Counters 1 and 2 must now be loaded with 0 to condition the Time-of-Day circuitry. Set Load register 1 to 0 as follows:
   a) Set the Data Pointer register to point to Load register 1 by writing 09 (hex) to the Am9513 Command port (Lines 42-43).
   b) Write 00 (hex) to the Data port to set the lower byte of the Load register (Lines 44-45).
   c) Write 00 (hex) to the Data port to set the upper byte (Line 46).
5. Set Load register 2 to 0 as follows:
   a) Set the Data Pointer register to point to Load register 2 by writing 0A (hex) to the Am9513 Command port (Lines 48-49).
   b) Write 00 (hex) to the Data port to set the lower byte of the Load register (Lines 50-51).
   c) Write 00 (hex) to the Data port to set the upper byte (Line 52).
6. Transfer the contents of Load registers 1 and 2 into Counters 1 and 2 by writing the "Load Counters 1 and 2" command (43 hex) to the Am9513 Command port (Lines 54-55).
7. Set the current time's seconds and tenths-of-seconds into Load register 1 as follows:
   a) Set the Data Pointer register to point to Counter 1's Load register by writing 09 (hex) to the Am9513's Command port (Lines 59-60).
   b) Write the current time's tenths-of-seconds to the Data port in the format T0 (hex), where T is a BCD number between 0 to 9 representing tenths of seconds. The low decade, used by the Time-of-Day prescaler, will usually be set to 0. (If a 100Hz input frequency is selected the low decade may be set to the current time's tens-of-milliseconds value.) (Lines 61-62)
   c) Write the current time's seconds to the Data port. This should be a number between 0 and 59 (decimal) in BCD format (Lines 63-64).
8. Set the current time's hours and minutes, using a 24 hour clock, into Load register 2 as follows:
   a) Set the Data Pointer register to point to Counter 2's Load register by writing 0A (hex) to the Am9513's Command port (Lines 66-67).
   b) Write the current time's minutes to the Data port. This number must be between 00 and 59 (decimal) in BCD format (Lines 68-69).
   c) Write the current time's hours to the Data port. This number must be between 00 and 23 (decimal) in BCD format (Lines 70-71).
9. Load Counters 1 and 2 with the current time by writing the "Load Counters 1 and 2" command (43 hex) to the Am9513's Command port (Lines 73-74).
10. Set Load register 1 to 0 by repeating Step 4 (Lines 78-82).
11. Set Load register 2 to 0 by repeating Step 5 (Lines 84-88).
12. Start the counters by writing the "Arm Counters 1 and 2" command (23 hex) to the Am9513's Command port (Lines 92-93).

SETTING THE ALARM TIME

1. Set Alarm register 1 to the seconds and tenths-of-seconds Alarm time as follows:
   a) Set the Data Pointer register to point to Alarm register 1 by writing 07 (hex) to the Am9513's Command port (Lines 110-111).
   b) Write the tenths-of-seconds Alarm time to the Am9513's Data port in the format T0 (hex) where T is a number between 0 and 9 in BCD format representing tenths-of-seconds. The lower decade will usually be set to 0. (When a 100Hz input frequency is selected, the lower decade may be set to the Alarm times tens-of-milliseconds.) (Lines 112-113)
   c) Write the seconds component of the Alarm time to the Data port. This should be a BCD number of 0 through 59 (decimal) (Lines 114-115).
2. Set Alarm register 2 to the hours and minutes Alarm time as follows:
   a) Set the Data Pointer register to point to Alarm register 2 by writing 0F (hex) to the Am9513's command port (Lines 117-118).
   b) Write the minute component of the Alarm time to the Data port. This should be a BCD encoded number of value 0 through 59 (decimal) (Lines 119-120).
   c) Write the hours component of the Alarm time to the Data port. This should be a BCD encoded number of value 0 through 23 (decimal) (Lines 121-122).
3. If Master Mode register bits MM2 and MM3 are not already both 1, set them to 1 now as follows: (These steps are not shown in the software listing.)
   a) Set the Data Pointer register to point to the Master Mode register by writing 17 (hex) to the Am9513's Command port.
   b) Read the lower Master Mode register byte.
   c) Perform a logical OR between this byte and 0C (hex). This sets bits 2 and 3 to 1.
   d) Repeat step a) to reset the Data Pointer register.
   e) Write the resultant byte from step c) to the Master Mode register. This updates the lower byte of the register.

READING THE CURRENT TIME

1. Save the contents of Counters 1 and 2 in Hold registers 1 and 2 by writing the "Save Counters 1 and 2" command (A3 hex) to the Am9513's Command port (Lines 130-131).

2. Read the value saved in Hold register 1 as follows:
   a) Set the Data Pointer register to point to Counter 1's Hold register by writing 11 (hex) to the Am9513's Command port (Lines 133-134).
   b) Read from the Data port to retrieve the low order Hold register byte. The upper decade contains the tenths-of-seconds component of the current time. The lower decade is used by the Time-of-Day prescaler. For 100Hz prescaling, it is tens-of-milliseconds (Lines 135-137).
   c) Read from the Data port again to retrieve the high order byte. This is the seconds component of the current time (Lines 138-139).
3. If the value read from Hold register 1 is 0000, Save Counter 2 again. Otherwise, go to Step 4. Counter 2 may be resaved by writing A2 (hex) to the Command port (Lines 140-149).
4. Read the contents of Hold register 2 as follows:
   a) Set the Data Pointer register to point to Load register 2 by writing 12 (hex) to the Am9513's Command port (Lines 151-152).
   b) Read the minutes component of the current time by reading a byte from the Data port (Lines 153-154).
   c) Read the hours component of the current time by reading another byte from the Data port (Lines 155-156).

SETTIME SOFTWARE USING MACROS

The programming examples given above can be further simplified by using macros. The listing in Figure 4-7 is a macro-coded version of the subroutine SETTIME for the 8080, 8085 and Z80.* This example provides a graphic illustration of the simplicity of usage of the Am9513 macros. The full output code generation of the macroassembler is shown in Figure 4-8 to illustrate the convenience of using macros. This code expansion is slightly different than the code in Figure 4-6 due to different counter starting points and some software efficiencies.

CONSOLE DRIVEN CLOCK RUNS UNDER CP/M

A further example written in C is presented; it provides a simple console-driven clock running under the CP/M* (ver 2.2) compatible AMDOS® operating system. The resolution of the clock is limited to one-second intervals since the Whitesmiths' C compiler unfortunately generates an I/O overhead of such a size that the loading time of the program makes further accuracy pointless.

The C example shown in Figure 4-9 illustrates the two ways of setting the various modes of operation and register values desired. The Master Mode register (master-reg in the listing) is statically initialized since the various field values are known at compile time. This declaration generates just two bytes of code corresponding to the 16 bits of the Master Mode register.

The two Counter Mode registers are dynamically initialized field by field (refer to the set modes procedure in the listing). Actually, static initialization could be used, since the desired field values are known at compile time (e.g., .base = BCD). For the purposes of illustration, however, dynamic initialization is employed.

The program protects against reading the time while a carry is in progress from counter 1 TC to the counter 2 input. Should the value read from counter 1 be zero then the contents of counter 2 are resaved to avoid possible misreading of the time.

*Z80 is a trademark of Zilog, Inc.
CP/M is a trademark of Digital Research, Inc.
AMDOS is a registered trademark of Advanced Micro Devices, Inc.

```
;       Coded for Am8080/Am8085 - also runs on Z80

        INCLUDE B:EQUS.MAC              ; Listed in Appendix 7
        INCLUDE B:8080.MAC              ; Listed in Appendix 3

SETTIME:
        MASTER          TOD_100HZ,ENABLE,ENABLE,F1,0,OF,BUS_8,OF,BCD
        MODE_REG        1,OFF_OC_TC,UP,BCD,MODE_DEF,F5,RISE,NO_GATE
        MODE_REG        2,ACT_HI_TC,UP,BCD,MODE_DEF,TC_NM1,RISE,NO_GATE
        LOAD_REG        1,0
        LOAD_REG        2,0
        LOAD            1,2             ; Enable TOD counting
        LOAD_REG        1,TIME+2,I
        LOAD_REG        2,TIME,I
        LOAD            1,2             ; Set the current time
        LOAD_REG        1,0
        LOAD_REG        2,0             ; Recall repeat reload values needed
        ARM             1,2             ; Time starts now
        RET

                                ; Current time hrs, min, secs, tenths
TIME:   DB      0,0,0,0
```

Figure 4-7. SETTIME Macro Listing for 8080, 8085 and Z80

```
      - Macro expansion of SETTIME.MAC example -

0100'           SETTIME:

                MASTER  TOD_100HZ,ENABLE,ENABLE,F1,0,CF,BUS_8,CF,BCD

0003              +  DLAB        SET     TOD_100HZ
0007              +  DLAB        SET     DLAB OR (ENABLE SHL 2)
000F              +  DLAB        SET     DLAB OR (ENABLE SHL 3)
00BF              +  DLAB        SET     DLAB OR (F1 SHL 4)
00BF              +  DLAB        SET     DLAB OR (0 SHL 8)
10BF              +  DLAB        SET     DLAB OR (CF SHL 12)
10BF              +  DLAB        SET     DLAB OR (0 SHL 13)
50BF              +  DLAB        SET     DLAB OR (CF SHL 14)
D0BF              +  DLAB        SET     DLAB OR (BCD SHL 15
0100'  3E 17      +              MVI     A,CTRL_GR OR (MASTER_ SHL 3)
0102'  D3 12      +              OUT     A_CTRL
0104'  3E BF      +              MVI     A, LOW DLAB
0106'  D3 10      +              OUT     A_DATA
0108'  3E D0      +              MVI     A, HIGH DLAB
010A'  D3 10                     OUT     A_DATA

                MODE_REG  1,OFF_OC_TC,UP,BCD,MODE_DEF,F5,RISE,NO_GATE

0004              +  DLAB        SET     OFF_OC_TC
000C              +  DLAB        SET     DLAB OR (UP SHL 3)
001C              +  DLAB        SET     DLAB OR (BCD SHL 4)
003C              +  DLAB        SET     DLAB OR (MODE_DEF SHL 5)
0F3C              +  DLAB        SET     DLAB OR (F5 SHL 8)
0F3C              +  DLAB        SET     DLAB OR (RISE SHL 12)
0F3C              +  DLAB        SET     DLAB OR (NO_GATE SHL 13)
010C'  3E 01                     MVI     A,1 OR (MODE_ SHL 3)
010E'  D3 12                     OUT     A_CTRL
0110'  3E 3C                     MVI     A, LOW DLAB
0112'  D3 10                     OUT     A_DATA
0114'  3E 0F                     MVI     A, HIGH DLAB
0116'  D3 10                     OUT     A_DATA

                MODE_REG  2,ACT_HI_TC,UP,BCD,MODE_DEF,TC_NM1,RISE,NO_GATE

0001              +  DLAB        SET     ACT_HI_TC
0009              +  DLAB        SET     DLAB OR (UP SHL 3)
0019              +  DLAB        SET     DLAB OR (BCD SHL 4)
0039              +  DLAB        SET     DLAB OR (MODE_DEF SHL 5)
0039              +  DLAB        SET     DLAB OR (TC_NM1 SHL 8)
0039              +  DLAB        SET     DLAB OR (RISE SHL 12)
0039              +  DLAB        SET     DLAB OR (NO_GATE SHL 13)
0118'  3E 02                     MVI     A,2 OR (MODE_ SHL 3)
011A'  D3 12                     OUT     A_CTRL
011C'  3E 19                     MVI     A, LOW DLAB
011E'  D3 10                     OUT     A_DATA
0120'  3E 00                     MVI     A, HIGH DLAB
0122'  D3 10                     OUT     A_DATA
```

Figure 4-8. Macro Assembler Output

```
                LOAD_REG        1,0

0124'  3E 09                    MVI     A,1 OR (LOAD_ SHL 3)
0126'  D3 12                    OUT     A_CTRL
0128'  3E 00                    MVI     A, LOW 0
012A'  D3 10                    OUT     A_DATA
012C'  3E 00                    MVI     A, HIGH 0
012E'  D3 10                    OUT     A_DATA

                LOAD_REG        2,0

0130'  3E 0A                    MVI     A,2 OR (LOAD_ SHL 3)
0132'  D3 12                    OUT     A_CTRL
0134'  3E 00                    MVI     A, LOW 0
0136'  D3 10                    OUT     A_DATA
0138'  3E 00                    MVI     A, HIGH 0
013A'  D3 10                    OUT     A_DATA

                LOAD            1,2             ; Enable TOD counting

0040              + DLAB        SET     40H
0042              + DLAB        SET     DLAB OR (1 SHL (2-1))
0043              + DLAB        SET     DLAB OR (1 SHL (1-1))
013C'  3E 43                    MVI     A,DLAB
013E'  D3 12                    OUT     A_CTRL

                LOAD_REG        1,TIME+2,I

0140'  3E 09                    MVI     A,1 OR (LOAD_ SHL 3)
0142'  D3 12                    OUT     A_CTRL
0144'  3A 017F'                 LDA     TIME+2
0147'  D3 10                    OUT     A_DATA
0149'  3A 0180'                 LDA     TIME+2+1
014C'  D3 10                    OUT     A_DATA

                LOAD_REG        2,TIME,I

014E'  3E 0A                    MVI     A,2 OR (LOAD_ SHL 3)
0150'  D3 12                    OUT     A_CTRL
0152'  3A 017D'                 LDA     TIME
0155'  D3 10                    OUT     A_DATA
0157'  3A 017E'                 LDA     TIME+1
015A'  D3 10                    OUT     A_DATA

                LOAD            1,2             ; Set the current time

0040              + DLAB        SET     40H
0042              + DLAB        SET     DLAB OR (1 SHL (2-1))
0043              + DLAB        SET     DLAB OR (1 SHL (1-1))
015C'  3E 43                    MVI     A,DLAB
015E'  D3 12                    OUT     A_CTRL
```

Figure 4-8. Macro Assembler Output (Cont.)

```
                LOAD_REG        1,0
0160'   3E 09                  MVI     A,1 OR (LOAD_ SEL 3)
0162'   D3 12                  OUT     A_CTRL
0164'   3E 00                  MVI     A, LOW 0
0166'   D3 10                  OUT     A_DATA
0168'   3E 00                  MVI     A, HIGH 0
016A'   D3 10                  OUT     A_DATA
                LOAD_REG        2,0             ; Recall repeat reload
                                                    values needed
016C'   3E 0A                  MVI     A,2 OR (LOAD_ SEL 3)
016E'   D3 12                  OUT     A_CTRL
0170'   3E 00                  MVI     A, LOW 0
0172'   D3 10                  OUT     A_DATA
0174'   3E 00                  MVI     A, HIGH 0
0176'   D3 10                  OUT     A_DATA

                ARM             1,2             ; Time starts now

0020              +  DLAB       SET     20H
0022              +  DLAB       SET     DLAB OR (1 SHL (2-1))
0023              +  DLAB       SET     DLAB OR (1 SHL (1-1))
0178'   3E 23                  MVI     A,DLAB
017A'   D3 12                  OUT     A_CTRL
017C'   C9                             RET

                                        ; Current time hrs, min,
017D'   00 00 00 00             TIME:   DB      0,0,0,0     secs, tenths

                END SETTIME
Macros:

ARM     CLEAR   DISARM  DPS     DRMSAV  FOUT    HOLD_R  LD_ARM
LOAD    LOAD_R  MASTER  MODE_R  N_TYPE  POINT   RESET   SAVE
SET_    STEP    S_MASK  S_TYPE

Symbols:

ACT_HI  0001    ACT_LO  0005    ALARM1  0000    ALARM2  0001
A_CTRL  0012    A_DATA  0010    BCD     0001    BINARY  0000
BUS_16  0001    BUS_8   0000    CTRL_G  0007    DISABL  0000
DLAB    0023    DOWN    0000    ENABLE  0001    F1      000B
F2      000C    F3      000D    F4      000E    F5      000F
FALL    0001    GATE_1  0006    GATE_2  0007    GATE_3  0008
GATE_4  0009    GATE_5  000A    HE_GAT  0006    HL_GAT  0004
HL_NM1  0003    HL_NP1  0002    HL_TC_  0001    HOLD_   0002
HOLD_C  0003    I       0000    LE_GAT  0007    LL_GAT  0005
LOAD_   0001    MASTER  0002    MODE_   0000    MODE_A  0000
MODE_D  0001    MODE_G  0002    MODE_J  0003    MODE_M  0004
MODE_P  0005    MODE_S  0006    MODE_V  0007    NO_GAT  0000
OF      0001    OFF_LO  0000    OFF_OC  0004    ON      0000
RISE    0000    SETTIM  0100'   SRC_1   0001    SRC_2   0002
SRC_3   0003    SRC_4   0004    SRC_5   0005    STATUS  0003
TC_NM1  0000    TC_TOG  0002    TIME    017D'   TOD_10  0003
TOD_50  0001    TOD_60  0002    TOD_OF  0000    UP      0001
No  Fatal error(s)

A>
```

Figure 4-8. Macro Assembler Output (Cont.)

```
/*      This file provides an example usage of the 9513 System Timing
controller device as a console clock. Whitesmiths' C example.
*/

#include "B:9513DEFS.H"                 /* Listed in Appendix 6 */
#include "B:9513RECS.H"                 /* Listed in Appendix 1 */

/*      Record structure definitions - Whitesmiths' C demands initialisers.

        Notice that the master_reg definition generates only 2 bytes of code,
  as would the counter definitions if they were statically initialised.
*/

master_type     master_reg      /* This is a static initialisation      */

                = {TOD_100HZ,DISABLE,DISABLE,F1,7,OFF,BUS_8,OFF,BCD} ;


channel_type    count[6]        /* array of the 5 counters on the 9513  */

                = {0,0,0,0,0,0,0,0,0,    /* dummy entry - index for count1 is 1 */
                   0,0,0,0,0,0,0,0,0,
                   0,0,0,0,0,0,0,0,0,
                   0,0,0,0,0,0,0,0,0,
                   0,0,0,0,0,0,0,0,0,
                   0,0,0,0,0,0,0,0,0
                   } ;


/*      Command constants       */

load    =       0X43 ;          /* LOAD counters 1 and 2        */
arm     =       0X23 ;          /* ARM counters 1 and 2         */
save    =       0XA3 ;          /* SAVE counters 1 and 2        */


_main ()

BEGIN
        int hrs_, min_ ;
        int temp , ac ;

        ac = getfmt ("%2hi%:%2hi",&hrs_,&min_) ;
        if (ac < 2)
        THEN
                read_time () ;  /* If XX:XX then set time, else read it */
        else
        BEGIN
                set_modes () ;                  /* Mode registers       */
                set_loads (hrs_, min_, 0) ;     /* Load registers       */
                output (CONTROL, &load) ;       /* LOAD command */
                output (CONTROL, &arm) ;        /* ARM command  */
                set_loads (0,0,0) ;     /* Load registers = 0 for next reload */
                read_time () ;                  /* Show if running      */
        END
END
```

Figure 4-9. A Console Driven Clock Written in 'C'

```
set_modes ()

BEGIN
                                /* set master mode reg - recall static init */
        point_to (CTRL_GROUP, MASTER) ;
        output (DATA, &master_reg) ;
                                /* set counter 1 mode - recall dynamic init */
        count[1].mode.output    = OFF_LO_TC ;
        count[1].mode.direction = UP ;
        count[1].mode.base      = BCD ;
        count[1].mode.control   = MODE_DEF ;
        count[1].mode.source    = F5 ;
        count[1].mode.edge      = RISE ;
        count[1].mode.gate      = NC_GATE ;
        point_to (1, MODE_) ;
        output (DATA,&count[1].mode) ;
                                /* set  counter 2 mode  */
        count[2].mode.output    = ACT_HI_TC ;
        count[2].mode.direction = UP ;
        count[2].mode.base      = BCD ;
        count[2].mode.control   = MODE_DEF ;
        count[2].mode.source    = TC_NM1 ;
        count[2].mode.edge      = RISE ;
        count[2].mode.gate      = NC_GATE ;
        point_to (2, MODE_) ;
        output (DATA,&count[2].mode) ;
END


set_loads (hrs, min, secs)

        int hrs, min, secs ;
BEGIN
        count[2].load = (hrs*256) + min ;
        count[1].load = secs * 256 ;
                                /* point and send counter 1 load        */
        point_to (1, LOAD_) ;
        output (DATA,&count[1].load) ;
                                /* point and send counter 2 load        */
        point_to (2, LOAD_) ;
        output (DATA,&count[2].load) ;
END


read_time ()

BEGIN
        unsigned hrs, min, secs, tenths ;

        output (CONTROL, &save) ;       /* Issue the SAVE command       */
        point_to (1, HOLD_) ;
        tenths = in (DATA) ;
        secs = in (DATA) ;              /* Read the Hold 1 register     */
        if ((secs == 0) && (tenths == 0))
        THEN                            /* Wups - missed a carry bit    */
                output (CONTROL, &save) ;
        point_to (2, HOLD_) ;           /* Now read correct Hold 2 reg  */
        min = in (DATA) ;
        hrs = in (DATA) ;
        putfmt ("At the bell it was%3hi:%2hi:%2hi\n",hrs,min,secs) ;
END
```

Figure 4-9. A Console Driven Clock Written in 'C' (Cont.)

```
point_to (channel, reg)

BEGIN
        data_type       data_ptr ;

        data_ptr.element = reg ;
        data_ptr.group = channel ;
        data_ptr.cmnd_code = 0 ;
        output (CONTROL,&data_ptr) ;
END


output (port, word)
        unsigned port, *word ;

BEGIN
        unsigned sent ;         /* This output routine has a Trace dump */

        out (port, (*word % 256)) ;
        if (port == DATA)
        THEN
        BEGIN
                out (port, (*word / 256)) ;
                sent = *word ;
        END
        else
                sent = *word % 256 ;

                                /* Delete the next statement to suppress dump
                                   - it shows what you send to the Am9513  */

/* trace off

        putfmt ("Output %2hi%6hi\n",port, sent) ;

*/

END


A>
```

Figure 4-9. A Console Driven Clock Written in 'C' (Cont.)