## Chapter 11

## System-on-a-Chip – A Case Study

11.1. It is not necessary to initialize the tone timer. Figure 11.6a should then be as follows:

```
#define
                            minute_timer (volatile int *) 0x5000
#define
                             tone_timer (volatile int *) 0x5020
#define
                             sliders (volatile int *) 0x5040
#define
                            pushbuttons (volatile int *) 0x5050
#define
                            display (int *) 0x5060
#define
                            LEDs (int *) 0x5070
#define
                             speaker (int *) 0x5080
#define
                             ADJUST(t, x) ((t + x) >= 1440) ? (t + x - 1440) : (t + x)
int
                             actual_time, alarm_time, alarm_active, time;
   /* Hex to 7-segment conversion table */
    unsigned char table [16] = \{0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x12, 0x02, 0x78, 0x12, 
              0x00, 0x18, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F};
    void DISP(time)
                                                                                                   /* Get 7-segment patterns for display. */
              *display = table[time / 600] << 24
                               table[(time % 600) / 60] << 16 |
                               table[(time % 60) / 10] << 8 |
                               table[(time % 10)];
    }
   main()
              actual_time = alarm_time = alarm_active = 0;
                                                                                                   /* Run in continuous mode. */
              *(tone\_timer + 1) = 0x6;
              *(minute_timer + 1) = 0x6; /* Run in continuous mode. */
              while (1)
                     if (*minute_timer == 3)
                                                                                                   /* One minute elapsed. */
                                                                                                   /* Clear the TO bit. */
                               *minute\_timer = 0;
                               actual_time = ADJUST(actual_time, 1);
                     }
```

11.2. It is not necessary to initialize the tone timer. Figure 11.7b should then be as follows:

	movi stbio	r6, 6 r6, (r3)	/* Turn on the two /* vertical LEDs.	*/ */
	movia movi sthio	r6, tone_timer r7, 6 r7, 4(r6)	/* Start the tone-timer.	*/
	movia	r6, minute_timer	/* Address of minute-timer.	*/
	addi	r7, r0, 7	/* Start the timer.	*/
	sthio movi	r7, 4(r6) r7, 1		
	wrctl	ienable, r7	/* Enable timer interrupts.	*/
	wretl	status, r7	/* Enable external interrupts.	*/
LOOP:	movia	r10, ACTUAL_TIME	/* Display the time of day.	*/
	call	DISP		
	ldbio	r7, (r2)		
	andi	r11, r7, 1	/* Check if alarm switch is on.	*/
	beq	r11, r0, NEXT		
	movi	r11,7	/* If yes, then turn on the	*/
	stbio	r11, (r3)	/* alarm LED.	*/
	movia	r9, ALARM_TIME		
	ldw	r11, (r9)	/* Have to compare alarm-time	*/
	ldw	r12, (r10)	/* with actual-time.	*/
	bne	r11, r12, NEXT	/* Should the alarm ring?	*/
	movia	r8, tone_timer		
DDIG I	movi	r12, 1		
RING_LC		DICD		
	call	DISP		
	ldbio	r7, (r2)	/* Check if alarm switch is on.	*/
	andi beq	r13, r7, 1 r13, r0, NEXT	/* Check II alarm switch is on.	*/
	ldhio	r9, (r8)	/* Read the tone-timer status.	*/
	sthio	r0, (r8)	/* Clear the TO bit.	*/
	andi	r9, r9, 1	/* Check if counter reached 0.	*/
	xor	r12, r9, r12	/* Generate the next square	*/
	movia	r11, speaker	/* wave half-cycle; send	*/
	stbio	r12, (r11)	/* signal to the speaker.	*/
	br	RING_LOOP		

...continued in Part c of Figure 11.7

11.3. If the time is greater then 720, then adjust the display by subtracting 720 and turn on the PM light. The program in Figure 11.6 can be modified as follows:

```
minute_timer (volatile int *) 0x5000
#define
#define
                        tone_timer (volatile int *) 0x5020
#define
                        sliders (volatile int *) 0x5040
#define
                        pushbuttons (volatile int *) 0x5050
                        display (int *) 0x5060
#define
#define
                        LEDs (int *) 0x5070
#define
                        speaker (int *) 0x5080
                        ADJUST(t, x) ((t + x) >= 1440) ? (t + x - 1440) : (t + x)
#define
int
                        actual_time, alarm_time, alarm_active, time, alarm_state;
   /* Hex to 7-segment conversion table */
   unsigned char table [16] = \{0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x12, 0x02, 0x78, 0x12, 
            0x00, 0x18, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F\};
   void initializeToneTimer()
            *(tone_timer + 2) = 0x0D40; /* Set the timeout period
            *(tone\_timer + 3) = 0x03;
                                                                                        /* for continuous operation. */
            *(tone\_timer + 1) = 0x6;
                                                                                        /* Start in continuous mode. */
                                                                                        /* Get 7-segment patterns for display. */
   void DISP(time)
           if (time > 720)
                   time = time -720;
                                                                                         /* It is PM. */
                   *LEDs = (alarm_state \mid 0x8);
            else
                   *LEDs = alarm_state;
                                                                                          /* It is AM. */
            *display = table[time / 600] << 24
                           table[(time % 600) / 60] << 16 |
                           table[(time % 60) / 10] << 8 |
                           table[(time % 10)];
   }
   main()
            actual_time = alarm_time = alarm_active = 0;
            initializeToneTimer();
                                                                                         /* Run in continuous mode. */
            *(minute_timer + 1) = 0x6;
            while (1)
                                                                                        /* One minute elapsed. */
                  if (*minute_timer == 3)
                                                                                         /* Clear the TO bit. */
                           *minute_timer = 0;
                           actual_time = ADJUST(actual_time, 1);
```

```
if ((*sliders \& 1) != 0)
                                        /* Check the alarm-on switch. */
                                         /* Turn on the alarm LED. */
              alarm\_state = 7;
              if (actual_time == alarm_time)
                   alarm\_active = 1;
                                        /* Start the alarm sound. */
              else
                   alarm_active = alarm_active & (*sliders & 1);
                                         /* Generate the square wave. */
              if (*tone_timer == 3)
                   *speaker = (*speaker ^{\wedge} 1) & alarm_active;
                                         /* Clear the TO bit. */
                   *tone\_timer = 0:
       }
       else
                                         /* Turn off the alarm LED. */
              alarm_state = 6;
              alarm\_active = 0;
       if ((*sliders & 4) != 0)
                                         /* Check the set-the-time-of-day switch. */
              DISP(actual_time);
                                        /* Display the time of day. */
              if ((*(pushbuttons + 3) \& 1) != 0) /* Set the minutes? */
                   actual_time = ADJUST(actual_time, 1);
              else if ((*(pushbuttons + 3) & 2) != 0) /* Set the hours? */
                   actual_time = ADJUST(actual_time, 60);
              *(pushbuttons + 3) = 0; /* Clear the edge-capture register. */
                                        /* Check the set-the-alarm-time switch. */
       else if ((*sliders & 2) != 0)
              DISP(alarm_time);
                                         /* Display the alarm time. */
              if ((*(pushbuttons + 3) & 1) != 0) /* Set the minutes? */
                   alarm_time = ADJUST(alarm_time, 1);
              else if ((*(pushbuttons + 3) & 2) != 0) /* Set the hours? */
                   alarm_time = ADJUST(alarm_time, 60);
              *(pushbuttons + 3) = 0; /* Clear the edge-capture register. */
       }
       else
                                         /* Display the time of day. */
              DISP(actual_time);
}
```

11.4. If the time is greater then 720, then adjust the display by subtracting 720 and turn on the PM light. The program in Figure 11.7 can be modified as follows:

```
minute_timer, 0x05000
         .equ
         .equ
                  tone_timer, 0x5020
                  sliders, 0x5040
         .equ
                  pushbuttons, 0x5050
         .equ
                  display, 0x5060
         .equ
         .equ
                  LEDs, 0x05070
         .equ
                  speaker, 0x5080
         .equ
                  ACTUAL_TIME, 0x1000
                  ALARM_TIME, 0x1010
         .equ
                  ALARM_STATE, 0x1020
         .equ
                  STACK, 0x2000
         .equ
_start:
         br
                  MAIN
         Interrupt handler
         .org
                  0x20
         subi
                                        /* Save registers.
                  sp, sp, 8
         stw
                  r^{2}, 0(sp)
         stw
                  ra, 4(sp)
         rdctl
                  et, ipending
                  et, r0, MAIN
                                        /* Error if not an external
                                                                            */
         beq
                                                                            */
                                        /* interrupt, treat as reset.
                                        /* Decrement ea to execute the
                                                                            */
         subi
                  ea, ea, 4
                                        /* interrupted instruction upon
                                                                            */
                                        /* return to the main program.
                                                                            */
                  r2, minute_timer
                                        /* Clear the TO bit in the
                                                                            */
         movia
                                                                            */
                                        /* minute-timer.
         sthio
                  r0, (r2)
                  UPDATE_TIME
                                        /* Call interrupt-service routine.
                                                                            */
         call
                                        /* Restore registers.
         ldw
                  r2, 0(sp)
         ldw
                  ra, 4(sp)
         addi
                  sp, sp, 8
         eret
         Main program
                                                                            */
MAIN:
         movia
                  sp, STACK
                                        /* Set up the stack pointer.
                                                                            */
                  r2, ALARM_TIME
                                        /* Clear the alarm-time buffer.
                                                                            */
         movia
         stw
                  r0, (r2)
                  r2, ACTUAL_TIME
                                        /* Clear the actual-time buffer.
         movia
                  r0, (r2)
         stw
                                        /* Address of slider switches.
                  r2, sliders
                                                                            */
         movia
         movia
                  r3, LEDs
                                        /* Address of LEDs.
                                                                            */
                  r4, display
                                        /* Address of 7-segment displays.
                                                                            */
         movia
                 r5, pushbuttons
                                        /* Address of pushbuttons.
                                                                            */
         movia
```

```
movi
                   r6, 6
                                                /* Turn on the two
                                                                                  */
                                                /* vertical LEDs.
                                                                                  */
          stbio
                   r6, (r3)
                   r6, tone_timer
          movia
                                                /* Set the tone-timer period.
                                                                                  */
          ori
                   r7, r0, 0x0D40
          sthio
                   r7, 8(r6)
                   r7, r0, 0x03
          ori
          sthio
                   r7, 12(r6)
                                                /* Start the tone-timer.
                                                                                  */
                   r7, 6
          movi
                   r7, 4(r6)
          sthio
          movia
                  r6, minute_timer
                                                /* Address of minute-timer.
                                                                                  */
                                                /* Start the timer.
          addi
                   r7, r0, 7
                                                                                  */
          sthio
                   r7, 4(r6)
                   r7, 1
          movi
                                                                                  */
                                                /* Enable timer interrupts.
          wrctl
                   ienable, r7
          wrctl
                   status, r7
                                                /* Enable external interrupts.
                                                                                  */
LOOP:
                  r10, ACTUAL_TIME
                                                /* Display the time of day.
                                                                                  */
         movia
                   DISP
          call
          ldbio
                   r7, (r2)
                                                /* Check if alarm switch is on.
                                                                                  */
          andi
                   r11, r7, 1
          beq
                   r11, r0, NEXT
          movi
                   r11,7
                                                /* If yes, then turn on the
                                                                                  */
          movia
                  r12, ALARM_STATE
                                                                                  */
                                                   alarm LED.
          stb
                   r11, (r12)
          movia
                   r9, ALARM_TIME
                                                                                  */
          ldw
                   r11, (r9)
                                                /* Have to compare alarm-time
          ldw
                   r12, (r10)
                                                /* with actual-time.
                                                                                  */
                   r11, r12, TEST_SLIDERS
                                               /* Should the alarm ring?
                                                                                  */
         bne
         movia
                  r8, tone_timer
                   r12, 1
          movi
RING_LOOP:
         call
                   DISP
          ldbio
                   r7, (r2)
                                                /* Check if alarm switch is on.
                                                                                  */
                   r13, r7, 1
          andi
                   r13, r0, NEXT
          beq
                   r9, (r8)
                                                /* Read the tone-timer status.
                                                                                  */
          ldhio
                                                /* Clear the TO bit.
                                                                                  */
          sthio
                   r0, (r8)
                                                /* Check if counter reached 0.
                                                                                  */
          andi
                   r9, r9, 1
                                                                                  */
                   r12, r9, r12
                                                /* Generate the next square
          xor
                                                /* wave half-cycle; send
                                                                                  */
          movia
                  r11, speaker
                                                /* signal to the speaker.
                                                                                  */
          stbio
                   r12, (r11)
          br
                   RING_LOOP
```

```
NEXT:
          movi
                   r11,6
                                           /* Turn off the alarm-on
                                                                                 */
                   r12, ALARM_STATE
          movia
                                                                                 */
                                           /* LED indicator.
          stb
                   r11, (r12)
TEST_SLIDERS:
          ldbio
                   r7, (r2)
                                                                                 */
                   r11, r7, 2
                                           /* Is set-alarm switch on?
          andi
                                           /* If not, test actual time.
          beq
                   r11, r0, SETACT
                                                                                 */
                   r10, ALARM_TIME
                                           /* Have to set the alarm time.
                                                                                 */
          movia
          br
                   SET_TIME
SETACT:
                                           /* Is set-time switch on?
          andi
                   r11, r7, 4
                                                                                 */
                                           /* All sliders are off.
                                                                                 */
          beq
                   r11, r0, LOOP
                   r10, ACTUAL_TIME
          movia
SET_TIME:
          call
                   DISP
                   SETSUB
          call
                   TEST_SLIDERS
          br
          Display the time on 7-segment displays.
                                                                                 */
DISP:
          subi
                   sp, sp, 24
                                           /* Save registers.
                                                                                 */
                   r11, 0(sp)
          stw
          stw
                   r12, 4(sp)
          stw
                   r13, 8(sp)
          stw
                   r14, 12(sp)
          stw
                   r15, 16(sp)
                   r16, 20(sp)
          stw
                                           /* Load the time to be displayed.
                                                                                 */
                   r11, (r10)
          ldw
                   r13, ALARM_STATE
          movia
          ldw
                   r13, (r13)
                   r12, r11, 720
          subi
          blt
                   r12, r0, AM
                   r13, r13, 0x8
                                           /* It is PM.
                                                                                 */
          ori
                   r11, r12
          mov
AM:
          stbio
                   r13, (r3)
                                           /* To determine the first digit of
                                                                                 */
          movi
                   r12,600
                                           /* hours, divide by 600.
                                                                                 */
          divu
                   r13, r11, r12
          ldb
                                           /* Get the 7-segment pattern.
                                                                                 */
                   r15, TABLE(r13)
                                                                                 */
                   r15, r15, 8
                                           /* Make space for next digit.
          slli
                                           /* Compute the remainder of the
                                                                                 */
          mul
                   r14, r13, r12
                                                                                 */
          sub
                   r11, r11, r14
                                           /* division operation.
          movi
                   r12, 60
                                           /* Divide the remainder by 60 to
                                                                                 */
          divu
                   r13, r11, r12
                                           /* get the second digit of hours.
                                                                                 */
                                           /* Get the 7-segment pattern,
          ldb
                   r16, TABLE(r13)
                                                                                 */
                   r15, r15, r16
                                              concatenate it to the first
                                                                                 */
          or
                                                                                 */
          slli
                   r15, r15, 8
                                              digit, and shift.
                                           /* Determine the minutes that have
                                                                                 */
          mul
                   r14, r13, r12
          sub
                   r11, r11, r14
                                           /* to be displayed.
                                                                                 */
```

...continued in Part d of Figure 11.7

11.5. Assuming that there is only one timer, with a timeout period of one millisecond, the program in Figure 11.6 can be modified as follows:

```
#define
                            timer (volatile int *) 0x5020
#define
                            sliders (volatile int *) 0x5040
#define
                            pushbuttons (volatile int *) 0x5050
#define
                            display (int *) 0x5060
#define
                            LEDs (int *) 0x5070
#define
                            speaker (int *) 0x5080
#define
                            ADJUST(t, x) ((t + x) >= 1440)? (t + x - 1440): (t + x)
int
                            actual_time, alarm_time, alarm_active, time, counter, tone;
   /* Hex to 7-segment conversion table */
    unsigned char table [16] = \{0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x12, 0x02, 0x78, 0x12, 
             0x00, 0x18, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F};
    void DISP(time)
                                                                                                            /* Get 7-segment patterns for display. */
              *display = table[time / 600] << 24 |
                               table[(time % 600) / 60] << 16
                               table[(time % 60) / 10] << 8 |
                               table[(time % 10)];
   }
   main()
              actual_time = alarm_time = alarm_active = 0;
              counter = tone = 0;
                                                                                                             /* Run in continuous mode. */
              *(timer + 1) = 0x6;
              while (1)
                    if (*timer == 3)
                                                                                                            /* One ms elapsed. */
                               *timer = 0;
                                                                                                            /* Clear the TO bit. */
                                                                                                            /* Generate the alarm sound. */
                               tone = tone ^{\wedge} 1;
                              if (counter < 60000)
                                                                                                            /* Test if one minute. */
                                      counter = counter + 1;
                               else
                                     counter = 0;
                                      actual_time = ADJUST(actual_time, 1);
```

```
if ((*sliders \& 1) != 0)
                                /* Check the alarm-on switch. */
{
                                /* Turn on the alarm LED. */
       *LEDs = 7;
      if (actual_time == alarm_time)
         alarm\_active = 1;
                                /* Start the alarm sound. */
      else
         alarm_active = alarm_active & (*sliders & 1);
       *speaker = tone & alarm_active; /* Output the square wave. */
}
else
{
                                /* Turn off the alarm LED. */
       *LEDs = 6;
      alarm\_active = 0;
if ((*sliders & 4) != 0)
                                /* Check the set-the-time-of-day switch. */
      DISP(actual_time);
                                /* Display the time of day. */
      if ((*(pushbuttons + 3) \& 1) != 0) /* Set the minutes? */
         actual_time = ADJUST(actual_time, 1);
      else if ((*(pushbuttons + 3) & 2) != 0) /* Set the hours? */
         actual_time = ADJUST(actual_time, 60);
       *(pushbuttons + 3) = 0; /* Clear the edge-capture register. */
else if ((*sliders & 2) != 0)
                                /* Check the set-the-alarm-time switch. */
      DISP(alarm_time);
                                /* Display the alarm time. */
      if ((*(pushbuttons + 3) & 1) != 0) /* Set the minutes? */
         alarm_time = ADJUST(alarm_time, 1);
      else if ((*(pushbuttons + 3) & 2) != 0) /* Set the hours? */
         alarm_time = ADJUST(alarm_time, 60);
       *(pushbuttons + 3) = 0; /* Clear the edge-capture register. */
}
else
      DISP(actual_time);
                                /* Display the time of day. */
```

11.6. Assuming that there is only one timer, with a timeout period of one millisecond, the program in Figure 11.7 can be modified as follows:

```
timer, 0x05000
         .equ
         .equ
                  sliders, 0x5040
                  pushbuttons, 0x5050
         .equ
                  display, 0x5060
         .equ
                  LEDs, 0x05070
         .equ
         .equ
                  speaker, 0x5080
         .equ
                  ACTUAL_TIME, 0x1000
                  ALARM_TIME, 0x1010
         .equ
                  COUNTER, 0x1020
         .equ
                  TONE, 0x1030
         .equ
         .equ
                  SIXTYTHOU, 60000
         .equ
                  STACK, 0x2000
_start:
         br
                  MAIN
         Interrupt handler
                  0x20
         .org
         subi
                  sp, sp, 8
                                        /* Save registers.
         stw
                  r2, 0(sp)
                  ra, 4(sp)
         stw
                  et, ipending
         rdctl
                                                                            */
                  et, r0, MAIN
                                        /* Error if not an external
         beq
                                                                            */
                                         /* interrupt, treat as reset.
         subi
                  ea, ea, 4
                                        /* Decrement ea to execute the
                                                                            */
                                        /* interrupted instruction upon
                                                                            */
                                        /* return to the main program.
                                                                            */
                                        /* Clear the TO bit
                                                                            */
                  r2, timer
         movia
                                        /* in the timer.
                                                                            */
         sthio
                  r0, (r2)
                                                                            */
                                        /* Call interrupt-service routine.
         call
                  UPDATE_TIME
                                        /* Restore registers.
         ldw
                  r2, 0(sp)
         ldw
                  ra, 4(sp)
         addi
                  sp, sp, 8
         eret
         Main program
                                                                            */
MAIN:
         movia
                  sp, STACK
                                        /* Set up the stack pointer.
                                                                            */
                                        /* Clear the alarm-time buffer.
         movia
                  r2, ALARM_TIME
                  r0, (r2)
         stw
                  r2, ACTUAL_TIME
                                        /* Clear the actual-time buffer.
         movia
                  r0, (r2)
         stw
                  r2, COUNTER
                                        /* Clear the counter buffer.
         movia
                  r0, (r2)
         stw
                  r2, TONE
                                        /* Clear the tone buffer.
                                                                            */
         movia
                  r0, (r2)
         stw
                                        /* Address of slider switches.
                  r2, sliders
                                                                            */
         movia
                  r3, LEDs
                                        /* Address of LEDs.
                                                                            */
         movia
                                        /* Address of 7-segment displays.
                                                                            */
         movia
                  r4, display
                                        /* Address of pushbuttons.
                                                                            */
         movia
                  r5, pushbuttons
```

```
/* Turn on the two
                                                                            */
         movi
                  r6, 6
                  r6, (r3)
                                          /* vertical LEDs.
                                                                            */
         stbio
                                                                            */
                                          /* Address of the timer.
                  r6, timer
          movia
                                          /* Start the timer.
                                                                            */
         movi
                  r7, 7
                  r7, 4(r6)
         sthio
                  r7, 1
         movi
                                          /* Enable timer interrupts.
                                                                            */
          wrctl
                  ienable, r7
                                          /* Enable external interrupts.
         wrctl
                  status, r7
                                                                            */
LOOP:
                                          /* Display the time of day.
         movia
                  r10, ACTUAL_TIME
                                                                            */
         call
                  DISP
         ldbio
                  r7, (r2)
                                          /* Check if alarm switch is on.
                                                                            */
         andi
                  r11, r7, 1
                  r11, r0, NEXT
         beq
                                                                            */
                  r11, 7
                                          /* If yes, then turn on the
         movi
         stbio
                  r11, (r3)
                                          /* alarm LED.
                                                                            */
                  r9, ALARM_TIME
         movia
                                                                            */
         ldw
                  r11, (r9)
                                          /* Have to compare alarm-time
                                                                            */
         ldw
                  r12, (r10)
                                          /* with actual-time.
                  r11, r12, NEXT
                                          /* Should the alarm ring?
                                                                            */
         bne
RING_LOOP:
         call
                  DISP
         ldbio
                  r7, (r2)
                                          /* Check if alarm switch is on.
                                                                            */
         andi
                  r13, r7, 1
         beq
                  r13, r0, NEXT
         movia
                  r11, speaker
         movia
                  r12, TONE
                  r12, (r12)
                                          /* Send the tone
                                                                            */
         ldw
                                          /* signal to the speaker.
         stbio
                  r12, (r11)
                                                                            */
                  RING_LOOP
         br
```

```
NEXT:
                   r11,6
                                           /* Turn off the alarm-on
                                                                                 */
         movi
                                           /* LED indicator.
                                                                                 */
          stbio
                   r11, (r3)
TEST_SLIDERS:
         ldbio
                   r7, (r2)
                                           /* Is set-alarm switch on?
                                                                                 */
          andi
                   r11, r7, 2
                                           /* If not, test actual time.
                                                                                 */
         beq
                   r11, r0, SETACT
          movia
                   r10, ALARM_TIME
                                           /* Have to set the alarm time.
                                                                                 */
                   SET_TIME
          br
SETACT:
          andi
                   r11, r7, 4
                                           /* Is set-time switch on?
                                                                                 */
                   r11, r0, LOOP
                                           /* All sliders are off.
                                                                                 */
         beq
         movia
                  r10, ACTUAL_TIME
SET_TIME:
                   DISP
          call
         call
                   SETSUB
                   TEST_SLIDERS
         br
          Display the time on 7-segment displays.
                                                                                 */
DISP:
          subi
                   sp, sp, 24
                                           /* Save registers.
          stw
                   r11, 0(sp)
                   r12, 4(sp)
          stw
          stw
                   r13, 8(sp)
                   r14, 12(sp)
          stw
          stw
                   r15, 16(sp)
                   r16, 20(sp)
          stw
          ldw
                   r11, (r10)
                                           /* Load the time to be displayed.
                                                                                 */
                                           /* To determine the first digit of
                                                                                 */
                   r12,600
         movi
          divu
                   r13, r11, r12
                                           /* hours, divide by 600.
                                                                                 */
                                                                                 */
         ldb
                   r15, TABLE(r13)
                                           /* Get the 7-segment pattern.
                   r15, r15, 8
                                           /* Make space for next digit.
                                                                                 */
          slli
                   r14, r13, r12
                                           /* Compute the remainder of the
                                                                                 */
          mul
                   r11, r11, r14
                                           /* division operation.
                                                                                 */
          sub
                                                                                 */
                   r12, 60
                                           /* Divide the remainder by 60 to
          movi
                                           /* get the second digit of hours.
                                                                                 */
          divu
                   r13, r11, r12
                                                                                 */
         ldb
                   r16, TABLE(r13)
                                           /* Get the 7-segment pattern,
                                           /* concatenate it to the first
                                                                                 */
          or
                   r15, r15, r16
                                           /* digit, and shift.
                                                                                 */
          slli
                   r15, r15, 8
                   r14, r13, r12
                                           /* Determine the minutes that have
                                                                                 */
          mul
                                                                                 */
          sub
                   r11, r11, r14
                                           /* to be displayed.
```

```
*/
                     r12, 10
                                         /* To determine the first digit of
            movi
                                                                                 */
            divu
                     r13, r11, r12
                                         /* minutes, divide by 10.
                                                                                 */
            ldb
                     r16, TABLE(r13)
                                         /* Get the 7-segment pattern,
                     r15, r15, r16
                                         /* concatenate it to the first
                                                                                 */
            or
                                                                                 */
            slli
                     r15, r15, 8
                                         /* two digits, and shift.
                                         /* Compute the remainder, which
                                                                                 */
            mul
                     r14, r13, r12
                                                                                 */
            sub
                     r11, r11, r14
                                         /* is the last digit.
            ldb
                     r16, TABLE(r11)
                                         /* Concatenate the last digit to
                                                                                 */
                                                                                 */
            or
                     r15, r15, r16
                                         /* the preceding 3 digits.
                     r11, display
            movia
                                                                                 */
                     r15, (r11)
                                         /* Display the obtained pattern.
            stw
                                                                                 */
            ldw
                     r11, 0(sp)
                                         /* Restore registers.
                     r12, 4(sp)
            ldw
            ldw
                     r13, 8(sp)
            ldw
                     r14, 12(sp)
            ldw
                     r15, 16(sp)
            ldw
                     r16, 20(sp)
            addi
                     sp, sp, 24
            ret
           Set the desired time.
                                          */
SETSUB:
                                         /* Save registers.
                                                                                 */
            subi
                     sp, sp, 16
            stw
                     r11, 0(sp)
                     r12, 4(sp)
            stw
                     r13, 8(sp)
            stw
                     r14, 12(sp)
            stw
            ldbio
                     r12, 12(r5)
                                         /* Test pushbuttons.
                                                                                 */
                                                                                 */
            stbio
                     r0, 12(r5)
                                         /* Clear edge-detection register.
                                         /* Is minute pushbutton pressed?
                                                                                 */
            andi
                     r13, r12, 1
                                                                                 */
           beq
                     r13, r0, HOURS
                                         /* If not, check hours.
            ldw
                     r11, (r10)
                                         /* Load present time.
                                                                                 */
                                                                                 */
            movi
                     r12, 60
                                         /* Divide by 60 to determine
                                                                                 */
            divu
                     r13, r11, r12
                                         /* the number of hours.
                                                                                 */
            mul
                     r14, r13, r12
                                         /* Remainder of division operation
                                                                                 */
            sub
                     r11, r11, r14
                                         /* is the number of minutes.
                                         /* Increment minutes.
                                                                                 */
            addi
                     r11, r11, 1
                                                                                 */
           blt
                     r11, r12, SAVEM
                                         /* Save if less than 60,
                                         /* otherwise set minutes to 00.
                                                                                 */
            mov
                     r11, r0
SAVEM:
                                                                                 */
           add
                     r11, r14, r11
                                         /* (hours x 60) + (updated minutes).
            stw
                     r11, (r10)
                                         /* Save the new time.
                                                                                 */
           br
                     DONE
```

```
HOURS:
              andi
                       r13, r12, 2
                                              /* Is hour pushbutton pressed?
                                                                                 */
                                                                                 */
              beq
                       r13, r0, DONE
                                              /* If not, then return.
              ldw
                                                                                 */
                       r11, (r10)
                                              /* Load present time in minutes.
              addi
                       r12, r11, 60
                                              /* Add 60 minutes.
                                                                                 */
              movi
                       r13, 1440
                                              /* Have to check if updated time
                                                                                 */
                                              /* is less than 24:00.
                                                                                 */
              blt
                       r12, r13, SAVEH
                                                                                 */
              sub
                       r12, r12, r13
                                              /* Roll over hours to 00.
                                                                                 */
SAVEH:
                       r12, (r10)
                                              /* Save the new time.
              stw
                                                                                 */
DONE:
              ldw
                       r11, 0(sp)
                                              /* Restore registers.
              ldw
                       r12, 4(sp)
              ldw
                       r13, 8(sp)
              ldw
                       r14, 12(sp)
              addi
                       sp, sp, 16
              ret
              Interrupt-service routine that updates actual-time
                                                                                 */
UPDATE_TIME:
                                              /* Save registers.
                                                                                 */
              subi
                       sp, sp, 12
              stw
                       r2, 0(sp)
              stw
                       r3, 4(sp)
                       r4, 8(sp)
              stw
              movia
                       r2, TONE
              ldw
                       r3, (r2)
              xori
                       r3, r3, 1
                                              /* Generate the square wave.
                                                                                 */
              stw
                       r3, (r2)
              movia
                       r2, COUNTER
              ldw
                       r3, (r2)
                       r4. SIXTYTHOU
                                              /* Test if one minute
                                                                                 */
              movia
                                                                                 */
              bge
                       r3, r4, INCTIME
                                              /* has elapsed.
                       r3, r3, 1
              addi
                       r3, (r2)
              stw
                       RESTORE
              br
INCTIME:
                       r0, (r2)
              stw
                       r2, ACTUAL_TIME
              movia
                                                                                 */
              ldw
                       r3, (r2)
                                              /* Load present time of day.
                                                                                 */
              addi
                       r3, r3, 1
                                              /* Increment by one minute.
                                              /* Done if updated time is
                                                                                 */
              movi
                       r4, 1440
                       r3, r4, SAVET
                                              /* less than 24:00.
                                                                                 */
              blt
                                              /* Otherwise, set to 00:00.
                                                                                 */
              mov
                       r3, r0
                                                                                 */
SAVET:
                       r3, (r2)
                                              /* Save updated time.
              stw
RESTORE:
              ldw
                       r2, 0(sp)
                                              /* Restore registers.
                                                                                 */
              ldw
                       r3, 4(sp)
              ldw
                       r4, 8(sp)
              addi
                       sp, sp, 12
              ret
/*
              Hex-digit to 7-segment conversion table
                                                                                 */
                       0x1050
              .org
TABLE:
              .byte
                       0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78
              .byte
                       0x00, 0x18, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F
              .end
```

11.7. The interrupt handler must determine which timer has raised an interrupt request, and then call the appropriate interrupt-service routine. The program in Figure 11.7 may be modified as follows:

```
minute_timer, 0x05000
          .equ
          .equ
                  tone_timer, 0x5020
                  sliders, 0x5040
          .equ
                  pushbuttons, 0x5050
          .equ
          .equ
                  display, 0x5060
          .equ
                  LEDs, 0x05070
          .equ
                  speaker, 0x5080
                  ACTUAL_TIME, 0x1000
          .equ
                  ALARM_TIME, 0x1010
          .equ
                  TONE, 0x1020
          .equ
                  STACK, 0x2000
          .equ
_start:
         br
                  MAIN
         Interrupt handler
                                                                                    */
                  0x20
          .org
                  sp, sp, 8
                                         /* Save registers.
                                                                                    */
         subi
         stw
                  r^{2}, 0(sp)
         stw
                  ra, 4(sp)
                  et, ipending
         rdctl
                  et, r0, MAIN
                                         /* Error if not an external interrupt.
                                                                                    */
         beq
         subi
                  ea, ea, 4
                                         /* Decrement ea to ensure proper return.
                                                                                    */
IRQ0:
                  r2, et, 1
         andi
         beq
                  r2, r0, IRQ2
                  r2, minute_timer
                                         /* Clear the TO bit in the
                                                                                    */
         movia
                                         /* minute-timer.
                                                                                    */
         sthio
                  r0, (r2)
                                                                                    */
         call
                  UPDATE_TIME
                                         /* Call interrupt-service routine.
IRQ2:
         andi
                  r2, et. 4
         beq
                  r2, r0, LAST
         movia
                  r2, tone_timer
                                         /* Clear the TO bit in the
                                                                                    */
                                         /* tone-timer.
                                                                                    */
         sthio
                  r0, (r2)
                                                                                    */
         call
                  RING
                                         /* Call interrupt-service routine.
LAST:
                                                                                    */
         ldw
                  r2, 0(sp)
                                         /* Restore registers.
         ldw
                  ra, 4(sp)
         addi
                  sp, sp, 8
         eret
                                                                                    */
         Main program
MAIN:
         movia
                  sp, STACK
                                         /* Set up the stack pointer.
                                                                                    */
                  r2, ALARM_TIME
                                         /* Clear the alarm-time buffer.
                                                                                    */
         movia
         stw
                  r0, (r2)
                  r2, ACTUAL_TIME
                                         /* Clear the actual-time buffer.
                                                                                    */
         movia
         stw
                  r0, (r2)
                  r2, TONE
                                         /* Clear the tone buffer.
                                                                                    */
         movia
         stw
                  r0, (r2)
                  r2, sliders
                                         /* Address of slider switches.
                                                                                    */
         movia
                                         /* Address of LEDs.
                                                                                    */
         movia
                  r3, LEDs
                                         /* Address of 7-segment displays.
                                                                                    */
         movia
                  r4, display
                  r5, pushbuttons
                                         /* Address of pushbuttons.
                                                                                    */
         movia
```

	movi	r6, 6	/* Turn on the two	*/
	stbio	r6, (r3)	/* vertical LEDs.	*/
	movia	r6, tone_timer		
	ori	r7, r0, 0x0D40	/* Set the tone-timer period.	*/
	sthio	r7, 8(r6)		
	ori	r7, r0, 0x03		
	sthio	r7, 12(r6)		
	movi	r7, 7	/* Start the tone-timer.	*/
	sthio	r7, 4(r6)		
	movia	r6, minute_timer	/* Address of minute-timer.	*/
	sthio	r7, 4(r6)	/* Start the minute-timer.	
	movi	r7, 5		
	wrctl	ienable, r7	/* Enable timer interrupts.	*/
	movi	r7, 1	-	
	wrctl	status, r7	/* Enable external interrupts.	*/
LOOP:	movia	r10, ACTUAL_TIME	/* Display the time of day.	*/
	call	DISP		
	ldbio	r7, (r2)		
	andi	r11, r7, 1	/* Check if alarm switch is on.	*/
	beq	r11, r0, NEXT		
	movi	r11, 7	/* If yes, then turn on the	*/
	stbio	r11, (r3)	/* alarm LED.	*/
	movia	r9, ALARM_TIME		
	ldw	r11, (r9)	/* Have to compare alarm-time	*/
	ldw	r12, (r10)	/* with actual-time.	*/
	bne	r11, r12, NEXT	/* Should the alarm ring?	*/
RING_L	OOP:		_	
	call	DISP		
	ldbio	r7, (r2)		
	andi	r13, r7, 1	/* Check if alarm switch is on.	*/
	beq	r13, r0, NEXT		
	movia	r11, speaker		
	movia	r12, TONE		
	ldw	r12, (r12)	/* Send the tone signal	*/
	stbio	r12, (r11)	/* to the speaker.	*/
	br	RING_LOOP	-	

```
NEXT:
         movi
                   r11,6
                                           /* Turn off the alarm-on
                                                                                 */
                                           /* LED indicator.
                                                                                 */
          stbio
                   r11, (r3)
TEST_SLIDERS:
         ldbio
                   r7, (r2)
                                           /* Is set-alarm switch on?
                                                                                 */
          andi
                   r11, r7, 2
                                           /* If not, test actual time.
                                                                                 */
         beq
                   r11, r0, SETACT
          movia
                   r10, ALARM_TIME
                                           /* Have to set the alarm time.
                                                                                 */
                   SET_TIME
          br
SETACT:
          andi
                   r11, r7, 4
                                           /* Is set-time switch on?
                                                                                 */
                   r11, r0, LOOP
                                           /* All sliders are off.
                                                                                 */
         beq
                  r10, ACTUAL_TIME
         movia
SET_TIME:
                   DISP
          call
         call
                   SETSUB
                   TEST_SLIDERS
         br
          Display the time on 7-segment displays.
                                                                                 */
DISP:
          subi
                   sp, sp, 24
                                           /* Save registers.
          stw
                   r11, 0(sp)
                   r12, 4(sp)
          stw
          stw
                   r13, 8(sp)
          stw
                   r14, 12(sp)
          stw
                   r15, 16(sp)
                   r16, 20(sp)
          stw
          ldw
                   r11, (r10)
                                           /* Load the time to be displayed.
                                                                                 */
                                           /* To determine the first digit of
                                                                                 */
                   r12,600
         movi
          divu
                   r13, r11, r12
                                           /* hours, divide by 600.
                                                                                 */
                                                                                 */
         ldb
                   r15, TABLE(r13)
                                           /* Get the 7-segment pattern.
                   r15, r15, 8
                                           /* Make space for next digit.
                                                                                 */
          slli
                   r14, r13, r12
                                           /* Compute the remainder of the
                                                                                 */
          mul
                   r11, r11, r14
                                           /* division operation.
                                                                                 */
          sub
                                                                                 */
                   r12, 60
                                           /* Divide the remainder by 60 to
          movi
                                           /* get the second digit of hours.
                                                                                 */
          divu
                   r13, r11, r12
                                           /* Get the 7-segment pattern,
                                                                                 */
         ldb
                   r16, TABLE(r13)
                                           /* concatenate it to the first
                                                                                 */
          or
                   r15, r15, r16
                                           /* digit, and shift.
                                                                                 */
          slli
                   r15, r15, 8
                   r14, r13, r12
                                           /* Determine the minutes that have
                                                                                 */
          mul
                                                                                 */
          sub
                   r11, r11, r14
                                           /* to be displayed.
```

```
r12, 10
                                          /* To determine the first digit of
                                                                                  */
            movi
                                                                                  */
            divu
                     r13, r11, r12
                                          /* minutes, divide by 10.
                                                                                  */
            ldb
                     r16, TABLE(r13)
                                          /* Get the 7-segment pattern,
                     r15, r15, r16
                                          /* concatenate it to the first
                                                                                  */
            or
            slli
                     r15, r15, 8
                                          /* two digits, and shift.
                                                                                  */
                                                                                  */
            mul
                     r14, r13, r12
                                          /* Compute the remainder, which
                                                                                  */
                     r11, r11, r14
                                          /* is the last digit.
            sub
            ldb
                                          /* Concatenate the last digit to
                                                                                  */
                     r16, TABLE(r11)
            or
                     r15, r15, r16
                                          /* the preceding 3 digits.
                                                                                  */
            movia
                     r11, display
                                                                                  */
                                          /* Display the obtained pattern.
            stw
                     r15, (r11)
                                                                                  */
            ldw
                     r11, 0(sp)
                                          /* Restore registers.
            ldw
                     r12, 4(sp)
            ldw
                     r13, 8(sp)
            ldw
                     r14, 12(sp)
            ldw
                     r15, 16(sp)
            ldw
                     r16, 20(sp)
            addi
                     sp, sp, 24
            ret
            Set the desired time.
                                          */
SETSUB:
                                                                                  */
            subi
                     sp, sp, 16
                                          /* Save registers.
            stw
                     r11, 0(sp)
            stw
                     r12, 4(sp)
                     r13, 8(sp)
            stw
                     r14, 12(sp)
            stw
                     r12, 12(r5)
                                          /* Test pushbuttons.
                                                                                  */
            ldbio
                                                                                  */
            stbio
                     r0, 12(r5)
                                          /* Clear edge-detection register.
                                          /* Is minute pushbutton pressed?
                                                                                  */
            andi
                     r13, r12, 1
                     r13, r0, HOURS
                                          /* If not, check hours.
                                                                                  */
            beq
                                                                                  */
            ldw
                     r11, (r10)
                                          /* Load present time.
                                                                                  */
                     r12, 60
                                          /* Divide by 60 to determine
            movi
                                                                                  */
            divu
                     r13, r11, r12
                                          /* the number of hours.
                                                                                  */
                                          /* Remainder of division operation
            mul
                     r14, r13, r12
                                                                                  */
            sub
                     r11, r11, r14
                                          /* is the number of minutes.
                                          /* Increment minutes.
                                                                                  */
            addi
                     r11, r11, 1
                     r11, r12, SAVEM
                                          /* Save if less than 60,
                                                                                  */
            blt
                                                                                  */
            mov
                     r11, r0
                                          /* otherwise set minutes to 00.
SAVEM:
            add
                     r11, r14, r11
                                          /* (hours x 60) + (updated minutes).
                                                                                  */
            stw
                     r11, (r10)
                                          /* Save the new time.
                                                                                  */
            br
                     DONE
                                                                                  */
HOURS:
            andi
                     r13, r12, 2
                                          /* Is hour pushbutton pressed?
                                                                                  */
                     r13, r0, DONE
                                          /* If not, then return.
            beq
            ldw
                     r11, (r10)
                                          /* Load present time in minutes.
                                                                                  */
            addi
                     r12, r11, 60
                                          /* Add 60 minutes.
                                                                                  */
            movi
                     r13, 1440
                                          /* Have to check if updated time
                                                                                  */
                                          /* is less than 24:00.
                                                                                  */
            blt
                     r12, r13, SAVEH
                                          /* Roll over hours to 00.
                                                                                  */
            sub
                     r12, r12, r13
SAVEH:
                                                                                  */
                                          /* Save the new time.
                     r12, (r10)
            stw
```

```
DONE:
           ldw
                    r11, 0(sp)
                                           /* Restore registers.
                                                                          */
                    r12, 4(sp)
           ldw
           ldw
                    r13, 8(sp)
           ldw
                    r14, 12(sp)
           addi
                    sp, sp, 16
           ret
           Interrupt-service routine that updates actual-time
                                                                          */
UPDATE_TIME:
                                                                          */
           subi
                    sp, sp, 16
                                           /* Save registers.
                    ra, (sp)
           stw
                    r2, 4(sp)
           stw
                    r3, 8(sp)
           stw
           stw
                    r4, 12(sp)
           movia
                    r2, ACTUAL_TIME
                                           /* Load present time of day.
                                                                          */
           ldw
                    r3, (r2)
           addi
                    r3, r3, 1
                                           /* Increment by one minute.
                                                                          */
                    r4, 1440
                                           /* Done if updated time is
                                                                          */
           movi
                    r3, r4, SAVET
                                           /* less than 24:00.
                                                                          */
           blt
           mov
                    r3, r0
                                           /* Otherwise, set to 00:00.
                                                                          */
SAVET:
                    r3, (r2)
                                           /* Save updated time.
                                                                          */
           stw
                                                                          */
                                           /* Restore registers.
           ldw
                    r4, 12(sp)
           ldw
                    r3, 8(sp)
           ldw
                    r2, 4(sp)
           ldw
                    ra, (sp)
           addi
                    sp, sp, 16
           ret
                                                                          */
           Interrupt-service routine that updates the tone signal
RING:
                                           /* Save registers.
                                                                          */
           subi
                    sp, sp, 12
                    ra, (sp)
           stw
                    r2, 4(sp)
           stw
                    r3, 8(sp)
           stw
                    r2, TONE
           movia
                                           /* Invert the logic value of
           ldw
                    r3, (r2)
                                                                          */
                                           /* the tone signal.
                                                                          */
           xori
                    r3, r3, 1
           stw
                    r3, (r2)
                                                                          */
           ldw
                    r3, 8(sp)
                                           /* Restore registers.
           ldw
                    r2, 4(sp)
           ldw
                    ra, (sp)
           addi
                    sp, sp, 12
           ret
/*
                                                                          */
           Hex-digit to 7-segment conversion table
           .org
                    0x1050
TABLE:
                    0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78
           .byte
           .byte
                    0x00, 0x18, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F
           .end
```

11.8. It is necessary to detect both the transition when a pushbutton is pressed and the fact that it may still be pressed after 0.5 second time delays. A counter is used to determine each 500-ms interval; it is incremented by using the tone-timer. A new variable "buttons" is introduced to represent the combined effect. The continuing state of pushbuttons is examined by reading the Data register of the corresponding PIO. Note that a pressed pushbutton creates logic 0 in the Data register.

```
#define
                         minute_timer (volatile int *) 0x5000
#define
                         tone_timer (volatile int *) 0x5020
#define
                         sliders (volatile int *) 0x5040
#define
                         pushbuttons (volatile int *) 0x5050
#define
                         display (int *) 0x5060
#define
                         LEDs (int *) 0x5070
#define
                         speaker (int *) 0x5080
#define
                         ADJUST(t, x) ((t + x) > = 1440)? (t + x - 1440): (t + x)
int
                         actual_time, alarm_time, alarm_active, time;
int
                         counter, delay, buttons;
   /* Hex to 7-segment conversion table */
   unsigned char table [16] = \{0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x12, 0x02, 0x78, 0x12, 
            0x00, 0x18, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F};
   void initializeToneTimer()
            *(tone_timer + 2) = 0x0D40; /* Set the timeout period
            *(tone\_timer + 3) = 0x03;
                                                                                            /* for continuous operation. */
                                                                                            /* Start in continuous mode. */
            *(tone\_timer + 1) = 0x6;
   void DISP(time)
                                                                                            /* Get 7-segment patterns for display. */
            *display = table[time / 600] << 24 |
                            table[(time % 600) / 60] << 16
                            table[(time % 60) / 10] << 8
                            table[(time % 10)];
   main()
   {
            actual_time = alarm_time = alarm_active = 0;
            counter = delay = buttons = 0;
            initializeToneTimer();
                                                                                            /* Run in continuous mode. */
            *(minute_timer + 1) = 0x6;
            while (1)
                   if (*minute_timer == 3)
                                                                                            /* One minute elapsed. */
                            *minute_timer = 0;
                                                                                            /* Clear the TO bit. */
                            actual_time = ADJUST(actual_time, 1);
                   }
```

```
if (*tone_timer == 3)
                                 /* One ms elapsed. */
       *tone\_timer = 0;
                                 /* Clear the TO bit. */
                                 /* Wait for 0.5 second. */
      if (counter < 500)
         counter = counter + 1;
      else
         counter = 0;
                                 /* Set for new test. */
         delay = 7;
if ((*sliders \& 1) != 0)
                                 /* Check the alarm-on switch. */
       *LEDs = 7;
                                 /* Turn on the alarm LED. */
      if (actual_time == alarm_time)
             alarm_active = 1; /* Start the alarm sound. */
      else
             alarm_active = alarm_active & (*sliders & 1);
                                 /* Generate the square wave. */
      if (*tone_timer == 3)
             *speaker = (*speaker ^{\wedge} 1) & alarm_active;
             *tone_timer = 0; /* Clear the TO bit. */
else
                                 /* Turn off the alarm LED. */
      *LEDs = 6;
      alarm\_active = 0;
buttons = *(pushbuttons + 3) | ((*pushbuttons ^ 7) & delay);
                                 /* Check the set-the-time-of-day switch. */
if ((*sliders & 4) != 0)
{
                                 /* Display the time of day. */
      DISP(actual_time);
      if ((*(buttons + 3) \& 1) != 0) /* Set the minutes? */
             actual_time = ADJUST(actual_time, 1);
      else if ((*(buttons + 3) & 2) != 0) /* Set the hours? */
             actual_time = ADJUST(actual_time, 60);
       *(pushbuttons + 3) = 0; /* Clear the edge-capture register. */
                                 /* Check the set-the-alarm-time switch. */
else if ((*sliders & 2) != 0)
      DISP(alarm_time);
                                 /* Display the alarm time. */
      if ((*(buttons + 3) & 1) != 0) /* Set the minutes? */
             alarm_time = ADJUST(alarm_time, 1);
      else if ((*(buttons + 3) & 2) != 0) /* Set the hours? */
             alarm_time = ADJUST(alarm_time, 60);
       *(pushbuttons + 3) = 0; /* Clear the edge-capture register. */
}
else
                                 /* Display the time of day. */
      DISP(actual_time);
delay = 0;
                                 /* Wait for the next test. */
```

}

11.9. It is necessary to detect both the transition when a pushbutton is pressed and the fact that it may still be pressed after 0.5 second time delays. A counter is used to determine each 500-ms interval; it is incremented by using the tone-timer. The continuing state of pushbuttons is examined by reading the Data register of the corresponding PIO. Note that a pressed pushbutton creates logic 0 in the Data register.

The program in Figure 11.7 can be modified by using a memory location COUNTER to keep track of the count, and changing the subroutine SETSUB in Figure 11.7d as follows:

<b>/</b> *	Set the desired time.		*/	
SETSUB:	subi	sp, sp, 20	/* Save registers.	*/
	stw	r11, 0(sp)		
	stw	r12, 4(sp)		
	stw	r13, 8(sp)		
	stw	r14, 12(sp)		
	stw	r15, 16(sp)		
	ldbio	r12, 12(r5)	/* Test pushbuttons.	*/
	stbio	r0, 12(r5)	/* Clear edge-detection register.	*/
	movia	r13, COUNTER		
	bne	r12, r0, CHECK		
	ldhio	r12, (r5)	/* Check if pushbutton is still pressed.	*/
	xori	r12, r12, 7	/* Invert for active low.	*/
	beq	r12, r0, DONE		
	movia	r14, tone_timer		
	ldhio	r15, (r14)	/* Check for timeout period	*/
	andi	r15, r15, 1	/* of the tone-timer.	*/
	beq	r15, r0, DONE		
	sthio	r0, (r14)		
	ldw	r14, (r13)	/* Use the counter to achive	*/
	addi	r14, r14, 1		
	movi	r15, 500	/* a desirable delay.	*/
	bge	r14, r15, COUNT		
	stw	r14, (r13)		
	br	DONE		
COUNT:	stw	r0, (r13)		
CHECK:	andi	r13, r12, 1	/* Is minute pushbutton pressed?	*/
	beq	r13, r0, HOURS	/* If not, check hours.	*/
	ldw	r11, (r10)	/* Load present time.	*/
	movi	r12, 60	/* Divide by 60 to determine	*/
	divu	r13, r11, r12	/* the number of hours.	*/
	mul	r14, r13, r12	/* Remainder of division operation	*/
	sub	r11, r11, r14	/* is the number of minutes.	*/
	addi	r11, r11, 1	/* Increment minutes.	*/
	blt	r11, r12, SAVEM	/* Save if less than 60,	*/
	mov	r11, r0	/* otherwise set minutes to 00.	*/

...and so on

## 11.10. The program of Figure 11.6 can be modified as follows:

```
#define
                                      minute_timer (volatile int *) 0x5000
#define
                                     tone_timer (volatile int *) 0x5020
#define
                                      sliders (volatile int *) 0x5040
#define
                                      pushbuttons (volatile int *) 0x5050
#define
                                      display (int *) 0x5060
#define
                                     LEDs (int *) 0x5070
#define
                                     speaker (int *) 0x5080
#define
                                      ADJUST(t, x) ((t + x) >= 1440) ? (t + x - 1440) : (t + x)
int
                                     actual_time, alarm_time, alarm_active, time, counter;
unsigned int
                                     flash;
   unsigned char table [16] = \{0x40, 0x79, 0x24, 0x30, 0x19, 0x12, 0x02, 0x78, 0x12, 0x02, 0x78, 0x12, 
            0x00, 0x18, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F, 0x3F};
   void initializeToneTimer()
            *(tone_timer + 2) = 0x0D40;
                                                                                                 /* Set the timeout period
            *(tone\_timer + 3) = 0x03;
                                                                                                 /* for continuous operation. */
            *(tone\_timer + 1) = 0x6;
                                                                                                 /* Start in continuous mode. */
   }
   void DISP(time)
                                                                                                 /* Get 7-segment patterns for display. */
            *display = table[time / 600] << 24
                           table[(time % 600) / 60] << 16
                           table[(time % 60) / 10] << 8 |
                           table[(time % 10)];
   }
   main()
            actual_time = alarm_time = alarm_active = 0;
            counter = flash = 0;
            initializeToneTimer();
            *(minute_timer + 1) = 0x6;
                                                                                                 /* Run in continuous mode. */
            while ((*sliders) == 0)
                  if (*tone_timer == 3)
                                                                                                 /* One ms elapsed. */
                           *tone\_timer = 0;
                                                                                                 /* Clear the TO bit. */
                           if (counter < 1000)
                                                                                                 /* Wait for one second. */
                                 counter = counter + 1;
                           else
                                 counter = 0;
                                 flash = flash ^{\wedge} 6;
                                                                                                 /* Flashing signal. */
                                  *LEDs = flash;
```

```
while (1)
       if (*minute_timer == 3)
                                         /* One minute elapsed. */
       {
                                         /* Clear the TO bit. */
               *minute\_timer = 0;
               actual_time = ADJUST(actual_time, 1);
       if ((*sliders \& 1) != 0)
                                         /* Check the alarm-on switch. */
               *LEDs = 7;
                                         /* Turn on the alarm LED. */
               if (actual_time == alarm_time)
                                         /* Start the alarm sound. */
                   alarm\_active = 1;
               else
                   alarm_active = alarm_active & (*sliders & 1);
               if (*tone_timer == 3)
                                         /* Generate the square wave. */
                   *speaker = (*speaker ^ 1) & alarm_active;
                   *tone\_timer = 0;
                                         /* Clear the TO bit. */
       }
       else
       {
               *LEDs = 6;
                                         /* Turn off the alarm LED. */
               alarm\_active = 0;
       if ((*sliders & 4) != 0)
                                         /* Check the set-the-time-of-day switch. */
               DISP(actual_time);
                                         /* Display the time of day. */
               if ((*(pushbuttons + 3) & 1) != 0) /* Set the minutes? */
                   actual_time = ADJUST(actual_time, 1);
               else if ((*(pushbuttons + 3) & 2) != 0) /* Set the hours? */
                   actual_time = ADJUST(actual_time, 60);
               *(pushbuttons + 3) = 0; /* Clear the edge-capture register. */
                                         /* Check the set-the-alarm-time switch. */
       else if ((*sliders & 2) != 0)
               DISP(alarm_time);
                                         /* Display the alarm time. */
               if ((*(pushbuttons + 3) \& 1) != 0) /* Set the minutes? */
                   alarm_time = ADJUST(alarm_time, 1);
               else if ((*(pushbuttons + 3) & 2) != 0) /* Set the hours? */
                   alarm_time = ADJUST(alarm_time, 60);
               *(pushbuttons + 3) = 0; /* Clear the edge-capture register. */
       }
       else
               DISP(actual_time);
                                         /* Display the time of day. */
}
```

11.11. It is only necessary to change the part (b) of the program in Figure 11.7. It can be modified as follows:

	movia ori sthio ori sthio	r6, tone_timer r7, r0, 0x0D40 r7, 8(r6) r7, r0, 0x03 r7, 12(r6)	/* Set the tone-timer period.	*/
	movi sthio	r7, 6 r7, 4(r6)	/* Start the tone-timer.	*/
	movia	r6, minute_timer	/* Address of minute-timer.	*/
	addi	r7, r0, 7	/* Start the timer.	*/
	sthio	r7, 4(r6)	, suite times.	•
	movi	r7, 1		
	wrctl	ienable, r7	/* Enable timer interrupts.	*/
	wrctl	status, r7	/* Enable external interrupts.	*/
	movia	r6, tone_timer	-	
	movi	r7, 1000	/* Time interval of 1 second.	*/
	mov	r8, r0	/* Use r8 as a counter.	*/
	movi	r9, 6	/* Pattern for vertical LEDs.	*/
FLASH:	stbio	r9, (r3)	/* Write to the vertical LEDs.	*/
	ldbio	r10, (r2)	/* If a slider switch is activated,	*/
	bne	r10, r0, TON	/* proceed with normal operation.	*/
SLID:	ldhio	r11, (r6)	/* Test if the tone-timer has	*/
	andi	r11, r11, 1	/* reached the timeout point.	*/
	beq	r11, r0, SLID		
	sthio	r0, (r6)	/* Clear the TO bit.	*/
	addi blt	r8, r8, 1 r8, r7, SLID	/* Increment the count.	*/
	mov	r8, r0	/* Reset the counter and	*/
	xori	r9, r9, 6	/* invert the LEDs signal.	*/
	br	FLASH		
TON:	movi	r6, 6	/* Turn on the two	*/
	stbio	r6, (r3)	/* vertical LEDs.	*/
LOOP:	movia call ldbio	r10, ACTUAL_TIME DISP r7, (r2)	/* Display the time of day.	*/
	andi beq	r11, r7, 1 r11, r0, NEXT	/* Check if alarm switch is on.	*/
	movi	r11,7	/* If yes, then turn on the	*/
	stbio	r11, (r3)	/* alarm LED.	*/
	movia	r9, ALARM_TIME		
	ldw	r11, (r9)	/* Have to compare alarm-time	*/
	ldw	r12, (r10)	/* with actual-time.	*/
	bne	r11, r12, NEXT	/* Should the alarm ring?	*/
	movia	r8, tone_timer	-	
	movi	r12, 1		

...and so on