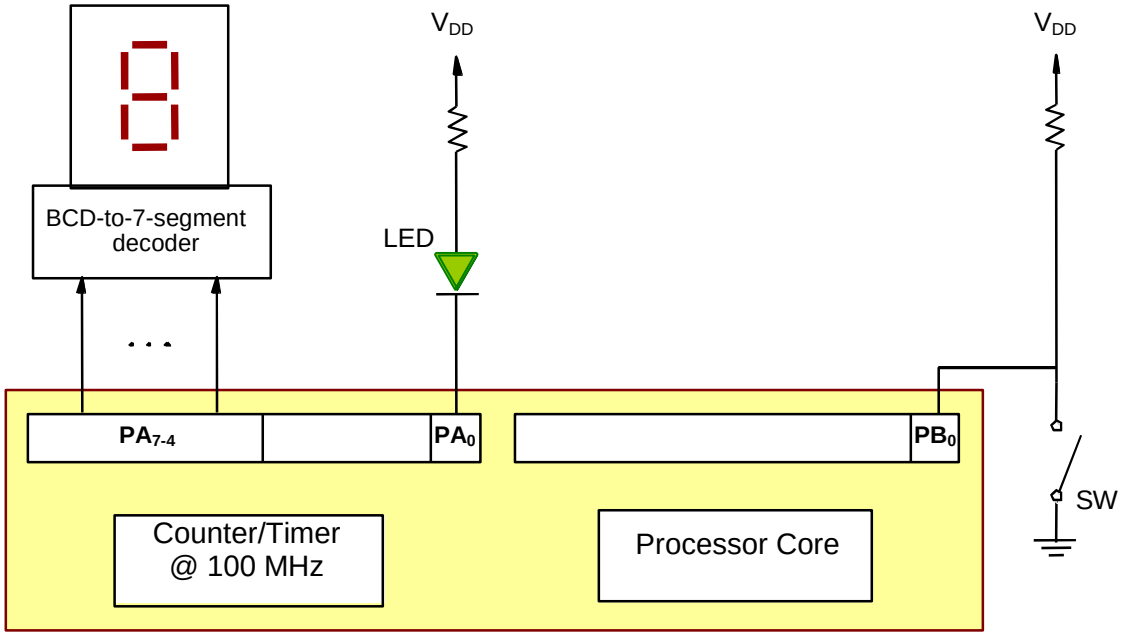
Solved Exercises 1

1. The textbook's microcontroller is used in a system below and is responsible for 2 tasks: (1) flipping the LED on/off state every time the **SW** key has been hit, and (2) incrementing the displayed digit every second, i.e., displaying the numerical sequence **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **0**, **1**, … . Write the corresponding C program, assuming that the **first task** is the main program, and the **second task** is an ISR, whose address is stored at memory location **0x20**. Also, assume that bit **6** of the processor status register (**PSR[6]**) is the processor's interrupt-enable bit, and **Port A** is always ready to receive data from the processor. Initially, the 7-segment display shows digit **0**, and the LED is off.

• *Main Program*: Every time the **SW** key is hit, i.e., pressed and then released (bit $\mathbf{PB_0}$ must first become 0 and then 1 again), the LED state must be flipped: if the LED is on, it must be turned off, and vice versa.

• *ISR*: The 100-MHz Counter/Timer must be configured to generate interrupts every second, and its ISR must increment the displayed digit. Incrementing **9** gives **0**.




```
#define PAOUT (volatile unsigned char *) 0xFFFFFFF1
#define PADIR (volatile unsigned char *) 0xFFFFFFF2
#define PBIN (volatile unsigned char *) 0xFFFFFFF3
#define PBDIR (volatile unsigned char *) 0xFFFFFFF5
#define CNTM (volatile unsigned int *) 0xFFFFFFD0
#define CTCON (volatile unsigned char *) 0xFFFFFFD8
#define CTSTAT (volatile unsigned char *) 0xFFFFFFD9
#define IVECT (volatile unsigned int *) (0x20)
```

```
interrupt void intserv();

unsigned char digit = 0;            /* Digit to be displayed */
unsigned char led = 0x1;            /* LED state: 0/1 = on/off */

int main() {
  *PADIR = 0xF1;                          /* Set Port A direction */
  *PBDIR = 0x00;                          /* Set Port B direction */
  *CTCON = 0x02;                          /* Stop Timer */
  *CTSTAT = 0x0;                          /* Clear "reached 0" flag */
  *CNTM = 100000000;                      /* Initialize Timer */
  *IVECT = (unsigned int *) &intserv;     /* Set interrupt vector */
  asm("MoveControl PSR,#0x40");           /* CPU responds to IRQ */
  *CTCON = 0x11;                          /* Enable Timer interrupts
                                           * and start counting */
  *PAOUT = 0x01;                          /* Display 0, turn LED off */

  while (1) {

    while ((*PBIN & 0x1) != 0);     /* Wait until SW is pressed */
    while ((*PBIN & 0x1) == 0);     /* Wait until SW is released */

    if (led == 0x1) led = 0x0;      /* If off, turn LED on */
    else led = 0x1;                 /* Else, turn LED off */

    *PAOUT = ((digit << 4) | led);  /* Update Port A */

    /* We can also put "*CTCON &= 0xEF;" before and "*CTCON |= 0x10;"
     * after the last statement, to make sure that intserv() is not
     * interfering with main() accessing shared digit/led/PAOUT */
  }

  exit(0);
}


interrupt void intserv() {
  *CTSTAT = 0x0;                    /* Clear "reached 0" flag */
  digit = (digit + 1)%10;           /* Increment digit */
  *PAOUT = ((digit << 4) | led);    /* Update Port A */
}
```

2. Solve Problem **10.7** from the textbook. NOTE: Connect Ports **A** and **B** to the four 7-segment decoders, letting $\mathbf{PA_{7-4}}$, $\mathbf{PA_{3-0}}$, $\mathbf{PB_{7-4}}$, and $\mathbf{PB_{3-0}}$ display the first, second, third, and fourth received digits, respectively. Assume that all four digits arrive immediately after the character **H** has been received.

```
#define   RBUF    (volatile unsigned char *) 0xFFFFFFE0
#define   SSTAT   (volatile unsigned char *) 0xFFFFFFE2
#define   PAOUT   (volatile unsigned char *) 0xFFFFFFF1
#define   PADIR   (volatile unsigned char *) 0xFFFFFFF2
#define   PBOUT   (volatile unsigned char *) 0xFFFFFFF4
#define   PBDIR   (volatile unsigned char *) 0xFFFFFFF5

char temp;
char digits[4];                          /* Buffer for received digits. */
int i;

void main()
{
    /* Initialize the parallel ports. */
    *PADIR = 0xFF;                       /* Configure Port A as output. */
    *PBDIR = 0xFF;                       /* Configure Port B as output. */

    /* Transfer the characters. */
    while (1)                            /* Infinite loop. */
    {
        while ((*SSTAT & 0x1) == 0);     /* Wait for a new character. */
        if (*RBUF == 'H')
        {
          for (i = 3; i >= 0; i--)
          {
            while ((*SSTAT & 0x1) == 0); /* Wait for the next digit. */
            digits[i] = *RBUF;           /* Save the new digit (ASCII). */
          }
          temp = digits[3] << 4;         /* Shift left first digit by 4 bits, */
          *PAOUT = temp | (digits[2] & 0xF);  /*   append second and send to A. */
          temp = digits[1] << 4;         /* Shift left third digit by 4 bits, */
          *PBOUT = temp | (digits[0] & 0xF);  /*   append fourth and send to B. */
        }
    }
}
```

3. Solve Problem **10.8** from the textbook. NOTE: Upon detecting the character **H**, the subsequent four digits have to be saved and displayed only when the fourth digit arrives. Interrupts must be used to detect the arrival of both **H** and the four digits (the ISR has to keep track of the received characters).

```
#define   RBUF      (volatile unsigned char *) 0xFFFFFFE0
#define   SCONT     (volatile unsigned char *) 0xFFFFFFE3
#define   PAOUT     (volatile unsigned char *) 0xFFFFFFF1
#define   PADIR     (volatile unsigned char *) 0xFFFFFFF2
#define   PBOUT     (volatile unsigned char *) 0xFFFFFFF4
#define   PBDIR     (volatile unsigned char *) 0xFFFFFFF5
#define   IVECT     (volatile unsigned int *) 0x20

  char temp;
  char digits[4];                                   /* Buffer for received digits. */
  int k;

  interrupt void intserv();

  void main()
  {
      /* Initialize the parallel ports. */
      *PADIR = 0xFF;                                /* Configure Port A as output. */
      *PBDIR = 0xFF;                                /* Configure Port B as output. */
      /* Initialize the interrupt mechanism. */
      *IVECT = (unsigned int *) &intserv;           /* Set the interrupt vector. */
      asm (”MoveControl     PSR, #0x40”);           /* Respond to IRQ interrupts. */
      *SCONT = 0x10;                                /* Enable receiver interrupts. */

      /* Transfer the characters. */
      k = 0;
      while (1);                                    /* Infinite loop */
  }

 /* Interrupt service routine. */
  interrupt void intserv()
  {
      if (k > 0)
      {
          k = k – 1;
          digits[k] = *RBUF;                        /* Save the new digit (ASCII). */
          if (k == 0)
          {
            temp = digits[3] << 4;                  /* Shift left first digit by 4 bits, */
            *PAOUT = temp | (digits[2] & 0xF);      /*   append second and send to A. */
            temp = digits[1] << 4;                  /* Shift left third digit by 4 bits */
            *PBOUT = temp | (digits[0] & 0xF);      /*   append fourth and send to B. */
          }
          else if (*RBUF == ’H’)
            k = 4;
      }
  }
```