Solution 2

1.

```
#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.

```
#define PBIN (volatile unsigned char *) 0xFFFFFFF3
#define PBOUT (volatile unsigned char *) 0xFFFFFFF4
```

```
#define PBDIR (volatile unsigned char *) 0xFFFFFFF5
#define PCONT (volatile unsigned char *) 0xFFFFFFF7
#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();

volatile unsigned char led = 0x4;    /* 0x0 = LED on, 0x4 = LED off */
volatile unsigned char digit = 0;    /* digit for display */

int main() {
  *CTCON = 0x2;                              /* Stop Timer (if running) */
  *PBDIR = 0xF4;                             /* Set Port B direction */
  *IVECT = (unsigned int *) &intserv;        /* Set interrupt vector */
  asm("MoveControl PSR,#0x40");              /* CPU responds to IRQ */
  *PCONT = 0x40;                             /* Enable PBIN interrupts */
  *PBOUT = 0x4;                              /* Turn off LED, display 0 */
  *CNTM = 100000000;                         /* 1-second timeout */
  *CTCON = 0x1;                              /* Start countdown */
  while (1) {
    *CTSTAT = 0x0;                    /* Clear "Reached 0" flag */
    while ((*CTSTAT & 0x1) == 0);     /* Wait until 0 reached */
    if (led == 0x4) led = 0x0;        /* If off, turn LED on */
    else led = 0x4;                   /* Else, turn LED off */
    *PBOUT = ((digit << 4) | led);    /* Update LED, same display */
  }
  exit(0);
}

interrupt void intserv() {
  unsigned char sample;
  sample = *PBIN & 0x3;               /* Read PBIN, isolate bits [1:0] */
  if (sample == 0x2) {                /* INC = 0 (increment), DEC = 1 */
    if (digit == 9) digit = 0;
    else digit = digit + 1;
  }
  else if (sample == 0x1) {           /* INC = 1, DEC = 0 (decrement) */
    if (digit == 0) digit = 9;
    else digit = digit - 1;
  }
  *PBOUT = ((digit << 4) | led);      /* Update display, same LED */
}
```

3.

The LCM (least common multiple) of all four periods is 90; hence, we only need to figure out our RM schedule in the time interval **[0, 90)**, after which it is repeated: