Solution 2

1. One of possible solutions is shown below.

```
#define PAOUT (volatile unsigned char *) 0xFFFFFFF1
#define PADIR (volatile unsigned char *) 0xFFFFFFF2
#define PBIN (volatile unsigned char *) 0xFFFFFFF3
#define PBOUT (volatile unsigned char *) 0xFFFFFFF4
#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();
volatile unsigned char digit1 = 0; /* DIGIT1 for display */
volatile unsigned char digit2 = 0; /* DIGIT2 for display */
volatile unsigned char leds = 0x1; /* LED1 on, LED2 off */

int main() {
  *PADIR = 0x6F;                    /* Set Port A direction */
  *PBDIR = 0xF0;                    /* Set Port B direction */
  *CTCON = 0x2;                     /* Stop Timer (if running) */
  *CNTM = 100000000;                /* Initialize: 1-s timeout */
  *CTSTAT = 0x0;                    /* Clear “Reached 0” flag */
  *IVECT = (unsigned int *) &intserv; /* Set interrupt vector */
  asm(“MoveControl PSR,#0x40”);     /* CPU responds to IRQ */
  *PAOUT = 0x20;              /* Initialize port A */
  *PBOUT = 0x00;              /* Initialize port B */
  *CTCON = 0x11;              /* Start Timer, enable interrupts */
  while (1) {
    while ((*PBIN & 0x01) != 0);    /* Wait for SW press */
    while ((*PBIN & 0x01) == 0);    /* Wait for SW release */
    leds ^= 0x1;                    /* Toggle LED flag
    *PAOUT ^= 0x60;                 /* Flip LED1/LED2 state
  }
  exit(0);
}

interrupt void intserv() {
  *CTSTAT = 0x0;              /* Clear “Reached 0” flag */
  if (leds == 0x1) {
    if (digit1 == 0) digit1 = 9;
    else digit1 = digit1 - 1; /* Decrement DIGIT1 */
    *PAOUT = (0x20 | digit1); /* Update port A, LED1 on, LED2 off */
  }
  else {
    if (digit2 == 0) digit2 = 9;
    else digit2 = digit2 - 1; /* Decrement DIGIT1 */
    *PBOUT = digit2 << 4;     /* Update port B */
  }
}
```

2. One of possible solutions is shown below.

```
#define PAIN (volatile unsigned char *) 0xFFFFFFF0
#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 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 digit1 = 0;        /* DIGIT1 for display */
volatile unsigned char digit2 = 0;        /* DIGIT2 for display */
volatile unsigned char leds = 0x1;        /* LED1 on, LED2 off */

int main() {
  *PADIR = 0x78;                          /* Set Port A direction */
  *PBDIR = 0xF5;                          /* Set Port B direction */
  *IVECT = (unsigned int *) &intserv;     /* Set interrupt vector */
  asm(“MoveControl PSR,#0x40”);          /* CPU responds to IRQ */
  *PCONT = 0x10;                          /* Enable PIN interrupts */
  *PAOUT = 0x0;                           /* Initialize port B */
  *PBOUT = 0x1;                           /* Initialize port A */
  *CTCON = 0x2;                           /* Stop Timer (if running) */
  *CNTM = 100000000;                      /* Initialize: 1-s timeout */
  *CTCON = 0x1;                           /* Start Timer */
  while (1) {
    *CTSTAT = 0x0;                        /* Clear “Reached 0” flag */
    while ((*CTSTAT & 0x1) == 0);         /* Wait until 0 reached */
    if (leds == 0x1) {
      digit1 = (digit1+1)%10;             /* Increment DIGIT1 */
      *PAOUT = digit1 << 3;               /* Update port A */
    }
    else {
      digit2 = (digit2+1)%10;             /* Increment DIGIT2 */
      *PBOUT = (digit2 << 4) | leds;      /* Update port B */
    }
  }
  exit(0);
}

interrupt void intserv() {
  if ((*PAIN & 0x80) == 0) return;  /* SW is pressed: ignore */
  else {                            /* SW is released: */
    leds ^= 0x5;                    /* Flip LEDs */
    *PBOUT = (digit2 << 4) | leds;  /* Update port B */
  }
}
```

3. The LCM (least common multiple) of all four periods is 120, i.e., we only need to determine our EDF schedule in the time interval **[0, 120)**, after which it is repeated.

EDF task priorities are: (1/30, 1/60, 1/90, 1/120) for T1 arriving at (0, 30, 60, 90); (1/40, 1/80, 1/120) for T2 arriving at (0, 40, 80); (1/50, 1/110) for T3 arriving at (0, 60); (1/100) for T4 arriving at (0).

t=0: T1
t=10: T2
t=20: T3
t=30: T1
t=40: T2
t=50: T4
t=60: T1 (T4 preempted)
t=70: T4
t=75: T3
t=85: T2
t=90: T1 (T2 preempted)
t=100:T2
t=105:Idle
t=120:Repeat…