Solution 1

1. There are many possible solutions. One of them is shown below.

```
#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 */
}
```

2. There are many possible solutions. One of them 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 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 */

int main() {
  *PADIR = 0x0F;                            /* Set Port A direction */
  *PBDIR = 0x81;                            /* 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 interrupts and
                                               start down-counting */
  *PAOUT = 0x0;                             /* Display 0 */
  *PBOUT = 0x80;                            /* Turn LED2 off, LED1 on */
  while (1) {
    while ((*PAIN & 0x80) != 0);     /* Wait until SW is pressed */
    while ((*PAIN & 0x80) == 0);     /* Wait until SW is released */
    *PBOUT ^= 0x81;                  /* Update Port B */
  }

  exit(0);
}

interrupt void intserv() {
  if (digit == 0) digit = 9;         /* Decrement digit */
  else digit--;
  *PAOUT = digit;                    /* Update Port A */
  *CTSTAT = 0x0;                     /* Clear "reached 0" flag */
}
```

3. Let **x** denote the I/O device activity percentage to be determined. Maximum I/O data rate for DMA transfer is $\mathbf{R_{I/O}/d_{I/O\text{-}DMA} = 1K}$ transfers/s. DMA cost: $\mathbf{(x*1K)(N_{DMA\text{-}start} + N_{DMA\text{-}end}) = x*2.4M}$ cycles/s.
Maximum I/O data rate for polling is $\mathbf{R_{I/O}/d_{I/O} = 128K}$ transfers/s. Polling cost: $\mathbf{(x*128K)N_{poll\text{-}ready} + ((1-x)*128K)N_{poll\text{-}not\text{-}ready} = x*51.2M + 51.2M}$ cycles/s.
We know that the DMA cost is 400 times cheaper than the polling cost; therefore, $\mathbf{400*(x*2.4M) = x*51.2M + 51.2M}$, which yields $\mathbf{x \approx 0.056}$ (i.e., **5.6%**).

(Note: $\mathbf{1K = 2^{10}}$ and $\mathbf{1M = 2^{20}}$.)