# Solution 1

**1.**

```
#define PBIN (volatile char *) 0xFFFFFFF3
#define PBOUT (volatile char *) 0xFFFFFFF4
#define PBDIR (volatile char *) 0xFFFFFFF5
#define PCONT (volatile char *) 0xFFFFFFF7
#define CNTM (volatile int *) 0xFFFFFFD0
#define CTCON (volatile char *) 0xFFFFFFD8
#define CTSTAT (volatile char *) 0xFFFFFFD9
#define IVECT (volatile int *) (0x20)

interrupt void intserv();

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

int main() {
  *PBDIR = 0xF4;                            /* Set Port B direction */
  *IVECT = (volatile 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) {
    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() {
  if ((*PBIN & 0x1) == 0) digit = (digit + 1)%10;  /* INC pressed */
  if ((*PBIN & 0x2) == 0) digit = (digit - 1)%10;  /* DEC pressed */
  *PBOUT = ((digit << 4) | led);    /* Update display, same LED */
}
```

**2.**

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

```
interrupt void intserv();

unsigned char led = 0x1;            /* 0x0 = LED on, 0x1 = LED off */
unsigned char digit = 0;            /* digit for display */

int main() {
  *PADIR = 0xFF;                    /* Configure Port A direction */
  *PBDIR = 0x0;                     /* Configure Port B direction */
  *CNTM = 100000000;                /* 100,000,000 cycles = 1 sec */

  *IVECT = (volatile int *) &intserv;     /* Set up interrupt vector */
  asm("MoveControl PSR,#0x40");           /* CPU responds to IRQ */
  *CTCON = 0x11;                          /* Enable timer interrupts
                                             and start countdown */
  *PAOUT = 0x1;                     /* Turn off LED, display 0 */
  while (1) {                       /* Infinite loop */
    while ((*PBIN & 0x1) != 0);       /* Wait for SW to be pressed */
    while ((*PBIN & 0x1) == 0);       /* Wait for SW to be released */
    digit = (digit + 1)%10;           /* Increment digit */
    *PAOUT = ((digit << 4) | led);    /* Update display, same LED */
  }
  exit(0);
}

interrupt void intserv() {
  if (led == 0x0) led = 0x1;          /* If on, turn LED off */
  else led = 0x0;                     /* If off, turn LED on */
  *PAOUT = ((digit << 4) | led);      /* Update LED, same display */
  *CTSTAT = 0x0;                      /* Clear "reached 0" flag */
}
```

**3.**

Maximum I/O rate for DMA transfer is $\mathbf{R_{I/O}/d_{I/O\text{-}DMA} = 1K}$ transfers/sec. DMA cost: $\mathbf{(0.1*1K)(N_{DMA\text{-}start} + N_{DMA\text{-}end}) = 240K}$ cycles/sec.

Maximum I/O rate for polling is $\mathbf{R_{I/O}/d_{I/O} = 128K}$ transfers/sec. Polling cost: $\mathbf{(0.1*128K)N_{poll\text{-}ready} + (0.9*128K)N_{poll\text{-}not\text{-}ready} = 55M}$ cycles/sec.

Thus, DMA is 235-times cheaper than polling. (Note: $\mathbf{1K = 2^{10}}$ and $\mathbf{1M = 2^{20}}$.)