Assignment 1

Due September 29, 14:59

1.

```
#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)
/* Define all neccesarry Ports Afresses */
interrupt void intserv();
unsigned char digit = 0;                              /* Digit to be displayed */
unsigned char led = 0x1;                              /* LED state: 0/1 = on/off */
int main() {
             *PBDIR = 0xF4;                           /* Set Port B direction 1111
                                                      X100 -> F4 */
             *CTCON = 0x02;                           /* Stop Timer */
             *IVECT = (unsigned int *) &intserv;      /* Set interrupt vector */
             asm(“MoveControl PSR,#0x40”);            /* CPU responds to IRQ */
             *SCONT = 0x10;                           /* Enable RBUF interrupts */
             *CTCON = 0x1;                            /* Start counting */
             *PBOUT = 0x04;                           /* Display 0, turn LED off */
while (1) {
       *CNTM = 100000000;                             /* Initialize Timer */
       *CTSTAT = 0x0;                                 /* Clear “reached 0” flag */
       *CTCON = 0x1;                                  /* Start countdown */
       while ((*CTSTAT & 0x1) == 0);                  /* Wait until 0 is reached */
       *CTCON = 0x2;                                  /* Stop countdown */
       if (led == 0x1 )
       {
             led = 0x0;                               /*switch led */
       }else{led = 0x1}

       *PBOUT = (unsigned char)((digit << led | 0x0); /* Turn on/off LED */

}
exit(0);
}
interrupt void intserv() {
*PBOUT = *RBUF;
if ( *PBIN & 0x2) == 0)){                      /*Dec pressed -> decrement digit */
       digit = ((digit -1) % 10 + 10) % 10;           /* -1 % 10 = 9 */
}else (if *PBIN & 0x1) == 0){
        digit = ((digit +1) % 10 + 10) % 10;
}                                              /* Inc pressed -> Increment digit */

       /* Probaly not working with unsigned char -> have to parse */

*PBOUT = ((digit << 4) | led);                        /* Update Port B */
}
```

2.

```
#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)
             /* Define all neccesarry Ports Afresses */
interrupt void intserv();
unsigned char digit = 0;                       /* Digit to be displayed */
unsigned char led = 0x01;                      /* LED1 state: on LED2 state: off */

int main() {
             *PADIR = 0x0F;                    /* Set Port A direction */
             *PBDIR = 0x81;                    /* Set Port B direction */
             *CTCON = 0x02;                    /* Stop Timer */
             *IVECT = (unsigned int *) &intserv;      /* Set interrupt vector */
             asm("MoveControl PSR,#0x40");     /* CPU responds to IRQ */
             *CNTM = 100000000;                /* Initialize Timer */
             *CTCON = 0x11;                    /* Enable Timer interrupts and start
      counting */
             *PAOUT = 0x0;                     /* Display Digit */
             *PBOUT = led;                     /* Initiate LEDs on Port B*/
while (1) {

      while ((*PAIN & 0x80) != 0);             /* Wait until SW is pressed */
      while ((*PBIN & 0x80) == 0);             /* Wait until SW is released */
      if (led == 0x01) led = 0x80;             /* If LED1 on and LED2 off, switch */
      else led = 0x01;                         /* Else, switch to LED1 on and LED2
off */
      *PBOUT = led;                            /* Update Port B */

}
exit(0);
}


interrupt void intserv() {
*CTSTAT = 0x0;                                 /* Clear "reached 0" flag */
digit = ((digit -1) % 10 + 10) % 10;           /* Decrement digit (-1 % 10 + 10 ) %
10-> 9 */
      /* Probaly not working with unsigned char -> have to parse */

*PAOUT = digit;                                /* Update Port A n */
}
```

3.

$R_{I/O}$ = 4 MB/s = **4×2^{20} B/s ≈ 4.19×10^{6} B/s (Bit per second)**
$d_{I/O\text{-}DMA}$ = 4 KB
$N_{DMA\text{-}start}$ = 1,600
$N_{DMA\text{-}end}$ = 800
$d_{I/O}$ = 32 B (Bits)
$N_{poll\text{-}ready}$ = 800
$N_{poll\text{-}not\text{-}ready}$ = 400

Searching for x : active percentage
y: inactive percentage

Accesing I/O device max rate :

$R_{I/O}$/ $d_{I/O}$ = **4.19×10^{6} B/s** / 32 B = 131,072 = 0,131 x 10^{6} times /second

➔ 0,131 x 10^{6} polls /second

DMA scenario:

RI/O / dI/O-DMA = 4.19×10^{6} B/s / 4KB = 1022.95 times /s

➔ **1022 accesses / s**

Cost of DMA:

Cdma = ($X\times\frac{Rio}{ddma}$)) × (**NDMA-start** + **NDMA-end**)

Cdma = ($X\times1022\ a/s$) × (**1600** + **800**)

Cdma = ($X\times1022\ a/s$) × (**2400**)

Cost of polling:

Cpoll = ($X\times\frac{Rio}{dio}$) × **Npoll-ready** + ((1-X) $\times\frac{Rio}{dio}$) × **Npoll-not-ready**

Cpoll = ($X\times0.131*10\hat{}6\times$ **800** + ((1-X) $\times0.131*10\hat{}6$) × **400**

$$\frac{Cpoll}{Cdma} = 400$$

$$\frac{Cpoll}{Cdma} = \frac{(X\times0.131*10\hat{}6)\times800+((1-X)\times0.131*10\hat{}6)\times400}{(X*1022)*(2400)}$$

$$400 = \frac{(X\times0.131*10\hat{}6)\times800+((1-X)\times0.131*10\hat{}6)\times400}{(X*1022)*(2400)}$$

$$400 = \frac{(X\times104800000+((1-X)\times52400000}{X*2452800}$$

$$981120000*X = (X\times104800000)+\left((1-X)\times52400000\right)$$

$$981120000*X = 104800000X+\ 52400000-52400000X$$

$$981120000*X = 52400000X+52400000$$

$$981120000*X = 52400000X+52400000$$

$$928720000*X = 52400000$$

$\mathbf{X} = \frac{52400000}{928720000}$ **= 0.05642 = 5.6 %**

-> at the percentage of 5.6% the DMA acces is already 400-times cheaper then the I/O acces