COE 306, Term 171

Introduction to Embedded Systems

Assignment# 2 Solution
Due date: Tuesday, Oct. 24, 2017

Q.1.

(a) Write a program that implements a decimal up-counter that counts from 0 to 9 and back to 0 (i.e., 0→1→2→…→9→0) and displays the result in a seven-segment display. Check the seven-segment display part number you have and then find out its data sheet to identify the configuration of pints. The MAN72A-like seven-segment display is shown below along with its pin configurations:

Pin No.	MAN3420A, 72A, 3620A, 3820A
1	Cathode A
2	Cathode F
3	Common Anode
4	No Pin
5	No Pin
6	Cathode D.P.
7	Cathode E
8	Cathode D
9	No Connection
10	Cathode C
11	Cathode G
12	No Pin
13	Cathode B
14	Common Anode

(b) Add two switches to your implementation and configure them as input and use them based on input polling such that whenever the first switch is pressed the speed of counting is increased by a factor of 2 relative to the latest speed and whenever the second switch is pressed the speed of counting is decreased by a factor of 2 relative to the latest speed. When the program starts initially the speed of counting should have a frequency of around 1 HZ.

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include <cr_section_macros.h>


int main(void) {

          LPC_GPIO0->FIODIR = (1<<11);
          LPC_GPIO0->FIOPIN = (1<<11);
```

```
            LPC_GPIO2->FIODIR |= 0x7f;
            LPC_GPIO0->FIODIR &= ~(1<<9);
            LPC_GPIO0->FIODIR &= ~(1<<8);
            int count;
            int i, b1, b2, lcount=10000000;
          while(1) {

                  b1 = LPC_GPIO0->FIOPIN & (1<<9);
                  b1 = b1 >>9;
                  if (!b1) {
                        lcount = lcount / 2;
                      if (lcount <= 312500) lcount=312500; // speed
increase upto 5 levels
                  }

                  b2 = LPC_GPIO0->FIOPIN & (1<<8);
                  b2 = b2 >>8;
                  if (!b2) {
                        lcount = lcount * 2;
                        if (lcount >= 32000000) lcount=32000000; //
speed decrease upto 5 levels
                  }




                  LPC_GPIO2->FIOPIN = ~decode(count);
                  for(i = 0; i < lcount; i++);

                  count=count+1;
                  if (count>9) count=0;


          }
          return 0 ;
}

int decode(int num){
      int res;          // GFEDCBA
      if(num == 0)
            res = 0x3F;
      else if(num == 1)
            res = 0x6;
      else if(num == 2)
            res = 0x5B;
      else if(num == 3)
            res = 0x4F;
      else if(num == 4)
            res = 0x66;
      else if(num == 5)
            res = 0x6D;
      else if(num == 6)
            res = 0x7D;
      else if(num == 7)
            res = 0x7;
      else if(num == 8)
            res = 0x7F;
      else
            res = 0x6F;
      return res;
```

```
}
```

(c) Use the two switches in the same way as used in part (b) to change the speed of counting but based on the use of interrupts instead of using polling. Write two input interrupt handlers, one for each switch, such that whenever any of the switches is pressed the handlers change the value of a global variable that changes the speed of counting.

```
#ifdef __USE_CMSIS
#include "LPC17xx.h"
#endif

#include <cr_section_macros.h>


int lcount=10000000;

void EINT0_IRQHandler()
{
      lcount = lcount / 2;
    if (lcount <= 312500) lcount=312500; // speed increase upto 5
levels
    printf("lcount: %d\n", lcount);


    for (int j=0; j<5000000; j++); // to avoid effect of bouncing
      LPC_SC->EXTINT |= 1; //reset interrupt

}

void EINT1_IRQHandler()
{
      lcount = lcount * 2;
      if (lcount >= 32000000) lcount=32000000; // speed decrease upto 5
levels
      printf("lcount: %d\n", lcount);


      for (int j=0; j<5000000; j++); // to avoid effect of bouncing
      LPC_SC->EXTINT |= 2; //reset interrupt
}

int main(void) {

          LPC_GPIO0->FIODIR = (1<<11);
          LPC_GPIO0->FIOPIN = (1<<11);
            LPC_GPIO2->FIODIR |= 0x7f;


            LPC_PINCON->PINSEL4     |=    (1<<20); // using pin p2.10
            LPC_PINCON->PINSEL4     |=    (1<<22); // using pin p2.11

            LPC_SC->EXTMODE  |= 3;  // set interrupt to edge level
            LPC_SC->EXTPOLAR |= 3; // set interrupt to rising edge

            NVIC_EnableIRQ(EINT0_IRQn);
```

```
            NVIC_EnableIRQ(EINT1_IRQn);


            int count;
            int i;
          while(1) {



                LPC_GPIO2->FIOPIN = ~decode(count);
                for(i = 0; i < lcount; i++);

                count=count+1;
                if (count>9) count=0;


          }
          return 0 ;
}

int decode(int num){
      int res;           // GFEDCBA
      if(num == 0)
            res = 0x3F;
      else if(num == 1)
            res = 0x6;
      else if(num == 2)
            res = 0x5B;
      else if(num == 3)
            res = 0x4F;
      else if(num == 4)
            res = 0x66;
      else if(num == 5)
            res = 0x6D;
      else if(num == 6)
            res = 0x7D;
      else if(num == 7)
            res = 0x7;
      else if(num == 8)
            res = 0x7F;
      else
            res = 0x6F;
      return res;
}
```

(d) Discuss your implementations in part (b) and (c) and which implementation is better and why?

When polling is used we can observe that sometimes when the switch is pressed it is not captured as this could be when the switch is pressed while CPU is executing the code waiting for the given delay or performing the display. On the other hand, based on using interrupt whenever the switch is pressed the interrupt handler code is executed which changes speed of count. Thus, no switch press is missed. However, we have to deal with

switch debouncing to ensure that the interrupt is not called multiple times when a switch is pressed.

(e) Include a link for a video demo for your implementations of part (b) and (c) on the LPCXpresso board.

Part (b) video demo link using polling:

https://www.dropbox.com/s/daaakjv0qnd13hm/Ass2_171_Polling.MOV?dl=0

Part (c) video demo link using interrupts:

https://www.dropbox.com/s/wtn3bqqlkmd1m4t/Ass2_171_Interrupts.MOV?dl=0