**ECE 447**

**Lab 1 – Segment Counter**

**QUY PHAM**

**G00459909**

1. **Introduction**:

In this lab 1, we will be using MP430 microcontroller to implement a two-digit counter and display it on two 7-segments. There are two buttons to control the counter. Button SW1 is for enabling the counter. Button SW2 is for toggling the count up or count down. When the counter reaches the maximum or minimum, it will switch the counting state (if the counter is counting up, it will switch to count down).

1. **Microcontroller Concepts:**

The microcontroller on-chip peripherals which we use:

* 7 pins of port P2 (H4) to display number on the oneth-digit 7-segment.
* 7 pins of port P6 (H8) to display number on the tenth-digit 7-segment.
* The on board power to power the two 7-segment display and the two buffers.

1. **Hardware Design:**



1. **Software Design:**

Start counter with zero

SW1 Press ?

No

Yes

No

Yes

Yes

No

Enable

Disable

Yes

Prev state = Enable ?

Display on the segments

Count down

Stop counter

Count up

Currently count up ?

SW2 Press ?

1. **Conclusion:**

The lab result was what we expected. The counter worked and the numbers were shown correctly on the 7-segment displays. I encountered some problems when working with the simulation and the hardware part.

For the simulation part, the problem was the delay loop. I should have used unsigned integer instead of just integer. That makes the condition for the loop always true and it looped forever.

For the hardware part, first problem was the segment A in one of the 7-segment. It was on all the time because one of the other wires in contact with it. Second problem was the timer. It was not approximate one second so the numbers shown on the 7-segment display too fast. To fix it, I added one more loop in the delay function to make it delay close to one second.

1. **Question:**
2. Show the equation you used to calculate the resistor value(s) for your circuit:
3. Show the maximum current draw for the segments of your 7-segment displays:
4. Describe the differences in your simulation-code and your hardware-code:

I changed the delay function so that it will delay an approximately one second. Everything else is the same.

1. Answer the following questions:
2. Which of the four buffers did you use, and why?

I used the 74HC245 buffer because it has 8 high current outputs. I used two of them for the two 7-segment displays.

1. Is the buffer you are using sinking current or sourcing current? Do you have a choice? Why?

Sinking current.

1. What is the maximum current capability of the buffer(s) you are using when it is sinking current and sourcing current?

Sinking current = 70mA and sourcing current = 35mA.

1. What are the pros and cons of using a resistor per LED segment?

* Pro:

1. **Source code:**

//constant and defines

#define WDTCTL \*(unsigned int \*)0x0120

#define WDTSTP 0x5A80

#define P1IN \*(unsigned int \*)0x0020 // buttons/switches

#define P1DIR \*(unsigned int \*)0x0022

#define P2IN \*(unsigned int \*)0x0028

#define P2OUT \*(unsigned int \*)0x0029

#define P2DIR \*(unsigned int \*)0x002A

#define P6IN \*(unsigned int \*)0x0034

#define P6OUT \*(unsigned int \*)0x0035

#define P6DIR \*(unsigned int \*)0x0036

#define BIT0 0x01

#define BIT1 0x02

#define BIT2 0x04

#define BIT3 0x08

#define BIT4 0x10

#define BIT5 0x20

#define BIT6 0x40

#define BIT7 0x80

// 7 segments

#define SEGA BIT0

#define SEGB BIT1

#define SEGC BIT2

#define SEGD BIT3

#define SEGE BIT4

#define SEGF BIT5

#define SEGG BIT6

// numbers from 0-9

#define NUM0 SEGA|SEGB|SEGC|SEGD|SEGE|SEGF

#define NUM1 SEGB|SEGC

#define NUM2 SEGA|SEGB|SEGD|SEGE|SEGG

#define NUM3 SEGA|SEGB|SEGC|SEGD|SEGG

#define NUM4 SEGB|SEGC|SEGF|SEGG

#define NUM5 SEGA|SEGC|SEGD|SEGF|SEGG

#define NUM6 SEGA|SEGC|SEGD|SEGE|SEGF|SEGG

#define NUM7 SEGA|SEGB|SEGC

#define NUM8 SEGA|SEGB|SEGC|SEGD|SEGE|SEGF|SEGG

#define NUM9 SEGA|SEGB|SEGC|SEGD|SEGF|SEGG

#define DELAY\_FACTOR 60000

#define DELAY\_FACTOR2 60000

#define COUNT\_MIN 0

#define COUNT\_MAX 99

int nums[] = {NUM0,NUM1,NUM2,NUM3,NUM4,NUM5,NUM6,NUM7,NUM8,NUM9};

unsigned int count;

int EnBtnState, CountUpBtnState;

int En, CountUp;

void Delay(int sec)

{

unsigned int k = 0;

unsigned int time = DELAY\_FACTOR\*sec;

for(unsigned int i=0; i<time; i++)

{

while (k<DELAY\_FACTOR2)

{

k ++;

}

}

}

int GetOneth(unsigned int num)

{

return (num >= 10) ? (num%100)%10 : num;

}

int GetTenth(unsigned int num)

{

return (num >= 10) ? (num%100)/10 : 0;

}

void CheckEnabledBtn()

{

if(!(P1IN & BIT0)) // active low

{

// check for raising edge

if(!EnBtnState)

{

En = (En == 1) ? 0 : 1;

EnBtnState = 1;

}

}

else

{

EnBtnState = 0;

}

}

void CheckCountUpBtn()

{

if(!(P1IN & BIT1)) // active low

{

// check for raising edge

if(!CountUpBtnState)

{

CountUp = (CountUp == 1) ? 0 : 1;

CountUpBtnState = 1;

}

}

else

{

CountUpBtnState = 0;

}

}

void GetCurCountNum()

{

if(En)

{

if(CountUp)

{

if(count < COUNT\_MAX)

{

count++;

}

else

{

count--;

CountUp = 0;

}

}

else // count down

{

if(count > COUNT\_MIN)

{

count--;

}

else

{

count++;

CountUp = 1;

}

}

}

}

void DisplayNum(unsigned int num)

{

P6OUT = ~nums[GetTenth(num)]; // active low

P2OUT = ~nums[GetOneth(num)];

}

int main(void) {

//STOP the watchdog timer

WDTCTL = WDTSTP;

P1DIR = 0; // set as input buttons

P2DIR = 0x007F; // set which bit to be outputs to 7 seg

P6DIR = 0x007F;

// int counter

count = COUNT\_MIN;

En = 0; // enabled toggle btn state

CountUp = 1; // count up/down toggle btn state

EnBtnState = 0;

CountUpBtnState = 0;

// button S1 is bit 0, S2 = bit 1 in P1IN

while(1)

{

CheckEnabledBtn();

CheckCountUpBtn();

GetCurCountNum();

DisplayNum(count);

Delay(1);

}

}