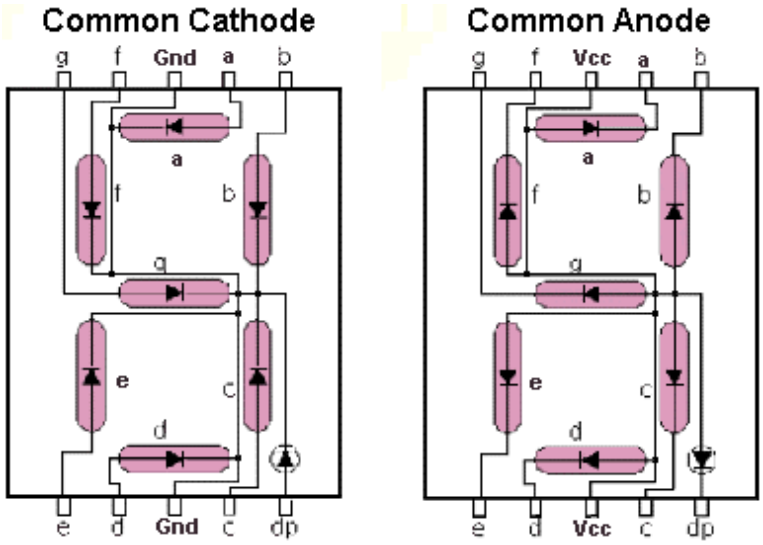
Embedded Project Hunter

August 28, 2015August 28, 2015

Control Seven Segment Display with MSP430G2553

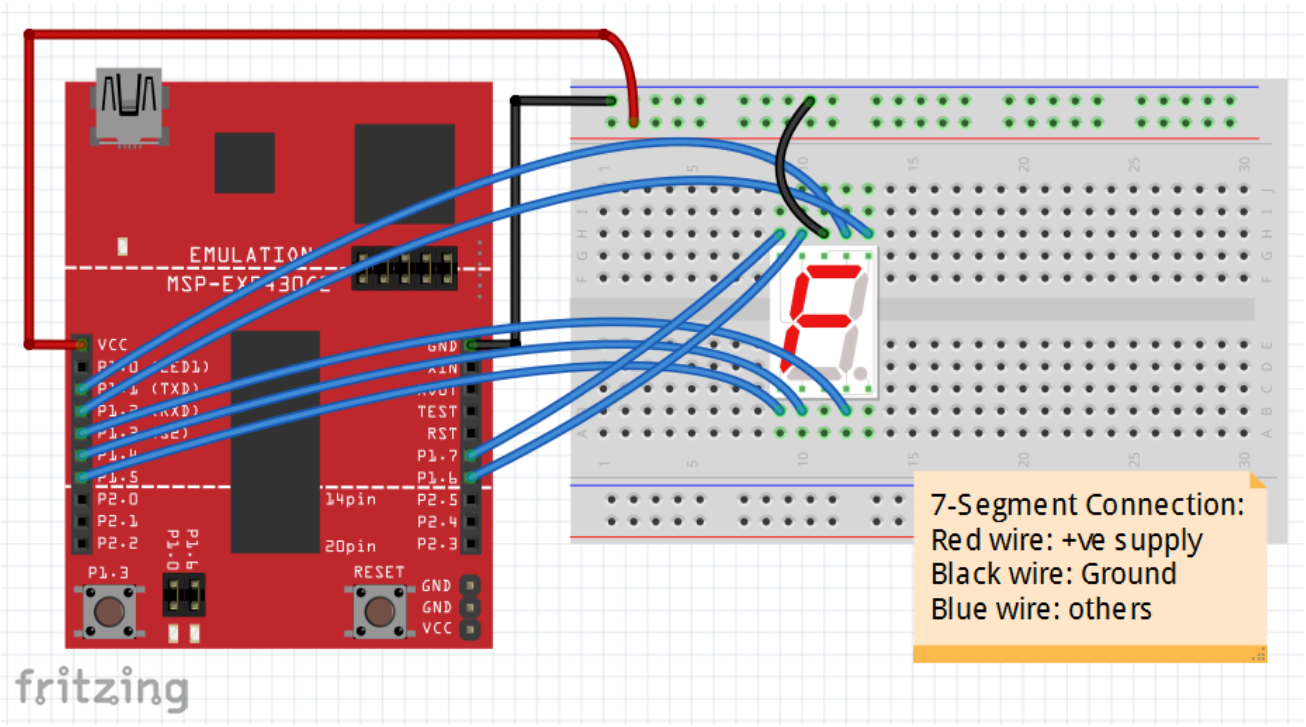
Indicators and displays are very important in projects. It allows us to know whether the data (input) has been processed correctly or not. The previous chapter introduce the ways to control simple LEDs. There are much more awesome displays in the market that you can find easily.

Seven-Segment display got their name because of it consists of seven segments LED that can be controlled to display variety of numbers and pattern. So how many LEDs are there in the seven-segment? Seven… Nope… In total there are eight LEDs which includes the small dots at the bottom right. Also, we have two types of seven-segment – Common Cathode and Common Anode. Common Cathode means all the LEDs share the same ground, while LEDs on common anode seven-segment display share the same power supply as shown in the figure below.



(https://embeddedprojecthunter.wordpress.com/wp-content/uploads/2015/08/7-segment.gif)

In this chapter, we are going to use common cathode seven-segment display. Basically, we are just going to on and off the LEDs just like what we do in the previous chapter, but in a more systematic way.



(https://embeddedprojecthunter.wordpress.com/wp-content/uploads/2015/08/seven-segment-connection-with-msp430g2553.png)

Note: I have tried connecting the 7-segment directly to the MSP430 mcu pins and it works perfectly. It depends on the components you used. Nevertheless, all the outputs should be connected to resistors to limit the current and prevent the 7-segment from burning. Also, Vcc is not connected directly to the 7-segment because the voltage needed for 7-segment will be supplied by the mcu.

Code:

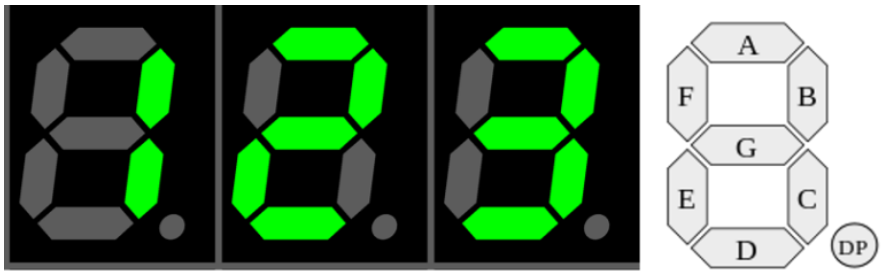
```
#include &amp;lt;msp430g2553.h&amp;gt;

#define a BIT1
#define b BIT2
#define c BIT3
#define d BIT4
#define e BIT5
#define f BIT6
#define g BIT7

void main()
{
    WDTCTL = WDTPW + WDTHOLD;
    P1DIR = a+b+c+d+e+f+g;

    while(1)
    {
        P1OUT = b+c;
        __delay_cycles(500000);
        P1OUT = a+b+g+e+d;
        __delay_cycles(500000);
        P1OUT = a+b+g+c+d;
        __delay_cycles(500000);
    }
}
```

Explanation:



(https://embeddedprojecthunter.wordpress.com/wp-content/uploads/2015/08/seven-segment-pin-number.png)

For instance, you wish to display “1” on the 7-segment, then only LED B and C will be turned on. For displaying “2”, LED A, B, D, E, and G will be turned on. For displaying “3”, LED A, B, C, D, and G will be turned on. Same thing happen if you want to display any other numbers.

In the code, it is made to be more systematic, whereby we do not need to write P1OUT = BIT1 + BIT2 + BIT3 + BIT4……. and so on to turn on and off the LEDs. In this example, we use #define to represent BIT1 with “a”, BIT2 with “b” and the list continues. Hence, our code will looks more organised and easier to interpret. For example, P1OUT = b + c; simply means that LED B and C are turned on, other pins are off, which eventually displaying “1” on the 7-segment.

Expected Output:

It continually display from 1 to 3 then start all over again and again and again (since it is in while loop).

Seven-segment Extra Training

Task 1: Modify the code to make the 7-Seg to count from 0-9 continuously.

Task 2: Is there any way of reducing the number of lines in the code?? For sure, there is a way!!

By using array!! Single dimension array is sufficient enough to do the work. As for now, for the sake of simplicity and to meet the purpose of learning, the sample code is given.

Task 3: How to make the 7 segment to change the number every time you pressed the button? Referring to the previous examples, try to write your own code before looking at the answer code.

Answer for Task 1:

```
#include <msp430g2553.h>

#define a BIT1
#define b BIT2
#define c BIT3
#define d BIT4
#define e BIT5
#define f BIT6
#define g BIT7

void main()
{
    WDTCTL = WDTPW + WDTHOLD;
    P1DIR = a+b+c+d+e+f+g;

    while(1)
    {
        P1OUT = a+b+c+d+e+f;
        __delay_cycles(500000);

        P1OUT = b+c;
        __delay_cycles(500000);

        P1OUT = a+b+g+e+d;
        __delay_cycles(500000);

        P1OUT = a+b+g+c+d;
        __delay_cycles(500000);

        P1OUT = f+g+b+c;
        __delay_cycles(500000);

        P1OUT = a+f+g+c+d;
        __delay_cycles(500000);

        P1OUT = a+f+e+d+c+g;
        __delay_cycles(500000);

        P1OUT = a+b+c;
        __delay_cycles(500000);

        P1OUT = a+b+c+d+e+f+g;
        __delay_cycles(500000);

        P1OUT = a+b+c+d+f+g;
        __delay_cycles(500000);
    }
}
```

Remarks:

Finally some hands-on work can be done! This code is for common cathode 7-Seg display. Good Luck Trying! Isn't it easy?

Answer for Task 2:

```
#include <msp430g2553.h>

#define a BIT1
#define b BIT2
#define c BIT3
#define d BIT4
#define e BIT5
#define f BIT6
#define g BIT7

int display[10]={a+b+c+d+e+f, b+c, a+b+g+e+d, a+b+g+c+d, f+g+b+c, a+f+g+c+d, a+f+e+d+
int n = 0;

void main()
{
    WDTCTL = WDTPW + WDTHOLD;
    P1DIR = a+b+c+d+e+f+g;

    while(1)
    {
        P1OUT = display[n];
        __delay_cycles(500000);
        n++;
        if (n==10)
        {
            n=0;
        }
    }
}
```

Remarks:

Bonus for all of you~~ Simplified and more efficient code for displaying 0 to 9 by using array!! Remember~ By producing same output, the shorter your code, the better you are!!

(This code is for common cathode 7-Seg display.)

Answer for Task 3:

```
#include <msp430g2553.h>

#define a BIT0
#define b BIT1
#define c BIT2
#define d BIT4
#define e BIT5
#define f BIT6
#define g BIT7

int display[10]={a+b+c+d+e+f, b+c, a+b+g+e+d, a+b+g+c+d, f+g+b+c, a+f+g+c+d, a+f+e+d+
int n = 0;

void main()
{
    WDTCTL = WDTPW + WDTHOLD;
    P1DIR = a+b+c+d+e+f+g;
    P1REN = BIT3;
    P1OUT = BIT3;

    while(1)
    {
        if((P1IN & BIT3)!=BIT3)
        {
            __delay_cycles(220000);
            P1OUT &= ~(a+b+c+d+e+f+g);
            P1OUT |= display[n];
            n++;
            if (n==10)
            {
                n=0;
            }
        }
    }
}
```

Remarks:

Control button is added. Noted that P1.3 is connected to the button, thus the same pin cannot be used to control the 7-segment. Hence, in this case, P1.0 is used. When you pressed the button, the number increase by 1. It is now become a digital counter!! Well done!! But it is using polling method, try it using interrupt method instead~

(This code is for common cathode 7-Seg display.)

Published by

embeddedprojecthunter

View all posts by embeddedprojecthunter
4 Comments

4 thoughts on “Control Seven Segment Display with MSP430G2553”

1. **Julián**
   July 13, 2016 at 12:23 am
   Perfect!!!!!!!!

   REPLY
   1. **embeddedprojecthunter**
      July 2, 2017 at 1:41 am
      Thanks for your kind comment, Julian. If you need any extra help, we are ready to assist. Cheers. 🙂

      REPLY
2. **Steve**
   March 22, 2018 at 6:52 pm
   How do you get two seven segment displays on at the same time? Or even better for a temperature controller?

   REPLY
   1. **embeddedprojecthunter**
      June 29, 2018 at 12:32 pm
      You can use two GPIO Pins to connect with the ground pin of 7 segments, switching them alternatively.

      REPLY