Digital Stopwatch

Program the ATmega Board to track elapsed time to 1/10th of a second.

Site: iCODE
Course: Machine Science Guides (Arduino Version)
Book: Digital Stopwatch
Printed by: Ivan Rudnicki
Date: Wednesday, July 30, 2014, 03:34 PM

Contents

Digital Stopwatch

In this project, you will use the ATmega Board to create a digital stopwatch. Your stopwatch will be accurate to within 1/10th of a second and feature Start, Stop, and Reset functions, just like the one in Figure 1.

Figure 1. A digital stopwatch.

Building the Stopwatch Circuit

The stopwatch code requires two button switches for start, stop, and reset controls. Figure 2 shows how to connect the two button switches.

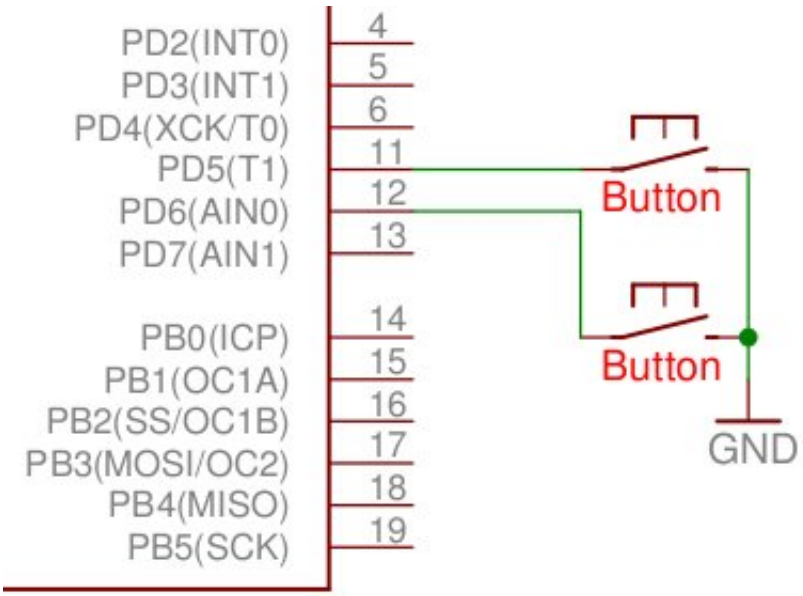

Figure 2. Schematic for adding buttons.

Storing and Displaying Time

In the first code example, you will create an array, called time, with four values: hours, minutes, seconds, and tenths of seconds. You will then write code to display these four values in a standard stopwatch format: 00:00:00.0.

1. Start a code file called time_display.c.
2. Type or paste the following code into the programming window:

```
#include <mxapi.h>
#include <lcd.h>

char time[4]={0,0,0,0};       //Declare array to store hours, minutes,
seconds, and tenths

int main(void)
{
    lcd_init();
    lcd_decimal(SECOND_LINE+3, time[0], 2);  //Display hours
    lcd_text(SECOND_LINE+5, ":");
    lcd_decimal(SECOND_LINE+6, time[1], 2);  //Display minutes
    lcd_text(SECOND_LINE+8, ":");
    lcd_decimal(SECOND_LINE+9, time[2], 2);  //Display seconds
    lcd_text(SECOND_LINE+11, ".");
    lcd_decimal(SECOND_LINE+12, time[3], 1); //Display tenths of seconds
    while(1==1);
}
```

3. Compile and test your new code.

Incrementing the Time Values

As time elapses, the values for hours, minutes, seconds, and tenths of a second increase in an orderly way. When the tenths of a second value reaches 9, the seconds value increases by 1, and the tenths value gets reset to 0. Likewise, when the seconds value reaches 59, the minutes value increases by 1 and the seconds get reset to 0. The code example below expresses this relationship among the the hours, minutes, seconds, and tenths of a second value.

1. Rename your code file increase_time.c.
2. Modify your code file, as follows:

```
#include <mxapi.h>
#include <lcd.h>

char time[4]={0,0,0,0};        //Declare array to store hours, minutes,
seconds, and tenths

int main(void)
{
    lcd_init();
    while(1==1)
    {
        lcd_decimal(SECOND_LINE+3, time[0], 2);  //Display hours
        lcd_text(SECOND_LINE+5, ":");
        lcd_decimal(SECOND_LINE+6, time[1], 2);  //Display minutes
        lcd_text(SECOND_LINE+8, ":");
        lcd_decimal(SECOND_LINE+9, time[2], 2);  //Display seconds
        lcd_text(SECOND_LINE+11, ".");
        lcd_decimal(SECOND_LINE+12, time[3], 1); //Display tenths of
seconds

        time[3]=time[3]+1;                       //Increment tenths of
seconds
        if(time[3]==9)
        {
            time[2]=time[2]+1;                   //Increment seconds
            time[3]=0;
        }
        if(time[2]==59)
        {
            time[1]=time[1]+1;                   //Increment minutes
            time[2]=0;
        }
        if(time[1]==59)
        {
            time[0]=time[0]+1;                   //Increment hours
            time[1]=0;
        }
        delay_ms(100);                           //Delay 1/10th of a
second
    }
}
```

3. Compile and test your new code.

Using Functions to Clean Up the Code

Since your stopwatch code will frequently involve displaying and increasing the time values, it is helpful at this stage to write two functions that take care of these tasks.

1. Rename your code file time_functions.c.
2. Modify your code file as follows:

```
#include <mxapi.h>
#include <lcd.h>

char time[4]={0,0,0,0};        //Declare array to store hours, minutes,
seconds, and tenths


void display_time()
{
    lcd_decimal(SECOND_LINE+3, time[0], 2);  //Display hours
    lcd_text(SECOND_LINE+5, ":");
    lcd_decimal(SECOND_LINE+6, time[1], 2);  //Display minutes
    lcd_text(SECOND_LINE+8, ":");
    lcd_decimal(SECOND_LINE+9, time[2], 2);  //Display seconds
    lcd_text(SECOND_LINE+11, ".");
    lcd_decimal(SECOND_LINE+12, time[3], 1); //Display tenths of seconds
}

void increment_time()
{
    time[3]=time[3]+1;                       //Increment tenths of
seconds
    if(time[3]==9)
    {
        time[2]=time[2]+1;                   //Increment seconds
        time[3]=0;
    }
    if(time[2]==59)
    {
        time[1]=time[1]+1;                   //Increment minutes
        time[2]=0;
    }
    if(time[1]==59)
    {
        time[0]=time[0]+1;                   //Increment hours
        time[1]=0;
    }
    delay_ms(100);                           //Delay 1/10th of a second
}


int main(void)
{
    lcd_init();
    while(1==1)
    {
```

```
        display_time();                         //Call display_time function
        increment_time();                       //Call increment_time
function
    }
}
```

3. Compile and test your new code.

Programming Start, Stop, and Reset

In the next code example, three modes of operation are defined for the stopwatch: an initial *waiting* state, where the display reads 00:00:00.0; a *running* state, in which the time values are constantly incremented and displayed; and a *stopped* state, where the time values are displayed but not incremented. A switch statement is used to cycle through the three states, based on input from the two button switches.

1. Rename your code file button_control.c.
2. Modify your code file as follows:

```
#include <mxapi.h>
#include <lcd.h>

#define waiting 0             //Define three operating states
#define running 1
#define stopped 2

char time[4]={0,0,0,0};

char mode;                    //Declare a variable called mode

void display_time()
{
    lcd_decimal(SECOND_LINE+3, time[0], 2);
    lcd_text(SECOND_LINE+5, ":");
    lcd_decimal(SECOND_LINE+6, time[1], 2);
    lcd_text(SECOND_LINE+8, ":");
    lcd_decimal(SECOND_LINE+9, time[2], 2);
    lcd_text(SECOND_LINE+11, ".");
    lcd_decimal(SECOND_LINE+12, time[3], 1);
}

void increment_time()
{
    time[3]=time[3]+1;
    if(time[3]==9)
    {
        time[2]=time[2]+1;
        time[3]=0;
    }
    if(time[2]==59)
    {
        time[1]=time[1]+1;
        time[2]=0;
    }
    if(time[1]==59)
    {
        time[0]=time[0]+1;
        time[1]=0;
    }
    delay_ms(100);
}
```

```
int main(void)
{
    lcd_init();
    input_pin(PORT_D5);
    pullup_on(PORT_D5);
    input_pin(PORT_D6);
    pullup_on(PORT_D6);

    while(1==1)
    {
        switch(mode)
        {
            case waiting:
            time[0]=0;                          //Set all time values to 0
            time[1]=0;
            time[2]=0;
            time[3]=0;
            display_time();
            while(pin_value(PORT_D5)==1);       //Wait for start button to be
pressed
            mode++;                             //Increment mode
            break;

            case running:
            while(pin_value(PORT_D6)==1)        //Until stop stop/reset
button is pressed
            {
                increment_time();               //Increment time values

                display_time();                 //Display time values
            }
            mode++;                             //Increment mode
            break;

            case stopped:
            delay_ms(300);
            while(pin_value(PORT_D6)==1);       //Wait for stop/reset button
            mode=0;                             //Set mode to 0
            break;
        }
    }
}
```

3. Compile and test your new code.

Calibrating the Stopwatch

With the code you have written, your stopwatch may not be perfectly accurate. However, this is easy to adjust by changing the duration of the delay statement in the increment_time function, as shown in the code excerpt below. A longer delay will slow down your stopwatch, while a shorter delay will speed it up. (In theory, the delay should be slightly shorter than 100 milliseconds, since some time is consumed in executing the code.)

1. Rename your code file final_calibration.c.
2. Check your stopwatch against a clock or wristwatch and calibrate accordingly.

```
void increment_time()
{
    time[3]=time[3]+1;
    if(time[3]==9)
    {
        time[2]=time[2]+1;
        time[3]=0;
    }
    if(time[2]==59)
    {
        time[1]=time[1]+1;
        time[2]=0;
    }
    if(time[1]==59)
    {
        time[0]=time[0]+1;
        time[1]=0;
    }
    delay_ms(100);            //Adjust this duration to calibrate
stopwatch
}
```

3. Compile and test your new code.