NANYANG POLYTECHNIC
EGR204 Microcontroller Applications

Laboratory Session 8

Course: Diploma in Robotics and Mechatronics

Module: EGR204 Microcontroller Applications

Experiment: 8

Title: Using the 8051 for counter and timer operations (Stop Watch)

Objective:

- The students will learn how to use the 8051 Timer to write a stop watch program in 'C' program for the 8051 microcontroller.

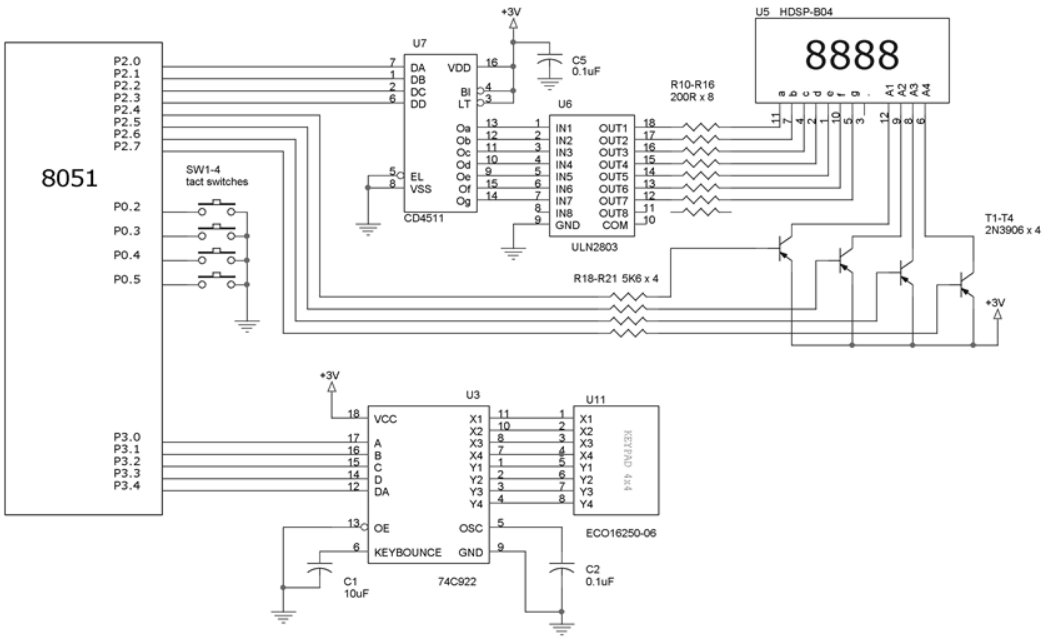
Learning Objectives:

- Learn how to use the 8051 timer to generate accurate timing.
- Write a stop watch program.

1. Introduction

Figure 1-1 shows how the 8051 is interfaced to a set of push buttons and a 4-digit display. This circuit will be used as the stop watch. Three buttons will be involved in the design of the stop watch: (1) the start button, (2) the stop button, and (3) the reset button.

Figure 1-2
The Stop Watch Circuit



2.1 Exercise 1: Running 4-Digit Display

Run the program in listing 2-1. At what rate is the display running?

Study the C statement below. If the timer is loaded with 0xD8F0, how long does it take the timer to overflow?

```
TH1=0xDB;
TL1=0xFF;
while (TF1==0)
{
   display(x);
}
```

Calculate the number of steps it will take to count from 0xDBFF to overflow.

0x10000 – 0xDBFF = 0x2401

0x2401 = 9216 (in decimal)

It will take 0x2401 steps for the timer to count from 0xd8f0 to overflow. 0x2710 is 10,000 in decimal. Since the 8051 is running off 11.059 MHz, each timer increment is 1.085 us. Therefore, 9,216 steps will be 10 ms.

So? At what rate is the display running? Is it accurate?

Listing 2-1

```
#include <f200.h>

void delay(unsigned long duration)
{
   while((duration--)!=0);
}

void setSystem();

void mux_display(a,b,c,d)
unsigned char a,b,c,d;
{
   ……..
}

void display(unsigned int number)
{
   ……
}


void main()
{
   unsigned int x=0;

   setSystem();

   TMOD=0x10;
   TR1=1;
   TF1=0;

   for (;;)
   {
      TH1=0xDB;
      TL1=0xFF;
      while (TF1==0)
      {
         display(x);
      }
      TF1=0;
      x++;
      if (x>9999) x=0;
   }
}
```

2.2 Exercise 2: Running 4-Digit Display At A Slower Rate

Change the main routine of listing 2-1 to the one shown in listing 2-2. Run the program. At what rate is the display running now?

The timer still overflow at a rate of 10 ms because the load value of 0xDBFF in still the same as in listing 2-1. However, for every timer overflow, a variable count is increment instead of x. Therefore, count is increment every 10 ms.

When count reaches 10, x is incremented. So at what rate is x incremented?

Listing 2-2

```
void main()
{
   unsigned int x=0;
   unsigned char count=0;

   setSystem();

   TMOD=0x10;
   TR1=1;
   TF1=0;

   for (;;)
   {
      TH1=0xDB;
      TL1=0xFF;
      while (TF1==0)
      {
         display(x);
      }
      TF1=0;
      count++;
      if (count==10)
      {
         count=0;
         x++;
         if (x>9999) x = 0;
      }
   }
}
```

3.1 Assignment 1: Accurate 1 sec timer

Modify the program in listing 2-2 so that your 4-digit display runs at a rate of **exactly 1 sec.**

3.2 Assignment 2: Design a Stop Watch

Use the program in listing 3-1 to design a stop watch timer program. The stop watch will have three buttons:

START
STOP
RESET

Listing 3-1
Stop Watch

```
void main()
{
   unsigned char mode=STOP;
   setSystem();

   TMOD=0x10;
   TR1=0;
```

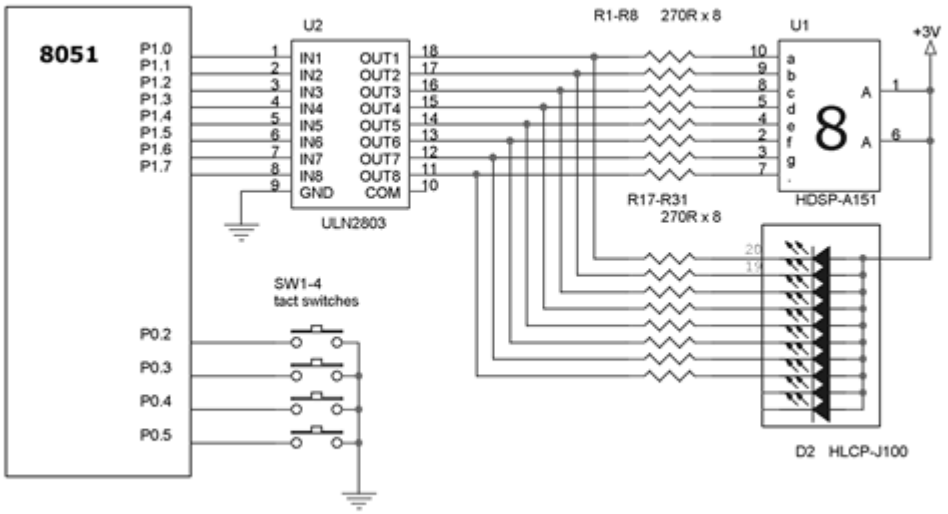
```
    TF1=0;
    TH1=0xDB;
    TL1=0xFF;

    for (;;)
    {
       display(x);

       if (P02==0) // Indicate RUN mode & start timer.
       if (P03==0) // Indicate Stop mode & stop timer.
       if (P04==0) // Reset display value to 0.

       if (mode==RUN)
       {
          if (TF1==1)
          {
             // Reload timer.
             // Reset overflow flag.
             // Increment count.
             // Check count value.
             // If 1 sec pass, reset count value
             //      and increment x value.
          }
       }
    }
}
```

Figure 3-1
Using the LED Circuit for Chronometer



3.3 Assignment 3: Add a Chronometer to the Stop Watch

Implement a LED chronometer in the stop watch you have designed in assignment 2. One LED will light up every 100 ms. When 1 sec has elapsed, the 4-digit display increment by one and the LEDs reset.

You would actually need 10 LEDs but make do with the 8 LEDs connected to PORT0. Make use of the led_table routine given in listing 3-2.

What is the use of a chronometer?

Listing 3-2

```
void led_table(unsigned char x)
{
   switch(x)
   {
      case 0: P1 = 0; break;
      case 1: P1 = 0x01; break;
      case 2: P1 = 0x03; break;
      case 3: P1 = 0x07; break;
      case 4: P1 = 0x0f; break;
      case 5: P1 = 0x1f; break;
      case 6: P1 = 0x3f; break;
      case 7: P1 = 0x7f; break;
      case 8: P1 = 0xff; break;
      default: P1= 0xff;
   }
}
```

4. Program Analysis

Analyse the program in listing 4-1. What do you see if you run the program. The mux_display routine in the program is same as the one you use in earlier lab to display the number in a,b,c,d on the 4-digit display.

Listing 4-1

```
void main()
{
   unsigned char hr=12,min=0,sec=0,sub_sec=0;
   unsigned char a,b,c,d;
   setSystem();

   TMOD=0x01;
   TF0=0;
   TR0=1;
   for(;;)
   {
      mux_display(a,b,c,d);
      if (TF0==1)
      {
         TF0=0;
         TH0=0xDB;
         TL0=0xFF;
         sub_sec++;
         if (sub_sec==100)
         {
            sec++;
            if (sec==60)
            {
               sec=0;
               min++;
```

```
                if (min==60)
                {
                    min=0;
                    hr++;
                    if (hr>12) hr=1;
                }
                a = hr/10;
                b = hr%10;
                c = min/10;
                d = min%10;
            }
        }
    }
}
```