EMBEDDED C Examples

Introduction

To look at some simple examples we are going to make use of an 8051 microcontroller simulator (EDSIM51). By Simulator I mean some software that simulates the actual physical hardware. The 8051 is a commonly used 8-bit microcontroller. The simulator is JAVA based and so is platform independent. I will put a copy on ICE, but you can download from this URL: https://www.edsim51.com/.

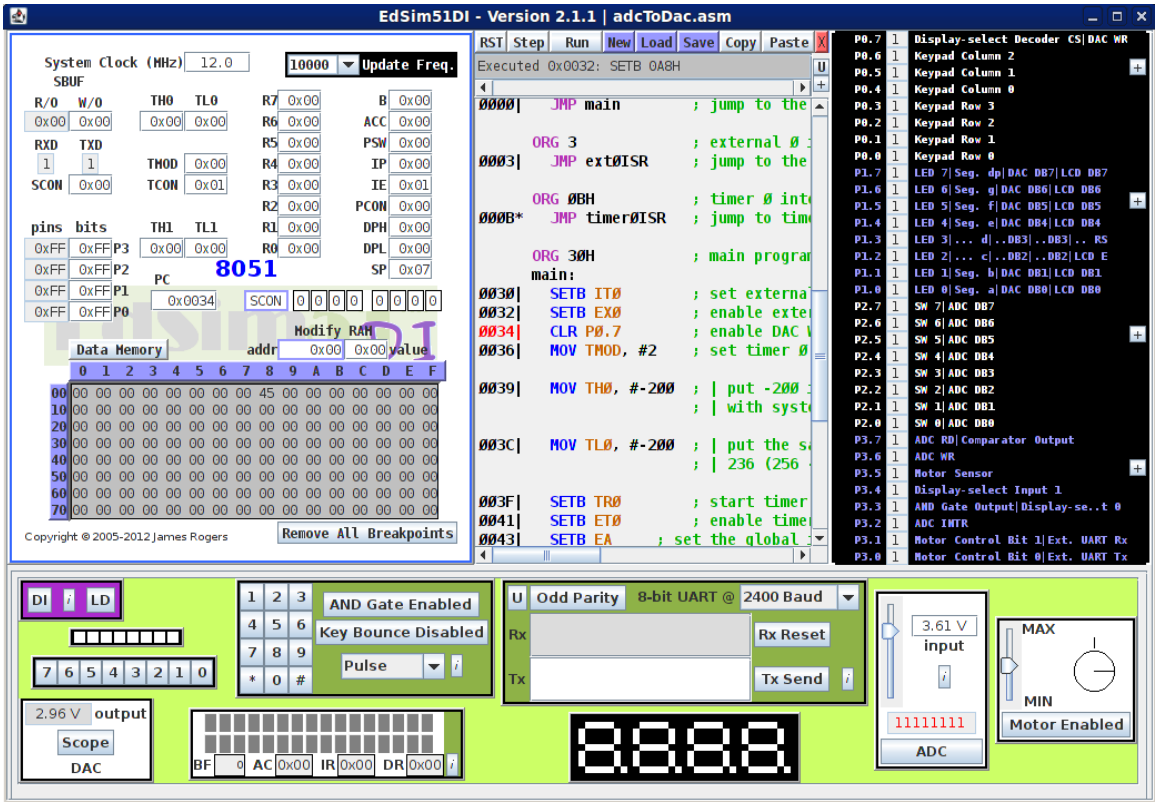

Figure 1. Screenshot of the EdSim51 Simulator.

I. The top left box gives the user access to all the 8051's registers, data memory and code memory.

II. In the centre is a textbox where the user either loads an assembly program or writes the code directly. Shown above is a program being single-stepped (execution is currently at location 0034H in code memory - hence that line is highlighted).

III. On the right is a list of the 32 port pins and what each one is connected to. The current value of the port pin is displayed here.

IV. The bottom panel shows all the peripherals that are connected to the 8051.

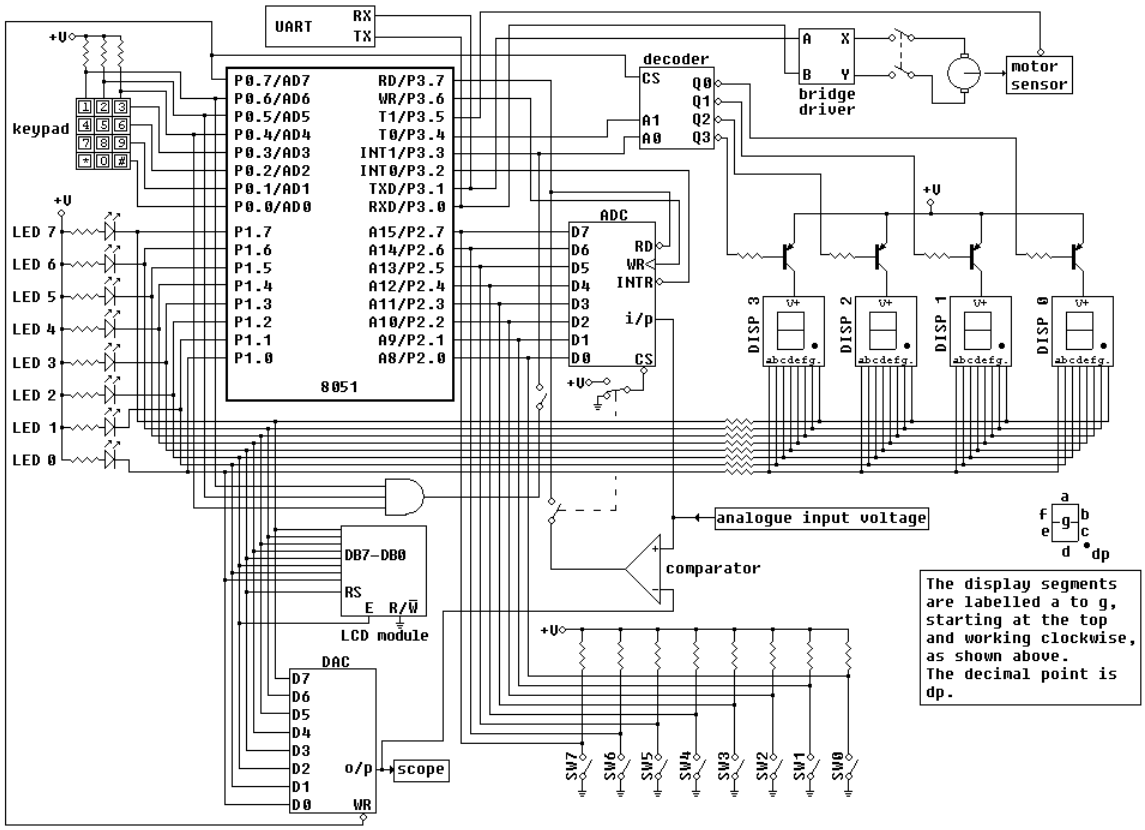

Figure 2 EdSim51 Simulator Peripheral Logic Diagram

Figure 2 shows how the peripheral devices are connected to the microcontroller.

Example 1 Flashing LED 1

Consider the C programme given here:

```
#include <reg51.h>
//header file
sbit portBit = P1^1;  //The sbit data type is used to access the bit addressable memory for Port 1_1.
void main()
{
        //main function starts
        unsigned int i;

        //Initializing Port1 pin1
```

```
        portBit = 0; //Make Pin1 o/p


        while (1) {
                //Infinite loop main application
                //comes here
                for(i=0;i<1000;i++)
                        ; //delay loop


                portBit = ~portBit;
                //complement Port1.1 (note the use of the ~ symbol to do this
                //this will blink LED connected on Port1.1
        }
}
```

The first problem we have is that we cannot load a C program into the EDSim51 simulator. We need first to generate a HEX file. We can do this using the Keil µVision compiler (version 4 or 5). You can use the compilers in the EEE Computer labs for this, but you can also download Keil µVision 5 from the Keil website (http://www2.keil.com/mdk5/uvision/) and run it on your own computer. Unfortunately only Windows executables are available. You will need to download and install MDK-Arm (Lite edition) and, in addition, the CX51 development tool which has compilers for the 8051.

Note: Creating a HEX file will be fully demonstrated during the lecture. You will need to take notes, or work out for yourself how to do it. But you should remember that building a project will not automatically create a HEX file from the C program. To do this you should follow the procedure below once you have written the C code:

1. Open the project in the Keil IDE.
2. Click the drop-down menu **Project**, then select **Options for Target**
3. Select the **Output** tab
4. Check **Create HEX File**
5. Click **OK**
6. Now when you build (or rebuild) all targets the project will create a HEX file, provided there are no errors.

The HEX file can be loaded into the EDSim51 simulator and this will be demonstrated in class.

Your first task is to get the LED 1 blinking using the program given. The delay used is a bit arbitrary, and you can adjust this to your liking by modifying the code.

Q1. What does the code do?

Q2. How would you modify the code for the LED connected to port 1 pin 2 ?

Example 2. Seven (7) Segment Display Up-count

```
/*****************************************************************************
**************

 * Description: This file contains the program to demonstrate a single digit up counter.

*****************************************************************************
************/

#include "reg51.h"

void DELAY_ms(unsigned int ms_Count)

{

   unsigned int i,j;

   for(i=0;i<ms_Count;i++)

   {

      for(j=0;j<100;j++);

   }

}


int main()

{

  char seg_code[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};

  char cnt=0;

    while(1)

   {

           for(cnt=0;cnt<=9;cnt++)     // loop to display 0-F

           {

              P1 = seg_code[cnt];       // Send the segment_code of number to be displayed */

               DELAY_ms(1);

                     }

           }

}
```

1. Compile this code and run it on the EDSim51 simulator.
2. Modify it to count from 9 to 0.

The seven segment displays are controlled by the decoder. The output of the decoder (Q0-Q1) will decide which 7 Segment display is active. The input to the decoder comes from pins 3_3 and 3_4 of the microcontroller.

3. Modify the program so the DISP 0 (the 7 segment display on the right) counts up.

Unfortunately the decoder only enables one 7 segment display at a time, so it is difficult to count in decimal beyond 9. You could make the delays very short and switch between enabling the different 7-segment displays to try and make it look continuous, and indeed in practice this would happen but the emulator doesn't work in real time. You can try it. It could look something like this:

```
/*********************************************************************************
**************

 * Description: This file contains the program to demonstrate a two digit up counter (i.e. up to 99).

*********************************************************************************
************/

#include "reg51.h"

sbit portBit1 = P3^3;  //The sbit data type is used to access the bit addressable memory for Port 3_3.

sbit portBit2 = P3^4;   //The sbit data type is used to access the bit addressable memory for Port 3_4.

sbit portBit3 = P0^7;   //This turns the decoder off and on

void DELAY_ms(unsigned int ms_Count)

{

   unsigned int i,j;

   for(i=0;i<ms_Count;i++)

   {

      for(j=0;j<10;j++);

   }

}

int main()

{

  char seg_code[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};

  char cnta=0;

            char cntb=0;
```

```
            char a=0;
  while(1)
 {
    for(cntb=0;cntb<=9;cntb++)
          {
           for(cnta=0;cnta<=9;cnta++)          // loop to display 0-9
                    {
                     for (a=0; a<=20; a++)
                              {
                              portBit3 = 0;
                              portBit1 = 1; //Make Pin3 o/p
                              portBit2 = 0; //Make Pin4 o/p
                              P1 =seg_code[cntb];
                              portBit3 = 1;
                              DELAY_ms(1);
                              portBit3 = 0;
                              portBit1 = 0; //Make Pin3 o/p
                              portBit2 = 0; //Make Pin4 o/p
                              P1 = seg_code[cnta];        // Send the segment_code of number to be
displayed
                              portBit3 = 1;
                              DELAY_ms(1);
                              }
                    }
          }
}
}
```

Example 3 Using Keypad 1 to turn on LED 1

```
#include<reg51.h>
sbit r1=P0^3;
sbit c1=P0^6;
```

```
sbit LED=P1^1;

void main()

{

c1=0;

while(1)

{

If (r1==0)

{

LED=0;

}

}

}
```

This example will turn on light 1 when you press 1 on the keyboard. **Note: You need to make sure the AND Gate is enabled on the keyboard.**

Discussion

From these example you should now have a good idea how we can control peripheral devices by controlling the outputs of the microcontroller pins.