HW 1: Basic I/O, MCU Programming, Interrupts, ADC

Souradeep Bhattacharya
EE128 Section: 001

October 29, 2017

Q1 MCU and Embedded Hardware

1. A CPU consists of several registers, an ALU, and a control unit. The control unit consists of a Program counter and status registers.
2. The program counter holds the address of the currently executing instruction.
3. Computer buses allow the CPU to talk to various other devices, such as IO controllers and memory, usually at high speed.
4. You can add two 32 bit numbers on a 16-bit CPU. This can be done by adding the LSBs and then adding the MSBs with the overflow. Then the data can be read and written with multiple load and store operations.

Q2 MCU Programming and I/O Operations

Given the following code:

```
#include <mc9s12dg256.h>

#define NR_ITEMS 10
#define IOREGS_BASE 0x0000
#define _IO8(off) *(unsigned char volatile *)(IOREGS_BASE + off)

#define PORTX _IO8(2)
#define DDRX _IO8(0)

// Interfacing with 7-segment Display
void display_output(int val) {
    unsigned char decoder[NR_ITEMS] = {0x7E, 0x30, 0x6D, 0x79, 0x33, 0x5B, 0x5F, 0x70, 0x7F,
        0x7B };

    PORTX = decoder[val];
}
void main(void) {
    long i;
    int val;
    DDRX=0xff; // Output mode

    val = 0;

    while (1) {
        display_output(val);
        val = (val + 1) % NR_ITEMS;
```

```
26              for (i = 0; i < 100000; i++); // S/W Delay
27          }
28      }
```

1. No, this code will not work. The maximum range for an int(which is 16-bits on this platform) is less then 100000. This will result in the code never leaving the for loop on line 26.
2. The code will work fine, because the display can only display from 0 to 9, and the char range encompasses this.
3. During the preprocessing stage of compiling every time it sees that value it will replace it with 10.
4. It seems to be using PORTA and DDRA to control the 7 segment display.
5. The volatile keyword indicates to the compiler that the variable may change at anytime; not relying on the code around it.

Q3 Exceptions and Interrupts

1. A maskable interrupt is one that can be ignored by the CPU. It must expressly have been enabled in order for the interrupt to impact the CPU. Non-maskable interrupts are ones that can not be ignored(Reset, unimplemented instruction trap.)
2. Interrupts are good for multiple IO events. They can be prioritized and they can also be ignored(disabled).
3. The reset interrupt is used to restore the MCU to a working state. This is commonly used to recover a locked out MCU.
4. What the currently executing instruction is. It is generally a bad idea to context switch in the middle of an instruction. This can be really bad during a memory read or write operation. It can also result because the the CPU is servicing a higher priority interrupt. Finally it can also result because the interrupt has been disabled.

Q4 I/O Ports

1. Switch debouncing is necessary because the switch is a mechanical component. A single push can cause the switch's contacts to make and break contacts several times in a fraction of a second. With the high speed of MCU's these can result in multiple input events even though the button was only pushed once. To get around this programmers implement switch debouncing.
2. If we don't put a current limiting resistor between the MCU's pins and the LED we can burn out the LED by over driving it with the current provided by the MCU's output pin.
3. The current drawn by the pin is equal to the current flowing through the resistor which is given by the following:

$$i = \frac{5V - 2.2V}{1500\Omega}$$
$$i = 18.67mA$$

Q5 Analog to Digital Conversion

1.

$$\frac{V_{RH} - V_{RL}}{2^n}$$

$$\frac{8-2}{2^{10}}$$

$$5.86mV$$

2.

$$V_k = V_{RL} + \frac{(V_{RH} - V_{RL}) * k}{2^n - 1}$$

$$V_k = 2 + \frac{6 * k}{2^{10} - 1}$$

$$V_5 = 2 + \frac{6 * 5}{1023} = 2.03V$$

$$V_{110} = 2 + \frac{6 * 110}{1023} = 2.64V$$

$$V_{250} = 2 + \frac{6 * 250}{1023} = 3.47V$$

$$V_{800} = 2 + \frac{6 * 800}{1023} = 6.69V$$

3.

$$\frac{1}{4000} * 12 = 0.003s$$