Introduction to Microcontrollers

John C Luciani Jr.

April 27, 2009

1 Microcontrollers

A microcontroller (μC) is a system on a chip (SOC). For most microcontroller applications the majority of the circuitry required to build a system will be contained in the μC package. The designer adds additional components specific to the new design and a small number of basic components (resistors, capacitors, crystal).

Some of the peripheral functions that are integrated into a μC are —

Digital I/O Provides the ability to control and read digital outputs. A digital output can be used to turn on and off an LED or motor. A digital input can be used to determine whether a switch is on or off. A digital line is either on or off.

A/D Conversion The ability to measure an analog voltage. Analog voltages are output by a variety of sensors (e.g. temperature, light, sound, motion). The output of a sensor would be connected to the μC A/D input.

D/A Conversion The ability to output analog voltages.

Serial Interface The serial interface is used to transfer data (1) between devices within a system and (2) between systems.

Timers Provides the ability to measure durations between events. A timer can also be used to create a real-time clock.

Flash Memory Used to store program code and data.

Static RAM Used to store program data.

More advanced microcontrollers would add –

- Larger Flash, more SRAM. Some may add a DRAM controller.
- Additional A/D inputs, D/A outputs, Digital I/O, timers and serial interfaces.
- Radio.

2 Programming

A typical μC program consists of two parts – a bootloader and the main application. A simple bootloader will do some basic initialization and then call the main application. Some bootloaders, like the Arduino bootloader, enable new versions of the main application to be downloaded through a serial port.

Listing 1 contains the listing for the μC version of **hello world**. All of the code listings use the Arduino software libraries.

Listing 1: Hello World

```
//
// Blink
//
// Modified from Arduino Blink
//
// Turns an LED on for one second,
// then off for one second,
// and so on...

// LED connected to digital pin 7

int ledPin = 7;

// run once, when the program starts

void setup()
{
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
}

// run over and over again

void loop()
{
  // turn the LED on
  // then wait for a second
  digitalWrite(ledPin, HIGH);
  delay(1000);
  // turn the LED off
  // then wait for a second
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

3 Hardware

The hardware section provides a simplified description of the basic modules in a typical μC like the common Atmel ATmega168 (Atmel, 2009).

3.1 Digital Input

A digital input measures a voltage. If the voltage is greater than 2V (3.3V system) the value read is considered a `1` or `HIGH`. If the voltage is less than 1V (3.3V system) the value read is considered a `0` or `LOW`. Voltage levels between 1V and 2V are indeterminant.

Listing 2: Reading a Digital Input

```
#define IN1 PIN2

boolean in1;

pinMode(IN1, INPUT);

in1 = digitalRead(IN1);
```

3.2 Digital Output

A digital output is used to output a voltage. Setting an output to a value of `1` produces a voltage greater than 2.3V (3.3V system). Setting an output to a value of `0` produces a voltage less than 0.6V (3.3V system).

Listing 3: Writing a Digital Output

```
#define OUT1 PIN3

pinMode(OUT1, OUTPUT);

digitalWrite(OUT1, HIGH);
```

3.3 A/D Conversion

The input to the A/D converter is a voltage (the A in A/D) and the output is a binary number (the D in A/D). Most μC's have multiple inputs (the ATmega168 has six). You connect a wire from the signal to be measured to the A/D input pin on the μC.

Listing 4 contains the code for reading input 0 of the D/A converter.

Listing 4: Measuring a Voltage

```
#define CH0 0
int reading;

reading = analogRead(CH0);
```

3.4 D/A Conversion

The input to the D/A converter is a binary number (the D in D/A) and the output is a voltage (the A in D/A). The ATmega168 does not have a D/A converter.

Listing 5 contains pseudo-code for outputing the voltage that corresponds to the value `255` to channel 1 of an D/A converter.

Listing 5: Outputing a Voltage

```
#define CH1   1

int voltage_code = 255;

analogWrite(CH1, voltage_code);
```

3.5 Serial

The ATmega168 has three types of serial communication ports (one of each):

UART (SCI) The UART is used to communicate to terminal devices such as an RS-232 port or a USB port. The wires are transmit and receive. The RS-232 port has additional control wires but the minimum number of communication wires is two.

SPI four wire system used to communicate with peripheral circuits within a system. The wires are data in, data out, clock and enable.

I^{2}C two wire system used to communicate with peripheral circuits within a system. The wires are data and clock. Similar function to SPI but reduces physical board space requirements. Trade-off is reduced board size (cost) with bandwidth.

Listing 6 lists the code required to read a byte from the UART. Listing 7 lists the code required to write a byte to the UART. Examples of communicating to an SPI and I^{2}C port are beyond the scope of this introduction.

Listing 6: Reading a byte from the UART

```
int in_byte;

if (Serial.available() > 0)
  in_byte = Serial.read();
```

Listing 7: Writing a byte to the UART

```
int out_byte = 65; // 'A'

Serial.print(out_byte, BYTE);
```

3.6 Timers

Implementing timers usually involves interrupt handling and is beyond the scope of this introduction.

3.7 Homework

Once you are familiar with the μC hardware and the Arduino tools you may to want to add your own circuitry. Maybe you want to have two μC's communicate. Maybe you want to have your μC communicate with your PC. An excellent reference is "Making Thing Talk" by Tom Igoe(Igoe, 2007). This book can be used as a "cookbook" as well as a detailed reference.

References

Atmel. (2009). 8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash. (Retrieved March 14, 2009, from http://www.atmel.com/dyn/resources/prod_documents/doc8025.pdf)

Banzi, M. (2009, January). Arduino Introduction. (Retrieved April 18, 2009, from http://arduino.cc/en/Guide/Introduction)

Igoe, T. (2007). *Making Things Talk*. Sebastopol, CA, USA: O'Reilly Media.