The Basic of AVR C Programming

Bits and Bytes

A bit represents one of two possible states: 1 or 0 (on/off, set/clear, high/low). Several bits together represent numerical values in binary, where each bit is one binary digit. An AVR microcontroller groups 8 bits together to form one byte wtih the Least Significant Bit (LSB) on the right. Each bit is numbered, starting from 0, at the LSB.

Consider the decimal number `15` for example. `15` represented in 8-bit binary is `00001111`. The 4 least significant bits 0, 1, 2, and 3, are set.

The following C code shows 3 ways in which a variable might be initialized to the decimal unsigned integer of 15.

```
uint8_t   a = 15;         /* decimal */

uint8_t   b = 0x0F;       /* hexidecimal */

uint8_t   c = 0b00001111; /* binary */
```

Bitwise Operations

Since individual bits often have a significant meaning when programming AVR
microcontrollers, bitwise operations are very important.

A single ampersand character (&) is the bitwise AND operator in AVR C.

```
uint8_t   a = 0xAA; /* 10101010 */

uint8_t  b = 0x0F; /* 00001111 */

uint8_t  = a & b;  /* 00001010 */
```

A single pipe character (|) is the bitwise OR operator
in AVR C.

```
uint8_t  a = 0xAA; /* 10101010 */

uint8_t   b = 0x0F; /* 00001111 */

uint8_t   c = a | b;  /* 10101111 */
```

The caret character (^) is the bitwise XOR operator
in AVR C.

```
uint8_t  a = 0xAA; /* 10101010 */

uint8_t  b = 0x0F; /* 00001111 */

uint8_t  c = a ^ b;  /* 10100101 */
```

A tilde character (~) is the NOT operator in AVR C.

```
uint8_t  a = 0xAA; /* 10101010 */

uint8_t  b = ~a;   /* 01010101 */
```

Two less-than symbols (<<) is the left shift operator and two greater-than symbols (>>) is the right shift operator in AVR C. The right side of the operator is the number of bits to shift.

```
uint8_t  a = 0x99; /* 10011001 */

uint8_t  b = a<<1; /* 00110010 */

int8_t    c = a>>3; /* 00010011 */
```

Clearing and Setting Bits

Setting and clearing a single bit, without changing any other bits, is a common task in AVR microcontroller programming. You will use these techniques over and over again.

When manipulating a single bit, it is often necessary to have a byte value in which only the bit of interest is set. This byte can then be used with bitwise operations to manipulate that one bit. Let’s call this a **bit value mask**. For example, the bit value mask for bit 2 would be `00000100` and the bit value mask for bit 6 would be `01000000`.

Since the number `1` is represented in binary with only bit 0 set, you can get the bit
value mask for a given bit by left shifting `1` by the bit number of interest. For example, to get the bit value mask for bit 2, left shift `1` by 2.

To **set a bit** in AVR C, OR the value with the bit value mask.

```
uint8_t  a = 0x08; /* 00001000 */

                          /*set bit 2 */

a |= (1<<2);      /* 00001100 */
```

Use multiple OR operators to set multiple bits.

```
uint8_t  a = 0x08;   /* 00001000 */

                          /*set bits 1 and 2 */

a |= (1<<2)|(1<<1); /* 00001110 */
```

To **clear a bit** in AVR C, NOT the bit value mask so that the bit of interest is the only bit cleared, and then AND that with the value.

```
uint8_t  a = 0x0F; /* 00001111 */

                      /* clear bit 2 */

a &= ~(1<<2);     /* 00001011 */
```

Use multiple OR operators to clear multiple bits.

```
uint8_t  a = 0x0F;      /* 00001111 */

                          /* clear bit 1 and 2 */

a &= ~((1<<2)|(1<<1)); /* 00001001 */
```

To **toggle a bit** in AVR C, XOR the value with the bit value mask.

```
uint8_t  a = 0x0F;  /* 00001111 */

                       /* toggle bit 2 */

a^= (1<<2);      /* 00001011 */

A^= (1<<2);      /* 00001111 */
```

The above instruction used to be on the website

http://www.avr-tutorials.com/functions/useful-embedded-avr-c-fuctions.

One may visit their website for more instruction.

AVR C Library

In order to use the register names defined in Atmega328 such as PORTx, SREG, SPH, SPL, etc., specific header files such as <avr/io.h> <util/delay.h> <avr/interrupt.h> have to be included. For a detail description of all the header files in the AVR C Library, please check the online document at

http://www.nongnu.org/avr-libc/user-manual/modules.html

Create a new C project in Atmel Studio, please follow the instructions at

http://www.avr-tutorials.com/avr-studio-6/creating-new-avr-c-project-avr-studio-6

You need to select the Atmega328 as a target device!!