Using avr-libc

avr-libc provides a subset of the standard C library for AVR microcontrollers.

avr-libc also provides necessary startup code for most applications.

Details (oh, are there details) are linked from the class web page at:

Many categories of libc modules to make embedded programming easier:
- Delay loops
- EEPROM access
- Character operations
- Integer conversions
- Math operations
- String operations
- Interrupts
More examples using avr-libc

Character operations...

```c
#include <string.h>

// right justify results on the LCD
for (i = 0; i <= (14 - (strlen(lcd_str) + strlen(lcd_str1))); i++)
    char2lcd(' ');  
```

Fake floating point math and print results

```c
#include <stdlib.h>

div_t fp_adc_result, fp_low_result;  // double fp_adc_result;

fp_adc_result = div(adc_result, 205);  // should be 204.8 to be exact
low  = fp_adc_result.rem;
high = fp_adc_result.quot;
fp_low_result = div((low*100), 205);  // set decimal fraction
itoa(fp_low_result.quot, lcd_str, 10);  // convert to ascii string
itoa(high, lcd_str1, 10);  // convert to ascii string
```
Using avr-libc

Examples using avr-libc

Character conversion:

```c
#include <stdlib.h>

itoa(frequency, lcd_freq_str, 10); // convert integer to ASCII string
```

Setup a interrupt service routine. The ISR created will save registers for you and will disable interrupts when entering the ISR.

```c
#include <avr/interrupt.h>

ISR(TIMER1_COMPA_vect){
    if (alarm_enable == 1) // toggle port F bit 4 if valid button pushed
        PORTF ^= 0x10;
}
```

```c
ISR(TIMER1_COMPA_vect){
    if (alarm_enable == 1) // toggle port F bit 4 if valid button pushed
        PORTF ^= 0x10;
}
```

```c
ISR(TIMER1_COMPA_vect){
    if (alarm_enable == 1) // toggle port F bit 4 if valid button pushed
        PORTF ^= 0x10;
}
```

```c
ISR(TIMER1_COMPA_vect){
    if (alarm_enable == 1) // toggle port F bit 4 if valid button pushed
        PORTF ^= 0x10;
}
```

```c
ISR(TIMER1_COMPA_vect){
    if (alarm_enable == 1) // toggle port F bit 4 if valid button pushed
        PORTF ^= 0x10;
}
```
Using avr-libc

Examples using avr-libc

Generating timing delays:

```c
#define F_CPU 16000000UL //16Mhz
#include <util/delay.h>

for(i=0; i<=3; i++) {_delay_ms(250);} //delay for 1 sec
```

Note: compiler optimizations must be enabled and the delay time must be an expression that is a known constant at compile-time.

#define F_CPU must be defined as a constant. It defines the CPU clock frequency (in Hertz).

The maximal possible delay is 262.14 ms / F_CPU in MHz. If the user requests more delay, resolution is limited to 0.1mS and user will not be notified. See avr-libc Function Documentation for details.

There is also a _delay_us function for smaller delays.
Examples using avr-libc

Accessing EEPROM memory:

```
#include <avr/eeprom.h>

/******************************************************************************
//                          save_settings
//Saves user settings into EEPROM when power is going down.
void save_settings(){
    eeprom_write_byte(&eeprom_alarm_volume, alarm_volume);
    eeprom_write_byte(&eeprom_alarm_level, alarm_level);
} // save_settings
/******************************************************************************/
```