Mattics Phi, Scott Leonard

Section 03/04

Spring 2012

Project 1

LCD Interface Using GPIOs

4/10/12

Instructor: John Oliver

The purpose of this project is to introduce the Arduino Uno and its associated development platform while reviewing the C programming language, timing diagrams, and digital analysis.  
  
The primary requirement for the system is to output the message “Hello World” to a LCD directly from the Arduino’s output pins.

This project required the Arduino Uno to print a simple message to a separate screen.  Also, this printed message needed to change upon the push of a button.  In order to accomplish this, the Arduino Uno needed basic instructions to “talk” with the display screen.  Separate, specific instructions were required to turn the screen on and then display characters with it.   Then, the Uno was given additional instructions to notice when the button was pushed.  This functionality met the major requirements of the projects, but the Uno also managed to print special characters as a bonus.

The selection of components used were very broad - as long as the components were able to accomplish what was intended, then they were put to use.  The LCD had to be 16x2 and must have an 8 bit parallel interface.  The LCD also has to have a resolution of each character to be at least 5 dots wide and 8 dots tall.  In order to complete the project, the use of an Arduino Uno Rev. 2 and a breadboard were required in order to wire up the system and the use of AVRStudio 5.1 was needed to help write the code.

To meet the given requirements for the project, the message “Hello World” needed to be displayed on the screen with the correct formatting (right justified, single line).  Upon the action of a button or switch, an alternate message needed to be displayed for the duration of the action, and then revert back to “Hello World”.  In addition, the LCD needed to print a special character or custom character.

The major components of this system are a personal computer with the AVR Studio 5.1, the Arduino Uno, an LCD, and some basic external components.  These are shown in **Figure 1**.  
  
Software: The Uno was programmed from the computer’s USB port.  The program handled the requirements for initializing the LCD.  Characters were printed to the screen by outputting their correct ASCII code in binary at the correct time.  Port B on the board was used for initialization and control, whereas Port D was used for character data from the software.  
  
Hardware: The Uno was powered throughout the project from its USB connection.  The LCD was powered directly from the Uno board.  Data and control lines were also directly connected to the Uno, with the exception of the contrast adjust pin Vo, which was wired to ground through a 2K ohm resistor.  The external switch was arranged to connect 5V from the board to ground, through a 2K ohm resistor.  The input voltage to the board was connected on the positive side of the switch.  These connections are shown in the wiring diagram in **Figure 2**.

Essentially, the first requirement of the project was to output “Hello World” to the LCD.  The pseudo-code for this requirement went like this:

*int main() {*  
 *Allow PORTD/PORTB to be set as digital output ports*  
 *Write code to start up display*  
 *Write “Hello World” to display*  
 *}*  
Of course, this is just a skeleton of what was actually written.  A more detailed version can be found on the attached sheet. Code examples for setting up the display and writing to the LCD can be seen under **Figure 3, Figure 4,** and **Figure 5** of the attached sheet.  In order to write to the LCD, implementation of the “LCD Write Cycle” had to be invoked.  One thing to keep in mind when sending instructions to the screen was to make sure the bus timings were accurate as to not make the delays too long or too short.  
 Initially, the ports labeled TX and RX (PD0, PD1), had to be hardcoded in order for them to be used as digital output ports.  In **Figure 2**, inspection of the photo shows how the LCD is wired to the Arduino Uno.  The terminals “E” and “RS” on the LCD were tired to PORTB 0:1 while terminal “RS” was tied to ground.  The terminals 1, 2, and 3 were tied to ground, +5V, and the potentiometer (for the backlight), respectively.  In order to get the backlight working, terminal 15 had to have a resistor or diode attached to the +5V line while the last terminal, 16, had to be tied to ground.  The Databus line (terminals 7-17) were to be tied to PORTD 0:7 in order to receive instructions from the microcontroller.

The system components were added to the project and tested one at a time.  First, communication between the Uno and the computer was established to establish programming procedures.  Next, the LCD was connected to the Uno to test its basic functionality in a simple power-on test.  Various functions were then added to the C code to establish full control of the screen.  First, a blinking cursor was displayed to indicate that the initialization procedure was correct.  Then, a single character was printed to test the data lines between the Uno and the LCD.  Once this was accomplished, complete words and special characters were sent to the screen.  Although the LCD handled basic formatting on the screen, like moving the cursor, special code was required to move the cursor between lines or change position.  
  
Once the LCD could print strings of characters on its two screens, a button was added to the system’s external circuitry.  This electrical switch was tested with a multimeter to verify basic functionality.  Connecting the switch to an IO pin on the Uno was trivial, but detecting the presence of a 5v input from the port was difficult inside the C code.  
  
Upon completion of the project, the system’s functionality was verified through a simple qualitative demo.  For the demo, the LCD printed the “Hello World” message, and then changed to a different message upon the push of a button.  After releasing the button, “Hello World” was displayed again.  To test the special characters, an image of a battery was also displayed in a separate test.    No special equipment was required for the final verification, but a multimeter was extremely helpful throughout the troubleshooting process.  
  
The most difficult part of this program was initializing the LCD screen for the first time.  Special attention to the data sheet was required to ensure proper timing during the initialization sequence.  Aside from these difficulties, no major problems were encountered on the project.

**Questions**

1. The microcontroller takes 21.586 ms to write characters to the screen from startup with the LCD still functional.

The total max data transfer rate in KBps is 1.71328 KBps.

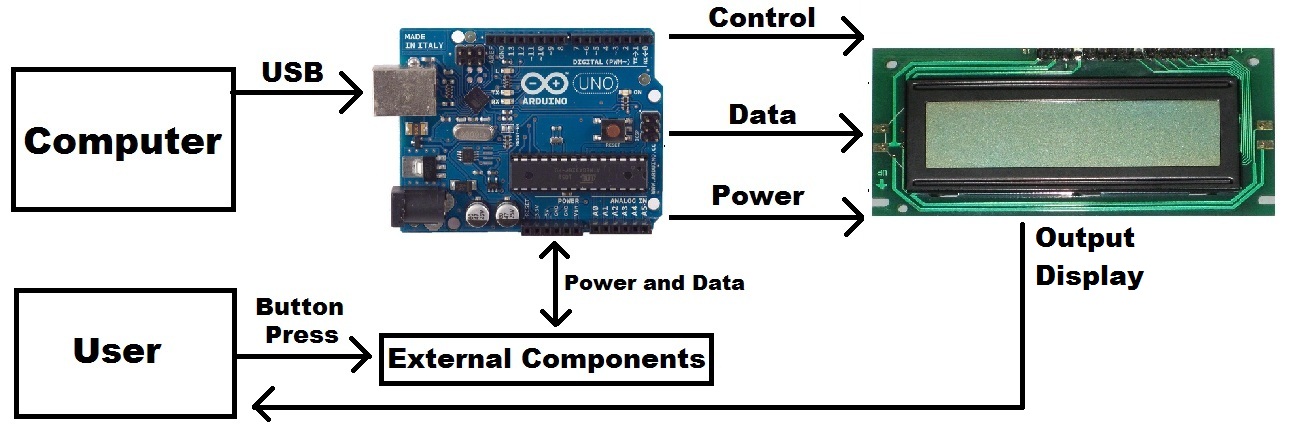
1. The max rate the ATmega328 can write out PortD in KBps is KBps.

**Conclusion**  
The initial step to this project was to understand how the timing diagram worked for each write cycle and to understand how to pass data along the databus into the LCD.  Understanding that was the first task and most likely the hardest task.  The project was accomplished by splitting it into many stages: understanding the datasheet, troubleshooting the ports to get the correct commands, understanding the datasheet again, understanding the code, getting the LCD to put out a cursor, printing the first character, printing out “Hello World”, printing out the special character, and then cleaning and annotating the code.  This way, by partitioning this project into smaller parts, the completion of this project was less hazardous to our health.

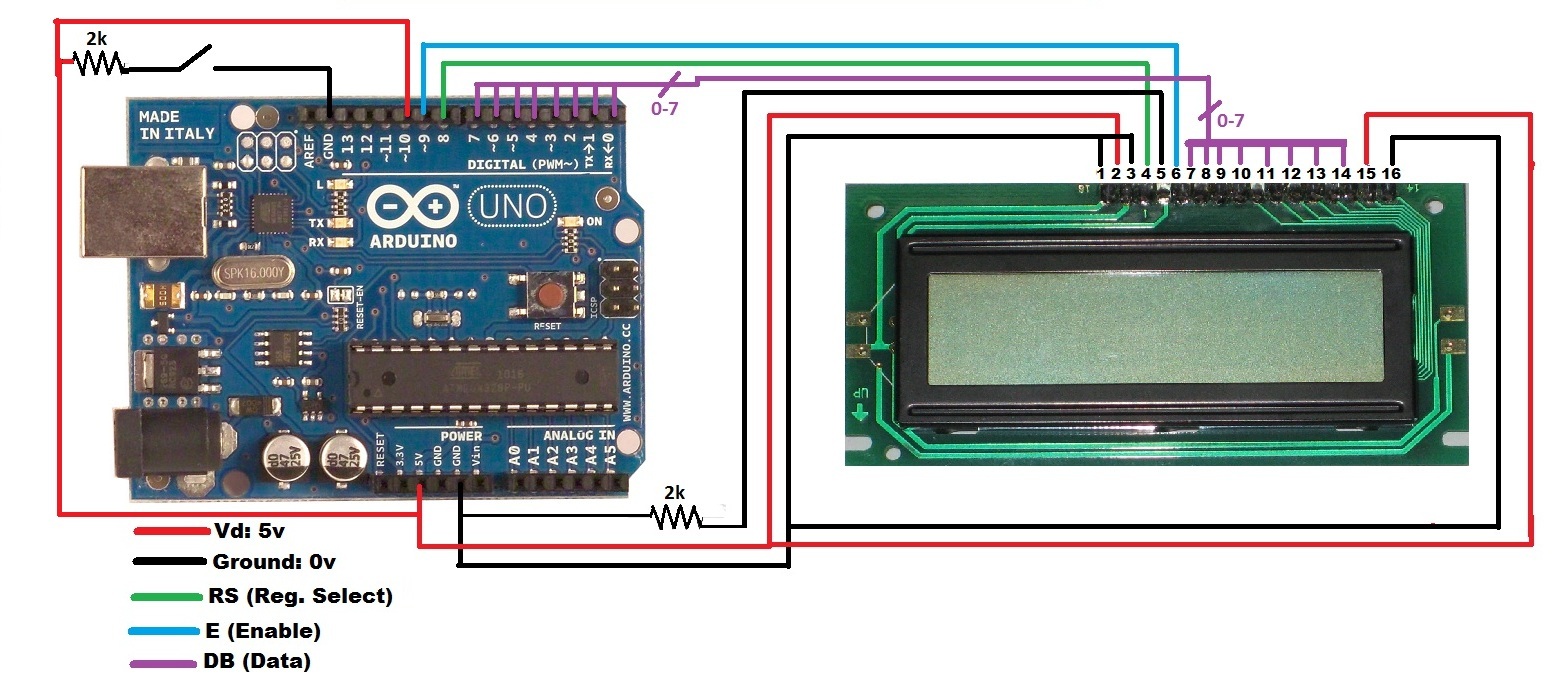
/\* Scott Leonard  
\* Mattics Phi  
\* Project 1 - "Hello World" + Special Character  
\*/  
  
#include <avr/io.h>  
#include <util/delay.h>  
  
#define F\_CPU 16000000ULs  
#define SET\_FUNCTION 0x38  
#define SET\_DISPLAY 0x0F  
#define SET\_CGRAM 0x40  
#define DISPLAY\_CLEAR 0x01  
  
void printFunction(char\*);  
void callInstruction(int);  
void printRandomCharacter();  
void printChar(int);  
void setUpDisplay();  
  
int main(void)   
{   
    int printed = 1;  
   
   UCSR0B = 0x00; // TX/RX to write I/O  
   \_delay\_ms(20);  
   DDRB = 0x03; // enables portB output (PB1 PB0)  
   DDRD = 0xFF; // enables portD output (All of PD 7:0)  
   setUpDisplay(); // function to set up display calls  
   
   if (PINB & 0x04) { // prints out initail sentence when turned on  
       printed = 1;  
   } else {  
       printed = 0;  
   }  
   
   while (1) {  
       if (PINB & 0x04) { // checks to see if switch is in on position  
           if (printed == 1) { // if it hasn't printed hello world yet, print hello world  
               callInstruction(DISPLAY\_CLEAR);  
               printFunction("Hello World!!");  
               printed = 0; // printed hello world, flag goes off  
           }  
       } else {  
           if (printed == 0) {  
               callInstruction(DISPLAY\_CLEAR);  
               printRandomCharacter();  
               printed = 1;  
           }  
       }  
   }  
   return 0;  
}  
  
void setUpDisplay()   
{  
   callInstruction(SET\_FUNCTION); // set function call  
   \_delay\_us(37);  
   callInstruction(SET\_DISPLAY); // set display call  
   \_delay\_us(37);  
   callInstruction(DISPLAY\_CLEAR);  // display clear call  
   \_delay\_ms(1.52);  
}  
  
void callInstruction(int functionCall)  
{  
   PORTB = 0x02; // set E high  
   \_delay\_us(1);  
   PORTD = functionCall;  
   \_delay\_us(1);  
   PORTB = 0x00; // E low  
   \_delay\_us(1);  
}  
  
void printFunction(char\* s)  
{  
   int i;  
   
   for (i = 0; i < strlen(s); i++) {  
       if (i == 16) { // check when it hits the 16th character (16 characters per row)  
           callInstruction(0xC0); // sets DDRAM to 40 which outputs to second row of LCD  
       }  
       PORTB |= 1<<PB0; // Sets RS to high  
       PORTD = s[i];  
       \_delay\_ms(50);  
       PORTB |= 1<<PB1; // set E high  
       \_delay\_ms(50);  
       PORTB &= ~(1<<PB1); // E low  
       \_delay\_ms(50);  
       PORTB &= ~(1<<PB0); // RS low  
       \_delay\_ms(50);  
   }  
}  
  
void printRandomCharacter()  
{  
   callInstruction(0x40); // Sets CGRAM to 0x40  
   printChar(0x0E);  
   printChar(0x1B);  
   printChar(0x11);  
   printChar(0x11);  
   printChar(0x11);  
   printChar(0x11);  
   printChar(0x11);  
   printChar(0x1F);  
   
   printChar(0x00);  
}  
  
void printChar(int db)  
{  
   PORTB = 0x03; // set E high  
   \_delay\_us(1);  
   PORTD = db; // output to DB  
   \_delay\_us(1);  
   PORTD = 0x00; // output nothing to port  
   PORTB = 0x00; // set RS/E low  
   \_delay\_us(1);  
}

References  
ATmega328 Datasheet  
PmodCLP Rev. A Datasheet  
http://www.circuitvalley.com/2012/02/lcd-custom-character-hd44780-16x2.html

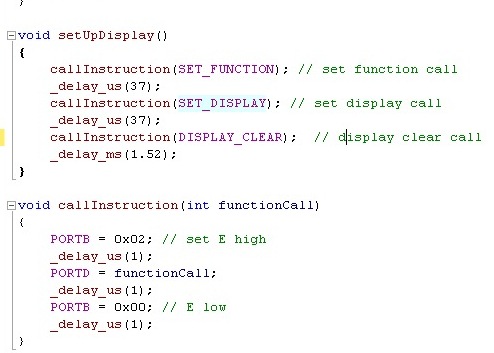
Figures and Tables:

****

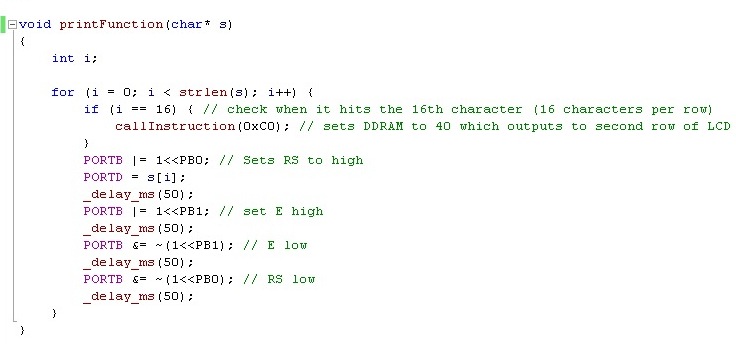
**Figure 1: System Block Diagram**

****

**Figure 2: Complete Wiring Diagram**

****

**Figure 3: Setting Up The Display**

****

**Figure 4: Setting Up To Write to Screen Pt. 1**

****

**Figure 5: Setting Up To Write to Screen Pt. 2**