C Programming through Arduino AVR-GCC

G V V Sharma*

***Abstract*—The objective of this manual is to introduce the student to the art of programming the arduino using pure C instead of the arduino IDE. This will allow the student to become familiar with the avr-gcc cross compiler and avrdude. The instructions in the manual should work for any linux system.**

1 Setup

1.1 Components

Component	Value	Quantity
Breadboard		1
Resistor	$\geq 220\Omega$	1
Arduino	Uno	1
Seven Segment Display	Common Anode	1
Jumper Wires		10

TABLE I

1.2 Hardware

The breadboard can be divided into 5 segments. In each of the green segements, the pins are internally connected so as to have the same voltage. Similarly, in the central segments, the pins in each column are internally connected in the same fashion as the blue columns.

*The author is with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: gadepall@iith.ac.in.

Problem 1. Plug the display to the breadboard in Fig. 1

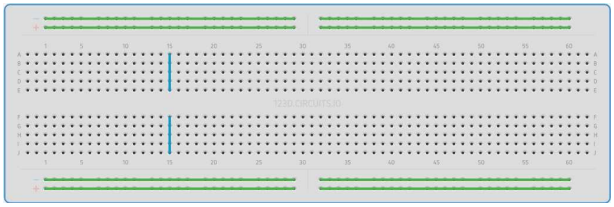

Fig. 1

The seven segment display in Fig. 2 has eight pins, a, b, c, d, e, f, g and *dot* that take an active LOW input, i.e. the LED will glow only if the input is connected to ground. Each of these pins is connected to an LED segment. The *dot* pin is reserved for the $\cdot$ LED.

Problem 2. Connect one end of the 1K resistor to the COM pin of the display and the other end to an extreme pin of the breadboard.

The Arduino Uno has some ground pins, analog input pins A0-A3 and digital pins D1-D13 that can be used for both input as well as output. It also has two power pins that can generate $3.3V$ and $5V$. In the following exercises, only the GND, $5V$ and digital pins will be used.

Problem 3. Connect the 5V pin of the arduino to an extreme pin that is in the same segment as the 1K resistor pin.

Problem 4. Connect the GND pin of the arduino to the opposite extreme pin of the

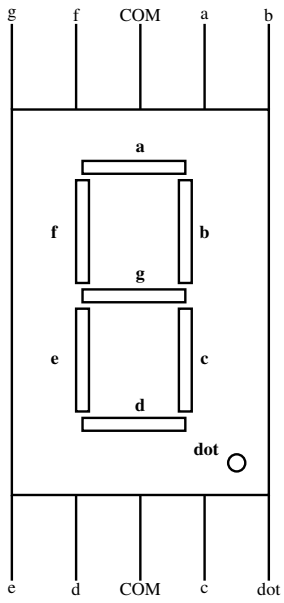

Fig. 2

breadboard

Problem 5. Connect the Arduino to the computer.

1.3 Software

Problem 6. Install AVRA,AVRDUDE and AVR-GCC on your Linux system through the following commands

```
sudo apt-get install avra
    avrdude gcc-avr
%Finding the port

sudo dmesg | grep tty
%The output will be something
    like
[     6.153362] cdc_acm
    1-1.2:1.0: ttyACM0: USB ACM
    device
%and your port number is
    ttyACM0
```

Problem 7. Open a text editor and type the following code. Save it as **Makefile**.

```
MCU=atmega328p

F_CPU=16000000
CC=avr-gcc
CFLAGS=-O2 -Wall -std=gnu11   -
    mmcu=$(MCU)
CPPFLAGS=-DF_CPU=$(F_CPU)L
OBJCOPY=avr-objcopy

CMD_FLASH=avrdude
MCU1=ATMEGA328P
DEVPORT=/dev/ttyACM0
BAUDRATE=115200

.PHONY: default
default: EXE_FILE HEX_FILE
    flash

EXE_FILE: $(FNAME).c
        $(CC) $(CFLAGS) $(
            CPPFLAGS) $< -o $(
            FNAME)
HEX_FILE: $(FNAME)
        $(OBJCOPY) -O ihex $< $
            (FNAME).hex
flash: $(FNAME).hex
        $(CMD_FLASH) -F -V -c
            arduino -p $(MCU1) -P
             $(DEVPORT) -b $(
            BAUDRATE) -U $@:w:$<
```

Problem 8. In the same directory, type the following program in a file. Save the file as **onoff.c**

```
//Turns LED on and off
#include <avr/io.h>
#include <util/delay.h>


int main (void)
{
```

```
  /* Arduino boards have a LED
     at PB5 */
 //set PB5, pin 13 of arduino
    as output
  DDRB     |= ((1 << DDB5));
  while (1) {
//turn led on
    PORTB = ((0 <<  PB5));

//turn led off
//     PORTB = ((1 <<  PB5));
  }

  /* . */
  return 0;

}
//Run the following command in
   the terminal
// to flash the hex file
//make FNAME=onoff
```

1.4 Arduino Pin Control

Problem 9. Now connect the dot pin of the display to pin 13 of the arduino. Open a terminal in the directory where **onoff.c** and **Makefile** are located and type **make FNAME=onoff**. The LED should glow.

Problem 10. Turn the LED off by modifying the above code.

Problem 11. Complete Table II for all the digital pins using Fig. 11

Port Pin	Digital Pin
PD2	2
PB5	13

TABLE II

Pin Number 13 of the arduino corresponds to PB5 of the ATMEGA328P of the microcontroller. While the Arduino IDE recognizes pin 13, the avr-gcc compiler recognizes only PB5. This mapping between the arduino pins and the microcontroller ports is available in Fig. 11 and is crucial for native C programming for the ardunio.

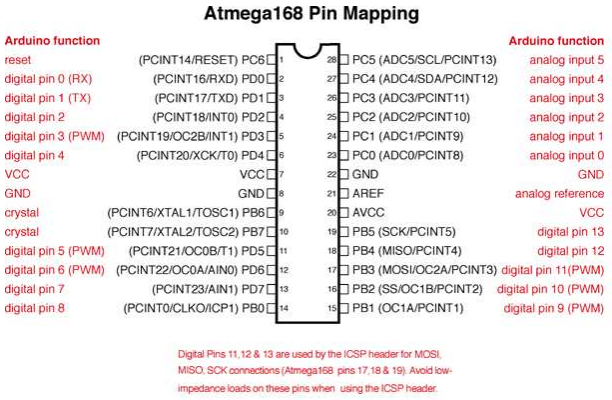

Fig. 11

2 C Programming

2.1 Seven Segment Display

Problem 12. Connect the $a-g$ pins of the display to the 2-8 pins of the arduino respectively. Run the following code.

```
//Prints a decimal number
//on the display
#include <avr/io.h>
#include <util/delay.h>

int main (void)
{


 //set PD2-PD7 as output pins 0
    xFC=0b11111100 (binary)
  DDRD     |= 0xFC;
  //set PB0 as output pin
  DDRB      |= ((1 << DDB0));

  while (1) {
//turn PB0 off
```

```
    PORTB = ((1 <<  PB0));
//turn PD2-PD7 on
    PORTD = 0b00000000;
  }

  /* . */
  return 0;

}
```

Problem 13. The output of Problem 12 can be explained by Table III below.

PD2	PD3	PD4	PD5	PD6	PD7	PB0	decimal
a	b	c	d	e	f	g	
0	0	0	0	0	0	1	0

TABLE III

The command $PORTD = 0b00000000$ puts the binary number $0b00000000$ into PORTD register. Note that PD2-PD7 will have the number 0. PD0-PD1 are unaffected, since $DDRD| = 0xFC(= 0b11111100)$ ensures that only PD2-PD7 are output ports. PD0-PD1 are unaffected and will retain their previous state.

Complete Table III for all numbers between 0-9. Use this information to display the numbers from 0-9 .

2.2 If-Else

Problem 14. Run the following program and test for dec=0 and dec=8.

```
//Prints the number 0 or 8
   using
//the if-else condition
#include <avr/io.h>
#include <util/delay.h>

int main (void)
{
        int dec = 0;

 //set PD2-PD7 as output pins
  DDRD   |= 0xFC;
  //set PB0 as output pin
  DDRB    |= ((1 << DDB0));

  while (1) {

        if(dec == 8)
        //turn PB0 on
                PORTB = ((0 <<
                    0));
        //turn PB0 off
    else if(dec == 0)
                PORTB = ((1 <<
                    0));
//turn PD2-PD7 on
    PORTD = 0b00000000;
  }

  /* . */
  return 0;

}
```

Note that PB0 can be denoted by the number 0 as well for ease in programming

2.3 Switch-case and Function

Problem 15. Write a function for writing a decimal number to the seven segment display.

Solution:

```
//Writing a function
//for display
#include <avr/io.h>
#include <util/delay.h>

void sevenseg(int);
int main (void)
{


 //set PD2-PD7 as output pins
  DDRD   |= 0xFC;
  //set 0 as output pin
```

```
  DDRB    |= ((1 << DDB0));

  while (1) {

        sevenseg(5);
        }
  /* . */
  return 0;

}
void sevenseg(int dec)
{
switch(dec)
{
case 0:
    PORTB = ((1 <<  0));
    PORTD = 0b00000000;
    break;
case 1:
    PORTB = ((1 <<  0));
    PORTD = 0b11100100;
    break;
case 2:
    PORTB = ((0 <<  0));
    PORTD = 0b10010000;
    break;
case 3:
        PORTB = ((0 <<  0));
        PORTD = 0b11000000;
        break;
case 4:
        PORTB = ((0 <<  0));
        PORTD = 0b01100100;
        break;
case 5:
        PORTB = ((0 <<  0));
        PORTD = 0b01001000;
        break;
case 6:
        PORTB = ((0 <<  0));
        PORTD = 0b00001000;
        break;
case 7:
        PORTB = ((1 <<  0));
        PORTD = 0b11100000;
        break;
case 8:
        PORTB = ((0 <<  0));
        PORTD = 0b00000000;
        break;
case 9:
        PORTB = ((0 <<  0));
        PORTD = 0b01000000;
        break;
default:
        PORTB = ((0 <<  0));
        PORTD = 0b00011000;
        break;
}
}
```

2.4 For loop

Problem 16. Use the function in Problem 15 along with a for loop to implement a decade counter.

Solution:

```
//Prints the numbers from 0
//to 9 in a loop
#include <avr/io.h>
#include <util/delay.h>

void sevenseg(int);
int main (void)
{

 int i;

  DDRD    |= 0xFC;
  //set 0 as output pin
  DDRB    |= ((1 << DDB0));

  while (1)
  {
        //loop counting from 0
           to 9
  for (i=0; i < 10; i++)
        {
```

```
                    sevenseg(i);
                    _delay_ms (1000
                       L);
          }
          }
   /* . */
   return 0;

}
void sevenseg(int dec)
{
switch(dec)
{
case 0:
     PORTB = ((1 <<  0));
     PORTD = 0b00000000;
     break;
case 1:
     PORTB = ((1 <<  0));
     PORTD = 0b11100100;
     break;
case 2:
     PORTB = ((0 <<  0));
     PORTD = 0b10010000;
     break;
case 3:
          PORTB = ((0 <<  0));
          PORTD = 0b11000000;
          break;
case 4:
          PORTB = ((0 <<  0));
          PORTD = 0b01100100;
          break;
case 5:
          PORTB = ((0 <<  0));
          PORTD = 0b01001000;
          break;
case 6:
          PORTB = ((0 <<  0));
          PORTD = 0b00001000;
          break;
case 7:
          PORTB = ((1 <<  0));
          PORTD = 0b11100000;
          break;
case 8:
          PORTB = ((0 <<  0));
          PORTD = 0b00000000;
          break;
case 9:
          PORTB = ((0 <<  0));
          PORTD = 0b01000000;
          break;
default:
          PORTB = ((0 <<  0));
          PORTD = 0b00011000;
          break;
}
}
```

2.5 Arrays and Headers

Problem 17. Copy the sevenseg() function into a file called sevenseg.h in the same directory. Now run the following code.

Solution:

```
//using an array
//for decade counter
#include <avr/io.h>
#include <util/delay.h>
#include "sevenseg.h"

void sevenseg(int);
int main (void)
{

//defining integer array p
 const int p[] = {0, 1, 2, 3,
    4, 5, 6, 7, 8, 9};
 int i;

  DDRD   |= 0xFC;
  //set 0 as output pin
  DDRB   |= ((1 << DDB0));

  while (1)
  {
```

```
        //loop counting from 0
           to 9
  for (i=0; i < 10; i++)
        {

                sevenseg(p[i]);
                _delay_ms (1000
                   L);
        }
        }
  /* . */
  return 0;

}
```

This program demonstrates how to use arrays in C. In the above code, p[i] is an array of integers. Also, the sevenseg.h file is known as a header file. Header files are useful for organising large C programs.