Mixing C and Assembly with Arduino

Neelmani Gautam and G V V Sharma*

Abstract—**This manual shows how write a function in assembly and call it in a C program while programming the ATMega328P microcontroller in the Arduino. This is done by controlling an LED.**

1 Components

Component	Value	Quantity
Resistor	$\geq 220\Omega$	1
Breadboard		1
Arduino	Uno	1
LED		1
Jumper Wires		20

TABLE I

2 LED control

Problem 1. Connect **pin 13** of the Arduino to an LED through the resistor.

Problem 2. Download the **Makefile** from http://tlc.iith.ac.in/img/EE2110/Makefile and save it in your working directory.

Problem 3. Write a C program for turning an LED on/off using AVR-GCC.

Solution: Save the following code in a file called **onoffavr.c**.

```
//Turns LED on and off using AVR-
   GCC
#include <avr/io.h>

//Function for enabling pin 13 as
   output
void init(void);

 int main (void)
{
  while (1) {

        init();
//turn led on
        PORTB = ((0 <<  PB5));

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

  return 0;

}
void init(void)
{
           /* Arduino boards have a
              LED at PB5 */
 //set PB5, pin 13 of arduino as
    output
 DDRB    |= ((1 << DDB5));

}
```

Problem 4. Suitably modify the **Makefile** to run the above code.

Solution: In the **Makefile**, make the following changes.

```
TARGET = onoffavr
ASRC =
```

Problem 5. Run **make** in the terminal to turn the LED on.

Problem 6. Modify Problem 3 to turn the LED off.

Neelmani is an undergraduate student at IIT Bhilai. email:neelmanig@iitbhilai.ac.in. *The author is with the Department of Electrical Engineering, Indian Institute of Technology, Hyderabad 502285 India e-mail: gadepall@iith.ac.in. All content in the manual released under GNU GPL. Free to use for anything.

2.1 GCC with Assembly

Problem 7. Write the **init** function as an assembly routine that can be called in the C program.

Solution: Save the following code in a file called **initasm.S**. It is important that I/O (e.g. PORTB) registers be typed in capital letters.

```
;enable pin 13 of Arduino as
   output
#define __SFR_OFFSET 0
#include <avr/io.h>

.global init

.section .text

init:
        LDI R16,0b00100000
        OUT DDRB,R16
```

Problem 8. Modify the C program in Problem. 3 to call the **init()** function from **initasm.S**. Save it as **onoff.c**.

Solution:

```
//Turns LED on and off
//through an assembly routine
#include <avr/io.h>

//Function declared in initasm.S
extern void init(void);

 int main (void)
{
  while (1) {
          init();
//turn led on
PORTB = ((0 << PB5));

//turn led off
//    PORTB = ((1 << PB5));
  }
  return 0;

}
```

Problem 9. Modify the **Makefile** for linking the C and assembly code above and execute the program.

Solution:

```
TARGET = onoff
ASRC =   initasm.S
```

3 Two Assembly Routines

Problem 10. Modify the C program in Problem. 8 to include a function for displaying the output. Name this file as **onoff2.c**

Solution:

```
//Turns LED on and off
//through an assembly routine
#include <avr/io.h>

//Function declared in initasm.S
extern void init(void);
//Function declared in displedasm.
   S
extern void disp_led(uint8_t);
 int main (void)
{
  while (1) {
          init();
        //turn led on/off
          disp_led(0);//0 or 1
             argument
  }
  return 0;

}
```

Problem 11. Write an assembly routine for displaying output through pin 13.

Solution:

```
;turning led on/off
#define __SFR_OFFSET 0
#include <avr/io.h>

.global disp_led

.section .text

disp_led:
        PUSH R24
        ;shifting LSB in r24 to 6
           th position
        LDI R20, 0b00000101      ;
           counter = 5
        RCALL looplsl           ;
            calling the looplsl
            routine
        OUT PORTB,R24           ;
            writing output to pins
            13 (PB5)
        POP R24
        RET

;loop for bit shift to left
looplsl:
        LSL R24                 ;
            left shift
        DEC r20                 ;
            counter --
        BRNE    looplsl ;if
            counter != 0
        RET
```

Problem 12. Modify the **Makefile** for linking **onoff2.c, initasm.S** and **displedasm.S** and execute the program.

Solution:

```
TARGET = onoff
ASRC =   initasm.S displedasm.S
```

Problem 13. Explain how the **disp_led(0)** function in **onoff2.c** is related to **Register R24** in **disp_led** routine in **displedasm.S**.

Solution: The function argument 0 in **disp_led(0)** is passed on to R24 in the assembly routine for further operations. Also, the registers R18-R24 are available for storing more function arguments according to the Table II. More details are avilable in official ATMEL AT1886 reference.

Register	r19	r18	r21	r20	r23	r22	r25	r24
Function Argument	b7	b6	b5	b4	b3	b2	b1	b0

TABLE II: Relationship between Register in assembly and function argument in C

4 Blink

Problem 14. Modify your codes for blinking an LED using the following routine for the delay.

```
;Delay routine for blinking LED
#define __SFR_OFFSET 0
#include <avr/io.h>

.global delay

.section .text

delay:
        DEC R24
        BRNE delay
        DEC R22
        BRNE delay
        DEC R20
        BRNE delay
        RET
```