Introduction to Bare Metal Programming in Arduino Uno

Components and supplies

Introduction

In this tutorial, we are going to see how to program the Arduino Uno without using the Arduino IDE. We will see how the Arduino IDE works under the hood.

How Arduino IDE works

The Arduino IDE uses the avr-gcc compiler and avrdude to upload our program in the microcontroller. So, we are going to compile using avr-gcc the source code (written in C) to obtain the corresponding object file.
Then through avr-gcc, we link the system libraries to the object file to produce the executable or the ELF file.
Using avr-objcopy, we can translate the executable into a binary file that can be uploaded in the Arduino board using avrdude.

Install the tools

The commands to install the tools are for Ubuntu/Debian machine.

```
>$ sudo apt-get update
>$ sudo apt-get upgrade -y
```

And then we install the package required by avr and avrdude.

```
>$ sudo apt-get install gcc-avr binutils-avr avr-libc
>$ sudo apt-get install avrdude
```

Type in the terminal *avr-* and press the tab twice (do not press enter) to see all the tools installed, and type *avrdude -v* to see the version of avrdude installed.

The code

Here's the code to blink the built-in led (**blink_led.c**).

```
#include <avr/io.h>
#include <util/delay.h>

#define MS_DELAY 3000

int main (void) {
    /*Set to one the fifth bit of DDRB to one
    **Set digital pin 13 to output mode */
    DDRB |= _BV(DDB5);

    while(1) {
        /*Set to one the fifth bit of PORTB to one
        **Set to HIGH the pin 13 */
        PORTB |= _BV(PORTB5);

        /*Wait 3000 ms */
        _delay_ms(MS_DELAY);

        /*Set to zero the fifth bit of PORTB
        **Set to LOW the pin 13 */
        PORTB &= ~_BV(PORTB5);

        /*Wait 3000 ms */
        _delay_ms(MS_DELAY);
    }
}
```

We need to include *"avr/io.h"* that contains all the utilities to manage the I/O of the microcontroller. For further information on this header file, I suggest you read the official documentation, which is useful and well-done, and I do not think that I can explain it better than the documentation does.
The other header file we include *"avr/delay.h"* is used only for the wait function.
When we include the I/O header file, we also include the *"avr/sfr_defs.h"* file in which are defined some useful macros we are going to use in our code. In particular, the _BV macro that executes a left bit shift of 1 by the number of positions specified as the argument.
`#define _BV(bit) (1 << (bit))`
Do not forget the official documentation!
We are going to use also three logical operator bit-wise:

- |= the bit-wise logical OR
- &= the logical bit-wise AND
- ~ the bit-wise logical NOT.

So, let's see how we say to the microcontroller to set in output mode the pin 13 of the Arduino Uno board. We need to set the fifth bit of the register DDRB to one:
`DDRB |= _BV(DDB5);`
Then we set to HIGH the same pin setting the fifth bit of PORTB to one:
`PORTB |= _BV(PORTB5);`
And we set to LOW the pin 13 setting the fifth bit of PORTB back to zero:
`PORTB &= ~_BV(PORTB5);`
The *io328p.h* header file defines DDB5 and PORTB5 as 5.

Compile and upload

Now we need to compile, so we first create the object file from the source code specifying the microcontroller in which we will run the program:

```
>$ avr-gcc -Os -DF_CPU=16000000UL -mmcu=atmega328p -c -o blink_led.o
blink_led.c
```

We create the executable:

```
>$ avr-gcc -mmcu=atmega328p blink_led.o -o blink_led
```

And we convert the executable to a binary file:

```
>$ avr-objcopy -O ihex -R .eeprom blink_led blink_led.hex
```

Finally, we can upload the binary file:

```
>$ avrdude -F -V -c arduino -p ATMEGA328P -P /dev/ttyACM0 -b 115200 -U
flash:w:blink_led.hex
```

Source

https://create.arduino.cc/projecthub/milanistef/introduction-to-bare-metal-programming-in-arduino-uno-f3e2b4

https://www.microchip.com/en-us/product/ATmega328P#datasheet-toggle

https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf

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