ATmega8 – Basic Input/Output Interfacing – Tutorial #6

T.K. HAREENDRAN

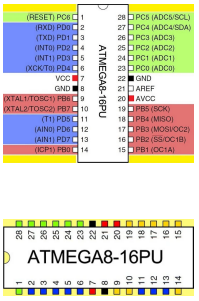
AVR tutorial

Share this:

ATmega8 microcontroller has 23 programmable input/output (I/O) pins which can be used for interfacing with external world. It is possible to configure them as input or output by setting a particular register value through programming. This IC comes in 3 different packages, but we are using the popular 28-Pin PDIP package (Atmega8-16PU). Note that Atmega8 is available in 2 versions; ATmega8 and Atmega8L. Atmega8L is a low frequency version which works up to 8MHz frequency.

The 23 I/O ports of ATmega8 are organised into 3 groups:

- Port B (PB0 to PB7)
- Port C (PC0 to PC6)
- Port D (PD0 to PD7)



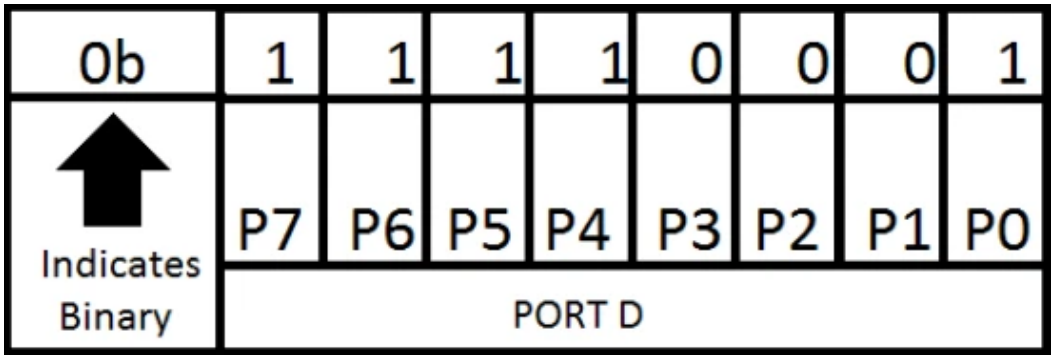
All of these I/O pins have secondary functions, which are shown in parenthesis on the pinout diagram shown here. Each of these registers are 8 bits wide, with each bit corresponding to a single pin (an exception is bit 7 of the Port C register -PC6- most often used as the RESET pin, not an I/O). Registers used for reading and writing to the I/O ports are described below.

Register	Type	Description
DDRB	Read/Write	Port B Data Direction Register
PORTB	Read/Write	Port B Data Register
PINB	Read only	Port B Input Register
DDRC	Read/Write	Port C Data Direction Register
PORTC	Read/Write	Port C Data Register
PINC	Read only	Port C Input Register
DDRD	Read/Write	Port D Data Direction Register
PORTD	Read/Write	Port D Data Register
PIND	Read only	Port D Input Register

Let us consider the following code. What this means?

PORTD = 0b11110001;

The rightmost bit (least significant bit-LSB) represents pin 0 of port D (PD0) whilst the leftmost bit (most significant bit-MSB) represents pin 7. Here, the value is expressed in Binary. The same thing in Hexadecimal (Hex) is 0xF(0bmeans Binary / 0x means Hexadecimal)!



As per this, P1-P2-P3 are Inputs , and others (P0-P4-P5-P6-P7) are Outputs.

Output Port

In our first example (LEDTEST) one LED is connected to Pin 5 of Port C (PC5 → MCU Pin28), configured as an output port. Next, try to understand the code snippet included here.

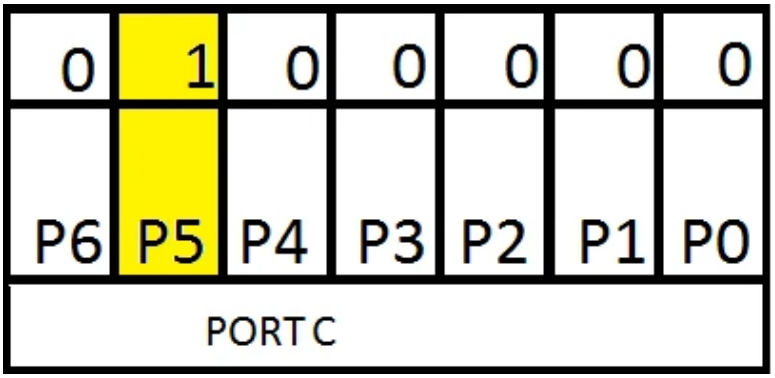
```
#include <inttypes.h>
#include <avr/io.h>
#include <util/delay.h>

int main(){                          // The main function

DDRC = 0b11111111;                     // Set all the pins of PORTC as output
```

In this code, “DDRC = 0b11111111” sets all the pins of PORTC as outputs. But in our first example “DDRC |= (1<<5);” is used to set the Data Direction Register to output. Confused? Don't worry; infact both lines are doing the same thing. In both cases, a literal value is being assigned to PORTC.

- DDRC = 0b11111111 → Sets all the pins of PORT C as outputs
- DDRC |= (1<<5) → Shifts binary representation of 1 to left 5 times (ie 00000001 to 00100000), which means pin 5 of PORTC (PC5) is configured as an output port!

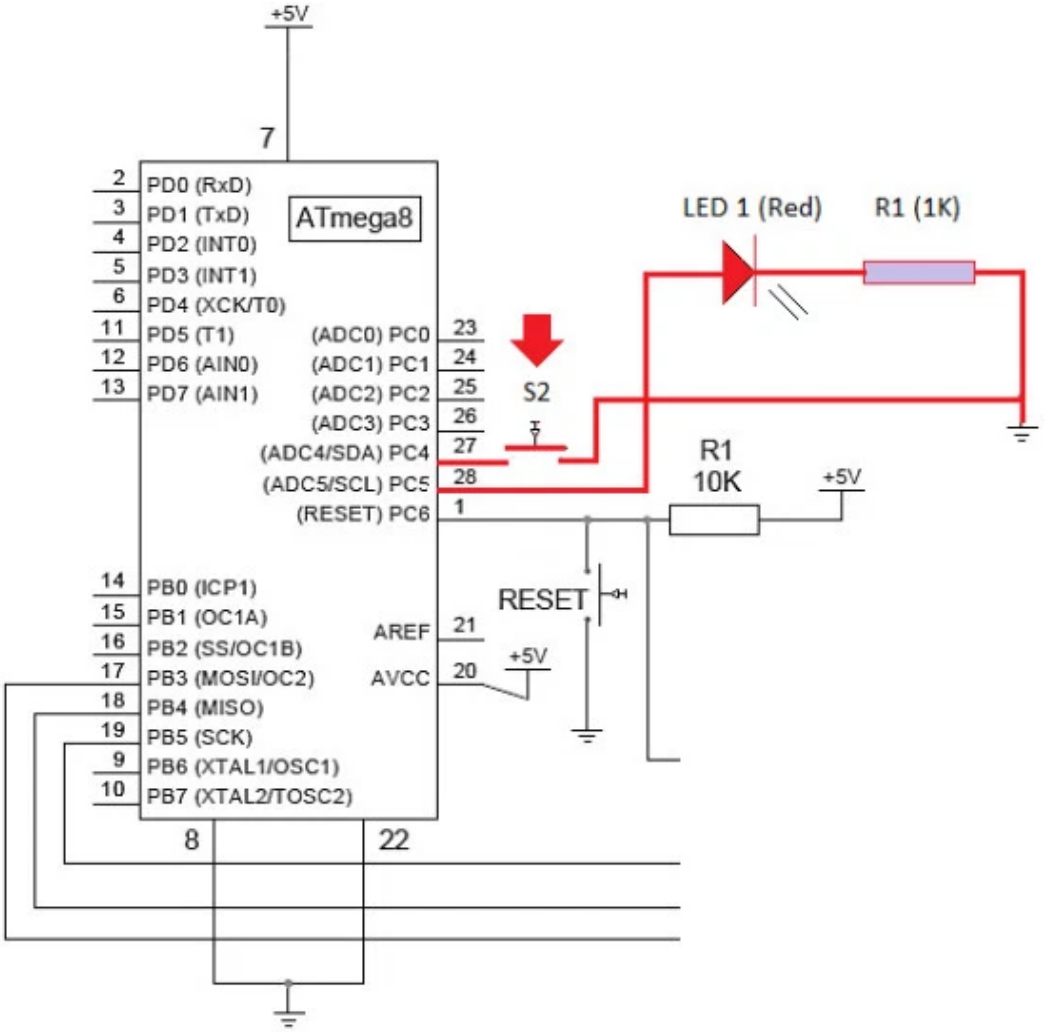


The C programming language includes a set of bitwise operators which apply logic operations to individual bits. These can be combined with _BVmacro to control individual pins. I will explain the Easy Coding Procedures immediately after the completion of the next part.

Input Port

Get ready to make an input connection with ATmega8. For this, take the breadboard and ensure that your first program (LEDTEST) is running well. Now remove the power supply and re-wire the breadboard as shown in the next circuit diagram. The work is very simple, because all you need is to add a pushbutton switch (at Pin

27) with the existing circuit. Next, write-compile-build and burn the associated code to ATmega 8 as before. If everything is OK, the LED at pin 28 starts blinking for 6 times when you push the pushbutton switch (S2) for a while. In this setup Pin 28 is output port, and Pin 27 is an input port. Code will be explicated in the forthcoming part, so stay tuned!



AVR BUTTON & LED PROJECT CODE

```
#define F_CPU 1000000UL
#include
#include
//Define functions
//======================
void ioinit(void);
void led_on(void);
void led_off(void);
//======================
int main (void)
{
   ioinit(); //Setup IO pins and defaults
   while (1)
   {
      if (bit_is_clear(PINC, 4))
      {
         for (int i=0;i<6;i++)
         {
            if (i>0)
               {
               _delay_ms(500);
               led_on();
               _delay_ms(500);
               led_off();
              }
         }
      }
   }
}

void ioinit (void)
{
   DDRC = 0b11101111; //Pin 27 of MCU as input
   PORTC = 0b00010000; //Enable internal pullup of pin 27
}

void led_on(void)
{
   PORTC |= _BV(PC5); //Pin 28 of MCU as output
}

void led_off(void)
{
   PORTC &= ~_BV(PC5);
}
```

→ Part 7: **ATmega8 – Basic Input/Output Interfacing 2**
← Part 5: **AVR Clock Source & Fuse Bits**