ATmega8 – Basic Input/Output Interfacing – Tutorial #7

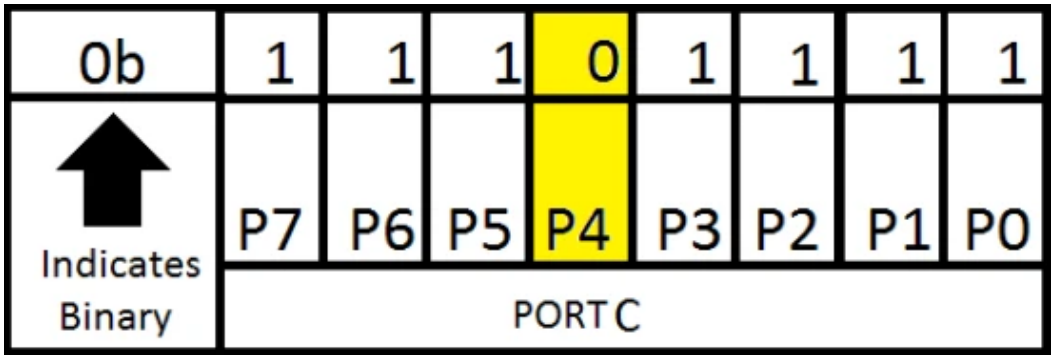
T.K. HAREENDRAN

AVR tutorial

Share this:

When it comes to ATmega8 basic I/O interfacing, we just learned how to connect a switch as an input device to control one LED connected as an output device.
Now a simple tour through selected areas of the code is necessary to figure out what's behind the process!

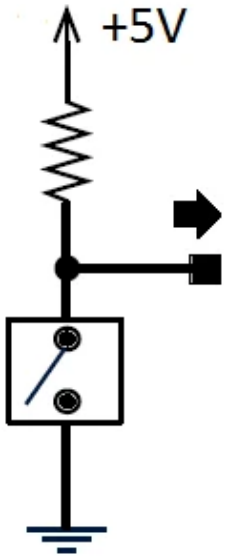
```
void ioinit (void)
{
    DDRC = 0b11101111;  //Pin 27 of MCU as input
    PORTC = 0b00010000; //Enable internal pullup of pin 27
}
```



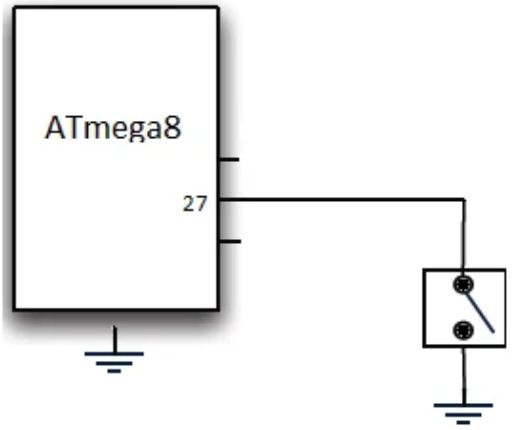
DDRC = 0b11101111; – sets Pin 27 of ATmega8 as input
This means PC4 of ATMega8 is configured as an input point.
Actually PC4 is Pin 27 of the MCU!

PORTC = 0b00010000; – this Enables internal pullup of pin 27

In the example one push button switch is used at pin 27 of the MCU. Usually, one resistor tied to Vcc is required for proper function of this switch as shown here.



Fortunately ATMega8 has internal “pull-up” resistors, with the help of which we can wire switches without using any external pull-up resistors. But how to turn on these internal pull-ups? It is easy; just write “HIGH”(1) on the required input. This sets the default value of the input. However, note that thisdoesn’t work the other way; you can’t set it to “LOW” (0) then wire the switch to +5V.



When moving to next portion of the code, it seems

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

PORTC |= _BV(PC5); – this sets Pin 28 of MCU as output

Here Bitwise (_BV) Operators are used. When programming ATmega8, you often need to address single pins. The _BV macro and bitwise operators come in handy here. _BV is a macro that takes a value between 0 and 7 and returns an 8 bit value with a “1” in the position nominate by the input parameter.

- _BV(0) gives you 0b00000001
- _BV(3) gives you 0b00001000
- _BV(i) is equivalent to 1<

Following this in our code **PORTC|=_BV(P-C5);** turns PC5 (pin 28 of MCU) on and **PORTC&=~-_BV(PC5);** turns PC5 off. Infact, PORTC=PORTC|_BV(PC5); turns PC5 on, but here compound assignment operator is used instead. In the next part of this tutorial, we first explore the basic bitwise operators available and then we learn how to combine them to perform certain common useful operations.

Moving back to our code

```
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();
            }
        }
    }
}
```

This means “if Button on PC4 (MCU pin 27) is pressed, blink LED on PC5 (MCU pin 28) 6 times”. Here, a lucid way of employing the “**bit_is_set or bit_is_clear**” macros is used for convenience.

Port Registers

Port registers are for lower-level and faster manipulation of the I/O pins of the microcontroller. Each port is controlled by three registers; The DDR register, determines whether the pin is an INPUT or OUTPUT. The PORT register controls whether the pin is HIGH or LOW, and the PIN register reads the state of INPUT pins set to input. DDR and PORT registers may be both written to, and read. PIN registers correspond to the state of inputs and may only be read. Each bit of these registers corresponds to a single pin; e.g. the low bit of DDRD, PORTD, and PIND refers to pin PD0.

Example:

- DDRD – The Port D Data Direction Register: read/write
- PORTD – The Port D Data Register: read/write
- PIND – The Port D Input Pins Register: read only

The DDRx register configures the I/O pins either Input or Output The PORTx register determines whether the output should be HIGH or LOW of the output pins.
This means, once the DDRx register has defined the output pins, we need to set them to give an output of HIGH or LOW. The PINx register gets the reading from the input pins of the MCU.

As promised, in the upcoming part, I will try to explain all about the basic bitwise operators and thereafter disclose how to combine them to perform certain common useful operations.

→ Part 8: **Basic Bitwise Operators + Part 1**
← Part 6: **ATmega8 – Basic Input/Output Interfacing 1**

Share this: