AVR Programming in C

CSE-425

Lecture-5

Reference: Eighth+Ninth chapter of textbook

Acknowledgment: Some websites

Sections

- AVR in C
- AVR I/O programming

Why program the AVR in C?

Assembly language produces a hex file that is much smaller than C, but programming in Assembly is tedious and time consuming while:

- It is easier and less time consuming to write in C
- C is easier to modify and update
- One can use code available in function libraries
- C code is portable to other microcontrollers with little or no modification

C data types for AVR

Some data types widely used by C compilers:

Data type	Size in Bits	Data range/usage
Unsigned char	8	0 to 255
char	8	-128 to +128
Unsigned int	16	0 to 65,535
int	16	-32,768 to +32,767
Unsigned long	32	0 to 4,294,967,295
long	32	-2,147,483,648 to +2,147,483,648
float	32	±1.175e-38 to ±3.402e38
double	32	±1.175e-38 to ±3.402e38

I/O Port pins and their functions

- In AVR microcontroller family, there are many ports available for I/O operations, depending on which family microcontroller we choose. For the ATmega32 40-pin chip, 32 Pins are available for I/O operation. The four ports PORTA, PORTB, PORTC, and PORTD are programmed for performing desired operation.
- The number of ports in AVR family varies depending on number of pins available on chip. The 8-pin AVR has port B only, while the 64-pin version has ports A to ports F, and the 100-pin AVR has ports A to ports L.

The table showing Numbers of ports in some AVR family members is shown below:

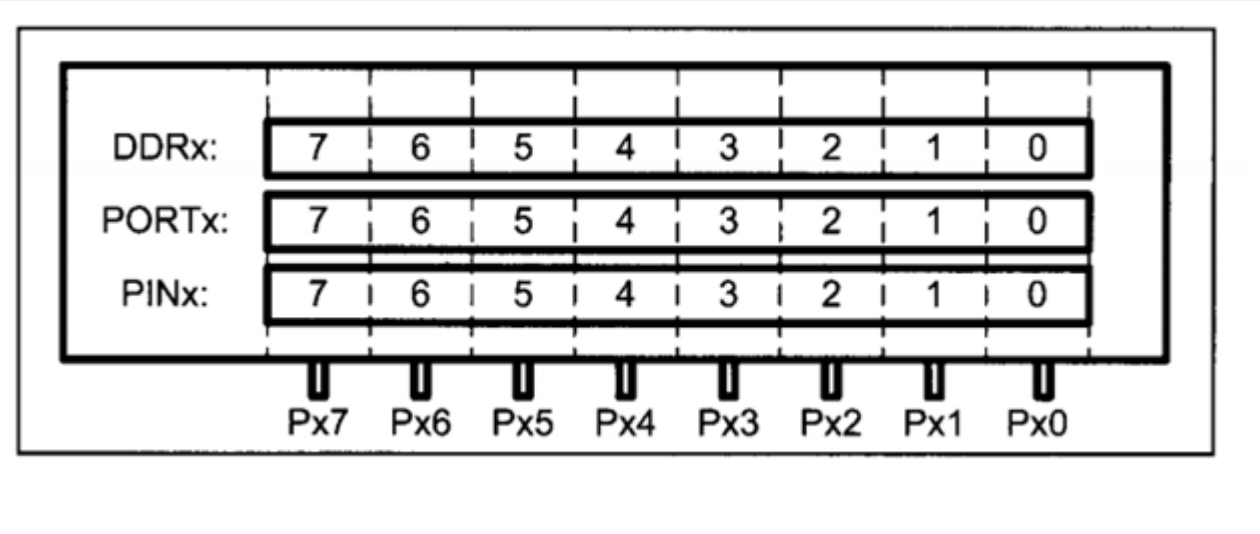
Pins	8-pin	28-pin	40-pin	64-pin	100-pin
Chip	ATtiny25/45/85	ATmega8/48/88	ATmega32/16	ATmega64/128	ATmega1280
Port A			X	X	X
Port B	6 bits	X	X	X	X
Port C		7 bits	X	X	X
Port D		X	X	X	X
Port E				X	X
Port F				X	X
Port G				5 bits	6 bits
Port H					X
Port J					X
Port K					X
Port L					X

Here, X indicates that the port is available. The 40-pin AVR has four ports for using any of the ports as an input or output port, it must be accordingly programmed. In AVR microcontroller not all ports have 8 pins. For example:-in the ATmega8, Port C has 7 pins.

AVR Registers

- Each port in AVR microcontroller has three I/O registers associated with it. They are designated as PORTx, DDRx and PINx. For example: - in case of Port B we have PORTB, DDRB, and PINB.
- Each of I/O registers is 8 bits wide, and each port has a maximum of 8 pins, therefore each bit of I/O registers affects one of the pins.
- For accessing I/O registers associated with the ports the common relationship between the registers and the pins of AVR microcontroller is used.

The relation between the Registers and the Pins of AVR is shown below:



Each port has 3 control I/O registers associated with it:

- **DDRx register**

- Data Direction Register configures the data direction of port pins. These registers are used for determining whether port pins will be used for input or output. On writing 1 to a bit in DDRx makes corresponding port pin as output, while writing 0 to a bit in DDRx makes corresponding port pin as input.

  For example:

  For making all pins of port A as output pins: DDRA = 0b11111111 (in binary) or DDRA = 0xFF (in hexa)

  For making all pins of port A as input pins: DDRA = 0b00000000 (in binary) or DDRA = 0x00 (in hexa)

  For making lower nibble of port B as output and higher nibble as input: DDRA = 0b00001111

- **PORTx register**

- The PORTx bits in the PORTx register have two functions. They can control the output state of a pin and the setup of an input pin.

As an Output:

When port is configured as output then PORTx register is used. When we set bits in DDRx to 1, corresponding pins becomes output pins. Now we can write the data into respective bits in PORTx register. This will immediately change the output state of pins according to data we have written on the ports.

If a '1' is written to the bit when the pin is configured as an output pin, the port pin is driven high. If a '0' is written to the bit when the pin is configured as an output pin, the port pin is driven low. For example:

To output 0xFF data on port B:

```
DDRB = 0b11111111;      //set all the pins of port B as outputs
PORTB = 0x11111111;         //all the pins are made high
```

To output 0xFF data on port B:

```
DDRB = 0b11111111;      //set all the pins of port B as outputs
PORTB = 0x00000000;         //all the pins are made low
```

As an Input:

If a '1' is written to the bit when the pin is configured as an input pin, the pull-up resistor is activated. If a ‘0’ is written to the bit when the pin is configured as an input pin, the port pin is tri-stated.

Note: While using on chip Analog to Digital Converter (ADC), ADC port pins must be used as tri stated input.

To make port B as tri stated input:

```
DDRB = 0x00;        //use port B as input
PORTB = 0x00;        //Disable pull-ups register and make it tri state
```

To make port B as input with pull-ups enabled:

```
DDRB = 0x00;        //use port B as input
PORTB = 0xFF;        //enable all pull ups
```

- **PINx register**

- PINx register used to read the data from port pins. In order to read the data from port pin, first we have to change the port's data direction to input. This is done by setting bits in DDRx to zero. If port is made output, then reading PINx register will give a data that has been output on port pins.
- There are two input modes. Either we can use port pins as internal pull up or as tri stated inputs as it has been explained before. For example:

For reading the data from port A**:**

```
DDRA = 0x00;    //Set port A as input
PORTA= 0xFF;   //enable all pull-ups
x = PINA;       //Read contents/data from the pins of port A
```

Practice Problems:

1. Write an AVR C program to toggle all the bits of PORT B 200 times.
2. Write an AVR C program to toggle the third bit of port B continuously with a 100 mili-second delay.
3. Write an AVR C program to toggle 4 bits of port C continuously with a 70% duty cycle.
4. Leds are connected to the pins of port B. Write an AVR code to show from 0-255 on the leds.
5. Leds are connected to port B. Write an AVR program that shows the leds term on sequentially a 100 millisecond delay.

Have a Wonderful Journey with Microcontroller !