EMBEDDED SYSTEMS CMPE-453

Department of Computer Engineering

I/O port Programming

GENERAL STRUCTURE OF A PROGRAM

```
Preamble
Include files
Macros
Global variables
Function definitions
Main method
 {
   //Chip configurations

   //Infinite event loop
   while(1)
   {

   }
 return (0);
 }
```

REGISTER CONFIGURATION FOR DIGITAL OUTPUT

- DDRx (Data Direction Register for Port x={B, C, D})
  - Used to define data direction of port(s).
  - Readable/writable

  - Setting a bit to 1 → corresponding pin of the given port as is set as OUTPUT pin
  - Setting a bit to 0 → corresponding pin of the given port as is set as INPUT pin
  - Default configuration: Input

  - Header file io.h need to be included to access the registers by their name

  - Individual bits are given names of DDxn

REGISTER CONFIGURATION FOR DIGITAL OUTPUT

- DDRx (Data Direction Register for Port x={B, C, D})
- DDRB:

DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0

- Example: Write a C-statement to configure PortD as Output Port.
  - DDRD = 0xFF; or
  - DDRD = 0b11111111; or
  - DDRD = 255;

REGISTER CONFIGURATION FOR DIGITAL OUTPUT

- PORTx (Data Register for Port x={B, C, D})
  - Used to set logic values (i.e. 0 or 1) on a port that has already been configured as output port.
  - Read/writable
  - Setting a bit to 1 → Logic 1 (i.e. Vcc) on corresponding output pin of the given port.
  - Setting a bit to 0 → Logic 0 (i.e. ground) on corresponding output pin of the given port.
  - Default configuration: Logic 0
  - Header file io.h need to be included to access the registers by their name
  - Individual bits are given names of Pxn

REGISTER CONFIGURATION FOR DIGITAL OUTPUT

- PORTx (Data Register for Port x={B, C, D})

- Example: Write C-statements to configure PORTC as ouput port and drive logic 1 on PC3 and PC6.

```
DDRC = 0xFF;
PORTC=0b01001000;
```

DIGITAL INPUT

- PINx (Port x ={B, C, D} input pins)

- If a port is configured as input port, the input data is made available in PINx register.

- Only readable.

- More on this later lectures

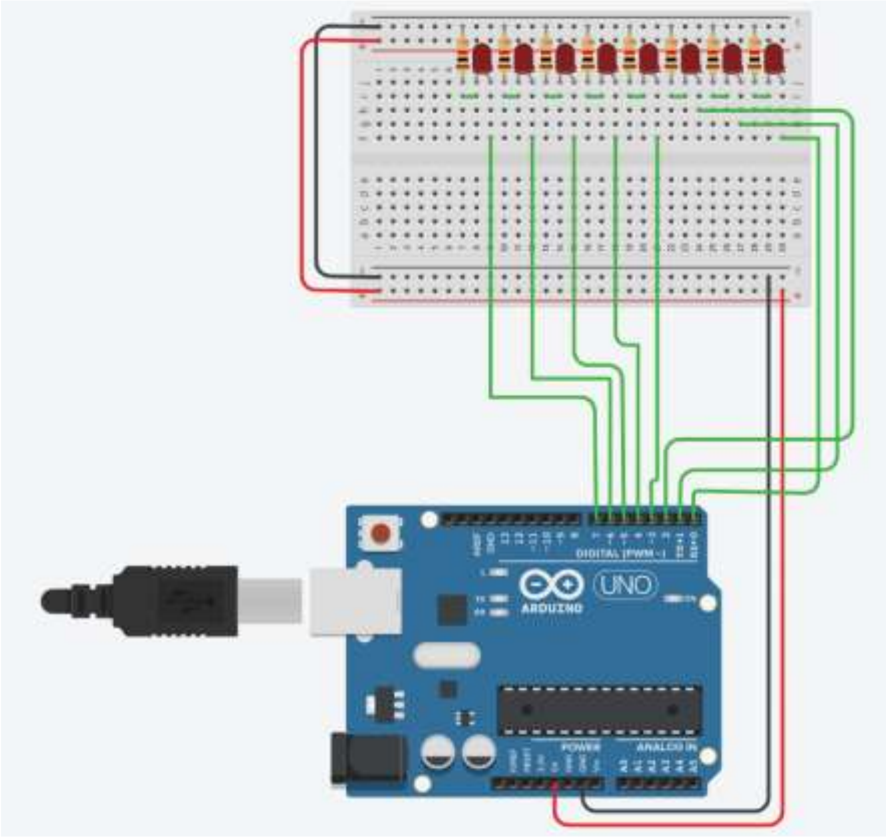
STEPS FOR USING A PORT AS OUTPUT

1. Configure the direction of the port by writing its DDR register.

2. Write the value in the PORT register to make it available on pins of the port.

EXAMPLE: TWO STEP FORWARD, ONE STEP BACK

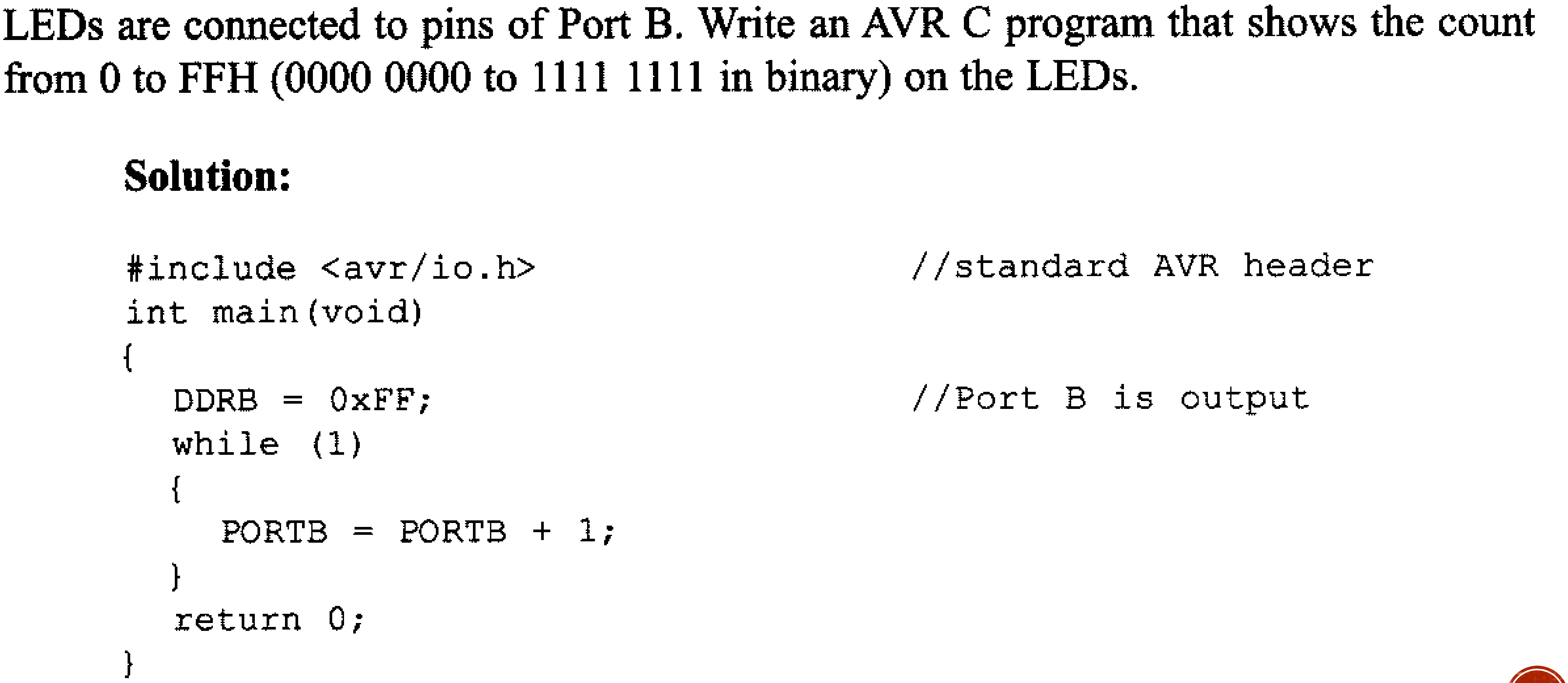
- Build an AVR-microcntroller based circuit that blinks 8 LEDs from right to left direction by taking «two steps forward, one step backward».
- Connect 8 LEDS with PORTD.
- NOTE: Each LED need to be grounded using a current limiting resistor (1K).



C-PROGRAM

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

//macros

#define DELAY 400

//method definations

void blinkLEDs(uint8_t byte)

          {

          PORTD=byte;

          _delay_ms(DELAY);

          }

//main method

int main (void)

          {

          //Initialization

          DDRD = 0xFF;

          //Event loop

while(1) {


blinkLEDs(0b00000001);
blinkLEDs(0b00000010);
blinkLEDs(0b00000001);
blinkLEDs(0b00000010);
blinkLEDs(0b00000100);
blinkLEDs(0b00000010);
blinkLEDs(0b00000100);
blinkLEDs(0b00001000);
blinkLEDs(0b00000100);
blinkLEDs(0b00001000);
blinkLEDs(0b00010000);
blinkLEDs(0b00001000);
blinkLEDs(0b00010000);
blinkLEDs(0b00100000);
blinkLEDs(0b00010000);
blinkLEDs(0b00100000);
blinkLEDs(0b01000000);
blinkLEDs(0b00100000);
blinkLEDs(0b01000000);
blinkLEDs(0b10000000);
blinkLEDs(0b01000000);
blinkLEDs(0b10000000);

          }

return 0;

}
```

EXERCISE-1

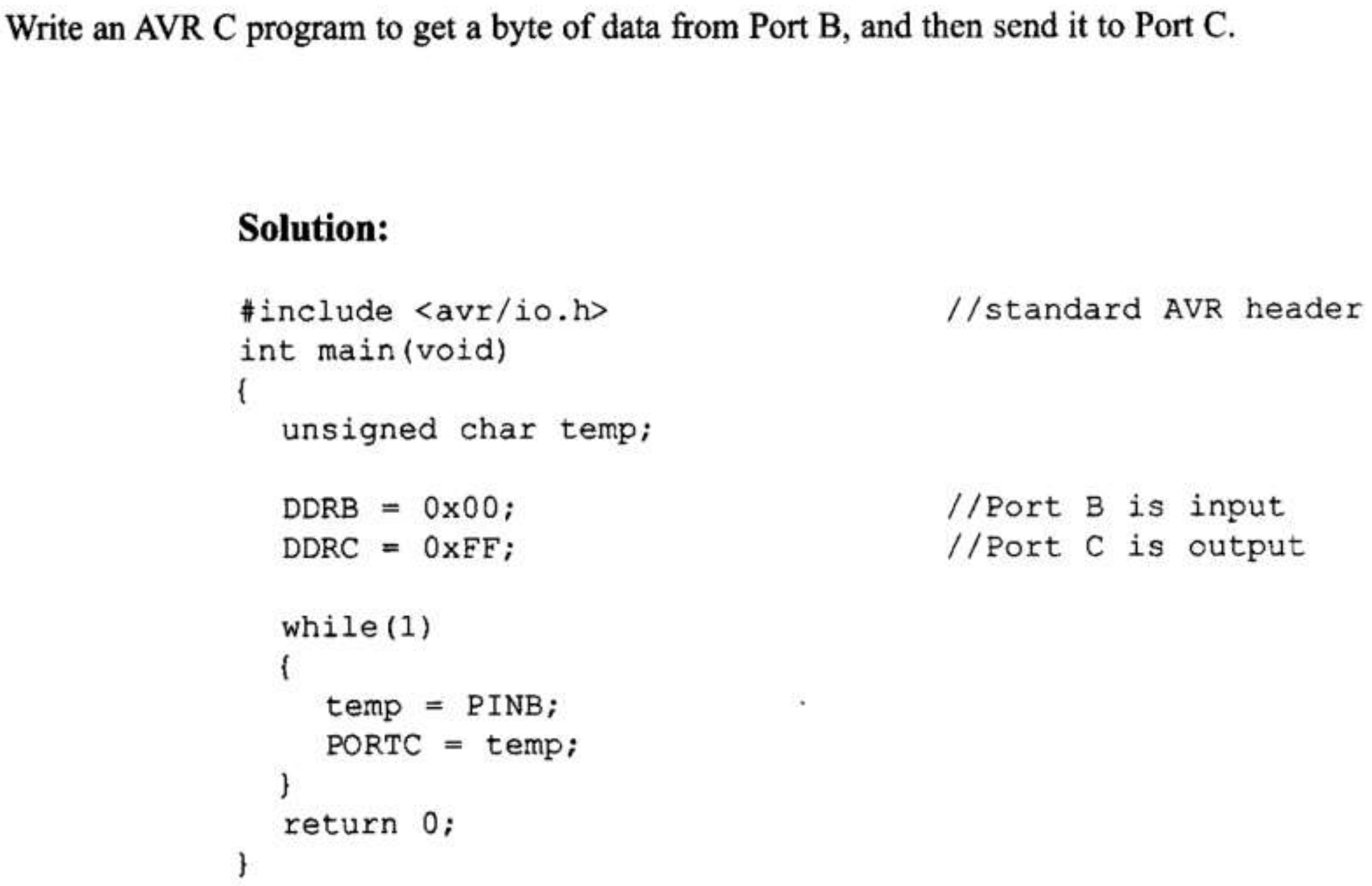


LEDs are connected to pins of Port B. Write an AVR C program that shows the count from 0 to FFH (0000 0000 to 1111 1111 in binary) on the LEDs.

Solution:

```
#include <avr/io.h>                    //standard AVR header
int main(void)
{
  DDRB = 0xFF;                         //Port B is output
  while (1)
  {
    PORTB = PORTB + 1;
  }
  return 0;
}
```

EXERCISE-2

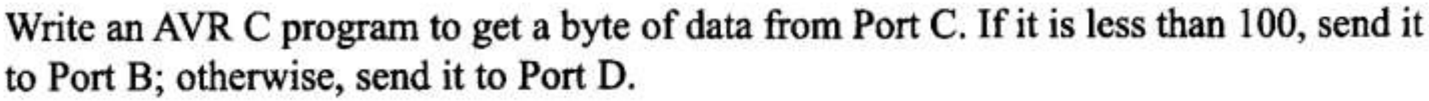


Write an AVR C program to get a byte of data from Port B, and then send it to Port C.

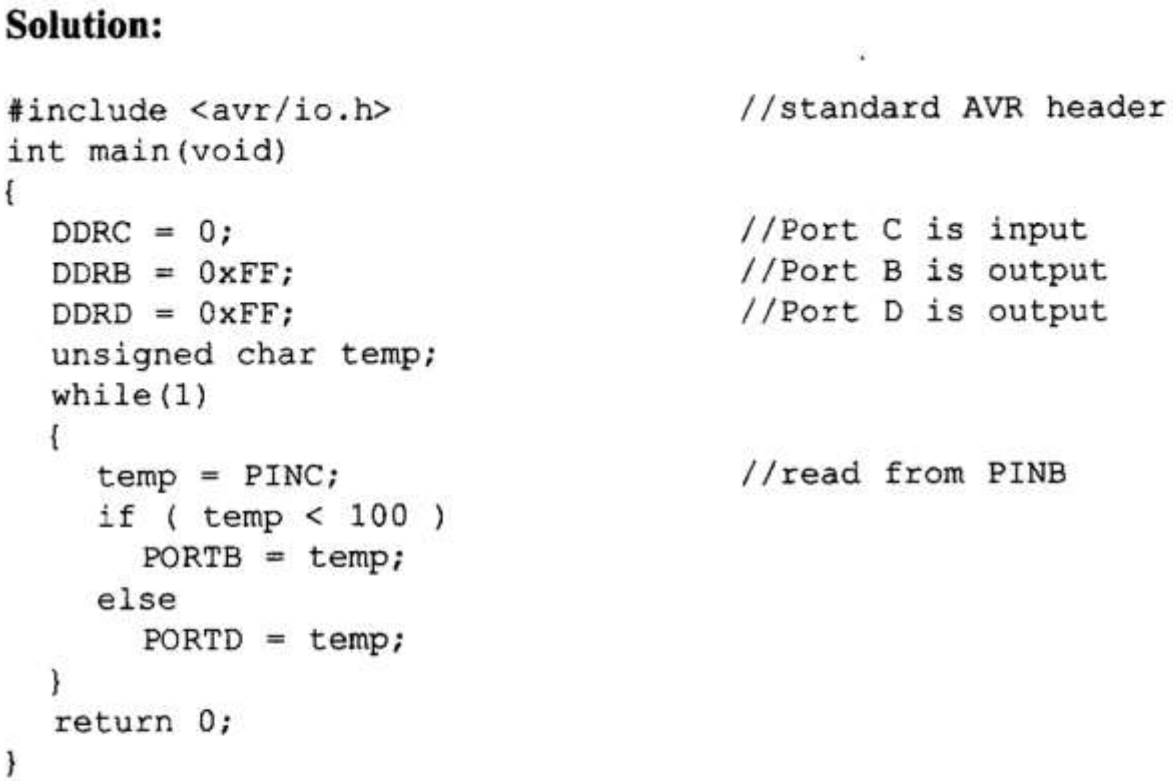
Solution:

```
#include <avr/io.h>                 //standard AVR header
int main(void)
{
  unsigned char temp;

  DDRB = 0x00;                      //Port B is input
  DDRC = 0xFF;                      //Port C is output

  while(1)
  {
    temp = PINB;
    PORTC = temp;
  }
  return 0;
}
```

EXERCISE-3



Write an AVR C program to get a byte of data from Port C. If it is less than 100, send it to Port B; otherwise, send it to Port D.



Solution:

```
#include <avr/io.h>                 //standard AVR header
int main(void)
{
  DDRC = 0;                         //Port C is input
  DDRB = 0xFF;                      //Port B is output
  DDRD = 0xFF;                      //Port D is output
  unsigned char temp;
  while(1)
  {
    temp = PINC;                    //read from PINB
    if ( temp < 100 )
      PORTB = temp;
    else
      PORTD = temp;
  }
  return 0;
}
```