```c
/*
 * Blink.c
 *
 * Created: 9/28/2020 18:23:26
 * Author: masau
 */
/*****************************************************************
 *
 * Blink a LED and use the function from the delay library.
 * ATmega328P (Arduino Uno), 16 MHz, AVR 8-bit Toolchain 3.6.2
 *
 * Copyright (c) 2018-2020 Tomas Fryza
 * Dept. of Radio Electronics, Brno University of Technology, Czech
 * This work is licensed under the terms of the MIT license.
 *
 *****************************************************************/

/* Defines -----------------------------------------------------------*/
#define LED_GREEN   PB5      // AVR pin where green LED is connected
#define SHORT_DELAY 1000     // Delay in milliseconds between individual bits
#define LONG_DELAY  2000     // Delay between individual characters "DE2"
#ifndef F_CPU
#define F_CPU 16000000       // CPU frequency in Hz required for delay function
#endif

/* Includes ----------------------------------------------------------*/
#include <util/delay.h>     // Functions for busy-wait delay loops
#include <avr/io.h>         // AVR device-specific IO definitions

/* Variables ---------------------------------------------------------*/

/* Function prototypes -----------------------------------------------*/

/* Functions ---------------------------------------------------------*/
/**
 * Toggle one LED and use the function from the delay library.
 */
int main(void)
{
    // Set pin as output in Data Direction Register
    // DDRB = DDRB or 0010 0000
    DDRB = DDRB | (1<<LED_GREEN);

    // Set pin LOW in Data Register (LED off)
    // PORTB = PORTB and 1101 1111

    // DE2 in Morse Code: 1001(D),10(E),111100(2)
    // Infinite loop
    while (1)
    {
        //Sending "D" : 1001
        PORTB |=(1<<LED_GREEN);           //The LED is set on(High) to send 1
        _delay_ms(SHORT_DELAY);           //1000ms pause before the next bit
        PORTB = PORTB & ~(1<<LED_GREEN);  //The LED is reset to Low to send 0
        _delay_ms(SHORT_DELAY);
        PORTB = PORTB & ~(1<<LED_GREEN);  // The LED is kept Low to send the Second 0
        _delay_ms(SHORT_DELAY);
        PORTB |=(1<<LED_GREEN);          // The LED is set on(High) to send 1
```

```
        _delay_ms(SHORT_DELAY);

        PORTB = PORTB & ~(1<<LED_GREEN);   //Reset LED to low
        _delay_ms(LONG_DELAY);          //2000ms pause before next character i.e. "E"


        // Sending "E" : 10
        PORTB |=(1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);
        PORTB = PORTB & ~(1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);

        _delay_ms(LONG_DELAY);     //2000ms pause before next character i.e "2"

        //Sending "2" : 111100
        PORTB |=(1<<LED_GREEN);    //The LED is set on to send 1 4x
        _delay_ms(SHORT_DELAY);    //1000ms pause between the bits
        PORTB |=(1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);
        PORTB |=(1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);
        PORTB |=(1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);
        PORTB = PORTB & ~(1<<LED_GREEN);   //The LED is reset to Low to send 0
        _delay_ms(SHORT_DELAY);            //1000ms pause before the next bit
        PORTB = PORTB & ~(1<<LED_GREEN);
        _delay_ms(SHORT_DELAY);

        // Invert LED in Data Register
        // PORTB = PORTB xor 0010 0000
        PORTB = PORTB ^ (1<<LED_GREEN);
    }

    // Will never reach this
    return 0;
}

/* Interrupt routines ----------------------------------------------*/
```

Question2

Symbol (Binary operators)	Meaning
|	Bitwise operator OR
&	Bitwise operator AND
^	Bitwise operator XOR
~	Binary one's complement is a unary operator (It inverts all bits)
<<	Left shift operator

Truth table

Operands		Operations		
A	B	A|B	A&B	A^B
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	0	0

Examples

A = 12 = 00001100

B = 25 = 00011001

1. **&** : Bitwise AND returns the output 1 only if the corresponding bits of the two operands is 1.

   The AND operation of A and B is:
   00001100
   & 00011001

   _____
   00001000 = 8 (In decimal)

2. **|** : The OR operation returns the output 1 if either or both of the corresponding bits of the two operands is 1.

   Bitwise OR Operation of A and B
   00001100
   | 00011001

   _____
   00011101 = 29 (In decimal)

3. **^** : The result of the operator XOR is 1 if corresponding bits of the two operands are opposite.

   Bitwise XOR Operation of A and B
   00001100
   ^ 00011001

   _____
   00010101 = 21 (In decimal)

4. **~ :** The bitwise complement operator has only one operand. It invert the values of the input to give the out put.

Bitwise complement Operation of A
~ 00001100

_____
11110011 = 243 (In decimal)

5. **<<** :Left shift operator shifts all bits towards left by a certain number of specified bits. The bit positions that have been vacated by the left shift operator are filled with 0.

   The shift operation of A is:
   A = 12 = 00001100
   A<< 1 = 00011000

Link 🔗 to GitHub repository
https://github.com/Masauso-L/Digital-electronics-2