LAB 01 – TOOLS

Josef Hula/212557/xhulaj00

1.

Directory URL : https://github.com/xhulaj/Digital-electronics-2

2.

Explanation of basic bitwise operators

This is a sign for an *OR* operator. It is a binary bitwise operator. Its output is 1, if 1 is in in at least one of coincident bits of operand, or in both, else it is output is 0.

&

This is a sign for an *AND* operator. It is a binary bitwise operator. Its output is 1, if both coincident bits are 1, else it is 0.

^

This is a sign for a *XOR* operator. It is a binary bitwise operator. Its output is 1, if 1 is in only one bit of coincident bits of operands, else it is 0.

~

This is a sign for a *COMPLEMENT* operator. It is a unary bitwise operator. Its output is its input, only each bit is inverted.

< <

This is a sign for *LEFT SHIFT* operator. The left operands value is moved left by the number of bits specified by the right operand.

3. **main.c CODE**

```
/* Defines -----------------------------------------------------------*/
#define DOT 200
#define DASH 600
#define SPACE_S 200
#define SPACE_L 1000

#define LED_GREEN   PB5     // AVR pin where green LED is connected
#define SHORT_DELAY 500      // Delay in miliseconds
#ifndef F_CPU
#define F_CPU 16000000      // CPU frequency in Hz required for delay func
```

```
#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
    PORTB = PORTB & ~(1<<LED_GREEN);

    // Infinite loop
    while (1)
    {
        // D
              PORTB = PORTB ^ (1<<LED_GREEN);
        _delay_ms(DASH);
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(SPACE_S);
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(DOT);
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(SPACE_S);
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(DOT);
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(SPACE_L);

              //E
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(DOT);
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(SPACE_L);

              //2
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(DOT);
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(SPACE_S);
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(DOT);
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(SPACE_S);
              PORTB = PORTB ^ (1<<LED_GREEN);
              _delay_ms(DASH);
```

```
                PORTB = PORTB ^ (1<<LED_GREEN);
                _delay_ms(SPACE_S);
                PORTB = PORTB ^ (1<<LED_GREEN);
                _delay_ms(DASH);
                PORTB = PORTB ^ (1<<LED_GREEN);
                _delay_ms(SPACE_S);
                PORTB = PORTB ^ (1<<LED_GREEN);
                _delay_ms(DASH);
                PORTB = PORTB ^ (1<<LED_GREEN);
                _delay_ms(SPACE_L);
                PORTB = PORTB ^ (1<<LED_GREEN);

                //end
                PORTB = PORTB & ~(1<<LED_GREEN);

    }

    // Will never reach this
    return 0;
}

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