UL/FS/MS - 2016
Microcontroller: AVR ATmega328P
Assistant: Gašper Škulj

I/O Ports

```
//Datasheet: 14. I/O Ports

#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRB=0b00100000;
    while (1)
    {
      if((PINB&0b00010000)!=0){
             PORTB^=0b00100000;
             _delay_ms(100);
      }
    }
}
```

Fast 8-bit PWM

```
// Datasheet: 15. 8-bit Timer/Counter0 with PWM

#include <avr/io.h>

int main(void)
{
	TCCR0A|=(1<<WGM00)|(1<<WGM01);//Mode 3 - Fast PWM
	TCCR0B|=(1<<CS00);//Prescaler
	OCR0A=0xFE;// Output Compare Register A
	DDRD|=(1<<PD6);//OC0A pin
	TCCR0A|=(1<<COM0A1);//Clear OC0A on Compare Match, set OC0A at BOTTOM,(non-
inverting mode).
	while(1){
	}
}
```

10-bit Analog-to-Digital Converter

```
// Datasheet: 24. Analog-to-Digital Converter

#include <avr/io.h>

int main(void)
{
        //PINS
        DDRB|=(1<<PB5);//LED 13 as OUTPUT
        PORTB&=!(1<<PB5);//LED 13 OFF

        //ADC
        ADMUX|=(1<<REFS0);//AVCC with external capacitor at AREF pin
        ADMUX|=(1<<ADLAR);//ADC Left Adjust Result
        ADMUX|=(1<<MUX0)|(1<<MUX2);//ADC5

        ADCSRA|=(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);//ADC Prescaler Select Bits
        ADCSRA|=(1<<ADEN);//ADC Enable

        while (1)
        {
                ADCSRA|=(1<<ADSC);//ADC Start Conversion
                while((ADCSRA&(1<<ADSC))!=0){
                }
                if(ADCH>=128){//ADCL and ADCH – The ADC Data Register
                        PORTB|=(1<<PB5);
                }
                else{
                        PORTB&=!(1<<PB5);
                }
        }
}
```

INT0 External Interrupt

```
// Datasheet: 13. External Interrupts

#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)
{
        DDRB |= 1<<PB5;// PB5 as output - LED 13
        PORTB &=~(1<<PB5);//PB5 LOW - LED 13 off
        PORTD |= 1<<PD2;//pull-up resistor on PD2 (INT0)

        EICRA|=1<<ISC01;//The falling edge of INT0 generates an interrupt request
        EIMSK|=1<<INT0;//External Interrupt Request 0 Enable

        sei();//Enable Global Interrupt

        while (1)
        {
        }
}

ISR(INT0_vect)
{
        PORTB^=1<<PB5;//toggle PB5 - LED 13
}
```

16-bit Timer Interrupt

```
// Datasheet: 16. 16-bit Timer

#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)
{
	DDRB|=1<<PB5;//PB5 as output - LED 13
	PORTB&=~(1<<PB5);// PB5 LOW -LED 13 off

	//Timer1
	TCCR1B|=1 << WGM12;//Mode 4 (CTC - Clear Timer on Compare match (CTC) mode)
	TCCR1B|=(1<<CS12);//clk/256 (From prescaler)
	OCR1A=0xFFFF;//Output Compare Register 1 A
	TIMSK1=(1 << OCIE1A);//Timer/Counter1, Output Compare A Match Interrupt Enable

	//enable interrupts
	sei();

	while (1)
	{
	}
}

ISR(TIMER1_COMPA_vect){
	PORTB^=1<<PB5;//Toggle PB5 - toggle LED 13
}
```

USART0 Serial communication

```
//20. USART0 - Serial Communication

#define F_CPU   16000000

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define BUAD    9600
#define BRC     ((F_CPU/16/BUAD) - 1)

#define RX_BUFFER_SIZE  128
#define TX_BUFFER_SIZE  128

char rxBuffer[RX_BUFFER_SIZE];
char txBuffer[TX_BUFFER_SIZE];

uint8_t rxReadPos = 0;
uint8_t rxWritePos = 0;
uint8_t txReadPos = 0;
uint8_t txWritePos = 0;

char readByte=0;

void txWrite(char c);
char rxRead(void);

int main(void)
{
       //serial communication
       UBRR0H = (BRC>>8);
       UBRR0L =  BRC;

       UCSR0B = (1<<RXEN0)|(1<<RXCIE0)|(1<<TXEN0)|(1<<TXCIE0);
       UCSR0C = (1<<UCSZ01)|(1<<UCSZ00);

       //enable interrupts
       sei();

    while (1)
    {
              while(!rxRead()){}
              switch(readByte){
                     case 'a':
                            txWrite('b');
                     break;
                     default:
                     break;
              }
    }
}

char rxRead(void)
{
        if(rxReadPos != rxWritePos)
        {
                readByte = rxBuffer[rxReadPos];
                rxReadPos++;

                if(rxReadPos >= RX_BUFFER_SIZE)
                {
                        rxReadPos = 0;
                }
                return 1;
        }
        return 0;
}

ISR(USART_RX_vect)
{
        rxBuffer[rxWritePos] = UDR0;

        rxWritePos++;

        if(rxWritePos >= RX_BUFFER_SIZE)
        {
                rxWritePos = 0;
        }
}

void txWrite(char c)
{
        txBuffer[txWritePos] = c;
        txWritePos++;

        if(txWritePos >= TX_BUFFER_SIZE)
        {
                txWritePos = 0;
        }

        if(UCSR0A & (1<<UDRE0))
        {
                UDR0 = txBuffer[txReadPos];
                txReadPos++;

                if(txReadPos >= TX_BUFFER_SIZE)
                {
                        txReadPos=0;
                }
        }
}

ISR(USART_TX_vect)
{
        if(txReadPos != txWritePos)
        {
                UDR0 = txBuffer[txReadPos];
        txReadPos++;

        if(txReadPos >= TX_BUFFER_SIZE)
        {
            txReadPos=0;
        }
    }
}
```