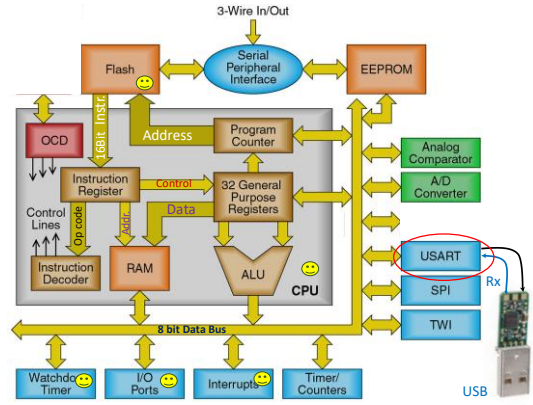
Týždeň	Prednášky	Cvičenia
1.	~~Úvod do predmetu. Číselné sústavy v elektronike, kódovanie čísel. Jazyk C. Úvod do digitálnej elektroniky.~~	~~Inštalácia: – IDE *Microchip Studio* – *driver* pre *Arduino kit*~~
2.	~~Polovodičové pamäte~~	~~Práca s *Microchip Studio*, definovanie projektu~~
3.	~~ATmega328, pamäť, generovanie hodinových signálov, manažment napájania, reset~~	~~Práca s *Microchip Studio* preklad, simulácia projektu~~
4.	~~Vstupy-výstupy~~	~~Práca so vstupmi a výstupmi~~
5.	~~Externé a interné prerušenia, Watchdog časovač~~	~~Práca s prerušeniami~~ ~~Výber zadania na zápočet~~
6.	~~Sériové komunikačné prostriedky USART – Tx~~	~~Práca s USART – TX.~~ ~~Práca na zadaní~~
7.	Sériové komunikačné prostriedky USART - Rx	Práca s USART – RX. Práca na zadaní
8.	TC0 - Normal mode, Fast PWM mode	Práca s TC0. Práca na zadaní
9.	TC0 – Fázovo korektná a CTC PWM	Práca s TC0. Práca na zadaní
10.	A/D prevodník	Programovanie A/D. Práca na zadaní
11.	EEPROM, *FUSE* bity	Programovanie EEPROM. Práca na zadaní
12.	I2C zbernica	**Zápočet - Praktická práca s MCU**
13.	Predtermín	**Zápočet - Praktická práca s MCU**

Architektúra mikrokontrolerov AVR



ATmega328P – Univerzálny Synchrónny/Asynchrónny Prijímač/Vysielač – USART *(24.)* (90-104)

UDR0 – *USART Data Register 0*

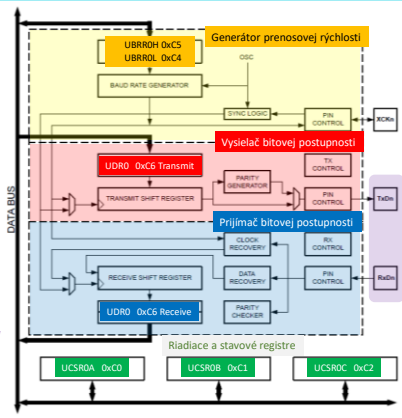
UCSR0A, UCSR0B, UCRS0C
USART Control and Status Register 0 A ,B, C

UBRR0L - *USART Baud Rate Register 0 Low*
UBRR0H - *USART Baud Rate Register 0 High*

UBRR0 = UBRR0H * 0x100 + UBRR0L

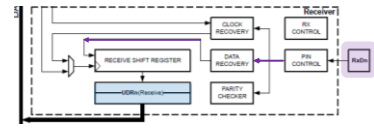
Režimy USART-u a určenie prenosovej rýchlosti: *(Table 24-1)*

- **Asynchrónny normálny mód**
$$Baud = \frac{f_{osc}}{16 * (UBRR0 + 1)}$$
- Asynchrónny 2x rýchlosť mód
$$Baud = \frac{f_{osc}}{8 * (UBRR0 + 1)}$$
- Synchrónny *Master* mód (SPI – *Serial Peripheral Interface*)
$$Baud = \frac{f_{osc}}{2 * (UBRR0 + 1)}$$



ATmega328P – režim UART – príjem dát *(24.8)*

- Príjem dát cez vývod Rx (PD2) sa povoľuje nastavením bitu UCSR0B.RXEN.
- Príjem dát sa začne, keď sa zistí platný štart bit na vývode Rx.
- Každý bit, ktorý nasleduje po štart bite, sa zapisuje do posuvného registra.
- Po prijatý stop bitu je *frame* kompletný a v posuvnom registri sa nachádza práve prijatý bajt.
- Obsah posuvného registra (prijatý bajt) sa presunie do dátového registra UDR0 a nastaví sa bit UCSR0A.RXC0, ktorý signalizuje, že bajt bol prijatý. UDR0 môže prijať až dva bajty.



Bit 7 – RXC0: USART Receive Complete
Bit 6 – TXC0: USART Transmit Complete
Bit 5 – UDRE0: USART Data Register Empty
Bit 4 – FE0: Frame Error
Bit 3 – DOR0: Data OverRun
Bit 2 – UPE0: USART Pariti Error
Bit 1 – U2X0: Double The USART Transmission Speed
Bit 0 – MPCM0: Multi-processor Communication Mode

Name: UCSR0A
Offset: 0xC0
Reset: 0x20
Property: -

USART Control and Status Register 0 A

Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

```
Prenos 5 až 8 dátových bitov
unsigned char USART_Receive (void)
{
    // Wait for data to be received
    while( !(UCSR0A & (1<<RXC0)) );
    // Get and return received data from buffer
    return UDR0;
}
```


Príklad príjmu znaku z UARTu testovaním bitu RXC0 (*Polling*)

Prijímač USART príznakovým bitom UCSR0A.RXC0 - *Receive Complete* (príjem kompletný) signalizuje, že v dátovom registri UDR0 sú prítomné nespracované dáta procesorom. Testovaním tohto bitu je možné presunúť prijatý bajt do požadovanej premennej.

```
//-------------------------------------------
int main (void) {

    char   znak;

    while(1) {
        if ((UCSR0A & (1<<RXC0))) {
            znak = UDR0;
            USART_Transmit_(znak);
        }
    }
}
//-------------------------------------------
```

Name: UCSR0A
Offset: 0xC0
Reset: 0x20
Property: -

Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Príklad príjmu znaku z Rx testovaním bitu RXC0 (*Polling*)

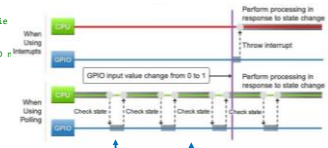
UART_Rxa.c

```
int main (void) {
    char    znak = 'B';          //inicializacia funkcie blikania

    wdt_enable(WDTO_2S);         //zmena WDT na 2sec
    DDRB |= (1<<PORTB5);         //nastavenie vyvodu LED na vystup
    USART_Init (UBRR);           //Inicalizacia UART
    USART_Transmit_Text ("\nTEST Komunikacie Tx a Rx");
    USART_Transmit_Text ("\nZ - Led On");
    USART_Transmit_Text ("\nV - Led OFF");
    USART_Transmit_Text ("\nB - Blikanie Led");

    while(1) {
        wdt_reset();
        if ((UCSR0A & (1<<RXC0))) {//ak je znak v UDR0, tak ho nacitaj
            znak = UDR0;
            USART_Transmit (LF); USART_Transmit (CR); USART_Transmit (znak);
            switch (znak) {
                case 'Z': USART_Transmit_Text("-Led On"); PORTB |= (1<<PORTB5); break;
                case 'V': USART_Transmit_Text("-Led Off"); PORTB &= ~(1<<PORTB5); break;
                case 'B': USART_Transmit_Text("-Blikanie Led"); break;
                default:  USART_Transmit_Text("-Napoznam"); break;
            }
        }
        if (znak == 'B') {//ak je znak B blikaj
            PORTB |= (1<<PORTB5); _delay_ms (50);
            PORTB &= ~(1<<PORTB5); _delay_ms (500);
            PORTB |= (1<<PORTB5); _delay_ms (50);
            PORTB &= ~(1<<PORTB5); _delay_ms (1000);
        }
    }
}
```

Príklad príjmu znaku z Rx testovaním bitu RXC0 (*Polling*) a vyslania textu.

UART_Rxb.c

```
int main (void) {
    char    znak = 'B';           //inicializacia funkcie

    wdt_enable(WDTO_2S);          //zmena WDT na 2sec
    DDRB |= (1<<PORTB5);          //nastavenie vyvodu LED p
    USART_Init (UBRR);            //Inicializacia UART
    sei();
    USART_Transmit_Text ("\nTEST Komunikacie Tx a Rx");
    USART_Transmit_Text ("\nR - Reset MCU");
    USART_Transmit_Text ("\nV - Led OFF");
    USART_Transmit_Text ("\nB - Blikanie Led");

    while(1) {
        wdt_reset();
        if ((UCSR0A & (1<<RXC0))) {//ak je znak v UDR0, tak ho nacitaj
            znak = UDR0;
            sprintf(buffer, "\n0x%02X %c-", znak, znak);
            USART_Transmit_Text(buffer);
            switch (znak) {
                case 'R': USART_Transmit_Text("Softverovy reset\n");
                          wdt_enable(WDTO_15MS);
                          while (1);
                case 'Z': USART_Transmit_Text("-Led On"); PORTB |= (1<<PORTB5); break;
                case 'V': USART_Transmit_Text("-Led Off"); PORTB &= ~(1<<PORTB5); break;
                case 'B': USART_Transmit_Text("-Blikanie Led"); break;
                case 'E': USART_Transmit_Text("INT0 Enabled\n");
                          EIFR |= 1<<INTF0; EIMSK |= 1<<INT0; break;
            }
        }
        PORTB |= (1<<PORTB5); _delay_ms (50);
        PORTB &= ~(1<<PORTB5); _delay_ms (500);
        PORTB |= (1<<PORTB5); _delay_ms (50);
        PORTB &= ~(1<<PORTB5); _delay_ms (1000);
        }
    }
}
```



7

ATmega328P – režim UART – príjem s využitím prerušenia *(24.8.3)*

- Ak je bit UCSROB.RXCIE0 nastavený na log. 1 (povolené generovanie prerušenia), potom ak je ukončený príjem (UCSR0A.RXC0), vykoná sa prerušenie 19 - *USART Rx Complete*.

```
ISR USART_RX_vect (void) {

}
```

19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete

Name: UCSR0B
Offset: 0xC1
Reset: 0x00
Property: -

Bit 7 – RXCIE0: *RX Complete Interrupt Enable*

USART Control and Status Register 0 B

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

8

Príklad príjmu textového príkazu s prerušením *USART Rx vect*

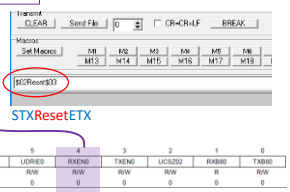
```
#define BUFLEN 10
#define STX 0x02
#define ETX 0x03

char   RxBuf[11], i;

//-----------------------------------------
ISR (USART_RX_vect) //prerusenie od RXCIE0
{
    if (UDR0 == STX) {i = 0; return;}
    if (i < BUFLEN) RxBuf[i++] = UDR0;
    if (UDR0 == ETX) RxBuf[--i] = '\0';
}                                  NULL
//-----------------------------------------
int main (void) {

    while(1) {
        wdt_reset ();
        if (!strcmp (RxBuf, „Reset")) {
            while (1);
        }
    }

}
//-----------------------------------------
```

\0 (*NULL*) sa používa na identifikáciu konca reťazca alebo poľa znakov. Definovaný reťazec je štandardne ukončený s \0.

- zistenie dĺžky reťazca - *strlen ()*,
- spájanie dvoch reťazcov - *strcat ()*,
- porovnanie dvoch reťazcov - *strcmp ()*.

```
char str1[6] = "Hello"; automaticky ukončí reťazec znakom \0
char str2[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
char str3[6] = {0x48, 0x65, 0x6C, 0x6C, 0x6F, 0x00};
```

```
RxBuf[0]-> R
RxBuf[1]-> e
RxBuf[2]-> s
RxBuf[3]-> e
RxBuf[4]-> t
RxBuf[5]-> '\0'
```



STXResetETX

Bit	7	6	5	4	3	2	1	0
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

9

Príklad príjmu znaku z Rx s použitím prerušenia *USART_RX_vect*

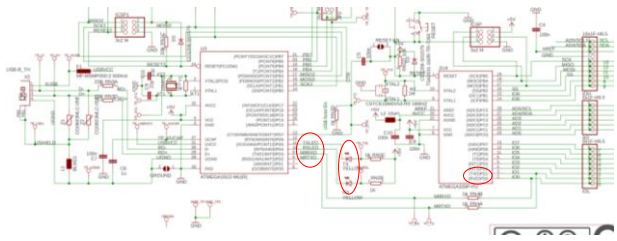
UART_Rxc.c

```
ISR (USART_RX_vect)//Obsluha prerusenie od RX
{
    char    temp, prikaz_ok = '?';

    temp = UDR0;
    if (temp == STX) {index = 0; return;}
    if (index < BUFLEN) RxBuf[index++] = temp;
    if (temp == ETX) {
        RxBuf[--index]='\0';
        USART_Transmit_Text(RxBuf);
        if (!strcmp (RxBuf, "Reset")) {wdt_enable(WDTO_15MS); while (1);}
        if (!strcmp (RxBuf, "Zapni")) {prikaz_ok = '+'; PORTB |= (1<<PORTB0);}
        if (!strcmp (RxBuf, "Vypni")) {prikaz_ok = '+'; PORTB &= ~(1<<PORTB0);}
        if (!strcmp (RxBuf, "EInt0")) {prikaz_ok = '+'; EIFR |= 1<<INTF0; EIMSK |= 1<<INT0;}
        sprintf (RxBuf, "%c\n\n",prikaz_ok);
        USART_Transmit_Text(RxBuf);
    }
    return;
}
//-----------------------------------------
int main (void) {

    wdt_enable(WDTO_2S);
    DDRB |= (1<<PORTB5) | (1<<PORTB0);//nastavenie vyvodov LED a LED1 na vystup
    USART_Init (UBRR);
    INT0_Init ();
    sei();

    USART_Transmit_Text ("\nTEST Komunikacie cez Tx a Rx");
    USART_Transmit_Text ("\nReset - Reset MCU");
    USART_Transmit_Text ("\nZapni - Zapne LED1");
    USART_Transmit_Text ("\nVypni - Vypne LED1");
    USART_Transmit_Text ("\nEInt0 - Povoli ext. prerusenie 0\n");
    while(1) {
        wdt_reset();
        PORTB |= (1<<PORTB5); _delay_ms (50);
        PORTB &= ~(1<<PORTB5); _delay_ms (500);
        PORTB |= (1<<PORTB5); _delay_ms (50);
        PORTB &= ~(1<<PORTB5); _delay_ms (1000);
    }
}
```


10

ATmega328P – režim UART Arduino UNO

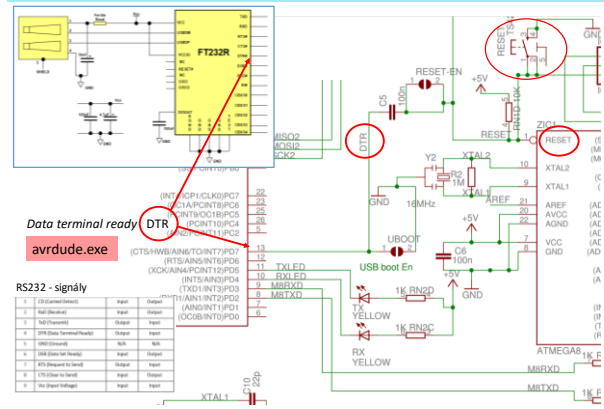
Asynchrónna sériová komunikácia mikrokontroléra ATmega328P sa realizuje pomocou vývodov PD1 (TXD) a PD0 (RXD).



Na vývojovej doske *Arduino UNO* sú tieto vývody prepojené s prevodníkom SERIAL/USB (FT232 alebo, ATmega16), ktorý zabezpečuje obojstrannú komunikáciu mikrokontroléra ATmega328P s počítačom cez USB zbernicu.
Aktivitu na týchto dvoch vodičoch je možné sledovať na pomocou dvoch led diód označených RX a TX, ktoré ovláda prevodník SERIAL/USB.

11

Resetovanie MCU signálom DTR za účelom nahrávania nového programu pomocou *Bootloader-u.*



12