Politechnika Opolska

L A B O R A T O R I U M

PRZEDMIOT:	**Technika Mikroprocesorowa**

KIERUNEK STUDIÓW:	***Automatyka i Robotyka***	ROK STUDIÓW:	***III***
SPECJALNOŚĆ:	**-**		
SEMESTR:	**VI**	ROK AKADEMICKI:	***2019/2020***

Nr ćwiczenia:	**3**	

Temat ćwiczenia:
Obsługa komunikacji UART

Ćwiczenie wykonali:					
Nazwisko:		*Imię:*	*Nazwisko:*		*Imię:*
1.	**Halek**	**Krzysztof**	**2.**	**Heisig**	**Henryk**
3.	**Pryszcz**	**Krzysztof**	**4.**		

Uwagi:	*Data:*	*Ocena za sprawozdanie:*

Termin zajęć:					
Data:	7.03.2020	Dzień tygodnia:	Sobota	Godzina:	15:40

1. Cel ćwiczenia

Celem ćwiczenia było napisanie programu, w którym zostanie wykorzystana komunikacja z mikrokontrolerem za pomocą protokołu UART. Elementem wykonawczym jest mikrokontroler MSP430G2553.

2. Zakres ćwiczenia

Naszym zadaniem było napisanie programu, który ma polegać na obsłudze komunikacji po protokole UART. Program jest wykonywany w momencie odebrania znaku na linii wejściowej (RxD) do której podłączony jest sygnał TxD z komputera sterującego. Mikrokontroler przetwarza kod który jest wywołany przez przerwanie i odpowiada sygnałem na linii komunikacyjnej nadającej (TxD). Komunikacja odbywa się na parametrach 9600 8n1.

Fragment odpowiadający za skonfigurowanie protokołu UART:

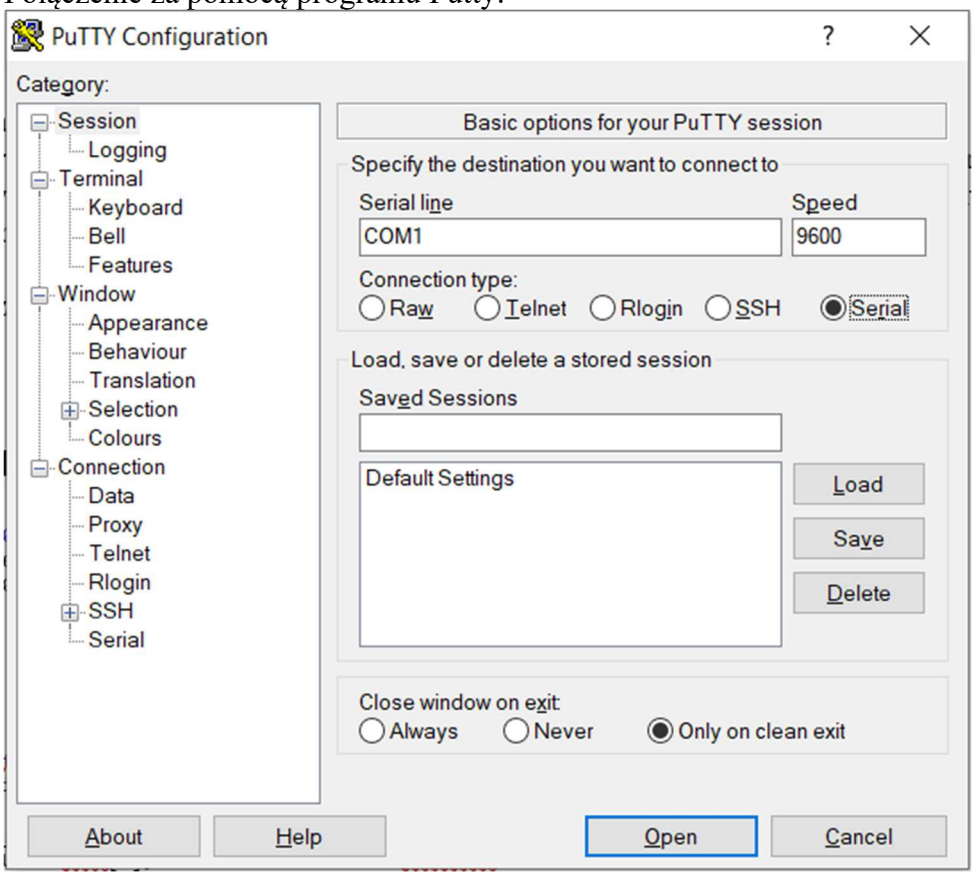
```
// Konfiguracja zegara dla UART
if (CALBC1_1MHZ==0xFF)                  // Check if calibration constant erased
{
    while(1);                           // do not load program
}
DCOCTL = 0;                             // Select lowest DCO settings
BCSCTL1 = CALBC1_1MHZ;                  // Set DCO to 1 MHz
DCOCTL = CALDCO_1MHZ;
// Ustawienie pinów P1.1 i P1.2 jako Rx i Tx
P1SEL = BIT1 + BIT2 ;                   // Select UART RX/TX function on P1.1,P1.2
P1SEL2 = BIT1 + BIT2;

// Konfiguracja parametrów transmisji UART
UCA0CTL1 |= UCSSEL_2;                   // UART Clock -> SMCLK
UCA0BR0 = 104;                          // Baud Rate Setting for 1MHz 9600
UCA0BR1 = 0;                            // Baud Rate Setting for 1MHz 9600
UCA0MCTL = UCBRS_1;                     // Modulation Setting for 1MHz 9600
UCA0CTL1 &= ~UCSWRST;                   // Initialize UART Module
// przerwanie uruchamiane poprzez odebranie sygnału na pinie Rx
IE2 |= UCA0RXIE;                        // Enable RX interrupt
```

Fragment kodu odpowiadający uruchamiany po odebraniu znaku:

```
#pragma vector=USCIAB0RX_VECTOR         // Przerwanie uruchamiane po odebraniu znaku
__interrupt void USCI0RX_ISR(void)
{
    while (!(IFG2&UCA0TXIFG));          // Uruchomienie odbieranie znaków
        if (flaga==1){
            UCA0TXBUF = UCA0RXBUF;      // echo - wypisanie do terminala odebranego znaku
            print("\t- wprowadzony znak\r\n");  //wunkcja wypisania stringa na terminal
            printNumber(UCA0RXBUF);     // funkcja zamiany wprowadzonego stringa jako int i
wypisanie go na terminal
            print("\t- kod w ASCII\r\n");
        }
        else{
            UCA0TXBUF = UCA0RXBUF;      // echo - wypisanie do terminala odebranego znaku
            P1OUT ^= GRN;               // zmiana stanu na przeciwny diody zielonej
            P1OUT ^= RED;               // zmiana stanu na przeciwny diody czerwonej
            print("\r\n");
        }
       }
```

Reszta kodu pozostała z poprzedniego ćwiczenia:
Przerwanie wykonane z przycisku zmienia sterowanie wyjściami między zapalaniem diod oraz wysyłaniem danych przez UART, ma to na celu pokazanie ze można się komunikować dwustronnie z płytką oraz jednostronne sterowanie wyjściami.

Połączenie za pomocą programu Putty:

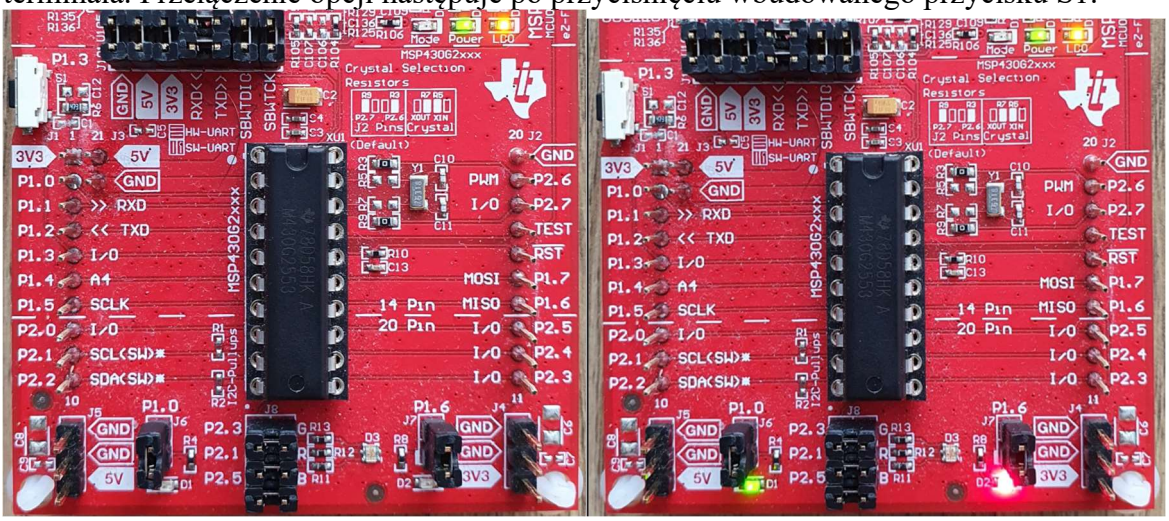


Przykładowe dane z terminala:

COM3 - PuTTY

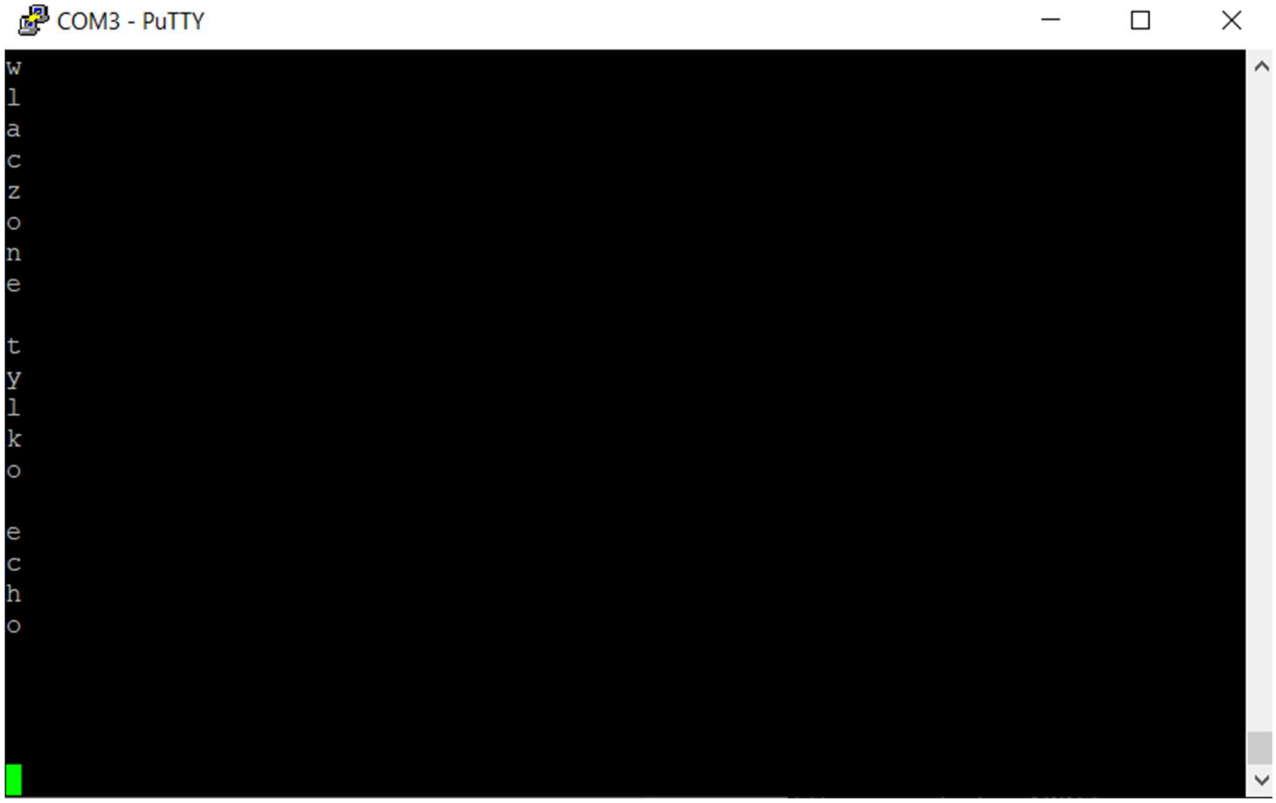
```
49      - kod w ASCII
2       - wprowadzony znak
50      - kod w ASCII
e       - wprowadzony znak
101     - kod w ASCII
r       - wprowadzony znak
114     - kod w ASCII
y       - wprowadzony znak
121     - kod w ASCII
u       - wprowadzony znak
117     - kod w ASCII
j       - wprowadzony znak
106     - kod w ASCII
g       - wprowadzony znak
103     - kod w ASCII
d       - wprowadzony znak
100     - kod w ASCII
s       - wprowadzony znak
115     - kod w ASCII
c       - wprowadzony znak
99      - kod w ASCII
j       - wprowadzony znak
106     - kod w ASCII
```

Dodatkową opcją jest sterowanie zapalaniem diod przez wysłanie dowolnego znaku z terminala. Przełączenie opcji następuje po przyciśnięciu wbudowanego przycisku S1.



Na zdjęciach widać zmienione położenie zworek aby połączyć się z mikrokontrolerem przez HW-UART.

W programie zostało włączone „echo” aby było widać wpisany znak:



3. Kod programu

```
#include <msp430.h>
#define RED BIT6                        // Red LED -> P1.6
#define GRN BIT0                        // Green LED -> P1.0
#define SW BIT3                         // Switch -> P1.3

int flaga = 1;                          // ustawienie zmiennej flaga


void print(char *text){                 // funkcja wysyłająca string na wyjście Tx znak po znaku
    unsigned int i = 0;
    while(text[i] != '\0')
    {
        while (!(IFG2&UCA0TXIFG));      // Check if TX is ongoing
        UCA0TXBUF = text[i];            // TX -> Received Char + 1
        i++;
    }
}

void printNumber(unsigned int num){     // funkcja pobierająca string i zamieniająca go na int
    char buf[6];
    char *str = &buf[5];

    *str = '\0';

    do
    {
        unsigned long m = num;
        num /= 10;
        char c = (m - 10 * num) + '0';
        *--str = c;
    } while(num);

    print(str);
}
```

```
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;           // Stop Watchdog

    //Konfiguracja I/O
    P1DIR |= RED;                       // Set LED pin -> Output
    P1OUT &= ~RED;                      // Wyłączenie diody zielonej
    P1DIR |= GRN;                       // Set LED pin -> Output
    P1OUT &= ~GRN;                      // wyłączenie diody czerwonej
    P1DIR &= ~SW;                       // Ustawienie wejścia na pinie P1.3 (SW2)
    P1REN |= SW;                        // włączenie rezystora podciągającego
    P1OUT |= SW;                        // ustawienie rezystora podciągającego jako pull-up

    // Konfiguracja zegara dla UART
    if (CALBC1_1MHZ==0xFF)              // Check if calibration constant erased
    {
        while(1);                       // do not load program
    }
    DCOCTL = 0;                         // Select lowest DCO settings
    BCSCTL1 = CALBC1_1MHZ;              // Set DCO to 1 MHz
    DCOCTL = CALDCO_1MHZ;
    // Ustawienie pinów P1.1 i P1.2 jako Rx i Tx
    P1SEL = BIT1 + BIT2 ;               // Select UART RX/TX function on P1.1,P1.2
    P1SEL2 = BIT1 + BIT2;

    // Konfiguracja parametrów transmisji UART
    UCA0CTL1 |= UCSSEL_2;               // UART Clock -> SMCLK
    UCA0BR0 = 104;                      // Baud Rate Setting for 1MHz 9600
    UCA0BR1 = 0;                        // Baud Rate Setting for 1MHz 9600
    UCA0MCTL = UCBRS_1;                 // Modulation Setting for 1MHz 9600
    UCA0CTL1 &= ~UCSWRST;               // Initialize UART Module
    // przerwanie uruchamiane poprzez odebranie sygnału na pinie Rx
    IE2 |= UCA0RXIE;                    // Enable RX interrupt

    //konfiguracja przerwania z przycisku
    P1IES &= ~SW;                       // Wybranie zbocza narastajacego na wyzwalanie przerwania
(puszczenie przycisku)
    P1IE |= SW;                         // Włączenie przerwania na przycisku

    __bis_SR_register(LPM0_bits + GIE); // Enter LPM0, Enable Interrupt
}

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void){          // Przerwanie wykonane przez naciśnięcie przycisku
    if (flaga==0){
        flaga=1;                        // zmienna flaga odpowiada za wybranie trybu pracy
    }
    else{
        flaga=0;
    }
    P1IFG &= ~SW;                       // Czyszczenie flagi przerwania na Przycisku
}

#pragma vector=USCIAB0RX_VECTOR         // Przerwanie uruchamiane po odebraniu znaku
__interrupt void USCI0RX_ISR(void)
{
    while (!(IFG2&UCA0TXIFG));          // Uruchomienie odbieranie znaków
        if (flaga==1){
            UCA0TXBUF = UCA0RXBUF;      // echo - wypisanie do terminala odebranego znaku
            print("\t- wprowadzony znak\r\n");  //wunkcja wypisania stringa na terminal
            printNumber(UCA0RXBUF);     // funkcja zamiany wprowadzonego stringa jako int i wypisanie go na
terminal
            print("\t- kod w ASCII\r\n");
        }
        else{
            UCA0TXBUF = UCA0RXBUF;      // echo - wypisanie do terminala odebranego znaku
            P1OUT ^= GRN;               // zmiana stanu na przeciwny diody zielonej
            P1OUT ^= RED;               // zmiana stanu na przeciwny diody czerwonej
            print("\r\n");
        }
}
```

4. Wnioski

Napisany program w środowisku Code Composer Studio został zapisany w pamięci mikrokontrolera MSP430G2553. Wykorzystując Lauchpad możemy użyć wbudowanego w niego konwertera USB<->UART i nawiązać komunikację z komputerem. Komunikacja została nawiązana za pomocą programu Putty ustawionego na łączenie się przez port COM3.