

LABORATORIUM

PRZEDMIOT:
Technika Mikroprocesorowa
KIERUNEK STUDIÓW:
AUTOMATYKA I ROBOTYKA
ROK STUDIÓW:
3
SPECJALNOŚĆ:
-
SEMESTR:
6
ROK AKADEMICKI:
2019/2020
Temat ćwiczenia:
Interfejs szeregowy UART służący do komunikacji – program 4
Ćwiczenie wykonali:

<u>Nazwisko:</u>	<u>Imię:</u>	<u>Nazwisko:</u>	<u>Imię:</u>
1. Idzi	Dawid	2. Pawlak	Kamil

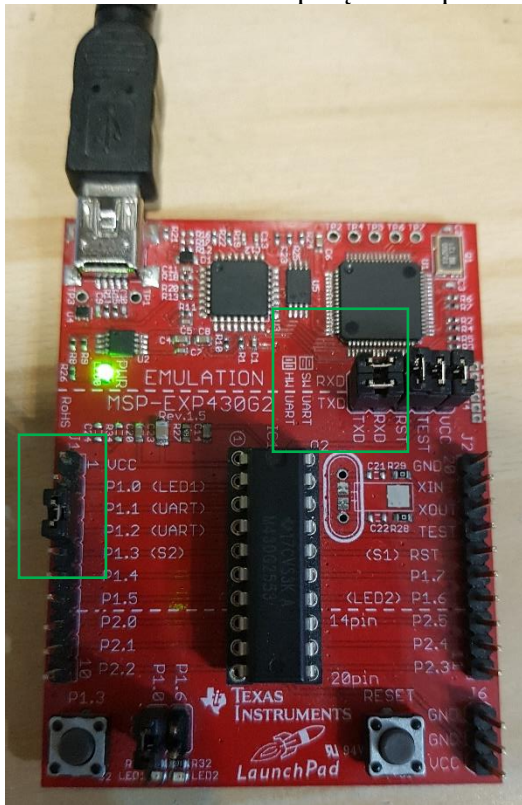
Uwagi:
Data:
Ocena za sprawozdanie:
Zajęcia zdalne.

1. Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z interfejsem UART. Przy wykorzystaniu mikrokontrolera MSP430G2 oraz języka programowania C. Szeregowy interfejs UART służy do przesyłania danych.

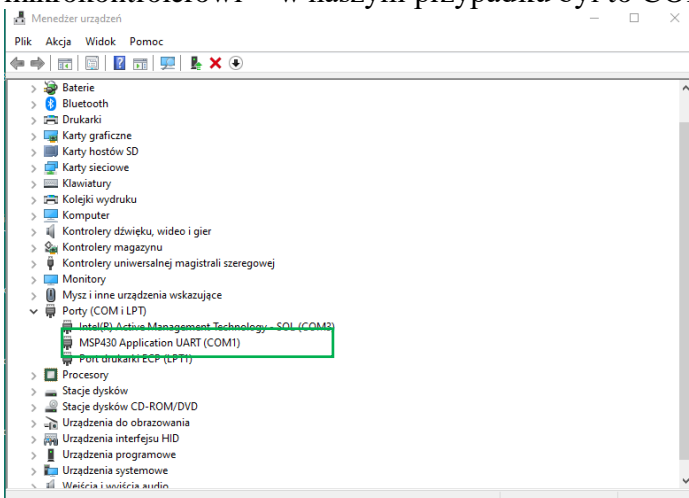
2. Zakres ćwiczenia

Aby zrealizować to ćwiczenie należało wiedzieć, iż mikrokontroler z rodziny MSP wspiera UART w trybie sprzętowym jak i programowym. W naszym przypadku użyliśmy sprzętowego i z kolei do tego wymagane było przestawienie zworek konfiguracyjnych (TXD oraz RXD, gdzie ta TXD służy do wysyłania pakietów danych, a RXD do odbierania) na mikrokontrolerze oraz połączenie pinów P1.1 oraz P1.2 – aby komunikacja mogła zaistnieć.



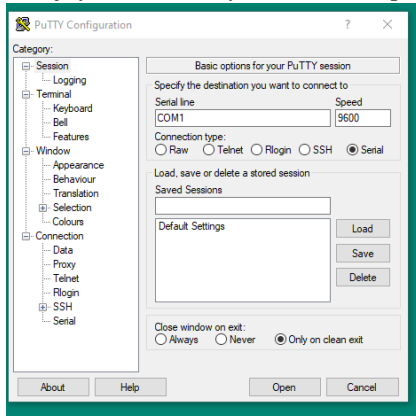
Rysunek 1 – widok mikrokontrolera ze zmienionymi ustawieniami zworek.

Następnie należało odnaleźć port z jaki został przypisany przez system Windows mikrokontrolerowi – w naszym przypadku był to COM1.



Rysunek 2 – widok okna menedżera urządzeń systemu Windows 10.

Kolejnym krokiem było ustawienie programu PuTTY do połączenia się z naszym kontrolerem.



Rysunek 3 – widok ustawień aplikacji Putty.

Po tych krokach można było przejść do napisania kodu programu.

3. Kod programu

```
#include <msp430.h>

void print(char* text)
{
    unsigned int i = 0;
    while (text[i] != '\0')
    {
        while (!(IFG2 & UCA0TXIFG)); // sprawdzenie czy if TX is ongoing
        UCA0TXBUF = text[i];         // TX -> Received Char + 1
        i++;
    }
}

void printNumber(unsigned int num)
{
    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
    if (CALBC1_1MHZ == 0xFF) // Sprawdź, czy stała kalibracji została
    //
    {
        while (1);
    }
    DCOCTL = 0; // Select lowest DCO settings
    BCSCTL1 = CALBC1_1MHZ; // Set DCO to 1 MHz
    DCOCTL = CALDCO_1MHZ;
```

```

P1SEL = BIT1 + BIT2;           // Select UART RX/TX function on P1.1,P1.2
P1SEL2 = BIT1 + BIT2;

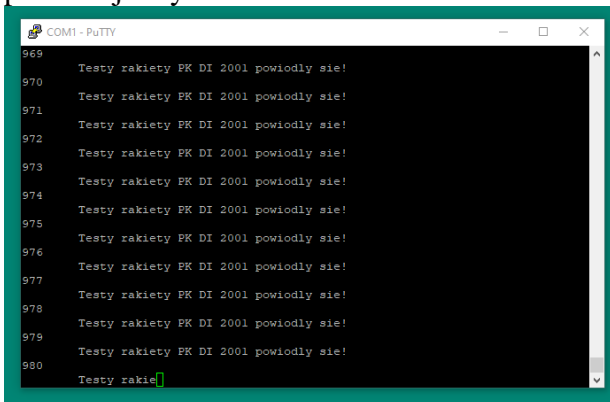
UCA0CTL1 |= UCSSEL_2;          // UART Clock -> SMCLK
UCA0BR0 = 104;                 // Baud Rate Setting for 1MHz 9600
UCA0BR1 = 0;                   // Baud Rate Setting for 1MHz 9600
UCA0MCTL = UCBRS_1;            // Modulation Setting for 1MHz 9600
UCA0CTL1 &= ~UCSWRST;          // Initialize UART Module

unsigned int count = 0;
while (1)
{
    printNumber(count);         //wypisz liczbę
    print("\r\n");              //wypisz pustą linię
    print("\tTesty rakiety PK DI 2001 powiodly sie!\r\n"); //wypisz dany
                                                                    tekst
    count++;                    //inkrementuj liczbę
    __delay_cycles(10000);
}
}

```

4. Działanie programu na obiekcie rzeczywistym

Aby ujrzeć działanie naszego zostało otwarte połączenie przez program Putty. Poniższy zrzut prezentuje wyniki.



Rysunek 4 – okno programu Putty

5. Wnioski

Ćwiczenie pokazało nam czym jest szeregowy interfejs UART oraz jak go można wykorzystywać w praktyce. Pokazany przykład został zaprezentowany w prostej formie, jednakże nic nie stanowi problemu, aby go rozbudować i wykorzystać ten interfejs do informowania, że jakaś operacja została wykonana. Przy projektowaniu takiego interfejsu komunikacji należy mieć na uwadze wszystkie wady interfejsów szeregowych oraz ich opóźnienia. Ćwiczenie zostało wykonane pomyślnie.