|  |  |
| --- | --- |
| logo_PO | **Politechnika Opolska** |

**LABORATORIUM**

|  |
| --- |
| **Technika Mikroprocesorowa** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Kierunek studiów: | AiR Ns | | Rok studiów: | | III |
| Semestr: | VI | Rok akademicki: | | 2019/2020 | |

|  |
| --- |
| *Temat ćwiczenia:* |
| Program UART |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| *Projekt wykonali:* | | | |
| *Nazwisko i imię:* | | *Nazwisko i imię:* | |
| **1.** | Szymon Słaboń | **2.** | Rychel Konrad |
| **3.** | Syguła Dariusz | **4.** |  |

|  |  |  |
| --- | --- | --- |
| *Ocena:* | *Data:* | *Uwagi:* |
|  |  |  |

**Wstęp**

Postawione przed nami zadanie polegało a napisaniu kodu do mikrokontrolera MSP-EXP430G2 firmy Texas Instruments. Ćwiczenie polegało na stworzeniu programu operującego na pinach 1.1 oraz 1.2.

**Opis programu**

Po uruchomieniu programu zaświeca się zaświecą się obydwie diody, a na wirtualnej konsoli pokaże się aktualna wartość licznika i.

Na początku programu widzimy instrukcję warunkową sprawdzającą stan stałej kalibracji, jeżeli jej nie ma program się nie uruchomi. Stała ta odpowiada za częstotliwość jednak tylko częściowo, w linii 39 widzimy UCA0BR0=104. Liczba to dzielnik przez który dzieli się częstotliwość zapisaną w stałej kalibracji w tym wypadku 1MHz i tak uzyskujemy właściwą częstotliwość mierzoną w bitach na sekundę (bps). Zabieg ten ma zapobiegać występującym stratom, które są szczególnie widoczne przy większych projektach. Dalej mamy konfigurację przerwań, oraz ich włączenie. Pętle przerwań nadajnika i odbiornika robią praktycznie to samo, zapalają diody z wyjątkiem nadajnika, który to inkrementuje zmienną i. W lini 51 widzimy jak nadaje jej aktualną wartość.

**Skrypt programu:**

|  |
| --- |
| #include "msp430g2553.h"  #define RED\_LED BIT0  #define GREEN\_LED BIT6  #define TXD BIT2  #define RXD BIT1  unsigned int i;  void main(void)  {  WDTCTL = WDTPW + WDTHOLD;    //Konfiguracja zegara    if (CALBC1\_1MHZ==0xFF) // Jeżeli stała kalibracji zostanie skasowana  {  while(1);  }  DCOCTL = 0; // Wybranie najniższych ustawień  BCSCTL1 = CALBC1\_1MHZ; // Ustawianie zakresu  DCOCTL = CALDCO\_1MHZ; // Ustawianie kroku i modulacji DCO    //Konfiguracja LED    P1DIR |= RED\_LED + GREEN\_LED; // P1.0 and P1.6 output  P1OUT &= ~RED\_LED + GREEN\_LED; // P1.0 and P1.6 = 0    //Konfiguracja pinów P1.1 (odbiornika) i P1.2 (nadajnika)//    P1SEL |= RXD + TXD;  P1SEL2 |= RXD + TXD;    //Konfiguracja UART    UCA0CTL1 |= UCSSEL\_2 + UCSWRST; // USCI Clock = SMCLK,USCI\_A0 disabled  UCA0BR0 = 104;  UCA0BR1 = 0;  UCA0MCTL = UCBRS\_1; // Modulation value = 1 from datasheet  UCA0STAT |= UCLISTEN; // włączanie trybu pętli zwrotnej  UCA0CTL1 &= ~UCSWRST; // Inicjowanie automatu stanów    //Konfiguracja przerwań    IE2 |= UCA0TXIE;  IE2 |= UCA0RXIE;    UCA0TXBUF = i;    \_\_bis\_SR\_register(CPUOFF + GIE);  }  // Przerwania nadajnika i odbiornika  #pragma vector = USCIAB0TX\_VECTOR  \_\_interrupt void TransmitInterrupt(void)  {  P1OUT ^= RED\_LED;  i++;  }  #pragma vector = USCIAB0RX\_VECTOR  \_\_interrupt void ReceiveInterrupt(void)  {  P1OUT ^= GREEN\_LED;//light up P1.0 Led on Tx  IFG2 &= ~UCA0RXIFG; // Clear RX flag  } |