Sprawozdanie z laboratorium

Architektury Komputerów

Laboratorium numer: 5

Temat: UART – Uniwersalny asynchroniczny nadajnik-odbiornik

Wykonujący ćwiczenie:

- Patryk Wójtowicz
- Magda Zaborowska

Studia dzienne I stopnia

Kierunek: Informatyka

Semestr: III Grupa zajęciowa: Lab 15

Prowadzący ćwiczenie:
Dr inż. Mirosław
Omieljanowicz

.....

OCENA

Data wykonania ćwiczenia

29.11.2021r.

.....

Data i podpis prowadzący

Cel zadania

Zapoznanie się się z działaniem protokołu UART. Napisanie zaawansowane oprogramowania, które obsłuży następujące czynności:

1. Wysłanie dowolnego znaku przy pomocy przycisków na płytce MSP430 EasyWeb2 do komputera przy użyciu UART, uzyskać jak najszybszą możliwą prędkość transmisji.
2. Odebranie dowolnych znaków tekstowych na płytce MSP430 EasyWeb2 z komputera przy użyciu UART, uzyskać jak najszybszą możliwą prędkość transmisji, wynik przestawić na wyświetlaczu LCD Hitachi 44780.
3. Przy pomocy programu komputerowego PuTTY, połączyć się z portem szeregowym do płytki MSP 430 Easy Web2 i uzyskać najszybszą możliwą prędkość transmisji.

Teoria

UART – jest to protokuł komunikacyjny (asynchroniczny nadajnik – odbiornik), umożliwia on prostą komunikację między urządzeniami go implementującymi. MSP430f149 umożliwia połączenie w trybie full-duplex (dwukierunkowa transmisja). Asynchroniczne działanie pozwala na niejednakowe taktowania zegarów, dzięki czemu urządzenia mogą mieć własne tępo i mimo to komunikacja odbywa się poprawnie. Na płytce MSP430 EasyWeb2 wbudowany jest interfejs standardu RS232 służacy do transmisji UART.

Mikrokontroler zapewnia dwa moduły UART: UART0 oraz UART1, w celach ćwiczeniowych korzystać będziemy z UART0. W płytce MSP430 EasyWeb2 domyślnie piny nie służą do transmisji UART, w tym celu należy ustawić logiczne jedynki w rejestrze P3SEL. Poniżej przedstawiono pełną konfigurację portów:

```
#define Tx 0x10                 // P3.4
#define Rx 0x20                 // P3.5
```

Kod 1 Zdefiniowanie pinów do konfiguracji UART [1]

```
P3SEL |= Tx + Rx;
P3DIR |= Tx;
P3DIR &= ~ Rx;
```

Kod 2 Inicjalizacja pinów do transmisji UART [1]

Aby transmisja była prawidłowa należy dostosować taktowanie zegara do tempa baud rate'u. W tym celu należy wyliczyć wartość podzielnika dla częstotliwości wybranego zegara. Dla zegara ACLK (oscylator LF), który został zastosowany w zadaniu częstotliwość taktowania to 8MHz. Aby wyliczyć podzielnik należy skorzystać z poniższego wzoru:

$$N = \frac{bound\ rate}{częstotliwość\ taktowania}$$

Zatem

$$N = \frac{115200}{8000000} \simeq 69.4444$$

Sygnał należy podzielić na 69.4444 aby otrzymać potrzebny baud rate. Liczba nie jest całkowita co może sprawić problem, ale producent przewidział taki scenariusz i dołączył mechanizm pozwalający w pewnym stopniu skorygować podzielnik używając również część ułamkowej:

$$Bound\ Rate = \frac{częstotliwość\ taktowania}{N} = \frac{częstotliwość\ taktowania}{część\ całkowita + \frac{1}{n}\sum_{i=0}^{n-1} bit_i}$$

Gdzie część całkowitą zapisuje się w rejestrze UBR00 i UBR10, natomiast bity z sumy szeregu widocznj w mianowniku zapisane są w rejestrze UMCTL0. Także aby otrzymać najbardziej dokładny podzielnik w przedstawionej sytuacji UBR00 powinno równać się 69 natomiast UMCTL0 np. 0x0F (4 bity). Wtedy:

$$Bound\ Rate = \frac{8MHz}{69 + 4/8} = \frac{8MHz}{69{,}5} \simeq 115108$$

```
UCTL0 |= SWRST; // Wyłącz UART
UCTL0 &=~SYNC; // UART mode
UCTL0 &=~SPB // Jeden stop bit
UCTL0 &=~PENA // Wyłącz parity checking
UCTL0 |= CHAR; // 8-bit
UTCTL0 |= SSEL0; // Ustaw zegar UART'a na ACLK(8MHz)
UBR00 = 69; // 8MHz/69
UBR10 = 0;
UMCTL0 = 0x0F; // 8MHz/(69 + 4/8)
ME1 |= (UTXE0|URXE0); // Włączenie UART0 (transmisja + odbieranie)
UCTL0 &= ~SWRST; // Włącz uart
IE1 |= URXIE0; // Włączenie przerwań do odbieraniu danych UART
_EINT(); // Włączenie Przerwań
```

Kod 3 Pełna inicjalizacja [1]

Do wysyłania danych służy funkcja z informatora laboratoryjnego. Dane przchowywane są w buforze UART'a w rejestrze TXBUF0. Aby wysłać dane należy zapisać tam dowolną wartość z zakresu 0-255. Należy także pamiętać, że konieczne jest sprawdzenie czy nie zostało przedtem uruchomione przerwanie, zatem należy sprawdzić flagę UTXIFG0 w rejestrze IFG1.

```
void UartCharTransmit(unsigned char znak)
{
 while ( (IFG1 & UTXIFG0)==0);          // Gdy układ nie jest zajęty
 TXBUF0=znak;                           // Wysyłamy znak
}
```

Kod 4 Funkcja wysyłania znaku przez UART [1]

Do dyspozycji mamy także funkcje wysłania całego napisu:

```
void UartStringTransmit(char * napis)
{
  while( *napis)
    UartCharTransmit(*napis ++);
}
```

Kod 5 Funkcja wysłania napisu przez UART [1]

Do odbierania danych pdobonie jak ich wysyłania, dane przechowywane są w buforze znajdujacym się w rejestrze RXBUF0. Przy odbierani danych skorzystanio z przerwań. Gdy pojawiają się nowe dane w buforze, generowane jest przerwanie. Odbieranie danych polega jedynie na odczycie zawartości bufora.

```
#pragma vector=UART0RX_VECTOR
__interrupt void usart0_rx (void)
{
  char ramka = RXBUF0;
}
```

Kod 6 Funkcja odbierania danych z bufora [1]

Realizacja zadania:

Rozpoczęto od zdeklarowania funkcji do transmisji:

```
void initUart( int Speed)
{
int i;
  BCSCTL1 |= XTS;                          // ACLK = LFXT1 = HF XTAL 8MHz
   do
  {
  IFG1 &= ~OFIFG;                          // Czyszczenie flgi OSCFault
    for (i = 0xFF; i > 0; i--);            // Odczekanie
  }
   while ((IFG1 & OFIFG));                 // Dopóki OSCFault jest ciągle ustawiona
   BCSCTL2 |= SELM1+SELM0 ;                // MCLK =LFXT1
   BCSCTL1 |= SELS;

 switch(Speed)                             // Wybieranie prędkości transmisji
  {
   case 115200:{
    ME1 |= UTXE0 + URXE0;                  // Włączenie USART0 TXD/RXD
    UCTL0 |= CHAR;                         // 8-bitów
    UTCTL0 |= SSEL0;                       // UCLK = ACLK
    UBR00 = 0x45;                          // 8MHz/Speed in Bauds
    UBR10 = 0x00;                          //
    UMCTL0 = 0x5B;                         // Modulation
    UCTL0 &= ~SWRST;                       // Inicjalizacja UARTA
    IE1 |= URXIE0;                         // Włączenie przerwań od RX
   break;
   }
  }
  }
```

Następnie napisano program:

```c
#include<msp430x14x.h>
#include "uart.h"
#include "lcd.h"
#include "portyUart.h"
#include "portyLcd.h"

#define KL1 BIT4 &P4IN
#define KL2 BIT5 &P4IN
#define KL3 BIT6 &P4IN
#define KL4 BIT7 &P4IN

char Bufor[32];                          // bufor odczytywanych danych
int low=0;                               // znacznik początku danych w buforze
int high=0;                              // zmacznik końca danych w buforze

void main(void)
{
 WDTCTL=WDTPW + WDTHOLD;                  // wyłączenie WDT

 InitPortsLcd();                          // inicjalizacja portów LCD
 InitLCD();                               // inicjalizacja LCD
 clearDisplay();                          // czyszczenie wyświetlacza
 initPortyUart();                         // inicjalizacja portow UART
 initUart(115200);                        // inicjalizacja UARTa prędkość transmisji 115200 Bitóww

 _EINT();                                 // włączenie przerwań
int i=0;
while(1)                                  // nieskończona pętla
 {

   if( (KL1) ==0)                         //Po naciśnięciu klawisza pierwszego
   {                                      // wysłanie literki P
    UartCharTransmit('P');
   for(long int i=0;i<300000;i++);        //przerwa
   }
   else if( (KL2) ==0)                    //Po naciśnięciu klawisza drugiego
```

Program 1 Kod programu część pierwsza

```c
   {                                      // wysłanie literki U
    UartCharTransmit('U');
    for(long  int i=0;i<300000;i++);      //przerwa
   }
   else if( (KL3) ==0)                    //Po naciśnięciu klawisza trzeciego
   {                                      // wysłanie literki T
    UartCharTransmit('T');
    for(long int i=0;i<300000;i++);       //przerwa
   }
   else if( (KL4) ==0)                    //Po naciśnięciu klawisza czwartego
   {                                      // wysłanie literki Y
    UartCharTransmit('Y');
    for(long int i=0;i<300000;i++);       //przerwa
   }

   while(high != low)                   // Gdy odebrano dane
    {
     putc(Bufor[low]);                  // Następuje wypisanie danych na wyświetlaczu
     if(low%33 ==0)                     // Zmiana pisania na drugi wiersz wyświetlacza
    {
        SEND_CMD(DD_RAM_ADDR2);
    }
    else if(low%17 ==0)                 // Wyczyszczenie ekranu po jego zapełnieniu
    {
```

Program 2 Kod programu część druga

```
61          clearDisplay();
62        }
63        low = (++low)%33;                  // Wyliczanie zmiennej
64        }
65    }
66  }
67  #pragma vector=UART0RX_VECTOR            // Procedura obsługi przerwania UART
68  __interrupt void usart0_rx (void)
69  {
70  Bufor[high]=RXBUF0;                      // Wpisanie odebranych danych do bufora
71  high=(++high)%33;                        // Inkrementowanie znacznika końca danych
72  }
```

Program 3 Kod programu część trzecia

Wnioski

Programowanie UART nauczyło nas wielu zastosowań transmisji asynchronicznej full-duplex. Sama konfiguracja UART umożliwiła uzyskać bit rate deklarowany przez producenta na 115200bit/s. Od teraz wiemy, jak programować systemy zintegrowane z innymi urządzeniami.

Bibliografia

[1] Informator Laboratoryjny
[2] https://pl.wikipedia.org/