Instytut Teleinformatyki

Wydział Fizyki, Matematyki i Informatyki
Politechnika Krakowska

Systemy Wbudowane

„Zastosowanie interfejsów SPI i I2C do komunikacji między układami”

laboratorium: 01

autor: mgr inż. Paweł Pławiak

Kraków, 2015

Spis treści

1. Wiadomości wstępne

Pierwsza część niniejszej instrukcji zawiera podstawowe wiadomości teoretyczne dotyczące omawianego tematu. Poznanie tych wiadomości umożliwi prawidłowe zrealizowanie praktycznej części laboratorium.

1.1. Moduł SPI

SPI jest szybkim, dupleksowym, synchronicznym interfejsem szeregowym. Pozwala on na łączenie jednego układu nadrzędnego z wieloma urządzeniami pobocznymi za pomocą czterech linii sygnałowych. W porównaniu z I^2C, interfejs SPI jest szybszy i prostszy jednak wymaga większej liczby linii sygnałowych i krótszych połączeń.

Bity konfiguracyjne rejestru SPCR:

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- SPIE – włączenie modułu przerwań dla modułu SPI
- SPE – włączenie modułu SPI
- DORD – bitowa kolejność transmisji
- MSTR – wybór trybu pracy
- CPOL – polaryzacja sygnału zegarowego
- CPHA – faza sygnału zegarowego
- SPR 1…0 – częstotliwość taktowania transmisji

Bity konfiguracyjne rejestru SPSR:

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- SPIF – znacznik zakończenia transmisji lub utraty panowania nad magistralą
- WCOL – kolizyjny zapis rejestru SPDR
- SPI2X – podwojenie prędkości transmisji

Bity konfiguracyjne rejestru SPDR:

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

SPDR - rejestr danych.

Szczegółowe informacje na temat interfejsu SPI znajdują się w nocie katalogowej na stronie producenta - http://www.atmel.com/Images/doc2503.pdf natomiast podstawowe informacje można znaleźć pod linkiem:
http://pl.wikipedia.org/wiki/Serial_Peripheral_Interface

1.1.1. Przykładowy kod

Obsługa modułu SPI:

```
#include <avr/io.h>

void SPI_Init(void)
{
DDRB = (1<<PB7)|(1<<PB4);
SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

char SPI_Transmit(char cData)
{
SPDR = cData;
while(!(SPSR & (1<<SPIF)));
return SPDR;
}
```


1.2. Termometr cyfrowy

Zestaw wyposażono w cyfrowy termometr TC77 (Microchip) pracujący na magistrali SPI. Linie sterujące termometrem (SCK, SIO, CS) są dostępne na złączu Con19.

Szczegółowe informacje na temat zasobów układu ewaluacyjnego znajdują się w dokumentacji ZL15AVR - http://dl.btc.pl/kamami_wa/zl15avr.pdf

1.2.1. Schemat podpięcia

SIO (U7 TEM/con8) - PB6 (con16)
CS (U7 TEM/con8) - PB4 (con16)
SCK (U7 TEM/con8) - PB7 (con16)

USART (RS232):
RxD (con7) - PD0 (con18)
TxD (con7) - PD1 (con18)

1.2.2. Pseudokod

Obsługa termometru cyfrowego – przykład komunikacji SPI, odczytywanie temperatury z sensora TC77 i transmitowanie jej wartości do PC za pomocą USART (port RS232).

```
#ifndef F_CPU
 Zdefiniowanie częstotliwości CPU
#endif

#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include "USART.h"
#include "SPI.h"

int main(void)
{
Zadeklarowanie zmiennych

Inicjalizacja USART z odpowiednią częstotliwością
Inicjalizacja SPI

do{
 Włączanie CS
```



```
Odczytywanie bajtów
Wyłącz CS

Przekonwertuj wartości do systemu HEX i wyślij do PC

}while(1);

return 0;
```

Proszę pamiętać o ustawieniu odpowiedniej częstotliwości w terminalu (parametr baud) lub w programie.

1.3. Magistrala I²C – moduł TWI

I²C jest jednym ze standardów w dziedzinie synchronicznych interfejsów szeregowych małego zasięgu. I²C służy do łączenia wielu nadajników-odbiorników do jednej, dwuprzewodowej magistrali szeregowej. Dołączone układy mogą pracować w trybie nadrzędnym lub podrzędnym. Mikrokontroler ATmega32 wyposażono w moduł TWI w celu ułatwienia obsługi I²C. Wspomniany moduł jest zgodny ze standardem I²C, umożliwia pracę z prędkością 400 kbps oraz umożliwia adresowanie do 128 różnych układów podłączonych do jednej magistrali.

Bity konfiguracyjne rejestru TWBR (rejestr szybkości transmisji):

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bity tego rejestru określają prędkość pracy modułu TWI w trybie nadrzędnym.

Bity konfiguracyjne rejestru TWCR (rejestr sterujący):

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- TWINT – znacznik przerwania dla modułu TWI
- TWEA – włączanie generacji potwierdzeń ACK
- TWSTA – generacja sygnału START
- TWSTO – generacja sygnału STOP
- TWWC – znacznik zapisu kolizyjnego
- TWEN – włączenie modułu TWI
- TWIE – uaktywnienie przerwań dla modułu TWI

Bity konfiguracyjne rejestru TWSR (rejestr stanu):

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- TWS7…3 – kod aktualnego stanu modułu TWI
- TWPS1…0 – ustawienie preskalera prędkości transmisji

Bity konfiguracyjne rejestru TWDR (rejestr buforu danych):

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

Rejestr używany jako dwukierunkowy bufor danych, jego zawartość zawsze stanowi ostatnie przesłane przez magistralę słowo.

Szczegółowe informacje na temat magistrali I^2C oraz modułu TWI znajdują się w nocie katalogowej na stronie producenta - http://www.atmel.com/Images/doc2503.pdf natomiast podstawowe informacje można znaleźć pod linkiem: http://en.wikipedia.org/wiki/I%C2%B2C

1.3.1. Przykładowy kod

Obsługa modułu TWI:

```
#include "TWI.h"

#define ADRW_ACK 18
#define DATA_ACK 28
//-------------------------------------
// Generate START signal
//-------------------------------------
void TWI_Start(void)
{
TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
while (!(TWCR & (1<<TWINT)));
}
//-------------------------------------
// Generate STOP signal
//-------------------------------------
void TWI_Stop(void)
{
TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
while ((TWCR & (1<<TWSTO)));
}
```



```
//-------------------------------------
// Write byte to slave
//-------------------------------------
char TWI_Write(char data)
{
TWDR = data;
TWCR = (1<<TWINT) | (1<<TWEN);
while (!(TWCR & (1<<TWINT)));
if((TWSR == ADRW_ACK) | (TWSR == DATA_ACK))
  return 0;
else
  return 1;
}
//-------------------------------------
// Read byte from slave
//-------------------------------------
char TWI_Read(char ack)
{
TWCR = ack
             ? ((1 << TWINT) | (1 << TWEN) | (1 << TWEA))
             : ((1 << TWINT) | (1 << TWEN)) ;
while (!(TWCR & (1<<TWINT)));
return TWDR;
}
//-------------------------------------
// Initialize TWI
//-------------------------------------
void TWI_Init(void)
{
TWBR = 100;
}
//-------------------------------------
// End of file
//-------------------------------------
```


1.4. Zegar czasu rzeczywistego

Zestaw ZL15AVR wyposażono w potencjometr P2, który może zostać wykorzystany do podawania napięcia z zakresu 0...5 V na wejścia przetwornika analogowo-cyfrowego mikrokontrolera AVR. Środkowe wyprowadzenie potencjometru dostępne jest na złączu Con7 (pin oznaczony P2).

Szczegółowe informacje na temat potencjometru analogowego znajdują się w dokumentacji zestaw ewaluacyjny ZL15AVR - http://dl.btc.pl/kamami_wa/zl15avr.pdf

1.4.1. Schemat podpięcia

Zegar czasu rzeczywistego:
SCL (U6 RTC/con7) - PC0 (con17)
SDA (U6 RTC/con7) - PC1 (con17)

USART (RS232):
RxD (con7) - PD0 (con18)
TxD (con7) - PD1 (con18)

1.4.2. Pseudokod

Obsługa zegara czasu rzeczywistego – odczytywanie wszystkich rejestrów zegara czasu rzeczywistego (U6 - M41T00) za pomocą komunikacji I2C oraz transmitowanie ich do PC przez USART.

```
#ifndef F_CPU
  Zdefiniowanie częstotliwości CPU
#endif

#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include "USART.h"
#include "TWI.h"

int main(void)
{

Zadeklarowanie zmiennych
Ustawienie wszystkich linii portu D jako wyjściowe

Inicjalizacja USART z odpowiednią częstotliwością

Inicjalizacja modułu TWI
```



```
do{
Rozpoczęcie transmisji
Wpisanie adresu M41T00
Wskazanie pierwszego rejestru
Powtórzenie startu transmisji
Wpisanie adresu dla odczytywanych danych

for()
 {

 Czytanie danych
 Rozpoczęcie transmisji przez USART
 Konwertowanie danych do kodu ASCII
 Konwertowanie wartości do systemu HEX

 }
Zatrzymanie transmisji

for()
 Opóźnienie 200 ms
}while(1);
return 0;
```

Proszę pamiętać o ustawieniu odpowiedniej częstotliwości w terminalu (parametr baud) lub w programie.

1.5. Zagadnienia do przygotowania

Przed przystąpieniem do realizacji laboratorium należy zapoznać się z zagadnieniami dotyczącymi:

- Magistrali SPI
- Magistrali I^2C i modułu TWI
- Zegara czasu rzeczywistego
- Termometru cyfrowego
- USART

Literatura:

[1] Rafał Baranowski, *„Mikrokontrolery AVR ATmega w praktyce”*
[2] Nota katalogowa mikrokontrolera na stronie producenta – http://www.atmel.com/Images/doc2503.pdf
[3] Dokumentacja zestawu ewaluacyjnego ZL15AVR – http://dl.btc.pl/kamami_wa/zl15avr.pdf
[4] http://pl.wikipedia.org/wiki/Serial_Peripheral_Interface
[5] Instrukcje do poprzednich ćwiczeń laboratoryjnych

2. Przebieg laboratorium

Druga część instrukcji zawiera zadania do praktycznej realizacji, które demonstrują zastosowanie technik z omawianego zagadnienia.

2.1. Zadanie 1. Na ocenę 3.0 (dst)

Proszę uruchomić program obsługi termometru, korzystając z przykładowych kodów na stronie Kamami (http://kamami.pl/zestawy-avr/46782-zl15avr.html). W sprawozdaniu należy zawrzeć opis znaczenia poszczególnych linii kodu.

2.2. Zadanie 2. Na ocenę 4.0 (db)

Proszę uruchomić program wyświetlający wartości wszystkich rejestrów zegara czasu rzeczywistego (U6 - M41T00) w terminalu portu szeregowego na PC przez USART (port RS232) za pomocą komunikacji I2C. Proszę uruchomić program wyświetlający wartość odczytanego czasu z zegara czasu rzeczywistego (U6 - M41T00) w terminalu portu szeregowego na PC przez USART (port RS232) za pomocą komunikacji I2C (TWI pracuje w trybie nadrzędnym). W sprawozdaniu należy zawrzeć opis znaczenia poszczególnych linii kodu.

Uwaga: należy usunąć z kodu linie oznaczone na czerwono

```
#ifndef F_CPU
 #define F_CPU   1000000    // 1 MHz
#endif
```

2.3. Zadanie 3. Na ocenę 5.0 (bdb)

Napisać program prezentujący graficznie wyniki odczytu temperatury.

