Pracownia Transmisji Danych, Instytut Fizyki UMK, Toruń

Instrukcja do ćwiczenia nr 9

Sposoby zabezpieczania transmisji danych przed zakłóceniami

Instrukcję przygotował mgr inż. Jarosław Czoków

I. Cel ćwiczenia

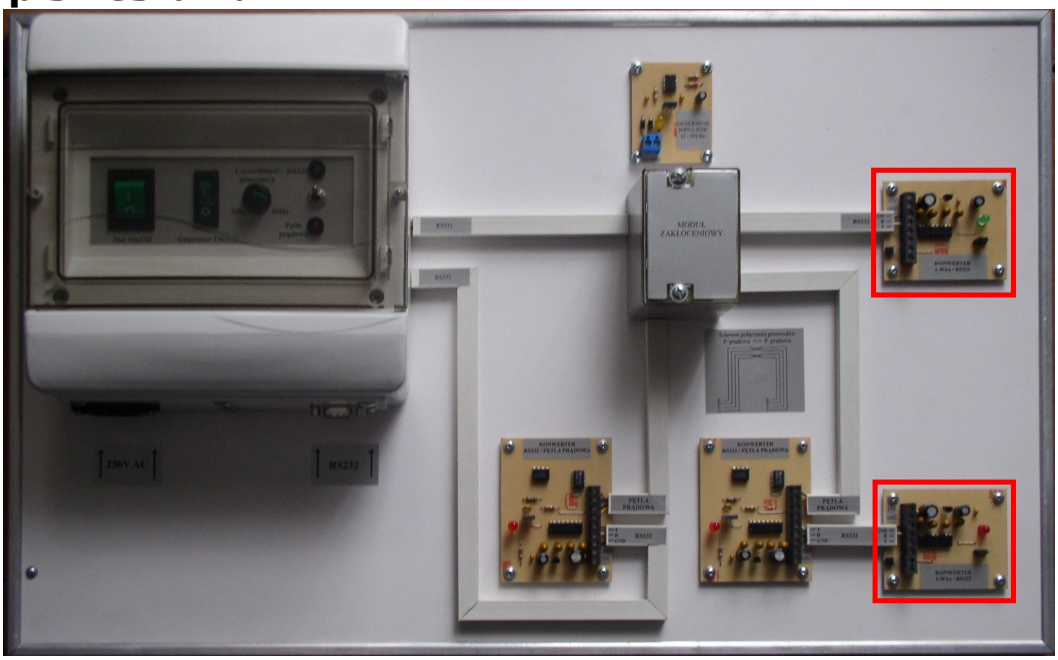
- Poznanie wpływu zakłóceń elektromagnetycznych na transmisję danych
- Zapoznanie się z metodą CRC detekcji błędów w transmisji danych
- Zapoznanie się ze standardem RS232 i transmisja danych w pętli prądowej
- Zapoznanie się z czujnikiem temperatury DS1820

II. Zagadnienia do przygotowania

- Przyczyny zakłóceń w szeregowej transmisji danych na duże odległości
- Podstawy standardów RS232, 1-Wire
- Transmisja danych w pętli prądowej
- Sposoby detekcji błędów w transmisji danych (bit parzystości, **CRC**)
- **Podstawy języka C, w szczególności operacje bitowe C (^,&,|, <<,>>)**

Uwaga! Dobre opanowanie zagadnień pogrubionych konieczne jest do wykonania ćwiczenia!

III. Opis zestawu



Rys.1.Zestaw do transmisji danych w pętli prądowej, czerwonymi ramkami zaznaczono płytki z czujnikami temperatury DS1820

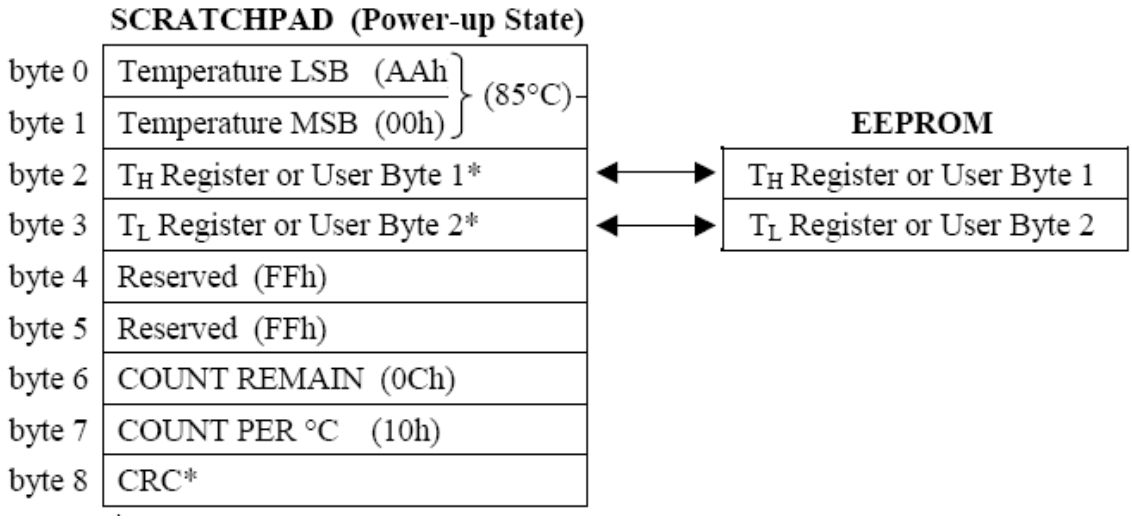
- Zestaw Rys.1 demonstruje wpływ zakłóceń elektromagnetycznych na transmisję danych w standardzie RS232 oraz w pętli prądowej. Na tablicy

znajdują się dwa czujniki temperatury DS1820[1]. Zestaw może pracować zasadniczo w dwóch trybach wybieranych na panelu czołowym: ”RS232” oraz „Pętla prądowa”. W obu tych trybach odczytywane są dane tylko z jednego czujnika. Komunikacja pomiędzy czujnikami a komputerem odbywa się w następujący sposób:

W trybie „RS232”:
-czujnik DS1820 (interfejs 1 Wire) – układ DS2480 [2] (konwerter 1 Wire/RS232) - port COM komputera

W trybie „Pętla prądowa”:
-czujnik DS1820 (interfejs 1 Wire)- układ DS2480 (konwerter 1 Wire/RS232)- konwersja sygnału napięciowego na sygnały w pętli prądowej (kluczowanie tranzystorem BD140) – konwersja na sygnały napięciowe zgodne z RS232 (transoptor HP4200, MAX232)– port COM komputera

W pierwszym trybie przewody TXD i RXD (standard RS232) owinięte są wokół elementu zakłócającego, co po jego włączeniu na panelu czołowym („Generator ON/OFF”), objawia się błędami w transmisji. W drugim trybie, transmisja odbywa się w pętli prądowej. Mimo tego, że przewody są również owinięte wokół elementu zakłócającego, błędy nie występują. Poprawność transmisji sprawdzana jest za pomocą algorytmu CRC (Cyclic Redundancy Check) [3].
Zasadniczą częścią pomiaru jest odczyt pamięci czujnika (scratch’a), mapa pamięci jest pokazana na Rys.2.



Rys.2. Mapa pamięci czujnika DS1820.

Bajt 0 zawiera odczyt temperatury z czujnika. Wartość tego rejestru podzielona przez dwa daje zgrubną temperaturę w stopniach Celcjusza. Bajt 1 zawiera znak odczytanej wartości. Bajty 2 i 3 to wartości graniczne temperatury dla których ma zostać wygenerowany alarm (ustawiane przez użytkownika). Bajty 6 i 7 służą do zwiększenia rozdzielczości pomiaru. Program używa ich aby wg specjalnego wzoru wyliczyć temperaturę z większą rozdzielczością [1].

Bajt 8 zawiera wyliczone CRC dla poprzedzających bajtów.

- Komunikacja komputera PC oraz zestawu odbywa się za pomocą programu *Pętla prądowa*, napisanego w języku C, który odczytuje temperaturę, wyświetla informacje na temat stanu transmisji oraz wyliczoną wartość CRC. Do zmian w kodzie programu używane jest środowisko Dev-C++. Dodatkowo aby jawnie podejrzeć transmisję na porcie COM1 komputera używany jest program Portmon.

  Obsługa programu *Pętla prądowa* jest intuicyjna. Należy zwrócić uwagę na to, że po każdym błędnym odczycie pamięci czujnika (scratch'a)następuje zatrzymanie programu, aż do naciśnięcia przycisku „Wznów". Program zatrzymuje się jedynie w przypadku błędnego odczytu pamięci czujnika gdy ponownie przeliczone CRC będzie różne od zera, natomiast w przypadku innych problemów z komunikacją występujących na skutek zakłóceń, np. z zaadresowaniem czujnika, program podejmuje automatycznie dalsze próby komunikacji.

Uwaga!!! Wszystko co jest potrzebne do wykonania ćwiczenia znajduje się na dysku D w katalogu „PTD09". Należy skopiować jego zawartość do katalogu „D:\users\nazwisko_studenta", a po zakończeniu ćwiczenia skasować zawartość nowo powstałego katalogu.

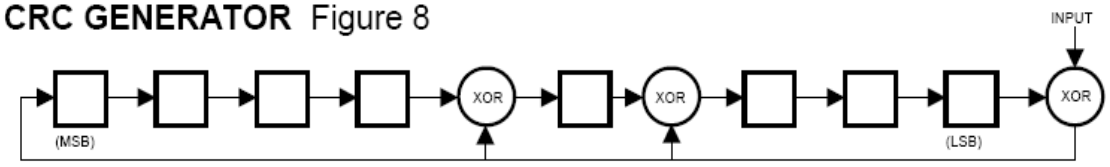
IV. Przebieg ćwiczenia

- Należy zapoznać się z programem *Pętla prądowa* i z działaniem zestawu.
- Zaobserwować występowanie bądź brak zakłóceń w przypadku transmisji w obu trybach.
  - włączyć najpierw program Portmon i ustawić go na śledzenie komunikacji na porcie COM (na pasku: *computer* > *conect local*, oraz zaznaczyć: *capture* > *capture events* oraz *capture* > *ports* > *com1*)
  - włączyć program *Pętla prądowa* (Pętla_prądowa>PetlaPrApp.exe)
  - zaobserwować wyniki transmisji dla obu trybów („RS232" oraz „Pętla prądowa") przy włączonym oraz wyłączonym elemencie zakłócającym („Generator On/Off")
  - odczyt pamięci czujnika następuje po komendzie BE hex wysłanej do czujnika, (przykład Listing.1.), porównać wynik odczytu scratch'a zakłócony z prawidłowym
- Napisać fragment kodu obliczający CRC za pomocą algorytmu odwzorowującego implementację sprzętową.
  - otworzyć źródło programu w środowisku Dev-C++ (Pętla_prądowa > PetlaPrApp (plik projektu środowiska Dev))
  - wskazówki odnośnie pisania samego kodu (gdzie to zrobić i na co uważać) można znaleźć w samym źródle programu w miejscu oznaczonym: „//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX":))
  - po skompilowaniu i uruchomieniu, program powinien wyświetlać wynik CRC obliczonego wg drugiej metody w odpowiednim miejscu, zgodny z wynikiem obliczonym wg pierwszej metody

Wskazówki.

Obliczenia CRC dokonuje się na kilka sposobów [3]. Sprzętową

implementację dla ośmiu bajtów danych (CRC8), przedstawia Rys.3. Dane są kolejno wprowadzane do rejestru przesuwnego, począwszy od najmniej znaczącego bitu bajta zerowego. Po wprowadzeniu ośmiu bajtów, w rejestrze znajduje się wyliczona wartość CRC dla otrzymanych danych. Algorytm CRC ma taką właściwość, że jeżeli wprowadzimy do rejestru jeszcze raz wartość wyliczonego CRC (czyli taką samą jaka się już w nim znajduje), w rejestrze będą same zera. Na tej właściwości opiera się mechanizm sprawdzania poprawności danych odebranych w trakcie transmisji. W pamięci scratch czujnika znajduje się dziewiąty bajt (o nr 8 na Rys.2.), który jest CRC wyliczonym dla pozostałych ośmiu bajtów. Po odebraniu danych z czujnika, dokonujemy ponownego przeliczenia CRC. Jeżeli po wprowadzeniu dziewiątego bajta z pamięci scratch do rejestru, będą się w nim znajdować same zera, to znaczy, że transmisja danych odbyła się bez błędów. W przeciwnym wypadku mamy do czynienia z zakłóconą transmisja. Innym sposobem obliczania CRC jest wygenerowana w specjalny sposób tablica ze wszystkimi możliwymi wartościami CRC. Program *Pętla prądowa* korzysta z tej metody aby obliczyć CRC. Wyniki CRC obliczone obiema metodami powinny się zgadzać.



Rys.3. Schemat blokowy algorytmu CRC.

```
194         0.00002430 Petla.exe      IRP_MJ_WRITE   Serial0 SUCCESS Length 1: BE
195         0.00381613 Petla.exe      IRP_MJ_READ    Serial0 SUCCESS Length 1: BE
196         0.00000475 Petla.exe      IOCTL_SERIAL_GET_COMMSTATUS   Serial0 SUCCESS
197         0.00000587 Petla.exe      IOCTL_SERIAL_PURGE     Serial0 SUCCESS Purge:    TXABORT    RXABORT
TXCLEAR RXCLEAR
198         0.00002263 Petla.exe      IRP_MJ_WRITE   Serial0 SUCCESS Length 1: FF
199         0.00380998 Petla.exe      IRP_MJ_READ    Serial0 SUCCESS Length 1: 31
200         0.00000587 Petla.exe      IOCTL_SERIAL_GET_COMMSTATUS   Serial0 SUCCESS
201         0.00000531 Petla.exe      IOCTL_SERIAL_PURGE     Serial0 SUCCESS Purge:    TXABORT    RXABORT
TXCLEAR RXCLEAR
202         0.00002291 Petla.exe      IRP_MJ_WRITE   Serial0 SUCCESS Length 1: FF
203         0.00382563 Petla.exe      IRP_MJ_READ    Serial0 SUCCESS Length 1: 00
204         0.00000503 Petla.exe      IOCTL_SERIAL_GET_COMMSTATUS   Serial0 SUCCESS
205         0.00000559 Petla.exe      IOCTL_SERIAL_PURGE     Serial0 SUCCESS Purge:    TXABORT    RXABORT
TXCLEAR RXCLEAR
206         0.00002291 Petla.exe      IRP_MJ_WRITE   Serial0 SUCCESS Length 1: FF
207         0.00379518 Petla.exe      IRP_MJ_READ    Serial0 SUCCESS Length 1: 1A
208         0.00000559 Petla.exe      IOCTL_SERIAL_GET_COMMSTATUS   Serial0 SUCCESS
209         0.00000531 Petla.exe      IOCTL_SERIAL_PURGE     Serial0 SUCCESS Purge:    TXABORT    RXABORT
TXCLEAR RXCLEAR
210         0.00002319 Petla.exe      IRP_MJ_WRITE   Serial0 SUCCESS Length 1: FF
211         0.00379490 Petla.exe      IRP_MJ_READ    Serial0 SUCCESS Length 1: 18
212         0.00000559 Petla.exe      IOCTL_SERIAL_GET_COMMSTATUS   Serial0 SUCCESS
213         0.00000531 Petla.exe      IOCTL_SERIAL_PURGE     Serial0 SUCCESS Purge:    TXABORT    RXABORT
TXCLEAR RXCLEAR
214         0.00002347 Petla.exe      IRP_MJ_WRITE   Serial0 SUCCESS Length 1: FF
215         0.00384881 Petla.exe      IRP_MJ_READ    Serial0 SUCCESS Length 1: FF
216         0.00000670 Petla.exe      IOCTL_SERIAL_GET_COMMSTATUS   Serial0 SUCCESS
217         0.00000531 Petla.exe      IOCTL_SERIAL_PURGE     Serial0 SUCCESS Purge:    TXABORT    RXABORT
TXCLEAR RXCLEAR
218         0.00002430 Petla.exe      IRP_MJ_WRITE   Serial0 SUCCESS Length 1: FF
219         0.00380719 Petla.exe      IRP_MJ_READ    Serial0 SUCCESS Length 1: FF
220         0.00000726 Petla.exe      IOCTL_SERIAL_GET_COMMSTATUS   Serial0 SUCCESS
221         0.00000615 Petla.exe      IOCTL_SERIAL_PURGE     Serial0 SUCCESS Purge:    TXABORT    RXABORT
TXCLEAR RXCLEAR
222         0.00002291 Petla.exe      IRP_MJ_WRITE   Serial0 SUCCESS Length 1: FF
223         0.00380970 Petla.exe      IRP_MJ_READ    Serial0 SUCCESS Length 1: 07
```

```
224        0.00000531 Petla.exe      IOCTL_SERIAL_GET_COMMSTATUS   Serial0 SUCCESS
225        0.00000475 Petla.exe      IOCTL_SERIAL_PURGE     Serial0 SUCCESS Purge:    TXABORT    RXABORT
TXCLEAR RXCLEAR
226        0.00002207 Petla.exe      IRP_MJ_WRITE   Serial0 SUCCESS Length 1: FF
227        0.00380523 Petla.exe      IRP_MJ_READ    Serial0 SUCCESS Length 1: 10
228        0.00000447 Petla.exe      IOCTL_SERIAL_GET_COMMSTATUS   Serial0 SUCCESS
229        0.00000503 Petla.exe      IOCTL_SERIAL_PURGE     Serial0 SUCCESS Purge:    TXABORT    RXABORT
TXCLEAR RXCLEAR
230        0.00002263 Petla.exe      IRP_MJ_WRITE   Serial0 SUCCESS Length 1: FF
231        0.00381026 Petla.exe      IRP_MJ_READ    Serial0 SUCCESS Length 1: A3
232        0.00000531 Petla.exe      IOCTL_SERIAL_GET_COMMSTATUS   Serial0 SUCCESS
233        0.00001425 Petla.exe      IOCTL_SERIAL_PURGE     Serial0 SUCCESS Purge:    TXABORT    RXABORT
TXCLEAR RXCLEAR
```

Listing.1. Odczyt mapy pamięci czujnika, 31 hex to odczyt temperatury, a A3 hex to CRC wyliczone dla poprzedzających bajtów, są to dane bez błędów, mogą zostać wykorzystane jako próbne dane do pracy nad algorytmem obliczającym CRC

V. Kryteria oceny ćwiczenia

- Napisanie kodu liczącego CRC.
- Znajomość zagadnień z punktu II

VI. Literatura

[1]. DS18S20 High Precision 1-Wire Digital Thermometer, Nota aplikacyjna DS1820-DS1820S.
[2] DS2480 Serial 1–WireTM Line Driver, Nota aplikacyjna DS2480ds.
[3].Understanding and Using Cyclic Redundancy Checks with Maxim iButton Products, Nota aplikacyjna AN27.