

Interfejsy komunikacyjne mikrokontrolerów

materiały do zajęć laboratoryjnych

Maciej Dzieaniakowski
Grzegorz Wrona

wersja 1.03

Wstęp	3
Podstawowe interfejsy komunikacyjne mikrokontrolerów (wiadomości wstępne)	3
<i>Wprowadzenie</i>	3
UART	7
I²C	10
SPI	13
Ćwiczenia	15
UART	15
<i>Zadanie</i>	15
<i>Przykładowe wyniki pomiarowe</i>	16
I²C	17
<i>Zadanie</i>	17
<i>Przykładowe wyniki pomiarowe</i>	18
Bit-banging (I²C)	19
<i>Zadanie</i>	19
<i>Realizacja</i>	19
SPI	22
<i>Zadanie</i>	22
<i>Przykładowe wyniki pomiarowe</i>	23
Pytania	24
Konfiguracja interfejsów komunikacyjnych (MSP430F169)	25
<i>tryb UART</i>	25
<i>tryb SPI</i>	27
<i>tryb I²C</i>	29
Literatura	31

Niniejszy materiał jest zmodyfikowaną wersją preskrytu opracowanego w ramach programu NERW

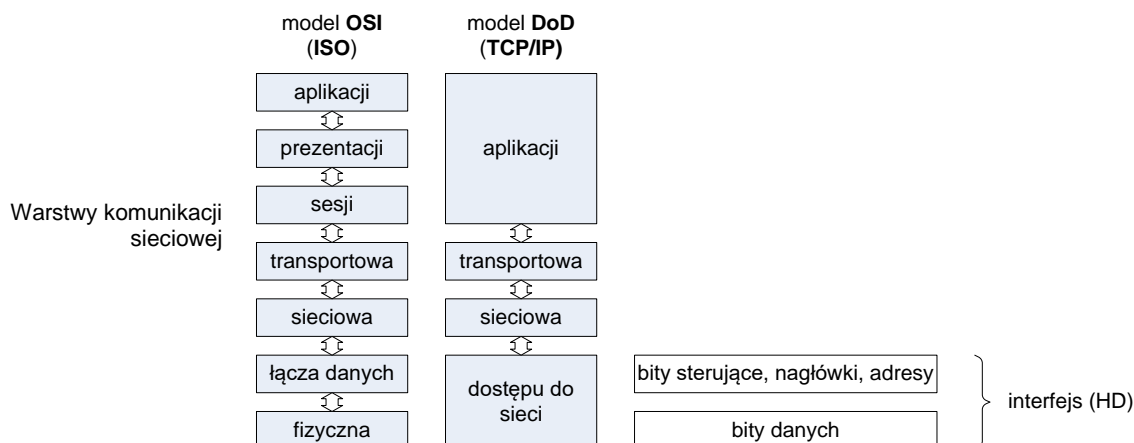
Wstęp

Celem przedmiotu „Interfejsy komunikacyjne mikrokontrolerów” jest zapoznanie uczestników zajęć z zagadnieniami transmisji szeregowej i podstawowymi interfejsami komunikacyjnymi - UART, RS232, SPI, I²C, Przedmiot składa się z wykładu i laboratorium. Niniejszy skrypt dotyczy laboratorium, zawiera jednak podstawowe wiadomości o interfejsach komunikacyjnych, ułatwiając przygotowanym uczestnikom zrozumienie i wykonanie zadań laboratoryjnych. Laboratorium obejmuje programowanie w assemblerze mikrokontrolera jednoukładowego rodziny MSP430 lub języku C. Celem jest stworzenie procedur obsługi wymienionych uprzednio interfejsów komunikacyjnych. Procedury, w tym procedury emulacyjne typu bit-banging, będą wykorzystane do komunikacji MCU z urządzeniami zewnętrznymi wskazanymi przez prowadzących (np. przetworniki ADC/DAC, ekspandery, pamięci, czujniki położenia i temperatury, RTC, układy WiFi, inne systemy μ C, itd.) oraz do komunikacji z innymi stanowiskami laboratoryjnymi.

Podstawowe interfejsy komunikacyjne mikrokontrolerów (wiadomości wstępne)**Wprowadzenie**

We wszystkich układach mikroprocesorowych zachodzi konieczność wymiany danych pomiędzy procesorem, a jego zewnętrznym otoczeniem. W przypadku mikrokontrolerów, w większości niemających wyprowadzonych szyn danych i adresowych, jedyną drogą komunikacji z otoczeniem są wbudowane interfejsy komunikacyjne. Zapewniają one nieomal nieograniczone możliwości rozbudowy układu mikroprocesorowego i kontaktu z innymi systemami procesorowymi.

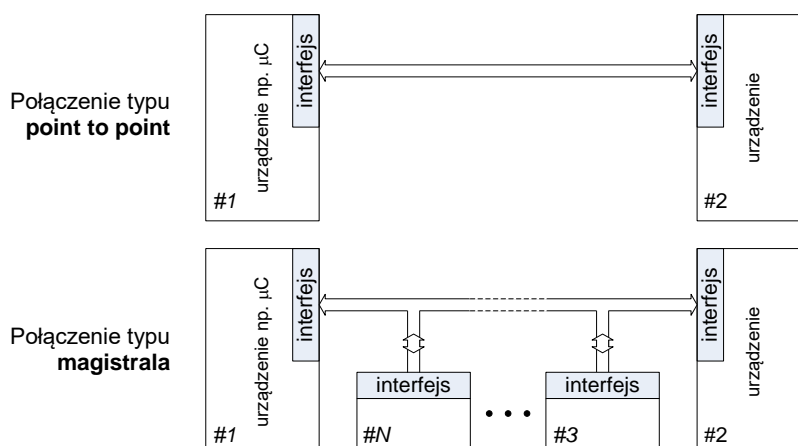
Interfejsy mikrokontrolerów są peryferyjnymi urządzeniami hardwareowymi i możemy je przypisać do grupy urządzeń I/O, będących sprzętowym fragmentem struktury mikrokontrolerów. Wpisują się jednocześnie w modele komunikacji sieciowej, stanowiąc ich najniższą warstwę. Warstwa dostępu do sieci (interfejs) to układ elektroniczny pozwalający na wymianę informacji (bity) z innymi, tego samego typu interfejsami. Poza tym podstawowym zadaniem, wysyłaniem i odbieraniem bitów, interfejsy mogą tworzyć nagłówki, bity sterujące, realizować pewne funkcje kontroli poprawności przesyłanych danych. Funkcje te spełniane są wyłącznie przez hardware interfejsu i są stałe dla danego typu interfejsu.



model OSI - akr. Open Systems Interconnection, lata 1984-1994, org. International Standard Organization

model DoD - akr. Department of Defense, akr. Transmission Control Protocol/Internet Protocol, lata 1973-1982, org. agencja DARPA Departamentu Obrony USA

Połączenia komunikacyjne uzyskiwane dzięki interfejsom, dzielą się w sposób naturalny na dwa rodzaje przedstawione poniżej.

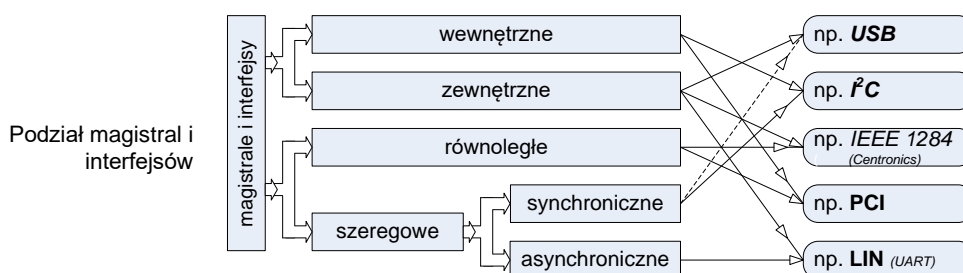


Bezwzględna większość standardowych interfejsów mikrokontrolerów zapewnia stworzenie połączenia sieciowego typu **magistrala**, będącej jednym z typów topologii sieciowych.

Urządzenia komunikujące się ze sobą dzięki magistrali pełnią w czasie transmisji odpowiednie zadania:

- **Master** - urządzenie rozpoczynające i kończące transmisję. W synchronicznych magistralach szeregowych generuje również sygnał zegarowy
- **Slave** - wykonuje polecenia Master'a (nadaje, odbiera, przetwarza, ...)
- **Nadajnik** - w danej chwili nadaje dane
- **Odbiornik** - w danej chwili odbiera dane

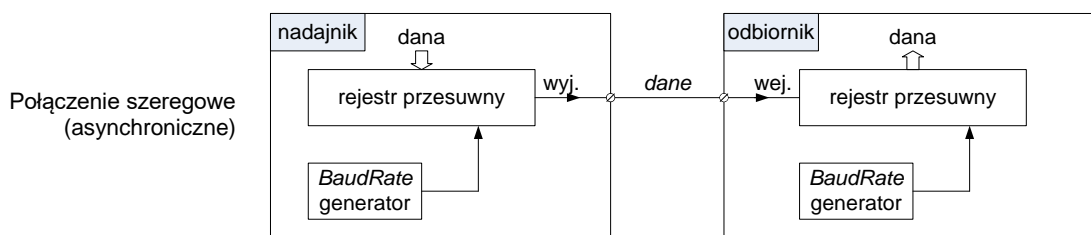
Interfejsy i oparte o nie magistrale, klasyfikowane są w sposób przedstawiony poniżej. Podział interfejsów na „wewnętrzne” i „zewnętrzne” wynika z zadań stawianych przed nimi. Określenie „wewnętrzne” obejmuje te interfejsy, które używane są do komunikacji w obrębie jednego systemu mikroprocesorowego np. do komunikacji pomiędzy procesorem i układami peryferyjnymi (RTC, ADC, pamięć, ...). Pozostałe interfejsy służą głównie do połączeń międzysystemowych (np. system $\mu C \leftrightarrow$ komputer, system $\mu C \leftrightarrow$ system μC , PC \leftrightarrow drukarka, ...).



Drugi podział – interfejsy „równoległe” i „szeregowe” – wynika ze sposobu przesyłania danych.

Interfejsy równoległe, przesyłają w jednym momencie wiele bitów (np. bajt). Nie są jednak obecnie bezpośrednio implementowane w mikrokontrolerach i ich rola w systemach wbudowanych opartych o μC jest ograniczona. Jednak możliwa jest budowa połączenia równoległego poprzez wykorzystanie programowe portów (GPIO) mikrokontrolera.

Interfejsy szeregowe oparte są o rejestry przesuwne. W nadajniku dana jest wprowadzana do rejestru w postaci równoległej (np. bajt) i zgodnie z taktowaniem generowanym przez generator przesuwana bit po bicie na wyjście szeregowe. W odbiorniku wejście szeregowe rejestru jest zatraskiwane, zgodnie z taktowaniem generowanym przez generator, w komórkę wejściowej rejestru i przesuwane dalej aż do wypełnienia rejestru. Dana z rejestru pobierana jest w postaci równoległej.



Szybkość transferu danych określana jest przez BaudRate generator definiujący chwile wysyłania (odbierania) bitów. BaudRate określa szybkość transferu bodu (ang. *baud*). Jeśli bod jest pojedynczym bitem to BaudRate jest równoważny z szybkością (częstotliwością) przesyłu bitów.

Wymaganą cechą połączenia szeregowego jest:

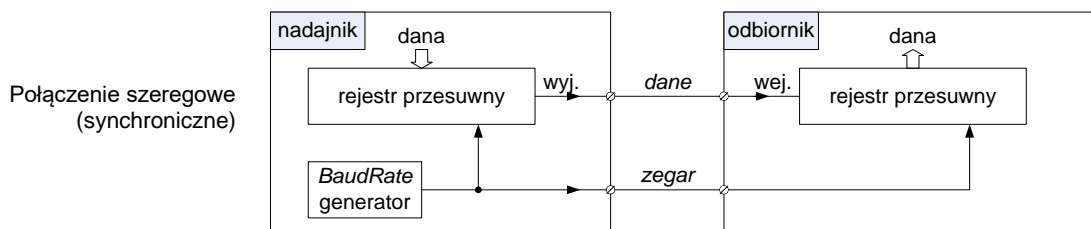
- $BaudRate_{nad} = BaudRate_{odb}$

Jeśli powyższy warunek nie jest spełniony to komunikacja nie będzie przebiegała prawidłowo.

W przypadku połączenia asynchronicznego tzn. takiego, w którym współpracujące urządzenia mają niezależne generatory ustalające szybkość przesyłu, stosowane są różne rozwiązania zapewniające współbieżność nadawania i odbierania kolejnych bitów w transmisji. Zależą one od m.in. rodzaju kodowania (NRZ/RZ) danych.

Typowym przedstawicielem transferu asynchronicznego jest interfejs UART (kodowanie NRZ) i oparte na nim połączenia LIN, RS232, ...

Interfejsy synchroniczne zapewniają identyczność BaudRate strony nadającej i odbierającej. Sygnał synchronizujący, wyznaczający chwile wysłania i odebrania każdego bitu na linii danych przesyłany jest oddzielną linią synchronizacyjną (linią zegara).



Przedstawicielami interfejsów synchronicznych są między innymi I²C i SPI. W obu stosowane jest kodowanie NRZ.

Bez względu na rodzaj stosowanych interfejsów mogą występować ograniczenia w ich pracy, wynikające z właściwości fizycznych kanału transmisyjnego. W przypadku połączeń elektrycznych (przewody, ścieżki płytek drukowanych, ...) zachodzą zjawiska związane z pojęciem linii długiej.

Linia długa to elektryczna linia dwuprzewodowa, której wymiar fizyczny jest porównywalny z długością fali napięcia przesyłanego sygnału.

Taka sytuacja jest typowa w przypadku przesyłania sygnałów prostokątnych. Zawierają one harmoniczne o wysokich częstotliwościach i znacznych amplitudach. Przebiegi w linii długiej muszą być rozpatrywane nie tylko, jako funkcje czasu, lecz również położenia.

Linie transmisyjną uważamy za linię długą, jeżeli spełniony jest jeden z poniższych warunków:

l – długość linii
 c – prędkość światła
 f – częstotliwość przebiegu
 t_0 – czas opóźnienia fali
 t_N – czas zbocza sygnału

$$l \geq c / (4 \times f)$$

dla przebiegów sinusoidalnych

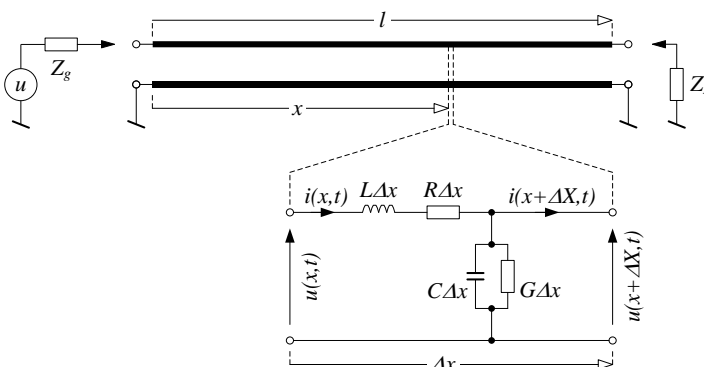
$$t_0 \geq t_N / 2$$

dla przebiegów prostokątnych

Można przyjąć schemat zastępczy linii długiej składający się szeregowo połączonych czwórników skupionych.

Schemat zastępczy linii długiej

R [Ω/m] - rezystancja jednostkowa
 G [S/m] - konduktancja jednostkowa
 L [H/m] - indukcyjność jednostkowa
 C [F/m] - pojemność jednostkowa



Przekształcając odpowiednio równania opisujące czwórnik otrzymujemy równania wiążące napięcie i prąd w linii z czasem i położeniem:

$$LC \frac{\partial^2 u}{\partial t^2} + (RC + LG) \frac{\partial u}{\partial t} + RG u - \frac{\partial^2 u}{\partial x^2} = 0$$

$$LC \frac{\partial^2 i}{\partial t^2} + (RC + LG) \frac{\partial i}{\partial t} + RG i - \frac{\partial^2 i}{\partial x^2} = 0$$

Parametry charakterystyczne linii długiej (linia bezstratna):

$$\vartheta = 1/\sqrt{LC} \quad t_0 = l/\sqrt{LC} \quad Z_0 = \sqrt{L/C}$$

prędkość fali przebiegu czas propagacji impedancja falowa

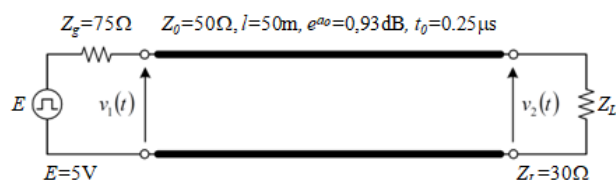
Możemy obliczyć współczynniki odbicia fal na początku i końcu linii:

$$\Gamma_g = \frac{Z_g - Z_0}{Z_g + Z_0} \quad \Gamma_L = \frac{Z_L - Z_0}{Z_L + Z_0}$$

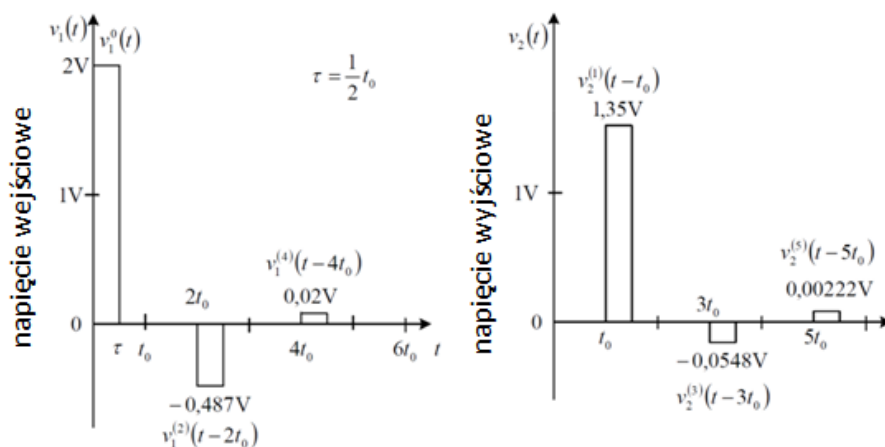
na wejściu linii na wyjściu linii

W przypadku niedopasowania impedancyjnego ($Z_g \neq Z_0$; $Z_L \neq Z_0$) współczynniki odbić fal nie są zerowe i na końcach linii pojawia się superpozycja odbitych fal.

Linia długa niedopasowana impedancyjnie na wejściu i wyjściu [AGH]



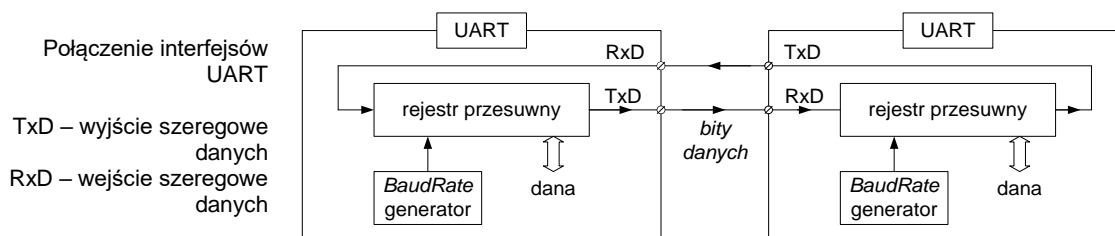
Przebiegi na końcach linii [AGH]



Problem niedopasowania impedancyjnego i wynikających z niego konsekwencji jest od strony technicznej bardzo trudny do rozwiązania, często skutkując koniecznością ograniczenia szybkości transmisji.

UART

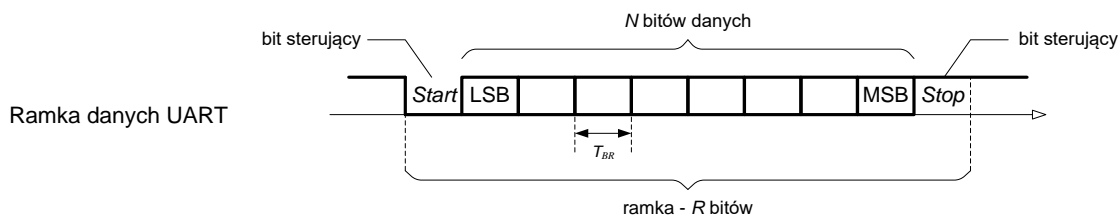
Asynchroniczny interfejs UART (**U**niversal **A**synchronous **R**eceiver **T**ransmitter) został opracowany w latach 60-ych ubiegłego wieku w USA. Do dnia dzisiejszego jest jednym z najczęściej implementowanych interfejsów w mikrokontrolerach. Był opracowany do połączeń typu point-to-point, jednak jego aplikacje pozwalają na tworzenie magistral – LIN, RS485, ...



Przedstawione, typowe połączenie interfejsów UART pozwala na jednoczesny przesył danych w dwu kierunkach (full-duplex) bądź tylko w jednym kierunku (half-duplex). Dane przesyłane są w formie ramek zawierających również bity sterujące:

- **Start** - początek ramki, stan niski (L), 1-b
- **Stop** - koniec ramki, stan wysoki (H), 1-b lub 2-b

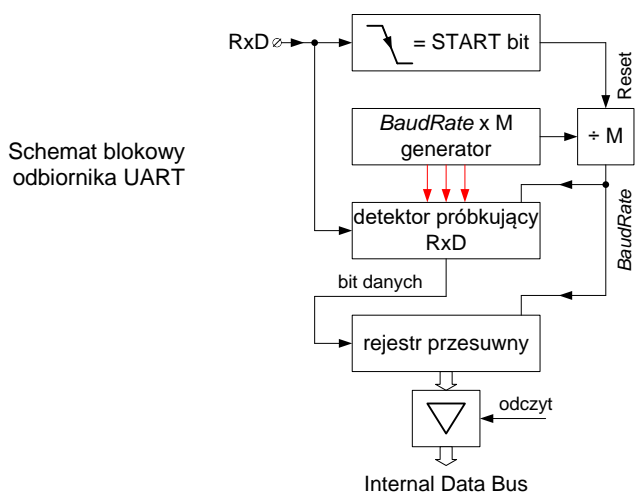
Ilość bitów danych w ramce może być ustawiona i zwiera się pomiędzy 4-b, a 12-b. Standardowa ramka zawiera: 1-b Stop, 8-b danych, 1-b Stop.



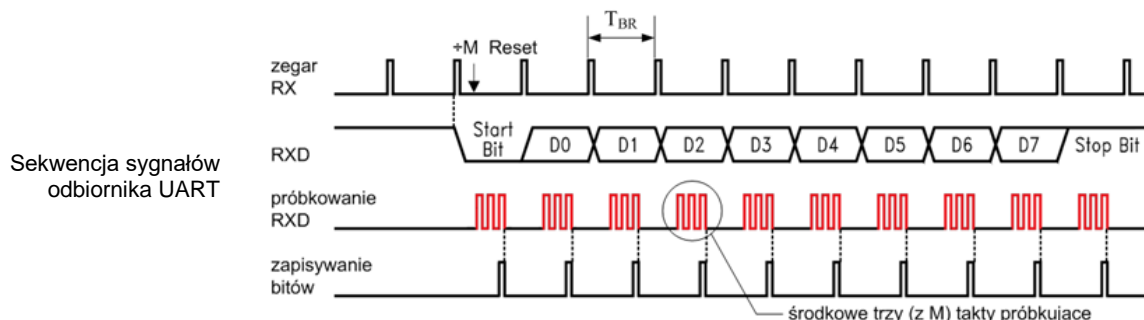
Prawidłowe, bez błędów odebranie wszystkich bitów w ramce wymaga spełnienia warunków:

- $\text{BaudRate}_{\text{nad}} = \text{BaudRate}_{\text{odb}}$
- rozpoczęcie odbioru w chwili pojawienia się bitu Start
- eliminacja zakłóceń transferu pojedynczych bitów

W praktyce nie można uniknąć różnicy nastaw i dryftów generatorów BaudRate oraz błędu wyznaczania początku odbieranej ramki. Mechanizm odbioru danych stosowany w UART pozwala jednak na dużą tolerancję parametrów transferu.



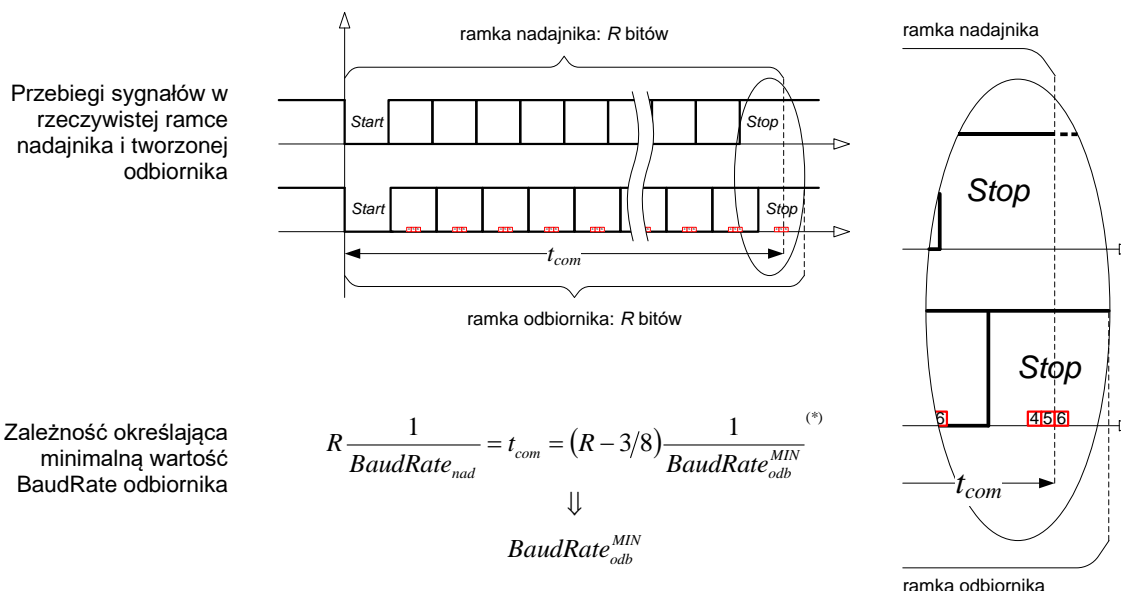
Początek odbieranej ramki określany jest przez komparator wykrywający zbocze opadające. Jest to początek bitu Startu i zliczania M taktów tworzących okres BaudRate odbiornika.



W każdym z trzech środkowych taktów M w okresie BaudRate, pobierana jest logiczna próбка odbieranego sygnału RXD. Jeśli co najmniej dwie sąsiadujące próbki mają tę samą wartość logiczną to wartość ta jest wprowadzana, jako bit odbieranej danej, na wejście rejestru przesuwanego. W przeciwnym wypadku odbiór danych jest wstrzymany, a w odpowiednim rejestrze ustawiana jest flaga przerwania informująca o błędzie transmisji.

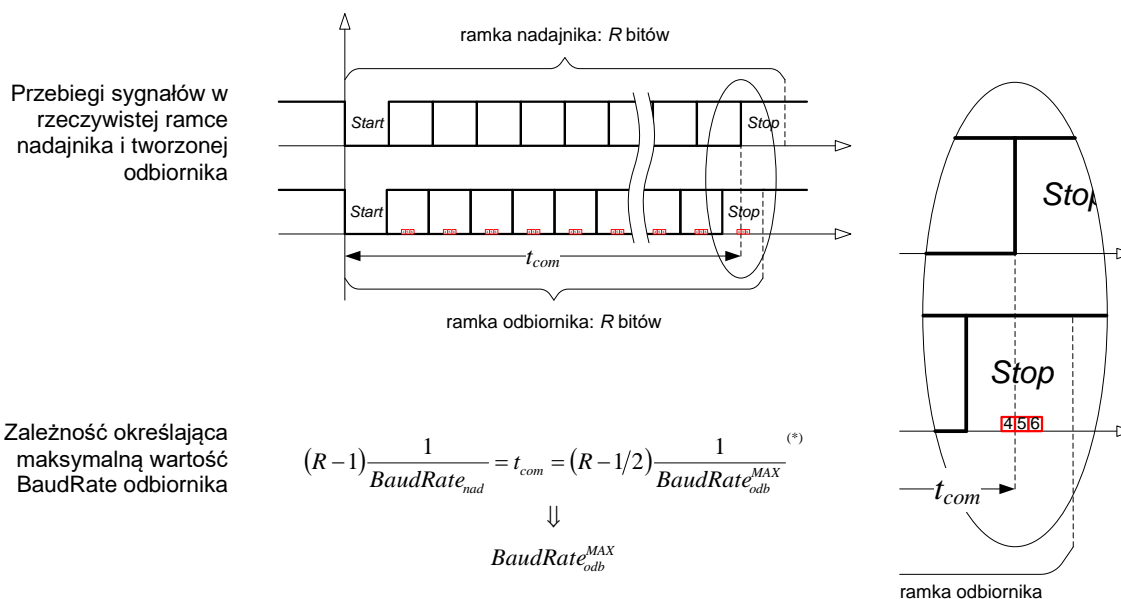
Próbki bitów dla tworzenia ramki odbiornika pobierane są tylko w wyznaczonych chwilach okresu BaudRate, natomiast bity ramki nadajnika pozostają niezmiennie w okresie BaudRate. To oznacza, że jeśli istnieje przesunięcie czasowe środków okresów BaudRate odbiornika i nadajnika (np. z powodu nierówności Baudrate'ów) to nadal istnieje możliwość poprawnego odbioru danych. Przesunięcie to nie może przekroczyć połowy okresu BaudRate, a stąd wynikają graniczne nierówności BaudRate, przy których transmisja przebiega poprawnie.

Minimalna wartość BaudRate odbiornika:



(*) przyjęto ilość taktów M=8 stosowaną w mikrokontrolerach rodziny TMS320Fxxxx (C2000 T)I

Maksymalna wartość BaudRate odbiornika:



(*) przyjęto ilość taktów $M=8$ stosowaną w mikrokontrolerach rodziny TMS320Fxxxx (C2000 TI)

Dla typowej ramki (1-b Start, 8-b danych, 1-b Stop) otrzymujemy:

$$\max. BaudRate_{odb} = BaudRate_{nad} \times 1,0555$$

$$\min. BaudRate_{odb} = BaudRate_{nad} \times 0,9625$$

Jak wynika z uprzednio przedstawionych zależności akceptowalny margines odchyłek częstotliwości BaudRate jest zależny przede wszystkim od długości (R) ramki. Krótka ramka zapewnia duży margines odchyłek BaudRate, a więc zwiększa pewność poprawnego transferu.

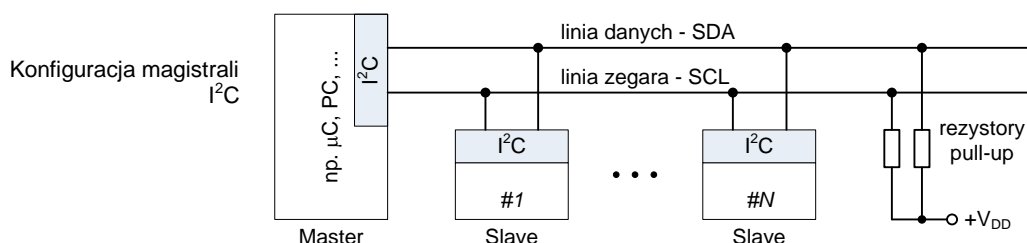
I²C

Synchroniczny interfejs I²C (**Inter Integrated Circuit**) został opracowany w latach 80-ych ubiegłego wieku przez firmę Philips, a następnie był sukcesywnie rozwijany. Dzisiaj istnieje kilka wariantów tego interfejsu, kompatybilnych w dół, oraz interfejsy innych firm oparte o koncepcję I²C.

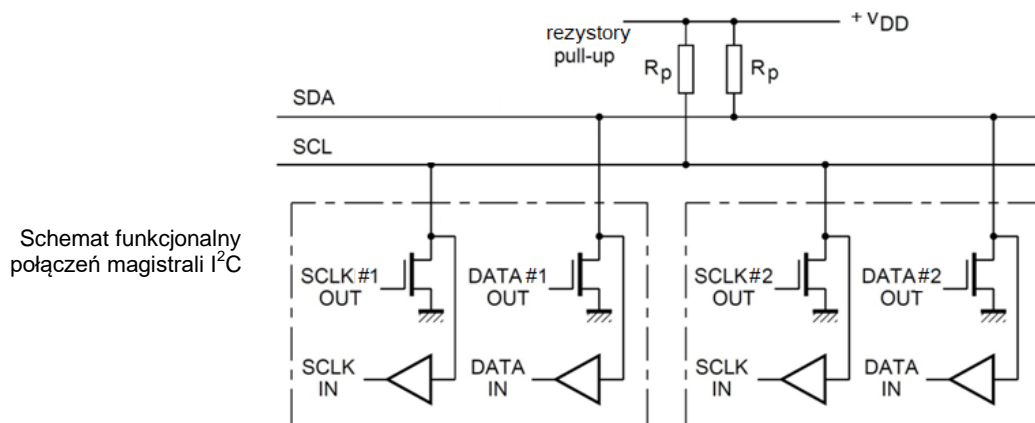
Kanałem przesyłowym magistrali I²C są dwie linie przekazujące bity:

- SDA – dwukierunkowa linia przesyłająca bity danych i bity sterujące pomiędzy wszystkimi urządzeniami dołączonymi do magistrali (Odbiorniki/Nadajniki). Transmisja może odbywać się tylko w trybie half-duplex
- SCL – linia zegara. Sygnał zegara jest generowany przez urządzenie Master. Jego zadaniem jest synchronizacja wszystkich bitów przesyłanych linią SDA. W niektórych przypadkach sygnał zegara może być wstrzymany przez urządzenie typu Slave

Obie linie są ustawione w stan wysoki (H - +V_{DD}) gdy dane nie są transmitowane. Taki stan jest dla Master'a informacją, iż może rozpocząć transmisję.



Wszystkie interfejsy I²C mają identyczne funkcjonalnie nadajniki i odbiorniki sygnałów na liniach SDA/SCL. Tranzystory nadające (układ open-dren lub open-kolektor) realizują, dzięki wspólnemu połączeniu drenów, bramki logiczne typu wired-AND¹⁾ odpowiednio na liniach SDA i SCL. Rezystory pull-up są składnikiem sprzętowym magistrali zapewniając stan wysoki liniom (wyjściom bramek wired-AND).

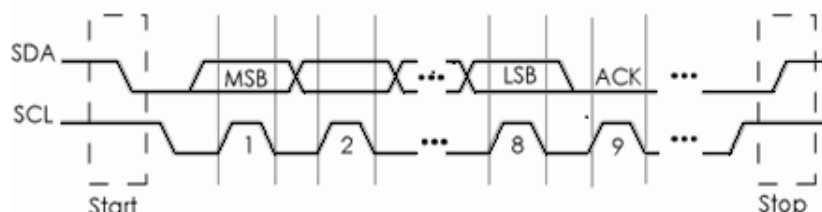


Odbiorniki sygnałów (bufory) gwarantują urządzeniom Master i Slave nie tylko odbiór danych, lecz również porównanie aktualnie wysyłanego bitu z rzeczywistym stanem linii. Ten mechanizm jest wykorzystywany do kontroli przesyłanych danych.

¹⁾ tzw. iloczyn montażowy

Transmisję rozpoczyna zawsze Master sygnałem Start, jednocześnie zaczynając generowanie sygnału zegara. Następnie kontynuuje nadawanie wysyłając pierwszy bajt. Jest on bajtem nagłówkowym zawierającym adres układu Slave oraz bit informujący wybrany Slave o kierunku transmisji danych – od Master do Slave lub od Slave do Master. Po bajcie nagłówkowym przesyłane są bajty danych. Każdy przesłany bajt kończy się bitem potwierdzenia ACK, którym Odbiornik potwierdza Nadajnikowi odebranie bajtu (8-miu bitów). Transmisję kończy zawsze Master sygnałem Stop.

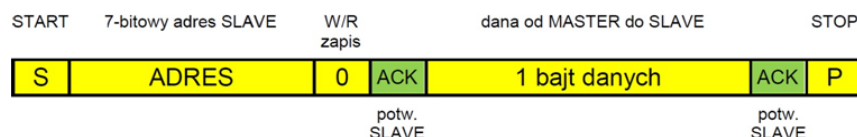
Przebiegi na liniach SDA/SCL w czasie transmisji.



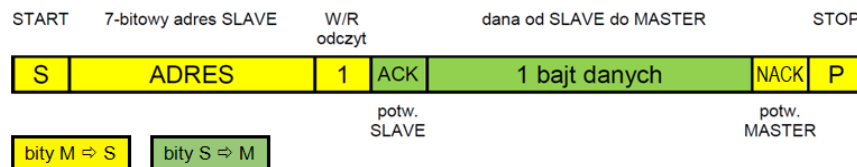
Każdy bit danych jest wysyłany przez Nadajnik i odbierany przez Odbiornik wówczas gdy stan linii zegara SCL jest wysoki (H). Bity danych zmieniają się wyłącznie gdy stan linii SCL jest niski (L).

Kompletne, najkrótsze transmisje (jeden bajt danych) ze wskazaniem bitów przesyłanych pomiędzy urządzeniami Master/Slave pokazano poniżej.

Transmisja
Master ⇒ Slave

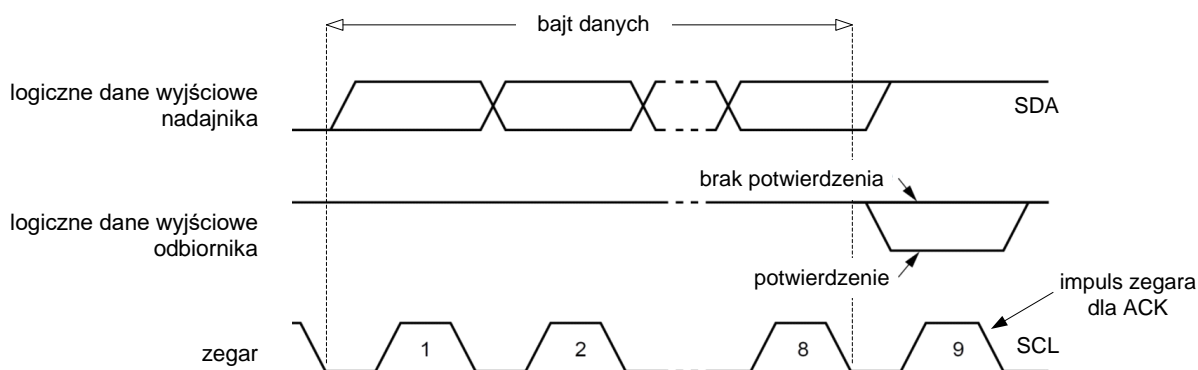


Transmisja
Slave ⇒ Master



Stosunek ilości przesłanych bitów danych do ilości bitów w całej transmisji decyduje o szybkości transferu danych, przy ustalonej częstotliwości BaudRate. Stosunek ten rośnie dla rosnącej długości transferu.

Mechanizm potwierdzania (ACK) odbieranych bajtów korzysta z zastosowanego połączenia wired-AND.



Potwierdzeniem odbioru przez Slave-odbiorcę (ACK) jest ściągnięcie linii SDA do zera, natomiast potwierdzeniem odbioru przez Master-odbiorcę (NACK) jest zwolnienie linii SDA (stan wysoki).

Warunkiem rozpoczęcia transmisji przez Master'a jest brak zajętości magistrali przez inną transmisję. Taki stan jest wskazywany przez stan wysoki na obu liniach przesyłowych SDA/SCL. Magistrala I²C jest magistralą wielo-masterową co oznacza, że transmisję może rozpocząć kilka Masterów, z których każdy jest niezależnym urządzeniem. We wszystkich magistralach wielo-masterowych stosowany jest mechanizm arbitrażu rozstrzygający, który Master w aktualnej chwili czasowej przejmie kontrolę nad transmisją.

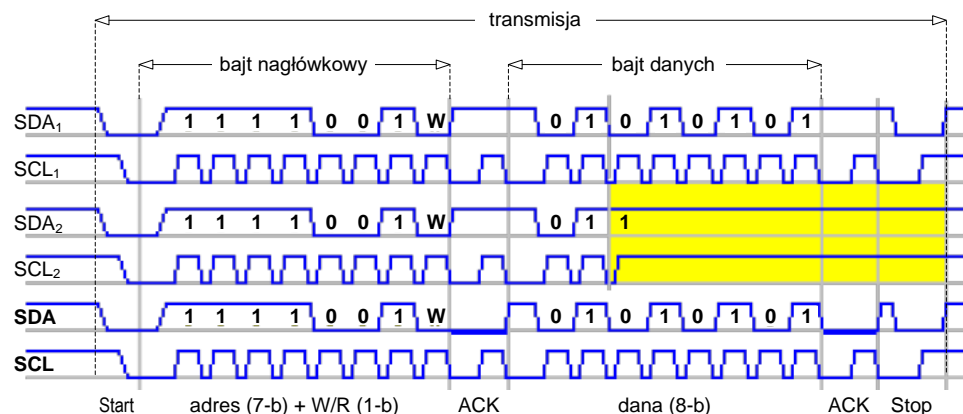
Zasady wbudowanego mechanizmu arbitrażu dla interfejsu I²C:

- dotyczy tylko urządzeń typu Master
- prowadzony jest wyłącznie na linii SDA podczas stanu wysokiego na SCL
- trwa przez cały czas transmisji
- kończy się gdy wysyłana przez układ Master dana logiczna nie jest równa stanowi linii SDA
- przegrywa Master, który wystawia logiczny stan wysoki (H) podczas gdy na linii SDA jest stan niski (L)
- Master przegrywający ustawia na swoim wyjściu danych stan wysoki (H) do końca transmisji

Przykład arbitrażu:

- dwa układy Master rozpoczynają jednocześnie transmisję
- oba nadają do układu Slave o adresie **1111001-b**
- Master₁ nadaje daną **01010101-b**
- Master₂ nadaje daną **011xxxxx-b**
- nierówność w transmisji pojawia się gdy nadawany jest trzeci bit danych (nadawana dana_{Master2} ≠ SDA)
- Master₂ przegrywa arbitraż i nie kończy swojej transmisji

Przykład arbitrażu:
przebiegi logiczne wysyłane przez Master₁ i Master₂ oraz rzeczywisty stan linii SDA/SCL w trakcie arbitrażu



Podstawowymi atutami interfejsu I²C, decydującymi o jego powszechnym stosowaniu, są:

- magistrala I²C jest wielo-masterowa
- tylko dwie linie przesyłowe (mała i tania płytka PCB)
- proste adresowanie urządzeń Slave
- mechanizm kontroli przepływu (ACK/NACK)
- możliwość dołączania i odłączania urządzeń w trakcie ruchu na magistrali
- praca z szybkimi i wolnymi urządzeniami (mechanizm wstrzymywania zegara przez Odbiorniki)

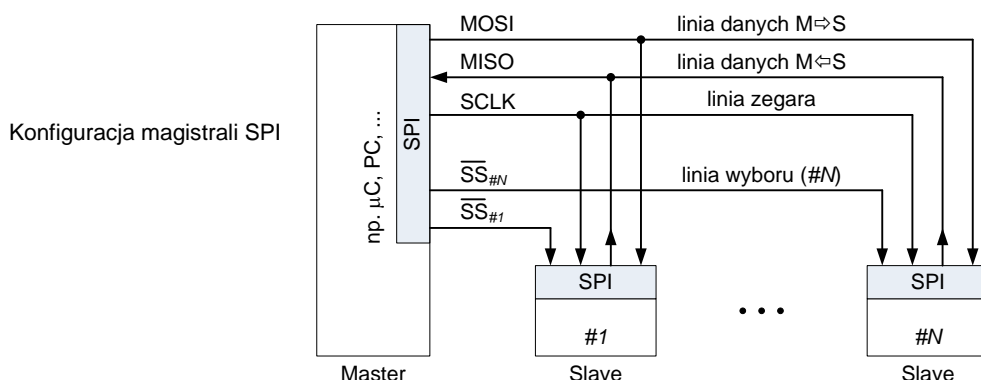
SPI

Synchroniczny interfejs SPI (**S**erial **P**eripheral **I**nterface) został opracowany pod koniec lat 70-ch przez firmę Motorola. Jego poprzednikiem jest interfejs Microwire stworzony w National Semiconductor, stosowany do dzisiaj i będący obecnie jednym z wariantów SPI.

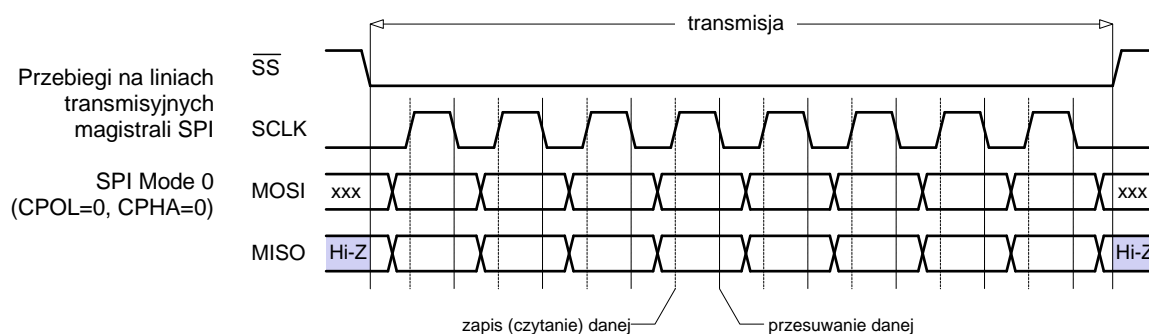
W wersji podstawowej magistralę tworzą cztery linie:

- MOSI - Master Output – Slave Input, linia danych od Master'a do Slave
- MISO - Master Input – Slave Output, linia danych od Slave do Master'a
- SCLK - linia zegara
- /SS - Slave Select, linia wyboru indywidualnego urządzenia Slave

Ilość linii /SS wynika z ilości urządzeń Slave dołączonych do magistrali. Typowo linie /SS tworzone są przez GPIO mikrokontrolera obsługującego interfejs.



Ze względu na istnienie dwu linii danych (MOSI, MISO) możliwy jest transfer full-duplex. Chwile przesyłu danych wyznaczone są zboczami sygnału zegarowego SCLK, które jednocześnie przesuwają zawartość rejestrów przesuwanych urządzenia Master i Slave odpowiednio.



Inicjalizacja interfejsu SPI wymaga ustawienia dwu bitów konfiguracyjnych – CPOL i CPHA. Decydują one o stanie spoczynkowym zegara (L/H) oraz wyborze zbocza wpisującego (czytającego) dane. Drugie zbocze przesuwają transmitowane dane.

Mode	CPOL	CPHA	zapis zbocze
0	0	0	↗
1	0	1	↘
2	1	0	↘
3	1	1	↗

Tabela trybów pracy interfejsu SPI

W zakresie trybu Mode0 interfejs SPI jest w pełni kompatybilny z interfejsem Microwire, popularnym i szeroko stosowanym przez wielu producentów układów Slave. Istnieją jednak różnice w nazewnictwie ze względu na brak istnienia oficjalnej specyfikacji SPI. W tabeli podano odpowiedniki nazw sygnałów SPI i Microwire.

Odpowiedniki nazw
sygnałów SPI/Microwire

SPI	Microwire
SCLK - Serial Shift Clock	SK - Serial Shift Clock
MOSI - Master Out Slave In	SO - Serial Out (Master i Slave)
MISO - Master In Slave Out	SI - Serial In (Master i Slave)
/SS - Slave Select	/CS - Chip Select

Tego rodzaju różnice występują w interfejsach wielu producentów. Ze względu na brak formalnego standardu SPI rozbieżności mogą być głębsze i obejmować cechy funkcjonalne:

- rejestry przesuwne we/wy - 8-b lub 16-b
- inne kolejności przesyłania bitów - MSB⇒LSB lub LSB⇒MSB
- inna definicja trybów pracy
- ...

Problemy kompatybilności można w większości rozwiązać programową obsługą interfejsu lub zabiegami sprzętowymi. W poważnych przypadkach konieczne jest jednak stosowanie procedur emulacyjnych typu bit-banging.

Mimo wymienionych uwag interfejsy SPI są bardzo szeroko stosowane, ze względu na ich zalety:

- komunikacja full-duplex
- duża szybkość
 - nadajniki Push-pull
 - brak bitów sterujących
- dane nie ograniczone do słów 8-b
- dowolny format transmisji
- bardzo prosty, sprzętowy wybór Slave'ów
- linie przesyłowe jednokierunkowe (łatwa izolacja galwaniczna)

Ćwiczenia

Przedstawione zadania należy traktować jako przykładowe, gdyż każde z nich porusza wybrane z wielu, tematy możliwe do realizacji. Aktualna treść zadań będzie podawana przez prowadzących zajęcia w tygodniu poprzedzającym realizację laboratoryjną.

We wszystkich przypadkach zadania będą zawierały do wykonania stałe części programistyczne:

- 1) konfiguracja wybranego interfejsu
- 2) protokół nawiązania połączenia (jeśli jest niezbędny)
- 3) aplikacja wykorzystująca utworzone łącze komunikacyjne

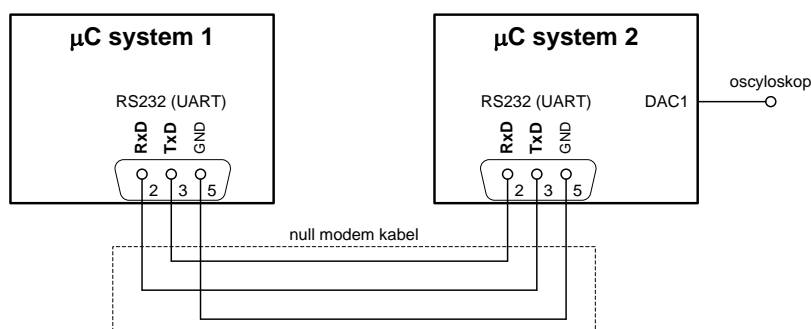
Weryfikacja realizacji zadań obejmuje ocenę sposobu i metod użytych przy komponowaniu programów oraz wyników badań obrazujących ich funkcjonowanie.

UART

Zadanie

Mikrokontroler systemu laboratoryjnego wyposażony jest w interfejs UART. Jego linie danych TxD/RxD są, poprzez układ scalony SP3232 konwertujący poziomy napięcie, doprowadzone do łącza komunikacyjnego. Możliwe jest dzięki temu połączenie dwu systemów laboratoryjnych i przesyłanie danych między nimi.

Połączenie komunikacyjne systemów laboratoryjnych



Celem połączenia jest przesłanie danych, wartości generowanego w czasie rzeczywistym przebiegu piłokształtnego, do drugiego systemu. W systemie odbierającym, dane należy przedstawić w formie analogowej korzystając z wbudowanego przetwornika DAC.

Należy napisać i uruchomić programy:

1. inicjalizacja interfejsu UART (system1 & system2)
2. protokół przesyłu danych (system1 & system2)
3. generator przebiegu piłokształtnego i wysyłanie jego wartości do UART (system1)
4. pobieranie danych z UART i przekazywanie do przetwornika DAC (system2)

Warunki zadania:

- programy 3. i 4. realizowane jako procedury obsługi przerwań
- w programie 3. obie funkcje (tworzenie przebiegu i wysyłanie) nie muszą być zsynchronizowane
- procedura przerwania nie może odwoływać się do programu głównego, ani korzystać z jego zasobów
- nie można używać opóźnień programowych (pętle)

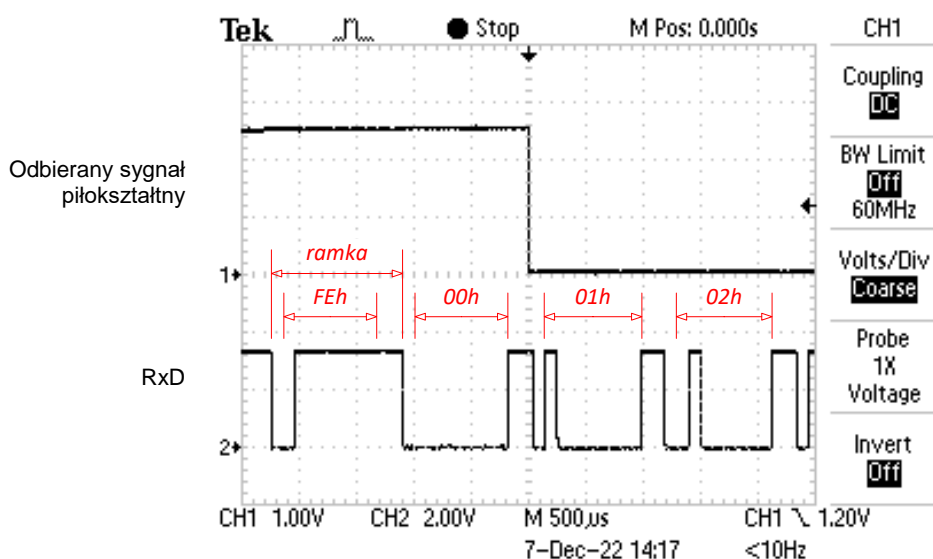
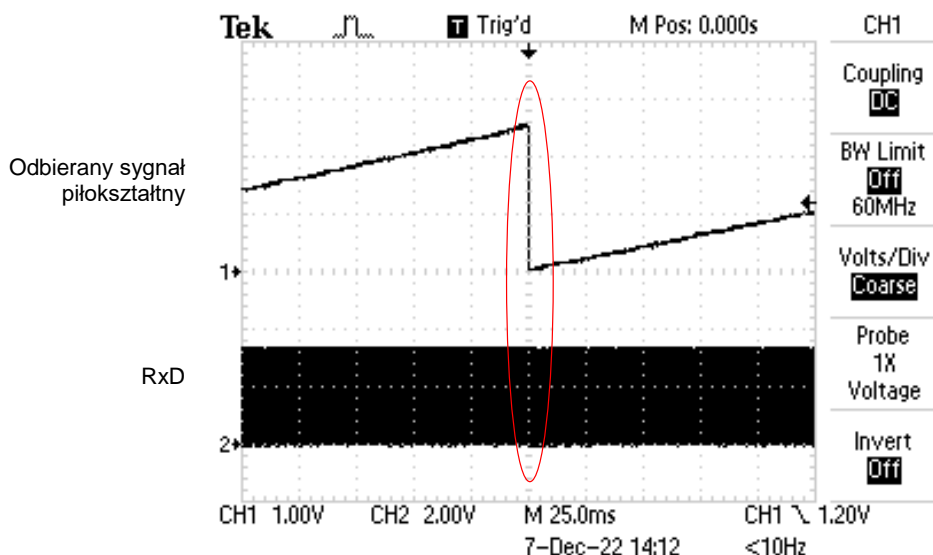
Uwagi:

- w części inicjalizacyjnej programu należy skonfigurować moduł komunikacyjny USART mikrokontrolera ustawiając tryb pracy UART i parametry interfejsu w rejestrach konfiguracyjnych. Szczegółowy opis interfejsu znajduje się w dokumencie slau049f.pdf (*MSP430x1xx Family User's guide*) dostępnym m.in. na stronie materiałów przedmiotu
- schemat połączenia układu interfejsu UART i jego otoczenia pokazano w rozdziale „blok 8: *Interfejsy asynchroniczne*”
- sygnał RxD doprowadzany jest do interfejsu UART (linia P3.7 μ C) z konwertera poziomu napięć SP3232 poprzez multiplexer 74HC253 („blok 0: *Schemat złożeniowy*”). Prawidłowe połączenie jest wówczas, gdy linia P4.7 sterująca multiplexerem jest ustawiona w stan niski, a linia P3.0 w stan wysoki
- karty katalogowe konwertera poziomu napięć SP3232 i multiplexera 74HC253 są dostępne m.in. na stronie materiałów przedmiotu

Przykładowe wyniki pomiarowe

Poniżej przedstawiono oscylogramy przebiegu analogowego i przebiegów linii odbierającej RxD z zaznaczonymi wartościami przesyłanych danych. Tworzenie próbek sygnału piłokształtnego zsynchronizowane z ich przesyłaniem do układu odbierającego. Format nadawanej ramki: 1-b Start, 8-b dana, 2-b Stop

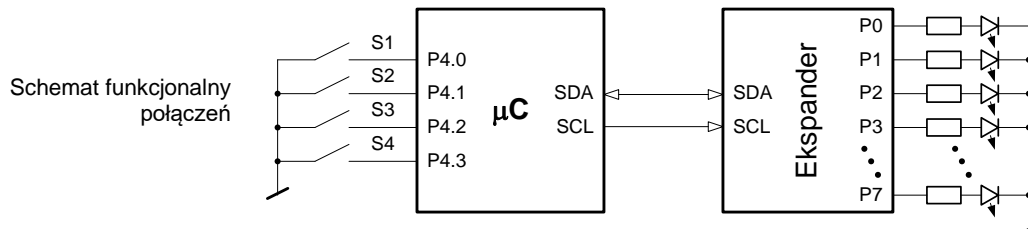
Pomiaru dokonano w punktach testowych TP5 (DAC1) i TP22 (RxD) korzystając z opisu zawartego w rozdziale „Schemat systemu mikroprocesorowego - *Punkty testowe*”



I²C

Zadanie

Na płycie edukacyjnej umieszczony został układ scalony PCF8574 (ekspander portów), który jest połączony do mikrokontrolera za pośrednictwem interfejsu I²C. Funkcją ekspandera jest zmiana postaci (szeregowa ↔ równoległa) przesyłanych danych. Połączenie szeregowe to interfejs I²C, a równoległe to 8-b port quasi dwukierunkowy. Do każdej linii portu dołączona jest dioda LED sygnalizująca jej stan logiczny. Łącznie diody tworzą 8. pozycyjną linijkę świetlną.



Należy napisać i uruchomić programy:

1. inicjalizujący interfejs I²C
2. obrazujący stan przycisków S1-S4 (port P4 mikrokontrolera) na linijce diod LED portu równoległego ekspandera, z użyciem interfejsu I²C

Warunki zadania:

- stan przycisków S1-S4 wyświetlany odpowiednio na liniach P0-P3 portu ekspandera
- program zawierający odczyt stanu przycisków, obsługę transmisji I²C, etc. musi być procedurą obsługi przerwania
 - od linii portu P1.3 (wired-AND stanu przycisków S1-S4)
 - lub
 - Timer'a (np. Timer_A)
- procedura przerwania nie może odwoływać się do programu głównego, ani korzystać z jego zasobów
- nie można używać opóźnień programowych (pętle)
- nie można ingerować w zegar systemowy

Uwagi:

- w części inicjalizacyjnej programu należy skonfigurować moduł komunikacyjny USART mikrokontrolera ustawiając tryb pracy I²C i parametry interfejsu w rejestrach konfiguracyjnych. Szczegółowy opis interfejsu znajduje się w dokumencie slau049f.pdf (*MSP430x1xx Family User's guide*) dostępnym m.in. na stronie materiałów przedmiotu.
- schemat połączenia układu ekspandera i jego otoczenia pokazano w rozdziale „Schemat systemu mikroprocesorowego - blok 3: Ekspander I²C”
- schemat połączenia przycisków S1-S4 do portu P4 mikrokontrolera pokazano w rozdziale „Schemat systemu mikroprocesorowego – blok 7: Wejścia dwustanowe”
- karta katalogowa ekspandera PCF8574 jest dostępna m.in. na stronie materiałów przedmiotu

Przykładowe wyniki
pomiarowe

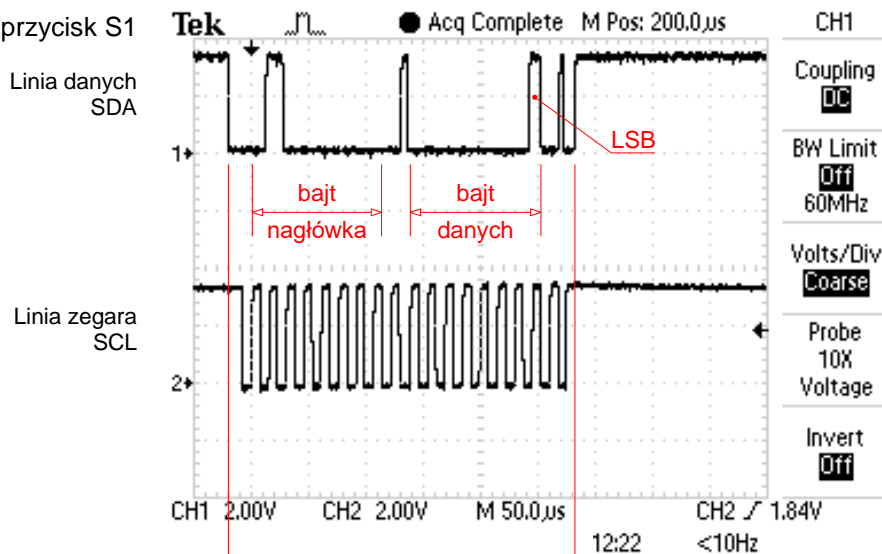
Oscylogramy przedstawiają przykładowe przebiegi na linii danych i linii zegara interfejsu I²C w czasie transmisji od mikrokontrolera do ekspandera.

Oscylogram górny pokazuje, że w obszarze bajtu danych przesyłana jest wartość 00000001b. Stan H najmłodszego bitu (LSB) odpowiada stanowi naciśnięcia przycisku S1 dołączonego do linii P4.0. Pozostałe trzy przyciski, dołączone do P4.1; P4.2; P4.3 odpowiednio, nie są naciśnięte. Cztery najstarsze bity przesyłanego bajtu są programowo kasowane przed transferem.

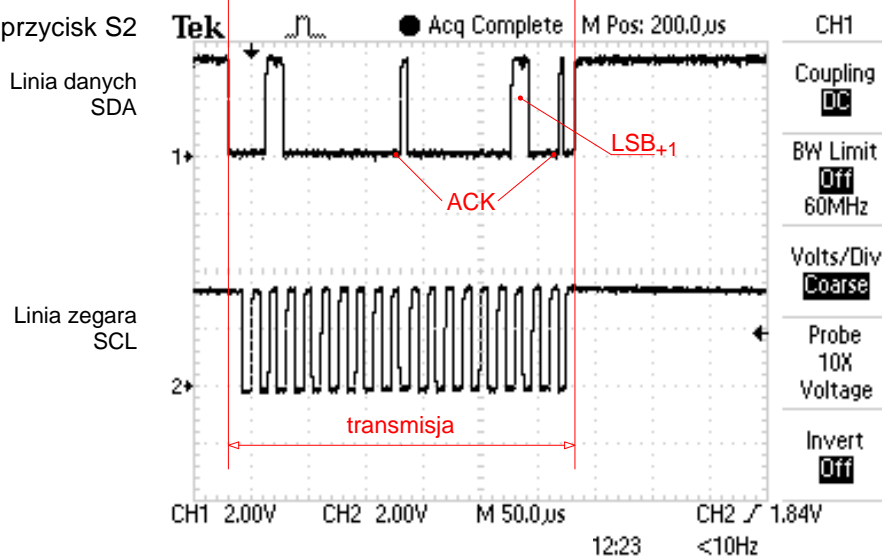
Oscylogram dolny obrazuje przesyłaną liczbę 00000010b, w obszarze bajtu danych. Stan H bitu LSB₊₁ odpowiada stanowi naciśnięcia przycisku S2 dołączonego do linii P4.1. Pozostałe przyciski nie są naciśnięte.

Pomiaru dokonano w punktach testowych TP23 (SCL) i TP29 (SDA) korzystając z opisu zawartego w rozdziale „Schemat systemu mikroprocesorowego - Punkty testowe”

Aktywny przycisk S1



Aktywny przycisk S2



Bit-banging (I^2C)**Zadanie****Należy** napisać i uruchomić programy:

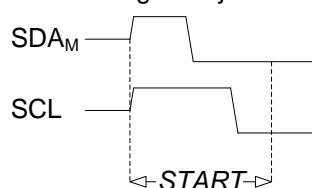
1. program bit-banging emulujący interfejs I^2C , zgodnie z niżej zamieszczonym opisem
2. program obrazujący stan przycisków S1-S4 (port P4 mikrokontrolera) na linijce diod LED portu równoległego ekspandera, z użyciem napisanej procedury bit-banging

Warunki zadania i pomiarów jak w zadaniu „ I^2C ”**Realizacja**programu
Bit-banging

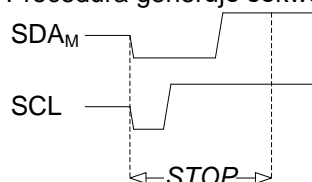
Podstawowym zadaniem programów typu bit-banging jest emulacja interfejsów sprzętowych. Stosowane są wówczas gdy mikrokontroler nie jest wyposażony w potrzebny interfejs, a rozwiązania sprzętowe nie są akceptowalne. Takie sytuacje zachodzą często w trakcie użytkowania systemu mikroprocesorowego. Oprogramowanie bit-banging musi emulować wszystkie funkcjonalności interfejsu, łącznie z następstwem czasowym względem generowanych/odbieranych przebiegów na wszystkich liniach transmisyjnych. Można stworzyć program bit-banging emulujący dowolny interfejs. Wadą oprogramowania może być relatywnie, w porównaniu z interfejsem sprzętowym, mała szybkość transferu. Z tego powodu bit-banging tworzone są głównie w językach niskopoziomowych (assemblerach).

Poniżej przedstawiono, jako przykładowy, szablon do realizacji programu bit-banging emulującego interfejs I^2C . Stałe fragmenty transmisji wyodrębniono w postaci procedur. Mogą one, wołane w odpowiedniej sekwencji, realizować dowolną transmisję. Rysunki obrazują następstwo przebiegów logicznych generowanych przez procedury. Dla dwu procedur, *START* oraz *TR_MS*, podano przykład realizacji w assemblerze MSP430. Przyjęto oznaczenia:

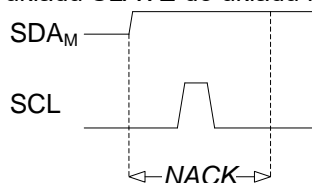
- DANA - arbitralnie wybrany rejestr z zakresu R4-R15
- SDA - linia P3.1 (wy/we – P3OUT/P3IN)
- SCL - linia P3.3 (wy – P3OUT)
- FLAG - arbitralnie wybrana flaga przerwania maskowalnego

Procedura *START* Procedura generuje sekwencję *START*.

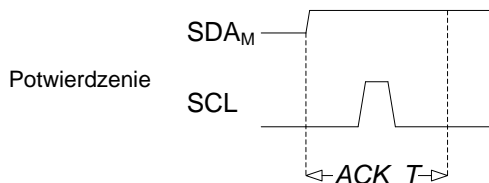
```
START: bis.b #00001010b, &P3DIR ; ustawienie P3.1&P3.3 jako wyjścia
      bis.b #00000010b, &P3OUT ; ustawienie SDA (P3.1) na stan H
      bis.b #00001000b, &P3OUT ; ustawienie SCL (P3.3) na stan H
      bic.b #00000010b, &P3OUT ; ustawienie SDA (P3.1) na stan L
      bic.b #00001000b, &P3OUT ; ustawienie SCL (P3.3) na stan L
      RET
```

Procedura *STOP* Procedura generuje sekwencję *STOP*.

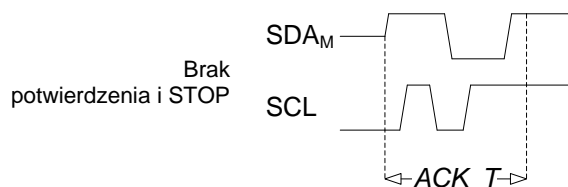
Procedura *NACK* Procedura potwierdzenia ze strony układu MASTER po transmisji danej od układu SLAVE do układu MASTER.



Procedura *ACK_T* Procedura kontroli potwierdzenia ze strony układu SLAVE po transmisji danej od układu MASTER do układu SLAVE.

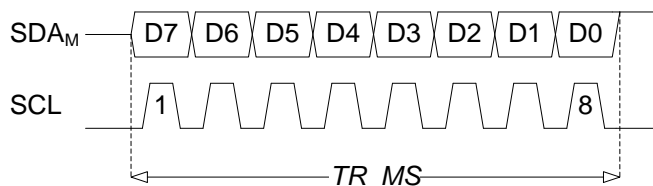


Procedura sprawdza potwierdzenie układu SLAVE. Jeżeli SLAVE nie potwierdza transmisji to procedura powinna wygenerować sekwencję STOP i ustawić flagę przerwania FLAG jako ostatnie działanie.



Komentarz: w przypadku wykrycia braku potwierdzenia zostanie ustawiona flaga przerwania FLAG. Jeśli przerwanie nie jest uaktywnione (założona maska) to ciąg dalszy programu transmisji zostanie wykonany. W przeciwnym, oczekiwanym, wypadku realizacja dalszych działań będzie zależała od zawartości procedury obsługi tego przerwania, w której należy uwzględnić fakt jej wywołania przed zakończeniem procedury *ACK_T*.

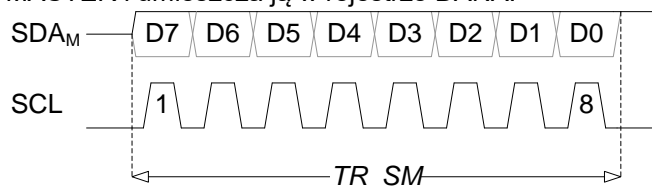
Procedura *TR_MS* Procedura dokonuje transmisji 8-mio bitowej danej zawartej w rejestrze DANA z układu MASTER do układu SLAVE. Daną może być bajt nagłówkowy - 7-mio bitowy adres + bit kierunku transmisji R/W (LSB).



```

TR_MS: bic.b #00000010b, &P3DIR ; ustawienie P3.1 jako wyjście
      bic.b #00001000b, &P3OUT ; ustawienie SCL (P3.3) na stan L
      ; przesłanie bitu #1 do Odbiornika
      rlc.b DANA ; bit #1 (MSB) do C
      jc jeden ; jeśli C=1 to jeden
      bic.b #00000010b, &P3OUT ; ustawienie SDA (P3.1) na stan L
      jmp dalej ;
jeden: bis.b #00000010b, &P3OUT ; ustawienie SDA (P3.1) na stan H
dalej: bis.b #00001000b, &P3OUT ; ustawienie SCL (P3.3) na stan H
      bic.b #00001000b, &P3OUT ; ustawienie SCL (P3.3) na stan L
      ; przesłanie bitu #2 do Odbiornika
      ...
      ...
      ; przesłanie bitu #8 do Odbiornika
      ...
      RET
  
```

Procedura *TR_SM* Procedura dokonuje transmisji 8-mio bitowej danej z układu SLAVE do układu MASTER i umieszcza ją w rejestrze DANA.



Odczyt bitów z linii SDA (P3IN) musi odbywać się w trakcie stanu wysokiego SCL. Bity powinny być wprowadzane do DANA na pozycję LSB i przesuwane w lewo.

Komponowanie programu transmisji

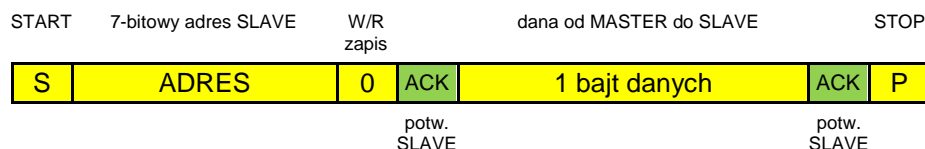
Przedstawione procedury pozwalają komponować programy obsługujące dowolnie przyjęte, w ramach standardu, transmisje. W podanych przykładach oznaczono kolorami kierunek przesyłanych informacji:

S od układu MASTER do SLAVE

S od układu SLAVE do MASTER

Przykład I:
transmisja M⇒S

Poniżej przedstawiono schemat podstawowej transmisji nadawczej, przekazującej 1 bajt danych z układu MASTER do SLAVE.

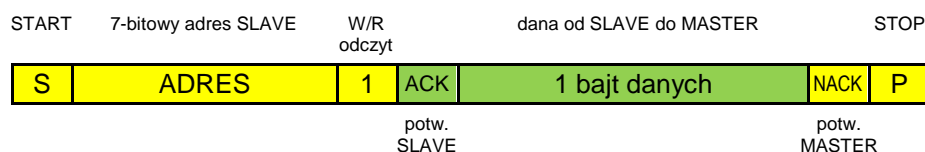


Realizacja transmisji przy użyciu omówionych procedur:

```
call    #START    ; generowanie sekwencji START
mov.b   XXX,DANA1 ;
call    #TR_MS    ; przekazanie adresu i bitu W/R
call    #ACK_T    ; sprawdzenie potwierdzenia układu SLAVE
mov.b   YYY,DANA2 ;
call    #TR_MS    ; przekazanie jednego bajtu danych do SLAVE
call    #ACK_T    ; sprawdzenie potwierdzenia układu SLAVE
call    #STOP     ; generowanie sekwencji STOP
```

Przykład II:
transmisja M⇐S

Poniżej przedstawiono schemat podstawowej transmisji odbiorczej, przekazującej 1 bajt danych z układu SLAVE do MASTER.



Realizacja transmisji przy użyciu omówionych procedur:

```
call    #START    ; generowanie sekwencji START
mov.b   XXX,DANA1 ;
call    #TR_MS    ; przekazanie adresu i bitu W/R
call    #ACK_T    ; sprawdzenie potwierdzenia układu SLAVE
call    #TR_SM    ; odebranie jednego bajtu danych od SLAVE
call    #NACK     ; potwierdzenie przez MASTER obioru bajtu
call    #STOP     ; generowanie sekwencji STOP
```

Po wykonaniu tego programu rejestr DANA będzie zawierał bajt danych, odebrany od układu SLAVE.

¹ - XXX jest rejestrem lub 8-bitową liczbą zawierającą na pozycji starszych bitów adres układu SLAVE, a na pozycji LSB „0” co oznacza kierunek przekazywania do SLAVE.

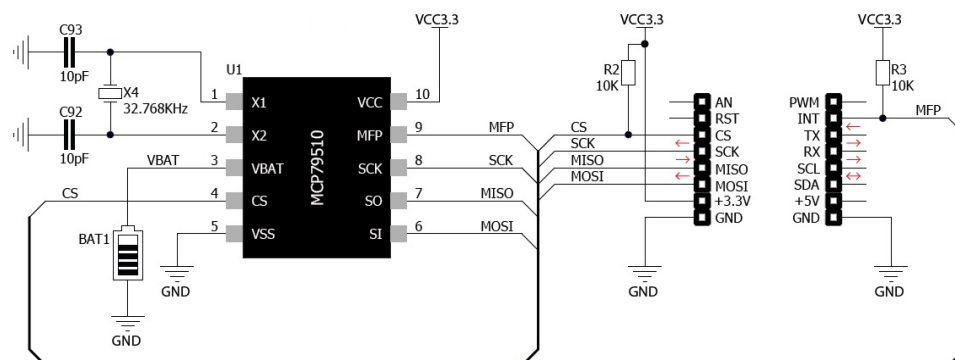
² - YYY jest rejestrem lub 8-bitową liczbą będącą daną przekazywaną do układu SLAVE

SPI

Zadanie

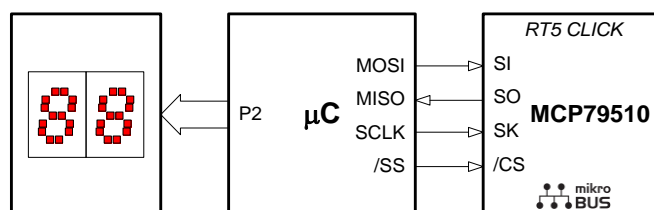
Na płycie edukacyjnej umieszczone są gniazda rozszerzeń zgodne ze standardem połączeń *mikroBUS*. Do jednego z gniazd (U\$1) dołączona jest płytki rozszerzeń zawierająca układ scalony MCP79510 (RTC/C – zegar czasu rzeczywistego/kalendarz). Układ, poprzez gniazdo *mikroBUS*, jest połączony z mikrokontrolerem za pośrednictwem interfejsu SPI.

Schemat ideowy połączeń układu RTC/C



Mikrokontroler ustawia parametry RTC/C oraz pobiera z niego dane. Pobrane dane mogą być obrazowane na wyświetlaczach będących składnikiem płyty edukacyjnej.

Schemat blokowy przepływu sygnałów



Należy napisać i uruchomić programy:

1. inicjujący interfejs SPI (Mode 0)
 2. a) pobierający cyklicznie z układu RTC (rejestr RTCSEC) aktualny stan dziesiątek i jednostki sekund
 - b) wyświetlający pobrane dane na wyświetlaczach 7-mio segmentowych dołączonych do portu P2
- Uwaga: należy dokonać przekształcenia pobieranej z RTC danej na kod BCD obsługiwany przez wyświetlacze

Warunki zadania:

- program obsługi transmisji SPI, wyświetlania etc. powinien być procedurą obsługi przerwania od Timer_A/Timer_B
- procedura przerwania nie może odwoływać się do programu głównego, ani korzystać z jego zasobów
- nie można używać opóźnień programowych (pętle)

Uwagi:

- w części inicjalizacyjnej programu należy skonfigurować moduł komunikacyjny USART mikrokontrolera ustawiając tryb pracy SPI i parametry interfejsu w rejestrach konfiguracyjnych. Szczegółowy opis interfejsu znajduje się w dokumencie slau049f.pdf (*MSP430x1xx Family User's guide*) dostępnym m.in. na stronie materiałów przedmiotu.
- schemat połączenia wyświetlaczy 7-mio segmentowych z portem P2 mikrokontrolera pokazano w rozdziale „Schemat systemu mikroprocesorowego – blok 5: Rozszerzenia linii GPIO”

- sygnał MISO doprowadzany jest do interfejsu SPI (linia P5.2 μ C) z gniazda *mikroBUS* (U\$1) poprzez multiplexer 74HC253 („blok 0: Schemat złozeniowy”). Prawidłowe połączenie jest wówczas, gdy linia P4.7 sterująca multiplexerem jest ustawiona w stan niski, a linia P3.0 w stan wysoki
- karty katalogowe układu MCP79510 (zegar RTC/C) i multiplexera 74HC253 są dostępne m.in. na stronie materiałów przedmiotu

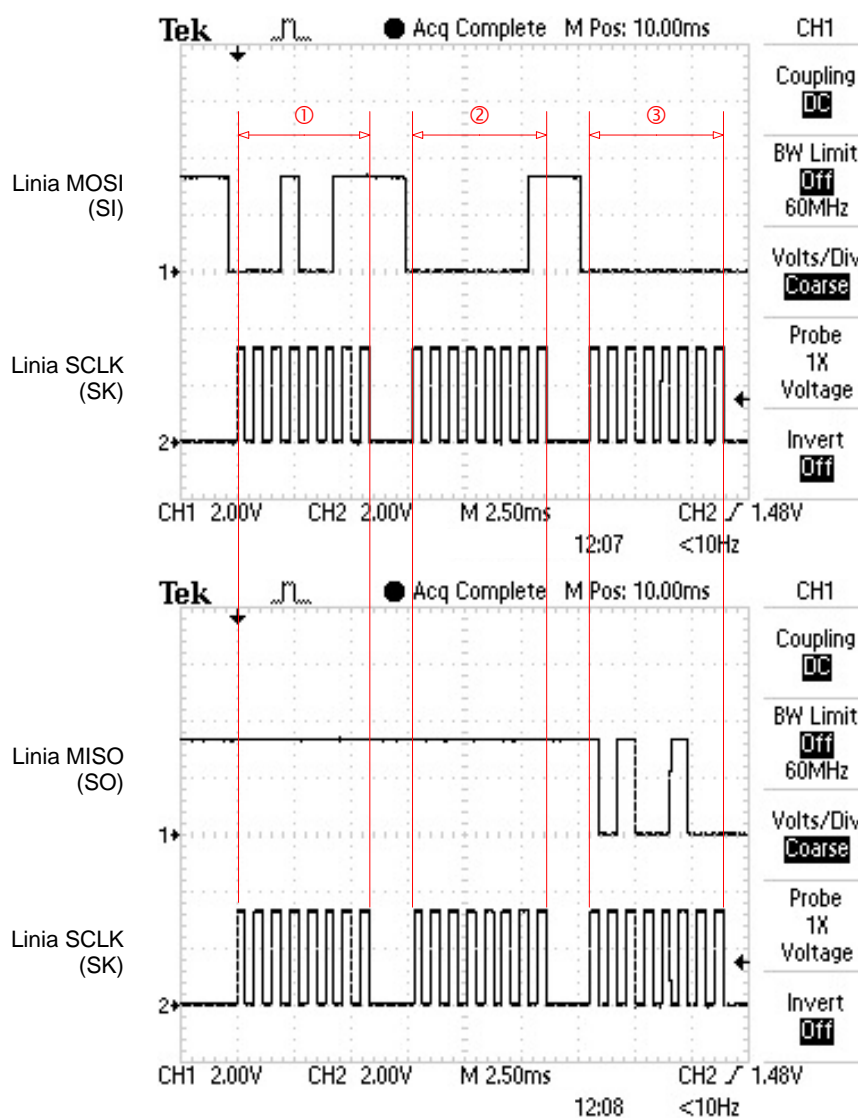
Przykładowe wyniki pomiarowe

Na oscylogramach pokazano pakiety przekazywanych danych:

- ① kierunek do RTC/C; wartość 00010011b; polecenie odczytu rejestru wskazanego adresem
- ② kierunek do RTC/C; wartość 00000001h; adres odczytywanego rejestru RTCSEC
- ③ kierunek do μ C; wartość 10100100b; zawartość rejestru RTCSEC

Bajt zawarty w rejestrze RTCSEC składa się z bitu sterującego „1” na pozycji MSB oraz liczby (0100100b) aktualnie odliczonych sekund, przedstawionej w upakowanym kodzie BCD. Odebrana liczba 24 jest przekazana do wyświetlaczy.

Pomiaru dokonano w punktach testowych TP27 (MISO-P5.1), TP30 (MOSI-P5.2) i TP28 (SCLK-P5.3), korzystając z opisu zawartego w rozdziale „Schemat systemu mikroprocesorowego - Punkty testowe”



Pytania

Osnową wszystkich zadań jest napisanie odpowiedniego programu i jego uruchomienie. Pytania i dyskusja w trakcie realizacji ćwiczenia, będą związane ze specyficznymi, zastosowanymi przez studenta rozwiązaniami programistycznymi (np. metoda konwersji kodów, sposób ochrony danych przy przerwaniach, organizacja obsługi przerwań, etc.).

Drugą grupą, niespecyficzną dla rozwiązań programistycznych, są pytania odnoszące się do wiedzy o interfejsach. Przykłady takich pytań przedstawiono poniżej.

1. Dwa urządzenia połączono interfejsami UART z tym samym BR. Nadajnik pracuje w trybie 8-b; 2-b Stop, natomiast odbiornik w trybie 8-b; 1-b Stop. Czy jest możliwe, aby transmisja przebiegała poprawnie? Uzasadnić.
2. UART nadaje ramkę (tryb 8-b, 1-b Stop) mając ustawiony BaudRate=9600Hz. Jaką wartość musi mieć BaudRate, aby nowa ramka (tryb 9-b, 2-b Stop) trwała tyle samo co poprzednia?
3. W interfejsie I²C:
 - a) Master wybiera urządzenia Slave poprzez ich adresy sprzętowe
 - b) Master wybiera urządzenia Slave poprzez dodatkową linię Slave Select
 - c) sygnał zegarowy jest generowany przez urządzenie Master lub Slave w zależności od kierunku transmisji
 - d) Master potwierdza nadanie 8 bitowej ramki poprzez wymuszenie stanu niskiego na linii danych
4. Dwa interfejsy, I²C oraz UART (tryb 8-b, 1-b Stop), mają przekazać do współpracujących z nimi urządzeń jeden 8-bitowy bajt danych, w pojedynczej i kompletnej transmisji. Jaki musi być stosunek BaudRate'ów (BR_{I2C}/BR_{UART}), aby obie transmisje trwały tyle samo?
5. Czy istnieje możliwość wykorzystania interfejsu SPI do bezpośredniego przekazania danych do UART? Uzasadnić.
6. SPI - w jaki sposób Master kończy transmisję (sygnalizuje jej koniec)?
7. Jakie ograniczenia określają maksymalną liczbę urządzeń, które można podłączyć do magistrali I²C?
8. Interfejs UART pracuje w trybie 8-b, 1-b Stop. Jaką liczbę musi przesłać, aby przy ciągłym nadawaniu ramek, na TxD powstał przebieg prostokątny o wypełnieniu 50% i częstotliwości $BR/10$?
9. Czy w magistrali I²C układ Odbiornik może wstrzymać transmisję?
10. Należy narysować przebieg sygnału TxD dla interfejsu UART (tryb 8-b; 1-b Stop), przekazującego bajt o wartości XXh^{*)}

^{*)} wartość podana przez prowadzącego

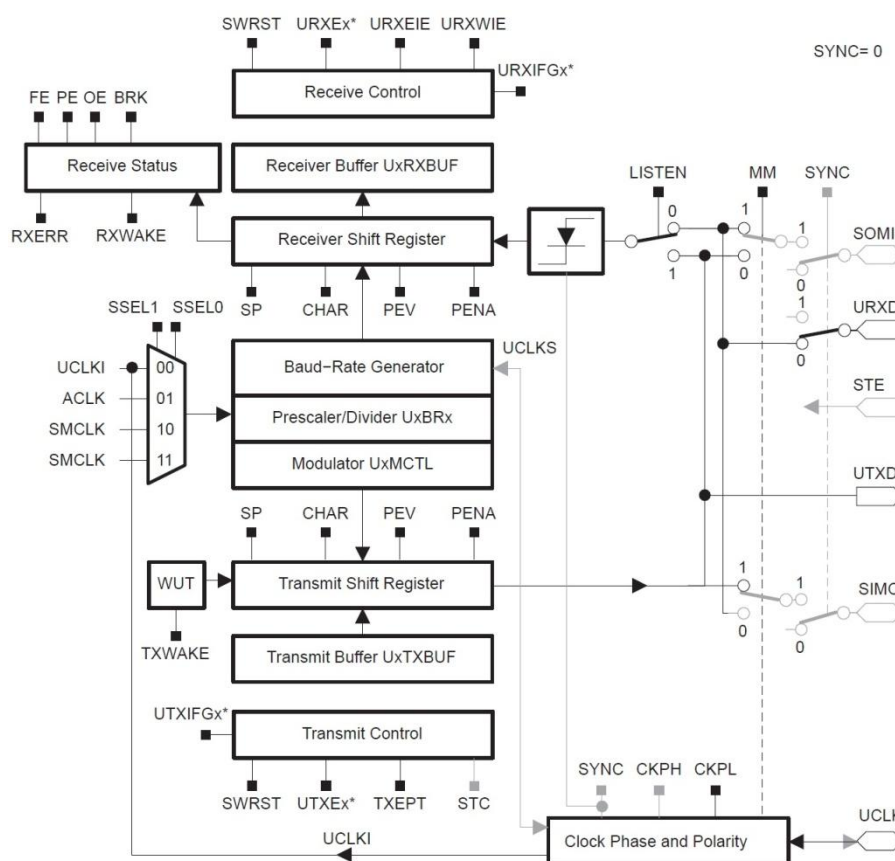
Konfiguracja interfejsów komunikacyjnych (MSP430F169)

W większości współczesnych mikrokontrolerów funkcje poszczególnych interfejsów realizowane są przez jeden, sprzętowy moduł komunikacyjny. Takie rozwiązanie stosowane jest również w procesorach serii MSP430 f-my Texas Instruments. Uniwersalny synchroniczny/asynchroniczny interfejs peryferyjny odbierający i nadający dane (USART) obsługuje oba rodzaje transmisji szeregowej. O wyborze rodzaju transmisji i jej cechach charakterystycznych decyduje przeprowadzona przez użytkownika konfiguracja.

W procesorze laboratoryjnym zaimplementowane są dwa moduły USART (USART0 i USART1). USART0 może być skonfigurowany do pracy w trybie UART lub SPI lub I²C, natomiast USART1 tylko do trybu UART lub SPI. Działanie i proces konfiguracji modułów są identyczne.

tryb UART

Schemat blokowy modułu USART⁽¹⁾ pracującego w trybie **UART**



Uwaga
Wymagany proces inicjalizacji lub ponownej konfiguracji USART do trybu **UART**

Wymagany proces inicjalizacji lub ponownej konfiguracji USART do trybu **UART**:

- 1) Ustaw bit SWRST (BIS.B #SWRST,&UxCTL) – *wprowadzenie modułu USART w stan początkowy RESET*
- 2) Zainicjuj wszystkie niezbędne rejestry USART zgodnie z oczekiwaną funkcjonalnością trybu UART
- 3) Włącz tryb UART (USART) za pomocą bitu MEx (USPIEx)
- 4) Wyczyść bit SWRST (BIC.B #SWRST,&UxCTL) – wyjście modułu USART ze stanu RESET i zezwolenie na działanie
- 5) Włącz przerwania (opcjonalnie) za pomocą IEx (URXIEx i/lub UTXIEx)

Nieprzestrzeganie tego procesu może spowodować nieprzewidywalne zachowanie USART

Przykładowy program
inicjalizacji modułu USART1
do trybu **UART**

W przedstawionym przykładzie inicjalizacji przyjęto arbitralnie wybór modułu USART1 oraz wszystkie parametry interfejsu UART.

```

;..... USART1, UART mode initialisation .....
bis.b  #SWRST,U1CTL      ;USART1 is held in reset state
bis.b  #UTXE1+URXE1,&ME2 ;zezwolenie na transmisję i odbiór
                                ;USART1 (TXD/RXD)
bis.b  #CHAR,&U1CTL      ;dane 8-bi
mov.b  #SSEL1,&U1TCTL    ;wybór źródła zegara (SMCLK)
mov.b  #041h,&U1BR0      ;Ustawienie Baud Rate 9600 Hz
mov.b  #003h,&U1BR1      ;dla oscylatora 8MHz
mov.b  #000h,&U1MCTL     ;brak modulacji Baud Rate
bic.b  #SWRST,&U1CTL     ;USART1 released for operation
bis.b  #URXIE1,&IE2      ;zgoda na przerwania odbiornika
                                ;USART1
; .....

```

Przykładowy program
obsługi interfejsu **UART**

Poniżej pokazano przykładowy program używający interfejsu UART do ciągłego przekazywania danych (stanu portu P4) do drugiego urządzenia UART (odbiornika).

Założenia:

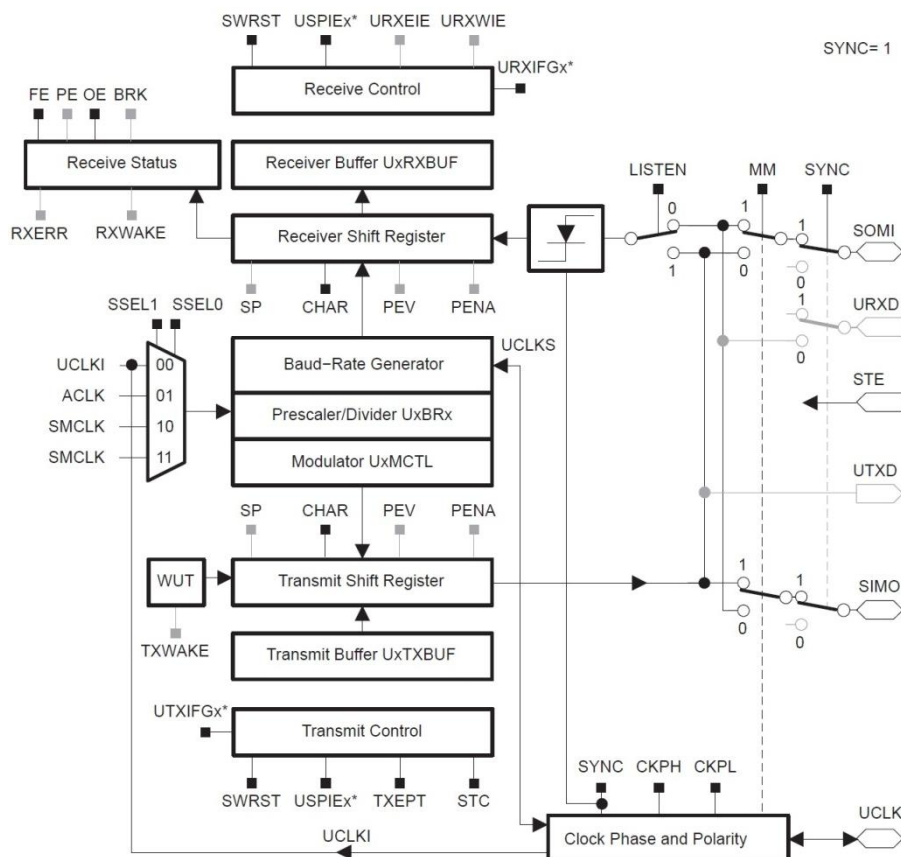
- interfejs UART skonfigurowany zgodnie z powyżej pokazaną procedurą
- sprawdzanie, metodą odpytywania, gotowości rejestru nadającego (TX) do przesłania następnej danej

```

;..... P4 ⇒ TX (USART1) .....
TX3  bit.b  #UTXIFG1,&IFG2 ;czy bufor USART1 TX jest gotowy?
      jz     TX3            ;jeśli nie to sprawdzaj nadal
      mov.b  &P4IN,&TXBUF1  ;wysłanie stanu portu P4
      jmp    TX3            ;pętla transmisji
; .....

```

tryb SPI

Schemat blokowy modułu
USART⁽¹⁾ pracującego w
trybie SPI**Uwaga**

Wymagany proces
inicjalizacji lub ponownej
konfiguracji USART
do trybu SPI

Wymagany proces inicjalizacji lub ponownej konfiguracji USART do trybu SPI:

- 1) Ustaw bit SWRST (BIS.B #SWRST,&UxCTL) – wprowadzenie modułu USART w stan początkowy RESET
- 2) Zainicjuj wszystkie niezbędne rejestry USART zgodnie z oczekiwaną funkcjonalnością trybu SPI
- 3) Włącz tryb SPI (USART) za pomocą bitu MEx (USPIEx)
- 4) Wyczyść bit SWRST (BIC.B #SWRST,&UxCTL) – wyjście modułu USART ze stanu RESET i zezwolenie na działanie
- 5) Włącz przerwania (opcjonalnie) za pomocą IEx (URXIEx i/lub UTXIEx)

Nieprzestrzeganie tego procesu może spowodować nieprzewidywalne zachowanie USART.

Przykładowy program
inicjalizacji modułu USART0
do trybu **SPI**

W przedstawionym przykładzie inicjalizacji przyjęto arbitralnie wybór modułu USART0 oraz wszystkie parametry interfejsu SPI.

```

;..... USART0, SPI mode initialisation .....
bis.b  #SWRST,U0CTL    ;USART0 is held in reset state
bic.b  #I2C,U0CTL      ;SPI mode enable
bis.b  #CHAR,U0CTL     ;8-bit data
bic.b  #LISTEN,U0CTL   ;Listen disabled
bis.b  #SYNC,U0CTL     ;Synchronous SPI mode
bis.b  #MM,U0CTL       ;Master mode
bis.b  #CKPH,U0TCTL    ;SPI mode 2 (for LTC1453)
bic.b  #CKPL,U0TCTL    ;
bic.b  #SSEL0,U0TCTL   ;SMCLK is source clock
bis.b  #SSEL1,U0TCTL   ;
bis.b  #STC,U0TCTL     ;STE disabled
bis.b  #USPIE0,ME1     ;module USART0 SPI enabled
mov.b  #02h,U0BR0      ;BaudRate=SMCLK/2
mov.b  #00h,U0BR1      ;
mov.b  #00h,U0MCTL     ;
bic.b  #SWRST,U0CTL    ;USART0 released for operation
bic.b  #UTXIE0,IE1     ;USART0 transmit interrupt disabled
bic.b  #URXIE0,IE1     ;USART0 receive interrupt disabled
;.....

```

Przykładowy program
obsługi interfejsu **SPI**

Poniżej pokazano przykładowy program używający interfejsu SPI do transferu danych.

Założenia:

- interfejs SPI skonfigurowany zgodnie z powyżej pokazaną procedurą
- przekazywanie danej 16-b zawartej w rejestrze R15 (HighB, LowB)
- sygnał /CS (/SS) przekazywany linią P3.0 portu P3
- układ SLAVE wymaga przyjmowania 16-b danej w dwu oddzielnych bajtach
- transmisja half-duplex MASTER ⇒ SLAVE

Przykładem układu SLAVE wymagającego spełnienia wyżej wymienionych założeń jest 12-bitowy przetwornik DAC typu LTC1453.

```

;..... R15 ⇒ SPI .....
swpb R15 ;
bic.b #01h,&P3OUT ;/CS=L _____ początek przekazywania danej
mov.b R15,U0TXBUF ;HighB to SPI
swpb R15 ;
A_00 bit.b #TXEPT,U0TCTL ;
jz A_00 ;
mov.b R15,U0TXBUF ;LowB to SPI
A_01 bit.b #TXEPT,U0TCTL ;
jz A_01 ;
bis.b #01h,&P3OUT ;/CS=H _____ koniec przekazywania danej
;.....

```


Przykładowy program
inicjalizacji modułu USART0
do trybu I²C

W przedstawionym przykładzie inicjalizacji modułu USART0 przyjęto arbitralnie wszystkie parametry interfejsu I²C.

```
;..... USART0, I2C mode initialisation .....  
    bis.b    #SWRST,U0CTL      ;USART0 is held in reset state  
    bis.b    #I2C+SYNC,&U0CTL  ;przełączenie USART0 w tryb I2C  
    bic.b    #I2CEN,U0CTL      ;blokada działania I2C  
    mov.b    #I2CSSEL1,&I2CTCTL ;Użycie SMCLK dla I2C  
    mov.b    #003h,&I2CSCLH    ;Ustawienie czasu stanu wysokiego SCL  
    mov.b    #003h,&I2CSCLL    ;Ustawienie czasu stanu niskiego SCL  
    mov.b    #001h,&I2CNDAT    ;Transmisja jednego bajtu  
    mov.w    #20h,&I2CSA       ;Ustawienie adresu slave  
    bis.b    #I2CEN,&U0CTL      ;Włączenie I2C  
    bic.b    #SWRST,&U0CTL      ;USART0 released for operation  
;.....
```

Literatura

- [1] „Introduction to SPI Interface”, Piyu Dhaker, Analog Devices, 2019
- [2] „The I²C-bus specification, version 2.1”, Philips Semiconductors, 2000
- [3] „I²C Communication Protocol: Understanding I2C Primer, PMBus, and SMBus”, Mary Grace Legaspi, Eric Peña, Analog Devices, 2021
- [4] „UART: A Hardware Communication Protocol, Understanding Universal Asynchronous Receiver/Transmitter”, Eric Peña, Mary Grace, Analog Devices, 2020
- [5] „MSP430x1xx Family User's guide” (slau049f.pdf), Texas Instruments, 2006 ⁽¹⁾
- [6] „Corrections to *MSP430x1xx Family User's Guide*” (slau049.pdf), Texas Instruments, 2016
- [7] „MSP430F15x, MSP430F16x, MSP430F161x Mixed Signal Microcontroller datasheet (Rev. G)”, Texas Instruments, 2011
- [8] „Lokalne interfejsy szeregowo w systemach cyfrowych”, Jacek Bogusz, BTC
- [9] „Szeregowe interfejsy cyfrowe”, Wojciech Mielczarek, HELION
- [10] „Teoria obwodów elektrycznych”, Stanisław Bolkowski, WNT