Wykład 10

Komunikacja

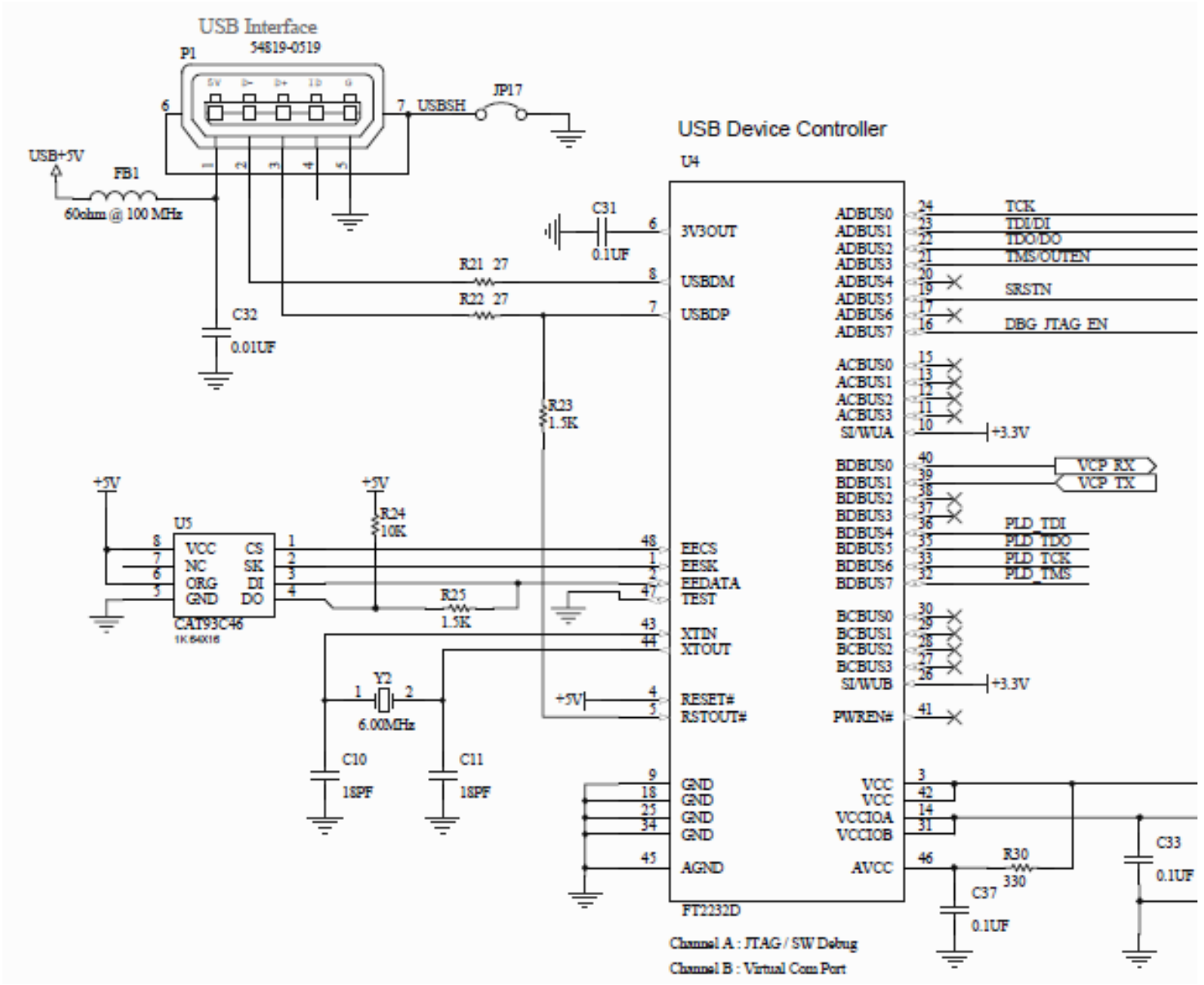
Interfejsy komunikacji szeregowej

- Universal Asynchronous Receiver/Transmitter (UART)
- Synchronous Serial Interface (SSI)
- Inter-Integrated Circuit (I2C)
- Ethernet

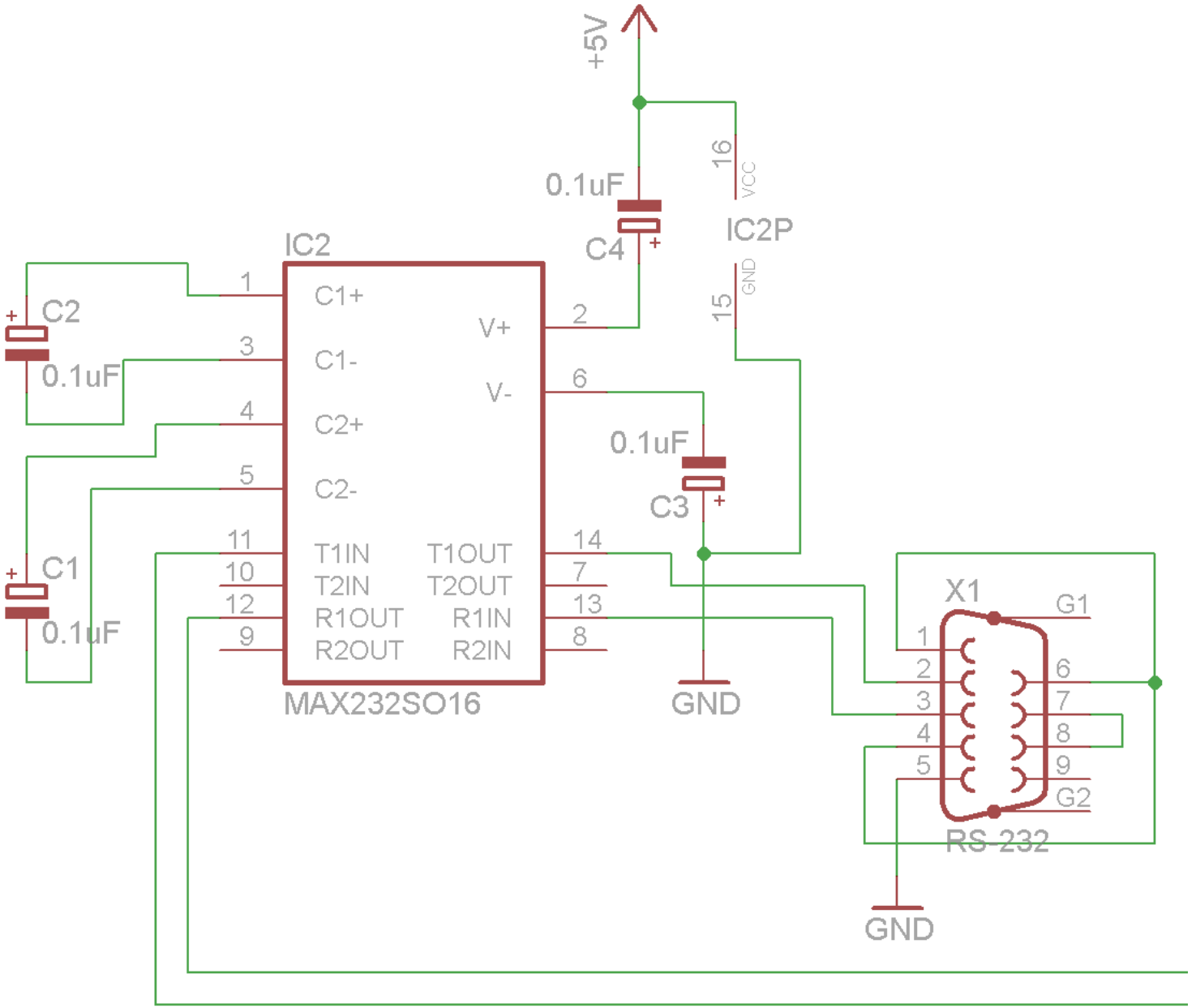
Universal Asynchronous Receiver/Transmitter (UART)

- Universal Asynchronous Receiver/Transmitter (UART) jest uniwersalnym w pełni programowalnym nadajnikiem/odbiornikiem do transmisji szeregowej, realizującym sprzętowo konwersję danych równoległych na postać szeregową i odwrotnie.
- Udostępnia interfejs zgodny funkcjonalnie z układem 16C550, lecz nie zapewniający zgodności na poziomie rejestrów.
- W mikrokontrolerze LM3S6965 wbudowane są trzy takie interfejsy.

UART – komunikacja przez USB



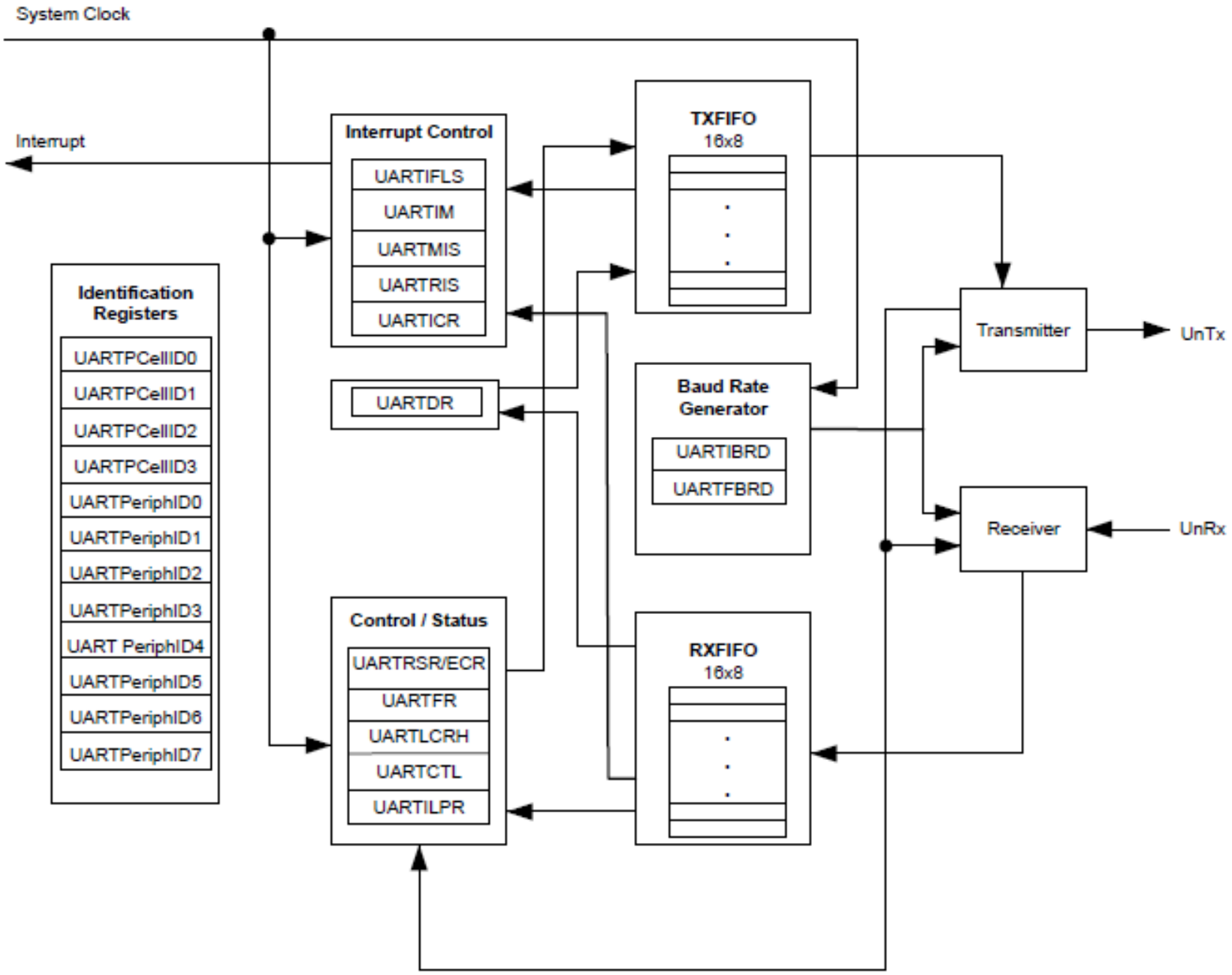
UART – komunikacja przez RS232



Universal Asynchronous Receiver/Transmitter (UART)

- Podstawowe cechy interfejsu:
  - niezależne bufory FIFO dla transmisji i odbioru danych,
  - programowalna długość buforów FIFO,
  - poziomy wyzwalania bufora FIFO 1/8, 1/4, 1/2, 3/4 i 7/8,
  - programowalny generator zapewniający transfer do 3.125 Mbps,
  - możliwość ustawienia dla trybu asynchronicznego bitów startu, stopu i parzystości,
  - 5, 6, 7 lub 8 bitów danych
  - generacja i detekcja bitów kontrolnych,
  - generacja 1 lub 2 bitów stopu,
  - dekoder/enkoder protokołu IrDA
  - wewnętrzna pętla (loopback) dla celów diagnostycznych..

Universal Asynchronous Receiver/Transmitter (UART)



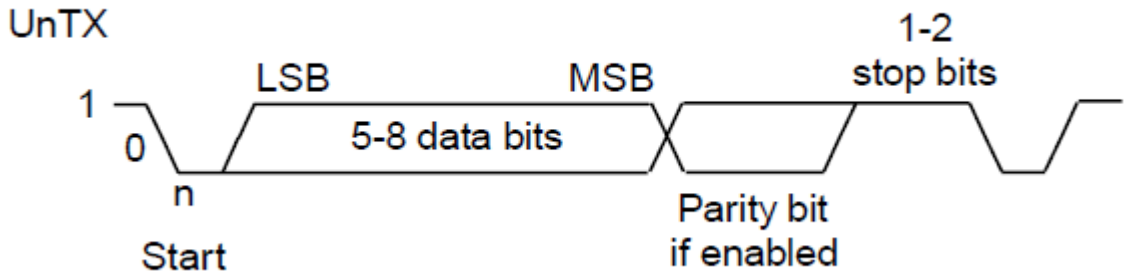
UART - konfiguracja

- Interfejs UART jest konfigurowany do transmisji lub odbioru za pomocą bitów TXE i RXE w rejestrze UART Control (UARTCTL).
- Transmisja lub odbiór są automatycznie uaktywniane po resecie. Przed ustawieniem bitów w rejestrze kontrolnym należy wyłączyć moduł poprzez skasowanie bitu UARTEN w rejestrze UARTCTL.
- Jeżeli UART jest wyłączony w trakcie transmisji, to przed zatrzymaniem aktualna operacja odbioru lub nadawania jest kończona.
- W przypadku realizacji funkcji portu podczerwieni IrDA, układ należy podłączyć do nadajnika/odbiornika podczerwieni w celu zapewnienia odpowiedniej warstwy fizycznej IrDA SIR. Funkcje SIR są ustawiane w rejestrze UARTCTL.

UART - Logika transmisji

- W celu wysłania danych, blok konwersji równoległej na szeregową odczytuje dane z bufora FIFO.
- Układy logiczne wysyłają na wyjście interfejsu szeregowy strumień danych poprzedzony bitami startu, następnie bity danych (pierwszy LSB), bit parzystości i bity stopu, w zależności od przyjętej konfiguracji w rejestrach kontrolnych.
- Przy odbiorze układy logiczne dokonują konwersji z postaci szeregowej na równoległą, odczytanego strumienia danych, po detekcji prawidłowego sygnału startu.
- Sprawdzanie bitów parzystości, błędów ramki, ciągłości linii jest dokonywane przez układ, a informacje te są dopisywane do danych znajdujących się w odbiorczym buforze FIFO

UART - Ramka danych



UART - Ustawianie prędkości transmisji

- Dzielnik pozwalający ustawić odpowiednia prędkość transmisji jest 22-bitową liczbą składającą się z 16-bitowej części całkowitej i 6-bitowej części ułamkowej.
- Liczba utworzona z tych dwóch wartości jest wykorzystywana przez układ generatora szybkości transmisji do określenia czasu trwania bitu.
- 16-bitowa wartość jest ładowana przez rejestr UART Integer Baud-Rate Divisor (UARTIBRD), a 6-bitowa wartość ułamkowa przez rejestr UART Fractional Baud-Rate Divisor (UARTFBRD).
- Zależność pomiędzy wartością dzielnika a szybkością zegara jest następująca:

$$BRD = BRDI + BRDF = SysClk / (16 * Baud\ Rate)$$

UART - Ustawianie prędkości transmisji

- Wartość 6-bitowa jaką należy wpisać do rejestru UARTFBRD może być obliczona poprzez przemnożenie przez 64 wartości ułamkowej otrzymanej ze wzoru i dodanie 0.5 w celu uwzględnienia błędów zaokrągleń.
- Można to przedstawić zależnością:

$$UARTFBRD[DIVFRAC] = integer(BRDF * 64 + 0.5)$$

- Moduł UART generuje dodatkowy wewnętrzny zegar o częstotliwości 16 razy większej od zegara transmisji, który wykorzystywany jest do detekcji błędów w czasie transmisji.

UART - Ustawianie prędkości transmisji

- Rejestry UARTIBRD, UARTFBRD wraz z rejestrem UART Line Control, High Byte (UARTLCRH) tworzą wewnętrzny rejestr 30-bitowy.
- Rejestr ten jest uaktualniany tylko wtedy, kiedy wykonywany jest zapis do rejestru UARTLCRH. W związku z tym po ustawieniu wartości dzielników w odpowiednich rejestrach, należy dokonać zapisu do UARTLCRH.

- Możliwe są więc następujące sekwencje:
  - zapis UARTIBRD, zapis UARTFBRD i zapis UARTLCRH,
  - zapis UARTFBRD, zapis UARTIBRD i zapis UARTLCRH,
  - zapis UARTIBRD i zapis UARTLCRH,
  - zapis UARTFBRD i zapis UARTLCRH,

UART - Operacje FIFO

- Moduł UART posiada dwa 16-pozycyjne bufory FIFO, jeden do nadawania, drugi do odbioru danych.
- Obydwa bufory FIFO są dostępne przez rejestr UART Data (UARTDR).
- Operacje odczytu z UARTDR zwracają 12-bitową wartość składającą się z 8 bitów danych i czterech bitów zawierających flagi błędów.
- Operacje zapisu do tego rejestru umieszczają 8-bitowe dane w buforze nadawczym FIFO.
- Po resecie obydwa bufory FIFO są zablokowane i pracują jako pojedyncze rejestry danych.
- FIFO można odblokować poprzez ustawienie bitu FEN w rejestrze UARTLCRH.

UART - Operacje FIFO

- Stan FIFO może być monitorowany poprzez rejestr UART Flag (UARTFR) i rejestr UART Receive Status (UARTFR).
- Sprzętowo monitorowane jest czy bufor jest pusty, pełny lub przepełniony. Rejestr UARTFR zawiera flagi pusty i pełny (bity TXFE, TXFF, RXFE i RXFF), natomiast UARTRSR pokazuje przepełnienie (bit OE).
- Moment generacji przerwania przez FIFO jest ustalany w rejestrze UART Interrupt FIFO Level Select (UARTFLS).
- Obydwa bufory FIFO mogą być indywidualnie konfigurowane. Dostępne ustawienia są następujące: 1/8, 1/4, 1/2, 3/4, i 7/8.
- Przykładowo jeżeli ustawiona jest wartość 1/4 to przerwanie jest generowane po odebraniu czterech bajtów danych.
- Po resecie układ ustawiony jest na wartość 1/2.

UART – Adresy bazowe

- Adresy bazowe:
  - UART0: 0x4000.C000
  - UART1: 0x4000.D000
  - UART2: 0x4000.E000

UART – Adresy rejestrów

Offset	Name	Type	Reset	Description page
0x000	UARTDR	R/W	0x0000.0000	UART Data
0x004	UARTRSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear
0x018	UARTFR	RO	0x0000.0090	UART Flag
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control
0x030	UARTCTL	R/W	0x0000.0300	UART Control
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear

UART – Funkcje API

- void UARTBreakCtl (unsigned long ulBase, tBoolean bBreakState)
- long UARTCharGet (unsigned long ulBase)
- long UARTCharNonBlockingGet (unsigned long ulBase)
- tBoolean UARTCharNonBlockingPut (unsigned long ulBase, unsigned char ucData)
- void UARTCharPut (unsigned long ulBase, unsigned char ucData)
- tBoolean UARTCharsAvail (unsigned long ulBase)
- void UARTConfigGet (unsigned long ulBase, unsigned long *pulBaud, unsigned long *pulConfig)
- void UARTConfigSet (unsigned long ulBase, unsigned long ulBaud, unsigned long ulConfig)
- void UARTDisable (unsigned long ulBase)
- void UARTDisableSIR (unsigned long ulBase)

UART – Funkcje API

- void UARTIntClear (unsigned long ulBase, unsigned long ulIntFlags)
- void UARTIntDisable (unsigned long ulBase, unsigned long ulIntFlags)
- void UARTIntEnable (unsigned long ulBase, unsigned long ulIntFlags)
- void UARTIntRegister (unsigned long ulBase, void (*pfnHandler) (void))
- unsigned long UARTIntStatus (unsigned long ulBase, tBoolean bMasked)
- void UARTIntUnregister (unsigned long ulBase)
- unsigned long UARTParityModeGet (unsigned long ulBase)
- void UARTParityModeSet (unsigned long ulBase, unsigned long ulParity)
- tBoolean UARTSpaceAvail (unsigned long ulBase)

Przykładowy program

Inter-Integrated Circuit (I2C)

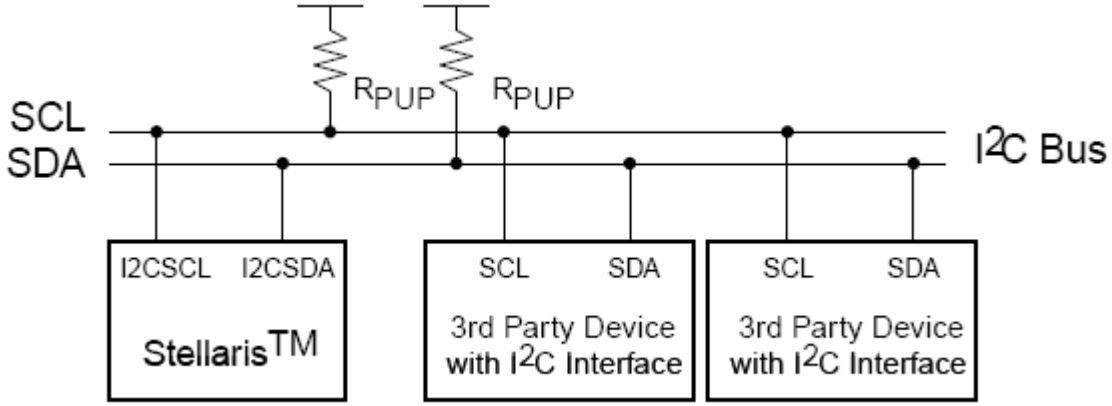
- Magistarala Inter-Integrated Circuit (I2C) została opracowana przez firmę PHILIPS z myślą o stosowaniu jej w urządzeniach audio-video do komunikacji pomiędzy wewnętrznymi układami.
- Umożliwia dwukierunkowy transfer danych między wieloma, podłączonymi do niej układami poprzez dwuprzewodowy interfejs(linie szeregową SDA i linię zegara SCL).
- Interfejs ten wykorzystywany jest do podłączania zewnętrznych elementów takich jak: pamięci szeregowe, urządzenia sieciowe, ekrany LCD, wejścia-wyjścia.
- Interfejs może być także wykorzystany do celów testowania i diagnostyki systemu.
- Urządzenia pracujące na magistrali mogą być realizowane jako master lub slave lub umożliwiają jednocześnie pracę jako master i slave.

Inter-Integrated Circuit (I2C)

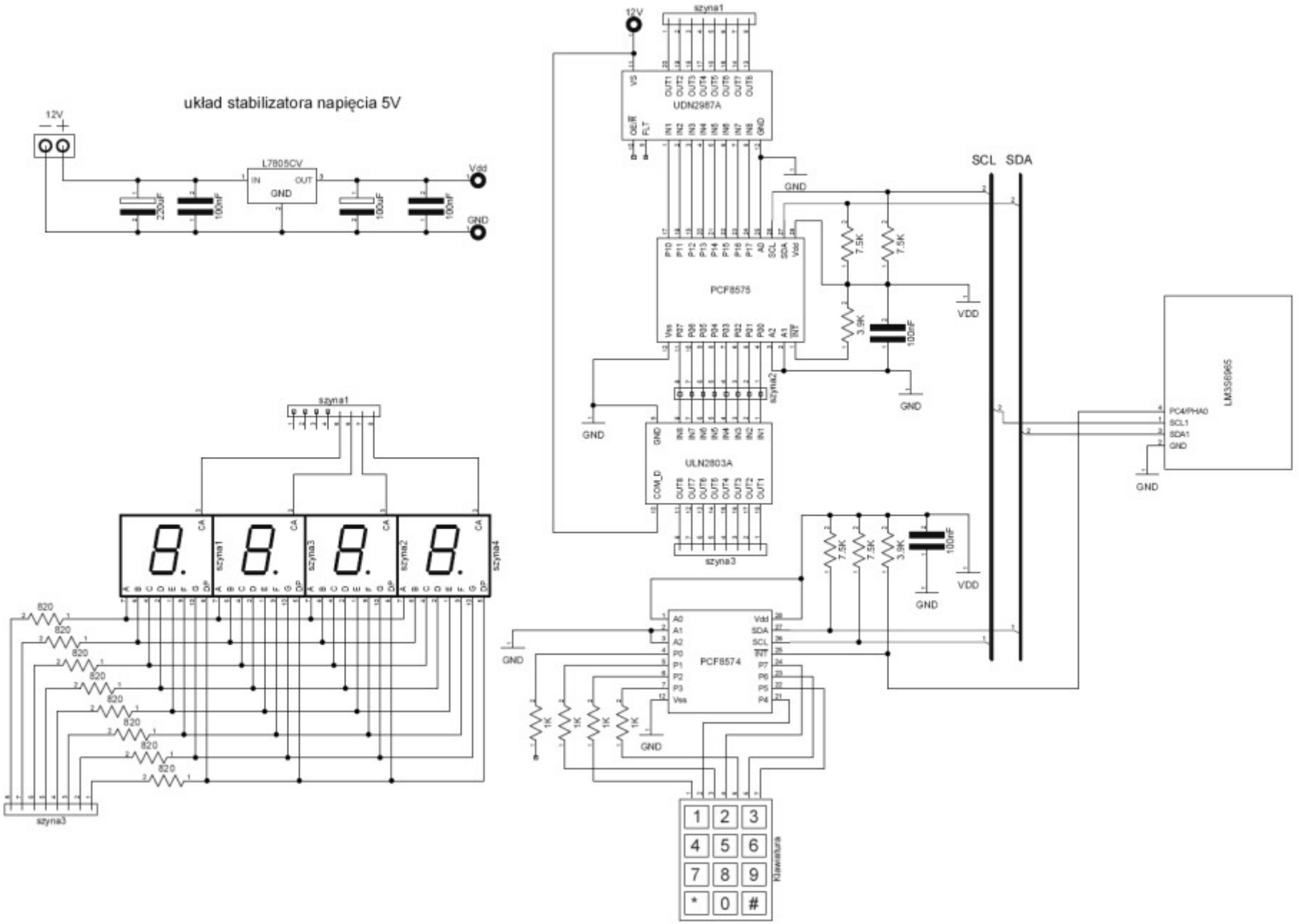
- LM3S6965 zawiera dwa moduły I2C pozwalające na komunikację dwukierunkową z innymi elementami I2C.
- Obie linie muszą być podpięte przez rezystor podciągający do przewodu zasilania, a końcówki układów podłączone do tych linii muszą być typu otwarty kolektor albo otwarty dren.
- Wartość rezystorów podciągających określa się na podstawie napięcia zasilania i pojemności szyny, na którą składają się pojemność linii i pojemność dołączonych układów.
- Zwykle stosuje się wartość z przedziału 4,7 kΩ do 8kΩ.
- Im większa pojemność szyny tym mniejsza powinna być wartość tych rezystancji.
- Zbyt duże wartość w połączeniu z dużą pojemnością układu wprowadzałyby zbyt duże opóźnienia w narastaniu napięcia na liniach szyny.

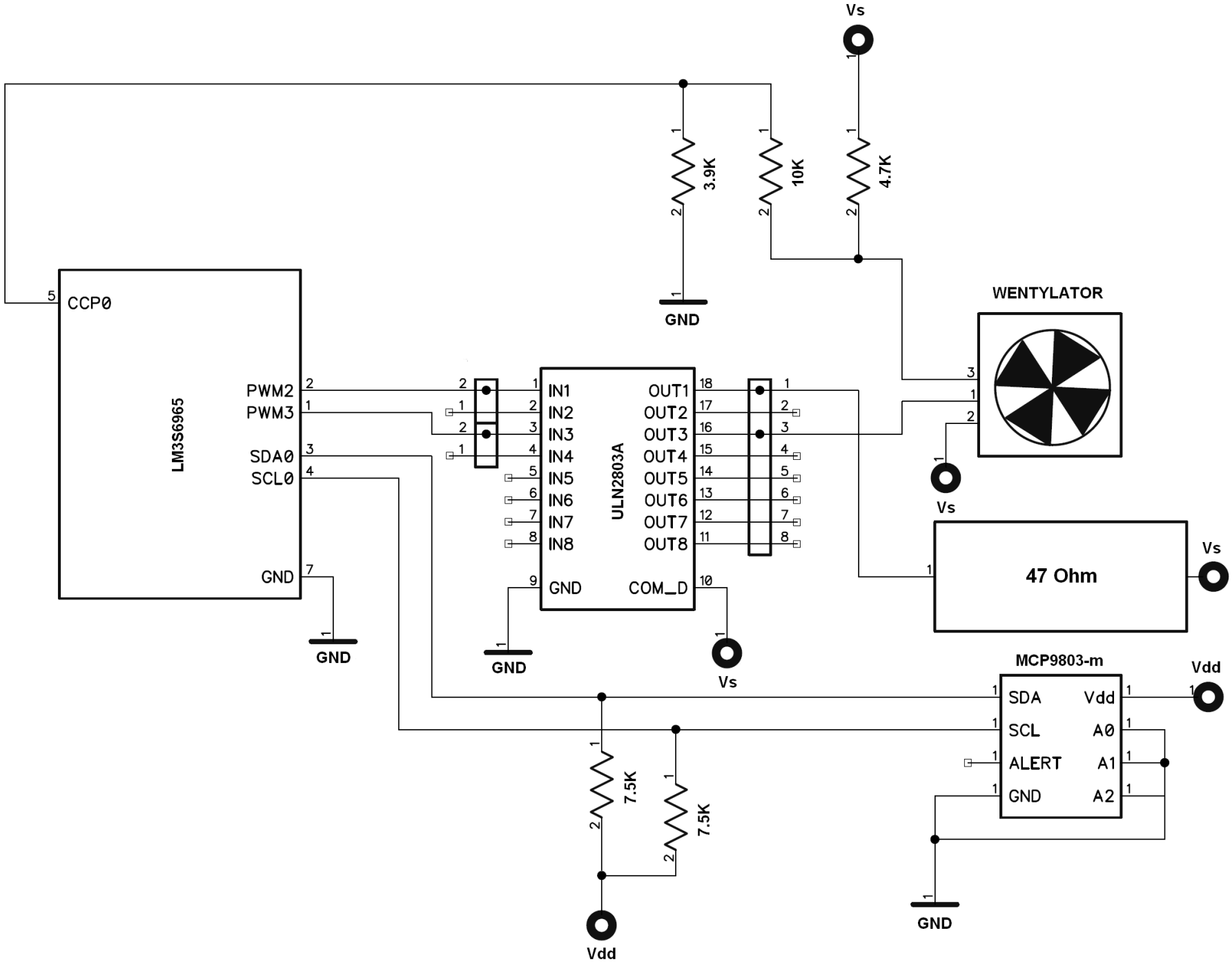
Inter-Integrated Circuit (I2C)

- W standardzie I2C, maksymalna pojemność szyny określona została na 400pF, co odpowiada bardzo rozbudowanej strukturze. Pojedyncze układy podpięte do linii I2C mają zwykle pojemność wyprowadzeń rzędu 5..10pF.
- W celach orientacyjnych podać można, że dla pojemności 400pF i napięcia zasilania 5V, wartość rezystorów podciągających ograniczyć należy do ok. 2,2kΩ .



Inter-Integrated Circuit (I2C)


układ stabilizatora napięcia 5V
12V
L7805CV
IN
OUT
GND
Vdd
220uF
100nF
100uF
100nF
GND
szyna1
szyna2
szyna3
szyna4
CA
A B C D E F G DP
820
820
820
820
820
820
820
szyna3
12V
szyna1
UDN2987A
GND
SCL SDA
7.5K
7.5K
PCF8575
VDD
3.9K
100nF
GND
szyna2
GND
ULN2803A
szyna3
LM35/60/65
PC4/PHA0
SCL1
SDA1
GND
GND
7.5K
7.5K
3.9K
100nF
VDD
GND
PCF8574
1K
1K
1K
1K
GND
1 2 3
4 5 6
7 8 9
* 0 #
Klawiatura


Vs
3.9K
10K
4.7K
GND
CCP0
LM3S6965
PWM2
PWM3
SDA0
SCL0
GND
IN1
IN2
IN3
IN4
IN5
IN6
IN7
IN8
GND
ULN2803A
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
OUT8
COM_D
WENTYLATOR
Vs
47 Ohm
Vs
GND
GND
Vs
MCP9803-m
SDA
SCL
ALERT
GND
Vdd
A0
A1
A2
Vdd
7.5K
7.5K
Vdd
GND

Inter-Integrated Circuit (I2C)

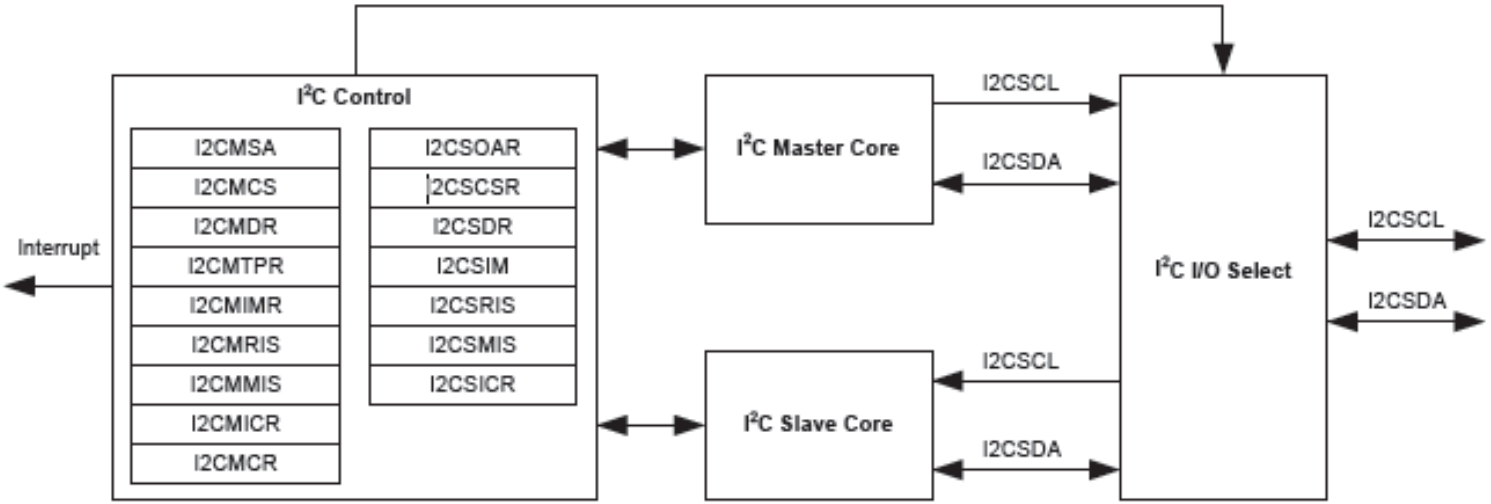
- Istnieją cztery tryby pracy portu:
  - Master Transmit
  - Master Receive
  - Slave transmit
  - Slave receive
  -
- Układ LM3S6965 pozwala na pracę przy dwóch prędkościach:
  - Standard (100kbps);
  - Fast (400kbps)
- Zarówno master jak i slave mogą generować przerwania, master I2C kiedy zakończy operację wysyłania lub odbioru.
- I2C slave generuje przerwania kiedy dane mają być wysyłane lub żądane przez mastera.

Inter-Integrated Circuit (I2C)

Table 15-1. Examples of I^2C Master Timer Period versus Speed Mode

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
4 Mhz	0x01	100 Kbps	-	-
6 Mhz	0x02	100 Kbps	-	-
12.5 Mhz	0x06	89 Kbps	0x01	312 Kbps
16.7 Mhz	0x08	93 Kbps	0x02	278 Kbps
20 Mhz	0x09	100 Kbps	0x02	333 Kbps
25 Mhz	0x0C	96.2 Kbps	0x03	312 Kbps
33Mhz	0x10	97.1 Kbps	0x04	330 Kbps
40Mhz	0x13	100 Kbps	0x04	400 Kbps

Inter-Integrated Circuit (I2C)



Inter-Integrated Circuit (I2C)

Figure 15-4. Complete Data Transfer with a 7-Bit Address

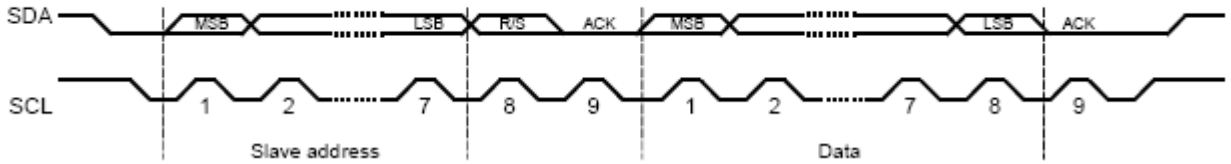
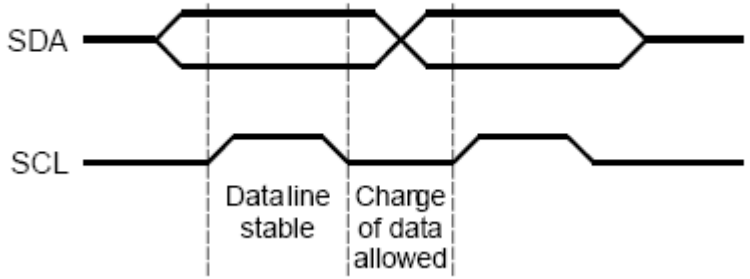


Figure 15-6. Data Validity During Bit Transfer on the I^2C Bus



Inter-Integrated Circuit (I2C)

- void I2CIntRegister (unsigned long ulBase, void (pfnHandler) (void))
- void I2CIntUnregister (unsigned long ulBase)
- tBoolean I2CMasterBusBusy (unsigned long ulBase)
- tBoolean I2CMasterBusy (unsigned long ulBase)
- void I2CMasterControl (unsigned long ulBase, unsigned long ulCmd)
- unsigned long I2CMasterDataGet (unsigned long ulBase)
- void I2CMasterDataPut (unsigned long ulBase, unsigned char ucData)
- void I2CMasterDisable (unsigned long ulBase)
- void I2CMasterEnable (unsigned long ulBase)

Inter-Integrated Circuit (I2C)

- unsigned long I2CMasterErr (unsigned long ulBase)
- void I2CMasterInitExpClk (unsigned long ulBase, unsigned long ulI2CClk, tBoolean bFast)
- void I2CMasterIntClear (unsigned long ulBase)
- void I2CMasterIntDisable (unsigned long ulBase)
- void I2CMasterIntEnable (unsigned long ulBase)
- tBoolean I2CMasterIntStatus (unsigned long ulBase, tBoolean bMasked)
- void I2CMasterSlaveAddrSet (unsigned long ulBase, unsigned char ucSlaveAddr, tBoolean
- bReceive)

Inter-Integrated Circuit (I2C)

- unsigned long I2CSlaveDataGet (unsigned long ulBase)
- void I2CSlaveDataPut (unsigned long ulBase, unsigned char ucData)
- void I2CSlaveDisable (unsigned long ulBase)
- void I2CSlaveEnable (unsigned long ulBase)
- void I2CSlaveInit (unsigned long ulBase, unsigned char ucSlaveAddr)
- void I2CSlaveIntClear (unsigned long ulBase)
- void I2CSlaveIntClearEx (unsigned long ulBase, unsigned long ulIntFlags)
- void I2CSlaveIntDisable (unsigned long ulBase)

Inter-Integrated Circuit (I2C)

```
//
// Initialize Master and Slave
//
I2CMasterInitExpClk(I2C_MASTER_BASE, SysCtlClockGet(), true);
//
// Specify slave address
//
I2CMasterSlaveAddrSet(I2C_MASTER_BASE, 0x3B, false);
//
// Place the character to be sent in the data register
//
I2CMasterDataPut(I2C_MASTER_BASE, 'Q');
//
// Initiate send of character from Master to Slave
//
I2CMasterControl(I2C_MASTER_BASE, I2C_MASTER_CMD_SINGLE_SEND);
//
// Delay until transmission completes
//
while(I2CMasterBusBusy(I2C_MASTER_BASE))
{
}
```

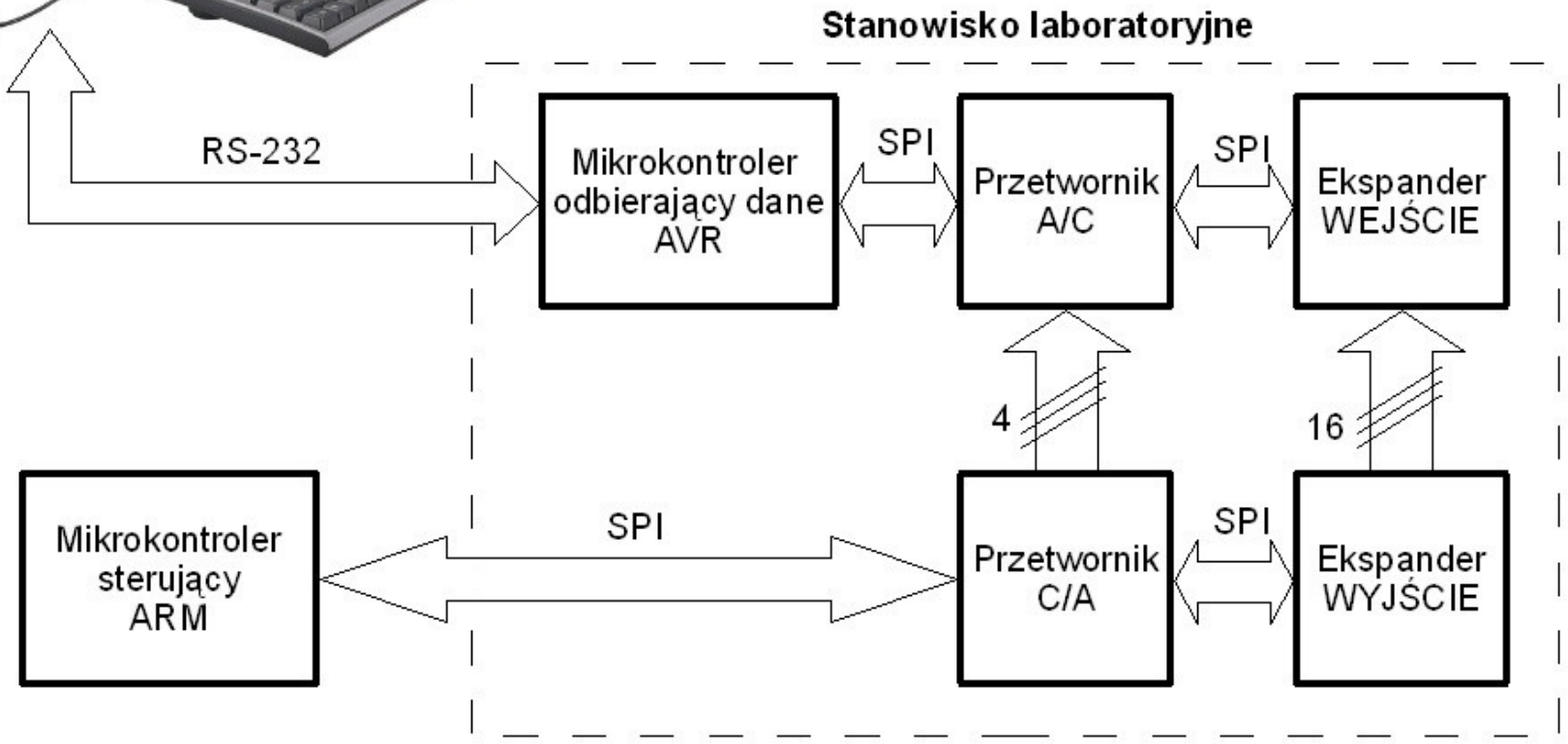
Inter-Integrated Circuit (I2C)

- void I2CSlaveIntDisableEx (unsigned long ulBase, unsigned long ulIntFlags)
- void I2CSlaveIntEnable (unsigned long ulBase)
- void I2CSlaveIntEnableEx (unsigned long ulBase, unsigned long ulIntFlags)
- tBoolean I2CSlaveIntStatus (unsigned long ulBase, tBoolean bMasked)
- unsigned long I2CSlaveIntStatusEx (unsigned long ulBase, tBoolean bMasked)
- unsigned long I2CSlaveStatus (unsigned long ulBase)

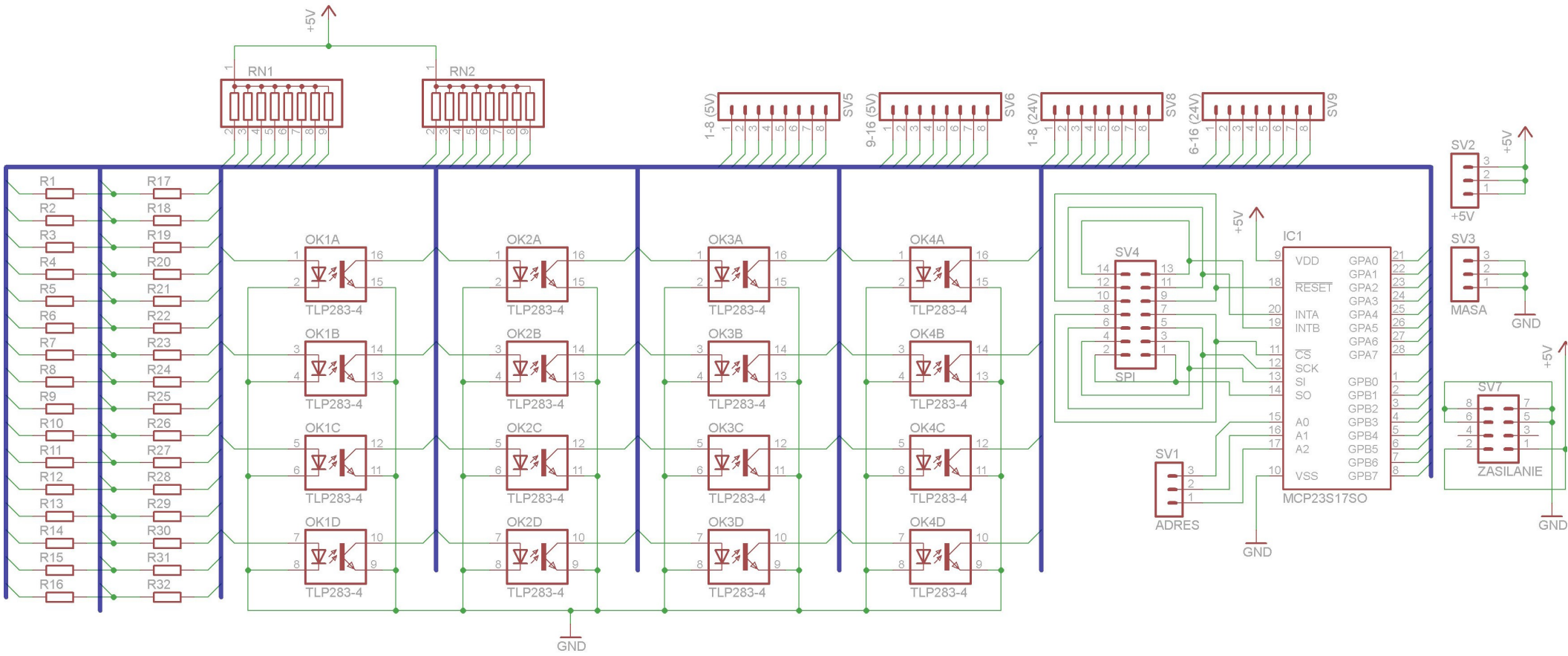
Synchronous Serial Interface (SSI)

- Synchronous Serial Interface (SSI) jest synchronicznym interfejsem szeregowym pozwalającym na komunikację w trybie master lub slave z urządzeniami obsługującymi ten sposób transmisji.
- Główne cechy interfejsu:
  - praca w trybie master lub slave,
  - programowo ustawiane szybkości transmisji,
  - niezależne bufory FIFO dla nadawania i odbioru (16-bitów, 8 poziomów),
  - interfejs programowy dla Freescale SPI, MICROWIRE, lub Texas Instruments,
  - programowalna wielkość ramki danych (od 4 do 16-bitów)
  - wewnętrzna pętla (loopback) dla celów diagnostycznych.

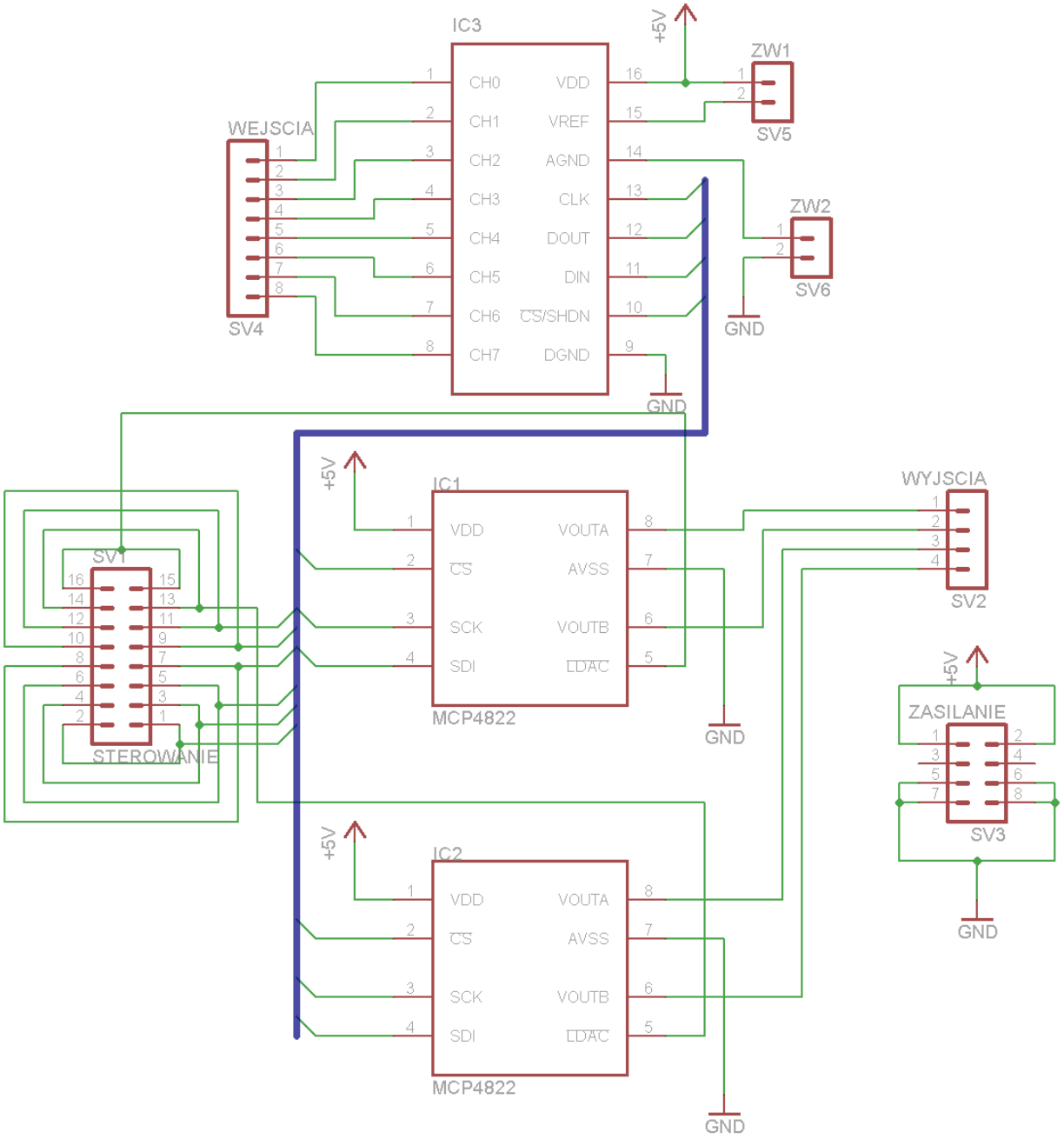
Synchronous Serial Interface (SSI)



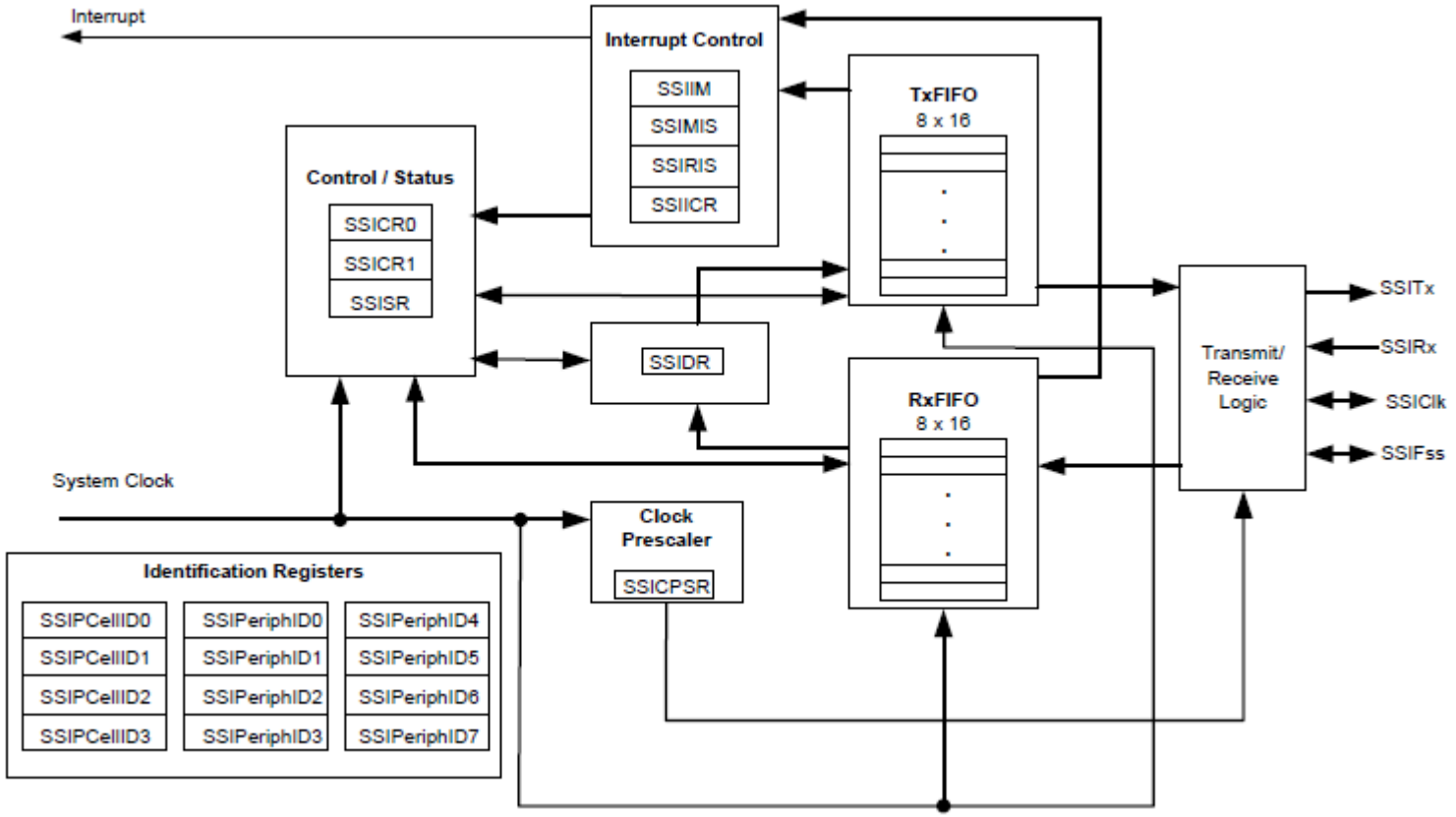
Synchronous Serial Interface (SSI)



Synchronous Serial Interface (SSI)



Synchronous Serial Interface (SSI)



SSI - Ustawianie szybkości transmisji

- Moduł SSI zawiera dzielnik zegara i układ skalujący na potrzeby ustawiania szybkości transmisji.
- Szybkość transmisji jest otrzymywana przez podzielenie 50 MHz zegara wejściowego.
- Pierwsze zegar jest dzielony przez parzystą wartość skalującą CPSDVSR (2 do 254), która jest ustawiana w rejestrze SSI Clock Prescale (SSICPSR).
- Następnie zegar jest dzielony przez wartość od 1 do 256, która jest 1+SCR, gdzie SCR jest wartością zapisaną w rejestrze SSI Control0 (SSICR0).

SSI - Ustawianie szybkości transmisji

- Częstotliwośc zegara wyjściowego jest określona jako:

$$FSSIClk = FSysClk / (CPSDVSR * (1 + SCR))$$

- Z obliczeń maksymalna predkośc transmisji wynosi 25 MHZ, jednak moduł może nie być zdolny do pracy z tą prędkością.
- Dla trybu master zegar musi być przynajmniej dwa razy szybszy niż SSIClk.
- Dla trybu slave zegar systemowy musi być przynajmniej 12 razy szybszy niż SSIClk.

SSI - Formaty ramki

- Ramka danych ma długość pomiędzy 4 a 16-bitów, w zależności od konfiguracji i jest transmitowana zaczynając od bitu MSB.
- W układzie istnieje możliwość ustawienia trzech rodzajów ramek:
  - Texas Instruments synchronous serial
  - Freescale SPI
  - MICROWIRE
- Dla wszystkich trzech formatów zegar (SSIClk) jest nieaktywny jeżeli SSI jest w trybie idle, a uaktywniany jest tylko na czas transmisji danych.

SSI - Formaty ramki

- Dla trybów Freescale SPI i MICROWIRE pin SSIFss ma stan niski i jest uaktywniany w czasie transmisji.
- Dla protokołu Texas Instruments bit ten zmienia wartość dla każdego taktu zegara.
- Tryb pracy MICROWIRE stosuje specjalną metodę wymiany komunikatów i zapewnia transmisję half-duplex, pozostałe dwa formaty umożliwiają transmisję w trybie full-duplex.

SSI - Formaty ramki

Figure 14-2. TI Synchronous Serial Frame Format (Single Transfer)

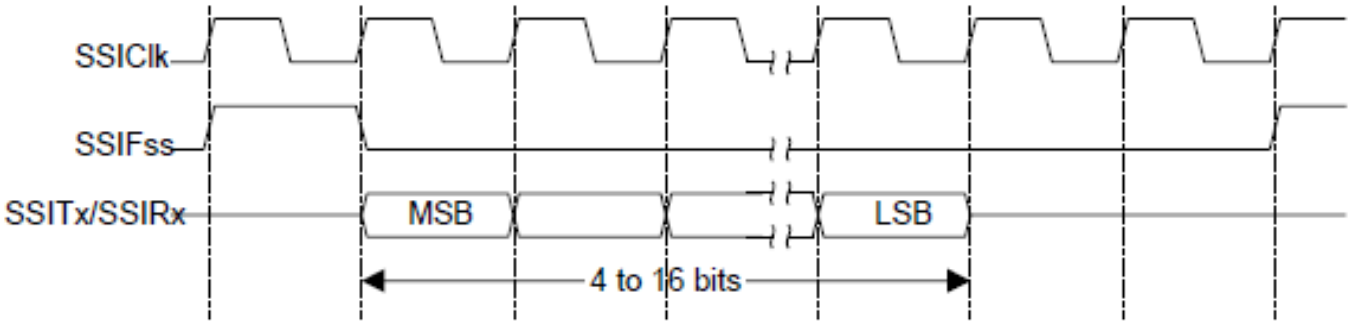
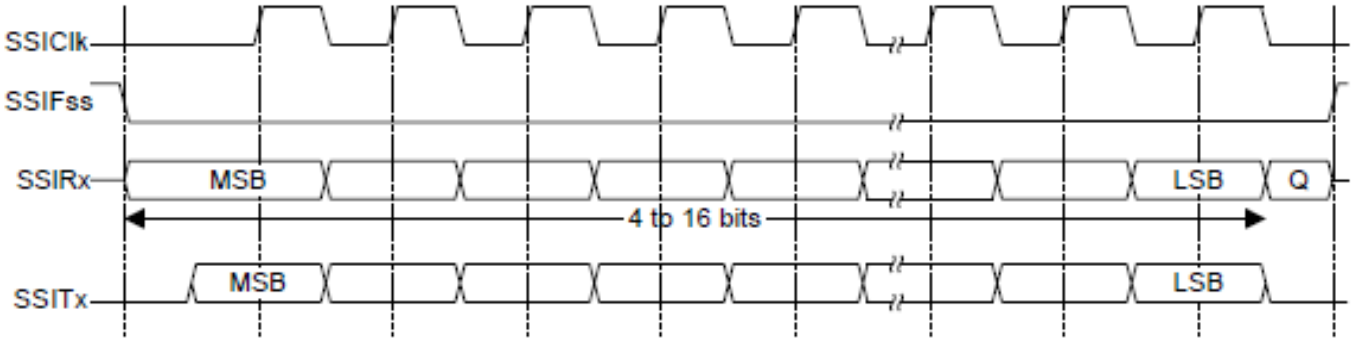


Figure 14-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0



SSI - Formaty ramki

Figure 14-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0

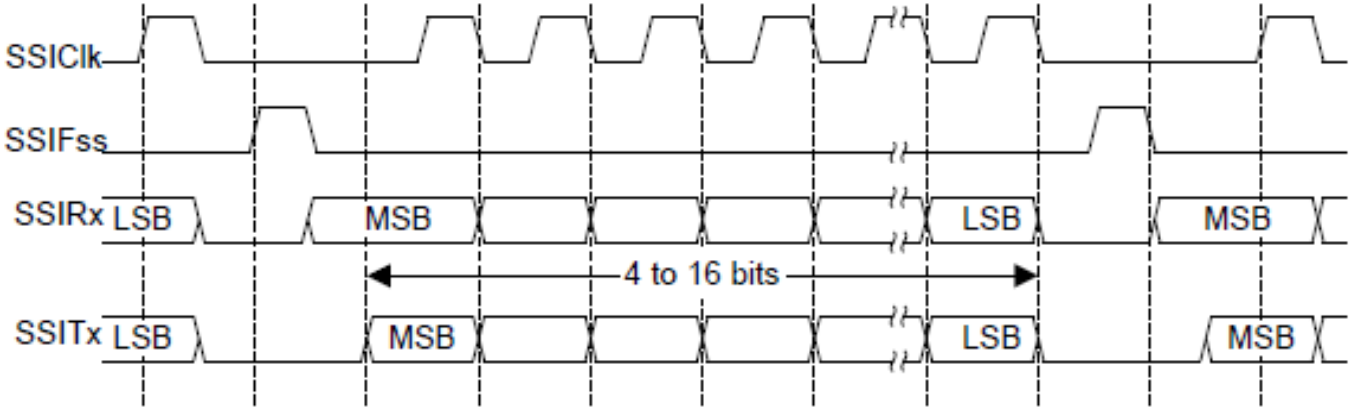
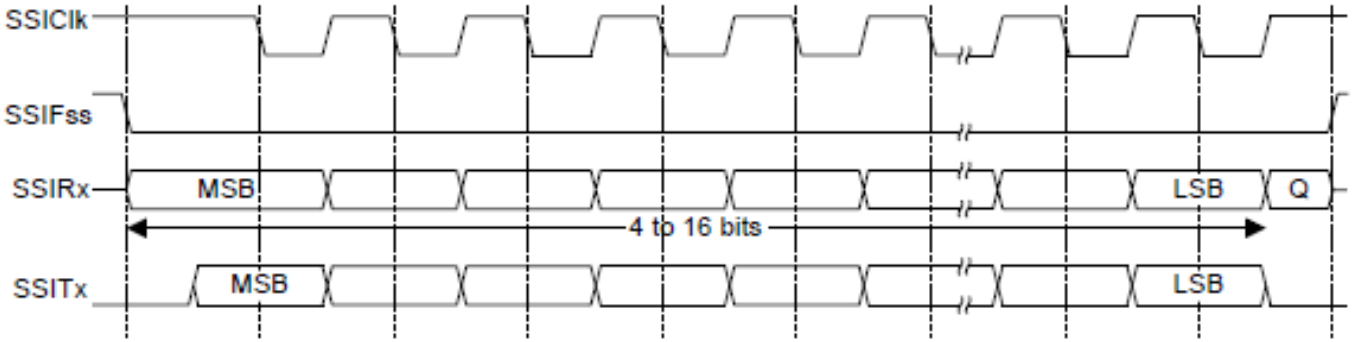


Figure 14-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0



SSI - Formaty ramki

Figure 14-10. MICROWIRE Frame Format (Single Frame)

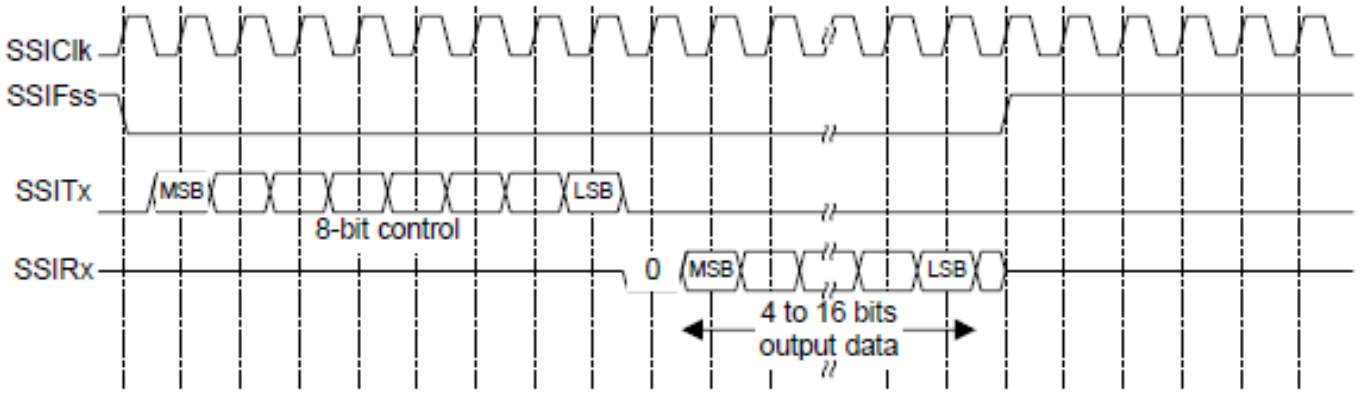
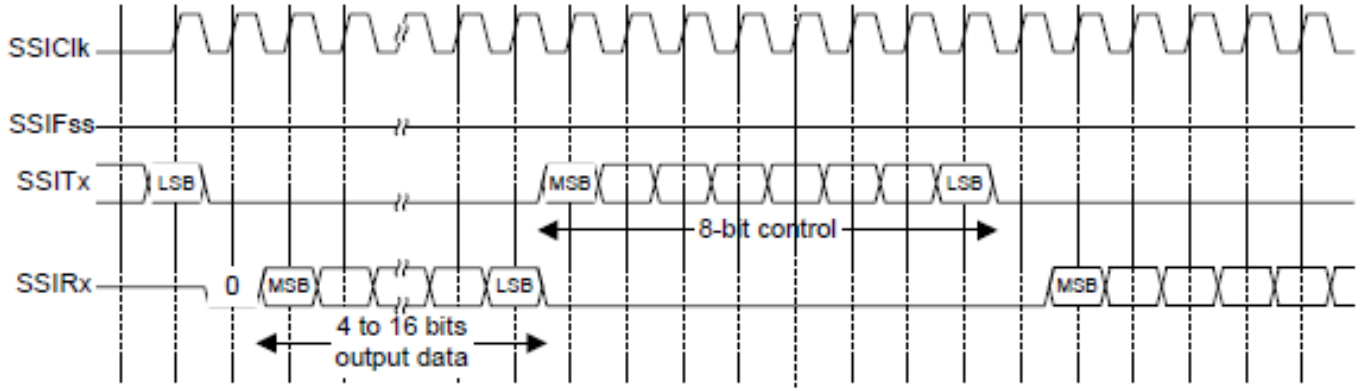


Figure 14-11. MICROWIRE Frame Format (Continuous Transfer)



SSI – Adresy i rejestry

Adresy bazowe

- SSI0: 0x4000.8000

Mapa rejestrów

Offset	Name	Type	Reset	Description page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1
0x008	SSIDR	R/W	0x0000.0000	SSI Data
0x00C	SSISR	RO	0x0000.0003	SSI Status
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask

SSI - Funkcje API

- void SSIConfig (unsigned long ulBase, unsigned long ulProtocol, unsigned long ulMode, unsigned
- long ulBitRate, unsigned long ulDataWidth)
- void SSIDataGet (unsigned long ulBase, unsigned long *pulData)
- long SSIDataNonBlockingGet (unsigned long ulBase, unsigned long *pulData)
- long SSIDataNonBlockingPut (unsigned long ulBase, unsigned long ulData)
- void SSIDataPut (unsigned long ulBase, unsigned long ulData)
- void SSIDisable (unsigned long ulBase)
- void SSIEnable (unsigned long ulBase)

SSI - Funkcje API

- void SSIIntClear (unsigned long ulBase, unsigned long ulIntFlags)
- void SSIIntDisable (unsigned long ulBase, unsigned long ulIntFlags)
- void SSIIntEnable (unsigned long ulBase, unsigned long ulIntFlags)
- void SSIIntRegister (unsigned long ulBase, void (*pfnHandler) (void))
- unsigned long SSIIntStatus (unsigned long ulBase, tBoolean bMasked)
- void SSIIntUnregister (unsigned long ulBase)

SSI - Przykłady wykorzystania

```
char *pcChars = "SSI Master send data.";
long lIdx;
//
// Configure the SSI.
//
SSIConfig(SSI_BASE, SSI_FRF_MOTO_MODE0, SSI_MODE_MASTER, 2000000, 8);
//
// Enable the SSI module.
//
SSIEnable(SSI_BASE);
//
// Send some data.
//
lIdx = 0;
while(pcChars[lIdx])
  {
    if(SSIDataPut(SSI_BASE, pcChars[lIdx]))
      {
        lIdx++;
      }
  }
```