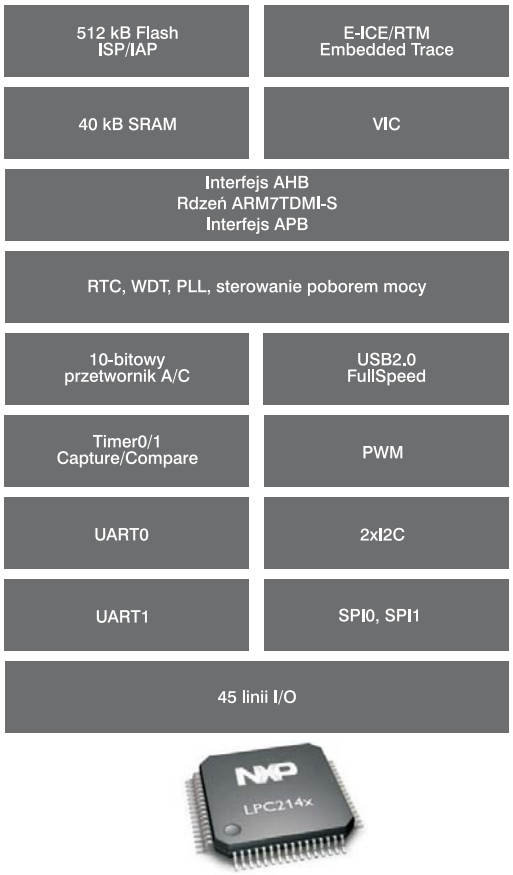
1.1. Mikrokontroler LPC2148

Mikrokontroler LPC2148 produkowany przez firmę NXP należy do popularnej rodziny mikrokontrolerów LPC2000, wyposażonych w 32-bitowy rdzeń ARM7TDMI-S. Jego maksymalna częstotliwość taktowania wewnętrznego wynosi 60 MHz. Mikrokontrolery LPC2148 wyposażono w pamięć programu typu Flash z możliwością programowania w systemie i aplikacji, której pojemność wynosi 512 kB. Także wewnętrzna pamięć RAM ma dużą pojemność, wynosi ona bowiem 40 kB.

Mikrokontroler LPC2148 charakteryzuje się bogatym wyposażeniem wewnętrznym, co widać na schemacie blokowym pokazanym na **rysunku 1.1**, ale w aplikacjach, na których skupiamy się w książce (czyli sterowaniu wyświetlaczy), będziemy korzystać przede wszystkim z linii portów wejścia-wyjścia oraz interfejsu SPI i z tego powodu te układy peryferyjne zostaną omówione szczegółowo w dalszej części rozdziału.

Opisywanie budowy i działania rdzenia, układu przerwań, systemu taktującego oraz układów peryferyjnych znacznie wykracza poza ramy tej książki. Czytelnikom zainteresowanym tą tematyką polecamy książkę Lucjana Bryndzy *LPC2000 Mikrokontrolery z rdzeniem ARM7* (także wydana przez Wydawnictwo BTC).



Rys. 1.1. Schemat blokowy mikrokontrolera LPC2148

Skupimy się zatem na układach peryferyjnych: portach i module SPI wykorzystywanych do obsługi opisywanych wyświetlaczy. Pozostałe informacje można znaleźć w dokumentacji mikrokontrolerów LPC dostępnej na stronie *www.nxp.com*.

1.2. Porty I/O mikrokontrolera LPC2148

Większość sterowników wyświetlaczy graficznych i alfanumerycznych ma możliwość komunikowania się z zewnętrznym układem sterowania za pomocą magistrali równoległej. Magistrale te są przystosowane do podłączenia wyświetlaczy do magistral systemowych (w przestrzeni adresowej) 8-bitowych mikrokontrolerów Intel 8080 lub Motorola 6800. W systemach budowanych z zastosowaniem mikrokontrolerów wyposażonych w wewnętrzną pamięć Flash, pamięć RAM i inne peryferie, magistrale zazwyczaj nie są wyprowadzane na zewnątrz, więc żeby sterowanie wyświetlaczami z magistralą równoległa było możliwe, trzeba ich pracę emulować programowo na uniwersalnych liniach portów mikrokontrolera. Możliwe jest też programowe emulowanie transmisji w standardzie SPI. Z tych powodów porty mikrokontrolera są podstawowym modułem peryferyjnym wykorzystywanym do sterowania wyświetlaczy.

Mikrokontroler LPC2148 ma 2 uniwersalne porty GPIO:

– 32-bitowy PORT0. Fizycznie są zaimplementowane 32 linie oznaczone P0.0...P0.31.
– 32-bitowy PORT1. Fizycznie zaimplementowano 16 linii oznaczonych P1.16...P1.31.

Wszystkie linie obu portów są współdzielone z wyprowadzeniami modułów peryferyjnych. O tym, jaką funkcję spełnia wyprowadzenie mikrokontrolera, decydują rejestry PINSEL0, PINSEL1 i PINSEL2 układu *Pin Connect Block*. Do każdego z wyprowadzeń przypisano 2 bity jednego z rejestru PINSELx, dzięki czemu każdej linii można przypisać 4 różne funkcje.

W **tabeli 1.1** pokazano znaczenie bitów rejestru PINSEL0. Opis pozostałych rejestrów PINSEL1 i PINSEL2 można znaleźć w dokumentacji mikrokontrolera. Wyjątkiem od zasady przydzielania funkcji wyprowadzeniom mikrokontrolera przez rejestry PINSEL są wejścia analogowe przetwornika analogowo-cyfrowego. Jeżeli przetwornik jest aktywny, to niezależnie od przydzielonej wyprowadzeniu funkcji mikrokontroler będzie odczytywał i poddawał konwersji napięcie na odpowiednim wyprowadzeniu. Warunkiem poprawnego konwertowania napięć na wejściu przetwornika jest skonfigurowanie go jako wejścia analogowego przez odpowiednie zaprogramowanie rejestru PINSEL2.

Po włączeniu zasilania wszystkie bity rejestrów PINSELx są wyzerowane i każde wyprowadzenie we/wy spełnia rolę linii GPIO.

W starszych wersjach mikrokontrolerów serii LPC2000 rejestry modułu portów były dołączone do rdzenia poprzez bramkę magistrali VPB (*VLSI Peripherial Bus*). Z tego powodu maksymalna częstotliwość zmian stanów na liniach portów była dużo niższa, niżby to wynikało z czasu wykonania rozkazów ustawiających i zerujących linie portów. W nowszych mikrokontrolerach dostęp do rejestrów sterujących portami jest również możliwy za pośrednictwem magistrali AMBA AHB (*Advanced*

Tab. 1.1. Funkcje bitów rejestru PINSEL0 i odpowiadające im funkcje linii I/O (po zerowaniu mikrokontrolera stany wszystkich bitów są „0")

Bity	Wartość		Funkcja
1:0	P0.0	00	GPIO Port 0.0
		01	TXD (UART0)
		10	PWM1
		11	–
3:2	P0.1	00	GPIO Port 0.1
		01	RxD (UART0)
		10	PWM3
		11	EINT0
5:4	P0.2	00	GPIO Port 0.2
		01	SCL0 (I2C0)
		10	Capture 0.0 (Timer 0)
		11	–
7:6	P0.3	00	GPIO Port 0.3
		01	SDA0 (I2C0)
		10	Match 0.0 (Timer 0)
		11	EINT1
9:8	P0.4	00	GPIO Port 0.4
		01	SCK0 (SPI0)
		10	Capture 0.1 (Timer 0)
		11	AD0.6
11:10	P0.5	00	GPIO Port 0.5
		01	MISO0 (SPI0)
		10	Match 0.1 (Timer 0)
		11	AD0.7
13:12	P0.6	00	GPIO Port 0.6
		01	MOSI0 (SPI0)
		10	Capture 0.2 (Timer 0)
		11	–/AD1.0
15:14	P0.7	00	GPIO Port 0.7
		01	SSEL0 (SPI0)
		10	PWM2
		11	EINT2
17:16	P0.8	00	GPIO Port 0.8
		01	TXD UART1
		10	PWM4
		11	–/AD1.1
19:18	P0.9	00	GPIO Port 0.9
		01	RxD (UART1)
		10	PWM6
		11	EINT3

cd. tab. 1.1

Bity	Wartość		Funkcja
21:20	P0.10	00	GPIO Port 0.10
21:20	P0.10	01	–/RTS (UART1)
		10	Capture 1.0 (Timer 1)
		11	–/AD1.2
23:22	P0.11	00	GPIO Port 0.11
		01	–/CTS (UART1)
		10	Capture 1.1 (Timer 1)
		11	SCL1 (I2C1)
25:24	P0.12	00	GPIO Port 0.12
		01	–/DSR (UART1)
		10	Match 1.0 (Timer 1)
		11	–/AD1.3
27:26	P0.13	00	GPIO Port 0.13
		01	–/DTR (UART1)
		10	Match 1.1 (Timer 1)
		11	–/AD1.4
29:28	P0.14	00	GPIO Port 0.14
		01	–/DCD (UART1)
		10	EINT1
		11	SDA1 (I2C1)
31:30	P0.15	00	GPIO Port 0.15
		01	–/RI (UART1)
		10	EINT2
		11	–/AD1.5

High Performance Bus). Żeby zapewnić programową kompatybilność ze starszymi wersjami mikrokontrolerów, wbudowano mechanizm przełączania trybu dostępu sterowany rejestrem SCS – *System Control and Status*, zamieszczony w **tabeli 1.2**.

Tab. 1.2. Rejestr SCS (bity po zerowaniu przyjmują wartość „0")

Bit	Nazwa	Wartość	Opis
0	GPIO0M	0	GPIO port 0 jest dostępny przez VPB w trybie kompatybilnym z rodziną LPC2000
		1	Włączony jest tryb GPIO dostępu przez AMBA AHB (*high speed*)
1	GPIO1M	0	GPIO port 1 jest dostępny przez VPB w trybie kompatybilnym z rodziną LPC2000
		1	Włączony jest tryb GPIO dostępu przez AMBA AHB (*high speed*)

Po wyzerowaniu mikrokontrolera bity GPIO0M i GPIO1M są ustawiane w stan „0", co powoduje, że porty pracują w trybie kompatybilnym ze starszymi wersjami mikrokontrolerów LPC2000. W takim trybie były sprawdzane wszystkie procedury obsługi wyświetlaczy. Aby wykorzystać tryb *high speed*, trzeba ustawić („1") bity GPIO0M i GPIO1M rejestru SCS oraz zmodyfikować procedury sterujące magistralą wyświetlaczy.

1.2.1. Rejestry IOxPIN

Kiedy GPI0M = 0 i GPIO1M = 0 porty GPIO są sterowane czterema rejestrami: IOxPIN, IOxSET, IOxDIR i IOxCLR. Bity rejestrów IOxPIN odzwierciedlają stan

Tab. 1.3. Budowa rejestru IOxPIN (rejestr tylko do odczytu)

IOPIN	Opis	Wartość bitów po wyzerowaniu mikrokontrolera
31:0	Bit 0 IO0PIN odpowiada linii P0.0 Bit 31 IO0PIN odpowiada P0.31 Bit 0 IO1PIN odpowiada linii P1.0 (*) Bit 31 IO1PIN odpowiada P1.31	Niezdefiniowana

(*) – fizycznie są zaimplementowane wyłącznie linie P1.16…P1.31

linii portów. Te rejestry są używane do odczytywania stanu wyprowadzeń mikrokontrolera przypisanych do linii portów. Jeżeli linia portu nie jest przypisana do wyprowadzenia mikrokontrolera przez zaprogramowanie rejestru PINSEL, to wartość bitu IOxPIN odpowiadająca tej linii jest przypadkowa.

1.2.2. Rejestry IOxSET

Rejestr IOSET jest używany do ustawiania w stan wysoki linii zaprogramowanej jako wyjściowa. Wpisanie jedynki do bitu rejestru IOSET powoduje ustawienie odpowiadającej temu bitowi linii portu w stan wysoki. Na przykład: ustawienie bitu IO1SET.20 spowoduje wymuszenie stanu wysokiego na linii P1.20 portu PORT1. Jeżeli bit rejestru IOxSET jest wyzerowany, nie wpływa to na stan linii portu.

Tab. 1.4. Budowa rejestru IOxSET (rejestr do zapisu i odczytu)

IOSET	Opis	Wartość bitów po wyzerowaniu mikrokontrolera
31:0	Bit 0 IOSET odpowiada linii P0.0 Bit 31 IO0SET odpowiada P0.31 Bit 0 IO1SET odpowiada linii P1.0 (*) Bit 31 IO1SET odpowiada P1.31	0x00000000

(*) – fizycznie są zaimplementowane wyłącznie linie P1.16…P1.31

1.2.3. Rejestry IOxCLR

Rejestr IOSET jest używany do ustawiania linii portu w stan niski. Linia musi być zaprogramowana jako wyjściowa. Wpisanie jedynki do bitu rejestru IOCLR powoduje wymuszenie stanu niskiego na odpowiadającej temu bitowi linii portu. Na przykład: ustawienie bitu IO0CLR.8 spowoduje wymuszenie stanu niskiego na linii P0.8 portu PORT0. Gdy bit rejestru IOxCLR jest wyzerowany, to nie wpływa to na stan linii portu.

Tab. 1.5. Budowa rejestru IOxCLR (rejestr do zapisu i odczytu)

IOCLR	Opis	Wartość bitów po wyzerowaniu mikrokontrolera
31:0	Bit 0 IO0CLR odpowiada linii P0.0 Bit 31 IO0CLR odpowiada P0.31 Bit 0 IO1CLR odpowiada linii P1.0 (*) Bit 31 IO1CLR odpowiada P1.31	0x00000000

(*) – fizycznie są zaimplementowane wyłącznie linie P1.16…P1.31

1.2.4. Rejestry IOxDIR

Rejestr IODIR steruje kierunkiem przepływu danych przez linie portu. Każdą z linii można ustawić jako wejściową, kiedy odpowiadający tej linii bit IOxDIR jest wyzerowany. Jeżeli bit IOxDIR jest ustawiony, to odpowiadająca mu linia jest wyjściem.

Tab. 1.6. Budowa rejestru IOxDIR

IODIR	Opis (rejestr do zapisu i odczytu)	Wartość bitów po wyzerowaniu mikrokontrolera
31:0	Bit 0 IO0DIR odpowiada linii P0.0 Bit 31 IO0DIR odpowiada P0.31 Bit 0 IO1DIR odpowiada linii P1.0 (*) Bit 31 IO1DIR odpowiada P1.31	0x00000000

(*) – fizycznie są zaimplementowane wyłącznie linie P1.16…P1.31

1.3. Interfejs SPI

Transmisja z wykorzystaniem interfejsu SPI jest chętnie wykorzystywana do sterowania wyświetlaczami oraz wszędzie tam, gdzie istotna jest miniaturyzacja urządzenia. Przykładem takich rozwiązań jest sterowanie wyświetlaczami w telefonach komórkowych. Transmisja SPI może być emulowana programowo, ale stosowanie sprzętowych interfejsów SPI upraszcza oprogramowanie i umożliwia wykorzystanie systemu przerwań w bardziej wymagających aplikacjach (obsługa wyświetlacza może się odbywać w tle).

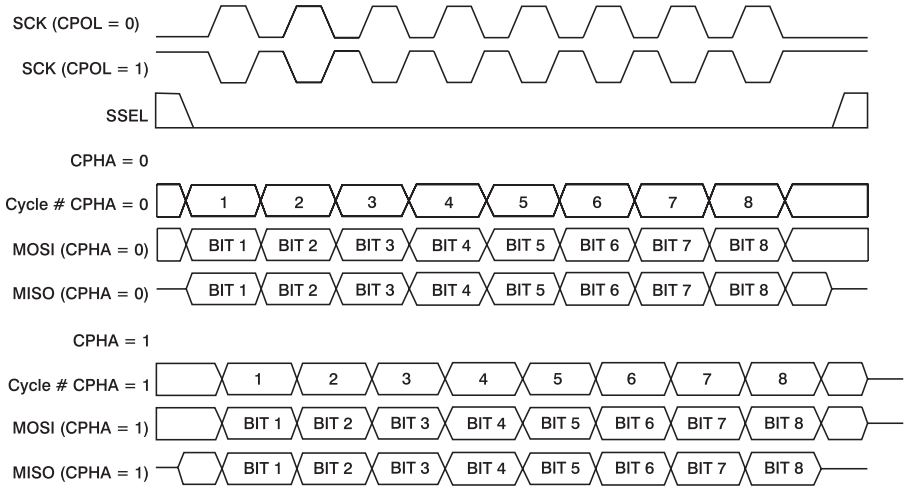
Mikrokontroler LPC2148 wyposażono w dwa interfejsy SPI (SPI0 i SPI1) zgodne ze standardem *Serial Peripherial Interface*. Mogą one pracować jako *master* lub *slave* w pełnym dupleksie (*full duplex*), a przesyłane dane mają programowaną długość od 8 do 16 bitów (tabela 1.8).

W przykładach sterowania wyświetlaczy przez interfejs SPI wykorzystano interfejs SPI0 i dlatego zostanie on tu dokładnie opisany.

1.3.1. Formaty danych SPI0

Na **rysunku 1.2** pokazano cztery możliwe formaty 8-bitowych danych używane podczas transmisji z wykorzystaniem sprzętowego interfejsu SPI mikrokontrolera LPC2148.

Format przesyłanych danych jest określany przez bity CPOL i CPHA rejestru *SPI Control Register*. Zależność pomiędzy przesyłanymi danymi i fazą sygnału zegarowego przedstawiono w **tabeli 1.7**.



Rys. 1.2. Możliwe formaty danych przesyłanych interfejsem SPI0 mikrokontrolerów LPC2148

Tab. 1.7. Zależność pomiędzy danymi i fazą sygnału zegarowego

Stany bitów CPOL i CPHA	Wystawienie pierwszego bitu	Wystawienie następnych bitów	Próbkowanie danych
CPOL = 0, CPHA = 0	Przed pierwszym narastającym zboczem SCK	SCK	SCK
CPOL = 0, CHPA = 1	Pierwsze narastające zbocze SCK	SCK	SCK
CPOL = 1, CPHA = 0	Przed pierwszym opadającym zboczem SCK	SCK	SCK
CPOL = 1, CPHA = 1	Pierwsze opadające zbocze SCK	SCK	SCK

1.3.2. Linie interfejsu SPI0

Warstwa fizyczna interfejsu SPI jest zbudowana z 4 linii: SCK0, SSEL0, MISO0 i MOSI0. Przebieg zegarowy na linii SCK0 (*Serial ClocK*) jest używany do synchronizacji przesyłania danych interfejsu SPI. Źródłem sygnału SCK jest zawsze układ *master*, natomiast układ *slave* jest zawsze odbiornikiem SCK.

Linie danych MISO0 i MOSI0 są jednokierunkowe. Linią MOSI0 (*Master Out Slave In*) są przesyłane dane z układu *master* (źródło danych) do układu *slave* (odbiornik danych). Linia MISO0 (*Master In Slave Out*) jest przeznaczona do przesyłania danych z układu *slave* (źródło sygnału) do układu *master* (odbiornik danych). Linia SSEL0 (*SPI Slave Select*) służy do selekcji aktywnego układu *slave*. Jeżeli do układu *master* trzeba dołączyć kilka układów *slave*, to *master* musi sterować tyloma dodatkowymi liniami SSEL, ile jest układów *slave*. W trakcie transmisji może być aktywna tylko jedna linia SSEL.

Linia P07 zaprogramowana jako SSL0 (tabela 1.1) w trybie *slave*, musi przejść w stan niski przed rozpoczęciem transferu danych i pozostawać w tym stanie do jego zakończenia. Jeżeli w trakcie transmisji danych SSEL0 przejdzie w stan wysoki, to układ *slave* przechodzi w stan *idle* i transmisja zostaje przerwana.

Gdy moduł SPI mikrokontrolera jest ustawiony w tryb *master*, to jego linia SSEL (P0.7) może być zaprogramowana jako wyjściowa GPIO do sterowania wejścia SSEL układu *slave*.

Kiedy P0.7 jest zaprogramowane jako SSL0 w trybie *master*, to musi być na niej „H”.
1. Tryb *slave*
P0.7 musi być zaprogramowane jako SSL0 (PINSEL 0).
2. Tryb *master*
P0.7 może być GPIO I np. sterować SSL układu *Slave* jeżeli P0.7 zostanie zaprogramowany jako SSL0 (PINSEL0), to musi być na niej stan „H”. Jeżeli nie, to nie ma transmisji i zgłaszany jest błąd *Mode Fault* – P1.5.

1.3.3. Rejestry interfejsu SPI0

Rejestr *SPI Control Register (S0SPCR)*

Rejestr *SPI Control Register* (**tabela 1.8**) służy do konfigurowania modułu SPI. Oprócz opisywanych już bitów CPHA i CPOL ustalających zależność pomiędzy przesyłanymi danymi i zegarem SCK, ustawia się tu format przesyłanych danych: długość słowa danych (8 bitów lub więcej), kolejność przesyłania, wybiera tryb transmisji *master*/*slave* i uaktywnia/blokuje zezwolenie na zgłaszanie przerwań.

Tab. 1.8. Rejestr *SPI Control Register (S0SPCR)*

Bit	Nazwa	Wartość	Opis
1:0	–	Zarezerwowane	Użytkownik nie powinien ustawiać tych bitów
2	BitEnable	0 1	SPI przesyła i odbiera 8 bitów SPI przesyła i odbiera zaprogramowaną liczbę bitów (bity 11:8)
3	CPHA	0/1	Zależność pomiędzy danymi z zegarem (tabela 1.7)
4	CPOL	0/1	Polaryzacja sygnału zegarowego (tabela 1.7)
5	MSTR	0 1	SPI pracuje w trybie *slave* SPI pracuje w trybie *master*
6	LSBF	0 1	Bit LSB jest przesyłany jako pierwszy Bit LSB jest przesyłany jako ostatni
7	SPIE	0 1	Zablokowane zgłaszanie przerwań przez moduł SPI Odblokowane zgłaszanie przerwań lub bit MODF staje się aktywny
11:8	BITS		Kiedy bit *BitEnable* jest ustawiony, to te bity ustalają długość transmitowanego słowa 1000 –> 8 bitów; 1001 –> 9 bitów; 1010 –> 10 bitów; 1011 –> 11 bitów; 1100 –> 12 bitów; 1101 –> 13 bitów 1110 –> 14 bitów; 1111 –> 15 bitów; 0000 –> 16 bitów
15:12	–	Zarezerwowane	

Rejestr *SPI Status Register (S0SPSR)*

Rejestr *SPI Stastus Register* pokazano w **tabeli 1.9.**

Tab. 1.9. Rejestr *SPI Status Register (S0SPSR)*

Bit	Nazwa	Opis
2:0	–	Zarezerwowane – użytkownik nie powinien ustawiać tych bitów, przy odczycie wartość niezdefiniowana
3	ABRT	Kiedy bit jest jedynką, to zaszło zdarzenie *slave abort*. Bit jest zerowany po odczytaniu rejestru
4	MODF	Kiedy bit jest jedynką, to wystąpił błąd *mode fault* – moduł jest w trybie MASTER i linia SSEL jest aktywna. Bit jest zerowany po odczytaniu rejestru
5	ROVR	Gdy bit jest jedynką, to wystąpił błąd nadpisania nieodczytanego bajtu odebranego z bufora SPI. Bit jest zerowany po odczytaniu rejestru
6	WCOL	Kiedy bit jest jedynką, to wystąpił błąd zapisania kolejnego bajtu do *SPI Data Register*, jeśli poprzedni transfer nie został zakończony. Bit jest zerowany po odczytaniu rejestru i ponownym dostępie do *SPI Data Register*
7	SPIF	Gdy bit jest jedynką, oznacza to, że transfer danych został zakończony. W trybie *master* bit jest ustawiany przy ostatnim cyklu przesyłania. W trybie *slave* SPIF jest ustawiany przy ostatnim zboczu próbkującym SPI – uwaga; nie jest to flaga przerwania. Bit jest zerowany po odczytaniu rejestru i ponownym dostępie do *SPI Data Register*

Rejestr danych *SPI Data Register (S0SPDR)*

Dane przeznaczone do wysłania są wpisywane do rejestru *SPI Data Register*. Rejestr może mieć długość 8 lub więcej bitów, zależnie od wcześniej zaprogramowanej długości słowa danych (maksymalnie 16 bitów). Po zapisaniu rejestru dane są natychmiast przesyłane do rejestru przesuwającego modułu SPI (dane nie są buforowane). Dane wpisane do *SPI Data Register* zostaną wysłane tylko wtedy, gdy zostanie zakończone przesyłanie poprzedniej danej (jest ustawiony bit SPIF). Informacja o zakończeniu przesyłania danych i o ewentualnych błędach zapisywania danych jest umieszczona w rejestrze *SPI Status Register*. Dane przeznaczone do odczytu są buforowane (bufor o długości 1 słowa).

Rejestr *SPI Clock Counter Register*

Wartość wpisana do 8-bitowego rejestru *SPI Clock Counter Register* musi być parzysta, większa lub równa 8 i określa częstotliwość zegara na linii SCK.

Ośmiobitowy licznik załadowany zawartością *SPI Counter Register* zlicza impulsy o częstotliwości PCLK.

1.4. Tryby pracy interfejsu

Tryb *master*

Programowanie trybu *master* trzeba zacząć od zaprogramowania licznika *SPI Counter Register* (określenie prędkości transmisji). W rejestrze *SPI Control Register* musi być ustawiony bit MSTR oraz określony format transmisji: długość słowa danych (bity *BitEnable* i BITS), kolejność wysyłania bitów (bit LBSF) oraz zależność pomiędzy danymi i fazą sygnału zegarowego (bity CPHA i CPOL).

Zapisanie danych do rejestru *SPI Data Register* rozpoczyna transmisję. Po zakończeniu przesyłania danych jest ustawiany bit SPIF w rejestrze *SPI Status Register*. Można też wtedy odczytać dane odebrane z *SPI Data Register* (jeżeli są do odczytania).

Bit SPIF jest zerowany po odczytaniu rejestru *SPI Status Register* i wpisaniu nowej danej do *SPI Data Register*.

Tryb *slave*

Moduł SPI pracuje w trybie *slave* po wyzerowaniu bitu MSTR w rejestrze *SPI Control Register*. W tym trybie, podobnie jak w trybie *master*, trzeba ustawić format przesyłanych danych oraz zależność pomiędzy danymi i fazą sygnału zegarowego.

Dane przeznaczone do przesłania do układu *master* są wpisywane do *SPI Data Register*. Wpisywanie jest możliwe przy wyzerowanym bicie SPIF. Dane są gotowe do odczytania, kiedy bit SPIF zostanie ustawiony.

1.5. Błędy zgłaszane przez interfejs SPI

Błąd *Read Overrun*

Błąd ten występuje, kiedy wewnętrzny bufor odebranych danych nie zostanie odczytany, a moduł SPI odbierze kolejną daną. Dana z bufora zostanie nadpisana odebraną daną i zostanie ustawiony bit ROVR w rejestrze statusu.

Błąd *Write Collision*

Ponieważ w czasie zapisu danych do przesłania nie ma bufora, to nie można wysyłać danych do rejestru *SPI Data Register*, kiedy trwa wysyłanie poprzedniej danej. Zapisanie rejestru *SPI Data Register*, gdy poprzednie dane są przesyłane, powoduje, że zapisywane dane są tracone i jest ustawiany bit kolizji zapisu WCOL.

Błąd *Mode Fault*

Błąd *Mode Fault* występuje, jeśli moduł SPI jest ustawiony w trybie *master*, a linia P0.7 zaprogramowana jako SSEL0 jest w stanie niskim. Błąd jest sygnalizowany przez ustawienie bitu MODF. Linie interfejsu SPI przechodzą w stan nieaktywny, a moduł zaczyna pracować w trybie *slave*.

Błąd *Slave Abort*

Kiedy moduł SPI w trybie *slave* jest w trakcie przesyłania danych i sygnał SSEL przejdzie w nieaktywny stan wysoki, to przesyłana (nadawana i odbierana) dana jest gubiona i w rejestrze *SPI Status Register* zostanie ustawiony bit ABRT.