Przegląd architektury ADSP-21161

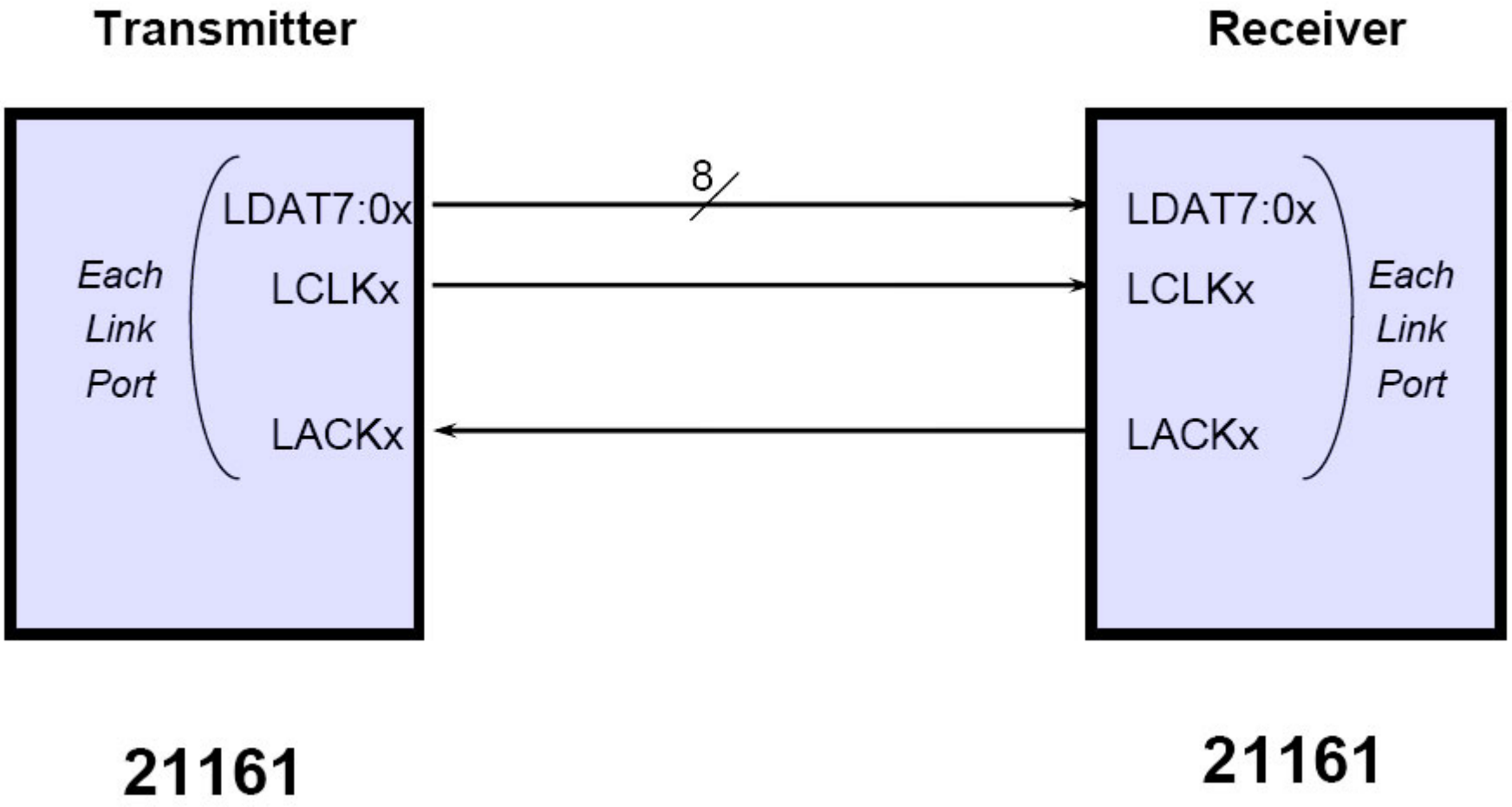
Sekcja 11

Porty łączności ADSP-21161

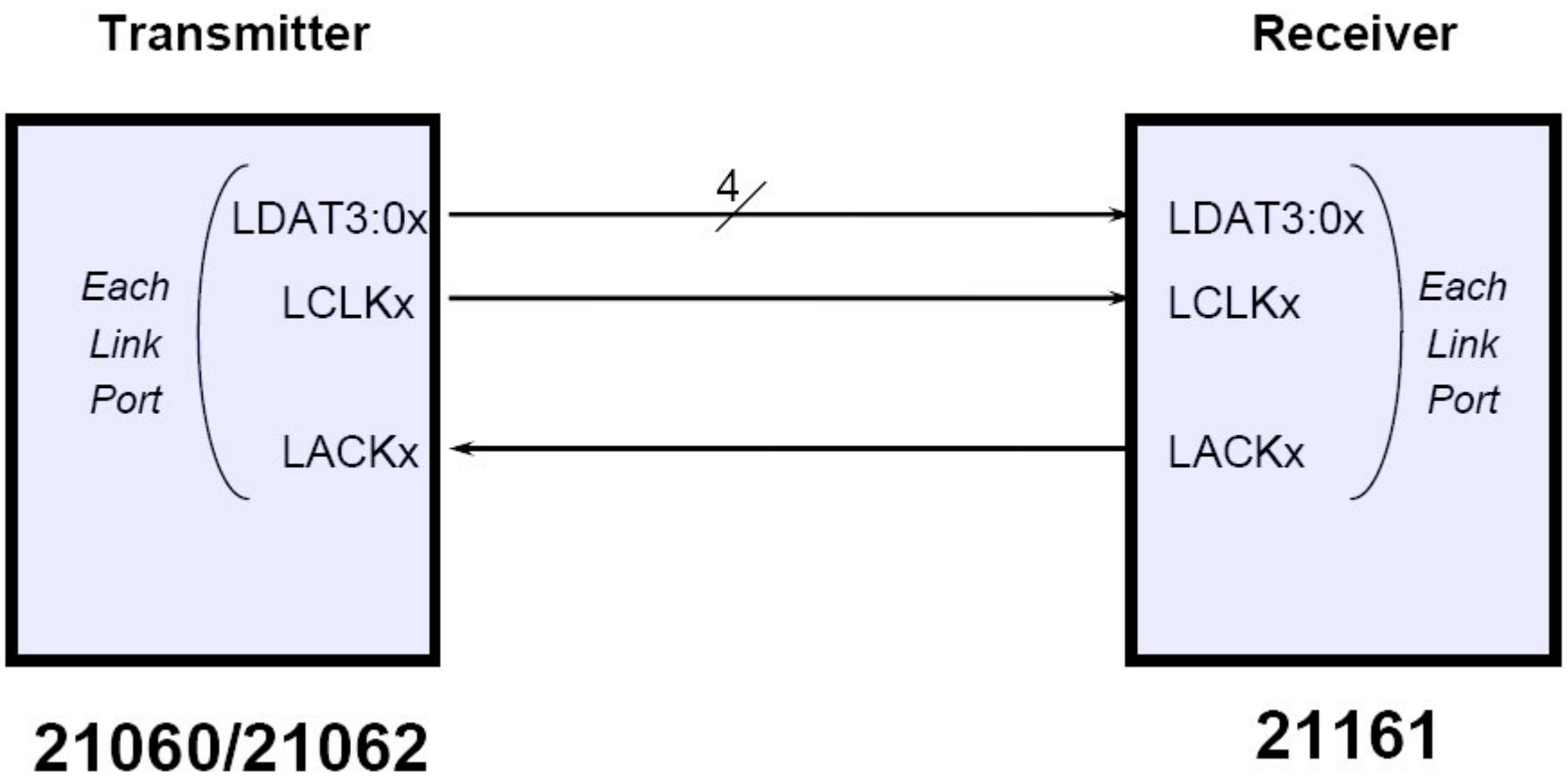
Porty łączności

- Szybkie porty równoległe do komunikacji SHARC-SHARC
- Dwa niezależne porty do komunikacji międzyprocesorowej punkt-punkt
- Obsługuje 8/4bitowe ścieżki danych 21161-21161 lub 21161-21060/21062
- Porty łączności działają niezależnie i jednocześnie
- Obsługuje handshaking dla procesorów DSP działających z różnymi szybkościami zegara
- porty można ustawić na nadawanie lub odbiór
- Podwójnie buforowane rejestry transmisji i odbioru danych
- kanały DMA do wewnętrznej 32 lub 48bitowej amięci
- Przepustowość przy zegarze 100MHz
  200 MB/s (32/48bitowe słowa)

Porty łączności ADSP-21161

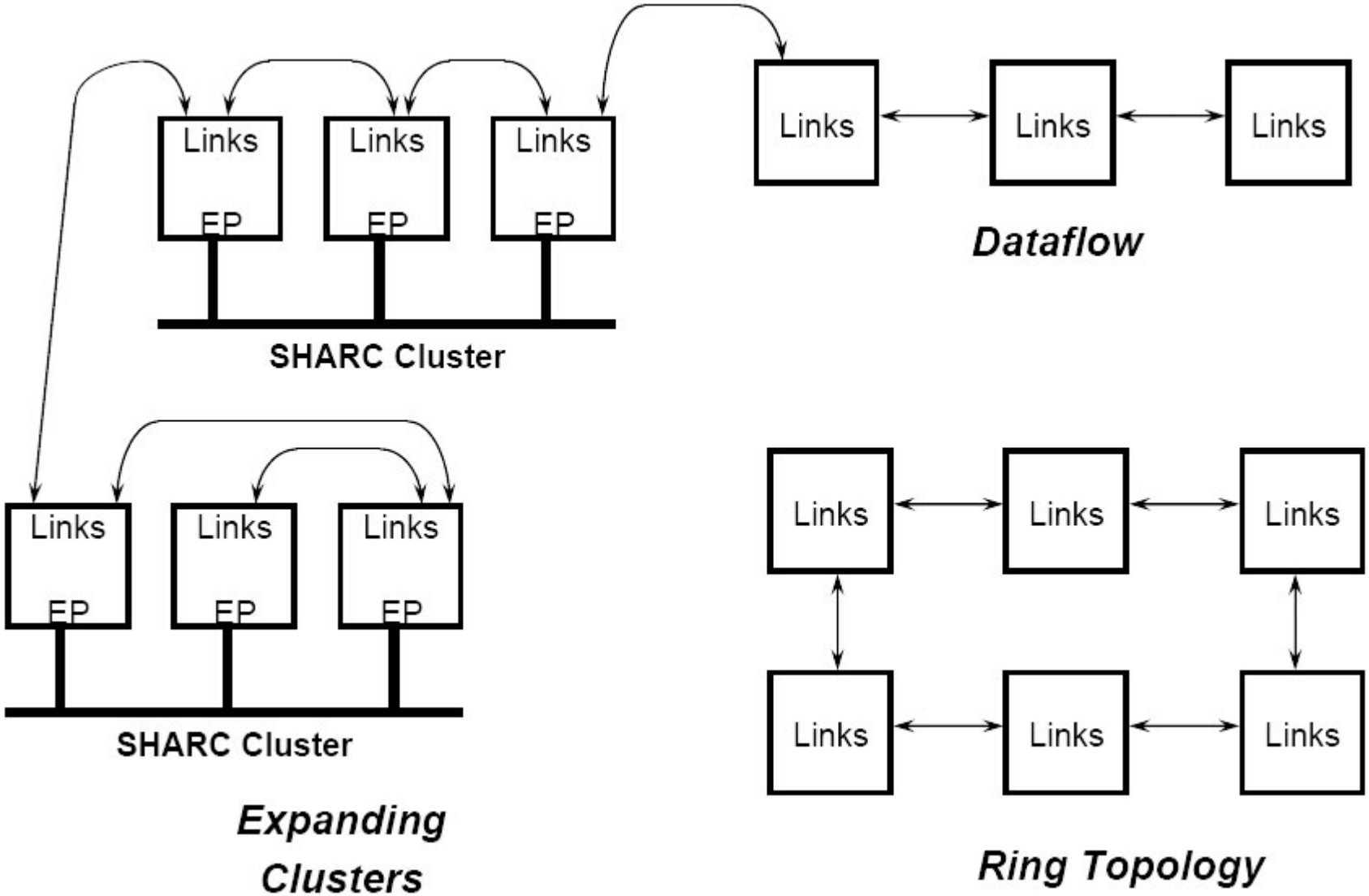


Porty łączności ADSP-21161

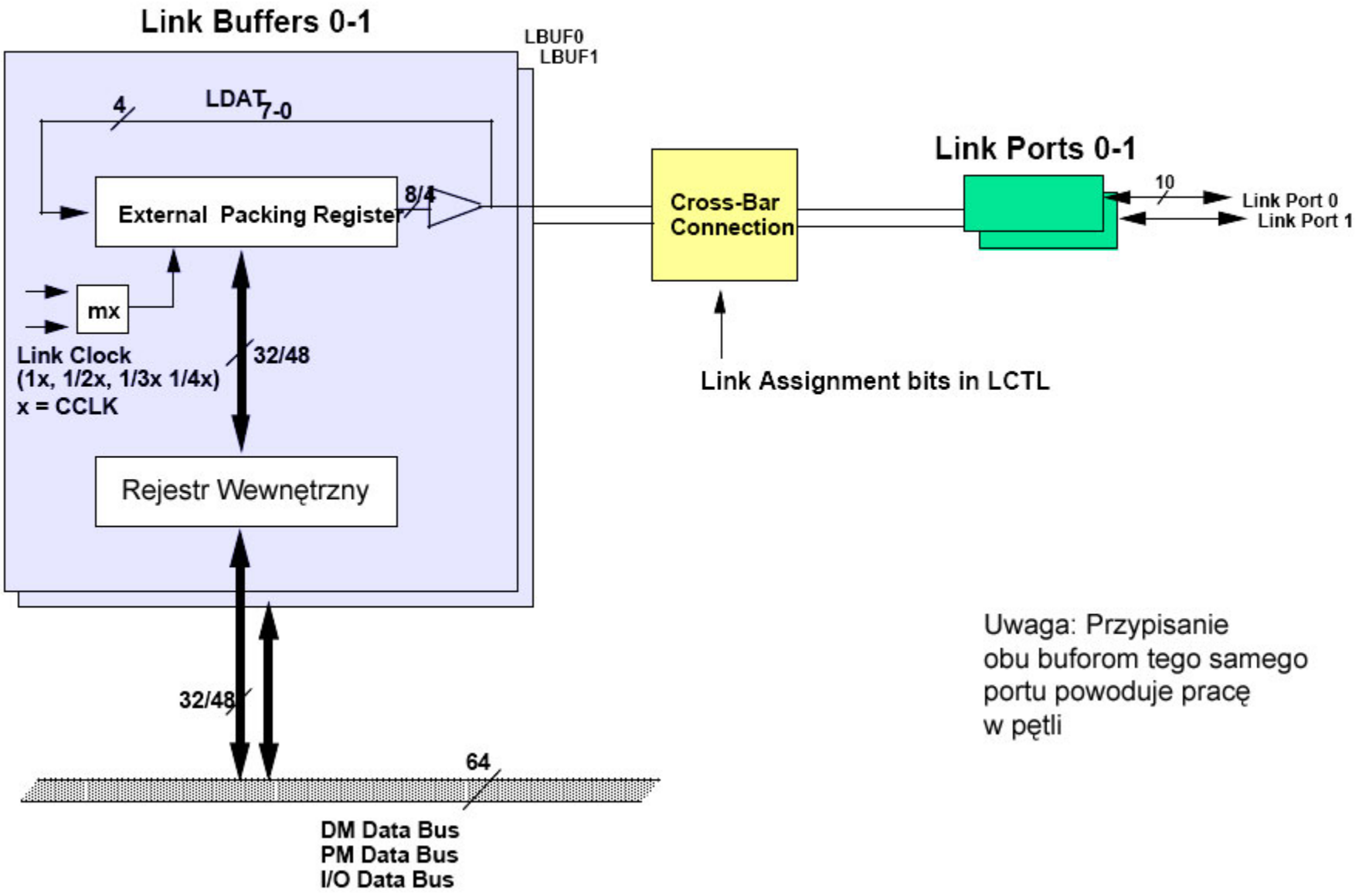


Bit L1DPWID rejestru LCTL ustala szerokość ścieżki danych

Przykład


Links
Links
Links
EP
EP
EP
SHARC Cluster
Links
Links
Links
Dataflow
Links
Links
Links
EP
EP
EP
SHARC Cluster
Expanding
Clusters
Links
Links
Links
Links
Links
Links
Ring Topology

Porty łączności ADSP-21161



Uwaga: Przypisanie obu buforom tego samego portu powoduje pracę w pętli

Dlaczego porty pozwalają na komunikację na dużą odległość?

- **Łącza same się synchronizują**
  - **zegar i dane są przesyłane jednocześnie**
  - **dane są wyprowadzane wraz z narastającym zboczem zegara i zatrzaskiwane z opadającym**

Działanie LACK

- **LACK zabrania transmisji następnego słowa, a nie aktualnego bajtu, więc zawsze można skończyć transmisję aktualnego słowa**

Wysterowanie linii transmisyjnej

- **linki są przystosowane do linii transmisyjnej o impedancji 50ohm, (lub większej),**
- **reguła kciuka: 1 ns opóźnienia propagacji na stopę**

Rejestry kontrolne

LCTL –Link Port Buffer Control Register

• Zwiera bity kontrolne do włączania lub ustawiania buforów, DMA, łańcuchowania, kierunku transmisji, wielkości słowa, szybkości transmisji, wewnętrznych pull-downów, i szerokości danych.

•Przypisuje port łączności buforowi łączności (jak LAR w 21160)

•Zawiera bity statusu, status pakowania i status błędu pakowania (jak w LCOM w 21160)

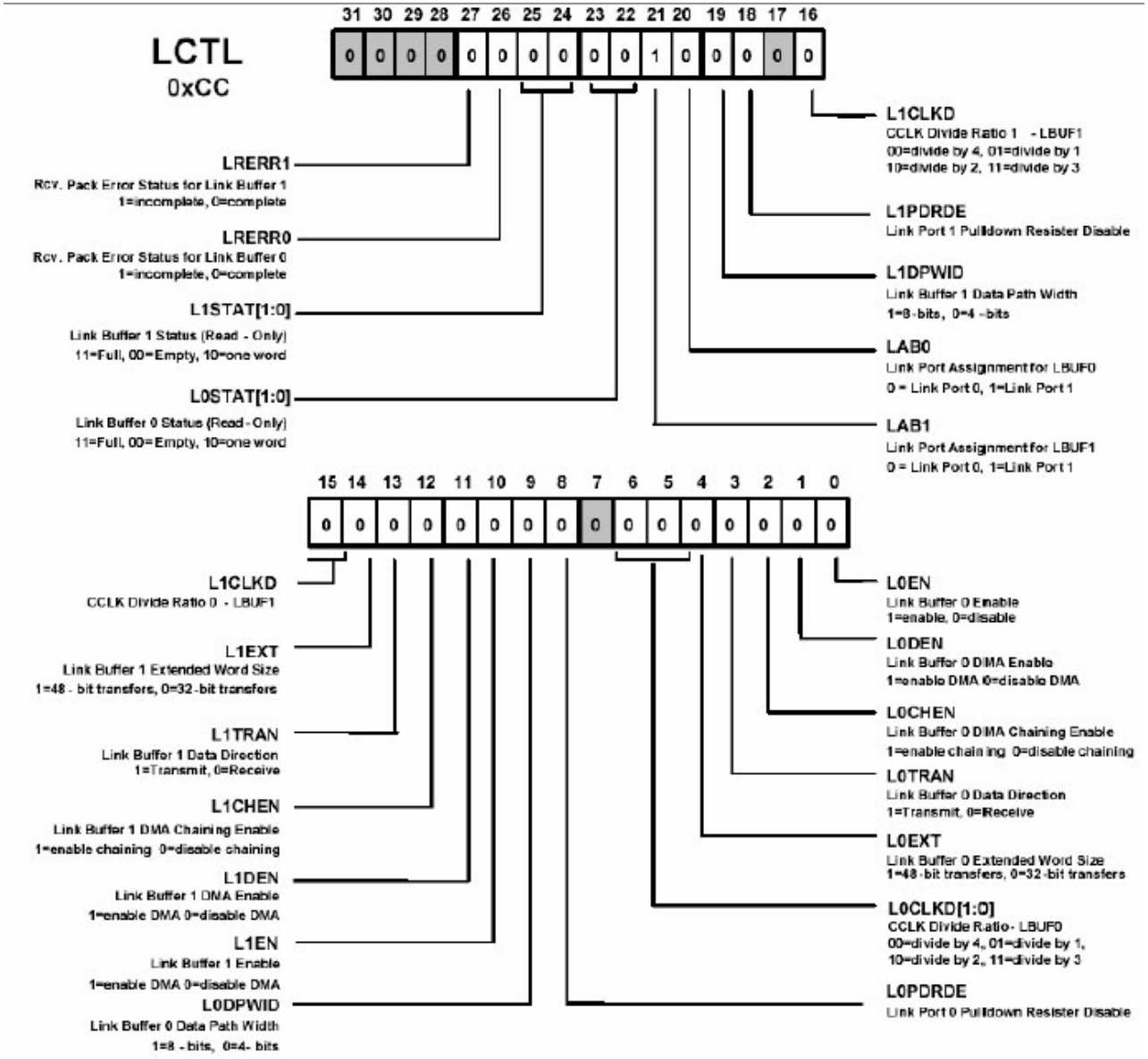
LSRQ - Link Port Service Request & Mask Register

− Przesyła i odbiera statusy i żądania maskownia

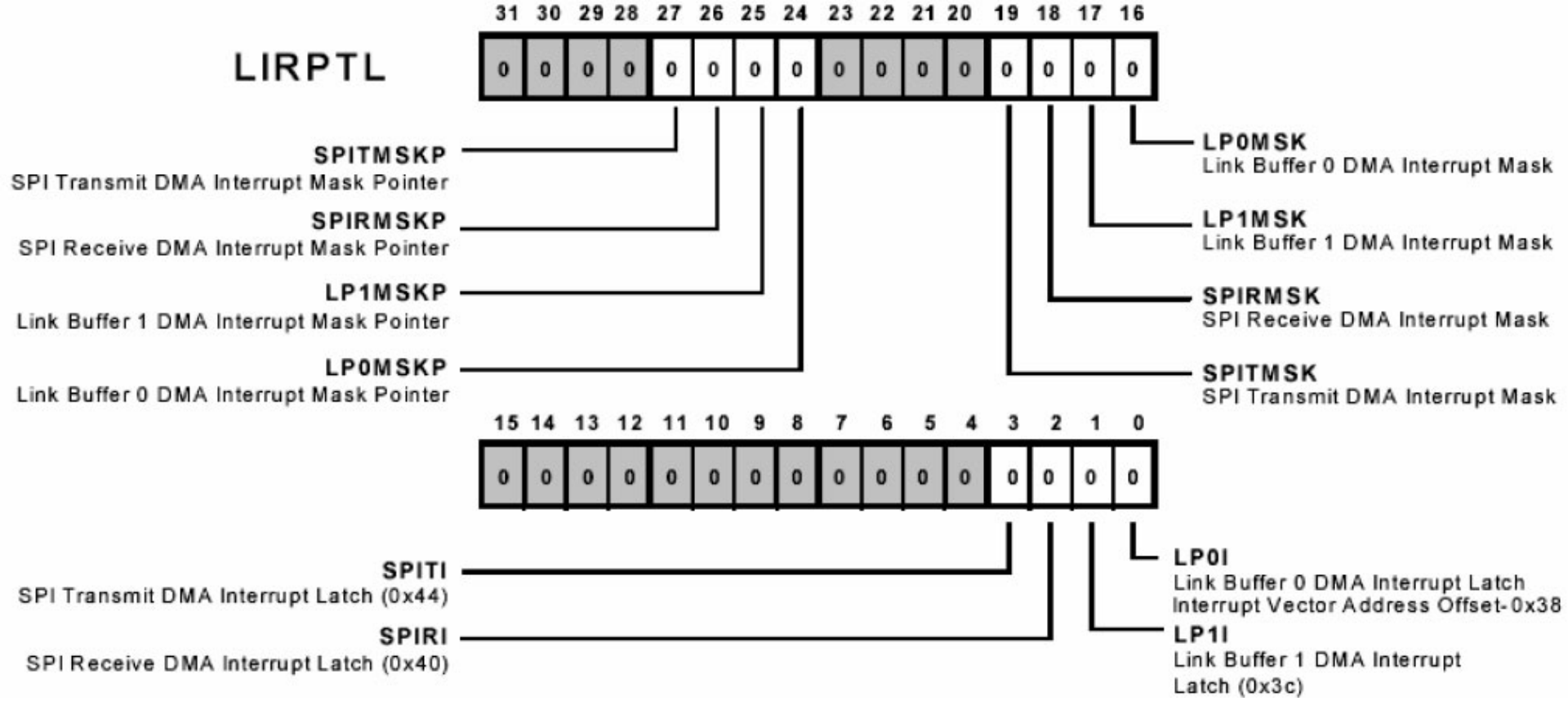
• **LIRPTL - Link Interrupt Latch Register**

− Wskazuje na status zatrzasku, wybiera maskowanie i wyświetla wskaźniki masek dla portów łączności

Porty łączności ADSP-21161



Rejestr LIRPTL



Przerwania portów łączności

- **Przerwania DMA**

- niełańcuchowe - kiedy kończy się transfer bloku DMA, występuje przerwanie
- łańcuchowe – można wybrać, które sygnały zakończenia transferu generują przerwanie

- **Przerwania sterowane rdzeniowo**

- jeśli bufor jest włączony, a DMA nie, to przerwanie następuje, gdy bufor nie jest pusty przed odbieraniem albo nie jest zapełniony przed nadawaniem (maskowalne w LIRPTL)
- bufor połączenia może być dostępny jako pamięciowo mapowany rejestr danych I/O
- można usunąć oczekujące przerwanie i zdjąć z niego maskę w tym samym cyklu bez jego obsługi
- w celu obsłużenia przerwania należy zdjąć z niego maskę w IMASKu
- ustawienie bitu LPISUMI w rejestrze IMASK
- aby wybrać port łączności należy ustawić bity w rejestrze LIRPTL

Przyporządkowanie bufora łączności do DMA

- Porty łączności współdzielą kanały 8 i 9 DMA z buforami nadawczymi i odbiorczymi SPI
  - kanał 8 jest współdzielony pomiędzy LBUF0 i SPIRX
  - kanał 9 jest współdzielony pomiędzy LBUF1 i SPITX

- Rejestr LIRPTL kontroluje funkcje zatrzaskiwania i maskowania przerwań nadawczych i odbiorczych portów łączności i SPI jednocześnie

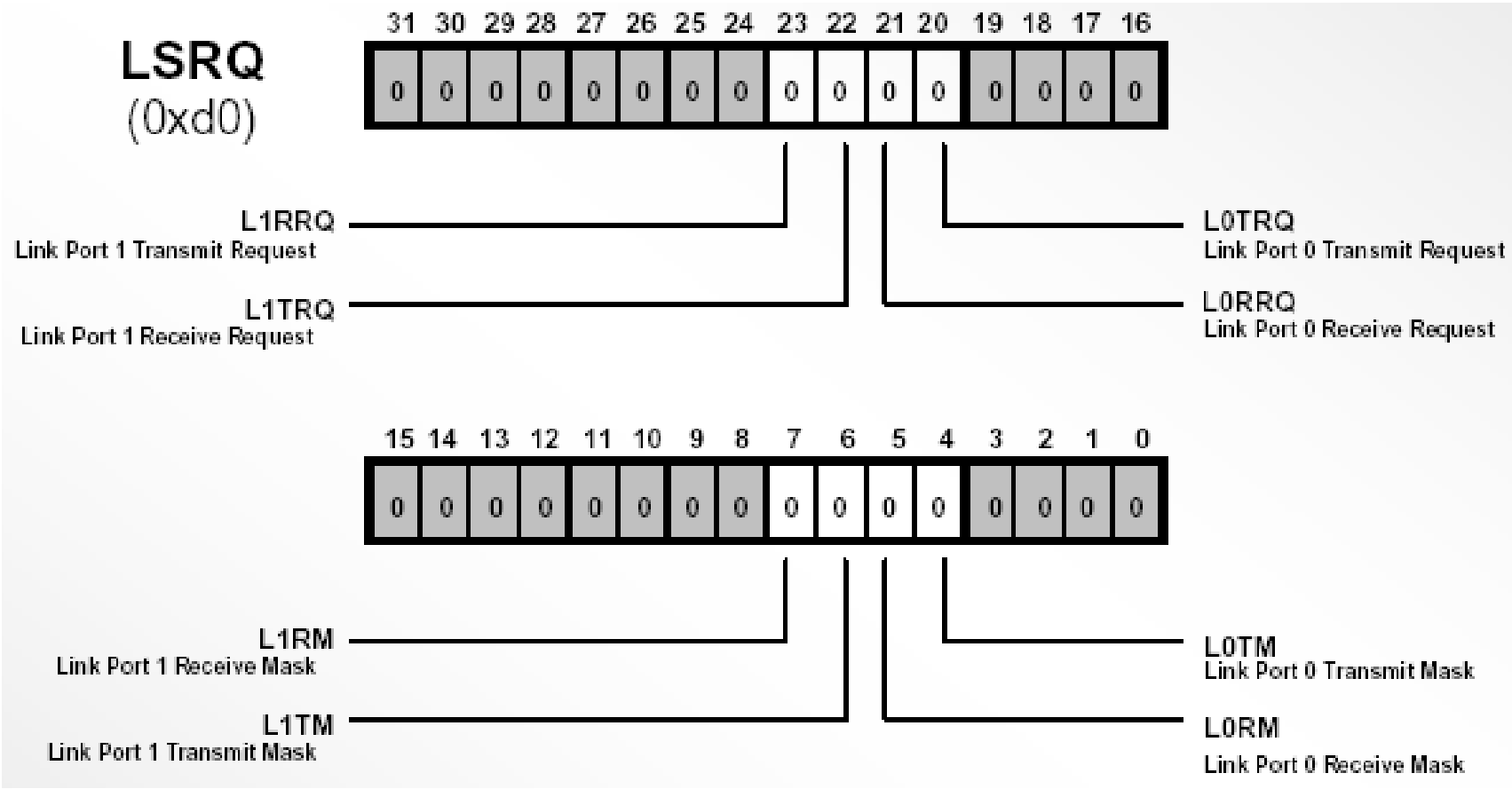
Przerwania żądania obsługi portów łączności

- Pozwalają dwóm procesorom na komunikowanie się bez wiedzy o kierunku transferu i porcie łączności, w którym transfer nastąpi
- Pozwalają na zażądanie przerwania przez nieaktywny port łączności, gdy nastąpi próba dostępu zewnętrznego
- Rejestr LSRQ jest mapowany pamięciowo:
  - LxTm Link Port Transmit Mask
  - LxRM Link Port Receive Mask
  - LxTRQ Link Port Transmit Request Status
  - LxRRQ Link Port Receive Request Status

LxTRQ=1 oznacza LxACK=1, LxTM=1 i LxEN=0

LxRRQ=1 oznacza LxCLK=1, LxRM=1 i LxEN=0

Rejestr żądań obsługi portów łączności



LxTRQ=1 oznacza LxACK=1, LxTM=1 i LxEN=0

LxRRQ=1 oznacza LxCLK=1, LxRM=1 i LxEN=0

ADSP-21161 Core-driven LINK Loopback Example:

```
#include "def21161.h"
.section/pm  pm_link;   /* Main code segment from  .ldf  file.*/
start:
    /* Link Assignment:LBUF0->port 0, LBUF1->port 0*/
    /* LCTL0 Register: 32-bit data, LBUF0=rx, LBUF1=tx, */
    /* LCLK divide ratio = ½ CCLK */
    ustat1=dm(LCTL);
    bit clr ustat1 L0TRAN | L0CLKD0 | LAB0 | LAB1| L1CLKD0;
    bit set ustat1 L1TRAN | L0CLKD1 | L1EN | L0EN |  L1CLKD1;
    dm(LCTL)=ustat1;
    nop;

    r0=0x11111111;   /* Test data to transmit.*/
    dm(LBUF1)=r0;    /* Write to LBUF1 to transmit.*/

polling:    ustat1=dm(LCTL);                    /* Wait for LBUF0 to have data   */
            bit tst ustat1 0x00C00000;
            if not tf jump polling;

    r1=dm(LBUF0);    /* Read, data is received */
    r0=0x22222222;   /* Test data to transmit.*/

wait:   jump wait;   /* Sits in loop when done */
```

Porty łączności, a zewnętrzne porty komunikacji

- **Porty łączności**
  - Pozwalają na równoczesną komunikację pomiędzy każdym SHARCiem, a dwoma innymi SHARCami poprzez dedykowane połączenia punkt-punkt
  - Komunikacja odbywa się z prędkością 100MB/s
- **Porty zewnętrzne (EP)**
  - Prędkość
    - Komunikacja poprzez EP zapewnia szersze pasmo pomiędzy dwoma SHARCami (400MB/s zakładając 100MHz-owy zegar)
  - EP to pojedyncze połączenie dla 6 SHARCów i pamięci zewnętrznej
    - Dwa SHARCe komunikują się w każdym cyklu
  - EP oferuje elastyczną wymianę danych i informacji kontrolnych
  - Model zasobów współdzielonych pozwala na prostszą strukturę programu

Zarówno porty łączności jak i EP są używane w wielu systemach

Interfejsy szeregowe ADSP-21161: SPORTy i SPI

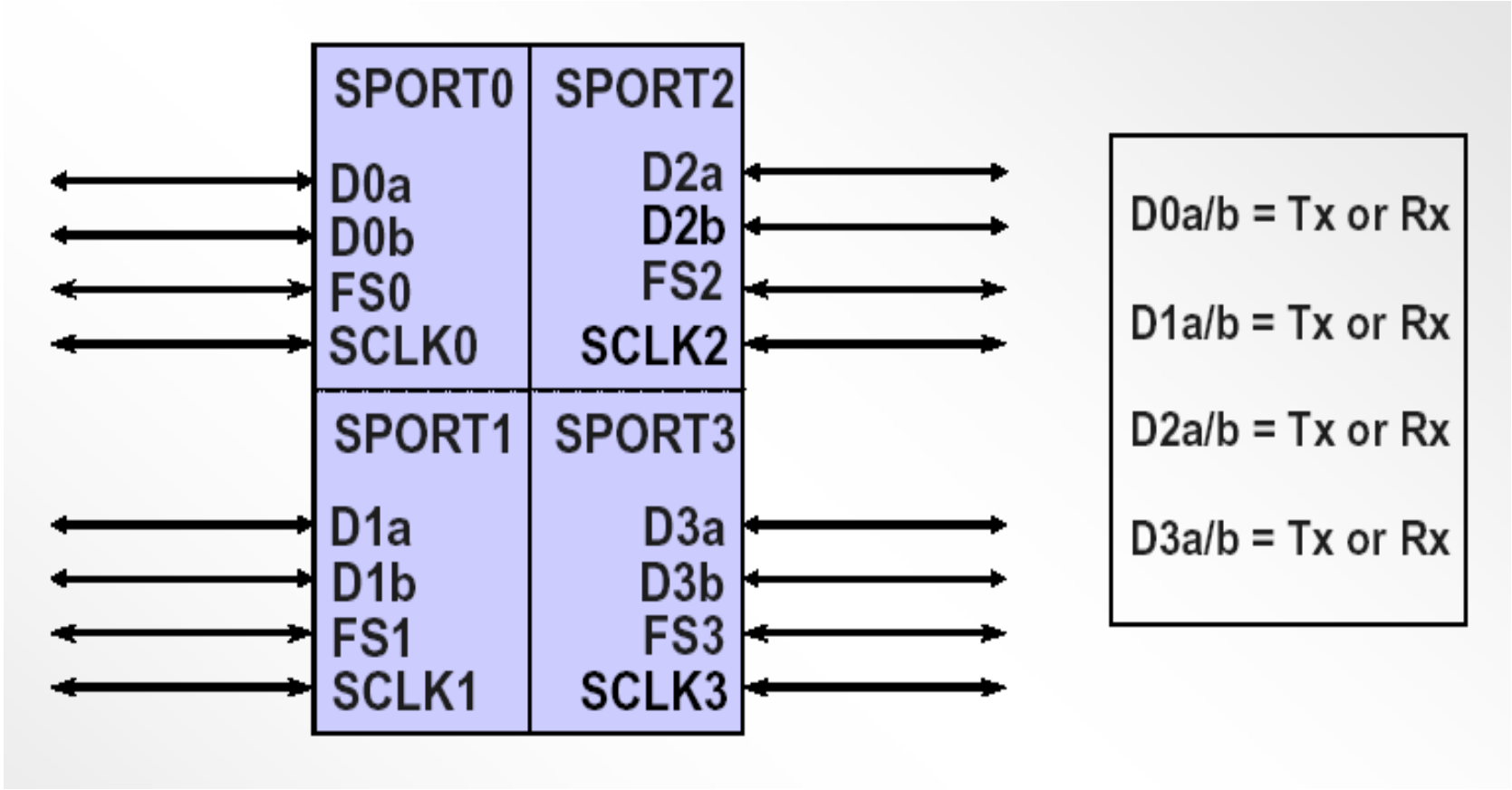
Interfejsy szeregowe ADSP-21161

- 4 niezależne, synchroniczne porty szeregowe (SPORTy) half-duplex
- Działa do połowy częstotliwości zegara
- SPORT posiada 2 linie danych, 1 zegarową i 1 synchronizacji ramek
- Kanały A i B danych szeregowych są konfigurowalne jako odbiorniki (wyjścia) lub nadajniki (wejścia)
- Tryb I2S jest wspierany we wszystkich 4 SPORTach
- Wsparcie TDM dla 128 kanałów na ramkę (H.100/H.110)
  - Dwa SPORTy muszą być połączone dla TDM
- Kompansja A-law/u-law wspierana w każdym SPORT'ie
  - 2 podstawowe porty kanału A (SP0 i SP1) są używane przy rozszerzaniu (expansion) , pozostałe 2 podstawowe porty (SP2 i SP3) są zdolne tylko do kompresji.
- Port kompatybilny z SPI
  - Interfejs dla hostowania mikrokontrolerów i urządzeń zewnętrznych
  - Możliwe bootowanie poprzez mikrokontroler będący hostem

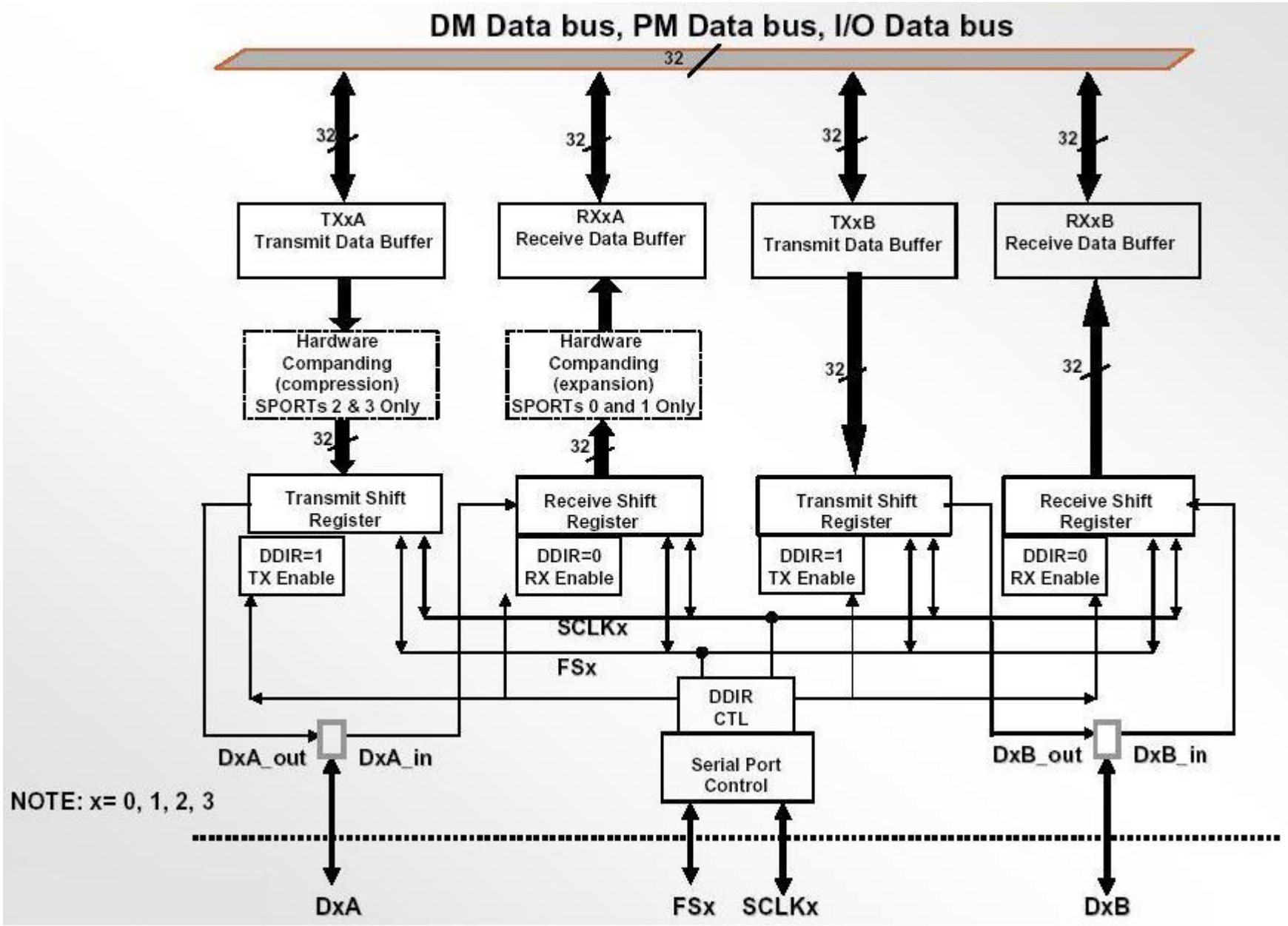
Możliwości portów szeregowych ADSP-21161

- Sterowane przerwaniami przesyłanie pojedynczych słów do/z pamięci on-chip kontrolowane przez rdzeń ADSP-21161
- Przesyłanie bloków słów kontrolowane do/z pamięci on-chip kontrolowane przez kontroler DMA
- Konfigurowalne:
  - Przesyłanie danych o długości od 3 do 32 bitów
  - Pierwszy bit MSB lub LSB
  - Zewnętrzne lub wewnętrzne źródło sygnału zegarowego i synchronizacji ramek
  - Wczesna synchronizacja ramki
  - Późna synchronizacja ramki
  - Bez synchronizacji
  - Tryb wielokanałowy (1-128 kan.) TDM (time division multiplexed)
  - Kierunek przesyłania danych dla kanałów A i B jest programowalny jako nadawczy lub odbiorczy

ADSP-21161 SHARC – SPORTy: 4 piny/SPORT

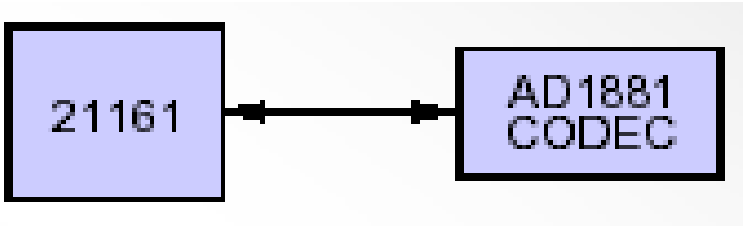


ADSP-21161 SPORT - schemat blokowy

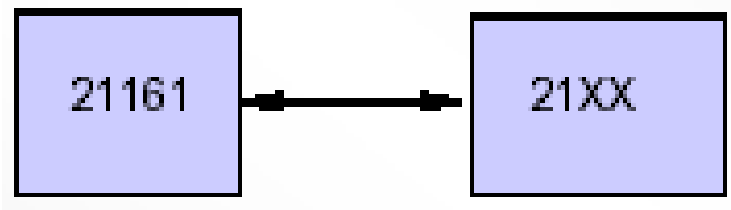


Przykłady zastosowania portów szeregowych

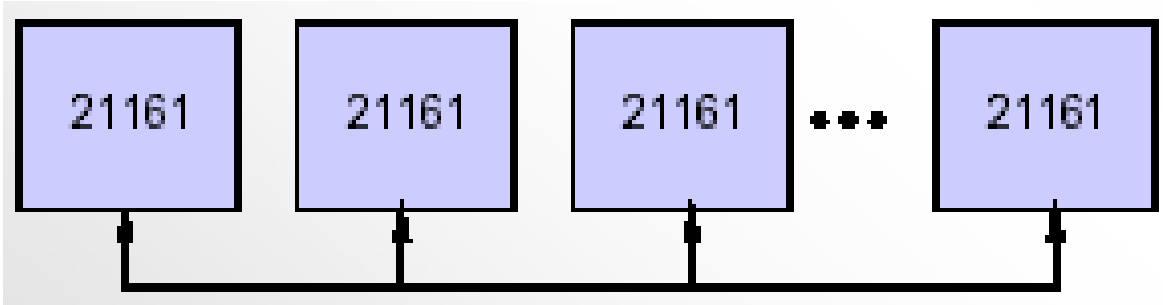
- Połączenie CODECa z portem szeregowym



- Połączenie dwóch DSP ze sobą



- Użycie portów szeregowych w komunikacji międzyprocesorowej



SPORT - rejestry IOP

- Szeregowy port danych dostępny poprzez mapowane pamięciowo rejestry danych:
  - TXxA, TXxB, RXxA, RXxB, gdzie x = 0, 1, 2, 3
- Konfiguracja 4 SPORTów przez mapowane pamięciowo rejestry kontrolne:

SPCTLx - SPORTx Control Register
DIVx - Clock and Frame Sync Divisors
CNTx - Count Register
MTxCSn - Multichannel Transmit Select
MTxCCSn - Multichannel Transmit Compand Select
MRnCSn - Multichannel Receive Select
MRnCCSn - Multichannel Receive Compand Select

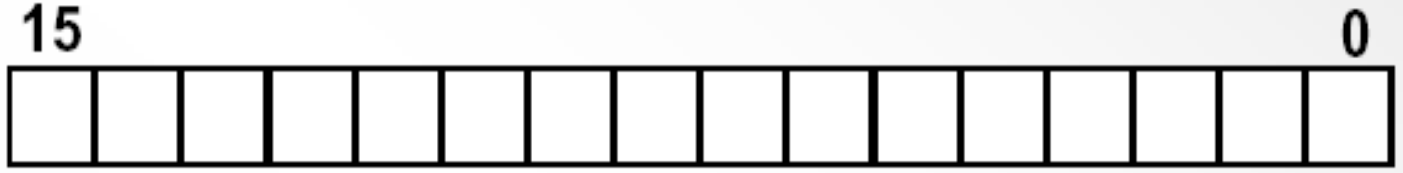
y = 0, 2 x = 1, 3 n = 0, 1, 2, 3

SP02MCTL, SP13MCTL – Multichannel Control Registers

CLKDIV: Serial Clock Divisor

DM(0x01C5,0x01D5,0x01E5,0x01F5)

CLKDIV (lower 16 bits of DIV0, DIV1, DIV2, DIV3)



- Używa się dla wewnętrznie wygenerowanych sygnałów zegarowych
- CLKDIV = (Core Clock/2(serial clock frequency)) - 1

- Przykład:

  Jeśli Core Clock pracuje na częstotliwości 100MHz, jaki CLKDIV jest wymagany dla uzyskania częstotliwości SCLK = 10MHz?

ODPOWIEDŹ:

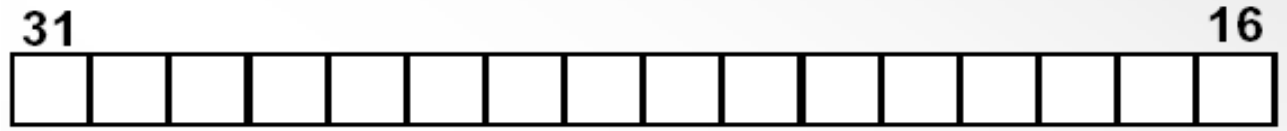
$$CLKDIV = \frac{100\ MHz}{2*10MHz} - 1$$

$$\Rightarrow \quad CLKDIV = 4$$

FSDIV: Frame Sync Divisor

DM(0x01C5,0x01D5,0x01E5,0x01F5)

FSDIV (upper 16 bits of DIV0, DIV1, DIV2, DIV3)



- Używa się dla wewnętrznie wygenerowanego sygnału synchronizacji odbioru ramki
- Liczba cykli SCLK pomiędzy wprowadzeniami FS = FSDIV + 1
- Używa się do zaprogramowania okresowych przerwań odbioru

FSDIV = (CLK/współczynnik próbkowania) - 1

- Przykład:

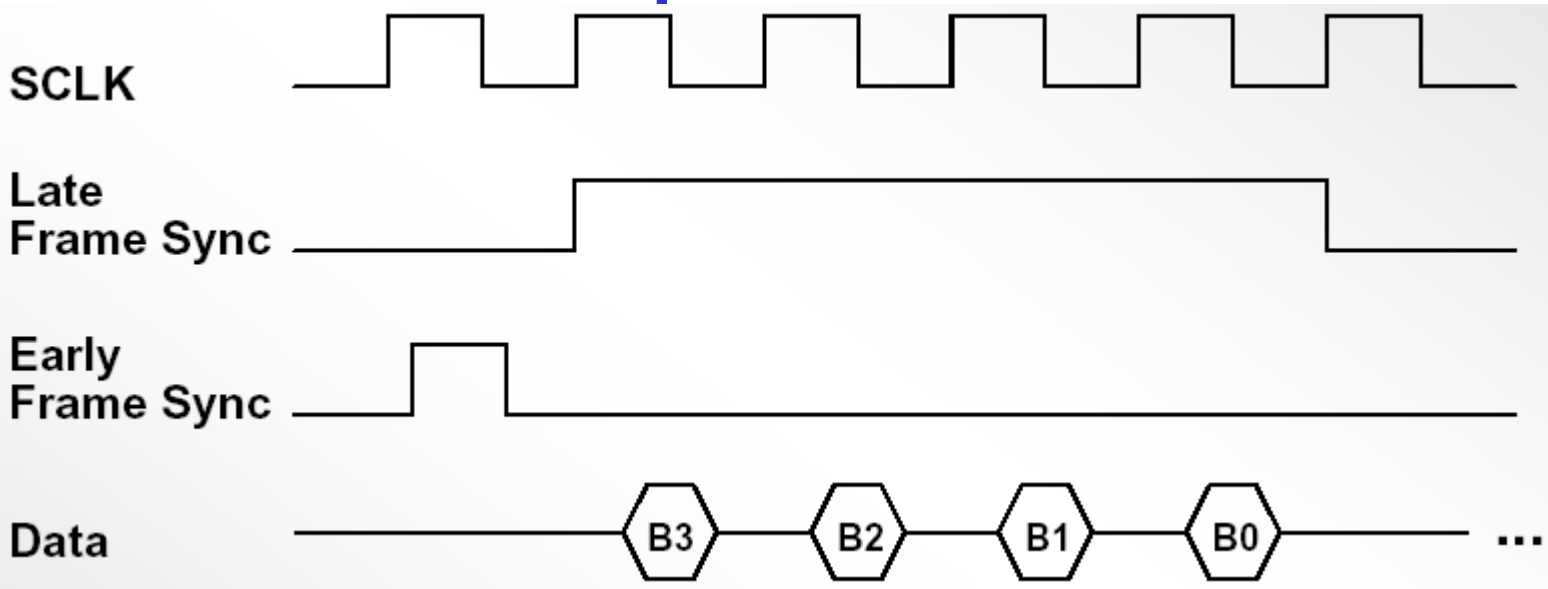
Jeśli SCLK = 10MHz, jaki FSDIV jest wymagany dla częstotliwości FS = 20kHz

ODPOWIEDŹ:

$$FSDIV = \frac{10\ MHz}{20\ KHz} - 1$$

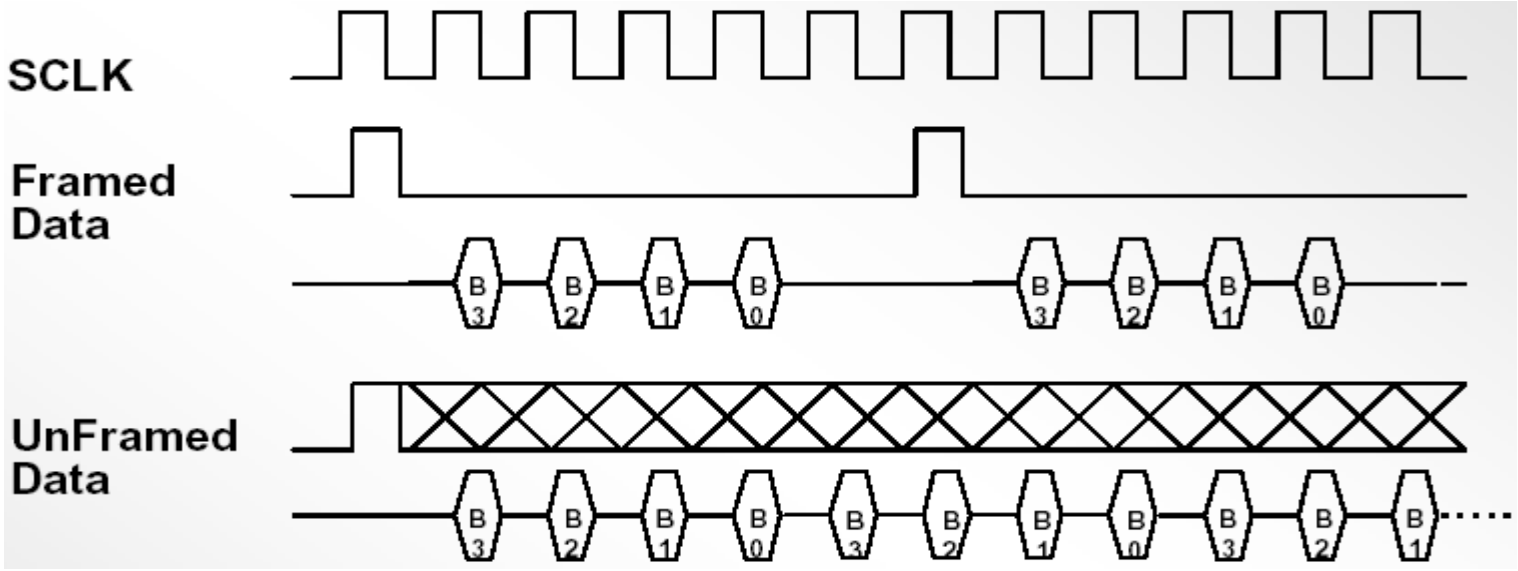
FSDIV = 499

Charakterystyki czasowe portów szeregowych: Wczesne i późne ramkowanie



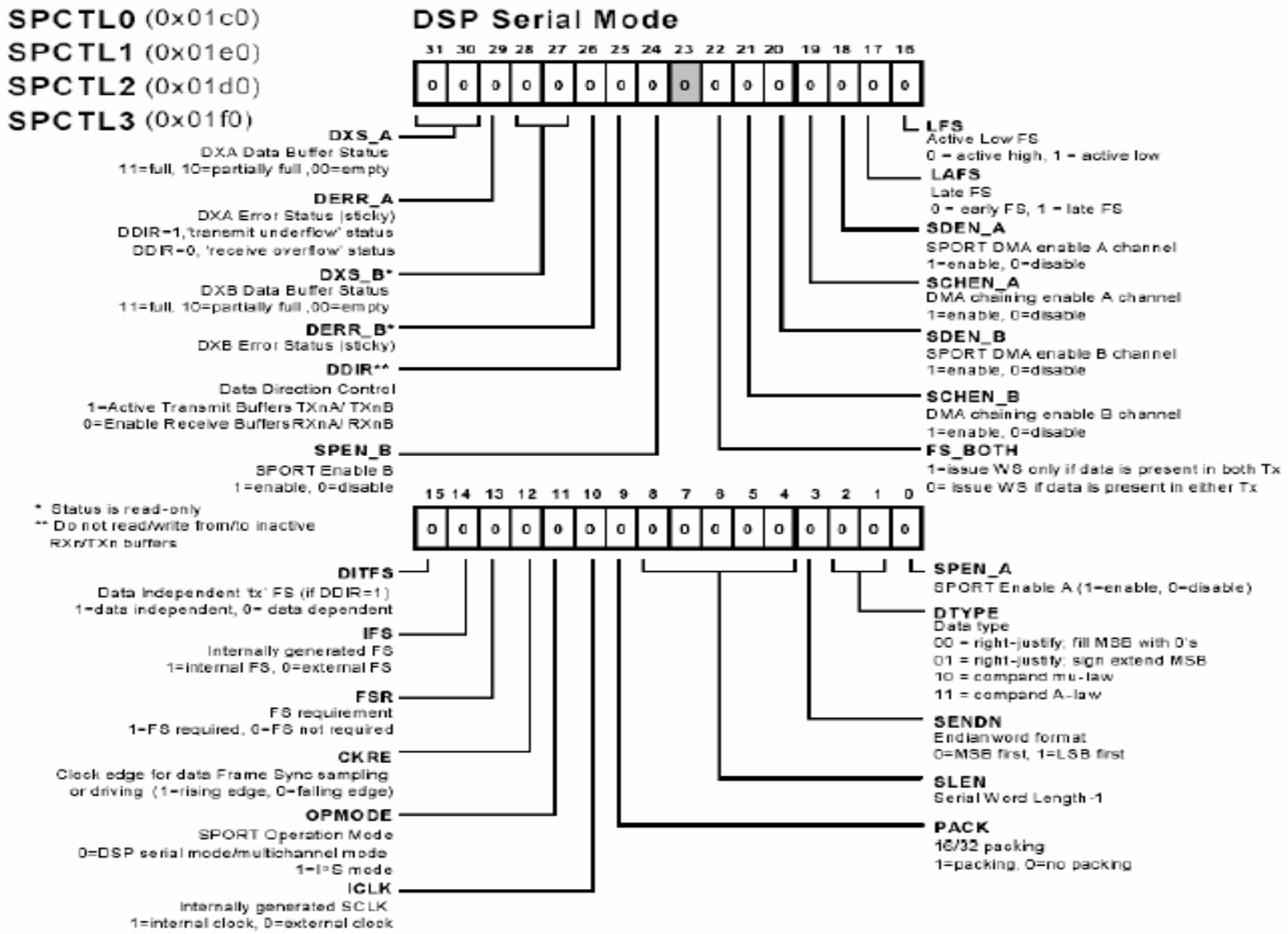
- LAFS (bit w rejestrze kontrolnym SPCTLx) = 0 dla wczesnej synchronizacji ramek, LAFS = 1 dla późnej synchronizacji ramek
- Wczesne ramkowanie: synchronizacja ramki wyprzedza dane o 1 cykl
  Późne ramkowanie: synchronizacja ramki sprawdzana tylko dla 1go bitu
- Pierwszy bit przesyłanych danych to MSB (SENDN=0) lub LSB (SENDN=1)
- Synchronizacja ramki i SCLK generowane wewnętrznie lub zewnętrznie

Charakterystyki czasowe portów szeregowych: Dane ramkowane lub nieramkowane

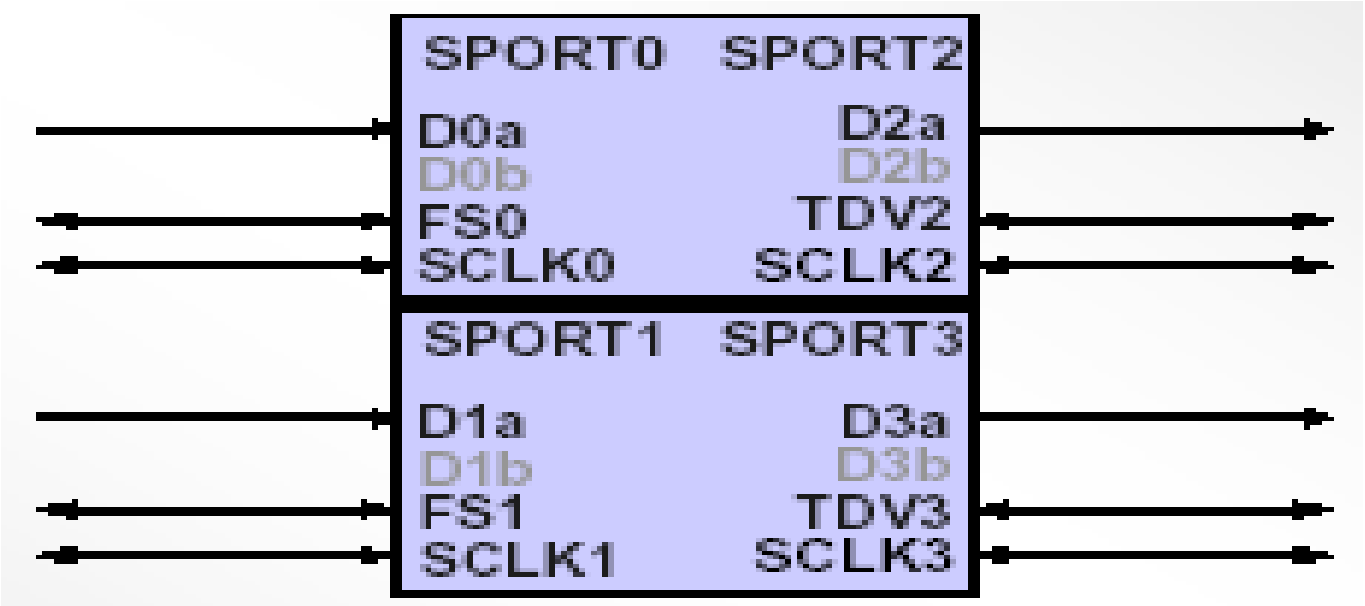


- FSR (w rejestrze kontrolnym SPCTLx) określa tryb ramkowany lub nieramkowany
- Tryb ramkowany wymaga sygnału ramkowania dla każdego słowa. W trybie nieramkowanym sygnał ten jest ignorowany po pierwszym słowie
- Tryb nieramkowany jest odpowiedni dla ciągłego odbioru
- Stan aktywny niski lub wysoki synchronizacji ramki ustalany jest przy pomocy bitu LFS rejestru kontrolnego SPCTLx

SPORT - rejestry kontrolne SPTCLx (IOP)



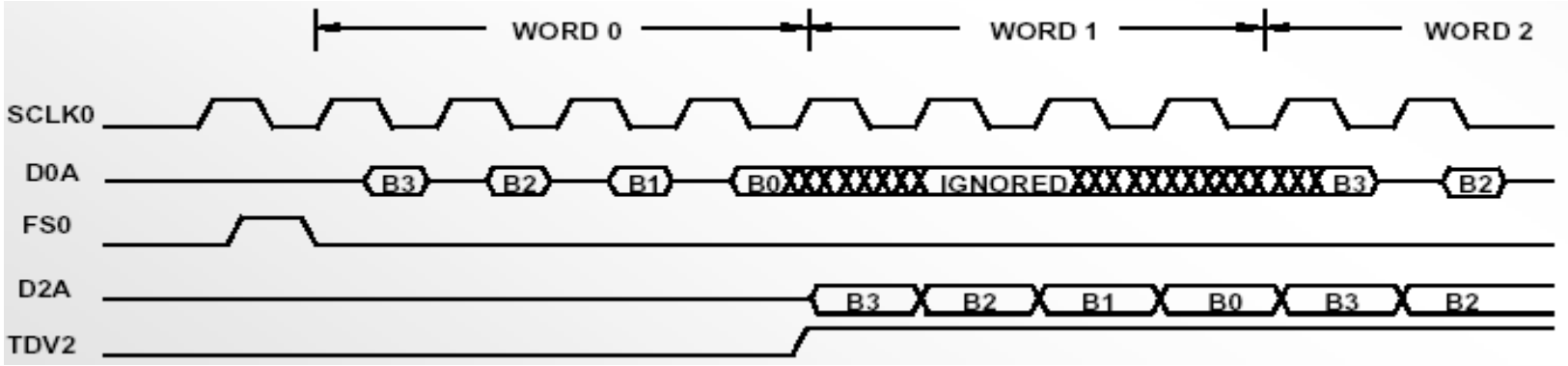
ADSP-21161 – praca wielokanałowa SPORTów



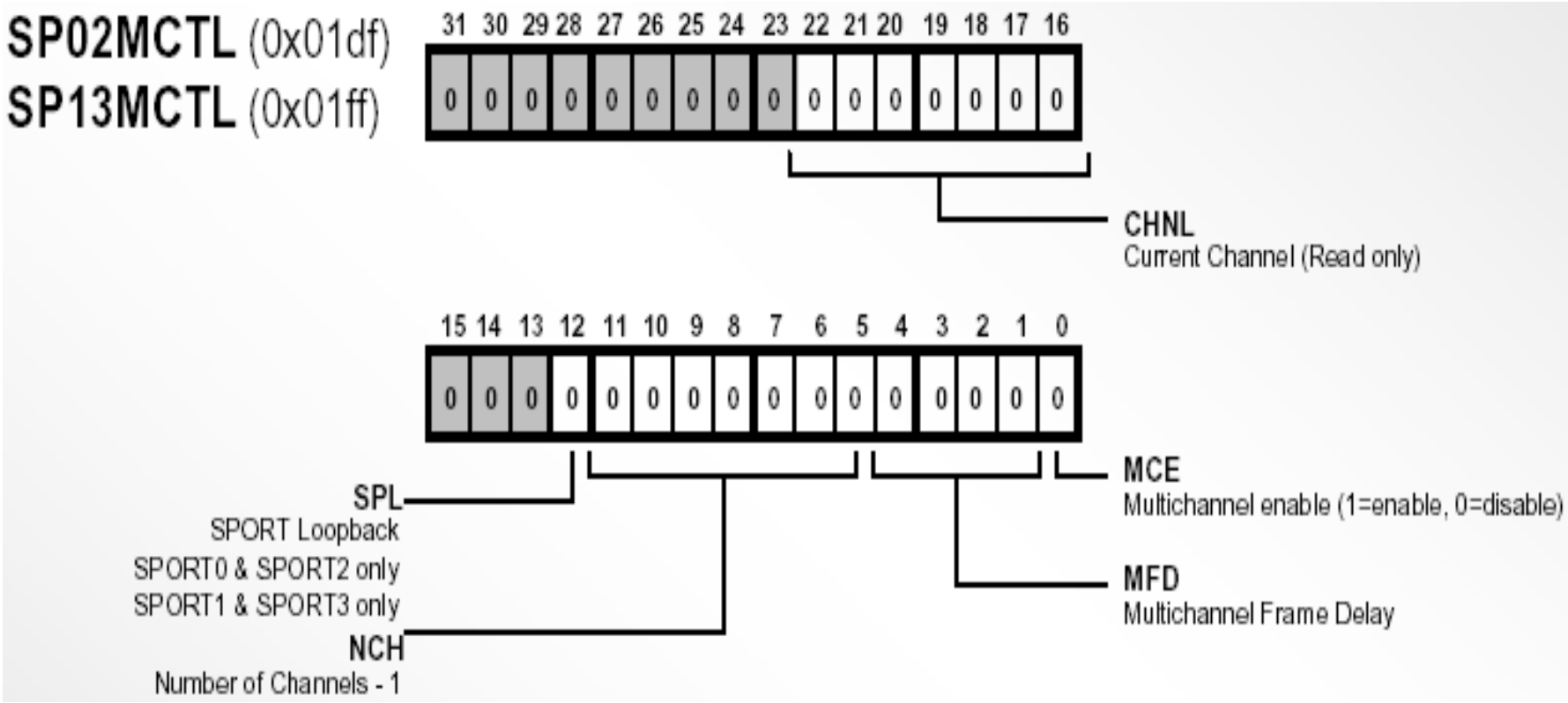
- Tryb pracy wielokanałowej
  - Wsparcie dla 128 kanałów TDM
  - Tylko piny kanału A są używane podczas pracy wielokanałowej. Kanał B jest wyłączony
  - SPORT0 i SPORT2 używane na jeden szeregowy port TDM
    - SPORT0 zawsze odbiera, a SPORT2 zawsze nadaje
  - SPORT1 i SPORT3 używane na jeden szeregowy port TDM
    - SPORT1 zawsze odbiera, a SPORT2 zawsze nadaje

Praca wielokanałowa

- Metoda TDM, gdzie dane są odbierane lub wysyłane różnymi kanałami współdzielącymi tę samą magistralę szeregową
- Liczba kanałow jest ustawiana przy pomocy bitów NCH rejestrów SP02MCTL albo SP13MCTL: NCH = (liczba kanałów do 128) – 1
- Niezależny wybór kanałów nadawczych i odbiorczych
- Tryb pracy wielokanałowej uaktywniany jest poprzez ustawienie bitu MCE = 1 w SP02MCTL/SP13MCTL
- FS0 lub FS1 sygnalizuje start ramki
- FS2 lub FS3 są używane jako TDV (Transfer Data Valid) dla logiki zewnętrznej. Aktywne tylko podczas przesyłania danych kanałami.
- Przykład: Odbiór kanałami 0 i 2 (SPORT0)
  Nadawanie kanałami 1 i 2 (SPORT2)



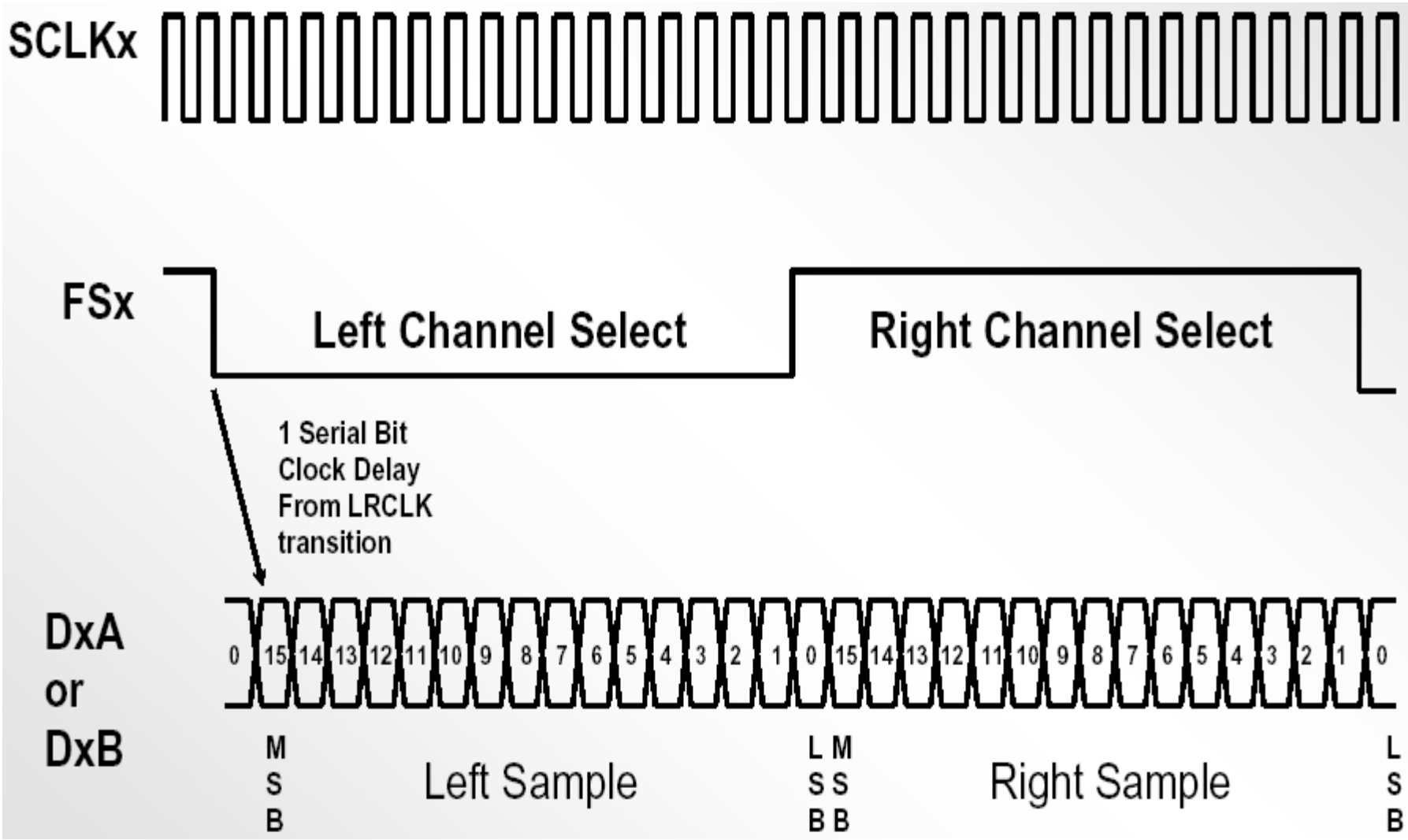
SP02MCTL, SP13MCTL
Rejestry kontroli wielokanałowej



ADSP-21161 SPORT – tryb pracy I2S

- Standard przemysłowy opracowany przez PHILIPSa dla transmisji audio poprzez 3-przewodowy interfejs
- Funkcjonalność I2S podobna do ADSP-21065L
- Dane przesyłane są zawsze w formacie MSB
- Możliwe transfery sterowane przerwaniami lub poprzez DMA
- Oba kanały SPORTx (DxA i DxB) zawsze przesyłają lub odbierają jednocześnie, każdy kieruje lub zatrzaskuje lewostronne i prawostronne dane I2S
- Zawiera Serial Clock, Word Select i Data
- Programowalność danych w 21161 SPORT (DDIR = tx albo rx) pozwala na
  - 8 nadajników I2S dla 16 wyjściowych kanałów audio
  - 8 odbiorników I2S dla 16 wejściowych kanałów audio
- Przykładowa konfiguracja:
  - Cyfrowy mixer 12 x 4 z 6 wejściowymi pinami I2S i 2 wyjściowymi
  - 1 SP/DIF RX, 2 konwertery stereo ADC i 5 konwerterów stereo DAC na potrzeby kina domowego lub samochodowych systemów audio

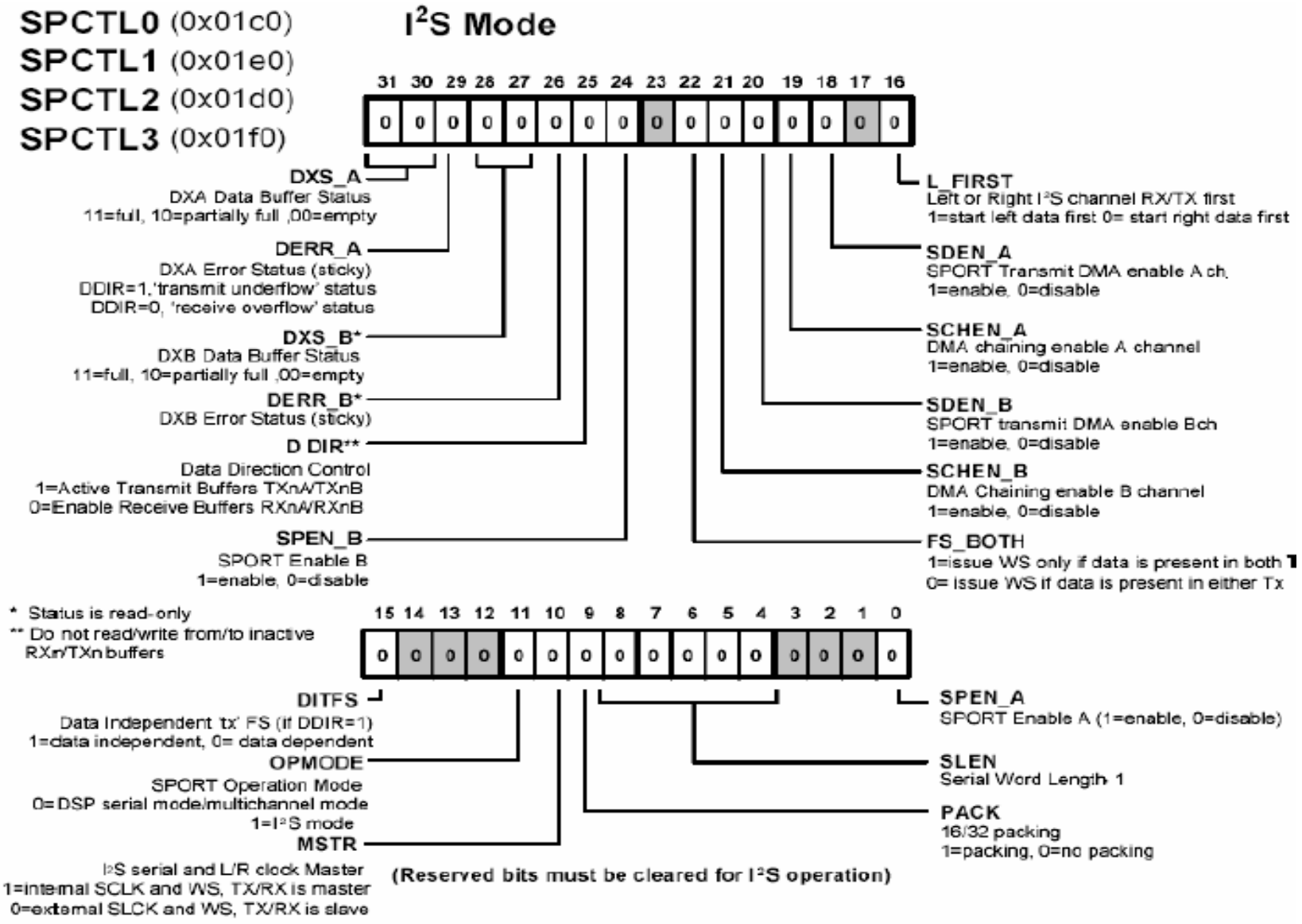
ADSP-21161 szeregowy protokół I²S



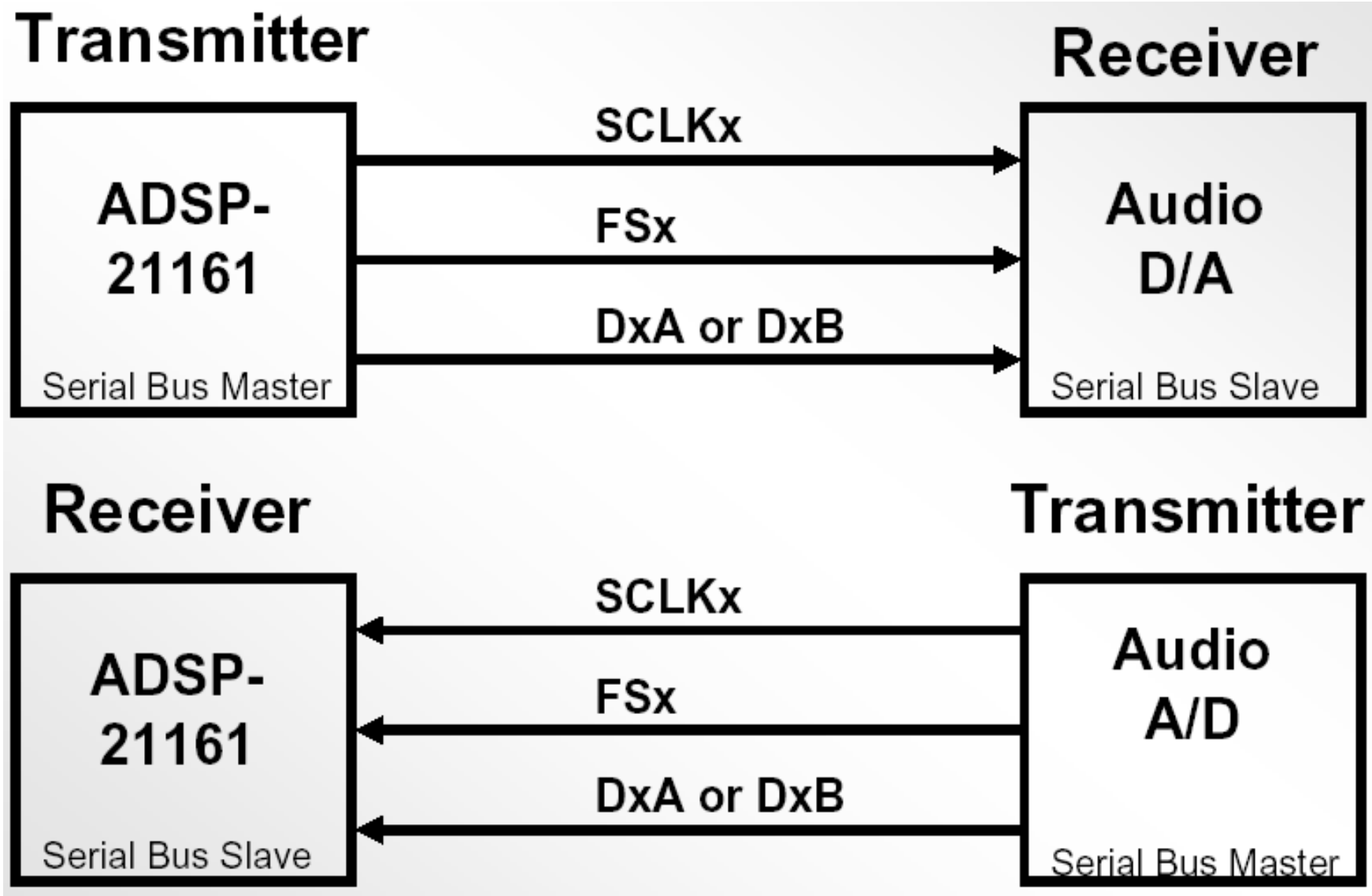
ADSP-21161 – bity kontroli I²S w SPCTLx

- I²S enable (OPMODE)
- Długość słowa (SLEN)
- Kolejność transferu w I²S (L_FIRST)
- Synchronizacja ramki /wybór słowa/ (FS_BOTH)
- Tryb Master (MSTR)
- Włączenie DMA (SDEN_A, SDEN_B)
- Włączenie DMA łańcuchowego (SCHEN_A, SCHEN_B)
- Odblokowanie kanału A lub B (SPEN_A, SPEN_B)

ADSP-21161 - bity kontroli I²S w SPCTLx



Protokół szeregowy I²S



Przerwania portu szeregowego

- **Metoda przerwań DMA**
  - Następuje po zakończeniu DMA
  - Łańcuchowa, niełańcuchowa
- **Metoda przerwań sterowanych rdzeniowo**
  - Występuje podczas nadawania, gdy bufor TX nie jest pełen
  - Występuje podczas odbierania, gdy bufor RX nie jest pusty

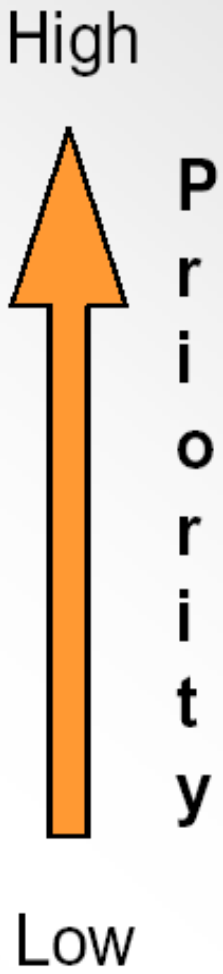
SPORT Interrupts

Interrupt Name	Interrupt
SP0I	SPORT0 DMA Channels 0 & 1 (Highest Priority)
SP1I	SPORT1 DMA Channels 2 & 3
SP2I	SPORT2 DMA Channels 4 & 5
SP3I	SPORT3 DMA Channels 6 & 7 (Lowest Priority)

ADSP-21161: kanał DMA SPORTa

8 kanałów DMA dla obsługi SPORTu:

- DMA Channel 0 RX0A/TX0A SPORT0 A data
- DMA Channel 1 RX0B/TX0B SPORT0 B data
- DMA Channel 2 RX1A/TX1A SPORT1 A data
- DMA Channel 3 RX1B/TX1B SPORT1 B data
- DMA Channel 4 RX2A/TX2A SPORT2 A data
- DMA Channel 5 RX2B/TX2B SPORT2 B data
- DMA Channel 6 RX3A/TX3A SPORT3 A data
- DMA Channel 7 RX3B/TX3B SPORT3 B data



Przesyłanie pojedynczych słów

- Bez przerwań
  - Proste programowanie
  - Zawieszenie rdzenia do czasu wystąpienia danych do przesłania
    - By zapobiec zawieszaniu należy ustawić bit BHD* w SYSCON
- Sterowane przerwaniami
  - Przerwanie jest generowane, gdy kompletne 32-bitowe słowo zostało odebrane w buforze RX lub gdy bufor TX nie jest zapełniony

*Buffer Hang Disable

- Nie należy zawieszać rdzenia umyślnie. Nie można czytać lub pisać do nieaktywnych buforów danych SPORTa.

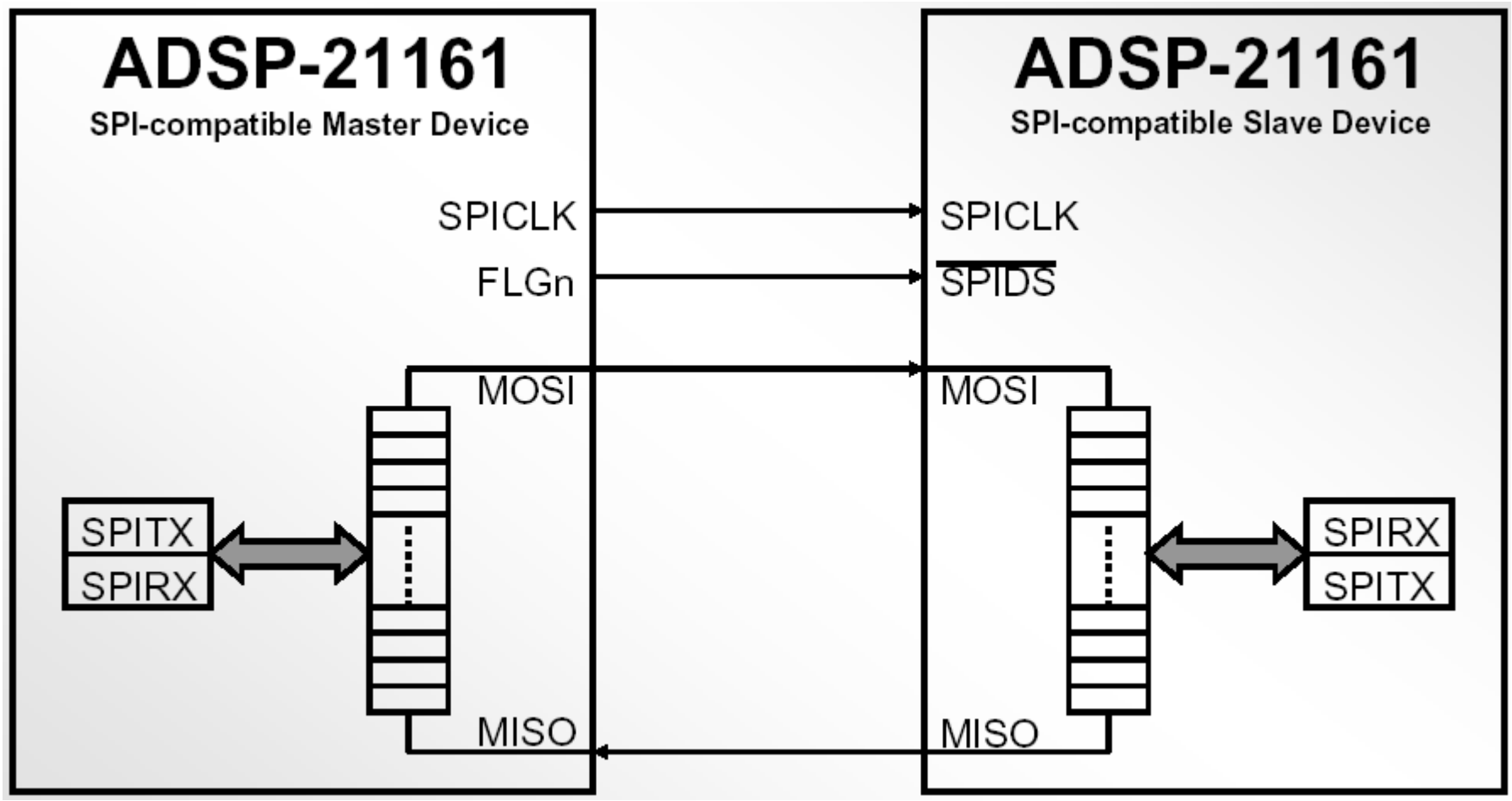
Szeregowy interfejs urządzeń zewnętrznych (SPI)

- SPI to standard przemysłowy 4-przewodowego interfejsu szeregowego (opracowany przez Motorolę), który pozwala na komunikację pomiędzy ADSP-21161, a innymi urządzeniami kompatybilnymi z SPI, takimi jak:
  - mikrokontrolery, mikroprocesory
  - ADC i DAC
  - cyfrowe urządzenia audio (SP/DIF, AES/EBU)
  - wyświetlacze LCD
  - konwertery próbkujące
  - rejestry przesuwne FPGA
- Port układu 21161 kompatybilny z SPI posiada następujące możliwości:
  - Praca w trybie Full Duplex
  - Tryb Master-Slave
  - Tryb Multimaster
  - Wyjścia typu otwarty dren
  - Programowalne Baud Rate, biegunowości i fazy zegarowe
  - 21161 może być uruchomione w trybie slave poprzez urządzenie SPI master
  - Słowa długości 8, 16 i 32 bitów

Piny szeregowego interfejsu urządzeń zewnętrznych

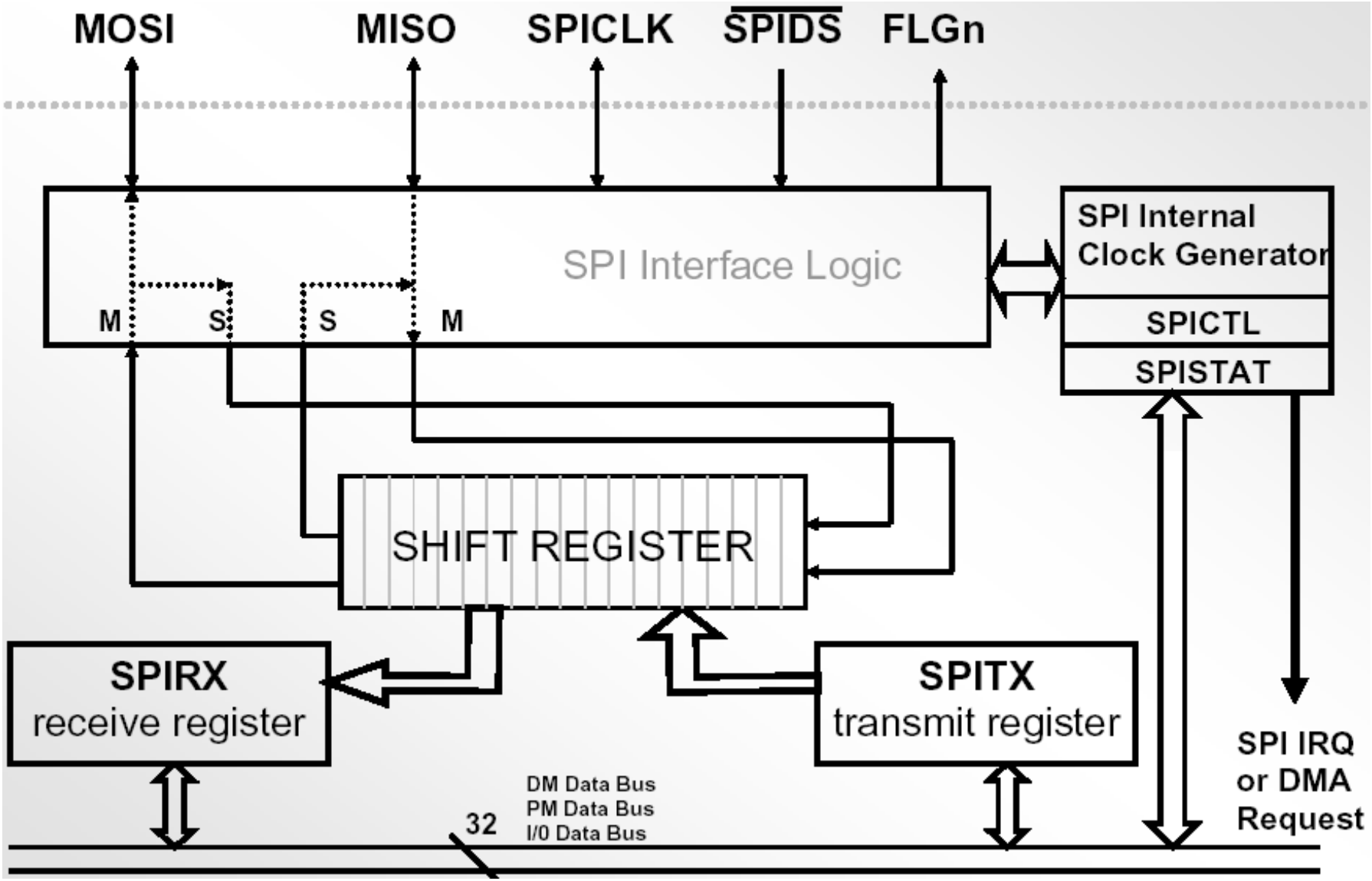
- Port kompatybilny z SPI to interfejs 4-przewodowy składający się z:
  - Pinu wyboru urządzenia SPIDS-SPI
    - Chip select dla urządzeń slave – piny SPI Master FLAG[0-3] połączone z pinem slave/SPIDS
  - SPICLK – pin sygnału zegarowego
    - generowany przez urządzenie master SPI
  - 2 piny danych:
    - MOSI – master out/slave in
      - MOSI – pin wyjściowy tx jeśli master SPI, wejściowy rx jeśli slave SPI
    - MISO – master in/slave out
      - MISO – pin wejściowy rx jeśli master SPI, wyjściowy tx jeśli slave SPI

Połączenie SPI między urządzeniami master i slave

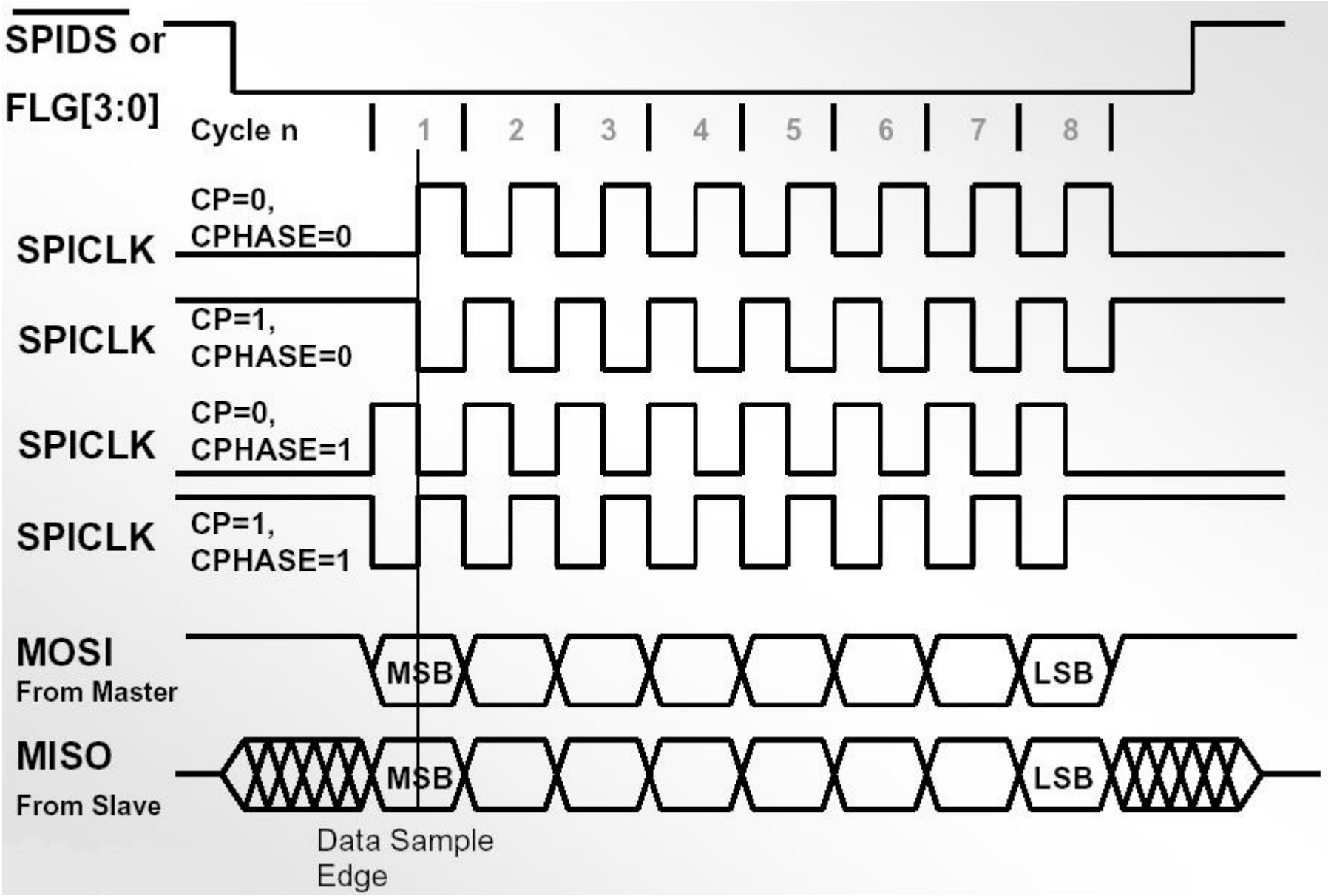


** FLGn to mogą być piny FLG0, FLG1, FLG2 albo FLAG3

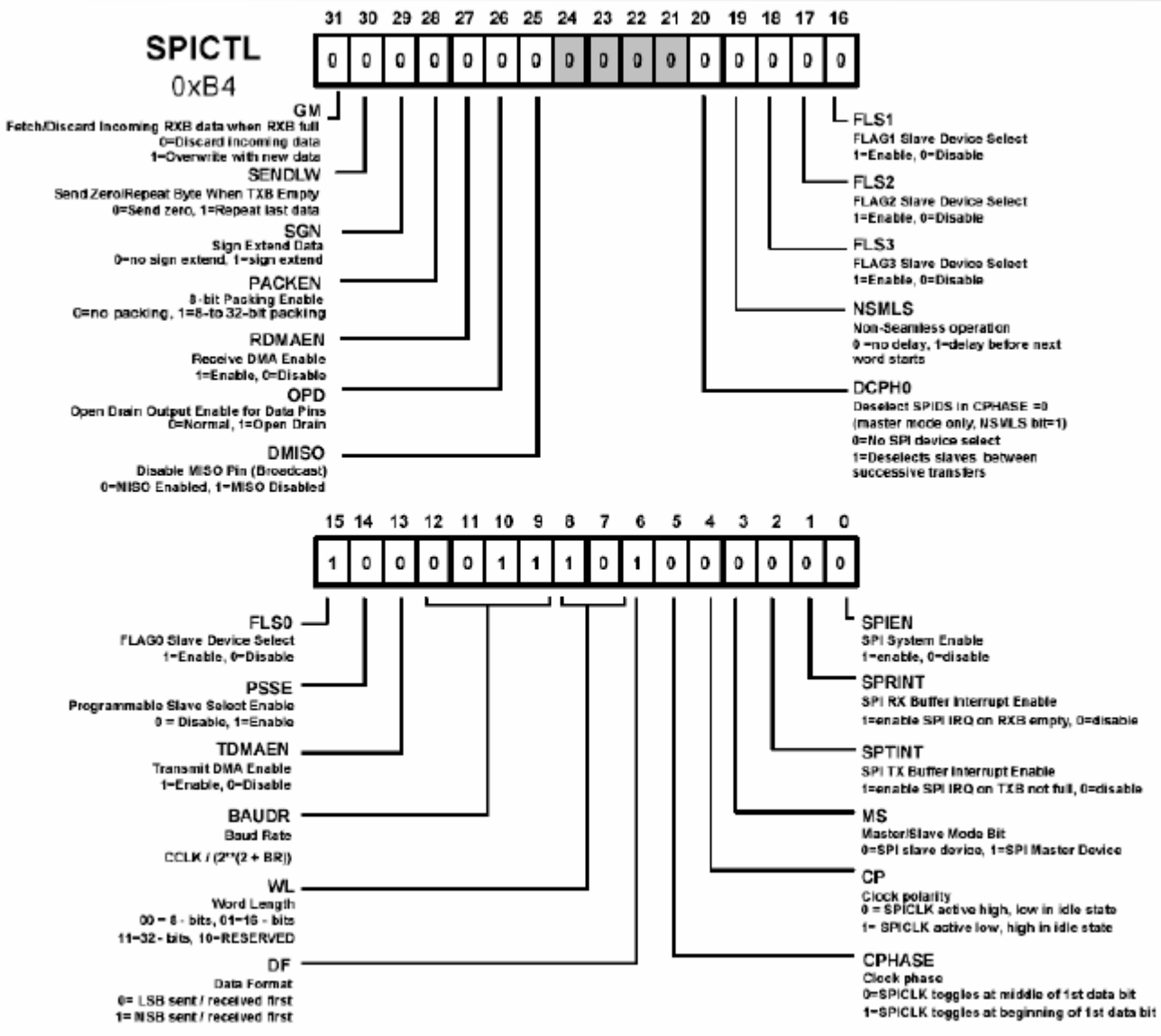
Schemat blokowy interfejsu szeregowego SPI



Przebiegi czasowe interfejsu SPI ADS-21161

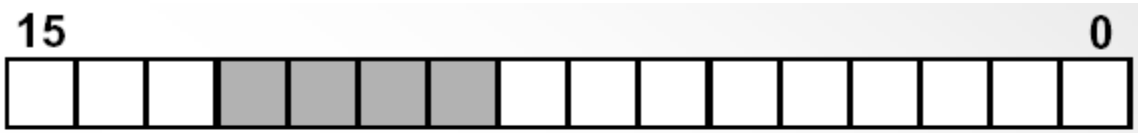


SPICTL - Rejestr kontrolny (IOP)



BAUDR: Baud Rate Generation (częst. SPICLK)

BAUDR (bity 9-12 w SPICTL)



- Used For Internally Generated SPICLK
- f_{SPICLK} = Core Clock / ($2^{2 + BAUDR}$)

- Przykłady:

  Jeśli zegar rdzenia pracuje na częstotliwości 100MHz, jaki BAUDR jest wymagany dla częstotliwości SPICLK = 25MHz

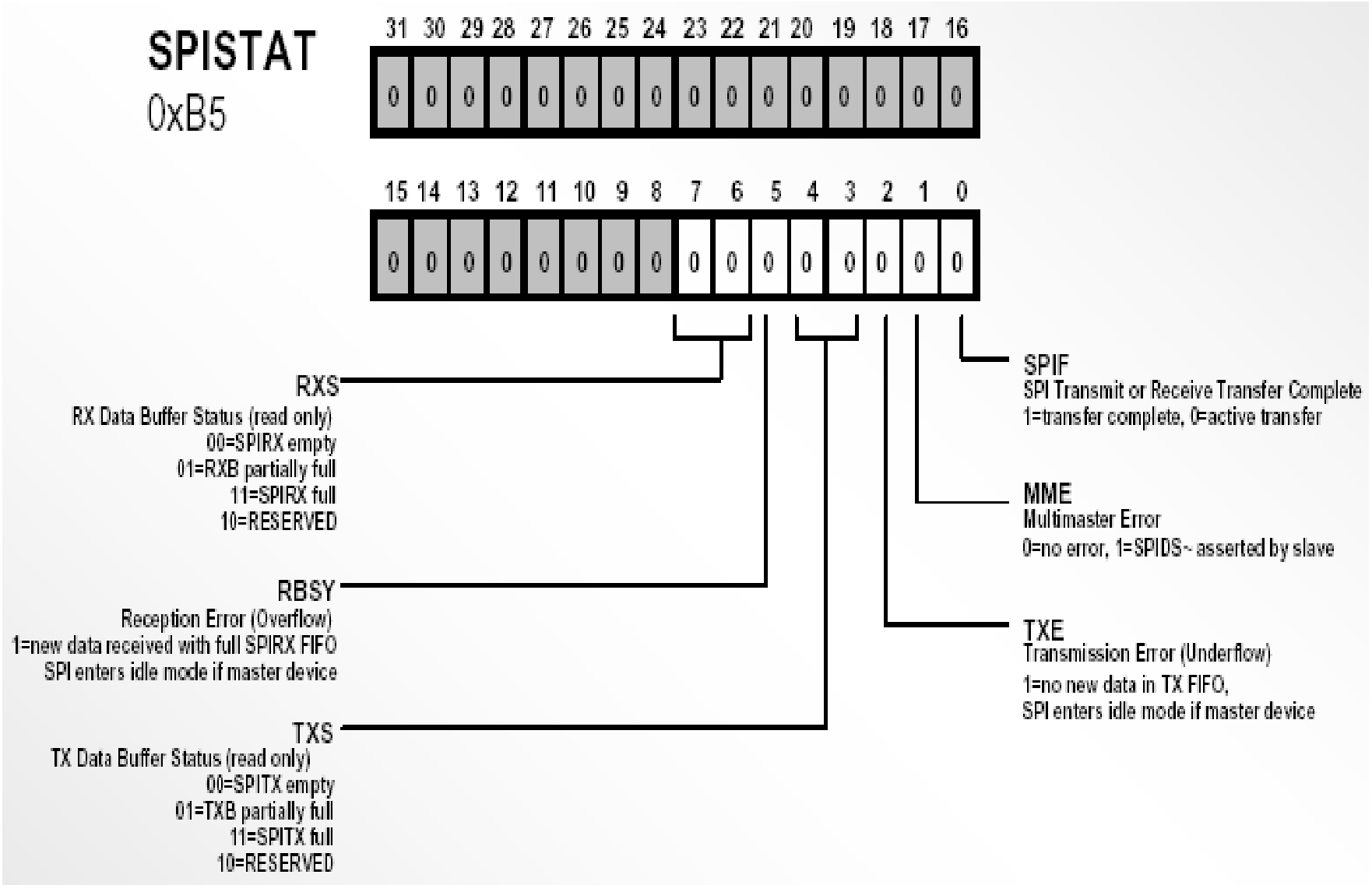
  ODPOWIEDŹ = $\frac{100\ MHz}{2^{(2 + BAUDR)}} = 25\ MHz$

  => BAUD = 0

  Jeśli zegar rdzenia pracuje na częstotliwości 100MHz, jaka częstotliwość SPICLK jest generowana jeśli BAUDR=0xD(13)?

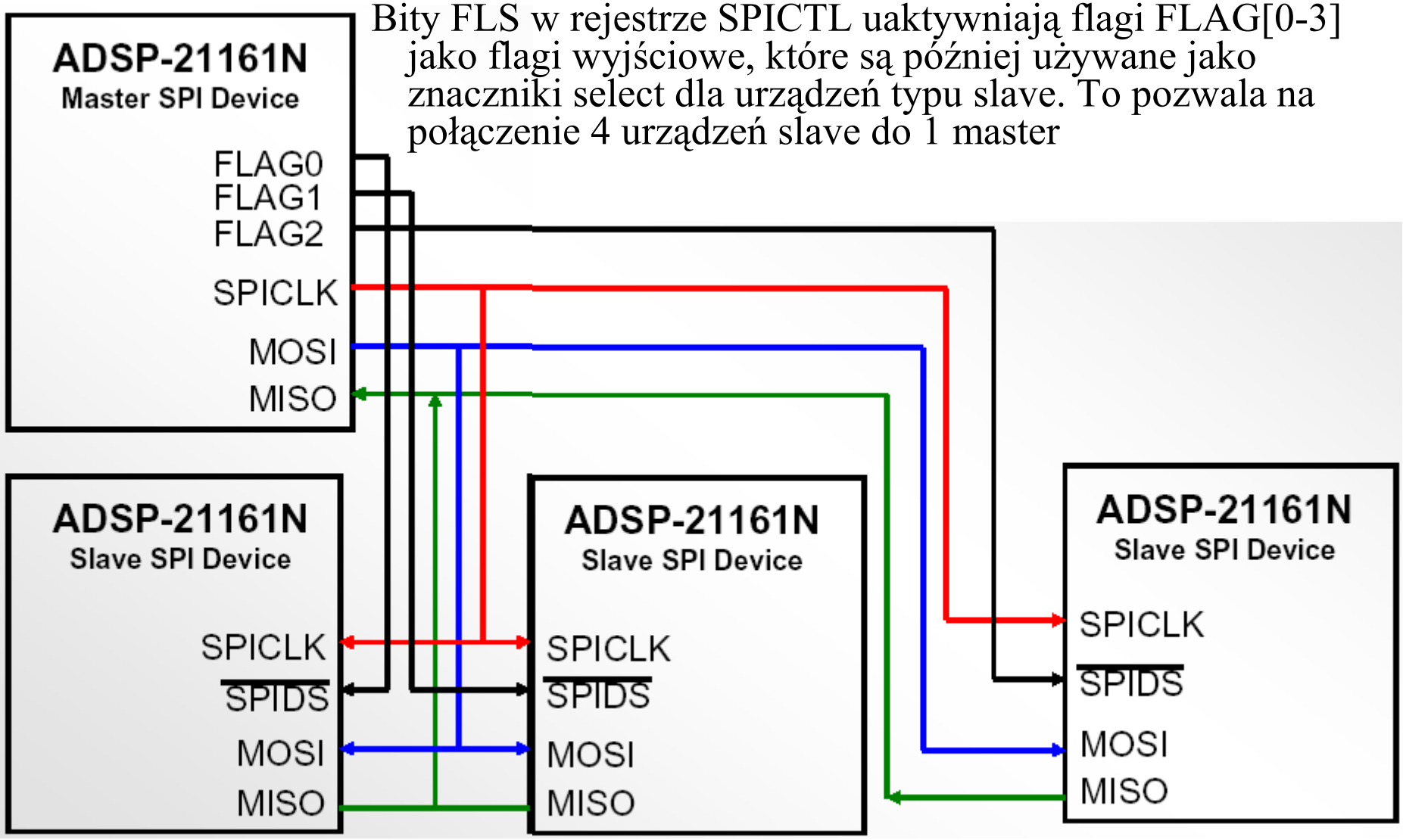
  ODPOWIEDŹ: $f_{SPICLK} = \frac{100\ MHz}{2^{(2 + 13)}}$ => 100e6/65536 = 1525 Hz

SPISTAT – Rejestr kontrolny SPI (IOP)



Interfejs Multislave SPI

Bity FLS w rejestrze SPICTL uaktywniają flagi FLAG[0-3] jako flagi wyjściowe, które są później używane jako znaczniki select dla urządzeń typu slave. To pozwala na połączenie 4 urządzeń slave do 1 master



Interfejs szeregowy SPI
Sygnały błędu i flagi

- Błąd Multi-Master (MME w SPISTAT)
  - Bit będzie ustawiony jeśli pin ~SPIDS urządzenia master ADSP-21161 jest wprowadzony w niski stan przez inne urządzenie systemu
  - Aby zapobiec konfliktowi dwóch sterowników, master reaguje w następujący sposób (jak tylko błąd zostanie wykryty):
    - bit kontrolny MS jest czyszczony konfigurując urządzenie SPI jako slave
    - system SPI zostaje wyłączony przez wyczyszczenie SPIEN
    - bit statusu MME jest ustawiany. Kod użytkownika może sprawdzić MME
- Błąd przesyłania (TXE w SPISTAT)
  - Bit będzie ustawiony jeśli nastąpi niedostateczny przepływ danych (transmission underflow), na przykład gdy SPI DMA jest aktywne albo gdy wystąpi przerwanie SPI, ale żadne dane nie są dostępne w kolejce FIFO (SPITX pusty)
- Błąd odbioru (RBSY w SPISTAT)
  - Bit będzie ustawiony jeśli urządzenie otrzymuje dane, ale nie ma już wolnego miejsca w kolejce FIFO

Inicjowanie systemu ADSP-21161

Przydatne definicje

• Loader	elfloader
• Boot Loader Kernel	Plik wykonywalny, który przeprowadza inicjalizację pamięci w miejscu docelowym. Wykonuje i nadpisuje siebie z programem wykonywalnym.
• Plik inicjujący (boot file - .ldr)	Plik wyjściowy Loadera który zawiera boot loadera i sformatowaną konfigurację systemu
• Booting (inicjalizacja)	Proces ładowania Boot Loadera, inicjalizacji pamięci i uruchamiania aplikacji w miejscu docelowym
• Pamięć	Pamięć danych, pamięć programu, pamięć wewnętrzna lub zewnętrzna

Inicjalizacja ADSP-21161

- ADSP-21161 posiada 5 trybów inicjalizacji
  - PROM Boot (BMS~ podłączony do EPROM chip select)
  - HOST Boot
  - Link Port Boot
  - Serial Boot poprzez SPI
  - No Boot Mode (procesor uruchamia się z pamięci zewnętrznej)
- Tryb inicjalizacji określony jest przez połączenia sprzętowe (hardwired) pinów EBOOT, LBOOT i BMS~
- DSP uruchamia połączone sprzętowo DMA aby załadować 256 słów instrukcji
- ADI dostarcza pliki jądra loadera dla trybów inicjalizacji PROM, Host, Link Port i SPI w narzędziach VisualDSP++

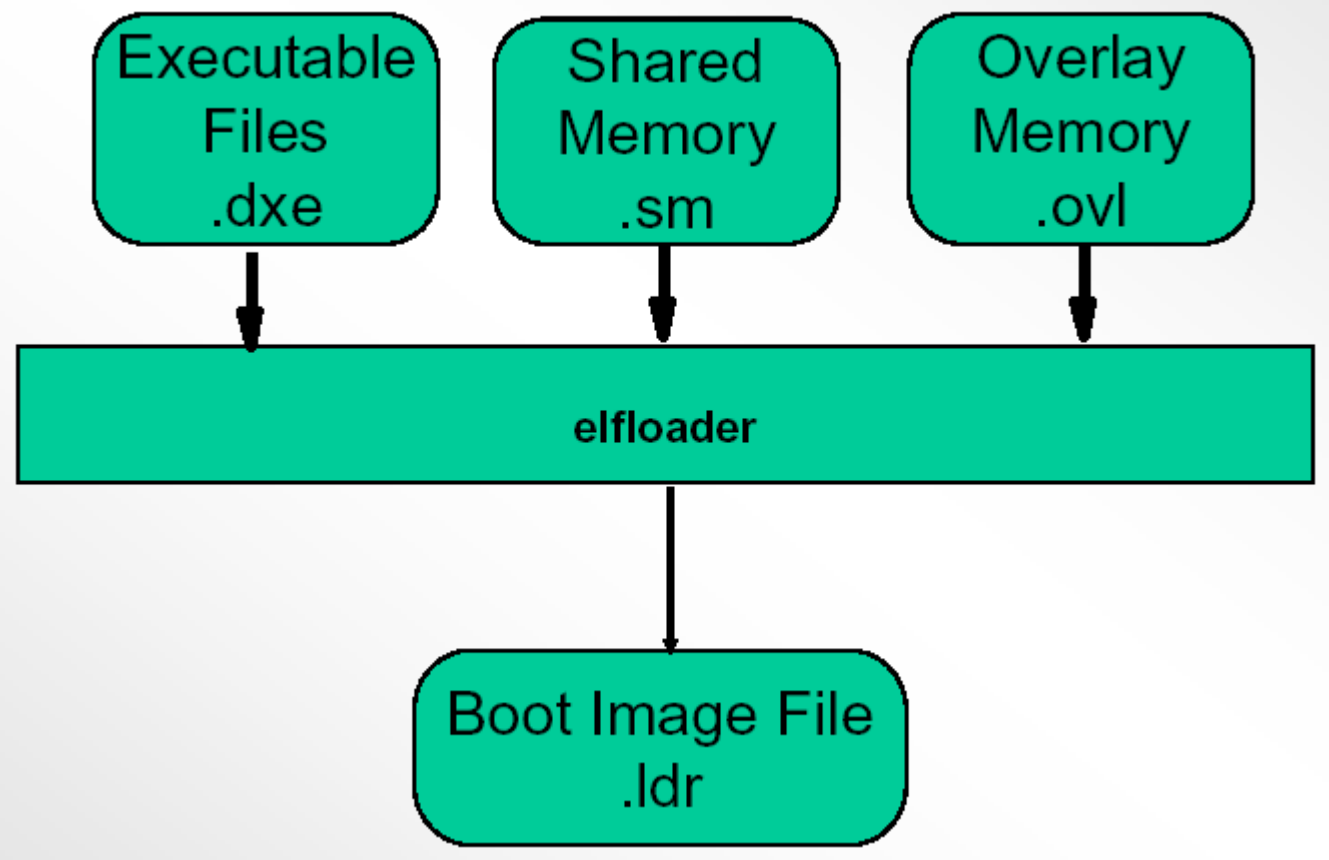
Inicjalizacja SHARC podczas podłączenia zasilania

- Źródła sekwencji inicjalizujących (EPROM, port hosta i port łączności) są wybierane poprzez ustawienie pinów

EBOOT	LBOOT	BMS	Boot Mode
1	0	output	EPROM (connect BMS~ to CS~)
0	0	1 (input)	Host Processor (or SHARC thru EP)
0	1	1 (input)	Link Port 4
0	0	0 (input)	No Booting, execute from ext memory
0	1	0 (input)	SPI Booting

- Ogólny scenariusz inicjalizacji
  - Kanał DMA 10 (albo kanał 8 dla inicjalizacji link/SPI) jest automatycznie skonfigurowany dla przesyłania 256 słów (48bitów) począwszy od lokacji 0x400000
  - Pierwsze 256 instrukcji to “jądro loadera”
  - Następnie jądro loadera jest uruchamiane i ładuje kod aplikacji do DMA oraz dane do zewnętrznej lub wewnętrznej pamięci
  - Ostatecznie, jądro loadera nadpisuje siebie pierwszymi 256ma słowami aplikacji

Środowisko VisualDSP++ Loadera



Środowisko Boot Loadera

- Środowisko loadera tworzy pliki ładowalne(boot-loadable) z plików wykonywalnych
- Źródła boot jądra loadera sa dostarczone wraz z narzędziami programowymi i mogą być modyfikowane przez użytkownika. Jądra (Kernels) wspierają inicjalizację zewnętrznych spakowanych instrukcji dla każdej szerokości magistrali zewnętrznej
    - EPROM Boot
        - 161_PROM_ASM
    - Host Boot
        - 161_HOST_ASM
    - Link Boot
        - 161_LINK_ASM
    - SPI Boot
        - 161_SPI_ASM
- Loader jest automatycznie wywołany jeśli opcja IDDE, aby wygenerować obraz pliku inicjalizującego została wybrana
- Loader może być wywołany z poziomu linii poleceń DOS przy użyciu polecenia “elfloader” z odpowiednimi plikami i switchami

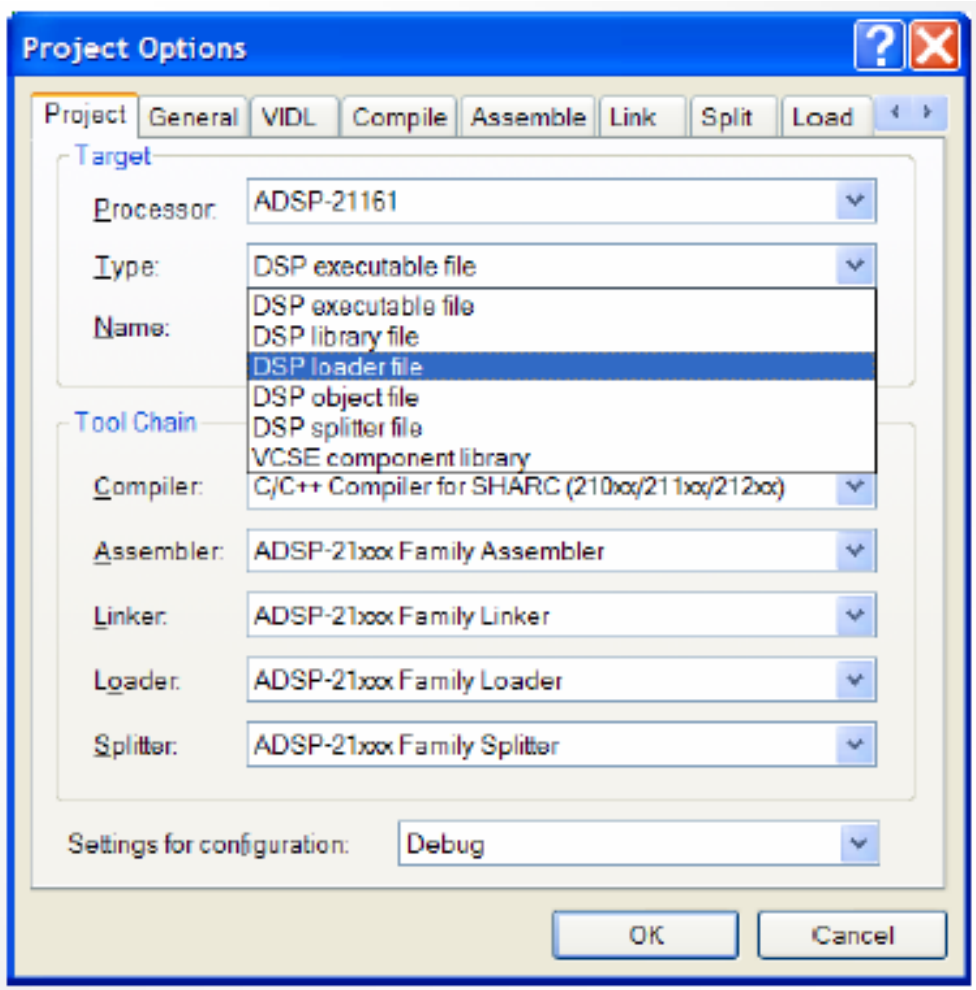
Jądro Loadera

- Jądro loadera inicjalizuje zmienne, macierze zerowe są przechowywane w skompresowanym formacie i inicjalizowane (jak wymagane jest w ANSI C)
- EP Host Booting: Jądro loadera zablokuje magistralę EP podczas inicjalizacji pamięci i przestanie przyjmować DMA slave z urządzenia Host dopóki nie zakończy się zapis do pamięci
  - To pozwoli na wstrzymanie potwierdzeń hosta (REDY) dopóki loader nie będzie gotowy
  - Może to zająć długi okres czasu jeśli inicjalizowana jest macierz zer
  - Interfejs hosta powinien zapewnić HBR pomiędzy przesyłaniem słów, aby pozwolić DSP na inicjalizację pamięci zewnętrznej poprzez magistralę

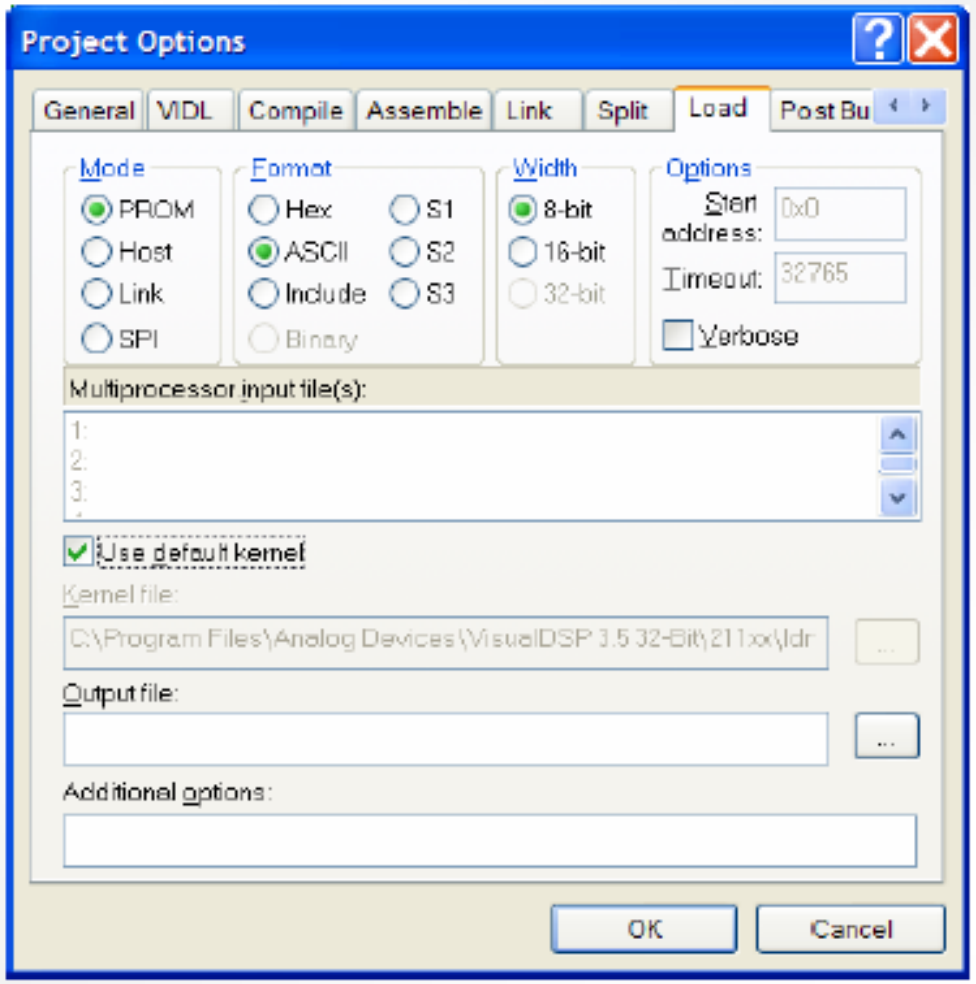
Środowisko Loadera - Jak tworzony jest plik .LDR?

- Boot Loader korzysta z 5-krokowego procesu aby stworzyć plik .ldr:
  1. Parsuje plik wykonywalny DSP
     Określa które segmenty są oparte na RAMie i kopiuje te sekcje do pliku boot. Każdy segment zawierający zinicjalizowane dane jest przetworzony przez loadera
  2. Oddziela inicjalizacje o zerowej wartości
     Kompresuje je do jednego 48-bitowego słowa dla efektywnego przechowania w EPROMie. Długość jest użyta do ustawienia LCNTR w zerowej inicjalizacji D0 LOOP.
  3. Jądro loadera
     Dodaje określony boot loader do pliku boot
  4. Bloki inicjalizacji pamięci
     Wszystkie informacje dotyczące segmentów inicjalizacji są wpisane do pliku boot.
  5. Ostateczna inicjalizacja
     Ostatnią informacją wpisywaną do pliku boot jest ostateczna inicjalizacja która nadpisuje składający sięz 256 słów jądra boot. Plik boot jest skonwertowany do odpowiedniego formatu (Intel hex-32, ASCII lub binarny)

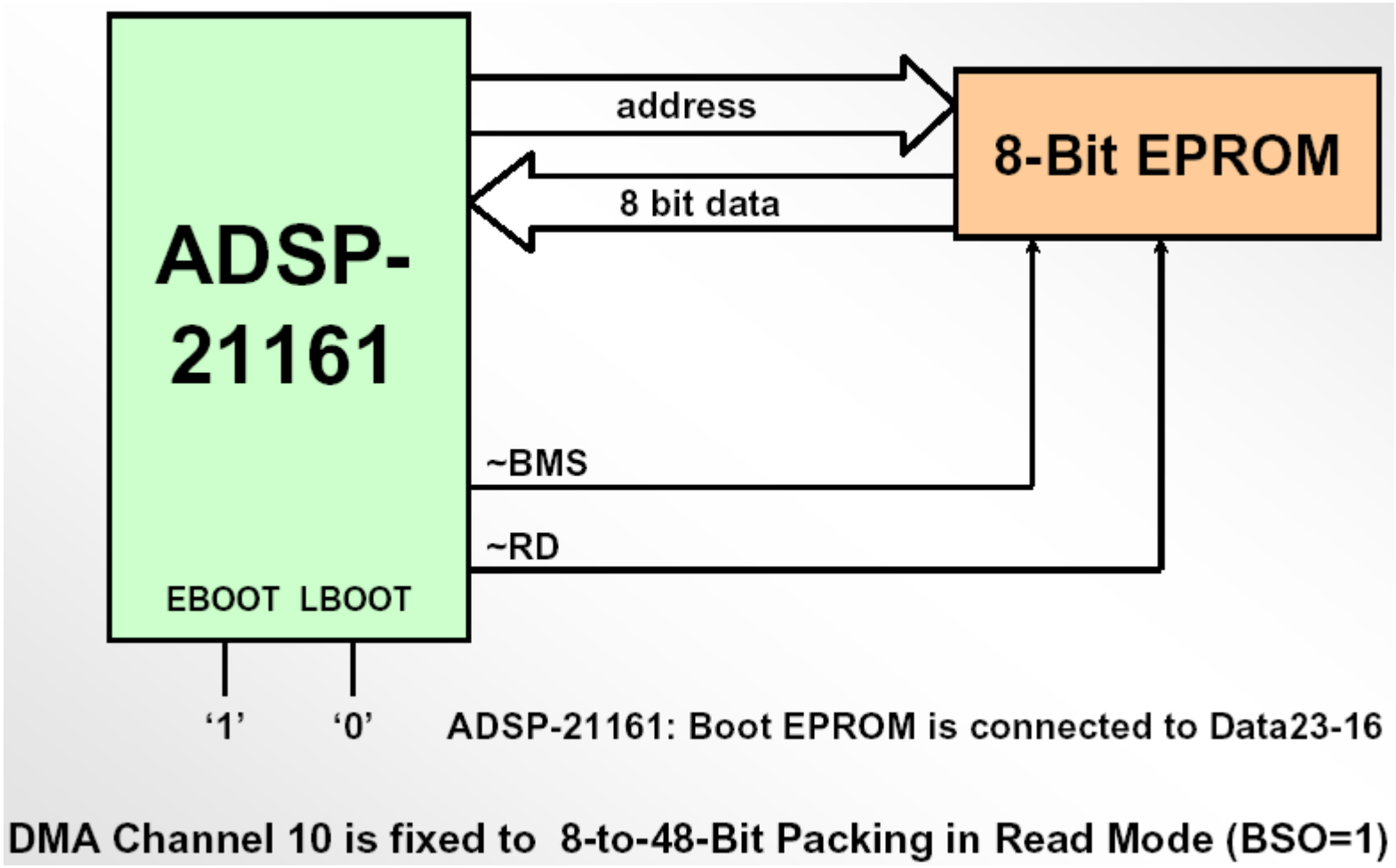
Format projektowego pliku wyjściowego



Visual DSP++: Strona właściwości Loadera



PROM Booting



DMA Channel 10 is fixed to 8-to-48-Bit Packing in Read Mode (BSO=1)

Procedura Host Booting

- Host ma dostęp do wszystkich rejestrów IOP
- Programowanie SYSCON (w dostępach 8-bitowych)
    - Ustawienie szerokości magistrali hosta (HBW)
    - Wybranie kolejności słów (HMSWF)
- Programowanie DMAC10 (w dostępach 8-bitowych)
    - Ustawienie trybu pakowania (PMODE)
- Dla każdego 48-bitowego słowa obrazu inicjalizującego:
    - Przyznanie ~HBR i ~CS
    - Oczekiwanie na ~HBG
    - Zapisanie wszystkich 3 podsłów do EPB0 (16 bitowy host)
    - *Unieważnienie ~CS i ~HBR
    - Czekanie przez kilka NOPów

* SHARC może próbkować nieaktywny sygnał ~HBR i pozwala na generowanie cykli przejściowych przez hosta. Oznacza to, że SHARC może zainicjalizować pamięć zewnętrzną poprzez magistralę

Procedura Link Port Booting

- Używane są Port Łączności 0, LBUF0 i kanał 8 DMA
- Transfery danych o szerokości 4-bitów
- Najbardziej znaczący półbajt musi być pobrany pierwszy

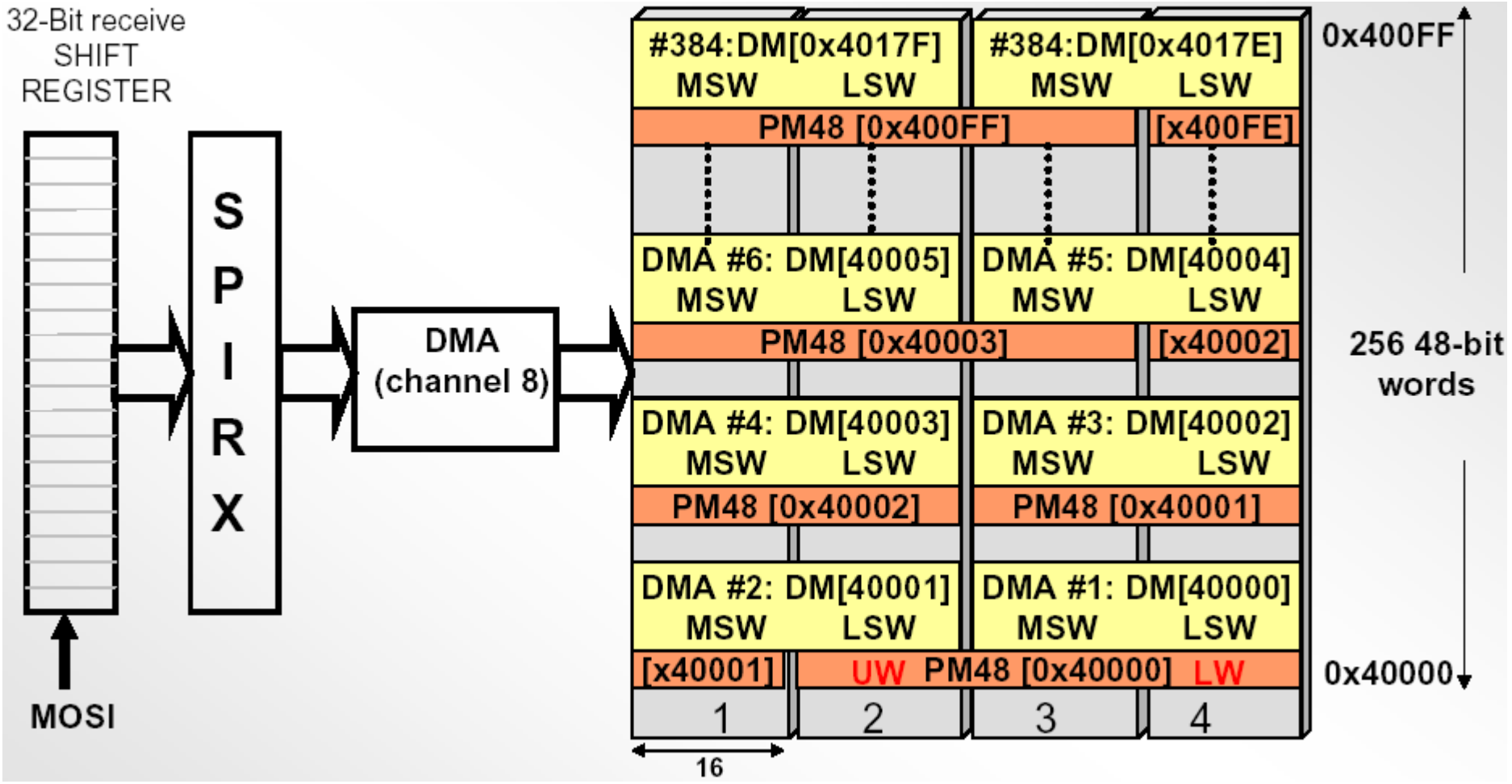
Bootowanie SPI

- Inicjalizacja poprzez SPI ma miejsce gdy EBOOT='0', LBOOT='1' i BMS(input)='0'
- DMA 384x32-bit jest wymagane aby zainicjalizować 256 instrukcji jądra loadera
  - Jądro loadera jest załadowane przy pomocy 32-bitowtch transferów do dwukolumnowych adresów, które wymagają 384 słów 32-bitowych aby załądować 256 instrukcji 48-bitowych
- Podczas Resetu rejestry parametryzujące kanał 8 DMA, jeśli skonfigurowane do bootowania z SPI, są zainicjalizowane do następujących warotści:
  - IISRX = 0x40000
  - IMSRX = 0x1
  - CSRX = 0x180 = 384 32-bitowych słów
  - GPSRX = niezainicjalizowany

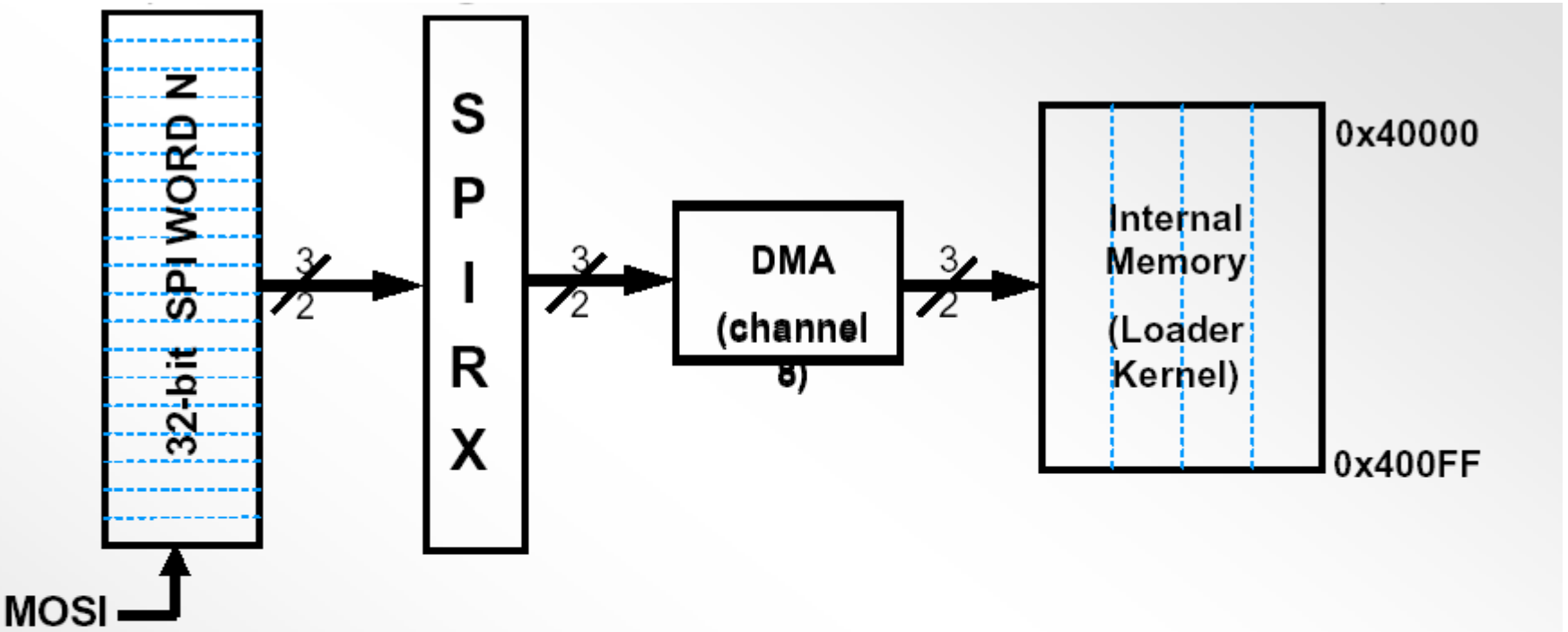
Rozmiar hosta inicjalizacji SPI

- W trybie SPI Boot, rozmiar rejestru przesuwnego hosta przyjmuje domyślnie rozmiar 32 bitów (WL = 11 w SPCTL)
  - Dla 16-bitowych hostów, dwa 16-bitowe słowa muszą zostać spakowane w 32-bitowym rejestrze przesuwnym SPI przed zapisaniem poprzez DMA do pamięci wewnętrznej
  - Dla 8-bitowych hostów, cztery 8-bitowe słowa muszą zostać spakowane w 32-bitowym rejestrze przesuwnym SPI przed zapisaniem poprzez DMA do pamięci wewnętrznej
- Bit “Seamless Operation” (SMLS) w SPICTL domyślnie zapewnia “brak opóźnień” pomiędzy kolejnymi słowami SPI. Pozwala to na odbiór i pakowanie czterech 8-bitowych lub dwóch 16-bitowych słów w rejestrze przesuwnym RX przed załadowaniem ich do SPIRX
- Po załadowaniu jądra loadera, program hosta SPI lub jądra loadera może ponownie ustawić nowy rozmiar hosta w rejestrze kontrolnym SPI aby odwzorować aktualny rozmiar słowa 8 lub 16-bitowego na potrzeby dodatkowych transferów danych

SPI Slave Boot
Wstępne ładowanie 256 instrukcji jądra Boot



SPI Slave Boot z 32-bitowych hostów SPI
Wstępne ładowanie 256 instrukcji jądra Boot



Serial Data From 32-bit SPI Host

Word#1: 0x33445566

Word#2: 0xCCDD1122

Word#3: 0x7788AABB

Word#4: 0x01234567

Word#5: 0x89ABCDEF

Word#6: 0x88885555

Example 48-bit Instruction Opcodes

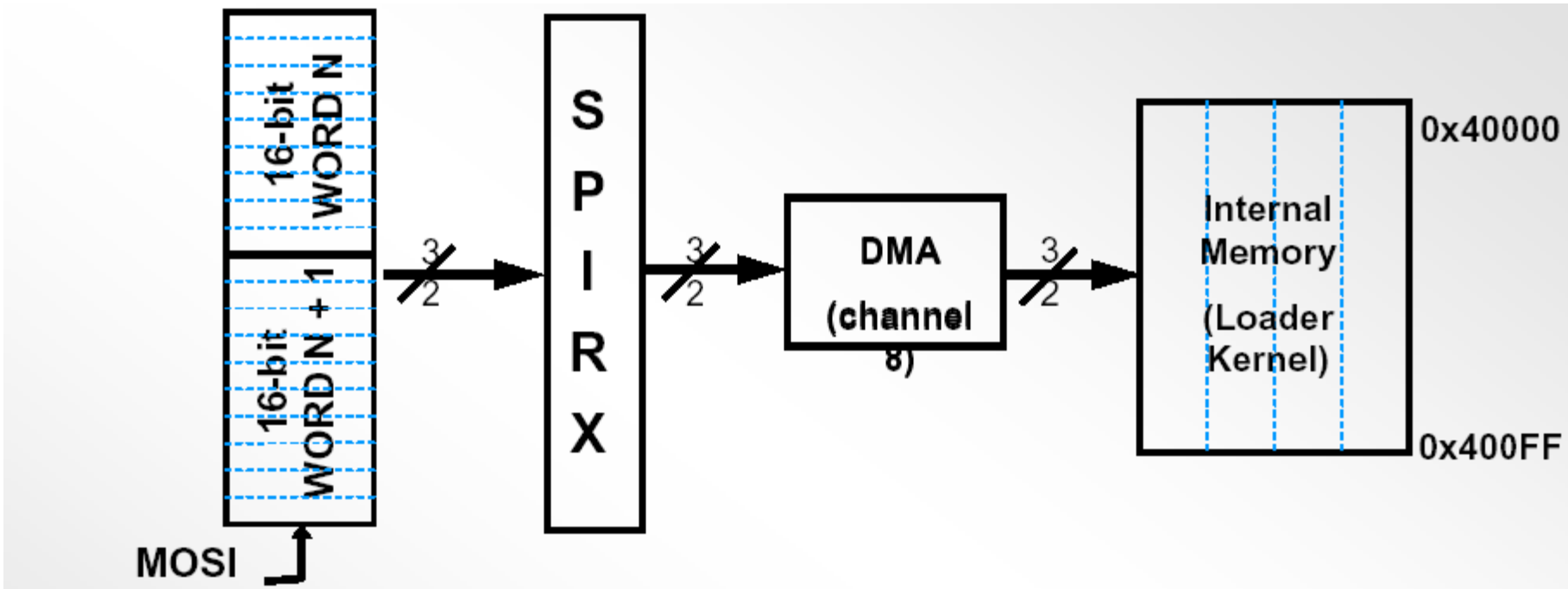
[0x40000] = 0x1122 33445566;

[0x40001] = 0x7788 AABBCCDD;

[0x40002] = 0xCDEF 01234567;

[0x40003] = 0x8888 555589AB;

SPI Slave Boot z 16-bitowych hostów SPI
Wstępne ładowanie 256 instrukcji jądra Boot



Serial Data From 16-bit SPI Host

Word#1: 0x5566	Word#2: 0x3344
Word#3: 0x1122	Word#4: 0xCCDD
Word#5: 0xAABB	Word#6: 0x7788
Word#7: 0x4567	Word#8: 0x0123
Word#9: 0xCDEF	Word#10: 0x89AB
Word#11: 0x5555	Word#12: 0x8888

Example 48-bit Instruction Opcodes

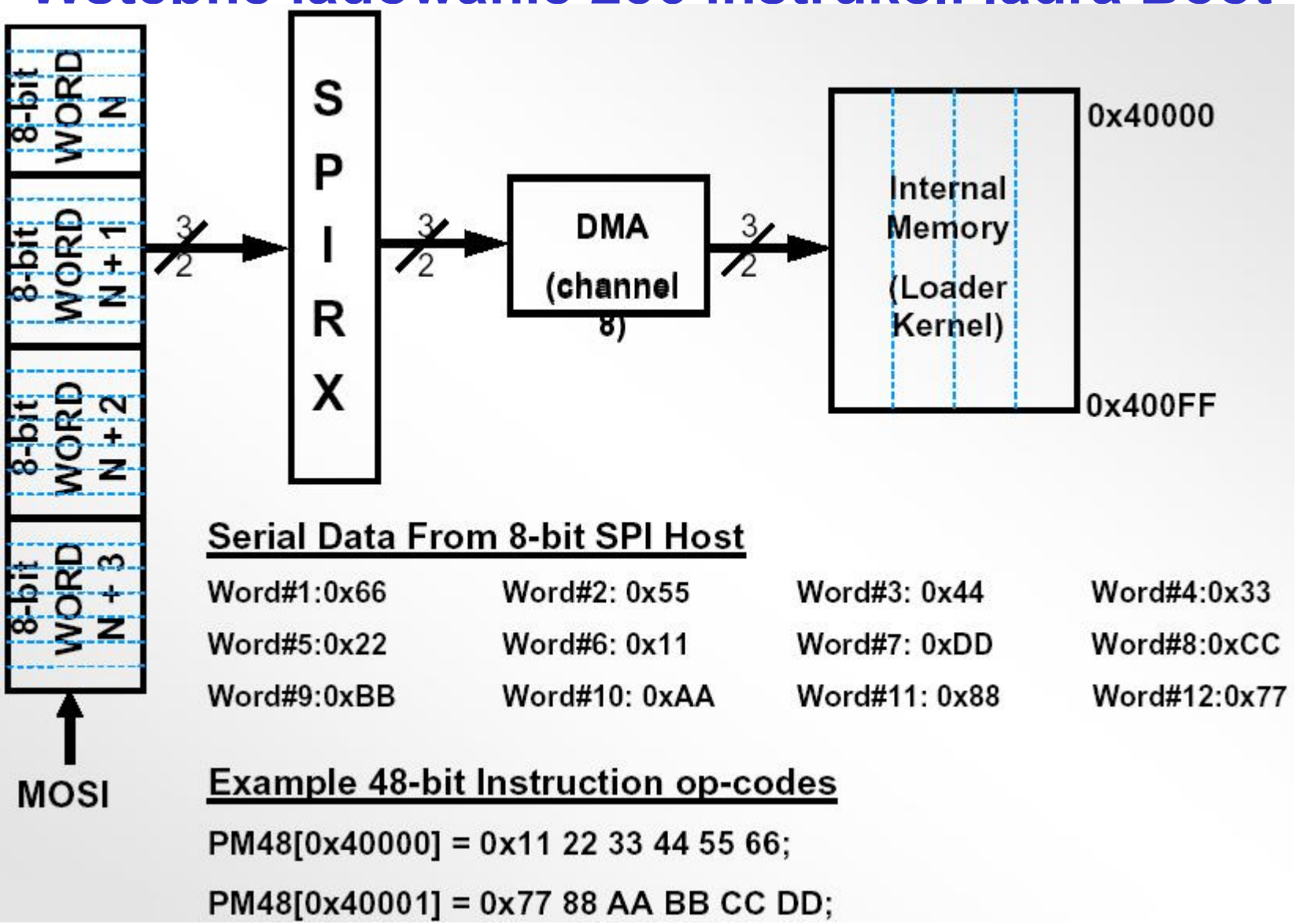
0x40000 = 0x1122 3344 5566;

0x40001 = 0x7788 AABB CCDD;

0x40002 = 0xCDEF 0123 4567;

0x40003 = 0x8888 5555 89AB;

SPI Slave Boot z 8-bitowych hostów SPI
Wstepne ładowanie 256 instrukcii jadra Boot



Bootowanie systemu wieloprocesorowego (1/2)

- EPROM
  - Wszystkie piny /BMS połączone z EPROM/OE
  - ID#1 zainicjalizuje się jako pierwszy, po nim kolejne.
  - Jądro Loadera EPROM przyjmuje pliki .dxe i odczytuje wartość pola ID w SYSTAT aby określić obszar pamięci EPROM do odczytania
  - Wszystkie procesory powinny mieć jakąś flagę programową lub sygnał sprzętowy (piny FLAG) aby można było stwierdzić, że inicjalizacja jest kompletna
- Port zewnętrzny
  - Host może zainicjalizować każdy DSP używając protokołu interfejsu Hosta
  - Host wybiera DSP do zainicjalizowania używając pinów /CS i /HBR
  - Kolejny ADSP-21160, który jest bootowany poprzez EPROM, hosta lub połączenie, może zainicjalizować inne poprzez EP (slave w trybie Host Boot)

Bootowanie systemu wieloprocesorowego (2/2)

- Port łączności

  - Zapis z uprzednio zainicjalizowanego ADSP-21161 do portu łączności LP0 urządzenia ADSP-21161 w trybie Link Boot.

- Port SPI

  - Zapis z urządzenia master SPI do SPIRX urządzenia ADSP-21161 w trybie SPI Boot. Do 4 urządzeń slave ADSP-21161 może być bootowanych równocześnie. Jeśli skonfigurowany do trybu SPI Boot, pin MISO jest wyłączany podczas resetu. To daje nam następujące możliwości

    • Szerokopasmowe jądro i identyczny kod aplikacji dla wszystkich SHARC slave z wszystkimi 4/SPIDS pinami wybieranymi równocześnie

    • Można załadować inny kod do każdego procesora sekwencyjnie sterując tylko jednym pinem /SPIDS w każdym SHARCu w dowolnym momencie

Symulacja Sekwencji Bootowania