

# **Narzedzia sprzetowe**

## **Sekcja 15**

# System oceny ADSP-21161N-EZLITE

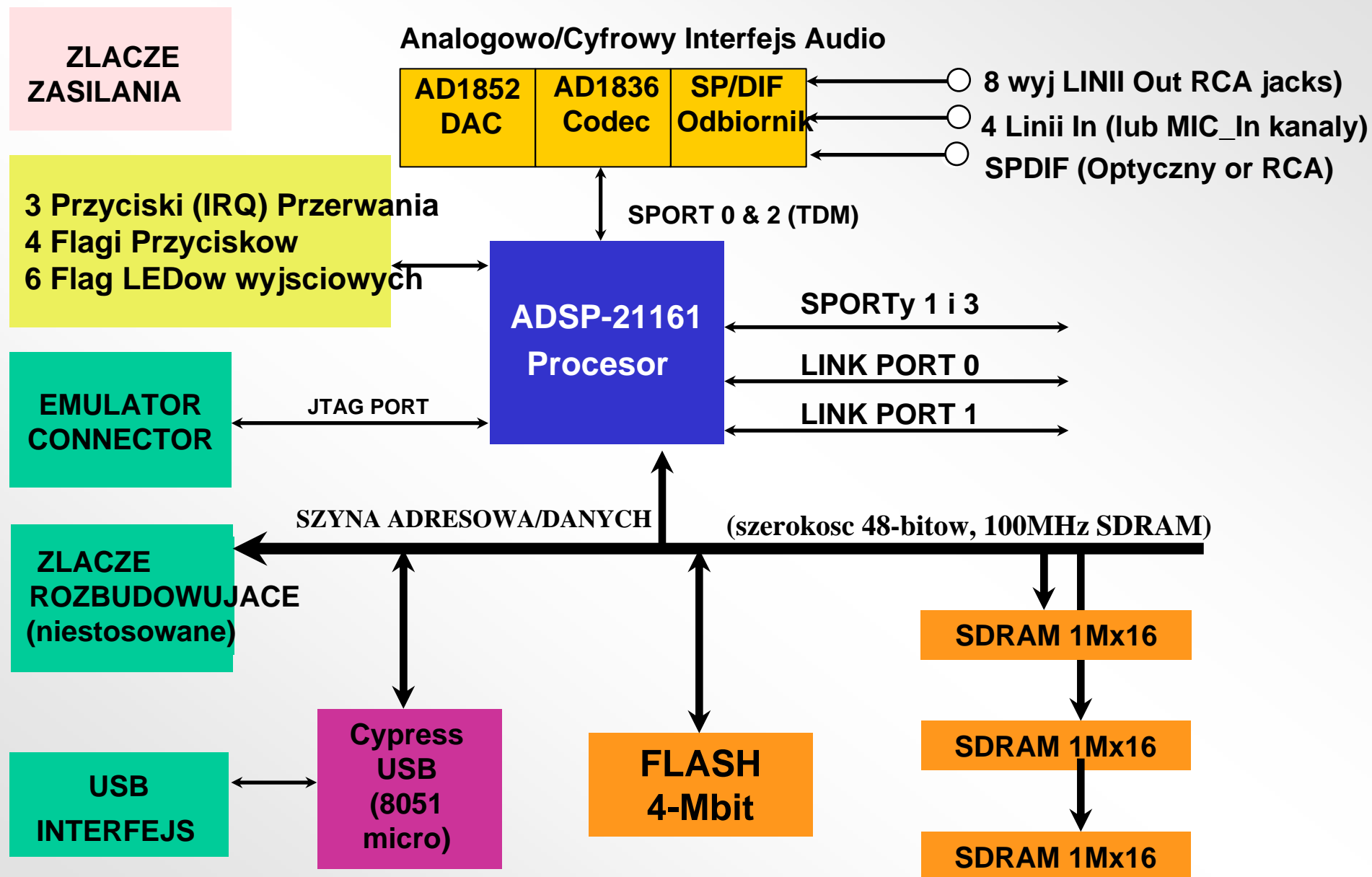
- **Autonomiczna Plyta wartosciowania ADSP-21161**
- **CD ROM**
  - **Zestaw wartosciowania VisualDSP Demo**
    - **Zawira bezterminowa licencje Demo VisualDSP++ (3.0)**
      - **IDDE, Kompilator C SHARC, Assembler, Linker, i Loader**
      - **Czolowy VisualDSP Debugger i Symulator SHARC**
      - **licencja Demo - limit uzytkownikow do 5K pamieci programu (instrukcje i/lub dane)**
    - **Manual uzytkownika, Schematic, BOM, Demos**
- **Akcesoria**
  - **Kable USB**
  - **Zasilacz**

# Wlasciwosci ADSP-21161N EZ-KIT-LITE

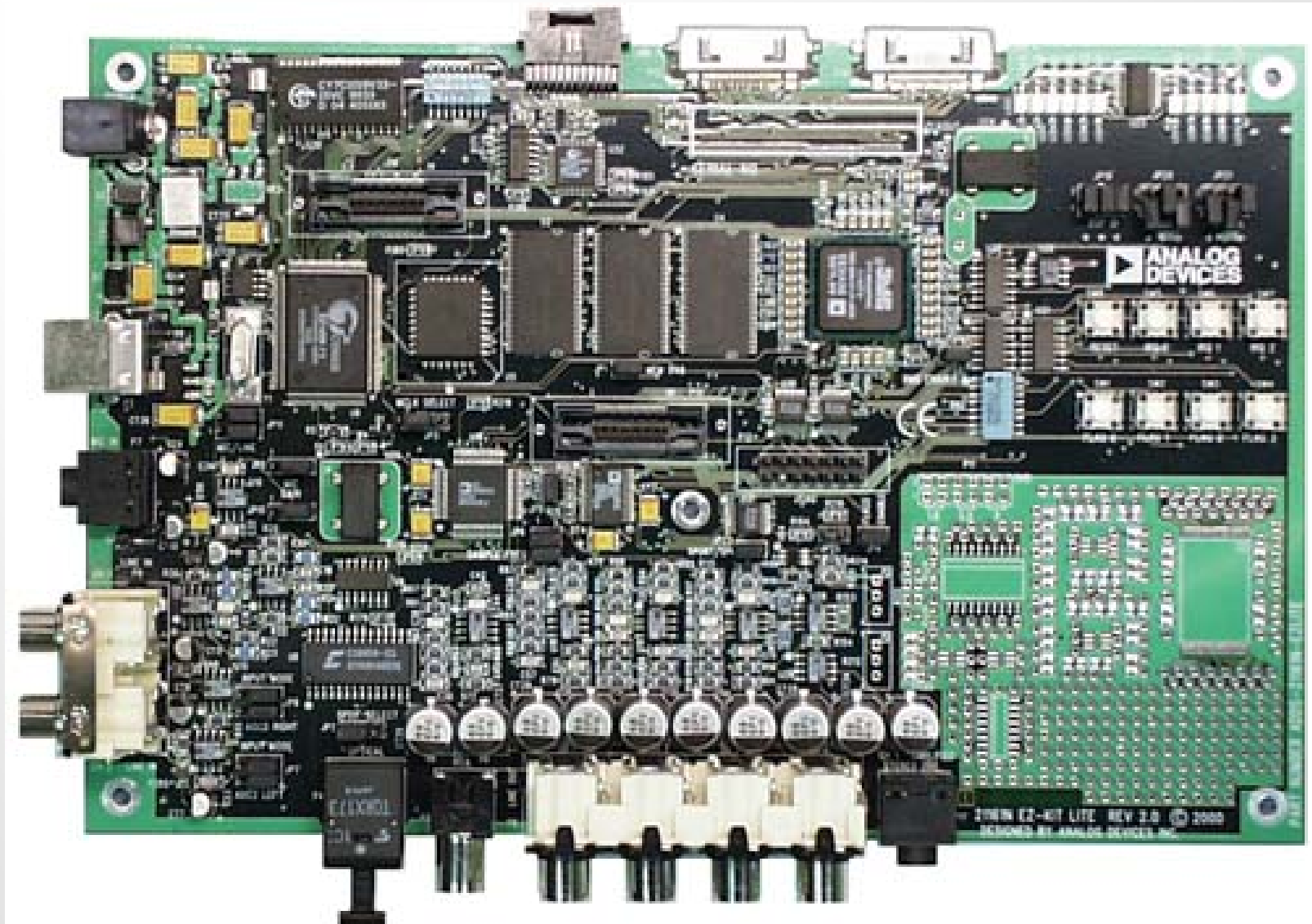
- **ADSP-21161N SHARC DSP - praca na 100 MHz**
- **Pamiec**
  - 1 M-bit (pamiec on-chip)
  - 1M x 48 SDRAM - praca na 100 MHz
  - 512K x 8-bit pamiec Flash
- **Mikrokontroler Cypress CY7C6403 EZ-USB z portem interfejsu JTAG**
- **AD1836 Codec (24-bit, 96 kHz, 4 ADCs, 6 DACs)**
- **AD1852 Stereo DAC (24-bit, 96 kHz)**
- **Crystal CS8414 24-bit, odbiornik 96 kHz SP/DIF**
- **1 Stereo Microphone I/P Jack, 1 Line-In RCA I/P Jack, 8 RCA O/P Jacks, wejscie Optical/RCA do wejscia cyfrowego audio**
- **EZ-ICE JTAG zlacze emulacji**
- **Zlacza rozbudowujace (EP, SPORTs, Link Ports)**
- **6 LEDow wyjsciowych, 4 wejsciove przyciski FLAG, 3 przyciski IRQ**

# 21161 EZ-KIT Lite

## Schemat blokowy



# ADSP-21161N EZ-KIT-LITE



# ADSP-21161 EZ-KIT Lite

## Programy demonstracyjne

- **Talk Through: Podstawowy 1836 codec audio Assemblera i C**
- **Drum Machine Demos: odtwarza pogłos z probkami tablicy kształtów używając Timera i funkcji generatora losowego brzęku**
- **FFT: rzeczywisty w czasie wykres wyjściowego widma audio**
- **SIMD FIR: przykłady pojedynczych i wielokanałowych gorno/dolnoprzepustowych filtrów FIR**
- **SIMD IIR Graphic Equalizer w C**
- **Pluck (“Stairway”): oparta na C synteza ciągów muzycznych**
- **Peter Gunn Theme: Assemblerowa synteza ciągów muz.**
- **Carnival Song: synteza muzyczna przez szeregową aproksymację fali sinusoidalnej Taylor'a**
- **Przykłady syntezy AM/FM**
- **Efekty audio (zniekształcenia, opóźnienie/echo, opóźnienia pogłosu, chor, tremolo, wibracje)**

# ADSP-21161 EZ-KIT-Lite USB Debugger

- **Wlasciwosci Debuggera VisualDSP++:**
  - Debugowanie z poziomu C i Assemblera
  - Wykonanie z pelna predkoscia
  - Pojedynczy krok
  - Programowe punkty wstrzymania
  - Podglad i modyfikacja rejestrow i pamieci
  - Wyrazenia
  - Okno wykresu pamieci z ciagla aktualizacja
  - Standardowe funkcje IO, np. printf(), itp. (nie czasu rzeczywistego)

# Wewnętrzne Emulatory (ICE)



# Wewnętrzne Emulatory

- Umożliwiają debugowanie na poziomie hardware'u
- Wspomagają większość właściwości uruchamiania co Symulator
- Dostęp do systemu TigerSHARC poprzez szeregowy JTAG interfejs
- Działanie nie-ingerencyjne
- Rozległy zakres punktów wstrzymania programowych i sprzętowych
- Wsparcie multiprocessora
- Środowisko VisualDSP Debugging
- Wsparcie poziomu debugowania źródeł C/C++
- ICE Hardware
  - PCI            --            HPPCI-ICE™
  - USB           --            USB-ICE & HP USB-ICE
- Sterowniki oprogramowania ICE dołączone są do VisualDSP++
  - Pełna wersja

# HPPCI-ICE

## Emulator PCI dla Analog Devices JTAG DSP

Nowy wysokiej jakości emulator PCI oferuje prędkość przesyłania kodu do 2.2 MB/sec z taktowaniem JTAG pięć razy szybszym niż jego poprzednia wersja z taką samą zdolnością do zamiany danych czasu rzeczywistego z hosta do aplikacji docelowej. Emulator PCI daje możliwość rozwiązania szybkiej emulacji dla Analog Devices (ADI) JTAG DSP najwyższego poziomu.

Nowy wysokiej jakości emulator PCI składa się z ekranowanego uchwytu i kabla umożliwiającego uruchomienie interfejsu we wszystkich ADI JTAG DSP. Emulator HP PCI automatycznie wykrywa napięcia 1.8V, 2.5V, 3.3V i 5.0V wskazywane na wyświetlaczu LED. Kable ciągnie się przez 6 stop od hosta PC do uchwytu emulatora, i 1 stop od uchwytu do docelowego DSP. Nowy bardzo długi kabel ciągnący się do emulatora, umożliwia zmniejszenie nieporządku w laboratorium sprzętowym.



**Numer Seryjny:** ADDS-HPPCI-ICE

# HPPCI-ICE

- **Wlasciwosci:**
  - Plug-n-Play, weryfikacja zgodnosci PCI 2.2
  - Pomoc multiple emulator
  - Wspomaganie napiecia wielo-DSP I/O
  - Zgodnosc i tolerancja 1.8V, 2.5V, oraz 3.3V
  - 5V tolerant and 3.3V compliant for 5V DSPs
  - Praca zegara JTAG do 50 MHz
  - Wsparcie multiprocesora
- **Wymagania systemu:**
  - Windows® 98, Windows NT, Windows 2000, lub Windows XP
  - Procesor Intel (lub porownywalny) 166MHz
  - Monitor VGA i karta video color
  - 50MB miejsca na dysku
  - 32 MB RAM
  - Jeden dostepny slot PCI

# Emulator USB oraz Wysokiej jakosci emulator USB

Ekonomiczny emulator USB oraz wysokiej szybkości wysokiej jakości (HP) emulator USB umożliwia przenosne, nie-ingerencyjne rozwiązania debugowania dla procesorów ADI JTAG i DSP. Te łatwe w obsłudze emulatory USB spełniają funkcje emulacji szerokiego zakresu, włącznie z krokowym i pełnej prędkości wykonaniem z pre-definiowanymi breakpoint'ami, i podglądem i/lub zmianą zawartości rejestrów i pamięci. Emulatory USB i HP USB umożliwiają użytkownikowi komunikacje ze wszystkimi procesorami ADI JTAG używającymi pełnej prędkości USB 1.0 lub szybkich USB 2.0 portów hosta PC. Aplikacje i dane mogą być łatwo (i szybko, gdy emulator HP USB połączony jest z portem high speed USB 2.0 hosta PC użytkownika) testowane i przekazywane pomiędzy emulatorami i oddzielnie dostępnym środowiskiem VisualDSP++™.



**Numery seryjne:**

**USB-Based Emulator – ADDS-USB-ICE**

**HP USB-Based Emulator – ADDS-HPUSB-ICE**

# USB-Based Emulator and HP USB-Based Emulator - Features

- Pomoc do wszystkich procesorow ADI JTAG i DSP
- Wspomaganie napiecia I/O milti procesorow i DSP I/O z automatycznym wykrywaniem
  - **zgodnosc i tolerancja 1.8V, 2.5V, i 3.3V**
  - **tolerancja dla 5V i zgodnocs 3.3V dla procesorow 5V i DSP**
- Wspomaganie multiprocesora
- Zlacze 14-pin JTAG
- 3-metrowy kabel USB do celow trudnodostepnych
- Poswiadczenie CE
- *USB-Based Emulator* –Zlacze i interfejs USB pelnej predkosci
- *HP USB-Based Emulator* –Zlacze i szybki interfejs USB 2.0 (kompatybilne z interfejsem USB 1.0 pelnej predkosci)
- *HP USB-Based Emulator* – JTAG clock operation up to 50 MHz

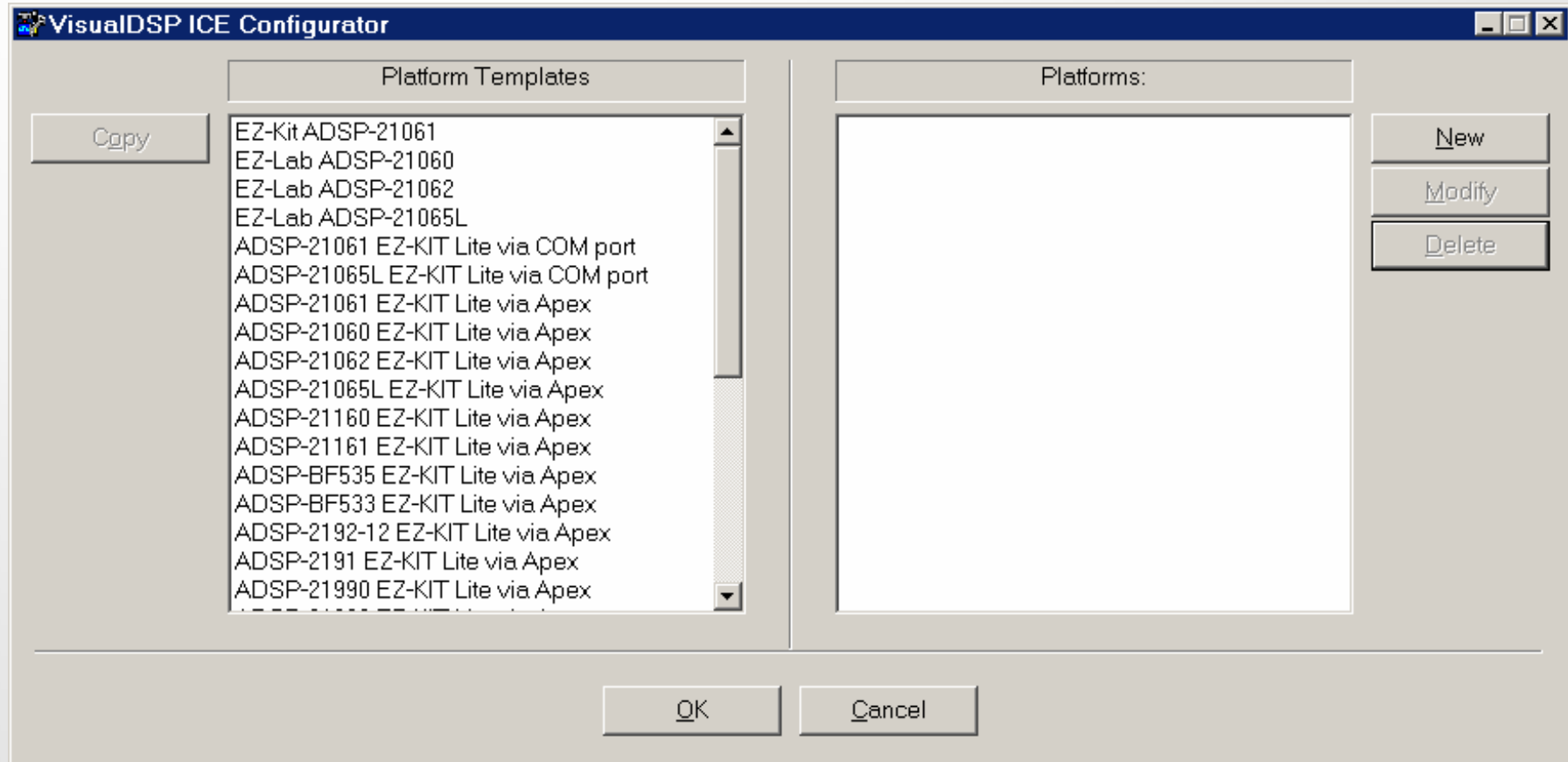
# Konfigurowanie ICE

# Konfiguracja JTAG ICE

- **Konfiguracja priorytetu ICE**
- **Identyfikacja platformy procesora w konfiguracji ICE**
- **Ustawienie typu ICE (np. HPPCI, itp)**

# Konfiguracja JTAG ICE

## Okno dialogowe Platform Templates





# Konfiguracja JTAG ICE

## Okno dialogowe Platform Properties

**Platform Properties** [?] [X]

**Platform**

Name:

Type:

Description:

Base Address (Hex):

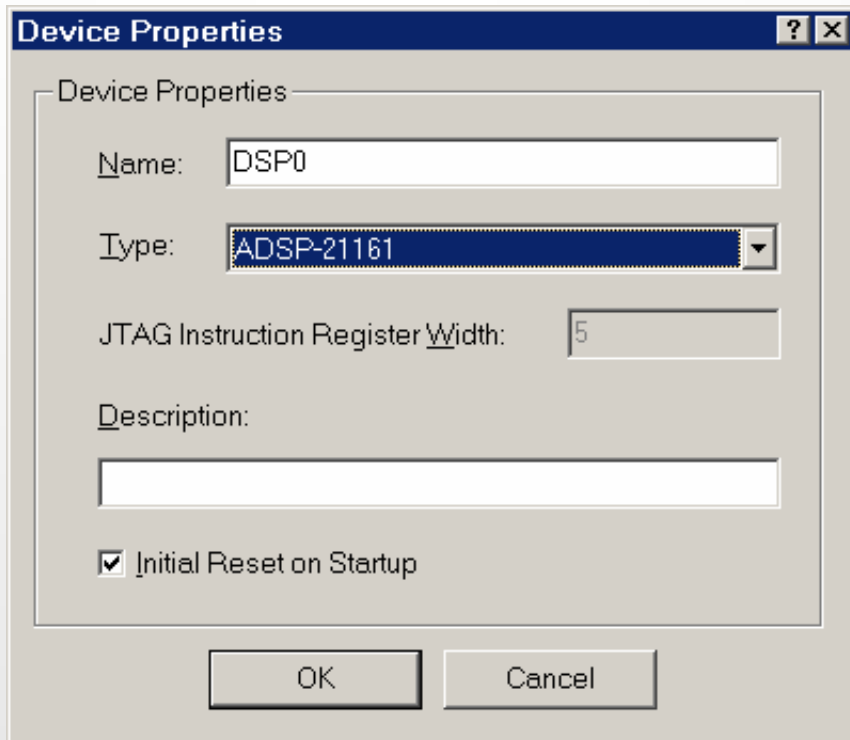
**Devices**

Devices listed in sequential order from TD0 to TDI

TD0	Device #0	<input type="button" value="New"/>
		<input type="button" value="Modify"/>
		<input type="button" value="Delete"/>
		<input type="button" value="Up"/>
		<input type="button" value="Down"/>
TDI		

# Konfiguracja JTAG ICE

## Okno dialogowe Device Properties



The screenshot shows the 'Device Properties' dialog box for a device named 'DSP0'. The 'Name' field contains 'DSP0'. The 'Type' dropdown menu is set to 'ADSP-21161'. The 'JTAG Instruction Register Width' is set to '5'. The 'Description' field is empty. The 'Initial Reset on Startup' checkbox is checked. The dialog has 'OK' and 'Cancel' buttons at the bottom.

Device Properties

Name: DSP0

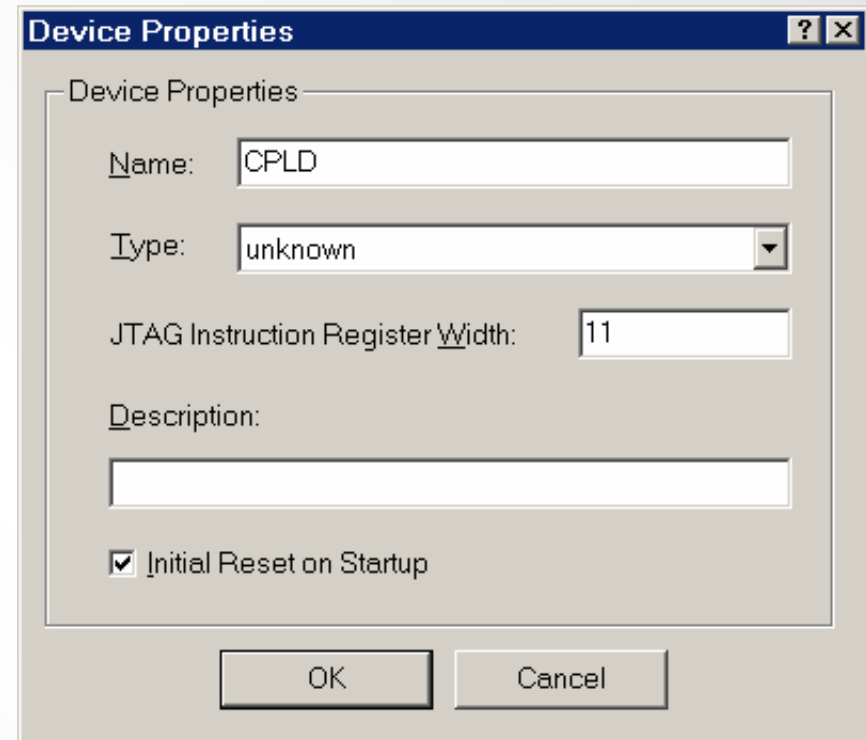
Type: ADSP-21161

JTAG Instruction Register Width: 5

Description:

Initial Reset on Startup

OK Cancel



The screenshot shows the 'Device Properties' dialog box for a device named 'CPLD'. The 'Name' field contains 'CPLD'. The 'Type' dropdown menu is set to 'unknown'. The 'JTAG Instruction Register Width' is set to '11'. The 'Description' field is empty. The 'Initial Reset on Startup' checkbox is checked. The dialog has 'OK' and 'Cancel' buttons at the bottom.

Device Properties

Name: CPLD

Type: unknown

JTAG Instruction Register Width: 11

Description:

Initial Reset on Startup

OK Cancel

# Konfiguracja JTAG ICE

## Okno dialogowe Platform Properties

The screenshot shows the 'Platform Properties' dialog box. It is divided into two main sections: 'Platform' and 'Devices'. The 'Platform' section contains fields for Name, Type, Description, and Base Address (Hex). The 'Devices' section contains a list of devices in sequential order from TD0 to TDI, with buttons for New, Modify, Delete, Up, and Down. The 'Modify' button is highlighted.

Platform
Name: My21161Target
Type: Apex-ICE
Description: My 21161 Target for APEC ICE
Base Address (Hex): 340

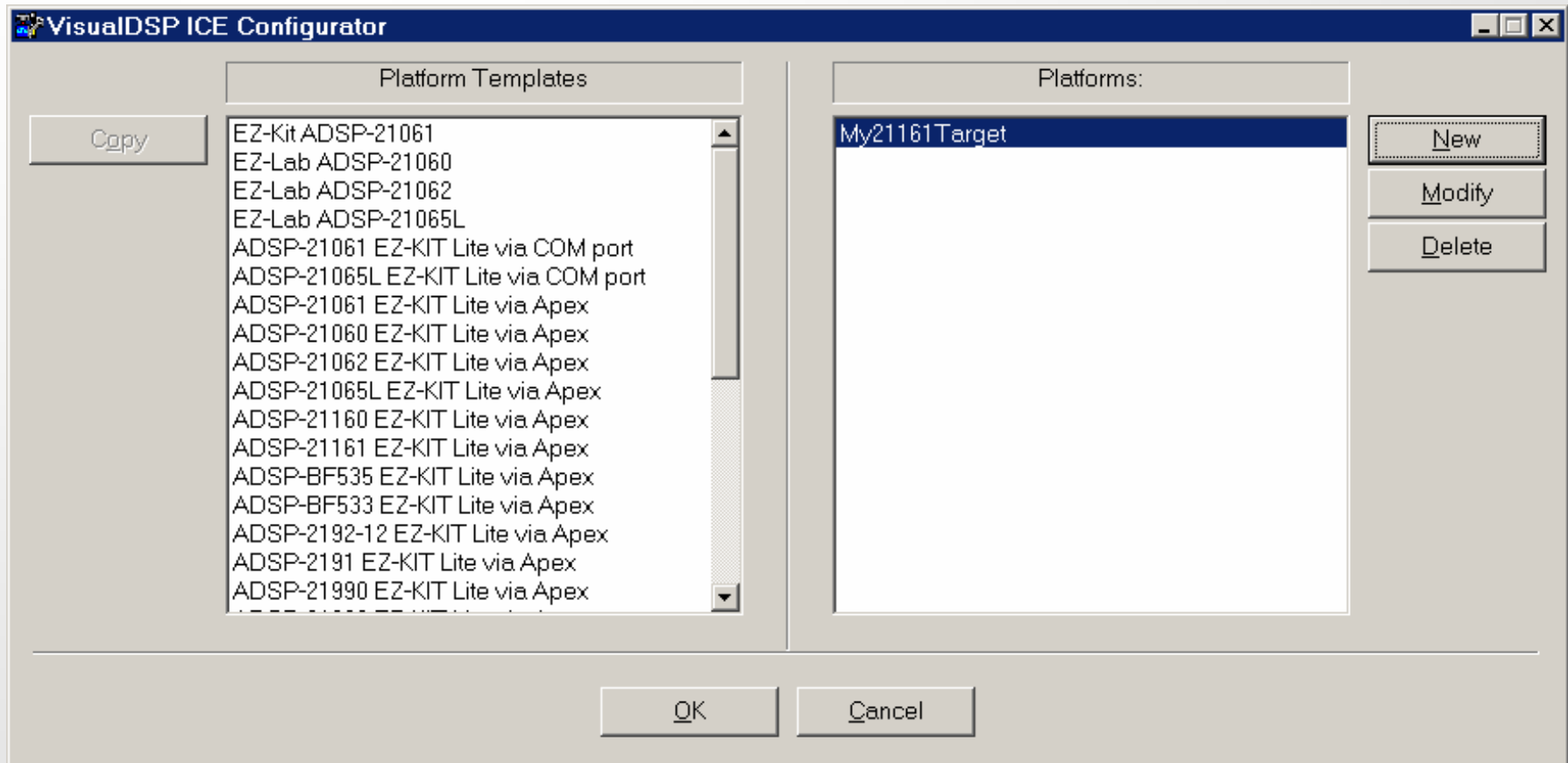
  

Devices	
Devices listed in sequential order from TD0 to TDI	
TD0	CPLD
	DSP0
	DSP1
TDI	

Buttons: New, Modify, Delete, Up, Down, OK, Cancel

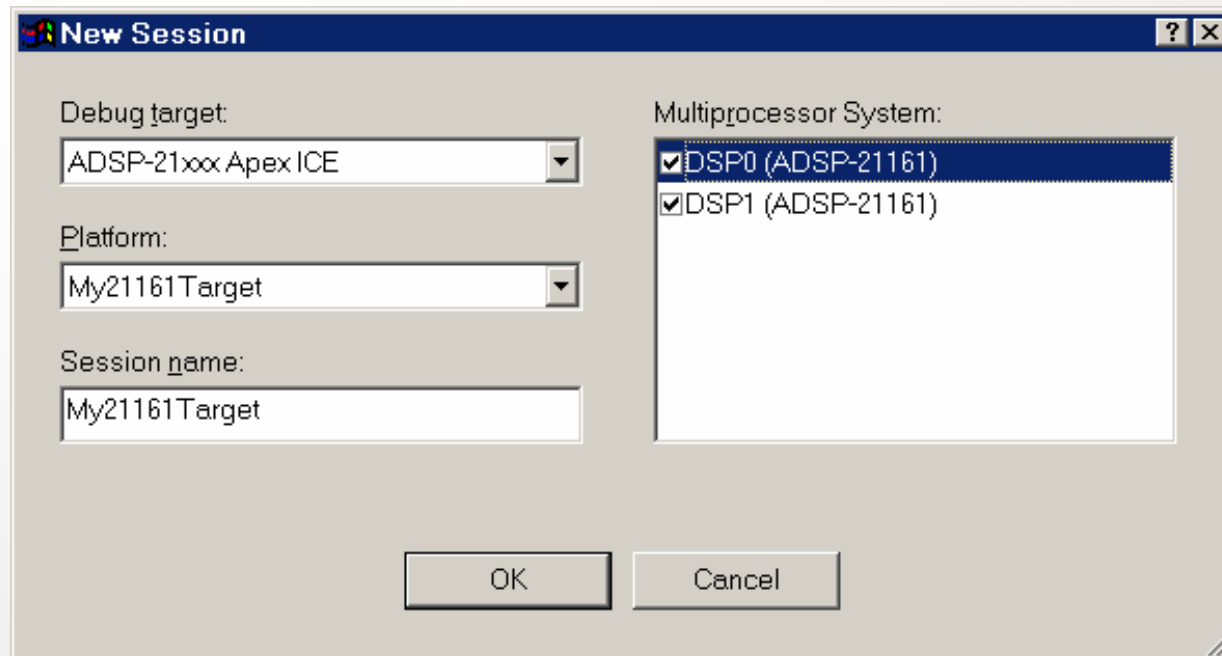
# Konfiguracja JTAG ICE

## Okno dialogowe Platform Templates



# VisualDSP++ IDDE

## Okno dialogowe dla sesji emulatora



# Pomoc ICE Multiprocesora

- **SHARC VisualDSP++ umożliwia:**
  - **Wybor kart indeksowych SHARC**
  - **Breakpoint może spowodować wstrzymanie dowolnego SHARC**
  - **Działanie jednocześnie dowolnego SHARC**
  - **Uruchmienie jednocześnie dowolnego SHARC**

# GUI multiprocesora

The screenshot displays the Analog Devices VisualDSP Debugger interface, showing a multi-processor environment with two processors (P0 and P1) running the same C program. The interface is divided into several panes:

- Source Code (P0: PRIMES.C):** Shows the C code for calculating the first twenty primes. The current execution point is at the start of the `main` function, where `int index = 0;` is highlighted.
- Disassembly (P0: Disassembly):** Shows the assembly instructions corresponding to the source code. The instruction `[00020117] r6=0;` is highlighted, indicating the current instruction being executed.
- System Control Registers (P1: System Control Registers):** Displays the status of various system registers, including `SYSCON`, `SYSTAT`, `MSGRO`, `MSGRI`, `MSGRR`, `MSGRS`, `MSGRE`, and `MSGRW`.
- Active Register File (P1: Active Register File):** Shows the current values of registers R0 through R7. For example, R0 is `0000000200` and R8 is `0002811C00`.

The status bar at the bottom indicates that the processor is halted (P1: Halted) at instruction 20117. The system clock shows 10:30 AM.

# Ladowanie multiprocesora

The screenshot displays the Analog Devices VisualDSP Debugger interface. A 'Load Multiprocessor Confirmation' dialog box is open in the center, listing two processors:

Processor	Program File Name
P0	C:\Program Files\Analog Devices\VisualDSP\21k\exam...
P1	C:\Program Files\Analog Devices\VisualDSP\21k\exam...

The background interface includes several windows:

- P0: Active Register File:** Lists registers R0 through R15 with their current values.
- P0: System Control Registers:** Shows various control registers like SYSICON, srst, bso, iivt, iwt, hpm, hmswf, hpflsh, MSGR0-7, pagsz, pageis, mmsvs, hidma, and ubw.
- P1: Disassembly:** Shows assembly code for processor P1, including instructions like nop, jump, bit clr, and dm.
- P0: Disassembly:** Shows assembly code for processor P0, including instructions like nop, jump, bit clr, and dm.

The status bar at the bottom indicates 'Ready' and 'P0: Unknown'.



# Menu Debug Multiprosesora

The screenshot displays the Analog Devices VisualDSP Debugger interface for a dual-processor target (ADSP-21062). The main window shows the source code for a program named 'PRIMES.C' running on two processors, P0 and P1. The code implements a prime number finding algorithm. The debugger is currently in a state where the 'Multiprosesora' menu is open, showing options like 'Run', 'Reset', and 'Restart'. Below the source code, the disassembly for both processors is visible, showing instructions such as register assignments, arithmetic operations, and control flow. At the bottom, a table shows the state of the processors: P0 and P1 are both 'Halted'.

**Processor Status Table:**

Processor	State
P0	Halted
P1	Halted

## Wybor sprzętowych (danych) punktów wstrzymania

**Hardware Breakpoints: DSP 0**

Data | Instruction | Other

Data Breakpoints

Enable	Start Address	End Address	Exclusive	Mode
<input type="checkbox"/> Breakpoint 1:	FFFFFFFF	00000000	<input type="checkbox"/>	Disabled
<input type="checkbox"/> Breakpoint 2:	FFFFFFFF	00000000	<input type="checkbox"/>	Disabled

Global Breakpoint Options

Skip N Breakpoint Events:

- Reset Skip Count on Initialization
- Restore Skip Count on Break
- AND All Breakpoints

OK Cancel

# Wybor sprzętowych (instrukcji) punktów wstrzymania

**Hardware Breakpoints: DSP 0**

Data | **Instruction** | Other

Instruction Breakpoints

Enable	Start Address	End Address	Exclusive
<input type="checkbox"/> Breakpoint 1	FFFFFFF	000000	<input type="checkbox"/>
<input type="checkbox"/> Breakpoint 2	FFFFFFF	000000	<input type="checkbox"/>
<input type="checkbox"/> Breakpoint 3	FFFFFFF	000000	<input type="checkbox"/>
<input type="checkbox"/> Breakpoint 4	FFFFFFF	000000	<input type="checkbox"/>

Global Breakpoint Options

Skip N Breakpoint Events:

Reset Skip Count on Initialization  
 Restore Skip Count on Break  
 AND All Breakpoints

OK Cancel

# Wybor sprzętowych (innych) punktów wstrzymania

**Hardware Breakpoints: DSP 0**

Data | Instruction | **Other**

Other Hardware Breakpoints

Enable	Start Address	End Address	Exclusive	Mode
<input type="checkbox"/> PM Data	FFFFFF	000000	<input type="checkbox"/>	Disabled
<input type="checkbox"/> I/O	01FFFF	000000	<input type="checkbox"/>	Disabled
<input type="checkbox"/> External Port	03FFFFFF	00000000	<input type="checkbox"/>	Disabled

Global Breakpoint Options

Skip N Breakpoint Events:

- Reset Skip Count on Initialization
- Restore Skip Count on Break
- AND All Breakpoints

OK Cancel

## Dozwolone punkty przerwania sprzetowego

Typ breakpoint	Access Type								
	Core			DMA			Slave (MMS or Host)		
	Int. RAM	Ext.	IOP Reg.	Int. RAM	Ext.	IOP Reg.	Int. RAM	Ext.	IOP Reg.
Dane DM (2-1)	Y	Y	Y						
Dane PM (1)	Y	Y	Y						
I/O (1)				Y			Y*		
Port zewn. (1)		Y			Y				
Instrukcja (4-1)	Y	Y	Y						

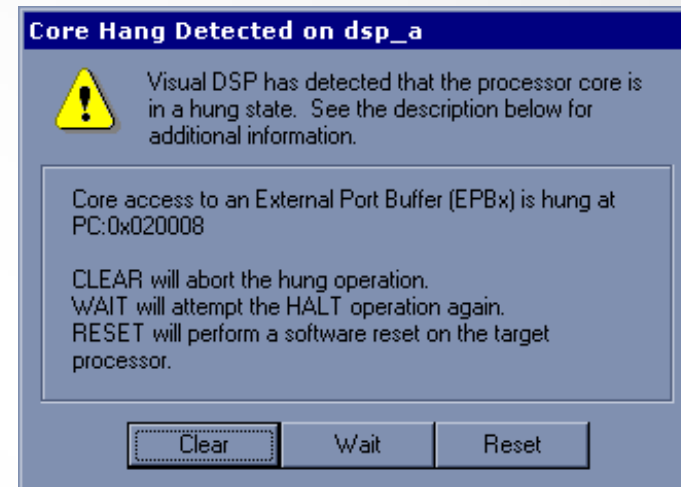
**Adresy punktów przerwania wewnątrz pamięci wewnętrznej procesora muszą być “bazowymi” adresami pamięci wewnętrznej, nie utożsamione z adresem MMS.**

# Przypadki zawieszenia rdzenia

- **ICE wymaga kontroli rdzenia SHARC przy operacjach**
  - Dolaczenie do celu
  - Zatrzymanie na zadanie uzytkownika
  - Zatrzymanie w punkcie wstrzymania (breakpoint)
  - Pojedynczy krok
- **Zawieszenie rdzenia moze byc spowodowane przez Sport, Link, Bufor EP i dostep do lokalizacji zewnetrznego portu**

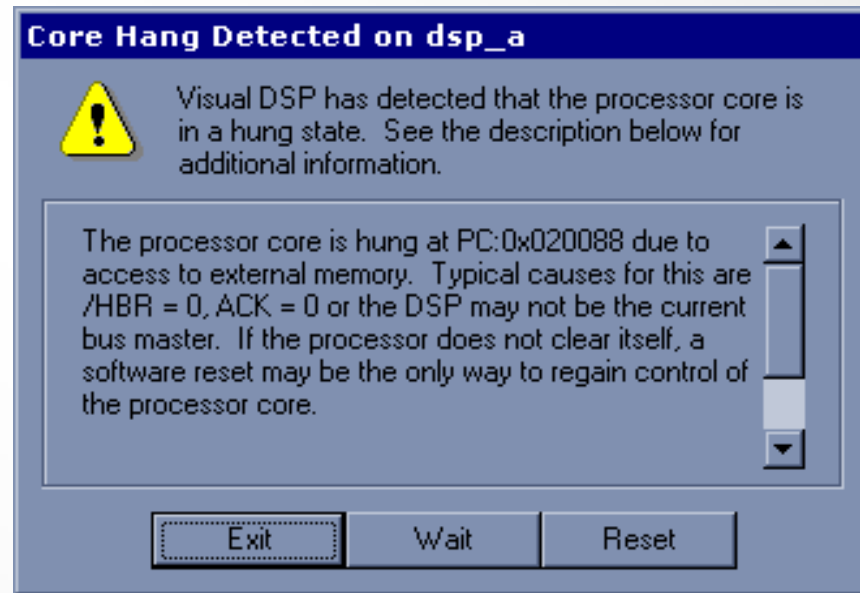
```
R0=DM(4); //Powoduje zawieszenie jesli bufor EPB0 jest pusty
```

# Zawieszenie rdzenia przy dostępie do bufora IOP



- Tego typu zawieszenie rdzenia występuje gdy:
  - Rdzen zapisuje do *Full Sport*, Link, lub bufora EP
  - Rdzen czyta z *Empty Sport*, Link, lub bufora EP

## Zawieszenie rdzenia przy dostępie portu zewnętrznego



- Występuje gdy pojawi się Rdzeń Odczyt/Zapis Adresu Portu Zewnętrznego oraz którakolwiek z sytuacji:
  - piny ACK or SBTS są w stanie niskim
  - inny SHARC jest Szyną Master
  - Host jest Szyną Master
- ICE musi zresetować procesor w celu wymuszenia czystego zawieszenia



# Narzedzia sprzetowe - cwiczenia

# 21161 EZ-KIT Lite / JTAG ICE Cwiczenie 1

## Mrugajacy LED

Napisz program na płytce 21161 EZ-KIT Lite, który będzie mrugał flag6 i flag8 LEDow 1 raz na sekunde. Użyj timera i flag pinow wyjsciowych. Utworz plik uruchomieniowy i użyj emulatora do wgrania kodu na płytke 21161 EZ-KIT lite.

Warianty programu:

- A) Napisz program zamiennie zapalajacy dwa LEDy na płytce EZ-KIT Lite z czestosciwoscia 1 Hz.
- B) Użyj push-button do zmiany czestosciwosci zapalania.
- C) Użyj push-button do *ustawienia* czestosciwosci. Nacisnij push-button przy czestotliwosci z jaka chcesz aby LEDy swiecily.

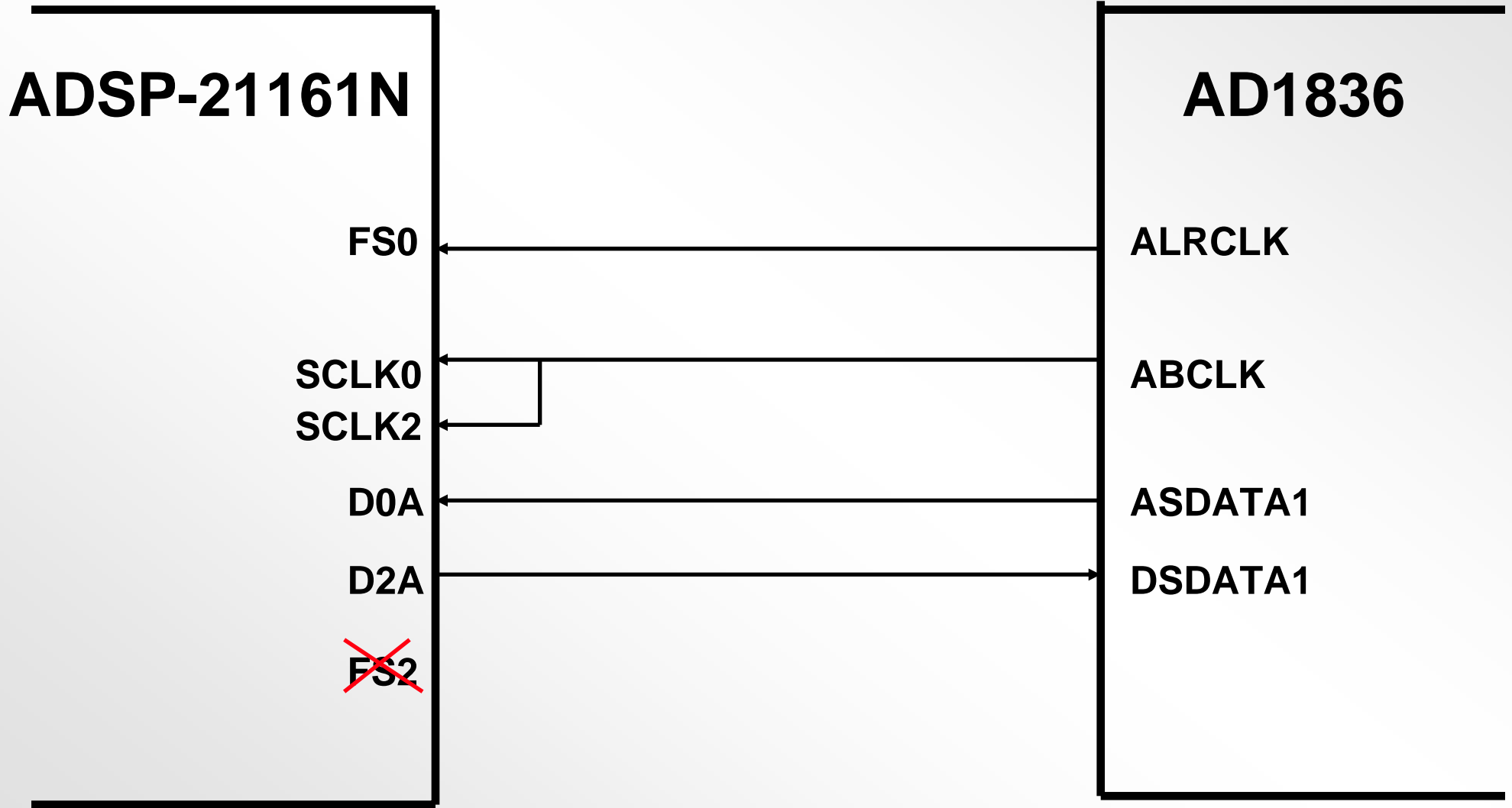
## Cwiczenia 2 i 3

- **Cwiczenia 2 i 3 wykorzystują wielokanałowy audio codec AD1836.**
  - Przekazywanie głosu
  - Echo/Opoznienie

# Interfejs AD1836 Audio

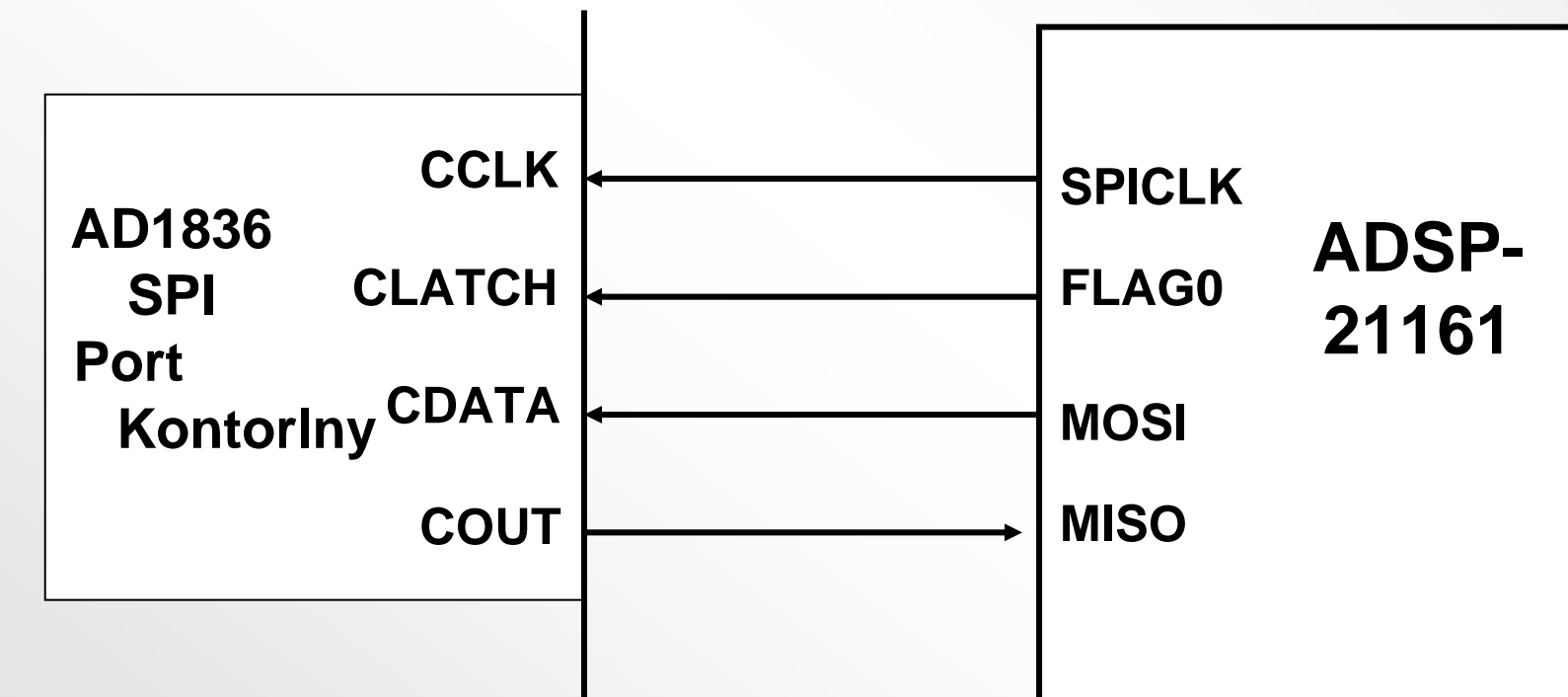
- **AD1836 jest wysokiej jakości, jedno-chipowym CODEC'iem dostarczającym 2 stereo I<sup>2</sup>S ADC i 3 stereo I<sup>2</sup>S DAC**
- **AD1836 ADCs/DACs obsługujący słowa danych 24-bitowe i 96 kHz próbkowania spełnia najnowsze wymagania wysokiej jakości audio.**
- **ADC i DAC posiadają 105 dB współczynnik S/N i zakres dynamiczny (15 do 30 dB wyższa jakość sygnału niż w codeck'ach komputerowych)**
- **AD1836 zawiera 6, niezależnych regulatorów głośności sterowanych przez kompatybilny szeregowy port kontrolny SPI.**
- **Dedykowany interfejs Phillips I<sup>2</sup>S do wszystkich ADC i DAC**
- **Schemat TDM pozwala na łatwy interfejs do SHARC używając tylko 1 SPORT**

# Interfejs ADSP-21161/AD1836 Szeregowego TDM

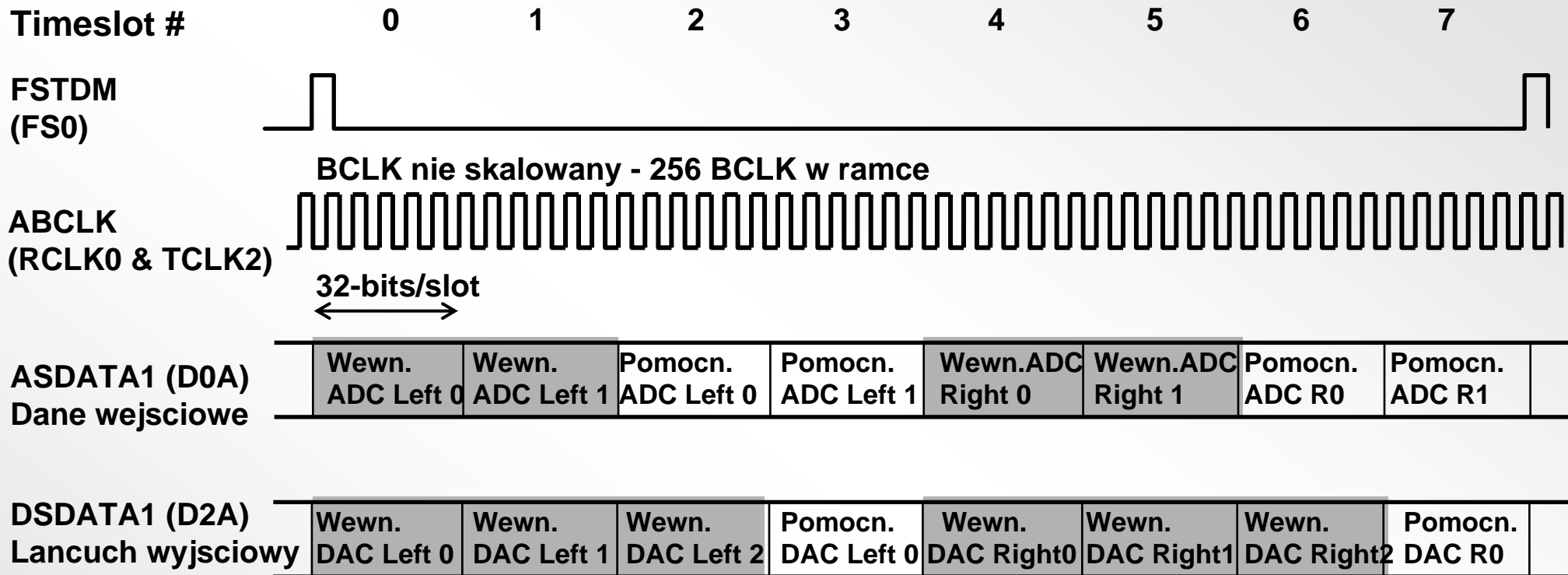


# Port Kontrolny AD1836 SPI łączy z portem ADSP-21161 SPI

- ADSP-21161 zawiera port SPI.
- ADSP-21161 jest programowany jako urządzenie SPI Master



# Protokół AD1836/ADS-21161 TDM



 = Szare pola wskazują na konwersje wewnętrznych źródeł do AD1836

FS2 (TVD2) nie połączony w trybie wielokanałowym

Dla 21161, zegary szeregowo są wewnętrznie podłączone w trybie wielokanałowym.

# Bufor SPORT0 Odbioru DMA

rx\_buf0a[8]

Timeslot #

rx_buf + Internal_ADC_L0
rx_buf + Internal_ADC_L1
rx_buf + AUX_ADC_L0
rx_buf + AUX_ADC_L1
rx_buf + Internal_ADC_R0
rx_buf + Internal_ADC_R1
rx_buf + AUX_ADC_R0
rx_buf + AUX_ADC_R1

0

1

2

3

4

5

6

7

```
.segment /dm dm_codec;
```

```
.var rx_buf0a[8]; //
```

```
Bufor SPORT0  
odbioru DMA
```

```
.endseg;
```

```
/*Def. AD1836 TDM Timeslot */
```

```
#define Internal_ADC_L0 0
```

```
#define Internal_ADC_L1 1
```

```
#define AUX_ADC_L0 2
```

```
#define AUX_ADC_L1 3
```

```
#define Internal_ADC_R0 4
```

```
#define Internal_ADC_R1 5
```

```
#define AUX_ADC_R0 6
```

```
#define AUX_ADC_R1 7
```



# Bufor SPORT2 Transmisji DMA

**tx\_buf2a[8]**

tx_buf2a + Internal_DAC_L0
tx_buf2a + Internal_DAC_L1
tx_buf2a + Internal_DAC_L2
tx_buf2a + AUX_DAC_L0
tx_buf2a + Internal_DAC_R0
tx_buf2a + Internal_DAC_R1
tx_buf2a + Internal_DAC_R1
tx_buf2a + AUX_DAC_R0

**Timeslot #**

0

1

2

3

4

5

6

7

```
.segment /dm dm_codec;  
  
.var tx_buf2a[8]; //  
Bufor SPORT2  
transmisji DMA  
  
.endseg;  
  
/*Def. AD1836 TDM Timeslot*/  
#define Internal_DAC_L0 0  
#define Internal_DAC_L1 1  
#define Internal_DAC_L2 2  
#define AUX_DAC_L0      3  
#define Internal_DAC_R0 4  
#define Internal_DAC_R1 5  
#define Internal_DAC_R2 6  
#define AUX_DAC_R0      7
```

# Tworzenie lancucha TCB SPORT0&2 DMA (automatyczna transmisja do D0A & D2A)

Deklaracja zmiennych TCB:

```
.var rcv0a_tcb[8]=0,0,0,0,rcv0a_tcb+7,8,1,rx_buf0a; /* odbiorczy tcb SPT0 */
.var xmit2a_tcb[8]=0,0,0,0,xmit2a_tcb_7,8,1,tx_buf2a; /* nadawczy tcb SPT2 */
```

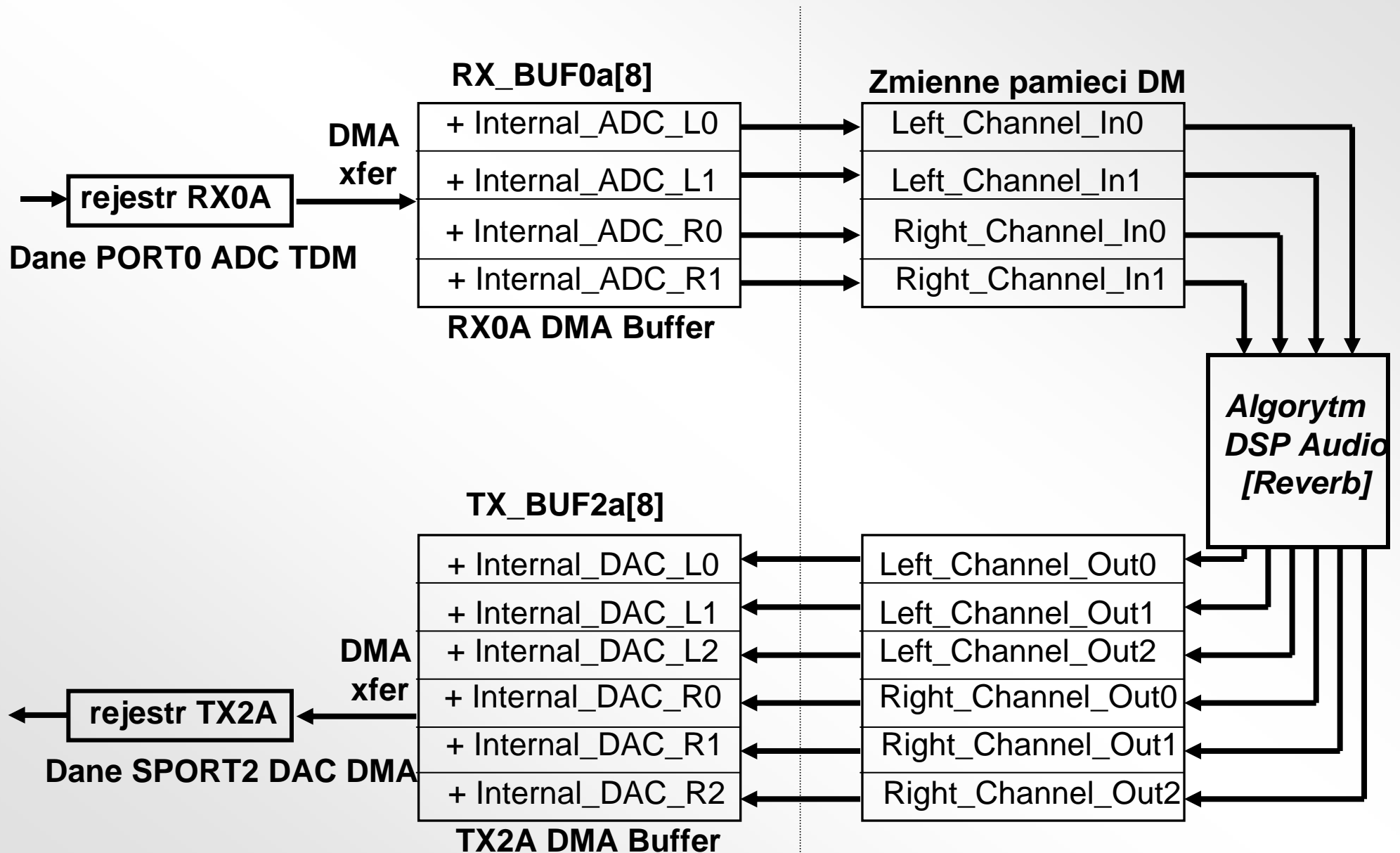
**rcv0a\_tcb[8]**

DM(rcv0a_tcb + 3)	Nie uzywany
DM(rcv0a_tcb + 4)	bit PCI i Adres [rcv1a_tcb + 7]
DM(rcv0a_tcb + 5)	8
DM(rcv0a_tcb + 6)	1
DM(rcv0a_tcb + 7)	Adres startowy rx_buf0a

**xmit2a\_tcb[8]**

DM(xmit2_tcb + 3)	Nie uzywany
DM(xmit2a_tcb + 4)	bit PCI i Adres [xmit1a_tcb + 7]
DM(xmit2a_tcb + 5)	8
DM(xmit2a_tcb + 6)	1
DM(xmit2a_tcb + 0)	Adres startowy tx_buf2a

# Struktura obsługi wywołania SPORT0 SR & DSP



# Interfejs oprogramowania AD1836 Codec (1)

- **Szeregowe Porty 0 & 2 oraz AD1836 sa inicjowane poprzez wywołanie “AD1836\_Codec\_Initialization” w “211xx\161\Talkthru\AD1836\_initialization”**
  - Wybierany jest współczynnik próbkowania AD1836 (domyslnie 48KHz)
  - AD1836 (wybor wejscowego jack'a mic oraz zworki JP11 ustawione na mic)
  - Sport 0 i SPORT2 ustawione do ramki 8 Kanalowego TDM
  - DMA TCB ustawione na Sport 0 RX DMA do "rx\_buf0a[8]"
  - DMA TCB ustawione na Sport 2 TX DMA z "tx\_buf2a[8]"

# Interfejs oprogramowania AD1836 Codec (2)

- **Zoacz przykład \211xx\workshop\talkthru “talkthru.asm”**
  - Sport 0 w konfiguracji odbiornika, Przerywa kazdy okres probkowania i skacze do “process\_audio”
  - Przetwarzanie probek codec'a moze byc wykonane wewnatrz funkcji lub procedury poprzez wywołanie “process\_audio” w pliku talkthru.asm.
- **Przykład przekazywania:**

```
process_audio:  
    r0 = DM(Left_Channel_In0); /* pobierz probke wejsciowa lewego ADC0 AD1836 */  
    DM(Left_Channel_Out0) = r0; /* wyslij lewa probke do AD1836 DAC0 */
```
- **Przetwarzanie moze byc wykonane na r0 przed przekazaniem na wyjcie.**

# Konwersja próbek AD1836 24-bitowego z/na 32-bitowe IEEE zmiennopozycyjne

- Obliczenia na typach zmiennopozycyjnych wymagają konwersji na próbki ADC/CA staloprzecinkowe
- Wartości zmiennoprzecinkowe są ponadto skalowane do zakresu +/-1.0 za pomocą konwersji integer na float ( $f0 = \text{float } r0 \text{ by } r1$ )...

```
r1 = -31;                /* skalowanie próbek do zakresu +/-1.0 */
r0 = DM(Left_Channel_In0); /* pobranie próbki z kanału lewego ADC0 AD1836 */
r2 = DM(Right_Channel_In0); /* pobranie próbki z kanału prawego ADC0 AD1836 */

f0 = float r0 by r1;     /* konwersja na typ zmiennopozycyjny */
f2 = float r2 by r1;

/* wstawienie algorytmu przetwarzania zmiennopozycyjnego */
call (pc, dsp_float_algorithm);

r1 = 31;                /* skalowanie wyniku spowrtoem do MSB */
r0 = fix f0 by r1;      /* konwersja powrotna na typ staloprzecinkowego */
r2 = fix f2 by r1;

DM(Left_Channel_Out0) = r0; /* wysłanie lewego wyniku do lewego DAC0 AD1836*/
DM(Right_Channel_Out0) = r2; /* wysłanie prawego wyniku do prawego DAC0 AD1836*/
```

# **EZ- KIT/Emulator Cwiczenie 2**

## **Przekazywanie głosu i Echo**

- 1) W języku assembler, użyj programu TALKTHRU.ASM do próbkowania swojego głosu zapisu do pamięci i odtworzenia głosu w głośniku. Wskazówki: Wejście mikrofonu próbkowane jest przez AD1836, który jest podłączony do SPORT0 i SPORT2 w trybie wielokanalowym.**
- 2) Zmodyfikuj napisany program przekazywania głosu aby utworzyć program echo. Użyj buforów cyklicznych do symulacji linii opóźnienia echa. Utwórz echo 1-sekundowe.**

### **Możliwości:**

- A) Dodaj sprzężenie zwrotne do programu echo.**
- B) Spróbuj odłuchać buforów cyklicznych w różnych miejscach i podsumuj wyniki.**
- C) W programie próbkującym głos, spróbuj odtwarzania próbek bufora w odwrotnej kolejności.**
- D) Spróbuj zmienić (zmodulować) współczynnik odtwarzanych próbek.**