ARINC 429 Bus Interface

DirectCore

Product Summary

Intended Use

- ARINC 429 Transmitter (Tx)
- ARINC 429 Receiver (Rx)

Key Features

- Supports ARINC Specification 429-16
- Configurable up to 16 Rx and 16 Tx Channels
- Programmable FIFO Depth
  - Up to 512 Words
- Programmable Interrupt Generation
  - Rx and Tx Channels independently
  - Up to 64 Words
- Configurable Label Memory Size
  - Rx and Tx Channels independently
  - Up to 256 Words
- Internal, Wrap-Around Testing
- Software Compatible with Legacy Devices
- Selectable Clock Speed
  - 1, 10, 16, or 20 MHz
- Selectable Data Rate on Each Channel
  - 12.5 100 kbps
  - Optional 50 kbps
- CPU Interface
  - Provides Direct CPU Access to Memory
  - Simple Interface to Core8051
- Memory
  - EDAC Support with RTAX-S Family
- ARINC 429 Bus Interface
  - Supports Standard Line Drivers and Receivers
- Available as Integrated Tx and Rx

Supported Families

- Fusion
- ProASIC®3/E
- ProASIC PLUS®
- Axcelerator®
- RTAX-S

Core Deliverables

- Evaluation Version
  - Compiled RTL Simulation Model, Compliant with the Actel Libero® Integrated Design Environment (IDE)
- Netlist Version
  - Structural VHDL and Verilog Netlists
- RTL version
  - VHDL or Verilog Core Source Code
  - Synthesis Scripts
- Verification Testbench – Verilog
- User Testbenches
  - Libero IDE Compatible
  - VHDL and Verilog

Development System

- Complete ARINC 429 Rx/Tx
- Implementation
  - Implemented in an APA600 Device
  - Controlled Via an External Terminal Using Core8051 and RS232 Links
- Includes Line Driver and Receiver Components

Synthesis and Simulation Support

- Directly Supported within the Actel Libero IDE
- Synthesis:
  - Synplicity®
  - Exemplar™
  - Synopsys®
- Simulation
  - Vital-Compliant VHDL Simulators
  - OVI-Compliant Verilog Simulators

Verification and Compliance

- Actel-Developed Simulation Testbench
- Core Implemented on the ARINC 429 Development System

Contents

General Description

Core429 provides a complete Transmitter (Tx) and Receiver (Rx). A typical system implementation using Core429 is shown in Figure 1.

The core consists of three main blocks: Transmit, Receive, and CPU Interface (Figure 1). Core429 requires connection to an external CPU. The CPU interface configures the transmit and receive control registers and initializes the label memory. The core interfaces to the ARINC 429 bus through an external ARINC 429 line driver and line receiver. A detailed description of the Rx interface and Tx interface is provided in the "Functional Description" section on page 5.

External Components

There are two external components required for proper operation of Core429:

- Standard ARINC 429 line driver
- Standard ARINC 429 line receiver

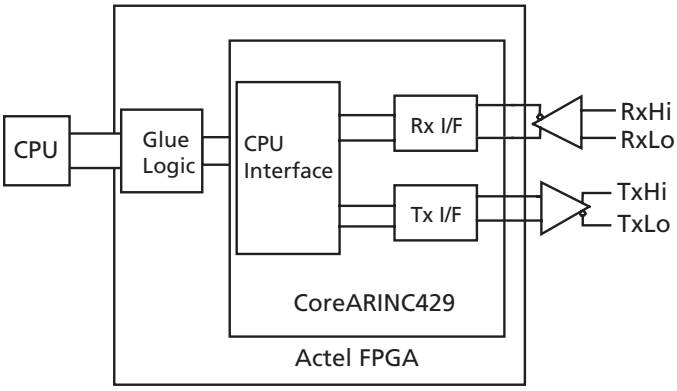


Figure 1 • **Typical Core429 System—One Tx and One Rx**

ARINC 429 Overview

ARINC 429 is a two-wire, point-to-point data bus that is application-specific for commercial and transport aircraft. The connection wires are twisted pairs. Words are 32 bits in length and most messages consist of a single data word. The specification defines the electrical standard and data characteristics and protocols.

ARINC 429 uses a unidirectional data bus standard (Tx and Rx are on separate ports) known as the Mark 33 Digital Information Transfer System (DITS). Messages are transmitted at 12.5, 50 (optional), or 100 kbps to other system elements that are monitoring the bus messages. The transmitter is always transmitting either 32-bit data words or the Null state.

The ARINC standard supports High, Low, and Null states (Figure 2). A minimum of four Null bits should be transmitted between ARINC words. No more than 20 receivers can be connected to a single bus (wire pair) and no less than one receiver, though there will normally be more.

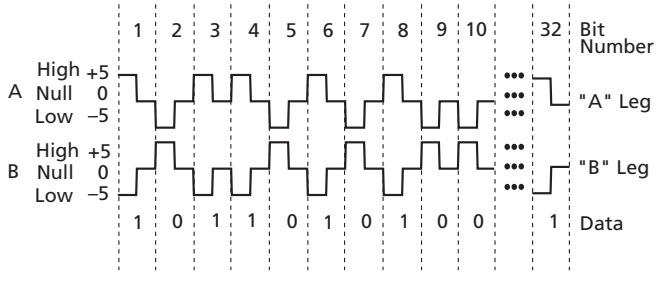


Figure 2 • **ARINC Standard**

Figure 3 on page 3 shows the bit positions of ARINC data.

Each ARINC word contains five fields:

- Parity

- Sign/Status Matrix
- Data
- Source/Destination Identifiers
- Label

The parity bit is bit 32 (the MSB). SSM is the Sign/Status Matrix and is included as bits 30 and 31. Bits 11 to 29 contain the data. Binary Coded Decimal (BCD) and binary encoding (BNR) are common ARINC data formats. Data formats can also be mixed. Bits 9 and 10 are Source/Destination Identifiers (SDI) and indicate for which receiver the data is intended. Bits 1 to 8 contain a label (label words) identifying the data type.

Label words are quite specific in ARINC 429. Each aircraft may be equipped with different electronic equipment and systems needing interconnection. A large amount of equipment may be involved, depending on the aircraft. The ARINC specification identifies the equipment ID, a series of digital identification numbers. Examples of equipment are Flight Management Computers, Inertial Reference Systems, Fuel Tanks, Tire Pressure Monitoring Systems, and GPS Sensors.

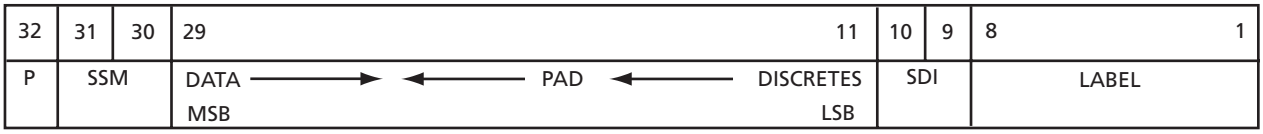


Figure 3 • **ARINC Data Bit Positions**

Transmission Order

The least significant bit of each byte, except the label, is transmitted first, and the label is transmitted ahead of the data in each case. The order of the bits transmitted on the ARINC bus is as follows:

8, 7, 6, 5, 4, 3, 2, 1, 9, 10, 11, 12, 13 … 32.

Core429 Device Requirements

Core429 can be implemented in several Actel FPGA devices. Table 1 through Table 5 on page 4 provide typical utilization figures using standard synthesis tools for different Core429 configurations. Table 1 assumes that the label size is set to 64 and FIFO depth is set to 64.

Table 1 • **Device Utilization for One Tx Module**

Family	Cells or Tiles			Memory Blocks	Device	Utilization
	Combinatorial	Sequential	Total			
Fusion	363	147	510	1	AFS600	4%
ProASIC3/E	363	147	510	1	A3PE600	4%
ProASIC^{PLUS}	441	146	587	1	APA075	19%
Axcelerator	212	145	357	1	AX125	18%
RTAX-S	258	171	429	1	RTAX250S	10%

Table 2 • **Device Utilization for One Rx Module**

Family	Cells or Tiles			Memory Blocks	Devices	Utilization
	Combinatorial	Sequential	Total			
Fusion	431	233	664	2	AFS600	5%
ProASIC3/E	431	233	664	2	A3PE600	5%
ProASIC^{PLUS}	588	236	824	2	APA075	27%
Axcelerator	307	234	541	2	AX125	27%
RTAX-S	350	259	609	2	RTAX250S	14%

Table 3 • **Device Utilization for One Rx and One Tx Module**

Family	Cells or Tiles			Memory Blocks	Device	Utilization
	Combinatorial	Sequential	Total			
Fusion	848	609	1,457	3	AFS600	10%
ProASIC3/E	848	609	1,457	3	A3PE600	10%
ProASIC^{PLUS}	1,084	377	1,461	3	APA075	48%
Axcelerator	518	378	896	3	AX125	44%
RTAX-S	604	429	1,033	3	RTAX250S	24%

Table 4 • **Device Utilization for 16 Rx and 16 Tx Modules**

Family	Cells or Tiles			Memory Blocks	Device	Utilization
	Combinatorial	Sequential	Total			
Fusion	13,435	9,614	23,049	48	AFS1500	60%
ProASIC3/E	13,435	9,614	23,049	48	A3PE1500	60%
ProASIC^{PLUS}	16,835	5,928	22,763	48	APA750	69%
Axcelerator	8,044	5,944	13,988	48	AX2000	43%
RTAX-S	9,594	6,745	16,339	48	RTAX2000S	51%

Table 5 • **Device Utilization for Legacy Mode (2 Rx and 1 Tx)**

Family	Cells or Tiles			Memory Blocks	Device	Utilization
	Combinational	Sequential	Total			
Fusion	1,444	1,068	2,512	5	AFS600	18%
ProASIC3/E	1,444	1,068	2,512	5	A3PE600	18%
ProASIC^{PLUS}	1,840	674	2,514	5	APA150	41%
Axcelerator	955	653	1,608	5	RTAX250S	20%
RTAX-S	1,062	729	1,791	5	RTAX250S	42%

Core429 clock rate can be programmed to be 1, 10, 16, or 20 MHz. All the Actel families listed above easily meet the required performance.

Core429 I/O requirements depend on the system requirements and the external interfaces. If the core and memory blocks are implemented within the FPGA and the CPU interface has a bidirectional data bus, then approximately 74 I/O pins are required to implement four Rx and four Tx modules. The core will require 62 pins to implement one Rx and one Tx module.

The core has various FIFO flags available for debugging purposes. These flags may not be needed in the final design and this will reduce the I/O count.

Memory Requirements

The number of memory blocks required differs, depending on whether each channel is configured the same or differently.

Each Channel Configured the Same

Use EQ 1 to calculate the number of memory blocks required if each channel is configured the same.

Number of memory blocks = NRx * (INT (LABEL_SIZE/X) + INT (RX_FIFO_DEPTH/Y) + NTx * INT (FIFO_DEPTH/Y),

EQ 1

where NRx is the number of receive channels, NTx is the number of transmit channels, INT is the function to round up to the next integer, and X and Y are defined in Table 6.

Each Channel Configured Differently

Use EQ 2 to calculate the number of memory blocks required if each channel is configured differently.

$$\text{Number of memory blocks} = \sum_{I=0}^{NTx-1} \text{INT}(\text{FIFO\_DEPTH}[I]/Y + \sum_{I=0}^{NRx-1} (\text{INT}(\text{LABEL\_SIZE}[I]/X) + \text{INT}(\text{FIFO\_DEPTH}[I]/Y)),$$

EQ 2

where NRx is the number of receive channels, NTx is the number of transmit channels, INT is the function to round up to the next integer, and X and Y are defined in Table 6.

Table 6 • **Memory Parameters**

Device Family	X	Y
Fusion	512	128
ProASIC3/E	512	128
ProASIC^{PLUS}	256	64
Axcelerator/RTAX-S	512	128

Examples for the ProASIC3/E Device Family

If the design has 2 receivers, 1 transmitter, 64 labels for each receiver, 32-words-deep FIFO for each receiver and transmitter, then

the number of memory blocks = 2 * (INT (64/512) + INT (32/128)) + 1 * INT (32/128) = 2 * (1 + 1) + 1 * (1) = 5.

If the design has 2 receivers, 1 transmitter, 32 labels for receiver # 1, 64 labels for receiver # 2, 32 words-deep FIFO for receiver # 1, 64-words-deep FIFO for receiver # 2, and 64-words-deep FIFO for transmitter, then

the number of memory blocks = INT (64/128) + (INT (32/512) + INT (32/128)) + (INT (64/512) + INT (64/128))

= 1 + (1 + 1) + (1 + 1) = 5.

Core429 Overview

Core429 provides a complete and flexible interface to a microprocessor and an ARINC 429 data bus. Connection to an ARINC 429 data bus requires additional line drivers and line receivers.

Core429 interfaces to a processor through the internal memory of the receiver. Core429 can be easily interfaced to an 8-, 16- or 32-bit data bus. Look-up tables loaded into memory enable the Core429 receive circuitry to filter and sort incoming data by label and destination bits. Core429 supports multiple (configurable) ARINC 429 receiver channels, and each receives data independently. The receiver data rates (high or low speed) can be programmed independently. Core429 can decode and sort data based on the ARINC 429 Label and SDI bits and stores it in FIFO. Each receiver uses programmable FIFO to buffer received data. Core429 supports multiple (configurable) ARINC 429 transmit channels and each channel can transmit data independently.

Default Mode

This is the recommended mode and allows the user to configure the core with user-defined transmit and receive channels.

Functional Description

The core has three main blocks: Transmit, Receive, and CPU interface. The core can be configured to provide up to 16 transmit and receive channels.

Figure 4 gives a functional description of the Rx block.

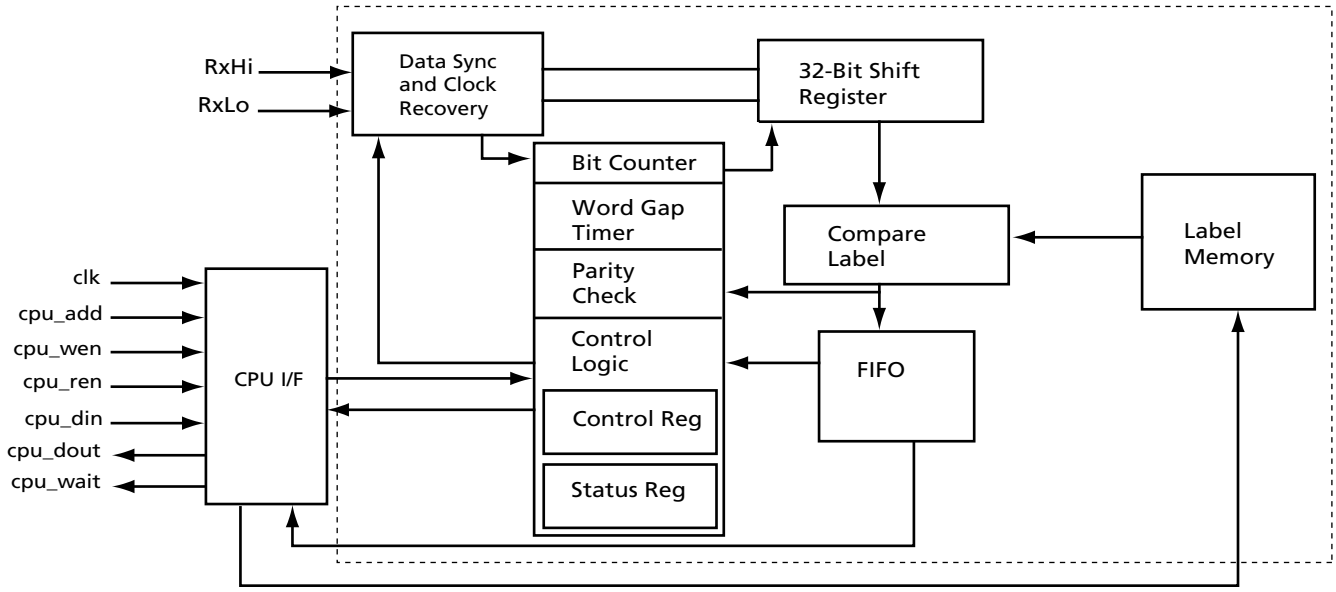


Figure 4 • **Core429 Rx Block Diagram**

The Rx block is responsible for recovering the clock from the input serial data and performs serial-to-parallel conversion and gap/parity check on the incoming data. It also interfaces with the CPU.

The Rx module contains two 8-bit registers. One is used for control function and the other is used for status. Refer to Table 14 on page 11 and Table 15 on page 11 for detailed descriptions of the control and status register bits. The CPU interface configures the internal RAM with the labels, which are used to compare against the incoming labels from the received ARINC data.

If the label-compare bit in the receive control register is enabled, then the data which matches its labels with the stored labels will be stored in the FIFO. If the label-compare bit in the receive control register is disabled, then the incoming data will be stored in the FIFO without comparing against the labels in RAM.

The core supports reloading label memory using bit 7 of the Rx control register. Note that when you set bit 7 to initialize the label memory, the old label content still exists, but the core keeps track only of the new label and does not use the old label during label compare.

The FIFO asserts three status signals:

- rx_fifo_empty: FIFO is empty
- rx_fifo_half_full: FIFO is filled up to the programmed RX_FIFO_LEVEL
- rx_fifo_full: FIFO is full

Depending on the FIFO status signals, the CPU will either read the FIFO before it overflows, or not attempt to read the FIFO if it is empty. The interrupt signal int_out_rx is generated when one of the FIFO status signals (rx_fifo_empty, rx_fifo_half_full, and rx_fifo_full) are high.

Figure 5 gives a functional description of the Tx block.

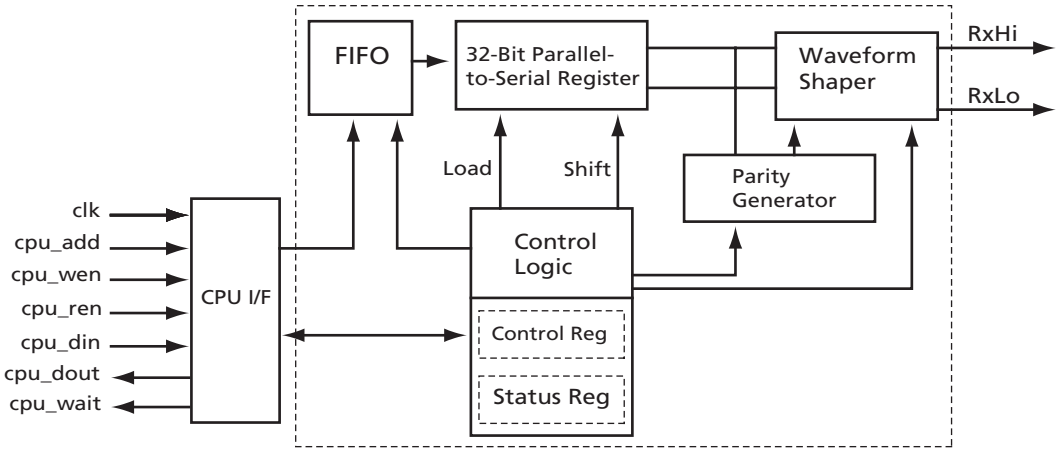


Figure 5 • **Core429 Tx Block Diagram**

The Tx module converts the 32-bit parallel data from the TX FIFO to serial data. It also inserts the parity bit into the ARINC data when parity is enabled. The CPU interface is used to fill the FIFO with ARINC data. The TX FIFO can hold up to 512 ARINC words of data. The transmission starts as soon as one complete ARINC word has been stored in the transmit FIFO.

The Tx module contains two 8-bit registers. One is used for a control function and the other is used for status. The CPU interface allows the system CPU to access the control and status registers within the core.

The TX FIFO asserts three status signals:

- tx_fifo_empty: TX FIFO is empty
- tx_fifo_half_full: TX FIFO is filled up to the programmed TX_FIFO_LEVEL
- tx_fifo_full: TX FIFO is full

Depending on the FIFO status signals, the CPU will either read the FIFO before it overflows, or not attempt to read the FIFO if it is empty. The interrupt signal int_out_tx is generated when one of the FIFO status signals (tx_fifo_empty, tx_fifo_half_full and tx_fifo_full) are high.

Legacy Mode

In this mode, there is a legacy interface block that communicates with the CPU interface. When legacy mode is enabled, the core supports two receive (Rx) channels and one transmit (Tx) channel only. This is not configurable.

Core Parameters

Core429 has several top-level Verilog parameters (VHDL generics) that are used to select the number of channels and FIFO sizes of the core that is implemented. Using these parameters allows the size of the core to be reduced when all the channels are not required.

For RTL versions, the parameters in Table 7 can be directly set. For netlist versions of the core, a netlist implementing four Tx and four Rx channels is provided as per the defaults above. Actel will supply netlists with alternative parameter settings on request.

Table 7 • **FIFO and Label Parameters**

Parameter Name	Description	Allowed Values	Default
CLK_FREQ	Clock Frequency	1, 10, 16, 20 MHz	1 MHz
CPU_DATA_WIDTH	CPU Data Bus Width	8, 16, 32 bits	8
RXN	Rx Channels	1 to 16	4
TXN	Tx Channels	1 to 16	4
LEGACY_MODE	0 = Normal mode; 1 = Legacy mode	0,1	0
LABEL_SIZE_i	Number of Labels for Rx Channel i	1 to 256	64
RX_FIFO_DEPTH_j	Depth of FIFO for Rx Channel j ARINC word	32, 64, 128, 256, 512	32
RX_FIFO_LEVEL_k	FIFO Level for Rx Channel k	1 to 64	16
TX_FIFO_DEPTH_l	Depth of FIFO for Tx Channel l ARINC word	32, 64, 128, 256, 512	32
TX_FIFO_LEVEL_m	FIFO Level for Tx Channel m	1 to 64	16
TXRXSPEED_n	When this parameter is set to '1', a bit rate of 100/50 kbps is selected. Otherwise selects a bit rate of 100/12.5 kbps. The bit rate can be changed for the Rx/Tx channel pair. Refer to the Tx and Rx control register bit descriptions in Table 14 on page 11 and Table 18 on page 12.	0, 1	0

Note: Where i, j, k, l, m, and n are from 0 to 15.

I/O Signal Descriptions

ARINC Interface

Table 8 • **Clock and Reset**

Name	Type	Description
clk	In	Master clock input (1, 10, 16, or 20 Mhz)
reset_n	In	Active low asynchronous reset
txa [TXN-1:0]	Out	ARINC transmit output A
txb [TXN-1:0]	Out	ARINC transmit output B
rxa [RXN-1:0]	In	ARINC receiver input A
rxb [RXN-1:0]	In	ARINC receiver input B

Default Mode Signals

Table 9 • **Core Interface Signals**

Name	Type	Description
int_out_rx[RXN-1:0]	Out	Interrupt from each receive channel. This interrupt is generated when one of the following conditions occur: • FIFO Empty • FIFO Full • FIFO is full up to the programmed RX_FIFO_LEVEL This is an active high signal.
int_out_tx[TXN-1:0]	Out	Interrupt from each transmit channel. This interrupt is generated when one of the following conditions occur: • FIFO Empty • FIFO Full • FIFO is full up to the programmed TX_FIFO_LEVEL This is an active high signal.
rx_fifo_full[RXN-1:0]	Out	RX FIFO full flag for each receive channel. This is an active high signal.
rx_fifo_half_full[RXN-1:0]	Out	RX FIFO programmed level flag for each receive channel. By default it is programmed to half full. This is an active high signal.
rx_fifo_empty[RXN-1:0]	Out	RX FIFO empty flag for each receive channel.This is an active high signal.
tx_fifo_full[TXN-1:0]	Out	TX FIFO full flag for each transmit channel. This is an active high signal.
tx_fifo_half_full[TXN-1:0]	Out	TX FIFO programmed level flag for each transmit channel. By default it is programmed to half full. This is an active high signal.
tx_fifo_empty[TXN-1:0]	Out	TX FIFO empty flag for each transmit channel. This is an active high signal.

CPU Interface

The CPU interface allows access to the Core429 internal registers, FIFO, and internal memory. This interface is synchronous to the clock.

Table 10 • **CPU Interface Signals**

Name	Type	Description
cpu_ren	In	CPU read enable, active low
cpu_wen	In	CPU write enable, active low
cpu_add [8:0]	In	CPU address
cpu_din [CPU_DATA_WIDTH-1:0]	In	CPU data input
cpu_dout [CPU_DATA_WIDTH-1:0]	Out	CPU data output
int_out	Out	Interrupt to CPU, active high. int_out is the OR function of int_out_rx and int_out_tx.
cpu_wait	Out	Indicates that the CPU should hold cpu_ren or cpu_wen active while the core completes the read or write operation.

Legacy Interface

The Legacy interface allows access to the Core429 internal registers, FIFO, and internal memory. This interface is synchronous to the clock. The Tx module contains two 8-bit registers. One is used for control function and the other is used for status.

Table 11 • **Legacy Interface Signals**

Name	Type	Description
data_ready1	Out	Receiver 1 data ready (FIFO not empty) flag
fifo_full1	Out	Receiver 1 FIFO full
half_full1	Out	Receiver 1 FIFO half full
data_ready2	Out	Receiver 2 data ready (FIFO not empty) flag
fifo_full2	Out	Receiver 2 FIFO full
half_full2	Out	Receiver 2 FIFO half full
transmit_fifo_full	Out	Transmit FIFO full
transmit_half_full	Out	Transmit FIFO half full
rsel	In	Receiver data half word selection
ctrl_n	In	Clock for control word register
str_n	In	Read status register if rsel = 0, read control register if rsel = 1
entx	In	Enable transmission
txr	Out	Transmitter ready flag. Goes low when ARINC word loaded into FIFO. Goes high after transmission and FIFO empty.
pl1_n	In	Latch enable for word 1 entered from data bus to transmitter FIFO
pl2_n	In	Latch enable for word 2 entered from data bus to transmitter FIFO. Must follow pl1_n.
en1_n	In	Data Bus control, enables receiver 1 data to outputs
en2_n	In	Data Bus control, enables receiver 2 data to outputs if en1_n is high
test	In	Disable transmitter output if high
dout	In/Out	Bidirectional data bus
data_valid	Out	Data is valid when data_valid = 1

Default Mode Operation

In the default mode, the core operates with the following register map.

CPU Address Map

The address bits 0 and 1 are used to create byte indexes.

For an 8-Bit CPU Data Bus:

00 – Byte 0
01 – Byte 1
10 – Byte 2
11 – Byte 3

For a 16-Bit CPU Data Bus:

00 – Lower half word
10 – Upper half word

For 32-Bit CPU Data Bus:

00 – Word

The address bits 2 and 3 select the registers within each Rx or Tx block (see "Address Map" on page 11).

The address bit 4 is used to determine Rx/Tx as follows:

0 – Rx

1 – Tx

The address bits 5, 6, 7, and 8 are used for decoding the 16 channels as follows:

0000 – Channel0

0001 – Channel1

. .

. .

1110 – Channel14

1111 – Channel15

Table 12 shows the CPU address bit information.

Table 12 • **CPU Address Bit Positions**

Channel Number				Tx/Rx	Register Index		Byte Index	
8	7	6	5	4	3	2	1	0
MSB	9-Bit CPU Address							LSB

Register Definitions

Rx Registers

Following is the detailed definition of cpu_add [3:2] decoding and the explanation of Data Register, Control Register, Status Register, and Label Memory Register (Table 13 through Table 16 on page 12).

Address Map

00 – Data Register

01 – Control Register

10 – Status Register

11 – Label Memory

Table 13 • **Rx Data Register**

Bit	Function	Reset State	Type	Description
31:0	Data	0	R	Read Data

Table 14 • **Rx Control Register**

Bit	Function	Reset State	Type	Description
0	Data rate	0	R/W	Data rate: 0 = 100Kb/s; 1 = 12.5 or 50 Kbps
1	Label recognition	0	R/W	Label compare: 0 = disable; 1 = enable
2	Enable 32nd bit as parity	0	R/W	0 = 32nd bit is data; 1 = 32nd bit is parity
3	Parity	0	R/W	Parity: 0 = odd; 1 = even
4	Decoder	0	R/W	0: SDI bit comparison disabled; 1: SDI bit comparison enabled; ARINC bits 9 and 10 must match bits 5 and 6 respectively.
5	Match header bit 9	0	R/W	If bit 4 is set then this bit should match the ARINC header bit 9 (SDI bit).
6	Match header bit 10	0	R/W	If bit 4 is set then this bit should match the ARINC header bit 10 (SDI bit).
7	Reload label memory	0	R/W	When bit 7 is set to '1', label memory address pointers are initialized to '000'. Set this bit to change the contents of the label memory.

Table 15 • **Rx Status Register**

Bit	Function	Reset State	Type	Description
0	FIFO empty	0	R	0 = not empty; 1 = empty
1	FIFO half full or programmed level	0	R	0 = Less than programmed level; 1 = FIFO is filled at least up to programmed level
2	FIFO full	0	R	0 = not full; 1 = full

Table 16 • **Rx Label Memory Register**

Bit	Function	Reset State	Type	Description
7:0	Label	0	R/W	Read/Write Labels

Tx Registers

Following is a detailed definition of cpu_add [3:2] decoding and an explanation of the Data Register, Pattern RAM, Control Register, and Status Register.

Address Map

00 – Data Register

01 – Control Register

10 – Status Register

11 – Unused

Table 17 • **Tx Data Register**

Bit	Function	Reset State	Type	Description
31:0	Data	0	W	Write Data

Table 18 • **Tx Control Register**

Bit	Function	Reset State	Type	Description
0	Data rate	0	R/W	Data rate: 0 = 100Kb/s; 1 = 12.5 or 50 Kbps
1	Loopback	0	R/W	0 = Disable loopback; 1 = Enable loopback
2	Enable 32nd bit as parity	0	R/W	0 = 32nd bit is data; 1 = 32nd bit is parity
3	Parity	0	R/W	Parity: 0 = odd; 1 = even

Table 19 • **Tx Status Register**

Bit	Function	Reset State	Type	Description
0	FIFO empty	0	R	0 = not empty; 1 = empty
1	FIFO half full or programmed level	0	R	0 = Less than half full or programmed level; 1 = Half full or programmed level
2	FIFO full	0	R	0 = not full; 1 = full

Label Memory Operation

The label memory is implemented using an internal memory block. The read address and write address are generated by internal counters. The read and write address counters can be reset by setting bit 7 of the receive (Rx) control register to '1'. The write counter increments each time the label memory register is written. The read counter increments every time the label memory register is read.

The label memory operation is shown in Figure 6.

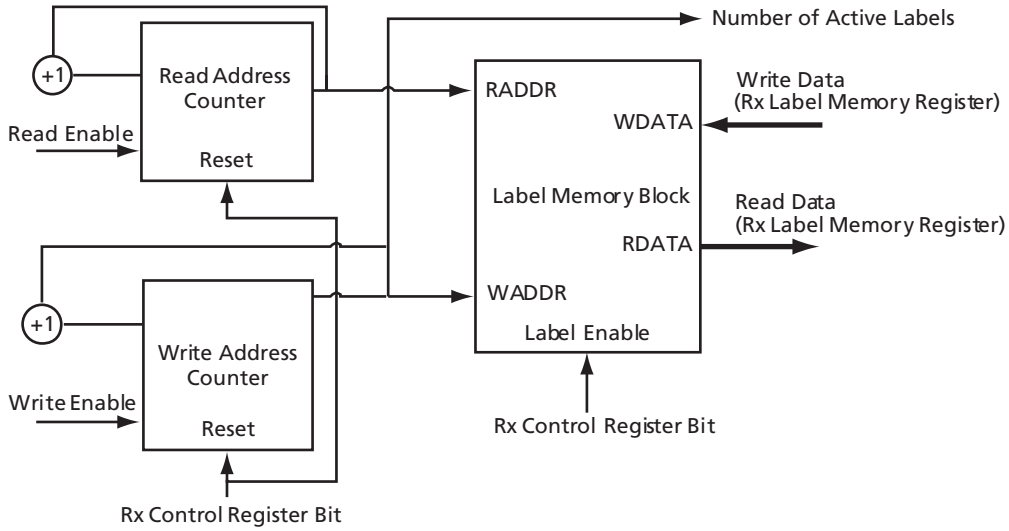


Figure 6 • **Label Memory Diagram**

To program labels, the CPU first resets the read and write counters by setting bit 7 of the receive (Rx) control register to '1'. Then the labels are written to the label memory. The core will compare the incoming ARINC word label (bit 1 to 8 of ARINC word) against the labels contained in the label memory. The contents of the label memory can be read by reading the label memory register. While writing to or reading from label memory, bit 1 of the receive (Rx) control register should be set to '0'.

To reload the label memory, set bit 7 of the receive (Rx) control register to '1'. The core will then ignore all previous labels and new labels can be written to the label memory.

Legacy Operation

In this mode, there is a legacy interface block that communicates with the CPU interface. When legacy mode is enabled, the core supports two receive (Rx) channels and one transmit (Tx) channel only. Legacy mode is not configurable to support multiple transmit and receive channels. The purpose of the legacy mode interface is to replace existing standard products.

Control Register

Core429 contains a 16-bit control register, which is used to configure the Rx and Tx channels. The control register bits 0 to 15 are loaded from the databus when CTRL_n is low. The control register contents are output on the databus when RSEL is high and STR_n is low. Each bit of the control register description is explained in Table 20.

Table 20 • **Legacy Control Register**

Bit	Function	Reset State	Type	Description
0	Receiver 1 data rate	0	R/W	Data rate: 0 = 100 kbps; 1 = 12.5 kbps. Note: Does not support 50 kbps.
1	Label compare	0	R/W	0 = disable; 1 = enable Load 16 labels using pl1_n/pl2_n Read 16 labels using en1_n/en2_n
2	Enable label recognition (Receiver 1)	0	R/W	0: Disable label recognition 1: Enable label recognition
3	Enable label recognition (Receiver 2)	0	R/W	0: Disable label recognition 1: Enable label recognition
4	Enable 32 bit as parity	0	R/W	0 = 32 bit is data; 1 = 32 bit is parity
5	Self test	1	R/W	0: The transmitter's digital outputs are internally connected to the receiver logic inputs. 1: Normal operation
6	Receiver 1 decoder	0	R/W	0: Receiver 1 decoder disabled 1: ARINC bits 9 and 10 must match bits 7 and 8 of the control register.
7	Match ARINC bit 9 (receiver 1)	0	R/W	If receiver 1 decoder is enabled, the ARINC bit 9 should match this bit.
8	Match ARINC bit 10 (receiver 1)	0	R/W	If receiver 1 decoder is enabled, the ARINC bit 10 should match this bit.
9	Receiver 2 decoder	0	R/W	0: Receiver 2 decoder disabled 1: ARINC bits 9 and 10 must match bits 10 and 11 of the control register.
10	Match ARINC bit 9 (receiver 2)	0	R/W	If receiver 2 decoder is enabled, the ARINC bit 9 should match this bit.
11	Match ARINC bit 10 (receiver 2)	0	R/W	If receiver 2 decoder is enabled, the ARINC bit 10 should match this bit.
12	Transmitter parity	0	R/W	Parity: 0 = odd; 1 = even
13	Transmitter data rate	0	R/W	Data rate: 0 = 100 kbps; 1 = 12.5 kbps. Note: Does not support 50 kbps.
14	Receiver 2 data rate	0	R/W	Data rate: 0 = 100 kbps; 1 = 12.5 kbps. Note: Does not support 50 kbps.

Status Register

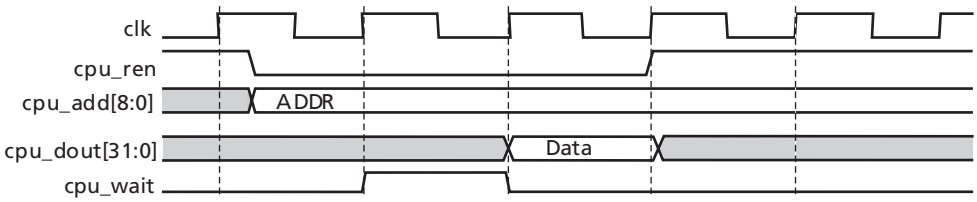
Core429 contains a 16-bit status register which can be read to determine the status of the ARINC receivers, data FIFOs, and transmitter. The contents of the status register are output on the databus when RSEL is low and STR is low. Each bit of the control register description is explained in Table 21.

Table 21 • **Legacy Status Register**

Bit	Function	Reset State	Type	Description
0	Receiver 1 FIFO Empty	0	R	0 = Receiver 1 FIFO not empty 1 = Receiver 1 FIFO empty
1	Receiver 1 FIFO Half Full	0	R	0 = Receiver 1 FIFO not half full 1 = Receiver 1 FIFO half full
2	Receiver 1 FIFO Full	0	R	0 = Receiver 1 FIFO not full 1 = Receiver 1 FIFO full
3	Receiver 2 FIFO Empty	0	R	0 = Receiver 2 FIFO not empty 1 = Receiver 2 FIFO empty
4	Receiver 2 FIFO Half Full	0	R	0 = Receiver 2 FIFO not half full 1 = Receiver 2 FIFO half full
5	Receiver 2 FIFO Full	0	R	0 = Receiver 2 FIFO not full 1 = Receiver 2 FIFO full
6	Transmitter FIFO Empty	0	R	0 = Transmitter FIFO not empty 1 = Transmitter FIFO empty
7	Transmitter FIFO Full	0	R	0 = Transmitter FIFO not full 1 = Transmitter FIFO full
8	Transmitter FIFO Half Full	0	R	0 = Transmitter FIFO not half full 1 = Transmitter FIFO half full

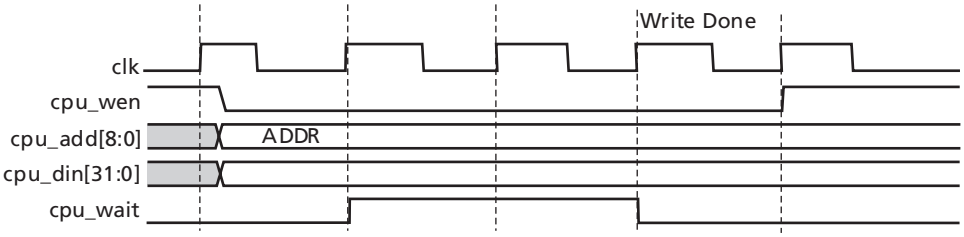
CPU Interface Timing for Default Mode

The CPU interface signals are synchronized to the Core429 master clock. Figure 7 through Figure 12 on page 17 show the waveforms for the CPU interface.



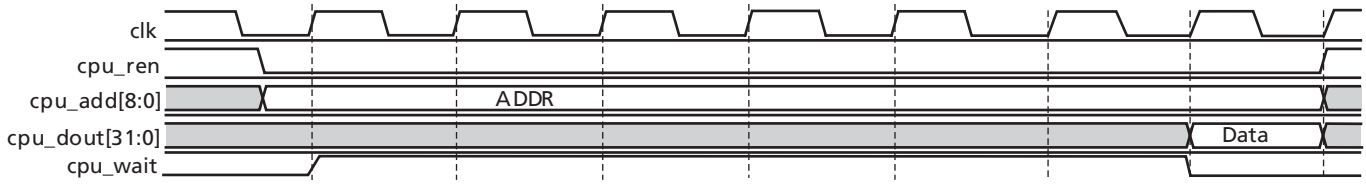
Note: cpu_ren should be deasserted on the next clock cycle after cpu_wait is deasserted. The read data is available one cycle after cpu_ren is sampled.

Figure 7 • **CPU Interface Control/Status Register Read Cycle**



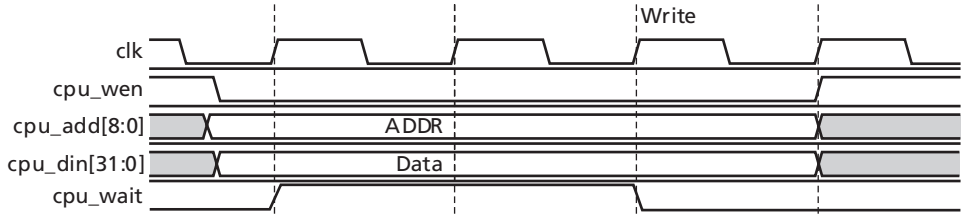
Note: cpu_wen should be deasserted on the next clock cycle after cpu_wait is deasserted. The write is done two cycles after cpu_wen is sampled.

Figure 8 • **CPU Interface Control Register Write Cycle**



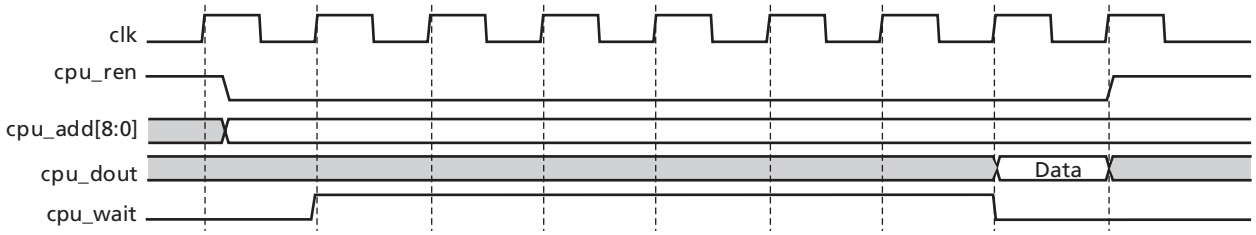
Note: cpu_ren should be deasserted on the next clock cycle after cpu_wait is deasserted. The read data is available six cycles after cpu_ren is sampled.

Figure 9 • **CPU Interface Data Register Read Cycle**



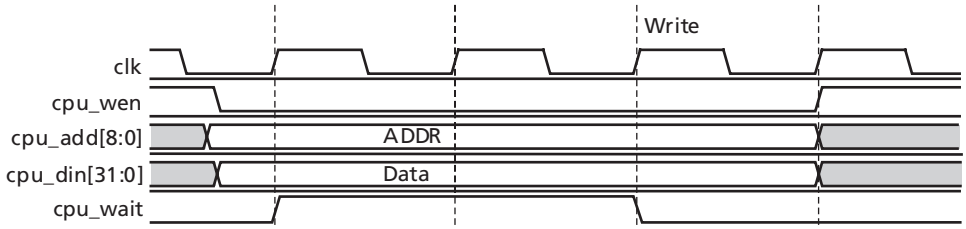
Note: cpu_wen should be deasserted on the next clock cycle after cpu_wait is deasserted. The write is done two cycles after cpu_wen is sampled.

Figure 10 • **CPU Interface Data Register Write Cycle**



Note: cpu_ren should be deasserted on the next clock cycle after cpu_wait is deasserted. The read data is available six cycles after cpu_ren is sampled.

Figure 11 • **CPU Interface Label Memory Read Cycle**



Note: cpu_wen should be deasserted on the next clock cycle after cpu_wait is deasserted. The write is done two cycles after cpu_wen is sampled.

Figure 12 • **CPU Interface Label Memory Write Cycle**

Clock Requirements

To meet the ARINC 429 transmission bit rate requirements, the Core429 clock input must be 1, 10, 16, or 20 MHz with a tolerance of ± 0.01%.

Core429 Verification

The comprehensive verification simulation testbench (included with the Netlist and RTL versions of the core) verifies correct operation of the Core429 macro. The verification testbench applies several tests to the Core429 macro, including:

- Receive Interface tests
- Transmit Interface tests
- CPU Interface tests
- Legacy Interface tests
- Loopback tests

Using the supplied user testbench as a guide, the user can easily customize the verification of the core by adding or removing tests.

Testbench

The CPU model sets up Core429 via the CPU interface and loads the transmit data. The transmit data will be sent to the receiver. The CPU model can retrieve the receive data through the CPU interface and compare it against the transmitted data.

The core comes with three testbenches: a full verification testbench that demonstrates full operation in Verilog, and two user testbenches, one in VHDL and the other in Verilog.

The user testbenches are intended to simplify core integration into the target system (Figure 13). This consists of the core connections to a CPU model and loopback logic that connects Tx output to the Rx input.

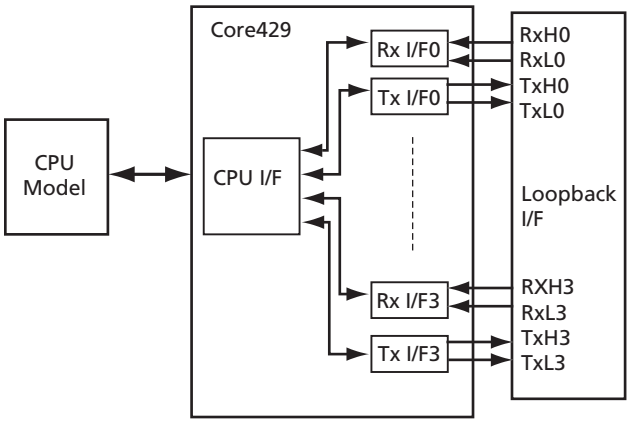


Figure 13 • **Testbench Diagram**

Line Drivers

Core429 needs ARINC 429 line drivers to drive the ARINC 429 data bus. Core429 is designed to directly interface to common ARINC 429 line drivers, such as the HOLT HI-8382/HI-8383, DDC DD-03182 or Device Engineering DEI1070.

Figure 14 shows the connections required from Core429 to the line drivers.

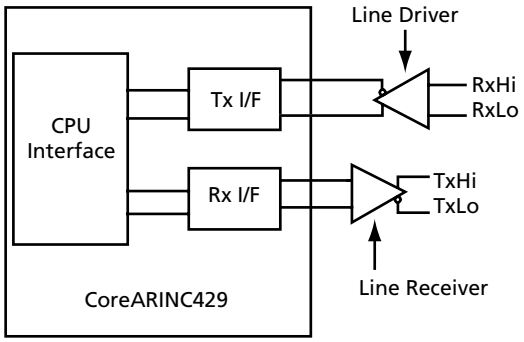


Figure 14 • **Core429 Line Driver and Line Receiver Interface**

Line Receivers

Core429 needs ARINC 429 line receivers to receive the ARINC 429 data bus. Core429 is designed to directly interface to common ARINC 429 line receivers, such as the HOLT HI-8588 or Device Engineering DEI3283. When using ProASIC^{PLUS}, RTAX-S, or Axcelerator FPGA families, level translators are required to connect the 5 V output levels of the Core429 line receivers to the 3.3 V input levels of the FPGA.

Figure 15 on page 19 shows the connections required from Core429 to the line receivers.

Loopback Interface

If the loopback bit in the transmit control register is enabled, the transmit outputs will be connected to the receive inputs. If there are equal numbers of transmit and receive channels, each transmit channel output is connected to the corresponding receive channel input. As an example, transmit channel 0 output is connected to receive channel 0 input.

If there are more receive channels than transmit channels, then the extra receive channels are connected to transmit channel 0. As an example, if we have two transmit channels (0 and 1) and four receive channels (0, 1, 2, and 3) then the connections are made as follows:

- Connect transmit channel 0 output to receive channel 0 inputs.
- Connect transmit channel 1 output to receive channel 1 input.
- Connect transmit channel 0 output to receive channel 2 input.
- Connect transmit channel 0 output to receive channel 3 input.

Development System

A complete ARINC 429 development system is also available, Actel part number "Core429-DEV-KIT". The development system uses an external terminal (PC) using a serial UART link to control Core429 with four Rx and four Tx channels implemented in a single ProASIC^{PLUS} APA600 FPGA.

The loopback interface logic allows the ARINC core to operate with the loopback mode. The development kit (Figure 15 on page 19) includes ARINC line drivers and line receivers.

On power-up, Core8051 will read the message from the ADC, which could be the aircraft fuel level or flap position, for example, and transmits over the transmit channel. The message will be transmitted to the receiver through the loopback interface. Then the message will be retrieved by Core8051 from the receiver and displayed on the LCD display.

Another way is to transmit the ADC message over the transmit channel through the line drivers to another system similar to the one described above. The message will go through the receive channel of the second system and can be displayed on the LCD display.

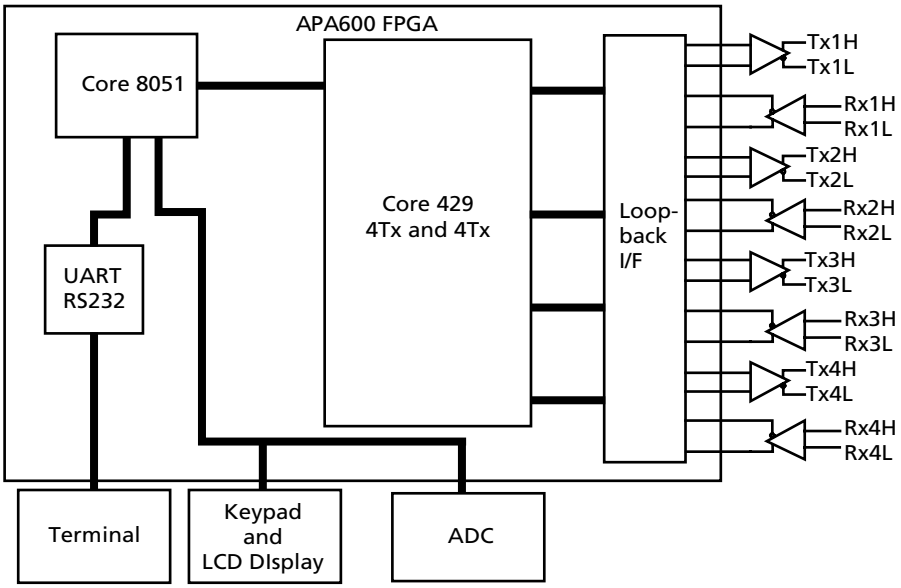


Figure 15 • **Typical Core429 System Diagram**

Ordering Information

Core429 can be ordered through your local Actel sales representative. It should be ordered using the following part number: Core429-XX, where XX is the appropriate value from Table 22:

Table 22 • **Ordering Codes**

XX	Description
EV	Evaluation Version
SN	Netlist for single use on Actel devices
AN	Netlist for unlimited use on Actel devices
SR	RTL for single use on Actel devices
AR	RTL for unlimited use on Actel devices
UR	RTL for use not restricted to Actel devices

The Evaluation board can also be ordered using the part number "Core429-DEV-KIT".

List of Changes

The following table lists critical changes that were made in the current version of the document.

Previous Version	Changes in Current Version (v5.0)	Page
v4.1	The "Key Features" section was updated to modify the selectable data rate on each channel.	1
	Figure 2 was added.	2
	The "General Description" section and "ARINC 429 Overview" section were updated.	2
	A paragraph was added to the end of the "Core429 Device Requirements" section.	3
	Table 6 was updated to add Fusion.	5
	Figure 3 was updated.	3
	The "Default Mode" section was added.	5
	The "Functional Description" section was updated.	5
	The "Legacy Mode" section was added.	7
	Table 7 was updated to add the TXRXSPEED_n parameter, and the table note was updated.	8
	The first four rows of Table 9 were moved to Table 8. Table 9 was updated to remove the tx_en signal, modify the int_out_tx signal description, and add the rx_fifo_full through tx_fifo_empty signals. The "Clock and Reset" section was renamed to "ARINC Interface" and the "ARINC Interface" section was renamed to "Default Mode Signals".	8–9
	Table 11 was moved to a later position in the document, just before the "Default Mode" section.	10
	Information about the Tx module was added to the "Legacy Interface" section.	10
	The channel decoding values were updated for the 32-bit CPU data bus in the "Default Mode Operation" section.	10
	The Address Map was updated in the "Rx Registers" section and the "Tx Registers" section.	11, 12
	Table 14 was updated to modify the data rate, decoder, match header bit 9, and match header bit 10 descriptions. Label memory address was renamed reload label memory, and its description was updated.	11
	Table 15 was updated to rename FIFO half full to FIFO half full or programmed level, and the description was modified.	11
	Table 16 was updated to rename Data to Label and update the type and description.	12
	Table 18 was updated to modify the data rate description.	12
	Table 19 was updated to modify the type of FIFO empty and FIFO full. FIFO half full was renamed to FIFO half full or programmed level, and its type and description were modified.	12
	The "Label Memory Operation" section was added.	13
	Information was added to the "Legacy Operation" section to clarify its purpose and configurability.	13
	Table 20 was updated to modify the description for receiver 1 data rate, label compare, match ARINC bit 10, match ARINC bit 9, transmitter data rate, and transmitter data rate.	14
	Table 21 was updated to change the type from R/W to R for all bits.	15
	The signal names cpu_clk and cpu_addr[7:0] were changed to clk and cpu_add[8:0] in Figure 7 through Figure 12. The wave forms were modified in Figure 7, Figure 10, Figure 11, and Figure 12, and notes were added to each figure.	16–17

v4.0	The title of Table 4 was updated. The Fusion device was changed to AFS1500.	4
v3.1	The "Supported Families" section was updated to include Fusion.	1
	The "Core429 Device Requirements" section was updated to include Fusion data.	3
v3.0	The "Core Deliverables" section was updated.	1
	Table 5 is new.	4
	Table 9 was updated.	9
	Figure 9 was updated.	16
v2.0	The "Core429 Device Requirements" section was updated.	3
	Table 1 was updated.	3
	Table 2 was updated.	3
	Table 3 was updated.	4
	Table 4 was updated.	4
	Table 20 was updated.	14

Datasheet Categories

In order to provide the latest information to designers, some datasheets are published before data has been fully characterized. Datasheets are designated as "Product Brief," "Advanced," and "Production." The definitions of these categories are as follows:

Product Brief

The product brief is a summarized version of an advanced or production datasheet containing general product information. This brief summarizes specific device and family information for unreleased products.

Advanced

This datasheet version contains initial estimated information based on simulation, other products, devices, or speed grades. This information can be used as estimates, but not for production.

Unmarked (production)

This datasheet version contains information that is considered to be final.

www.actel.com

Actel Corporation

2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Phone 650.318.4200
Fax 650.318.4600

Actel Europe Ltd.

Dunlop House, Riverside Way
Camberley, Surrey GU15 3YL
United Kingdom
Phone +44 (0) 1276 401 450
Fax +44 (0) 1276 401 490

Actel Japan
www.jp.actel.com

EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Phone +81.03.3445.7671
Fax +81.03.3445.7668

Actel Hong Kong
www.actel.com.cn

Suite 2114, Two Pacific Place
88 Queensway, Admiralty
Hong Kong
Phone +852 2185 6460
Fax +852 2185 6488

51700055-5/9.06