AltaCore-ARINC™
ARINC (RX & TX) Protocol Engine Specifications-Users Manual

Part Number: 14102-00000-E4
Cage Code: 4RK27 • NAICS: 334118

Alta Data Technologies LLC
4901 Rockaway Blvd., Building A
Rio Rancho, NM 87124 USA
(tel) 505-994-3111 • www.altadt.com

CUSTOMER NOTES:

Revision Control History

Rev E4 Release Date: 1 March 2014

Note to the Reader and End-User:

This document is provided for information only and is © Alta Data Technologies LLC 2011-13. While Alta strives to provide the most accurate information, there may be errors and omissions in this document. Alta disclaims all liability in document errors and any product usage. By using an Alta product, the customer or end user agrees (1) to accept Alta's Standard Terms and Conditions of Sale, Standard Warranty and Software License and (2) to not hold Alta Members, Employees, Contractors or Sales & Support Representatives responsible for any loss or legal liability, tangible or intangible, from any document errors or any product usage.

The product described in this document is not US ITAR controlled. Use of Alta products or documentation in violation of local usage, waste discard and export control rules, or in violation of US ITAR regulations, voids product warranty and shall not be supported. This document may be distributed to support government programs and projects. Third party person, company or consultant distribution is not allowed without Alta's written permission.

AltaCore-ARINC, ***AltaCore-ARINC-1553, AltaCore-ARINC, AltaAPI***, ***AltaView*** and ***AltaRTVal*** are Trademarks of Alta Data Technologies LLC, Rio Rancho, New Mexico USA

Contact:

We welcome comments and suggestions. Please contact us at 888-429-1553 (toll free in US) or 505-994-3111 or visit our web site for support submit forms at www.altadt.com or email us at alta.info@altadt.com or alta.support@altadt.com.

AltaCore-ARINC™

Table of Contents

<~~~>

AltaCore-ARINC™
Document Introduction

Introduction

This document is the Specifications-User's Manual Document for Alta Data Technologies' (ADT) ***AltaCore-ARINC*** protocol engine (PE), which supports various ARINC communications (usually based on ARINC-429 physical layer). This document specifies the design of the product and provides an overview for the reader who wants a detailed understanding of the ***AltaCore-ARINC*** design. The reader should also reference the ***AltaAPI*** manual (found on the CD or Alta web site) and source code for detailed descriptions of the software/backplane interface to ***AltaCore-ARINC***.

> **NOTE:** For most customer applications, the PE's design and processes are transparent to the application as the ***AltaAPI*** manages PE setup and execution. Most customers can skip this entire manual and simply refer to the ***AltaAPI*** manual and example programs for their ARINC application requirements. It is also recommended that the reader review ***AltaView*** products to learn about advanced avionics products.

This document is divided into sections that describe the major data structures of the ***AltaCore-ARINC*** PE:

- Introduction (This Section)
- Global Registers (Card Level)
- PE Root Device Level Registers (Channel Bank Level)
- Transmit (TX) Data Control
  - Signal Generator (SG)
  - Playback (PB) – Estimated Release 1 Sept 2008.
- Receive (RX) Communications
  - Channel Level RX
  - Multi Channel RX
- Interrupts (Int)

This Introduction section provides an overview of the ***AltaCore-ARINC*** architecture and should be reviewed before reading the various data structure sections.

Terminology

- This manual assumes the reader is very familiar with the general communications standards such as ARINC-419/429/573/575/717 and RS-232/422/485 style communications.
- msec = milli seconds; µsec = micro seconds; ηsec = nano seconds.

- An Alta **card/board** can contain one or more ***devices.*** A **device** is 16 channels (RX/TX configuration will vary). Another term for device is **bank.**
    - **Device = Bank** (This is group of ARINC 16 or less channels)**.**
- When discussing "**Global**" registers or values, these apply to ALL devices.
- The term "**ROOT"** register or value is for a specific device.

AltaCore-ARINC Architecture and Document Basics

AltaCore-ARINC is a 32-bit based PE that can support up to 16 ARINC channels (a "device" or "bank of channels"). Many Alta cards have one or two banks per card. The pure FPGA 32-bit design greatly enhances backplane/processor throughput as compared to other older 16-bit interfaces.

Memory Mapped Architecture

The ***AltaCore-ARINC*** protocol engine (PE) utilizes dedicated on-chip (FPGA on-chip memory) and/or on-board Extended Memory (depending on the PE configuration).

The memory of the PE/Card appears as a contiguous memory map to the host backplane regardless of the on-chip or on-board configuration (additional PE FPGA logic provides the arbitration for host access and various PE channel access). For most Alta card designs, an ***AltaCore-ARINC*** product uses high speed Quad Data Rate RAM or ZBT RAM, but the design is flexible and can be integrated to most common RAM channel architectures.

The ***AltaAPI*** or customer's application addresses each RX or TX channel through memory offset or software pointer schemes. The ***AltaAPI*** user/API's manual provides details of host driver and memory mapping requirements. Your Alta Product Hardware Manual (on the CD or the Alta web site download) details the memory configuration for your product (most card level products have one mega bytes of dedicated on-board memory per 16 channel bank), pin-outs and special configurations. Figure Intro-1 shows the basic memory mapping configuration for a single bank of 16 RX/TX channels. Special configurations may vary.

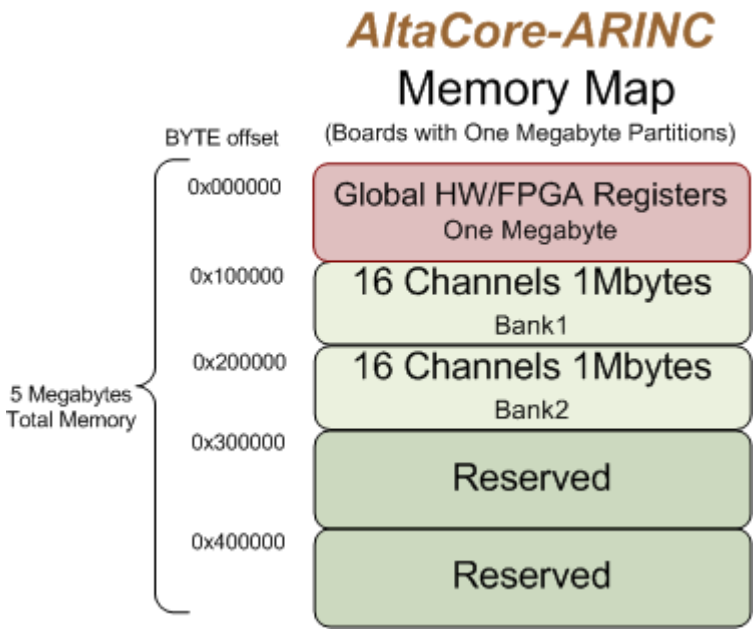


Figure Intro-1: Basic ***AltaCore-ARINC*** Memory Map

Global Registers

Global Registers are settings and values for ALL devices. These controls are usually for card-level functions and are not necessarily tied to a channel. When ***AltaCore-ARINC*** is sold as embedded IP, the device may NOT include Global Functions (this is specified at time of quote/order).

RX and TX Buffer and Interrupt Data Structures: Root verses Extended Memory

RX and TX functions have two levels of data structures that control execution: **Root Registers** and **Extended Memory** Data Structures that direct sub-functions and provide data buffering. Root Registers reside in the FPGA on-chip memory and have fixed address offsets absolute to the card's and channel's base address. Extended Memory Pointer Tables and Buffers have relative addresses whose base (root) values are set in the Root Registers. Each major Root or Extended Memory function typically has Control, Status (CSR's) or Setup Registers that provide setup options (control) and execution feedback (status).

All options and data structures are detailed in the following manual sections.

> **NOTE:** Several data structures of the PE are pointers to other tables or buffer data structures. The user/API must take great care to never populate a utilized structure with a null pointer or improper value (a value outside the on-board memory range). For example, if an RX Channel is turned-on and the Root or secondary pointer arrays are not properly programmed, then the PE will have unpredictable results (like any design that has a bad pointer reference). The PE

will not check for null or improper values and bad pointers will cause unpredictable behavior. The ***AltaAPI*** properly manages and provides allocation/destruction functions for the user/API application.

Memory Offset References

Memory references in this document are in byte offsets. The PE memory (a common word) is 32-bit atomic machine, 4-byte aligned.

Packed Word and Bit Data Structures

AltaCore-ARINC data structures are typically packed 32-bit word structures that may have embedded bit fields or single bit settings. The right most bit is LSB bit 0 (zero) and the left most bit is MSB 31. The general rule is that a bit value of 1 (one) is an "on/set/positive" and a 0 (zero) is "off/clear/negative."

All data structures will be labeled "**W**" (Write) if the user/API's application (or ***AltaAPI***) has the capability to write the value of the respective data element (word, bit or packed bits). Unless noted in the description, assume all data structures may be read by the user/API's application. The safest way to set control bits is to read the respective register using an OR (to set) or AND (to clear) in the selected bits and then write back. Great care must be taken to not set or change values of active registers during execution.

Execution Times, Channel Timing, Time Tags

Execution times provided herein are respective to the PE and do not include backplane transfer times. In general, after execution of a transmit function, the user/API should allow 20 µsecs settling time (process overhead) for accurate future transmission timing (for example, the user/API needs to provide 20 µsecs of non activity at the end of a TX minor frame to assure accurate Minor Frame/Minor Frame timing to 20 µsecs). There are some special cases that require more than 20 µsecs and the reader should reference the respective section for detailed timing requirements.

The user/API will have several options for programming gaps (idle/dead time between words or label/word lists). These values are typically 32-bit, 100 nsec tick values.

Reserve Bits and Words

Many bits and words are "Reserved" for future use or are used by the PE execution. All Reserved BITS should assume a value of zero. Reserved Words/Bits values should not be changed and in most cases should default to zero. If a reserved word is part of a larger data structure word packet, assume a value of zero for initialization, but then do not write to this word. For updating packed bit-word structures, the user/API's application should first read the word and then logically OR-in the desired one bits (or AND-out the zero bits) and then write back to the same location (please see the ***AltaAPI*** source for

numerous examples). The API source code is provided and illustrates the proper set and rules for memory access – for custom applications, please use the API as a design guide.

Figure Color Codes

In the following figures, a brown outline/white box signifies a bit/word/area that is programmed by the user/API. The light gray is usually a reserved bit/word/area and should not be changed by the user/API (the PE will update or use this element for processing). Most gray/reserved areas should be set to zero and most of these areas can be read by the user/API without affecting PE processing.

AltaCore-ARINC Global Registers (Card Level)

Global Card Level Registers

The area that occupies the first megabyte of the card memory map contains backplane and global card level settings and status values that affect processing for all devices. This area also contains the digital discrete, time and Built-In-Test (BIT) values of the card.

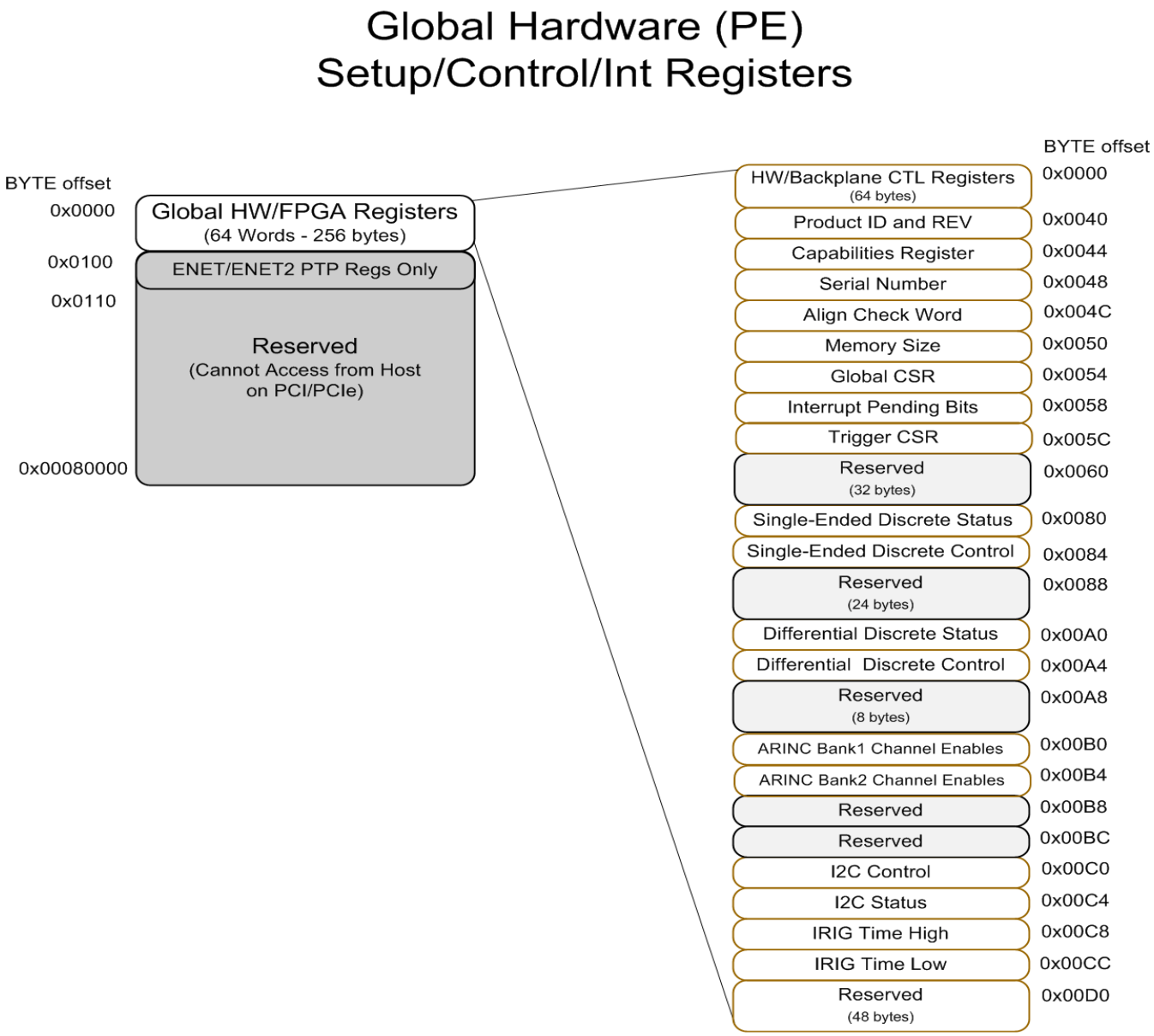
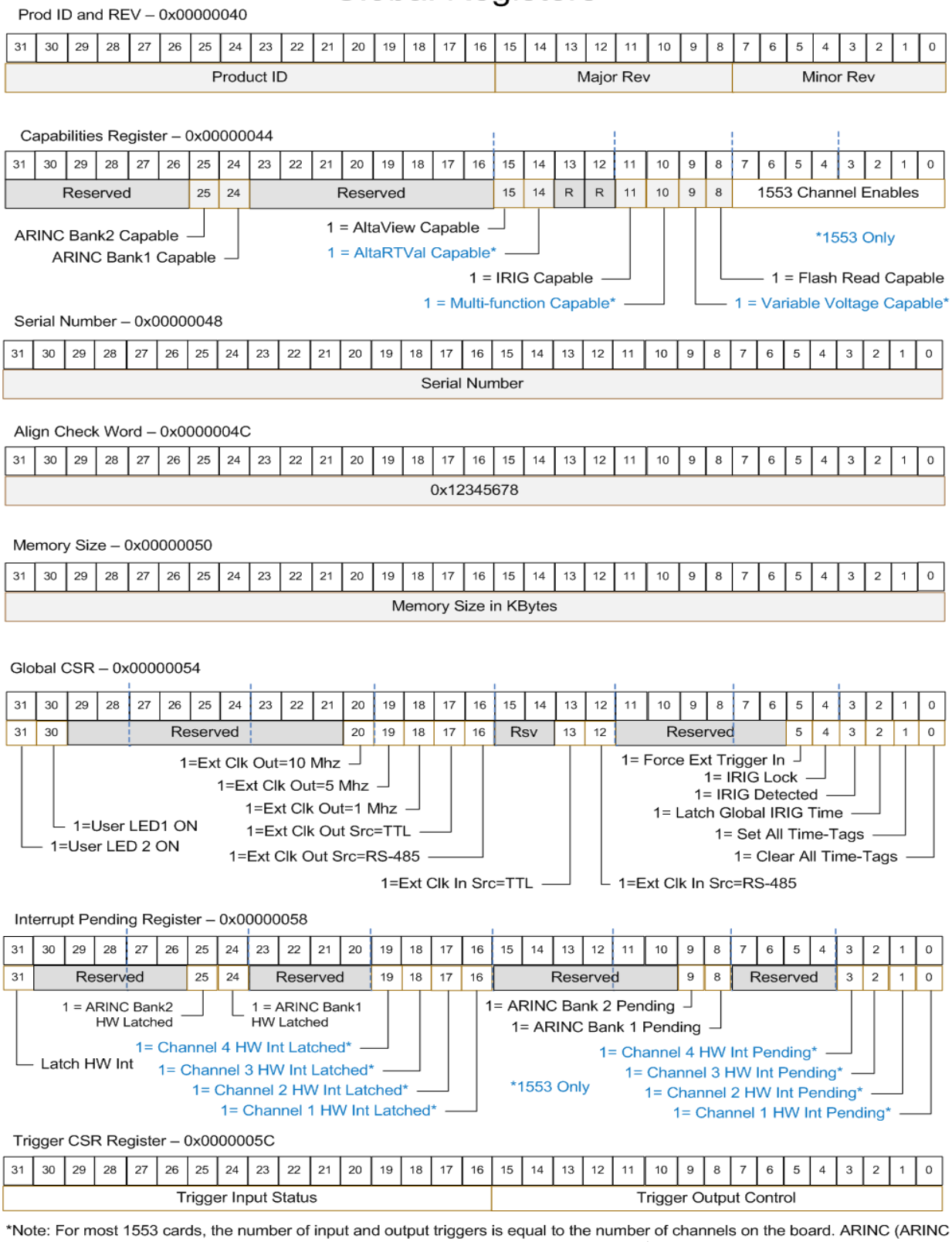


Figure Global Registers-1: Global Card Level Registers

HW/Backplane CTL Registers (16 words - 0x00000000-0000000F)

The first 16 words of the memory map are reserved for backplane configuration registers. Please see your product's Hardware Manual for details on these registers.

Global Registers



*Note: For most 1553 cards, the number of input and output triggers is equal to the number of channels on the board. ARINC (ARINC typically have only one trigger per bank. See your hardware manual for details on triggers/pin-outs available for your configuration.

Figure Global Registers-2: Device Global Registers

Global Registers
(continued)

Single-Ended Discrete Status Register – 0x00000080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bi-Directional Discrete Input Status (1=not active, 0=active)																															

Single-Ended Discrete Output Register – 0x00000084

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bi-Directional Discrete Output Control (1=drive output low)																															

Differential Discrete Status Register – 0x000000A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																Discrete Input Status (1=active, 0=not active)															

Differential Discrete Output Register – 0x000000A4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Output Enable (1=Enable Transmit)																Output Control (1 = drive output high)															

*Note: The number of single-ended and differential discretes varies from board to board. See individual board HW manuals for details on discrete counts.

I2C Control Register – 0x000000C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								23	22	21	20	Reserved												Write Data							

1= Start Command
1= Stop Command
Write Command
1= Read Command

I2C Status Register – 0x000000C4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								23	22					17	Reserved									Read Data							

1= ACK Received
1= Busy
1= Transfer In-Progress

Figure Global Registers-2A: Device Global Registers

Product ID and Rev: 0x00000040

This register provides the ***AltaCore-ARINC***/FPGA Program ID and Revision.

Capabilities Register: 0x00000044

This register provides settings for product options and capabilities. This allows Alta software to verify purchased or configured configuration.

NOTE: *These Bits Apply to 1553 Channels Only. These should be treated as reserve for ARINC only cards.

Bits 0-5: 1553 Channel Enables*

The channel enable bits indicate which channels are enabled on the board. For instance, if a two channel board was purchased, Bits 0 and 1 should be set to indicate that Channels 1 and 2 are enabled. (1553 Cards Only)

Bit 8: Flash Read Capable

This bit is set to a one if the Flash Read option is enabled on the board. This option enables the user/API to program setup data into on board flash. On power-up or reset, the setup data is copied from Flash to PE memory space. For more information regarding this feature, please contact Alta.

Bit 9: Variable Voltage Capable*

This bit is set to a one if the Variable Volt option is enabled on the board.

Bit 10: Multi-Function Capable*

This bit is set to one if the Multi-Function option is enabled on the board. When the Multi-Function bit is set to a one, TX, RT, and BM functions can be run simultaneously. When this bit is zero, the PE is operating in Dual-Function mode where only TX/BM or RT/BM functions can run simultaneously. For example, TX and RT functions cannot be enabled at the same time.

Bit 11: IRIG Capable

This bit is set to one if IRIG capability is enabled on the board.

Bit 14: ***AltaRTVal*** Capable*

This bit is set to one if ***AltaRTVal*** capability is enabled on the board. This option must be purchased on a board-by-board basis to allow ***AltaRTVal*** software to run with the board. (1553 Cards Only)

Bit 15: ***AltaView*** Capable

This bit is set to one if FULL ***AltaView*** capability is enabled on the board. This option must be purchased on a board-by-board basis to allow ***AltaView*** software to run with the board.

Bits 16-23: Reserved

Bit 24: ARINC Bank1 Capable

This bit is set to one if the first device bank of ARINC channels is available. If this bit is set to one, then read Device-Level PE Root register 0x000C to determine the number of RX and TX channels for the device bank (different cards can have different RX/TX selection capability – see your card's Getting Started & Hardware Manuals for details).

Bit 25: ARINC Bank2 Capable

This bit is set to one if the second device bank of ARINC channels is available. If this bit is set to one, then read device-level PE Root register 0x000C to determine the number of RX and TX channels for the device bank (different cards can have different RX/TX selection capability – see your card's Getting Started & Hardware Manuals for details).

Bits 26-31: Reserved

Serial Number: 0x00000048

This register provides the Born on Date and Incremental Serial Number of the card (the two left most segments seen on the card label – "MMYY-NUMBER"). The 16 MSBs make-up 4 BCD nibbles that provide the MMYY (M=Month, Y=Year) digits. The 16 LSBs represent the serial number 65535/99999 value.

Alignment Test Register: 0x0000004C

This register provides a simple 1,2,3,4,5,6,7,8 value in each nibble (0x12345678 word) so that an operating system driver developer can check for proper word/byte alignment. Nibble bits 28-31 has the value 1 (one).

Memory Size: 0x00000050

This register tells the user/API the size in K bytes for the total card. Only the lower 16 bits are used and for most cards this should be a value of 0x00001400 (5120 Kbytes = 5 Mbytes).

Global CSR: W: 0x00000054

This word is provided for user/API applications that want to control all channel time tags and HW Interrupt enables from one location. All of these settings are duplicated at the channel level for multi applications. This is a packed bit structure whose bits are defined in the following paragraphs.

Bit 0: Clear All Time Tags: W

This bit is set to one by the user/API to direct the PE to clear/zero all channel time tags (PE Time High & Low clocks), which are utilized to time tag ARINC label/word

list (Receive Packets – RxPs). Time clearing occurs <1µsec from this bit being set. This bit is self clearing.

Bit 1: Set all Time Tags: W

This bit is set to one by the user/API to direct the PE to load all local PE Time High & Low registers (offset 0x0024/28) at the same time (with the user/API loaded values). This allows all channels to have their clocks jammed with user/API time simultaneously. Otherwise, the user/API application would need to perform this bit setting for each PE channel and time values would be skewed. Time loading occurs <1µsec from this bit being set. This bit is self clearing.

Bit 2: Latch Global IRIG Time: W

This bit is set to a one by the user/API to load the current IRIG time into the IRIG Time Global Registers (offsets 0x00C8/CC). This bit is self clearing.

Bit 3: IRIG Detected: R

This read only bit set to a one indicates that activity has been detected on the IRIG input signal. Once activity has been detected this bit will remain set until no activity has been detected for 5 seconds. This bit does not mean IRIG lock has occurred (see bit 4 below), it is simply an indicator that “something” has been detected on the input signal within the last 5 seconds.

Bit 4: IRIG Lock: R

This read only bit set to a one indicates that the IRIG decoder has locked on a good IRIG signal. To achieve IRIG Lock the IRIG decoder must detect at least one valid IRIG time frame. IRIG Lock will be lost if at any point the IRIG decoder detects an invalid time frame. Once IRIG Lock is lost this bit will get set to zero and the IRIG decoder will go into search mode looking for another valid IRIG time frame.

Bit 5: Force Ext Trigger In: W

This bit set to a one by the user/API forces an input trigger to all channels simultaneously. Setting this bit overrides the external trigger signal on each channel. This bit provides the means for a software trigger as opposed to and external HW trigger. This bit is self-clearing.

For example, the user/API can setup the TX on each channel to wait for an external input trigger. Once the TXs are setup, enabled and waiting for a trigger, this bit can be set to start all TXs simultaneously.

Bits 6-11: Reserved

Bit 12: Ext Clk In Src=RS-485: W

Setting this bit to a one will select the differential “DDISC1” RS-485 lines as the external clock input. See the individual board HW manual for pin assignment for the DDISC1 (+) and (-) signals.

Bit 13: Ext Clk In Src=TTL: W

Setting this bit to a one will select the differential Ext TTL Clock line as the external clock input. See the individual board HW manual for pin assignment for the External TTL Clock I/O Signal.

Bits 14-15: Reserved

Bit 16: Ext Clk Out Src=RS-485: W

Setting this bit to a one will select the differential “DDISC1” RS-485 lines as the external clock output. See the individual board HW manual for pin assignment for the DDISC1 (+) and (-) signals.

> **NOTE:** If the RS-485 option is used for the external clock, the input and output clocks use the SAME PINS. Therefore, the input clock for all channels on the board are automatically connected to the output clock, so you can test the external clock on those channels without any external connection. Of course, if you need to synchronize with other boards then one board should be configured to output a clock signal and connected to the appropriate external clock input lines on the other boards (which are configured to accept the external clock signal and will NOT generate an output clock).

Bit 17: Ext Clk Out Src=TTL: W

Setting this bit to a one will select the differential Ext TTL Clock line as the external clock output. See the individual board HW manual for pin assignment for the External TTL Clock I/O Signal.

> **NOTE:** If the Ext TTL Clock option is used for the external clock, the input and output clocks use the SAME PIN. Therefore, the input clock for all channels on the board are automatically connected to the output clock, so you can test the external clock on those channels without any external connection. Of course, if you need to synchronize with other boards then one board should be configured to output a clock signal and connected to the appropriate external clock input lines on the other boards (which are configured to accept the external clock signal and will NOT generate an output clock).

Bit 18: Ext Clk Out=1 MHz: W

Setting this bit to a one will set the Ext Clock output frequency to 1 MHz.

Bit 19: Ext Clk Out=5 MHz: W

Setting this bit to a one will set the Ext Clock output frequency to 5 MHz.

Bit 20: Ext Clk Out=10 MHz: W

Setting this bit to a one will set the Ext Clock output frequency to 10 MHz.

Bits 21-29: Reserved

Bit 30: user/API LED1: W

Setting this bit to a one will turn on user/API Led1. See the individual board HW manual for user/API LED1 locations.

Bit 31: user/API LED2: W

Setting this bit to a one will turn on user/API Led2. See the individual board HW manual for user/API LED2 locations.

Summary of Multi Application verses Single Application Interrupt Options with *AltaCore-ARINC*

Multi Applications Scenarios: (usually one application per device for interrupts): All Interrupt signals are handled at the PE device level. You only need to read the next section and the Interrupt Section for details.

Single Application Scenarios: You have a choice to handle interrupt management at the device level or handle interrupt pending at the global level, which is why the following word and bits are provided. If you want to handle interrupts strictly at the device level, then treat this as a multi application scenario. If you want to handle interrupt pending signals at a single card/global level (which may save time on host reads/writes), then read the following paragraphs and the Interrupt Section of this manual for this supported option.

Global Interrupt Register: W: 0x00000058

This register is provided for applications that want to control all device interrupt hardware actions from one location verses managing each PE device individually (multi application processes would need to manage each device as an independent, logical device). These bits are duplicated at the PE device level. This is a packed bit structure whose bit values are defined in the following paragraphs. A brief discussion follows to provide the general overview of the bit values and proper execution action of reading the bits.

This register provides a pending and latching bit value for each of the ARINC channel devices. Bit 31 can be set to a one by the user/API to “latch” active interrupt signals and immediately clear the cards channel/bank interrupt pending signal to the host

backplane/system (all channel/bank interrupt pending signals, or bits, are OR'ed together to set the backplane interrupt pending signal). So bit 31 provides the application a single bit to clear the card's interrupt pending bit to not hold-off other possible interrupt pending conditions in the system (from other cards). This would not be multi application safe as independent application would need to service their respective channel/bank: there is a mechanism to clear interrupt pending at the channel/bank level, too.

In the PE Root Control Word (channel level), there is a "Hardware (HW) Interrupt On" bit that controls the interrupt pending condition to drive the card level backplane/system interrupt hardware line. Please see PE Root Registers – Control and Status Word for channel level interrupt settings.

Bits 0-3: Interrupt Pending

These bits, one per 1553 PE channel or ARINC device bank, are set to a one by the PE when an interrupt event(s) has posted for the respective channel/bank (please see the diagram for the channel mapping). When an interrupt has occurred (or when the host application conducts a polling event), the user/API application reads this register to see what channel/bank caused the interrupt. When the channel/bank level Interrupt Pending bit in the Root PE Status Word is cleared (set to zero), or when bit 31 of this word is set to one, then the respective pending bit is cleared (and if the hardware interrupt is turned on, then the backplane/system, interrupt pending signal is cleared). The user/API does not clear these bits to clear an interrupt signal.

Bits 4-15: Reserved

Bits 16-19: Interrupt Latched: W

These bits are provided as a copy of the pending bits in this word when the hardware interrupt signal is cleared (from writing a one to bit 31). After the user/API interrupt service routine has written a one to bit 31, these bits hold the current value of the pending bits (please see the diagram for the bit position for respective channel/bank). This allows user/API applications that want to handle interrupts at the global level to quickly clear the hardware event from the backplane/system and then read these bits to see which channel/bank caused the interrupt. The user/API application should clear these respective bits for the next interrupt event handler. Multi application processes would read interrupt pending events only at the PE Channel/Bank Level (described above).

Bits 20-30: Reserved

Bit 31: Latch Hardware Int: W

This self clearing bit is set to one by the user/API to direct the PE to clear the hardware interrupt pending signal to the backplane/system, clear the interrupt pending bits in this word and copy their value to the interrupt latch bits in this word (all action described in the previous paragraphs). Software polling interrupt event handlers do not touch this bit.

Trigger CSR: W: 0x0000005C

This word is provided for user/API applications that want to know the state of the input triggers and is also used to override the output triggers. Normally the output triggers are controlled by the PE when the "generate output trigger" bit is set in a PE control register. The number of input and output triggers is equal to the number of channels on the board.

Bit 15-0: Trigger Output Control: W

Trigger outputs can be driven low (Active) by setting the corresponding bit in this register to a one. Bit0=Trigger1, Bit1=Trigger 2, etc.

Bit 31-16: Trigger Input Status:

These bits show the status of the input triggers. A one for a given bit indicates that the trigger is in-active and a zero indicates the trigger is active. Bit16=Trigger1, Bit18=Trigger 2, etc.

Discrete Configurations – See Product Hardware Manual

See your product's Hardware Manual for Discrete Configuration and Pin-Outs. Most ARINC cards have limited discrete configurations (usually only 1-8).

Single-Ended Discrete Status Register: 0x00000080

This word shows the input status of the bi-directional avionics discretes. A one for a given bit indicates that the discrete is in-active and a zero indicates the discrete is active. Bit0=Discrete1, Bit1= Discrete2, etc.

NOTE: The input voltage should not exceed 30 Volts. Exceeding this rating can damage the board.

Single-Ended Discrete Output Register: W: 0x00000084

This word is used to control the output state of the bi-directional avionics discretes. Discrete outputs can be driven low (Active) by setting the corresponding bit in this register to a one. Bit0=Discrete1, Bit1= Discrete2, etc.

NOTE: The bi-directional Avionics Discretes can sink up to 1 Amp when active. Exceeding this rating can damage the board.

Differential Discrete Status Register: 0x000000A0

This word shows the input status of the differential RS485 discretes. A one for a given bit indicates that the discrete is active and a zero indicates the discrete is in-active. Bit0=Discrete1, Bit1= Discrete2, etc.

Differential Discrete Output Register: W: 0x000000A4

This word is used to control the output state of the differential RS485 discretes.

Bit 15-0: Output Control: W

Discrete outputs can be driven high (Active) by setting the corresponding bit in this register to a one. Bit0=Discrete1, Bit1= Discrete2, etc. The corresponding transmit enable bit must be set before this bit is valid. Bit0=Discrete1, Bit1= Discrete2, etc.

Bit 31-16: Output Enable: W

These bits control the output enables of the uni-directional differential RS485. If a given bit is set to a one, the discrete output is enabled (transmit mode). If a given bit is set to a zero, the discrete output is disabled (receive mode). Bit16=Discrete1, Bit17= Discrete2, etc.

I2C Control Register: W: 0x000000C0

This word is used to control the I2C logic on the board. See the individual board HW manual for more details on this register. The user/API's application does not need to know details of this register – the ***AltaAPI*** handles all overhead and management of these registers – please see source code of ***AltaAPI*** for details.

I2C Status Register: 0x000000C4

This word provides the status of the I2C logic on the board. See the individual board HW manual for more details on this register. The user/API's application does not need to know details of this register – the ***AltaAPI*** handles all overhead and management of these registers – please see source code of ***AltaAPI*** for details.

IRIG Time High: 0x000000C8

This word shows current IRIG Time High value. To latch the current IRIG time in this word, the Latch IRIG Time (Bit 2) in the Global CSR should be set prior to reading this word.

IRIG Time Low: 0x000000CC

This word shows current IRIG Time Low value. To latch the current IRIG time in this word, the Latch IRIG Time (Bit 2) in the Global CSR should be set prior to reading this word.

NOTE: The IRIG times latched in the registers described above are one second behind. The user/API's code will need to adjust the time +1 second. Check your AltaAPI release notes for this +1 second adjust made to the IRIG read function.

PTP IEEE-1588 Registers: 0x0100-0x010C

These 4 registers are for ENET/ENET2 devices for key PTP IEEE-1588 Time Sync and Control-Status (CSR) registers. ENET devices can synchronize time via 1588 grand masters in a similar manner to IRIG synchronization. If engaged (via user software API settings), this overrides IRIG sync decoding.

For details on implementation, please contact Alta for access to the PTP IEEE-1588 Design Implementation Application Note and reference the appropriate sample C programs. Contact Alta at alta.support@altadt.com or 888-429-1553 x2.

<~~~>

AltaCore-ARINC: Root PE Device/Bank Registers

Introduction

There are several control and status settings at the device level for ARINC functions and events. The following figure provides an overview of the Root registers memory map. This section will describe the Root PE Registers that are common for ARINC channels functions of the respective device (bank of channels). The subsequent sections of manual will detail the Root Registers for the respective RX, TX, Interrupt functions (as shown in the figure below).

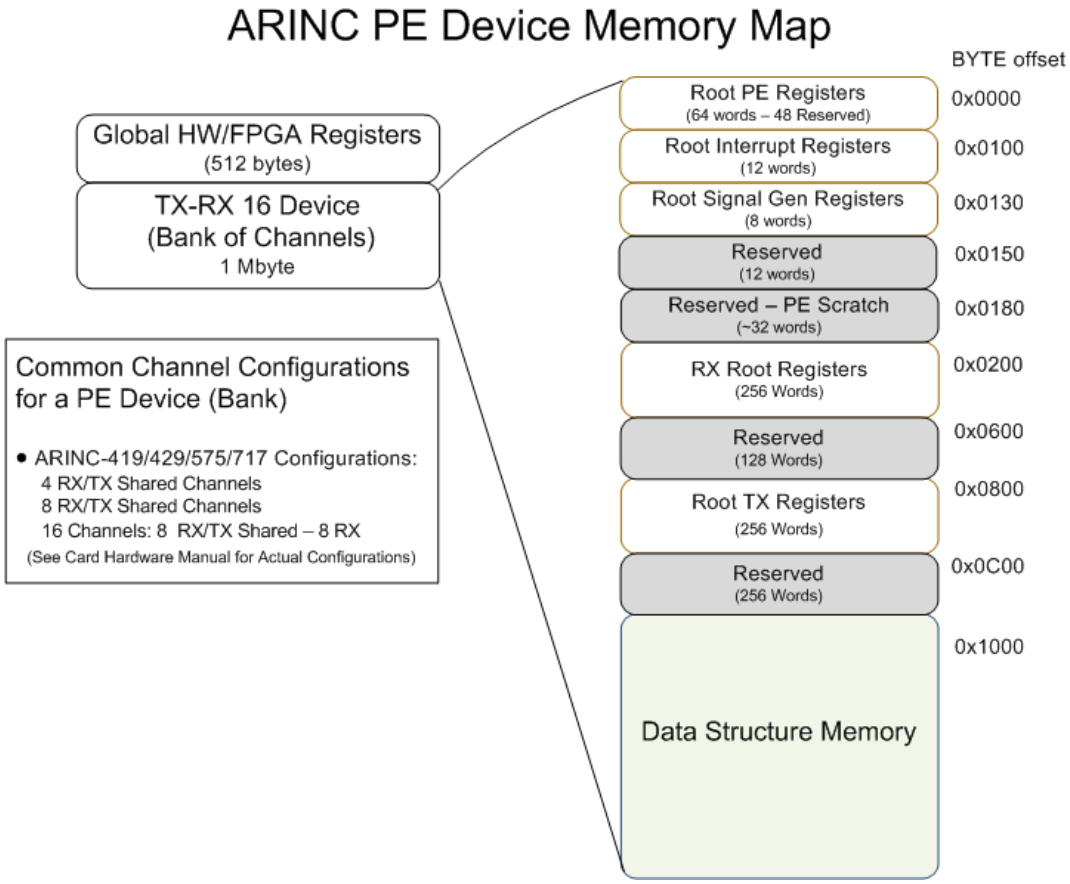


Figure PE Root-1: Root ARINC PE Device Memory Map

Each of the major Root Register areas shown above are detailed in their respective sections of this manual. This section will now detail the Root PE Registers (0x0000-0x00FF) that affect top-level for all device (channel) functions. The following figure and paragraphs details this area.

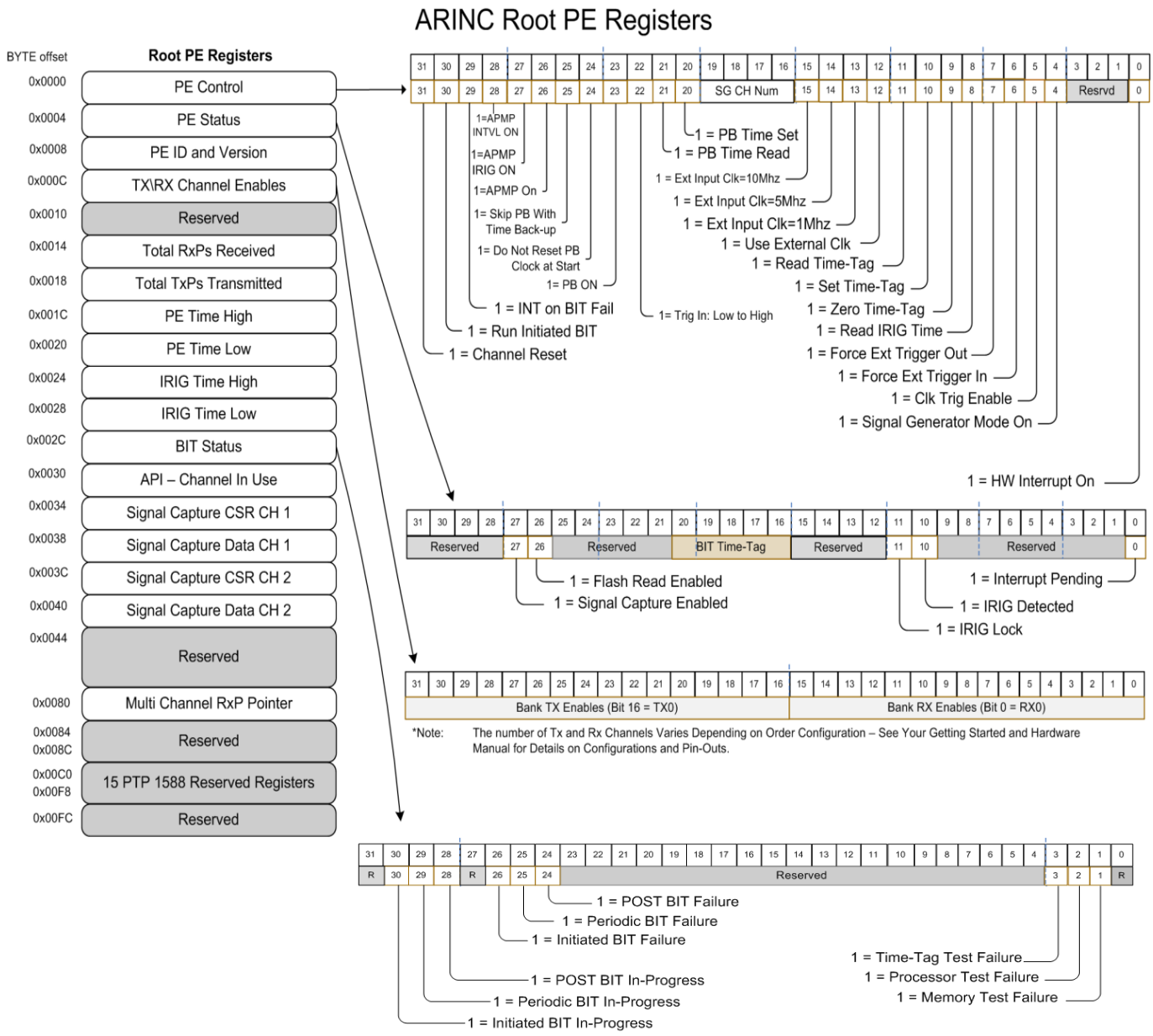


Figure PE Root-2: Root PE Registers

Root PE Control Word: W: 0x0000

This register is a packed bit structure and provides key PE control for the user/API. The following paragraphs detail the bit settings.

Bit 0: HW Interrupt On: W

This bit is set to a one by the user/API to direct the PE to enable hardware interrupts to the backplane (or interrupt line of an FPGA core build). Interrupt events from RX or TX Channels are logged in the Interrupt Queue regardless of this setting (see the respective section for details). Setting this bit to zero requires the user/API's host application to poll the interrupt queue to service

events. Setting this to a one will require the user/API's host application to have an operating system interrupt service routine (ISR) to handle the backplane interrupt and then read the Interrupt Queue to service the events.

Bit 4: Signal Generator Mode On: W

This bit is set to a one by the user/API to direct the PE to start the Signal Generator (see the Signal Generator (SG) section of this document for details. Normal channel TX or PB cannot be operating during SG mode.

Bit 5: Clock (CLK) Trigger Enable: W

This bit is set to one by the user/API to direct the PE to NOT increment the RX Time Tag (PE Device) clock or the Playback control clock until an external trigger has been detected. This allows the user to jam all the appropriate device RxP time tag clocks and playback clocks with a system value and then have an external trigger synchronize the starting of the clocks (either incrementing off the PE's internal 50 MHz clock or an external 1, 5 or 10 MHz clock).

Bit 6: Force External Trigger In: W

This bit is set by the user/API to direct the PE to set external triggers for all devices and any of their PE functions that may be stalled for ext trigger (for example the user/API can set Wait for Ext Trigger on TX and BM). This bit is extremely useful to ensure all TXs start within 5 µsecs of each other. Most competitive products require software control of each channel resulting in huge delays between channels. This bit is self clearing.

Bit 7: Force External Trigger Out: W

This bit is set by the user/API to direct the PE to override its output trigger. Normally the output trigger is controlled by the PE when the "generate output trigger" bit is set in a PE control register. This bit allows a software application to generate an output trigger on demand. This bit is self clearing.

Bit 8: Read IRIG Time

This bit is set to a one by the user/API to direct the PE to latch (for reading) the PE's internal 64-bit, 20 ηsec time and the most current IRIG Time High & Low values to the Root PE Registers Time High & Low (offsets 0x001C/20) and IRIG Time & Low registers (offsets 0x0024/28). See IRIG Time High & Low Registers later in this section for more discussion on IRIG time. The latency is <1µsec. If IRIG Time is not being used, then the value in the IRIG time registers is unpredictable. This bit is self clearing.

NOTE: The IRIG times latched in the registers described above are one second behind. The user/API's code will need to adjust the time +1 second. Check your AltaAPI release notes for this +1 second adjust made to the IRIG read function.

Time Tag Discussion (Time High & Low):

Each PE Channel has a 64-bit, 20 ηsec internal timer that is used to time tag all label/word lists in all modes (Time High & Low in every Receive Packet, RxP, see the RX Section of this manual). The PE provides the user/API a 2-word (Root Time High & Low – Root PE Register offset 0x001C/20) window to access the internal PE timer. These Root Time High & Low registers are used to program or read the internal timer per the bit selections discussed in the ensuing paragraphs.

Bit 9: Zero Time-Tag: W

This bit is set to a one by the user/API to direct the PE to zero/clear the PE Time High & Low registers (offsets 0x001C/20), which is used for time stamping label/words. The PE clears this bit when the clock registers have been set to zero.

Bit 10: Set Time-Tag: W

This bit is set to a one by the user/API to direct the PE to read the user/API written values in Time High & Low and jam/set the internal PE time registers (offsets 0x001C/20), which is used for time stamping RxP label/words. The PE will clear this bit when the read/jam is finished. The delay from this bit being set to the internal time tag register being jammed is <1µsec (latency). This bit is self clearing.

Bit 11: Read Time-Tag: W

This bit is set to a one by the user/API to direct the PE to latch the internal time to the PE Time High & Low registers (this allows the user/API to read the time values and is an excellent “heart beat” check on the PE – this is clock used for RXP time stamping). The latency is <1µsec. The PE will clear this bit when the time values have been written.

Bit 12: Use External Signal:W

This bit is set to a one by the user/API to direct the PE to use the selected (RS-485 or TTL) external input to drive the clock count (PE Time High & Low).

Bit 13: Ext Input Clk = 1 Mhz

Setting this bit to a one will indicate to the PE that the external input clock frequency is 1 MHz.

Bit 14: Ext Input Clk = 5 Mhz

Setting this bit to a one will indicate to the PE that the external input clock frequency is 5 MHz.

Bit 15: Ext Input Clk = 10 Mhz

Setting this bit to a one will indicate to the PE that the external input clock frequency is 10 MHz.

Bits 16-19: Reserved

Bit 20: PB Time Set: W

This bit is set to a one by the user/API to direct the PE to read the user/API written values in Time High & Low and jam/set the internal PE time registers (offsets 0x001C/20), which is used for PB (Plaback timer functions. The PE will clear this bit when the read/jam is finished. The delay from this bit being set to the internal time tag register being jammed is <1µsec (latency). This bit is self clearing.

Bit 21: PB Time Read: W

This bit is set to a one by the user/API to direct the PE to latch the internal time to the PE Time High & Low registers (this allows the user/API to read the time values and is an excellent "heart beat" check on the PE – this is clock used for PB transmission control). The latency is <1µsec. The PE will clear this bit when the time values have been written.

Bit 22: Trigger In Low to High: W

This bit is set to one by the user/API to direct the PE to accept a Low to High (0 to 1 edge) trigger verses the default high to low edge trigger.

Bit 23: PB On: W

This bit is set to one by the user/API to direct the PE to start PB transmission across all selected TX channels. The user must also set the TX channel PB On bit in the respective TX/PB CSR1 Word.

Bit 24: Do Not Reset PB Clock at Start: W

This bit is set to one by the user/API if the PE should NOT reset the playback internal clock at the start. This bit must be set for absolute timing. Usually, this user/API will have set/preset the PB internal clock with a desired value prior to starting PB (setting bit 0 to one).

Bit 25: Skip PXPs with Time Back-up: W

This bit is set to one by the user/API if the PE should skip PXPs with a time value less than the PB internal clock.

Bit 26: ENET APMP On – ENET-A429 ONLY

This bit is set to one to direct the PE to enable Alta Passive Monitor Protocol (APMP) for Multi-Channel ARINC channels. APMP mode is a feature for ENET-A429 devices only and will provide automatic UDP bridging of ARINC RXP label/words. Only RX Channels that have multi-channel mode enable in their

respective Setup1 channel register will be have their RXPs added to the APMP payload.

Bit 27: APMP IRIG On – ENET-A429 Only

This bit is set to one to direct the PE to insert PE and IRIG Time words in the first four words of the APMP payload. Please see the end of the RX Section and the IRIG bookmark: Root IRIG Time High & Low: 0x0024/28.

Bit 28: APMP INTVL On – ENET-A429 Only

This bit is set to one to direct the PE to inset PE and Interval Time words in the first four words of the APMP payload. Please see the end of the RX Section and the bookmark: Root Interval Timer: 0x004C

Bit 29: INT on BIT Fail

This bit set to a one will cause an interrupt to be generated when a BIT failure is detected. Note that an interrupt will only get generated on each instance that bits 7-0 of the Bit Status Register (offset 0x002C) are set. For example, if a BIT Failure is detected multiple times, only one interrupt will get generated for that failure until the bit is set to zero by the user/API and another BIT Test failure is detected causing the bit to be set again. The user/API should clear the BIT status register once it has detected and processed the BIT failure.

Bit 30: Run Initiated BIT

This bit is set to a one by the user/API to direct the PE run Initiated BIT. This bit is self clearing.

Bit 31: Reset Channel

This bit is set to a one by the user/API to reset the PE channel. A channel reset will force all PE registers to a power-on state. For example, POST BIT will run, Time-Tag registers will get reset to zero, etc. This bit is self clearing.

> NOTE: POST BIT performs a memory test which may cause some of the user/API data to be cleared. Once a channel reset performed, the contents of the PE registers and memory must be initialized again by the user/API.

Root PE Status Word: 0x0004

This word provides the user/API status on four key indicators: Interrupt Pending for the Channel, IRIG Signal Detection. This is a packed bit structure whose values are detailed in the following paragraphs.

Bit 0: Interrupt Pending: W

This bit is set to a one by the PE to indicate that an interrupt event is pending in the channels interrupt queue. If the Root PE Control Word Bit 0, HW Interrupt On, is set to a one, then a backplane/system interrupt signal has been set. This

bit MUST BE cleared (set to zero) by the user/API to clear the backplane/system signal. Even if the HW Interrupt On bit is not on (so the user/API has decided against the channel being able to generate a backplane/system interrupt signal), the user/API should still clear this bit during an interrupt queue polling event so the bit can be utilized for the next interrupt event.

Bits 1-9: Reserved

Bit 10: IRIG Detected

This read only bit set to a one indicates that activity has been detected on the IRIG input signal. Once activity has been detected this bit will remain set until no activity has been detected for 5 seconds. This bit does not mean IRIG lock has occurred (see bit 11 below), it is simply an indicator that “something” has been detected on the input signal within the last 5 seconds.

Bit 11: IRIG Lock

This read only bit set to a one indicates that the IRIG decoder has locked on a good IRIG signal. To achieve IRIG Lock the IRIG decoder must detect at least one valid IRIG time frame. IRIG Lock will be lost if at any point the IRIG decoder detects an invalid time frame. Once IRIG Lock is lost this bit will get set to zero and the IRIG decoder will go into search mode looking for another valid IRIG time frame.

Bits 12-13: Reserved

Bits 16-20: Bit Time Tag

These bits contain the 5 least significant bits of the time-tag register and are used by the PE for Time-Tag BIT testing.

Bits 21-25: Reserved

Bit 26: Flash Read Enable

This bit set to a one indicates that the Flash Read option is enabled for this PE. This option enables the user/API to program setup data into on board flash. On power-up or reset, the setup data is copied from Flash to PE memory space. For more information regarding this feature, please contact Alta Technical Support.

Bit 27: Signal Capture Enabled

This bit set to a one indicates that an analog-to-digital (ADC) converter is present on this channel and that it is capable of capturing the channel signal.

Bits 28-21: Reserved

Root PE Product ID & Version: 0x0008

When the PE starts-up, this register is set with the PE's Product ID and Version Number.

Root PE RX/TX Enables: W: 0x000C

This word is set by the PE with the number of valid transmitters and receivers configured for the card purchase. If a bit is set to a one, then the corresponding channel is configured on the card (this register allows the user/API to know what channels are physically installed on the card – individual RX and TX start stop bits in the respective Root Control words will activate the channel). Bits 0-15 correspond to RX channels 0-15, and bits 16-31 correspond to TX channels 0-15.

Reserved: 0x0010

Root Total RxPs Received: W: 0x0014

This register is set to a zero by the PE at initialization and then incremented for every RxP (Label/Word) that is received. The user/API can set this to any initial value.

Root Total TxPs Transmitted: W: 0x0014

This register is set to a zero by the PE at initialization and then incremented for every TxP (Label/Word) that is transmitted. The user/API can set this to any initial value.

Root Time High & Time Low: W: 0x001C/20

These two registers make-up a 64-bit, 20 ηsec time value and provide a user/API window to the internal 64-bit time register of the PE (used to time stamp TxP). These registers are used for reading the current internal PE Time registers or these registers are set by the user/API to jam/set the internal PE Time registers with the held value. Root PE Control Word bits (described early in this section) control the action of these registers for read and write. The Time High register holds the 32 MSB bits and the Time Low register holds the lower 32 LSBs. The resolution is 20 ηsec/bit.

Root IRIG Time High & Low: 0x0024/28

These two words provide a read-only user/API window to the PE's internal 64-bit IRIG-B

Time Word. IRIG-B decoding is built-in to most ***AltaCore-ARINC*** PE configurations, but the function is turned-on/off via the Global Hardware Capabilities Register. If IRIG Time is not being used, then the value in the IRIG time registers is unpredictable.

The following diagram provides the structure of these words (IRIG-B time is encoded **DAY-HOUR-MINUTE-SECOND** in TXD nibble digits).

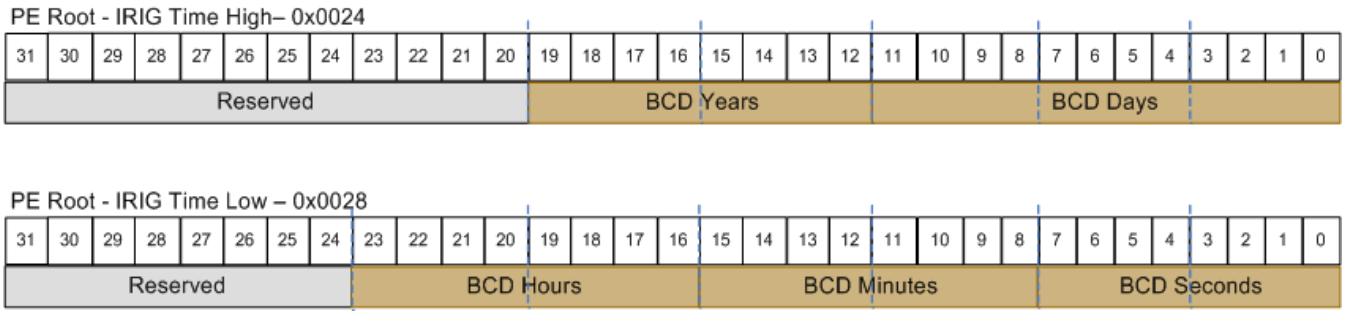


Figure Root PE-3: IRIG Time High & Low Format

When a valid IRIG-B one second string has been decoded, the PE will automatically latch the current PE Time High & Low, and the IRIG Time High & Low Registers into internal registers. To read the current PE Time High & Low and correlating IRIG Time High & Low Registers, the user/API's application must set the Read IRIG Time bit in the Root PE Control Word to force the most current values of the internal registers to be latched to the Time High & Low and the IRIG Time High & Low registers: this allows the user/API's application to correlate the time between the PE's internal clock and the decoded IRIG signal. Only the PE's internal clock is used for time stamping label/word lists.

NOTE: The IRIG times latched in the registers described above are one second behind. The user/API's code will need to adjust the time +1 second. Check your AltaAPI release notes for this +1 second adjust made to the IRIG read function.

NOTE: Alta IRIG decoder support s:

- IRIG-B002, Format B, DC Level Shift, no index count interval, BCD code expression
- IRIG-B102, Format B, Amplitude Modulated, no index count interval, BCD code expression

Root BIT Status: 0x002C

This word provides the status of BIT operation for the PE.

Bit 0: Encoder/Decoder Test Failure

This bit indicates that the PE has detected a failure on its Encoder/Decoder Wrap Test. The Encoder/Decoder Test only runs during POST BIT.

Bit 1: Memory Test Failure

This bit indicates that the PE has detected a failure during on its Memory Test. The Memory Test only runs during POST BIT.

Bit 2: Processor Test Failure

This bit indicates that the PE has detected a failure on its Processor Test. The Processor Test runs during POST, Periodic, and Initiated BIT.

Bit 3: Time-Tag Test Failure

This bit indicates that the PE has detected a failure on its Time-Tag Test. The Time-Tag Test runs during POST, Periodic, and Initiated BIT.

Bits 4-23: Reserved

Bit 24: POST BIT Failure

This bit indicates that the PE has detected a failure during POST BIT.

Bit 25: Periodic BIT Failure

This bit indicates that the PE has detected a failure during Periodic BIT.

Bit 26: Initiated BIT Failure

This bit indicates that the PE has detected a failure during Initiated BIT.

Bit 27: Reserved

Bit 28: POST BIT In-Progress

This bit indicates that the PE is currently running POST BIT.

Bit 29: Periodic BIT In-Progress

This bit indicates that the PE is currently running Periodic BIT.

Bit 30: Initiated BIT In-Progress

This bit indicates that the PE is currently running Initiated BIT.

Bit 31: Reserved

Root Signal Capture CSR Channel A: 0x0034

Signal Capture Discussion

Not all Alta boards/channels include the ADC needed for the Signal Capture Feature. See the individual board HW manual for information regarding the availability of this feature.

The Signal Capture Feature uses an analog-to-digital converter (ADC) to capture the electrical signal on the first RX channel of the card. The Signal Capture feature will capture 2048 samples at a rate of 2MHz, or 500 nanoseconds per sample. Therefore the sample buffer contains 1024 milliseconds of data. Each sample is an 8-bit (256 step) value representing the differential voltage on the first receive channel (RX channel 0). The Signal Capture data should be accurate to within 500mV, but is not calibrated.

The FIFO is in words (512 words) with 4 ADC bytes per word (the byte positions are provided below – “Root Signal Data”).

To convert the raw ADC data to a stub voltage representation, use the following formula:

Bus Voltage = (ADC data – 128) * 22 (voltage divider) * 2mv (step voltage)
Reducing the above gives:
Bus Voltage = (ADC data – 128) * 0.044

NOTE: The Signal Capture provides simple voltage and timing data and i**s NOT a calibrated, precision instrument.** The Signal Capture Feature does NOT replace a calibrated oscilloscope for voltage or timing measurements.

The following steps should be performed to acquire Signal Capture data from the PE.

1. Set the Trigger on Any Activity bit in the Signal Capture CSR
2. Wait for Data Ready bit to get set in the Signal Capture CSR
3. Read data from the Signal Capture Data Register. Note: The Data Register is 32-bit word contains four ADC byte samples.
4. Keep reading data until the FIFO Not Empty bit is set to zero by the PE.

Bit 0: Trigger on Any Activity: W

This bit set to a one will cause the signal capture detection circuit to trigger on any activity detected on the first RX channel. This bit will self clear once the trigger condition is detected.

Bit 1: Trigger on Label/SDI – Not Implemented

Bits 2-5: Reserved

Bits 6-15: Label/SDI Value

Bits 16-29: Reserved

Bit 30: FIFO Not Empty

This bit set to a one indicates that the Signal Capture data FIFO is not empty and that more data is available to be read by the user/API.

Bit 31: Data Ready: W

This bit set to a one indicates that a trigger has occurred and that Signal Capture data is available to the user/API.

Root Signal Data A: 0x0038

Bits 0-7: Data 0
Bits 8-15: Data 1
Bits 16-23: Data 2
Bits 24-31: Data 3

Root Signal Capture CSR RX Channel 0: 0x003C

Bits the same as RX Channel 0

Root 0x0040-0x0048 Reserved

Root Interval Timer: 0x004C

The Interval Timer provides a 24-bit, 1 µsec timer (~16 sec max) that can generate external triggers (pulse per second generator) and be synchronized to an external trigger. This can be ideal for an RS-485 or TTL external trigger output (controlled by the Root PE Control Register 0x0000) for synchronizing events (RX and TX or PB) events across device banks or cards. Also provided is an option to generate a hardware interrupt when the time value is reached (may be useful for applications where the user cannot have a software timer/signal to activate an event).

This register contains 6 control bits in the LSB positions and the 24 MSBs are the 1 usec clock ticks on when the timer would reset back to zero (so the timer will count up from zero and then reset when the 24-bit value is reached – this reset can cause the interrupt or external trigger). The bit definitions follow:

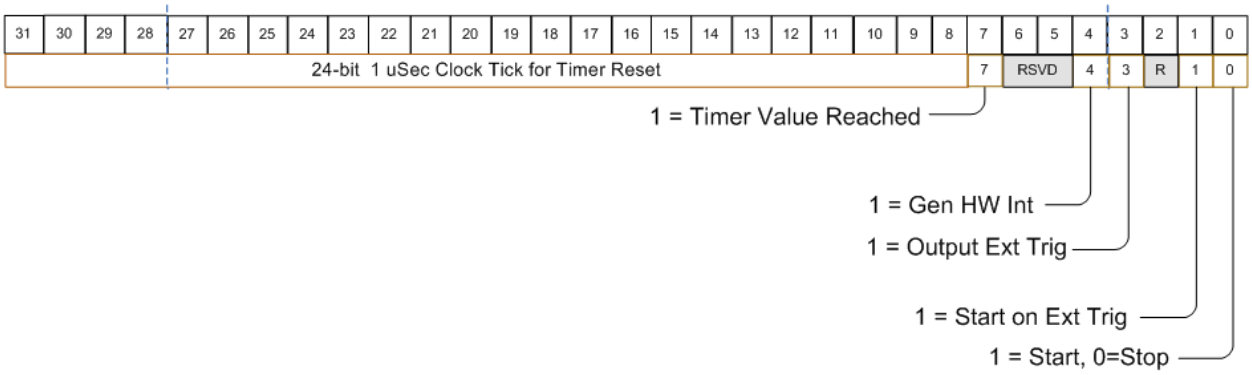


Figure Root PE-4: Interval Timer Register

Bit 0: Timer Start/Stop: W

This bit is set to one by the user/API to start the timer and set to zero by the user/API to stop/reset the timer. The user/API must stop the timer to change any other control bit values, then start again. This bit must be set to one along with Bits 1-4 options.

Bit 1: Start on Ext Trig: W

This bit is set to one by the user/API to have the timer not start until an input trigger is detected. Bit 0 must also be set to one for this action.

Bit 2: Reserved

Bit 3: Output Ext Trig: W

This bit is set to one by the user/API to cause an external trigger output at the timer reset rate (24-bit 1 µsec time). For TTL or RS-485 trigger selection, see Root PE Control Register 0x0000 settings. Bit 0 must also be set to one for this action.

Bit 4: Gen HW Int: W

This bit is set to one by the user/API to force the PE to generate a hardware interrupt for this channel device at the program time rate. **BE CAREFUL WITH THIS SETTTING. An interrupt time that is too fast will probably cause the computer/OS to crash!!** See Interrupt Section later in this manual for description on how this event is logged into the Interrupt Queue. Bit 0 must also be set to one for this action.

Bits 5-6: RESERVED

Bits 7: Time Value Reached: W

This bit is set to one by the PE when the 24-bit time value has been reached. The user should reset this bit to zero as desired for their application.

Bits 8-31: Time Interval Reset Value: W

These 24 bits are set by the user/API to the desired 1 µsec tick value has been reached – the timer then resets back to zero and starts counting up again as long as Bit 0 is set to one. The maximum value is ~16 seconds (2^{24}/1000000). When the timer reaches maximum value, then the options in bits 3-4 can execute and Bit 7 will be set to one.

PTP IEEE-1588 Registers: 0x00C0-0x00F8

These 15 registers are for ENET/ENET2 devices for key PTP IEEE-1588 Time Sync and Control-Status (CSR) registers. ENET devices can synchronize time via 1588 grand masters in a similar manner to IRIG synchronization. If engaged (via user software API settings), this overrides IRIG sync decoding.

For details on implementation, please contact Alta for access to the PTP IEEE-1588 Design Implementation Application Note and reference the appropriate sample C programs. Contact Alta at alta.support@altadt.com or 888-429-1553 x2.

<~~~>

AltaCore-ARINC: Transmit (TX)

TX Data Structures

This section reviews the data structures for programming TX Channels. The user/API has great flexibility in defining ARINC labels/words for transmission (TX), including the frequency settings for label/word list(s) and the raw signal encoding of label/words for each channel. The ***AltaCore-ARINC*** can transmit almost any bit/word encoding for 1-32 bit words in single transmission or frequency controlled packets.

> **NOTE:** The Root TX Registers are also shared with the Playback (PB) Registers, and thus, Figure TX-1 will have acronyms with TX/PB designations. For this section, ignore the PB designation. Please see the Playback section for descriptions of Playback operations.

TX Basics

Channel transmission is controlled through a series or Root PE and Root TX registers, and a key Extended Memory linked-list data structure known as **TX Control Blocks (TX-CB)**. The Root TX Registers also provide a method to perform a one –shot, aperiodic transmission or a label/word. Often, one-shots are the only method required for simple test or host timing controlled applications – simply load up a label/word packet (called a **Transmit Packet – or TxP**) and transmit away.

Through **TX-CBs**, the user/API has a wealth of options to transmit lists of labels/words (**TxPs**) at 500 usec tick intervals for full frequency control. One-shot TxPs can be used in conjunction with TX-CBs to inject aperiodic label/words.

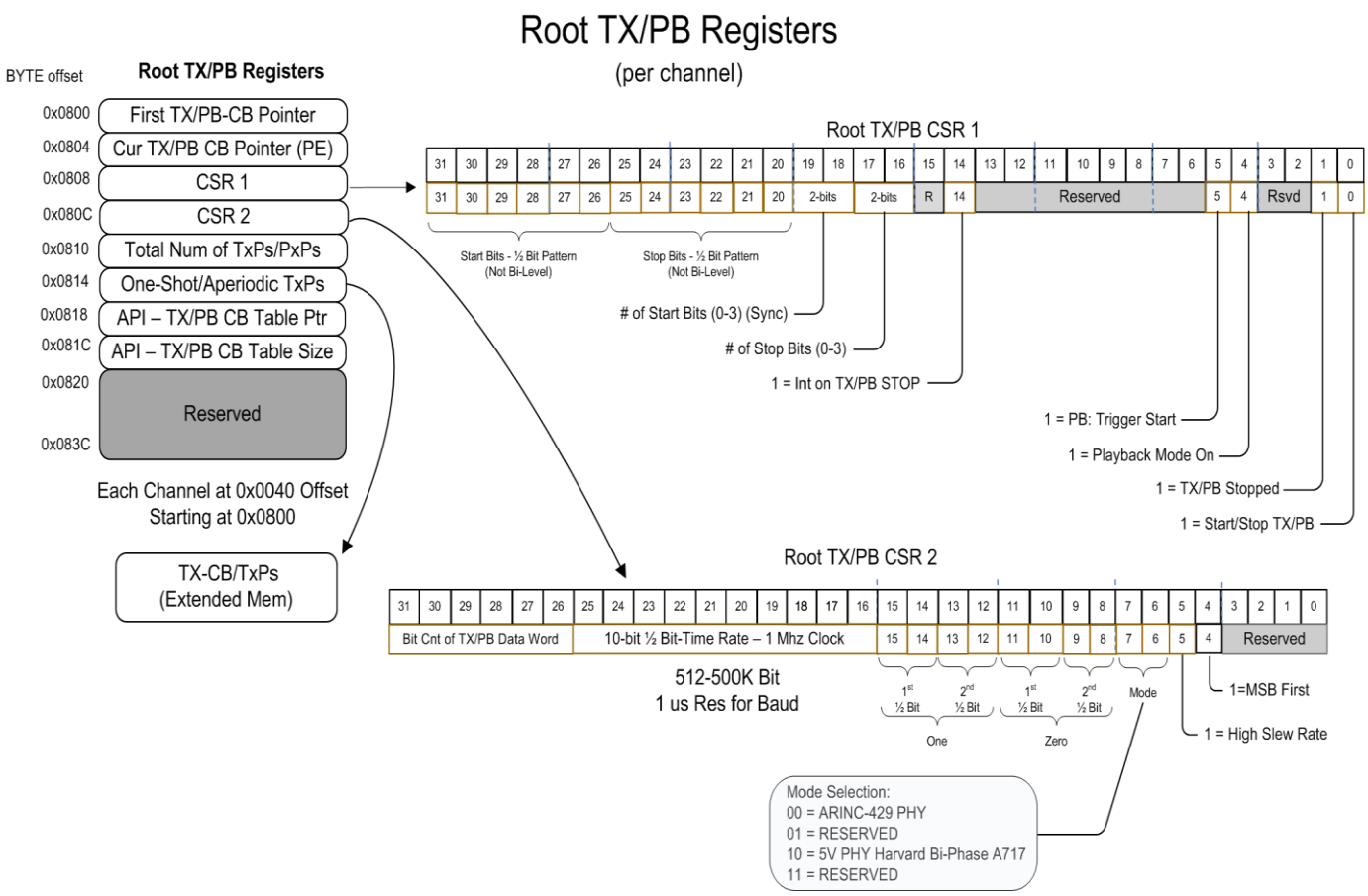


Figure TX-1: Root TX Registers and Root TX CSR

Root First TX-CB Address: W: 0x0800

This word is set by the user/API with the address of the first TX-CB to be executed when Bit 0 of the Root TX CSR is set to one by the user/API.

Root Current TX-CB Pointer: W: 0x0804

This word is set by the PE whenever the next TX-CB is sequenced for execution. The user/API can read this word to see where the TX transmission sequence is currently located.

When the PE halts transmission (null pointer in the last TX-CB Head Pointer, Root TX CSR Bit 0 set to zero), then this value will be set to 0x00000000. Bit 1 of the Root TX CSR (described next) is set to one when ALL TX processing has completed for the respective channel – this bit should be read by the user/API to signal when the PE has finished TX processing.

Root TX CSR1: W: 0x0808

Root TX CSR1 word controls global TX functions such as start/stop, interrupts options, error detection flags, framing controls and defines stop/start bits for label/words of TX channel. The following paragraphs describe the control/status bits.

Bit 0: Start/Stop TX: W

1=Start TX. This Bit is set by the user/API to start and stop the TX transmission. When this bit is set to a one, the PE will start TX processing with the TX-CB pointed to by the First TX-CB Address at location 0x00000100. The user/API clearing this bit to a zero will cause the TX to halt at the end of the current TX-CB (including any TxP time value). Other bit options in this Root TX CSR and TX-CB CSR will direct TX transmission personality (wait for external or software triggers, etc...).

Setting this bit to a zero will cause the respective transmit line driver to put in a tri-state mode.

> **NOTE:** TX-CB transmission will start within 250 µsecs of this bit being set (usually less than 20 µsecs, but channel execution order may induce delays to 250 µsecs). When turning the channel off, the user/API must allow 20 µsecs after Bit 1 (TX Stopped Bit described next) has been set to one before turning the channel back on again (this extra time ensures the PE is ready for transmission).

> **NOTE:** The user can have precise transmission starting times between channels by setting the Trigger In bit on a TxP and then setting the Force External Trigger bit in the Root PE Control Register. An example would be to have 4 transmit channels and have the first TxP in each channel's TX-CB with the Trigger In Bit set. Then the application could have an external trigger or set the Force External Trigger to one. The label/words would transmit within 2 µsecs between channels.

Bit 1: TX Stopped

1=TX Stopped. This bit is set to a one by the PE when transmission has stopped for any reason and has completed all execution overhead. The user/API should clear this bit to zero before the start/restart of the TX-CB.

Bits 2-3: Reserved

Bit 4: TX Playback On: W – See Playback Section for Details

For TX functions, set this bit to zero.

Bit 5: Do Not Reset PB Clock at Start: W - See Playback Section for Details

For TX functions, set this bit to zero.

Bit 6: PB: Skip PxPs With Time Back-Up: W - See Playback Section for Details

For TX functions, set this bit to zero.

Bit 7-13: Reserved

Bit 14: Interrupt of TX Stopped: W

1= Interrupt on TX Stopped. This bit is set to a one by the user/API to direct the PE to generate an interrupt whenever the PE has stopped transmission (for any reason).

Bit 15: Reserved

Bits 16-17: Number of Stop Bits: W – NI in the version.

These bits are set by the user/API with the number of whole stop bits for label/word. ARINC-429 and 717 would set this value to a zero. Each label/word can have 0-3 stop bits, and the ½ bit logic pattern is set in corresponding bits 20-25. This feature is mainly provided for future RS-232/422/485 style communications and for rare ARINC encodings.

Bits 18-19: Number of Start Bits: W – NI in the version.

These bits are set by the user/API with the number of whole start bits for label/word. ARINC-429 and 717 would set this value to a zero. Each label/word can have 0-3 start bits, and the ½ bit logic pattern is set in corresponding bits 26-31. This feature is mainly provided for future RS-232/422/485 style communications and for rare ARINC encodings. If there are start bits designate, these could also be used as a sync pattern (for example, a 1553 encoding would have a value of 3).

Bits 20-25: Stop Bits - ½ Bit Pattern: W – NI in the version.

These bits are set by the user/API to direct the PE to transmit the ½ bit stop pattern. These values are NOT bi-level values, but are the actual logic transmission pattern for the Stop Bit(s) in ½ bit intervals. The pattern is transmitted from left to right (MSB first) the number of whole bits programmed in bits 16-17.

Bits 26-31: Start Bits - ½ Bit Pattern: W – NI in the version.

These bits are set by the user/API to direct the PE to transmit the ½ bit start pattern. These values are NOT bi-level values, but are the actual logic transmission pattern for the Start Bit(s) in ½ bit intervals. The pattern is transmitted from left to right (MSB is the FBT) the number of whole bits programmed in bits 19-20.

Root TX CSR2: W: 0x080C

Root TX CSR2 word defines additional label/word characteristics for the TX channel, such as MSB/LSB direction, Slew Rate (high or low), bit encoding, bit rate and bits per word.

Bit 4: 1=MSB First: W

This bit is set to a one by the user/API to direct the PE to transmit the label/word from bit 31 (working down to bit zero). **The default and proper setting for most protocols/systems is for this bit to be set to a zero** (which directs the PE to transmit the label/word starting with bit 0, left-most, and to work up to bit 31).

Bit 5: 1= High Slew Rate: W

This bit is set to one by the user/API to direct the PE to have a high slew rate for 100K ARINC-429 baud rates. This bit should be set to a zero for rates below 20K bits per second.

Mode Encoding Bits

Bits 6&7 allow the user to set the encoding mode for bits and change the physical layer (PHY) connections. The user has the selection for normal 429 PHY transmitters or split-+-leg transmission using two channels for Harvard Bi-Phase transmission (the first four channels of each bank can have only their + legs specified to create a 0-5V differential signal).

Values 0x1 and 0x3 are RESERVED values and should not be used.

Bits 6 & 7 = 0x0: Normal ARINC-429 Physical Receiver

These bits should be set to zero by the user/API for normal ARINC-429 receiving. The ½ bit decoding bits (8-11) should set to Bi-Polar values (0x84).

Bits 6 & 7 = 0x2: 5V Harvard Bi-Phase (717) Only – Two TX Channels

These bits are is set to two by the user/API to direct the PE. This setting will force a TX channel pair (channels 1&2 and 3&4 pairs of each bank) to differential drive the 0-5V + leg of the ARINC-429 drivers (the odd channel + is the positive differential signal and the even channel + leg is the negative differential signal) to create Harvard Bi-Phase 717 encoding. This setting is provided for older ARINC-717 DFDRS systems. If this mode is set, the ensuring half bit encoding values are ignored. This setting ONLY applies to the first four TX channels of each bank.

Bit Encoding

AltaCore-ARINC can transmit virtually any bit pattern by allowing the user/API to designate the logic “1” and “0” encoding pattern of the bi-level line drivers chips. The Alta ARINC card line drivers (TX chips) have four states (two bit patterns) to drive the +/- legs of the ARINC channel:

- 11 = Tri-State (which is not used as part of the bit encoding scheme)
- 10 = High Voltage
- 01 = Low Voltage

- 00 = Mid Level Voltage (ground)

The user/API will set the values of bits 8-15 with the half bit line driver patterns to designate the encoding of a full bit for a logic 1 and 0. These encoding patterns are used in conjunction with the value in bits 15-24 that designate the ½ bit transmission rate. Thus, the user/API will program the PE with the ½ bit transmission rate, as well as, the ½ bit encoding scheme to allow virtually any NRZL or Bi-Polar/Phase encoding scheme allowed by the ARINC line driver's slew rates and frequency response. ARINC-429 standard encoding is Bi-Polar.

The Tri-State (11) value cannot be used in these encoding values.

Bits 8-11: Logic Zero Encoding: W

These bits direct the PE on how to encode a logic 0 (zero). The user/API sets the half bit patterns that make a logic 0. ARINC-429 should have a 4 bit pattern of 01-00.

Bits 8-9: Zero - 2nd ½ Bit: W

These two bits are set by the user/API with the 2nd ½ bit pattern for a logic zero. ARINC-429 should have a two bit value of "00."

Bits 10-11: Zero - 1st ½ Bit: W

These two bits are set by the user/API with the 1st ½ bit pattern for a logic zero. ARINC-429 should have a two bit value of "01."

Bits 12-15: Logic One Encoding: W

These bits direct the PE on how to encode a logic 1 (one). The user/API sets the half bit patterns that make a logic 1. ARINC-429 should have a 4 bit pattern of 10-00.

Bits 12-13: One - 2nd ½ Bit: W

These two bits are set by the user/API with the 2nd ½ bit pattern for a logic one. ARINC-429 should have a two bit value of "00."

Bits 14-15: One - 1st ½ Bit: W

These two bits are set by the user/API with the 1st ½ bit pattern for a logic one. ARINC-429 should have a two bit value of "10."

Programming TX Bit (Baud) Rate

Bits 16-25: ½ Bit-Time Rate: W

These 10 bits encode the bit rate for the TxP data transmission. The 10-bit value represents the clock tick value for a ½ bit time based on a 1 (one) µsec clock. The minimum setting is a value of one for 2 µsec for a ½ bit (resulting in a 250K bits per second – 200K is specified maximum of the card). The maximum setting

of decimal 1024 would provide represent 1024 µsec for a ½ bit resulting in a bit rate of 512 bits per second. This half bit clock tick will drive the bit encoding values as set in bits 8-15 described above.

For ARINC-429 12.5K baud, the 10 bit value should be: 40 (40 µsec ½ bits).
For ARINC-429 100K baud, the 10 bit value should be: 5 (5 µsec ½ bits).
For ARINC-717 768 baud, the 10 bit value should be: 651 (651 µsec ½ bits).

Programming TX Bits Per Word (word length)

Bits 26-31: Bit Count for TX Data Word: W

These bits are set by the user/API with the actual bit count to be transmitted (1-63). This number includes parity bit (if designated). Bit 6 MSB setting will direct bits being shifted out from bit 0 (normal – LSB first) or bit 31 (MSB first). Normal setting for ARINC-429 is 32 (which include parity).

If the value is greater than 32 (for injecting bit count errors for testing), then a zero value will be shifted out for the 33-63 bit count.

Root One-Shot/Aperiodic TxP: W: 0x0814

This register is set by the user/API with the address of a single TX-CB/TxP List (to execute 1-N TxPs as defined by the TX-CB). The TX-CB and its' TxP label/word list will be executed at the next 500 usec clock tick with a resolution of 50 usecs. If a periodic chain of TX-CBs are being executed, this TX-CB will not be transmitted until after the current TX-CB has completed (including the TX-CBs label/word list). Many applications will only use this method for application controlled transmissions.

Root API - TX-CB Table Pointer: W: 0x0818

The API allows the user/API to reference label/word lists (TX Control Blocks) by a TC-CB number rather than requiring the user/API to work directly with memory offsets on the board for the TX-CBs. Therefore, the API creates a table in board memory where it stores the memory offset to the TX-CB for each TX-CB number. This register is used to store the pointer to the start of this table. By placing this information in board memory (rather than keeping it in a software variable in the API), this enables the user/API to find the TX-CB when inspecting board memory to troubleshoot problems. The user/API can look at a memory dump to verify that the data structures are setup correctly and can follow this pointer to find all TX-CB that the API has allocated on the board.

Root API - TX-CB Table Size: W: 0x081C

This register is used with the previous register to define the TX-CB table used by the API to manage TX-CB chains. This register provides the size (in 32-bit words) of the TX-CB table. This allows the API to check for invalid TX-CB numbers (a TX-CB number that exceeds the TX-CB table size is invalid).

RESERVED: 0x0820-0x083C

Transmit Control Blocks (TX-CB)

TX-CBs provide timing and transmission options for a table of 1-N TxPs. A TX-CB is an array of control, status and pointer words that are link-listed to create a transmission chain of 1-N TX-CBs. Each TX-CB references (points to) a 1-65535 TxP Table.

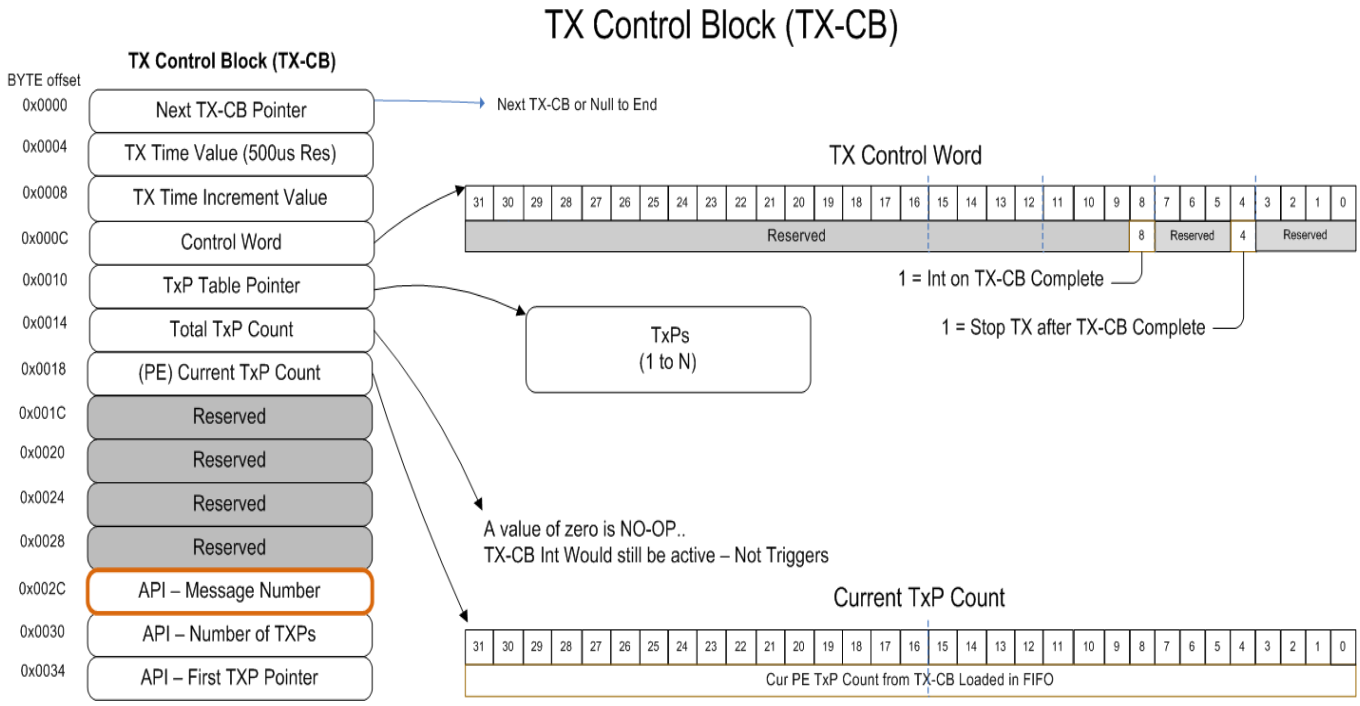


Figure TX-2: TX Control Block (TX-CB)

The PE scans the entire TX-CB chain every 500 usecs to see if a TX-CB is scheduled for transmission. Each TX-CB has a time value and increment to allow the user to precisely schedule a list of TXPs associated with each TX-CB. Further, each TXP has an inter-Label/word time gap that can be specified (to insert gaps between TXPs of a specific TX-CB).

> **EXAMPLE**: Often a system is based on a list of label/words that occur at different time periods (frequency) within a one second master time frame. A TX-CB would represent each different time period for the list of label/words to be transmitted. The resolution of time period definition for a TX-CB is 500 usec; so if the user wants a TX-CB to be executed at 10 msecs, the TX Time Value (offset 0x0004 in the TX-CB) would be set to 20. If the TX-CB is to execute EVERY 10 msecs, then the Time Increment Value (offset 0x0008 of the TX-CB) would also be set to 20. Each TX-CB can have a different time period and increment value. The TX-CB chain links should be organized in ascending order from low time periods to

high time periods. At least 1 msec of time separation is recommended between TX-CBs to maintain highly accurate TX-CB start times.

Even more accurate label/word timing can be achieved by packing TXPs in to single/common TX-CBs and specifying the Pre-TX Delay Time of the TXP (TXPs are loaded in to a hardware controlled FIFO where the delay time is counted down and then transmitted for extremely accurate gap controls – within 1 usec).

TX-CB and TXP definitions are detailed in the following paragraphs.

Next TX-CB Pointer: W: 0x0000

The first word of a TX-CB is a pointer to the next TX-CB. The last TX-CB in the chain should have a Next TX-CB value of Null (zero).

TX Time Value: W: 0x0004

This word represents the time period in 500 usec ticks to transmit this TX-CB chain's TXP label/word list from the beginning of chain execution (not a gap value). Example, if the start time period is 100 msec, then this value would be 200. If the time value has expired from the beginning of the chain execution, then the TXP list will be immediately be executed. The PE will add the TX Time Increment Value to this word after TX-CB execution. This allows the TX-CB schedule time to increment to a user defined value. The user should not stop/start TX transmission without resetting this time value.

TX-CB execute will be highly accurate (within 50 usecs) with at least 1 msec of time separation between TX-CB TX Time Values.

TX Time Increment: W: 0x0008

This word represents the increment or interval (in 500 usec ticks) that the user wants the TX-CB TXP label/word list to be executed. This value is added to the TX Time Value (see above) to track the next time period for TX-CB transmission.

If label/word transmission time accuracy greater than 1 msec is desired, then it is recommended that TXPs be packed in to common TX-CB lists and inter-TXP gap times be utilized to control gap times. For most systems, TX-CB time intervals of 1 msec are more than sufficient.

Control Word: W: 0x000C

This word is a packed bit data structure that directs execution of the TX-CB. The bits are defined as follows.

Bits 0-3: Reserved

Bit 4: 1=Stop TX after TxP Complete: W

This bit is set to a one by the user/API to direct the PE to stop transmission when the TxP label/word list of this TX-CB has completed. This bit is most often used to allow a “one-shot” transmission for a TxP list (or chain of TX-CBs). Some user applications prefer to control transmission timing (for one or more label/words) and this bit can be used to prevent cyclic transmission of TX-CBs. (If you want a simple one-shot action for a list of TxPs without period timing, then setup a single TX-CB with the TxP list, set this bit to one, and set the TX Time Value, 0x0004, to one).

Bit 8: 1=Interrupt on TxP Complete: W

This bit is set to a one by the user/API to direct the PE to generate an interrupt when the TX-CB is complete.

Bits 9-31: Reserved

TxP Table Pointer: W: 0x0010

This word is set by the user/API with the starting address of the TxP Table for this TX-CB. The TxP Table is a simple array (contiguous area of memory) of 1-N TxPs for transmission. The TX-CB controls the timing/frequency and other user options in executing a TxP Table. The number of TxPs in the Data Table is programmed in word offset 0x0014 later in the TX-CB.

TxP Count: W: 0x0014

This word is set by the user/API with the total number (N) of 4 word TxPs pointed to by the TxP Table Pointer at offset 0x0004. The value is a 32-bit unsigned integer or 1-N. (Card/device memory probably the maximum value to well the maximum – see your card configuration for memory details – most cards have 1 Mbyte per 16 channel bank, so the user/API must limit the actual value to manage the memory for all channels).

A value of zero will cause the TX-CB not to be executed and transmission logic will move on to the next TX-CB.

(PE) Current TxP Count: 0x0018

This word is set by the PE with the next TxP count (used as a 4 word offset in the TxP Table) to be executed. The PE increments this value when the current TxP is complete. This is a count-up value and when this value equals TxP Count (0x000C), then transmission will move on to the next TX-CB. The PE will reset this value to a zero at the start of TX-CB execution.

RESERVED: 0x001C-0028

API – TxP Number: 0x002C

The API uses integer numbers rather than memory pointers to identify TX-CBs. This word is used by the API to store the referenced TX-CB number.

API – Number of TxPs: 0x0030

The API allocates TxP for each TX Data Table (TX-CB). This word is used by the API to store the number of TxPs allocated for this TX Data Table.

API – First TxP Pointer: 0x0034

This word provides a pointer to the first TxP for this TX Data Table. The API uses this pointer to read or write TxP data for the TX Data Table.

Transmit Data Table

Each TX-CB has a pointer word that provides the first address offset to a Transmit Data Table (simple array) of 1-65535 Transmit Packets. TxPs are detailed in the next paragraph.

TxP: Transmit Packet Data Structure

A Transmit Packet (TxP) is a four word array that provide control, trigger, interrupt, parity, timing and data values for transmission of the label/word. The first word provides the user options on control, the second word is a reserved word, the third word is a 32-bit, 100 nsec pre-transmit time delay value and the fourth word is the data values. Channel Root TX CSR words provide other key setup variables for encoding, bit rate and word length.

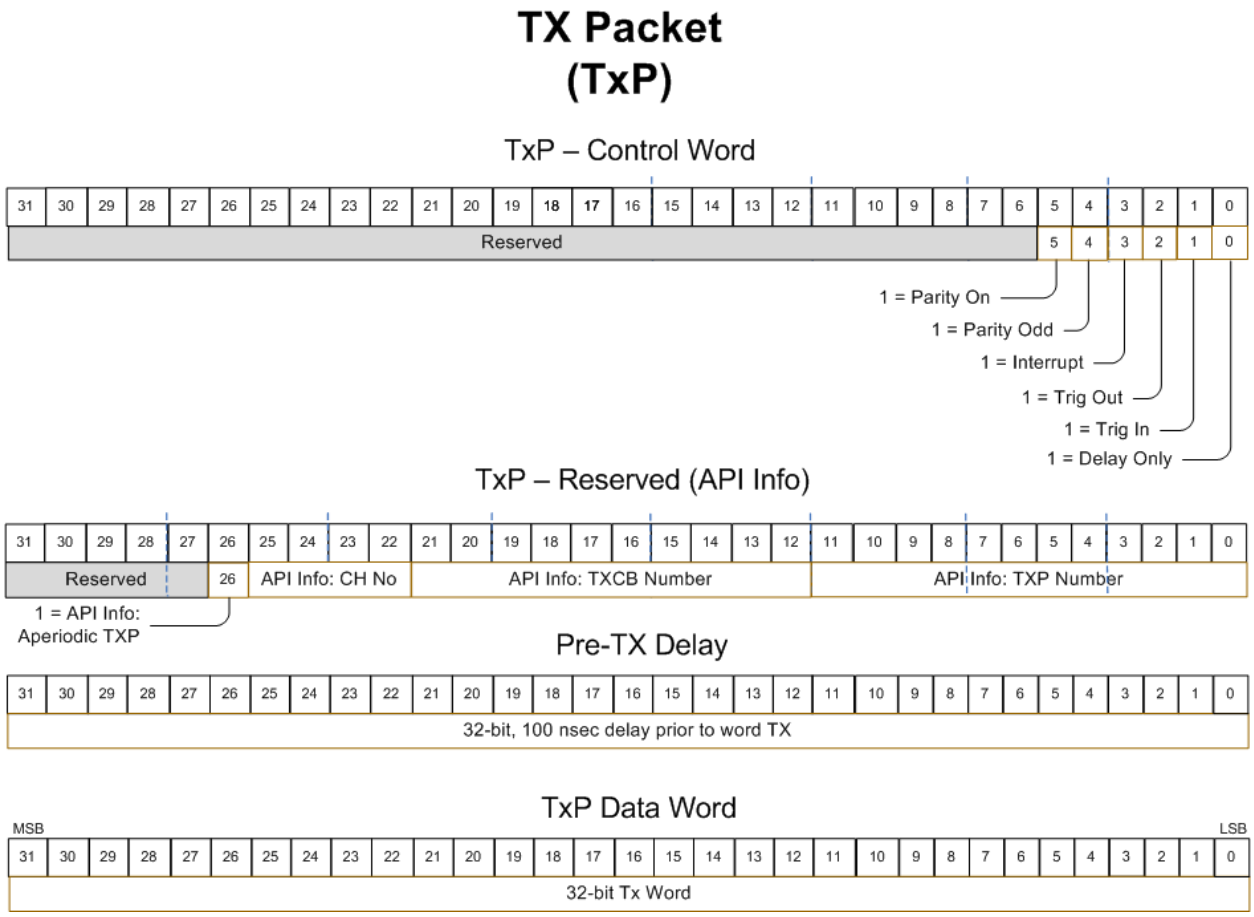


Figure TX-3: Transmit Packet (TxP)

For ARINC-717 operations, the user would want to configure separate TX-CBs for each subframe of data and have no time gaps between TXPs. ARINC-429 label/words should have at least 4 bit times of time gap between TXPs.

TxP Control Word: W: 0x0000

The first word of the TxP is a packed data structure that provides transmit options and sets parity encoding. These bit settings are detailed in the following paragraphs

Bit 0: 1=Delay Only: W

This bit is set to a one by the user/API to direct the PE to NOT transmit a label/word value and only execute the time delay (idle/mid voltage state) programmed in TxP word 3. Trigger and Interrupt settings for the TxP are still active.

Bit 1: 1=Trig In: W

This bit is set to a one by the user/API to direct the API to stall TxP processing and transmission until a trigger in event has occurred.

Bit 2: 1=Trig Out: W

This bit is set to a one by the user/API to direct the PE to pulse the channel's external trigger line. This is a 1 µsec second pulse.

Bit 3: 1=Interrupt: W

This bit is set by the user/API to direct the PE to generate an interrupt event when the word has finished transmitting. Hardware interrupt signal is controlled by the Root PE Control and Status Register. See Interrupt section of this manual for details on the interrupt events and the interrupt queue.

Bit 4: 1 = Parity On: W

This bit is set to a one by the user/API to direct the PE to add a parity bit to the end of the word (prior to stop bits). The value of parity, odd or even, is determined by the setting of the next bit. ARINC-429 default is one for this bit. This bit could be set to zero to inject a "no parity" error condition for ARINC-429 labels.

Bit 5: 1=Parity Odd: W

This bit is set to a one by the user/API to direct the PE have an odd parity value. A value of zero would force an even parity value. The default for ARINC-429 is odd parity (this bit set to one). This bit can be set to zero to inject a parity error for ARINC-429 labels.

Bits 6-31: Reserved

TxP Word 2 – Reserved (API) Info: 0x0004

This word is reserved for API information to track the TXP in software handlers. The **AltaAPI** loads channel, TX-CB, TXP number in this word so that user software functions can quickly reference this TXP in various functions (like interrupt service events). Bit 26 is set to one by the **AltaAPI** if the TXP is an Aperiodic TXP (verses a TX-CB TXP). The PE does not process any of these values – these are user/API set information bits.

TxP Time Gap: W: 0x0008

The third word of the TxP is a set by the user/API for the time gap prior to label/word transmission. The value is 32-bit, 100 nsec. For standard ARINC-429, there this value should be at least the time value for 4 bit times (usually much greater). A value of zero will cause contiguous transmission between TxPs of the same Transmit Data Table.

TxP Data Word: W: 0x000C

This word is set by the user/API with the bit values for the label/word transmission. If bit 6 (MSB control bit) of the Root TX CSR2 word is set to a zero then bits will be shifted out starting with bit zero (right most). This is the normal setting for standard ARINC-429 and 717. If the MSB control bit is one, then transmission will shift from the left-most bit 31.

The PE will transmit this word has a raw bit pattern, so for ARINC-429 applications, the user/API will need to have the label field MSB-LSB swapped prior to loading to this transmit word.

<~~~>

AltaCore-ARINC: Playback (PB)

PB Data Structures

This section reviews the data structures for programming Playback functions using TX Channels. Playback is essentially a clock synchronized transmission function where the user often wants to replay recorded RxP files for re-transmission. This function can also be used as an ARINC transmitter for applications where time correlated transmission (such as static models) is desired over setting-up TX frequency controls using TX-CBs (with this playback function, the user can simple have lists of RxPs with time stamp values to control their transmission).

> **NOTE:** The Root TX Registers are also shared with the Playback (PB) Registers, and thus, Figure TX-1 will have acronyms with TX/PB designations. For this section, ignore the TX designation. Please see the Transmit (TX) section for descriptions of Transmit operations.

PB Basics

Channel transmission is controlled through a series or Root PE and Root PB registers, and a key Extended Memory linked-list data structure known as **PB Control Blocks (PB-CB)**. Through **PB-CBs**, the user/API can build tables of transmission packets called Playback Transmission Packets (**PxPs**). **PxPs** are essentially RxP structures used to transmit a user generated or file archived list of RxPs (error injection is not reproduced, but the user can set basic bit-rate and encoding formats for the playback session just as in regular TX setup).

Playback Transmission Timing

The PE playback logic for time controlled transmission is fairly straight forward. When the PE reads a PXP, the time value and the ARINC word/label is loaded to a lower level PE transmission encoder. The encoder does a tight hardware 20 ηsec comparison of the time value (regardless of possible Root PE Control Word external clock time setup/synchronization), and when the PXP time value is <= to the internal PB clock, then the word of the PXP is transmitted.

The user has an option to “Skip PXPs with Time Back-up” in the Root PE Control Word. If this bit is set, then the PE Playback logic changes slightly. If the bit is set, then the PE will look at each PXP time value and compare if the time is less-than (<) the PB internal clock: if the time is less-than, then it is skipped completely (and the interrupt setting of the skipped PXP is ignored); if the time is => (equal to or greater than), then the PE loads the PXP to the lower-level transmission encoder and the 20 ηsec time comparison logic takes over to very tightly control/compare when the word starts to be transmitted. Once the PXP time and data information is loaded to the low-level PE encoder, transmission accuracy is within 10 µsec.

PXP Timing Discussion – Relative Timing

For default, relative timing the PE Playback engine has an internal clock that is used to compare against each PXP's user set time. For most applications using relative timing, this PE Playback internal clock is reset to zero when the Playback function is started – so future PXP transmission times must be from a relative value of zero. When Playback starts, the clock resets to zero and then the PE compares each PXP Time High/Low (set by the user) to the internal clock value: If the PXP Time High/Low is <= to the PE Playback internal clock, then the PXP is executed (transmitted). If the PXP Time High/Low is > than the internal clock, then the PE stalls/loops until the PXP time is <= to the internal clock.

AltaAPI PXPWrite() automatically takes a RXP string and makes the offset-zero time calculations, so the user can simple pass RXPs from a file to the API call and not worry about the time offsets to zero. If the user wants to have a start delay, then the user should increase the RXP Time High/Low by the desired start delay time prior to calling the AltaAPI Playback functions.

Relative Timing is the default setting of PB data structures (control bits are set to zero).

PXP Timing Discussion – Absolute Timing (AT)

The ***AltaCore*** PE also has options for users to have PXP transmission timing controlled by the absolute value of the PXP time words (which probably came from a RXP time value). This simple means the PB internal clock is not reset to zero and the user must preset the PB time value through the Root PE Control Word time control bits. Once started, the PB internal clock will count up and look for PXPs with time values <= to the internal PB clock (just like with relative timing).

The user must set the Root PE Control Word "Do Not Reset PB Clock at Start" bit to have absolute timing. The user can also set the "Skip PXP with Time Back-up" for the PE to skip over PXPs that have a time value < than the internal clock – this is useful only for applications where the user as preload PXPs and does not want transmission started until a certain time has elapsed (most applications will simply want to "load and go").

The following paragraphs detail playback data structures.

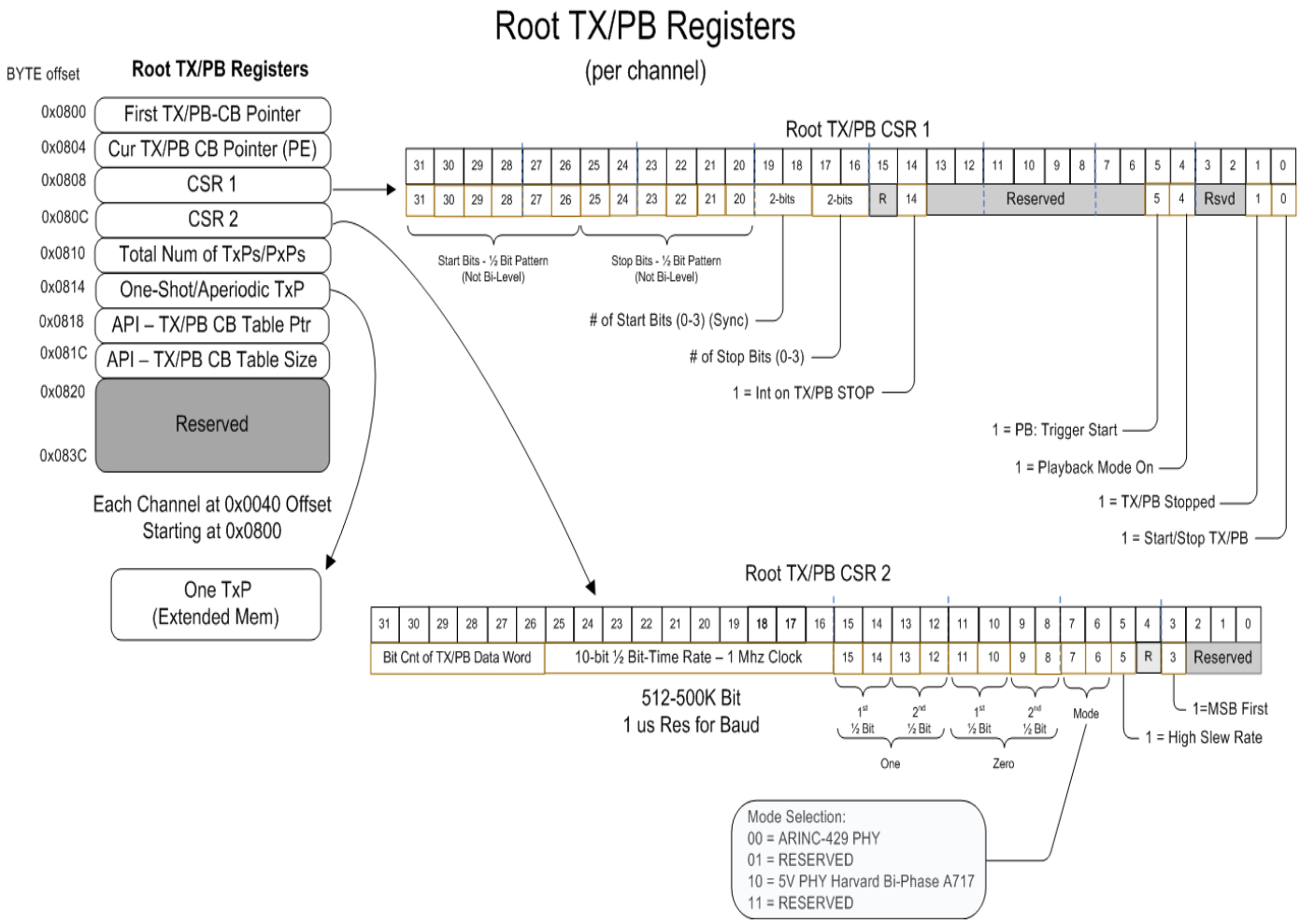


Figure PB-1: Root TX & PB Registers

Root First PB-CB Address: W: 0x0800

This word is set by the user/API with the address of the first PB-CB to be executed when Bit 0 of the Root TX/PB CSR is set to one by the user/API.

Root Current PB-CB Pointer: W: 0x0804

This word is set by the PE whenever the next PB-CB is sequenced for execution. The user/API can read this word to see where the PB transmission sequence is currently located.

When the PE halts transmission (null pointer in the last PB-CB Head Pointer, Root TX/PB CSR Bit 0 set to zero), then this value will be set to 0x00000000. Bit 1 of the Root TX/PB CSR (described next) is set to one when ALL PB processing has completed for the respective channel – this bit should be read by the user/API to signal when the PE has completed.

Root TX/PB CSR1: W: 0x0808

Root TX/PB CSR1 word controls global PB functions such as start/stop, interrupts options, error detection flags, framing controls and defines stop/start bits for label/words of TX channel. The following paragraphs describe the control/status bits.

Bit 0: Start/Stop TX/PB: W

1=Start TX. This Bit is set by the user/API to start and stop the PB transmission. When this bit is set to a one, the PE will start PB transmission starting with the PB-CB pointed to by the First PB-CB Address at location 0x00000100. The user/API clearing this bit to a zero will cause the PB to halt at the end of the current PB-CB (including any TxP time value). Other bit options in this Root TX/PB CSR and PB-CB CSR will direct transmission personality (wait for external or software triggers, etc...).

Setting this bit to a zero will cause the respective transmit line driver to put in a tri-state mode.

NOTE: PB-CB transmission will start within 250 µsecs of this bit being set (usually less than 20 µsecs, but channel execution order may induce delays to 20 µsecs per channel). When turning the channel off, the user/API must allow 20 µsecs after Bit 1 (TX/PB Stopped Bit described next) has been set to one before turning the channel back on again (this extra time ensures the PE is ready for transmission).

NOTE: The user can have precise transmission starting times between channels by setting the Trigger In bit in CSR1 and then setting the Force External Trigger bit in the Root PE Control Register to start PB transmission across channels. The label/words would transmit within 50 µsecs between channels.

Other methods to synchronize playback between channels is to use an external clock and Enable Clock Trigger feature. By using an external clock or trigger, the first PxP (once playback is turned on with this Start bit), will be loaded in the encoder transmit FIFOs, but the transmission will occur until the PxP time is met. The user can jam the playback clock to a known value and have all the channel PxP times set to the relative or absolute time value for transmission (and the time sequencing will be very precise). The must have at least 25 usecs inter PxP gap times to maintain precise transmission between PxPs/channels.

Bit 1: TX/PB Stopped

1=TX Stopped. This bit is set to a one by the PE when transmission has stopped for any reason and has completed all transmission execution and overhead. The user/API should clear this bit to zero before the start/restart of the PB-CB.

Bits 2-3: Reserved

Bit 4: TX Playback On: W

This bit is set to one by the user/API to start playback function for this channel. This bit would be ignored is regular TX functions are executing. Each TX channel being used for playback must have this bit set, and then the Playback (PB) On but in the Root PE Control would must be set to start playback transmission across channels.

Bit 5: PB: Trigger Start: W

This bit is set to one by the user/API to direct the PE to delay playback transmission until a trigger input is set.

Bit 6-13: Reserved

Bit 14: Interrupt of TX Stopped: W

1= Interrupt on PB Stopped. This bit is set to a one by the user/API to direct the PE to generate an interrupt whenever the PE has stopped transmission (for any reason).

Bit 15: Reserved

Bits 16-17: Number of Stop Bits: W – NI in the version.

These bits are set by the user/API with the number of whole stop bits for label/word. ARINC-429 and 717 would set this value to a zero. Each label/word can have 0-3 stop bits, and the ½ bit logic pattern is set in corresponding bits 20-25. This feature is mainly provided for future RS-232/422/485 style communications and for rare ARINC encodings.

Bits 18-19: Number of Start Bits: W – NI in the version.

These bits are set by the user/API with the number of whole start bits for label/word. ARINC-429 and 717 would set this value to a zero. Each label/word can have 0-3 start bits, and the ½ bit logic pattern is set in corresponding bits 26-31. This feature is mainly provided for future RS-232/422/485 style communications and for rare ARINC encodings. If there are start bits designate, these could also be used as a sync pattern (for example, a 1553 encoding would have a value of 3).

Bits 20-25: Stop Bits - ½ Bit Pattern: W – NI in the version.

These bits are set by the user/API to direct the PE to transmit the ½ bit stop pattern. These values are NOT bi-level values, but are the actual logic transmission pattern for the Stop Bit(s) in ½ bit intervals. The pattern is transmitted from left to right (MSB first) the number of whole bits programmed in bits 16-17.

Bits 26-31: Start Bits - ½ Bit Pattern: W – NI in the version.

These bits are set by the user/API to direct the PE to transmit the ½ bit start pattern. These values are NOT bi-level values, but are the actual logic transmission pattern for the Start Bit(s) in ½ bit intervals. The pattern is transmitted from left to right (MSB is the FBT) the number of whole bits programmed in bits 19-20.

Root TX/PB CSR2: W: 0x080C

Root TX CSR2 word defines additional label/word characteristics for the TX channel, such as MSB/LSB direction, Slew Rate (high or low), bit encoding, bit rate and bits per word.

Bit 4: 1=MSB First: W

This bit is set to a one by the user/API to direct the PE to transmit the label/word from bit 31 (working down to bit zero). **The default and proper setting for most protocols/systems is for this bit to be set to a zero** (which directs the PE to transmit the label/word starting with bit 0, left-most, and to work up to bit 31).

Bit 5: 1= High Slew Rate: W

This bit is set to one by the user/API to direct the PE to have a high slew rate for 100K ARINC-429 baud rates. This bit should be set to a zero for rates below 20K bits per second.

Mode Encoding Bits

Bits 6&7 allow the user to set the encoding mode for bits and change the physical layer (PHY) connections. The user has the selection for normal 429 PHY transmitters or split-+-leg transmission using two channels for Harvard Bi-Phase transmission (the first four channels of each bank can have only their + legs specified to create a 0-5V differential signal).

Values 0x1 and 0x3 are RESERVED values and should not be used.

Bits 6 & 7 = 0x0: Normal ARINC-429 Physical Receiver (NA for Playback)

Bits 6 & 7 = 0x2: 5V Harvard Bi-Phase (717) Only – Two TX Channels

These bits are is set to two by the user/API to direct the PE. This setting will force a TX channel pair (channels 1&2 and 3&4 pairs of each bank) to differential drive the 0-5V + leg of the ARINC-429 drivers (the odd channel + is the positive

differential signal and the even channel + leg is the negative differential signal) to create Harvard Bi-Phase 717 encoding. This setting is provided for older ARINC-717 DFDRS systems. If this mode is set, the ensuring half bit encoding values are ignored. This setting ONLY applies to the first four TX channels of each bank.

Bit Encoding

AltaCore-ARINC can transmit virtually any bit pattern by allowing the user/API to designate the logic "1" and "0" encoding pattern of the bi-level line drivers chips. The Alta ARINC card line drivers (TX chips) have four states (two bit patterns) to drive the +/- legs of the ARINC channel:

- 11 = Tri-State (which is not used as part of the bit encoding scheme)
- 10 = High Voltage
- 01 = Low Voltage
- 00 = Mid Level Voltage (ground)

The user/API will set the values of bits 8-15 with the half bit line driver patterns to designate the encoding of a full bit for a logic 1 and 0. These encoding patterns are used in conjunction with the value in bits 15-24 that designate the ½ bit transmission rate. Thus, the user/API will program the PE with the ½ bit transmission rate, as well as, the ½ bit encoding scheme to allow virtually any NRZL or Bi-Polar/Phase encoding scheme allowed by the ARINC line driver's slew rates and frequency response. ARINC-429 standard encoding is Bi-Polar.

The Tri-State (11) value cannot be used in these encoding values.

Bits 8-11: Logic Zero Encoding: W

These bits direct the PE on how to encode a logic 0 (zero). The user/API sets the half bit patterns that make a logic 0. ARINC-429 should have a 4 bit pattern of 01-00.

Bits 8-9: Zero - 2nd ½ Bit: W

These two bits are set by the user/API with the 2nd ½ bit pattern for a logic zero. ARINC-429 should have a two bit value of "00."

Bits 10-11: Zero - 1st ½ Bit: W

These two bits are set by the user/API with the 1st ½ bit pattern for a logic zero. ARINC-429 should have a two bit value of "01."

Bits 12-15: Logic One Encoding: W

These bits direct the PE on how to encode a logic 1 (one). The user/API sets the half bit patterns that make a logic 1. ARINC-429 should have a 4 bit pattern of 10-00.

Bits 12-13: One - 2nd ½ Bit: W

These two bits are set by the user/API with the 2nd ½ bit pattern for a logic one. ARINC-429 should have a two bit value of "00."

Bits 14-15: One - 1st ½ Bit: W

These two bits are set by the user/API with the 1st ½ bit pattern for a logic one. ARINC-429 should have a two bit value of "10."

Programming TX Bit (Baud) Rate

Bits 16-25: ½ Bit-Time Rate: W

These 10 bits encode the bit rate for the TxP data transmission. The 10-bit value represents the clock tick value for a ½ bit time based on a 1 (one) µsec clock. The minimum setting is a value of one for 2 µsec for a ½ bit (resulting in a 250K bits per second – 200K is specified maximum of the card). The maximum setting of decimal 1024 would provide represent 1024 µsec for a ½ bit resulting in a bit rate of 512 bits per second. This half bit clock tick will drive the bit encoding values as set in bits 8-15 described above.

For ARINC-429 12.5K baud the 10 bit value should be: 40 (40 µsec ½ bits).
For ARINC-429 100K baud the 10 bit value should be: 5 (5 µsec ½ bits).
For ARINC-717 768 baud the 10 bit value should be: 651 (651 µsec ½ bits).

Programming TX Bits Per Word (word length)

Bits 26-31: Bit Count for TX Playback Data Word: W

These bits are set by the user/API with the actual bit count to be transmitted (1-63). This number includes parity bit (if designated). Bit 6 MSB setting will direct bits being shifted out from bit 0 (normal – LSB first) or bit 31 (MSB first). Normal setting for ARINC-429 is 32 (which include parity).

If the value is greater than 32 (for injecting bit count errors for testing), then a zero value will be shifted out for the 33-63 bit count.

Reserved: 0x0810 (One Shot Does not Apply to Playback)

Root API - PB-CB Table Pointer: W: 0x0818

The API allows the user/API to reference label/word lists (TX/PB Control Blocks) by a PB-CB number rather than requiring the user/API to work directly with memory offsets on the board for the PB-CBs. Therefore, the API creates a table in board memory where it stores the memory offset to the PB-CB for each PB-CB number. This register is used to store the pointer to the start of this table. By placing this information in board memory (rather than keeping it in a software variable in the API), this enables the user/API to find the PB-CB when inspecting board memory to troubleshoot problems.

The user/API can look at a memory dump to verify that the data structures are setup correctly and can follow this pointer to find all PB-CB that the API has allocated on the board.

Root API - PB-CB Table Size: W: 0x081C

This register is used with the previous register to define the PB-CB table used by the API to manage PB-CB chains. This register provides the size (in 32-bit words) of the PB-CB table. This allows the API to check for invalid PB-CB numbers (a PB-CB number that exceeds the PB-CB table size is invalid).

Playback Control Blocks (PB-CB)

PB-CBs provide timing and transmission options for a table of 1-N PxPs. A PB-CB is an array of control, status and pointer words that are link-listed to create a transmission chain of 1-N PB-CBs. Each PB-CB references (points to) a 1-65535 PxP Table.

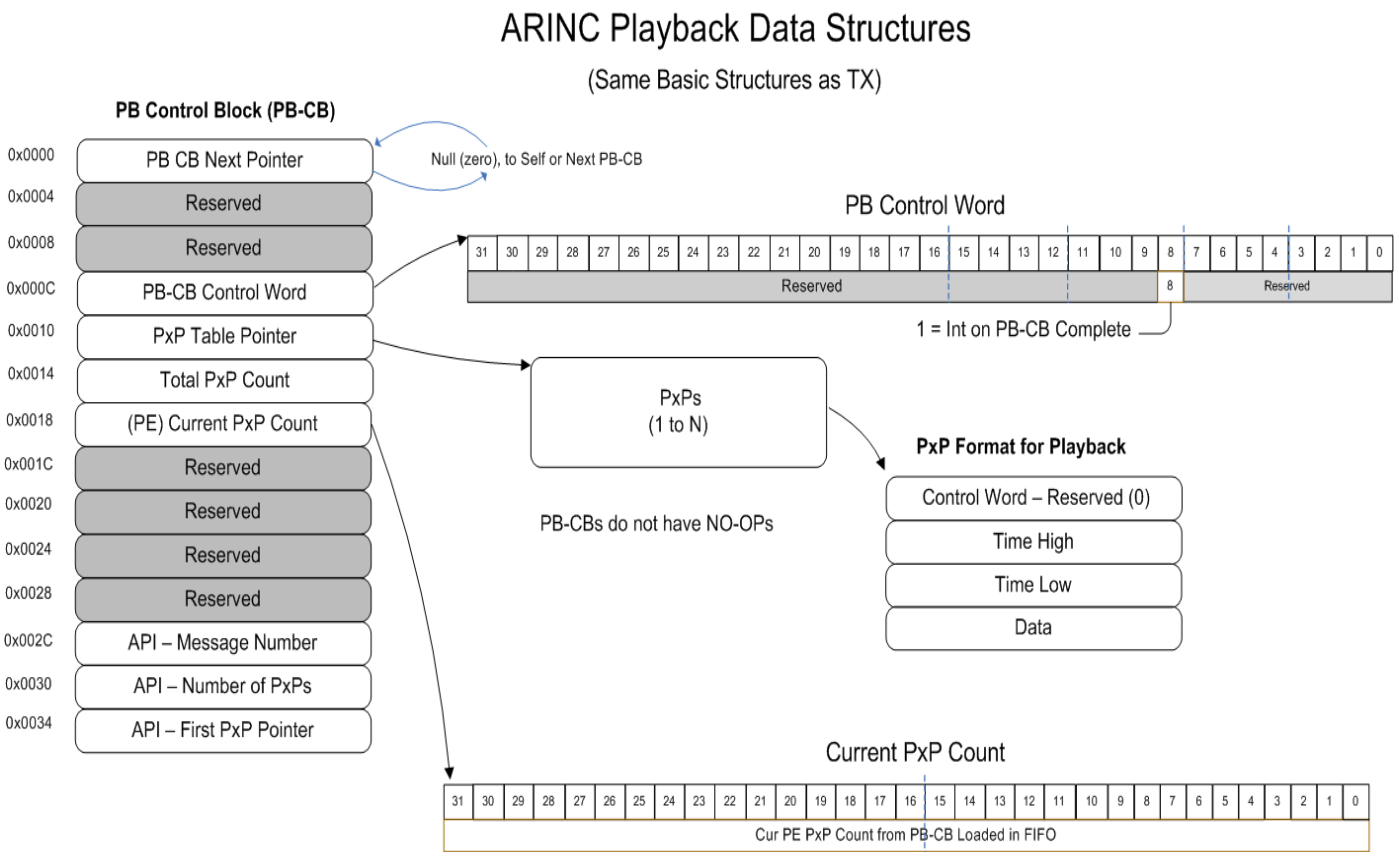


Figure PB-2: Playback Control Block (PB-CB) and PXP Format

The PE immediately starts to execute PXPs of a PB-CB at Root PE Control Word PB start. Each PXP has time of day time stamp that will be used to control transmission start for the PXP Data word. The PE will transmit the PXP Data word when its' 64-bit time is less than or equal to playback clock timer. There are several options for controlling the playback timer and directing the PE to NOT transmit back-up time (PXP time values less than the current playback time value) – Please see the discussion on Root PE Control Word for details on playback clock/PXP transmit control options. These Root PE Control Word settings apply to ALL TX channels.

Next PB-CB Pointer: W: 0x0000

The first word of a PB-CB is a pointer to the next PB-CB. The last PB-CB in the chain should have a Next PB-CB value of Null (zero).

RESERVED: 0x0004-0x0008

Control Word: W: 0x000C

This word is a packed bit data structure that directs execution of the PB-CB. The bits are defined as follows.

Bits 0-7: Reserved

Bit 8: 1=Interrupt on PBCB Complete: W

This bit is set to a one by the user/API to direct the PE to generate an interrupt when the PB-CB is complete.

Bits 9-31: Reserved

PxP Table Pointer: W: 0x0010

This word is set by the user/API with the starting address of the PxP Table for this PB-CB. The TxP Table is a simple array (contiguous area of memory) of 1-N PxPs for transmission. The PB-CB controls the timing/frequency and other user options in executing a PxP Table. The number of PxPs in the Data Table is programmed in word offset 0x000C later in the PB-CB.

PxP Count: W: 0x0014

This word is set by the user/API with the total number (N) of 4 word PxPs pointed to by the PxP Table Pointer at offset 0x0004. The value is a 32-bit unsigned integer or 1-N. (Card/device memory probably the maximum value to well the maximum – see your card configuration for memory details – most cards have 512K-1Mbyte per 16 channel bank, so the user/API must limit the actual value to manage the memory for all channels).

A value of zero will cause the PB-CB not to be executed and transmission logic will move on to the next PB-CB.

(PE) Current PxP Count: 0x0018

This word is set by the PE with the next PxP count (used as a 4 word offset in the PxP Table) to be executed. The PE increments this value when the current PxP is complete. This is a count-up value and when this value equals PxP Count (0x000C), then transmission will move on to the next PB-CB. The PE will reset this value to a zero at the start of PB-CB execution.

RESERVED: 0x001C-0x0028

API – PxP Number: 0x002C

The API uses integer numbers rather than memory pointers to identify PB-CBs. This word is used by the API to store the referenced PB-CB number.

API – Number of PxPs: 0x0030

The API allocates PxP for each PXP Data Table (PB-CB). This word is used by the API to store the number of PxPs allocated for this PXP Data Table.

API – First PxP Pointer: 0x0034

This word provides a pointer to the first PxP for this Data Table. The API uses this pointer to read or write PxP data for the PB Data Table.

Playback Data Table

Each PB-CB has a pointer word that provides the first address offset to a Playback Data Table (simple array) of 1-65535 PxPs. PxPs are detailed in the next paragraph.

Playback Data Packets (PxPs)

PxPs are a 4 word data structure that holds time tag and data value for each ARINC label/word that occurs on a channel. PxPs are very similar to RxPs, but not identical.

> **NOTE:** The AltaAPI only copies the RxP Time High/Low and Data Word to PxPs.

PxP Control/Status1: W: 0x0000

The first word of an RxP is the RxP Control/Status Word that contains RxP control options and status information. This is currently not used for playback and should be set to zero.

PxP Time Stamp Words (High/Low): 0x0004 & 0x0008

The second word and third word of the RxP provide a 64-bit, 20 ηsec time stamp for the label/word. The second word, 0x0004, is the upper 32 bits and the third word, 0x0008, is the lower 32 bits. Clock control is set through Global and Root PE registers.

The PE will delay PXP word transmission until the PXP time is less than or equal to the playback clock. The user/API can control if the playback clock is to reset at PB start and if the clock is suppose to not increment (start counting) until a trigger is latched. Further, the user/API can setup external clock, clock type (RS-485 or TTL) and playback clock increment frequency (1, 5, 10 MHz) in the Root PE Control Word.

PxP Data Word

This word is set by the PE with the data value of the received label/word. The bits per word and bit orientation (MSB/LSB) determine the bit pattern. Most words are LSB first so the first bit with be in position 0 (zero). If MSB first is selected, then the first bit will be in position 31.

> **NOTE:** If you are looping-back a TX channel to an RX channel with MSB first selected and the bits per word (BPW) is less than 32, then the bits will be shifted left (<<) in the RxP word by <<(31-BPW). For example, a 16 bit word will be transmitted starting at bit 15, but will be received in location 31.

<~~~>

AltaCore-ARINC: Signal Generator (SG)

Signal Generator

The ***AltaCore-ARINC*** PE provides unique capability to transmit signal vectors at 100 ηsec intervals to generate custom digital waveform patterns. This allows the user/API to create detailed signal patterns, ideal for error testing. Alta uses this feature ourselves to execute detailed error injection for testing of the ***AltaCore-ARINC.*** This is the most advanced signal generation capability on the market today.

Signal Generation (SG) is used instead of normal transmission. Channel transmission (TX options) cannot be active at the same time as the Signal Generator.

Overview of Signal Generation

The PE shall provide signal generation functions through data structures of bi-level state vectors, which represent channel voltage states of TX line drivers. The vectors are represented as bits in packed words that are linked-listed (called Signal Generator Control Blocks – SGCB) together to provide convenient boundaries for user/API defined intervals or label/word lists. There is a 32-bit time gap word in the link to provide Inter SGCB time gaps. The follow paragraphs detail the Signal Generator data structures.

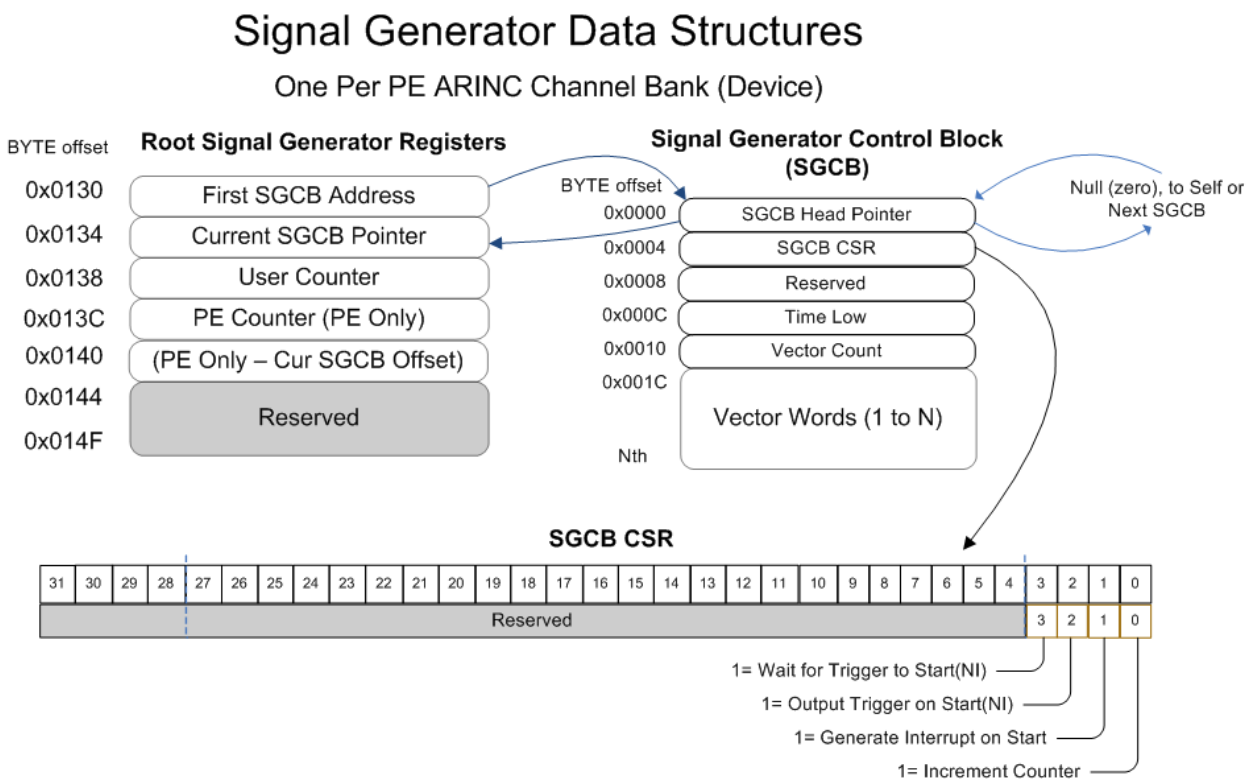


Figure SG-1: Signal Generator Data Structures

Root PE Signal Generator Control Bits

Please review the Root PE Control Word for starting the Signal Generator. Bit 4 of the Root PE Control Word is used to start this function.

> **NOTE:** The PE will check the TX start bit to make sure it is set to a zero before starting Signal Generator. If the TX start bit is set in the Root TX CSR, then Signal Generator will not operate.

> **NOTE:** The user/API can stop Signal Generator by setting a loop count or by setting this bit back to a zero. This bit is checked prior to starting each PXP. The current PXP will completely execute.

2-Bit, Bi-Level Vectors

Bi-Level vectors directly encode the bi-level (2 signal lines) to the ARINC TX line driver; every two bits of a 32-bit word encodes one signal vector (100 ηsec) clock time. The main use for Bi-Level vectors is for detailed testing of ARINC decoders where the user/API may want to control the signal generation to test exacting channel conditions. A two bit vector setting of “00” will encode a ground state (0 voltage). A vector setting of “10” will encode a positive voltage state and a “01” will encode negative voltage state. A vector setting of “11” will encode a tri-state condition (or TX idle time for most Alta ARINC cards).

The user/API can monitor memory usage and Signal Generator execution process and “page” or reload vector packets on the fly. Most applications, however, will load only a few label/word lists at a time, execute them, and then check testing status. The ***AltaAPI*** provides memory management tools to check available memory while building your process, and there are several example programs to help kick start this programming process.

Root Signal Generator Registers

Root First SGCB Address: W: 0x0130

This register is set by the user/API and provides the PE the starting address of the first SGCB. The Signal Generator On bit in the Root PE CSR controls the starting and stopping of Signal Generator function.

Root Current Signal Generator Address: W: 0x0134

This register is set by the PE and provides the user/API a monitor value to indicate the current Signal Generator data structure link of the chain that was loaded for PE execution/sequencing. The user/API should not alter the contents of the current SGCB.

> **NOTE:** The PE will set this register to 0X00000000 when Signal Generator stops.

Root Signal Generator Count: W: 0x0138

This word allows the user/API to set a loop count for the PE to execute the link-list of SGCBs. This word is defined by the user/API (0x00000001-0xFFFFFFFF values) and is the total loop count.

> NOTE: A value of 0x00000000 will direct the PE to IGNORE the Starting SGCB Bit and will cause an infinite loop of SGCB link-list execution. The user/API can stop Signal Generator function by writing a zero to the Signal Generator Off bit in the Root PE CSR.

Root PE Count: W: 0x013C

This word is reserved for the PE for tracking the current count. When the PE count in this location equals the Signal Generator Count in the above register, the PE will write 0x00000000 to the current SGCB Address and will clear Signal Generator Mode ON bit in the Root CSR register.

Root PE Current Vector Location: 0x0140

This word is reserved for the PE for tracking the offset to the current vector word.

Root Reserved 0x0144-0x014F

Signal Generator Control Blocks (SGCB) – Extended Memory

This variable length data structure provides vector state and timing for the PE to generate for Signal Generator/signal construction. This is a linked-list structure with a Next pointer, a SGCB Control and Status Register, one reserved word, one time gap word, a vector count word and a variable list of words that contain the raw vectors.

Next SGCB Pointer: W: 0x0000

This word is set by the user/API to point to the next SGCB. A zero value will cause Signal Generator execution to stop (and a value of 0x00000000 will be written to the Root Signal Generator Current Pointer). The user/API may see up to 50 µsecs of signal idle between SGCBs.

SGCB Control and Status Register: W: 0x0004

This register provides control and status options for SGCB execution.

Bit 0: Increment Counter: W

The user/API sets this bit to designate the first SGCB link of the chain. This bit is only used with Signal Generator count as described in paragraph above. When this bit is set, the PE will increment the PE count by one when the given SGCB is processed.

Bit 1: Generate Interrupt on Start: W

This bit is set to a one by the user/API to direct the PE to generate an interrupt when the SGCB has started transmission.

Bit 2: Output Trigger: W – Not Implemented

This bit is set to a one by the user/API to direct the PE to generate an output trigger when the SGCB has started transmission.

Bit 3: Input Trigger Start: W – Not Implemented

This bit is set to a one by the user/API to direct the PE to stall the transmission of the SGCB until an input trigger has been detected.

Bits 4-15: Reserve

Bits 16-20: Channel Selection Bits: W

These bits are set by the user/API to select the TX channel for the SGCB transmission (0-15 depending on the card configuration).

Bits 21-31: Reserve

Reserved Word: 0x0008

Gap Time: W: 0x000C

This word is a 32-bit, 100-ηsec time value for inserting a zero level gap. A time of zero will cause immediate execution without gaps. The user/API may see up to 50 µsecs of delay (idle time) between SGCBs.

NOTE: To ensure no signal jitter that may results from switching between SGCBs, it is recommended that the user/API pack all vectors between gaps in the same SGCB. Switching between channels will cause up to 50 µsec of jitter (TX delay).

Vector Count: W: 0x0010

This 32 bit word provides the raw 2-bit vector count for states to be transmitted in this SGCB (each 2-bit vector is one count). If Vector count is zero, the given SGCB is treated as a NO-OP and execution jumps to the next SGCB.

NOTE: If Vector Counts do not align with the last word of the list, the PE will execute from the MSB (bit 31) downward to the last count (with the remaining LSB bits not clocked-out/transmitted).

SGCB Vector Bit Words: W: 0x0014-Variable

The following one to N words are the packed vector states. The number of words is dependent on the number of states designated in the SGCB Vector Definition Word (described above).

> **NOTE:** Frequency responses of line drivers may not support very high frequency (non standard ARINC-429) state changes.

<~~~>

AltaCore-ARINC: Receive (RX)

AltaCore-ARINC protocol engine (PE) provides a rich set of functions to decode and buffer ARINC communication labels/words (such as ARINC-429/575 – just about any ARINC-429 Physical Layer, PHY, compatible signal). There are 3 simultaneous methods to receive/buffer data to match almost any data sorting, logging and current value look-up requirement.

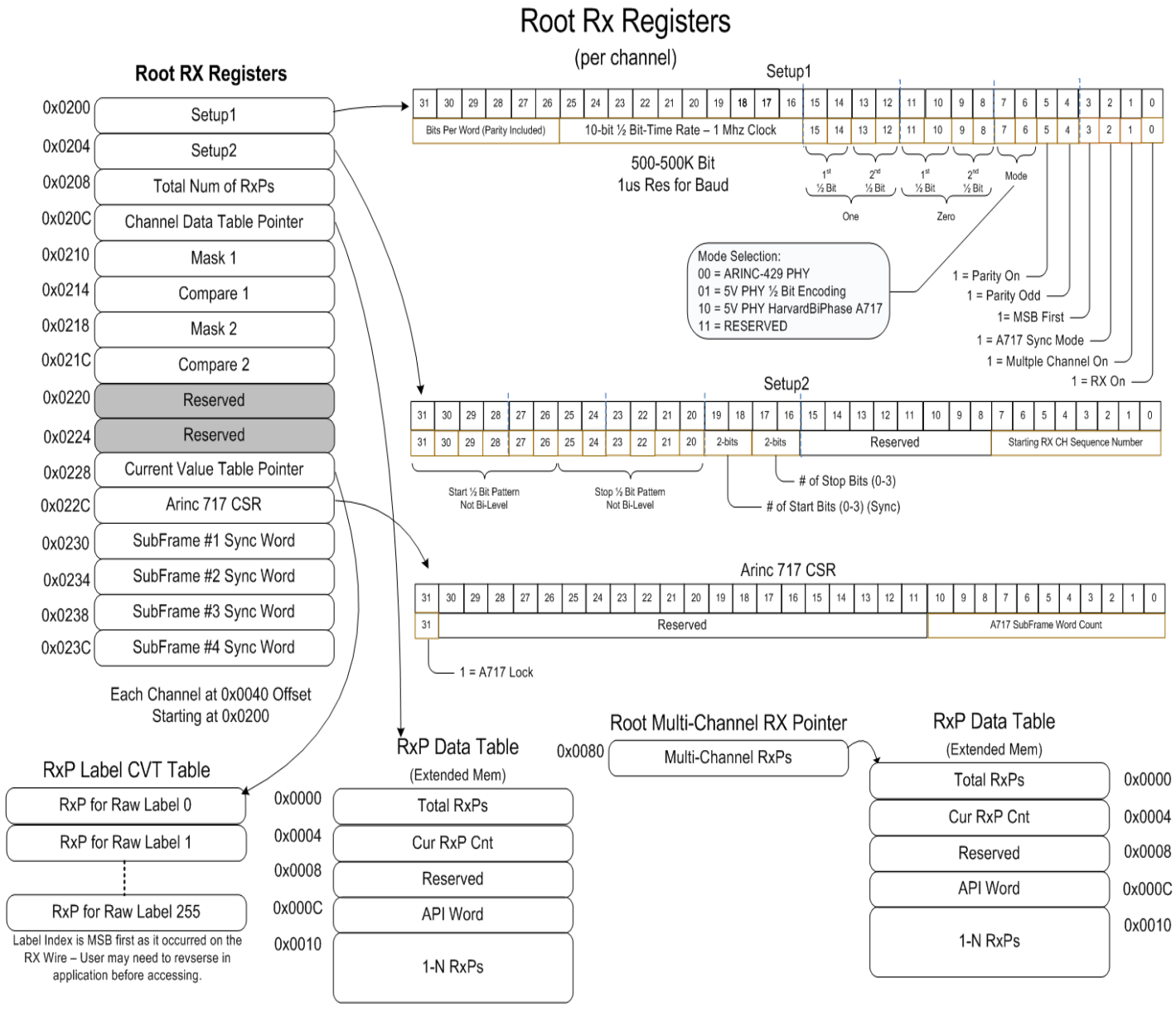


Figure RX-1: Root RX Setup and Pointer Registers

The user/API has several options to set bit encoding, bit (baud) rates, interrupt options, start/sync/stop bits/patterns, parity settings and label/word lengths. Bit (baud) rates and decoding settings are programmable to allow for just about any label/word format of

500-200K bit rates and 8-32 bit words with an ARINC-429 physical layer (PHY) compatible signal.

Labels/words are packed in to a key data structure called a Receive Packet (RxP) that provides control/status bit and timing values for the received labels/words. RxPs can be buffered in three simultaneous schemes: channel level or in a multi-channel RxP Data Tables; or in a raw label-indexed current value table (CVT). RxP Data Tables are a circular buffer programmable in length from 1-65535 RxPs. The CVT simply holds the latest RxP for the word's raw label value (first 8 bits of the 32-bit word). Each RxP has an interrupt and output trigger selection.

In addition to ARINC-429 PHY words, the first two RX channels have special decode selections to bypass the usual ARINC-429 decoder. This allows non 429 PHY words like ARINC-573/717 and discrete triggers events to be decoded and time tagged. When selected, this input is a 5V comparator circuit.

Each channel has RX Root setup and pointer registers that the user/API must setup to define channel decoding and buffering locations. There are also two mask/compare interrupt options per channel so a specific label/word can be trapped and reported to the user application.

All data structures are detailed in the following paragraphs.

Root RX Setup Registers: W: 0x0200-0x05FC

Each RX channel has 16 Root Setup Registers that are programmed by the user/API to define the channel's decoding parameters. These words are detailed in the following paragraphs.

Root Setup1: W: 0x0200 (each channel is 0x0040 Offset to 0x0200)

The Setup1 register is programmed by the user/API with channel specific decoding parameters. The channel settings are detailed in the following figure and paragraphs.

Bit 0: RX On: W

This bit is set to a one by the user/API to direct PE to receive data from this channel. The decode parameters are set in the following bits and withSetup2 register.

Bit 1: MC On (Multi-Channel RX): W

This bit is set to one by the user/API to direct the PE to copy RX label/word packets to the Multi-Channel (MC) RXP Data Table for the respective channel.

Bit 2: A717 Mode On: W

This bit is set to one by the user/API when the user wants to receive ARINC-717 framed data. When this bit is set, the user must also program the four frame sync words (see description later in this section). The user/API must also have enough RXPs designated in the channel level, and if selected, the MC RXP Data Tables. This setting ONLY applies to the first two RX channels of each bank.

Bit 3: 1=MSB First: W

This bit is set to a one by the user/API to direct the PE to transmit the label/word from bit 31 (working down to bit zero). **The default and proper setting for most protocols/systems is for this bit to be set to a zero,** which directs the PE to transmit the label/word starting with bit 0 and to work up to bit 31.

Bit 4: 1 = Parity On: W

This bit is set to a one by the user/API to direct the PE to add a parity bit to the end of the word (prior to stop bits). The value of parity, odd or even, is determined by the setting of the next bit. ARINC-429 default is one for this bit. This bit could be set to zero to inject a “no parity” error condition for ARINC-429 labels.

Bit 5: 1=Parity Odd: W

This bit is set to a one by the user/API to direct the PE have an odd parity value. A value of zero would force an even parity value. The default for ARINC-429 is odd parity (this bit set to a one). This bit can be set to a zero to inject a parity error for ARINC-429 labels.

Mode Decoding Selection

The next two bits allow the user to designate the decoding method using standard 429 physical receivers or basic 5V comparator. These bits should be set to zero (429 only) for channels 3+. Only channels 1 & 2 can have 5V, non-429 selection. Do not set these bits to the value of 0x3.

Bits 6 & 7 = 0x0: Normal ARINC-429 Physical Receiver

These bits should be set to zero by the user/API for normal ARINC-429 receiving. The ½ bit decoding bits (8-11) should set to Bi-Polar values (0x84).

Bits 6 & 7 = 0x1: 5V Comparator Receiver

These bits should be set to a value of 0x1 by the user/API for by-passing the ARINC-429 receiver and using simple 5V comparators. This is ideal for time

tagging trigger events or other raw bit encodings. Bits 8-11 would be programmed for the user specifications. For example, if the user was time tagging a trigger going high-to-low, then bits 8-11 would be 0xF0. This setting ONLY applies to the first two RX channels of each bank.

Bits 6 & 7 = 0x2: 5V Comparator Receiver – Harvard Bi-Phase (717) Only

These bits are is set to two by the user/API to direct the PE to by-pass the ARINC-429 receivers and receive Harvard Bi-Phase bits as used with older 717 systems. Harvard Bi-Phase encoding, which requires two transmit channels for encoding the positive and negative legs of the bi-phase encoding (see Root PE Registers for channel pairing details). This setting is provided for older ARINC-717 DFDRS systems. If this mode is set, the ensuring half bit encoding values are ignored. This setting ONLY applies to the first two RX channels of each bank.

Bit Encoding

AltaCore-ARINC can receive virtually any bit pattern by allowing the user/API to designate the logic "1" and "0" encoding pattern of the bi-level line receiver circuit. The Alta ARINC card line drivers (TX chips) have four states (two bit patterns) to drive the +/- legs of the ARINC channel:

- 11 = Mid Level Voltage (ground)
- 10 = High Voltage
- 01 = Low Voltage
- 00 = Mid Level Voltage (ground)

The user/API will set the values of bits 8-15 with the half bit receiver bi-level patterns to designate the decoding of a full bit for a logic 1 and 0. These decoding patterns are used in conjunction with the value in bits 15-24 that designate the ½ bit reception rate. Thus, the user/API will program the PE with the ½ bit reception rate, the ½ bit decoding and mode selection scheme to allow virtually any NRZL or Bi-Polar/Bi-Phase encoding scheme allowed by the ARINC line driver's slew rates and frequency response (most Bi-Phase or differtial decodings with +5/GND can be only be done on channels 1&2 by-passing the ARINC receivers – see previous paragraphs). ARINC-429 standard decoding is Bi-Polar.

Bits 8-11: Logic Zero Encoding: W

These bits direct the PE on how to decode a logic 0 (zero). The user/API sets the half bit patterns that make a logic 0. ARINC-429 should have a 4 bit pattern of 01-00.

Bits 8-9: Zero - 2nd ½ Bit: W

These two bits are set by the user/API with the 2nd ½ bit pattern for a logic zero. ARINC-429 should have a two bit value of "00."

Bits 10-11: Zero - 1st ½ Bit: W

These two bits are set by the user/API with the 1st ½ bit pattern for a logic zero. ARINC-429 should have a two bit value of "01."

Bits 12-15: Logic One Encoding: W

These bits direct the PE on how to decode a logic 1 (one). The user/API sets the half bit patterns that make a logic 1. ARINC-429 should have a 4 bit pattern of 10-00.

Bits 12-13: One - 2nd ½ Bit: W

These two bits are set by the user/API with the 2nd ½ bit pattern for a logic one. ARINC-429 should have a two bit value of "00."

Bits 14-15: One - 1st ½ Bit: W

These two bits are set by the user/API with the 1st ½ bit pattern for a logic one. ARINC-429 should have a two bit value of "10."

RX Bit (Baud) Rate

Bits 15-24: Bit Rate (½ bit time): W

These 10 bits encode the bit rate for the RxP data reception. The 10-bit value represents the clock tick value for a ½ bit time based on a 1 (one) µsec clock. The minimum setting is a value of two, which is a 2 µsec clock tick for a ½ bit (resulting in a 250K bits per second – 200K is the spec maximum of the card). The maximum setting of decimal 1024 would provide represent 1024 µsec for a ½ bit resulting in a bit rate of 512 bits per second. This half bit clock tick will drive the bit decoding values as set in bits 8-15 described above.

For ARINC-429 12.5K bit rate, the 10 bit value should be: 40 (40 µsec ½ bits).
For ARINC-429 100K bit rate, the 10 bit value should be: 5 (5 µsec ½ bits).
For ARINC-717 768 bit rate, the 10 bit value should be: 651 (651 µsec ½ bits).

Bits 26-31: Bits Per Word (Including Parity): W

This field is set by the PE with the actual bits per word to be decoded, including parity. The MSB setting will determine where bits are placed the RxP Data Word: if MSB is turned on, then bits will start being received in position 31. With MSB turned off, which is standard for most ARINC settings, then bits will be received in starting location 0 (zero).

ARINC-429 should have this value set to 31 (0x1F – 6 MSB bits). Parity should also be turned on with odd parity selected.

ARINC-717 should have this value set to 12 (0x0C – 6 MSB bits). Parity should also be turned on with odd parity selected.

Root Setup2: W: 0x0004

The RX CH Setup2 register is required to be setup to set decoding parameters. This word's settings are detailed in the following paragraphs.

Bits 0-7: Starting Sequence Number: W

This field is starting count for a sequence number that will be included in each RxP (This feature provides a level of safety to the host system to make sure label/words have not been dropped). The PE will place the sequence number in the RxP and then increment this value for the next RxP. It is recommended the user/API set this value to zero at initialization.

Bits 8-15: Reserved.

Bits 16-17: Number of Stop Bits: W – NI in the version.

These bits are set by the user/API with the number of whole stop bits for label/word. ARINC-429 and 717 would set this value to a zero. Each label/word can have 0-3 stop bits, and the ½ bit logic pattern is set in corresponding bits 20-25. This feature is mainly provided for future RS-232/422/485 style communications and for rare ARINC encodings.

Bits 18-19: Number of Start Bits: W – NI in the version.

These bits are set by the user/API with the number of whole start bits for label/word. ARINC-429 and 717 would set this value to a zero. Each label/word can have 0-3 start bits, and the ½ bit logic pattern is set in corresponding bits 26-31. This feature is mainly provided for future RS-232/422/485 style communications and for rare ARINC encodings. If there are start bits designate, these could also be used as a sync pattern (for example, a 1553 encoding would have a value of 3).

Bits 20-25: Stop Bits - ½ Bit Pattern: W – NI in the version.

These bits are set by the user/API to direct the PE to receive a ½ bit stop pattern. These values are NOT bi-level values, but are the actual logic receive pattern for the Stop Bit(s) in ½ bit intervals. The pattern is referenced from left to right (MSB is FBT) the number of whole bits programmed in bits 16-17.

Bits 26-31: Start Bits - ½ Bit Pattern: W – NI in the version.

These bits are set by the user/API to direct the PE to receive a ½ bit start pattern. These values are NOT bi-level values, but are the actual logic receive pattern for the Start Bit(s) in ½ bit intervals. The pattern is referenced from left to right (MSB is FBT) the number of whole bits programmed in bits 19-20. For example, a 1553 Command or Status sync would have a bit pattern of 111000.

Root Total Num of RxPs: W: 0x0008

This word is a simple counter for the total number of RxP (Receive Packets). The user/API can write a starting value (usually zero) for this word and the PE will simple increment the value when an RxP has been received.

Root Channel Data Table Pointer: W: 0x000C

This is set by the user/API with the address for the channel's RX Data Table. The Data Table is where RxPs will be buffered for user/API access. RxP Data Table structures are defined later in this section.

Root Channel Mask1: W: 0x00010

This word is set by the user/API with a logical AND mask that is applied to each received label/word. A value of zero will direct the PE to not perform a compare function. When a mask is applied, the PE compares the result to the next register word, Compare, for a match. If there is a match, the PE will generate an interrupt queue event (See Interrupt Section of manual) and an external trigger.

Root Channel Compare1: W: 0x0014

This word is set by the user/API with the match value after the logical AND is applied from the Mask1 register (see previous paragraph). If the label/word value matches this value after the AND function, then an interrupt queue event (See Interrupt Section of manual).

Root Channel Mask2: W: 0x0018

This word is set by the user/API with a logical AND mask that is applied to each received label/word. A value of zero will direct the PE to not perform a compare function. When a mask is applied, the PE compares the result to the next register word, Compare, for a match. If there is a match, the PE will generate an interrupt queue event (See Interrupt Section of manual) and an external trigger.

Root Channel Compare2: W: 0x001C

This word is set by the user/API with the match value after the logical AND is applied from the Mask1 register (see previous paragraph). If the label/word value matches this value after the AND function, then an interrupt queue event (See Interrupt Section of manual).

RESERVED - 0x0020-0024

Root Current Value Table (CVT) Pointer: W: 0x0028

This pointer is set by the user/API to direct the API to store an RXP to a raw label indexed 256 table element position. The table is a simple array of RXPs indexed by the raw, MSB oriented (MSB right most bit) label value. The user application probably needs to reverse the label value to LSB right most bit for index offset look-up. This receive buffer feature is ideal for application that simply need to look-up the label value at a periodic rate. The user's application may want to make two successive reads in a row and compare the time values to make sure the read did not occur during an RXP update (make sure the time values are the same to ensure the RXP update is accurate).

Root A717 CSR: W: 0x002C

This word provides control and status values for ARINC-717 operation. The register is only available for the first two RX channels of a bank (please see your hardware manual for channel configuration of your product). The following fields apply:

Bits 0-10: SubFrame Word Count: W

These bits are set by the user/API to direct the PE with the number of words between the four word sync patter (set by the user/API in the subsequent four words). Valid values are 8-2048 decimal.

Bits 11-30: Reserved

Bit 31: A717 Lock

This bit is set by the PE to inform the user/API that the PE has either found a valid four word sync pattern (Lock) or if the PE is in a search condition. A value of one represents Lock and zero represents a search condition.

ARINC-717 SubFrame Sync Words

The next four words only apply to the first two RX channels of a device bank. Please see your hardware manual for channel configuration of your product. Other channels of the device bank should consider these words Reserved.

Root SubFrame Sync Word1: W: 0x0030

This word is set by the user with the first word subframe sync pattern for ARINC-717 subframes. Bits 20-31 are programmed by the user/API.

Root SubFrame Sync Word2: W: 0x0034

This word is set by the user with the second word subframe sync pattern for ARINC-717 subframes. Bits 20-31 are programmed by the user/API.

Root SubFrame Sync Word3: W: 0x0038

This word is set by the user with the third word subframe sync pattern for ARINC-717 subframes. Bits 20-31 are programmed by the user/API.

Root SubFrame Sync Word4: W: 0x003C

This word is set by the user with the fourth word subframe sync pattern for ARINC-717 subframes. Bits 20-31 are programmed by the user/API.

Root Multi-Channel RX Pointer: W: 0x0080

This register is a pointer that is set by the user/API to address a single RxP Data Table. This function provides buffering of label/words (RxPs) from all selected channels into a single RxP Data Table. This feature is ideal for combined monitoring or recording of multiple channels to a single contiguous area of memory. RxP Data Tables are detailed later in this section. The user/API selects an individual RxP to be copied to the Multi Channel RxP Data Table by setting Bit 1 of the RxP Control/Status Word (so the user/API may would to set all RxPs for a given channel to have all channel data copied to this RxP Data Table).

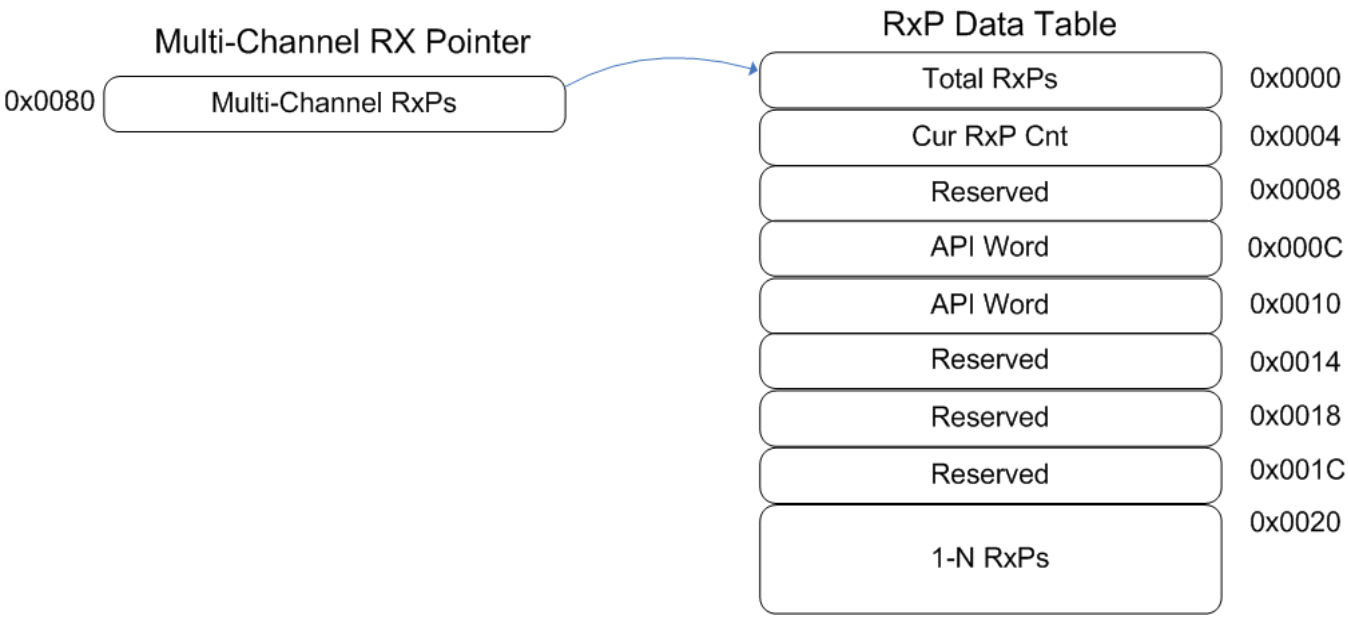


Figure RX-2: Root Multi Channel RX Pointer – RxP Data Table

RxP Data Tables: W

The RxP Data Table structure provides a circular buffer for 1-N RxPs. The RxP Data Table begins with an eight word header with words the user/API programs to define the Total RxP Count and the PE sets with the Current RxP Count. The other six words are used for API tracking and reserved words for future use. After the eight word header

are the actual 4 word Receive Packets (RxPs). All of these words and RxPs are detailed in the following paragraphs.

Total RxP Count: W: 0x0000

This word is set by the user/API with an unsigned 32-bit value to designate the total number of RxPs in the RxP Data Table. A value of zero will cause the RxP to not be copied to this table (so the value must be >0). The value will be much less than the full 32-bit range as card/device memory is much, much less than 2^{32} RxPs. The user/API must set this word to an appropriate value for RxP buffering and still manage memory usage.

Current RxP Count: 0x0004

This value is set by the PE to track the current location of the next RxP to be received. The PE will count up when an RxP has been copied to the table and will reset to a zero when the Count value matches Total RxP Count (0x0000). The user/API should not write to this location.

RxP: Receive Packet Data Structure

RxPs are a 4 word data structure that holds control/status, sequence number, channel number, time tag, data value (and API reference word) for each ARINC label/word that occurs on a channel.

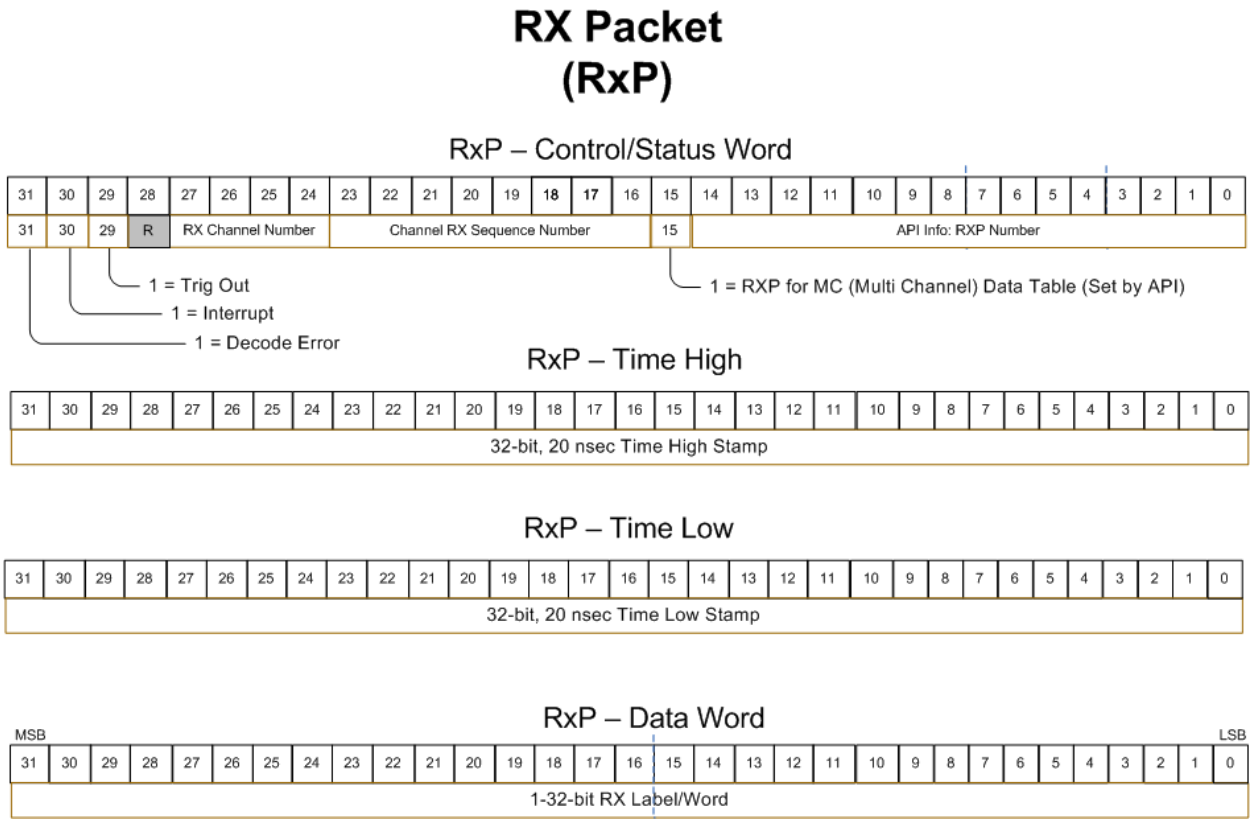


Figure RX-3: Receive Packet (RxP)

RxP Control/Status1: W: 0x0000

The first word of an RxP is the RxP Control/Status Word that contains RxP control options and status information. The following paragraphs detail this word.

Bits 0-15: API Info RXP Number

These 16 LSB are set to one by the user/API to signify the RXP number in the RX and MC RXP data table. Bit 15 is set ONLY for MC data tables.

Bits 16-23: API Info RXP Number

These 8 bits are set by the PE with an incrementing value to provide a sequence number (rolling serial number). This is useful when reading an RXP data table to see if label/words were dropped.

Bits 24-27: RX Channel Number

These 4 bits are set by the PE with the channel number RX for the RXP.

BIT 28: Reserved

Bit 29: Trigger Out: W

This bit is set to one by the user/API to direct the PE to generate an external trigger when this RXP has been received/completed. See the Global and Root PE section, and your hardware manual for more details on external interrupts

Bit 30: Interrupt: W

This bit is set to one by the user/API to direct the PE to generate an interrupt event in the interrupt queue and hardware interrupt if enabled. See the Root PE and Interrupt Section for more details on servicing interrupts.

Bit 31: Decode Error

This bit is set to one by the PE to tell the user that the label/word had a decode error as set in the Root RX Setup registers. Often, this signifies a parity or other decoding error.

RxP Time Stamp Words: 0x0004 & 0x0008

The second word and third word of the RxP provide a 64-bit, 1 µsec time stamp for the label/word. The second word, 0x0004, is the upper 32 bits and the third word, 0x0008, is the lower 32 bits. Clock control is set through Global and Root PE registers.

RxP Data Word

This word is set by the PE with the data value of the received label/word. The bits per word and bit orientation (MSB/LSB) determine the bit pattern. Most words are LSB first so the first bit with be in position 0 (zero). If MSB first is selected, then the first bit will be in position 31.

NOTE: If you are looping-back a TX channel to an RX channel with MSB first selected and the bits per word (BPW) is less than 32, then the bits will be shifted left (<<) in the RxP word by <<(31-BPW). For example, a 16 bit word will be transmitted starting at bit 15, but will be received in location 31.

ENET APMP UDP Format

The ENET-A429 product offers a broadcast UDP format for Multi-Channel RX channels that will auto bridge ARINC RX data to a broadcast UDP packet. **Only channels selected for Multi-Channel (See RX Root Setup1 register) will have their RXPs included in the UDP broadcast.** This mode is called Alta Passive Monitor Protocol (**APMP**).

The payload section (after Alta Header) is made up of 4 words of time reference followed by RXP from the different RX channels. **The PE will send the APMP packet every 200 msec or every 85 RXPs, whichever comes first.** This can be an ideal function to implement for data recorders or collectors on the local area network (LAN) system. Please see the AltaAPI manual and example programs for implementation details.

The ARINC APMP UDP Packet is shown in the following figure.

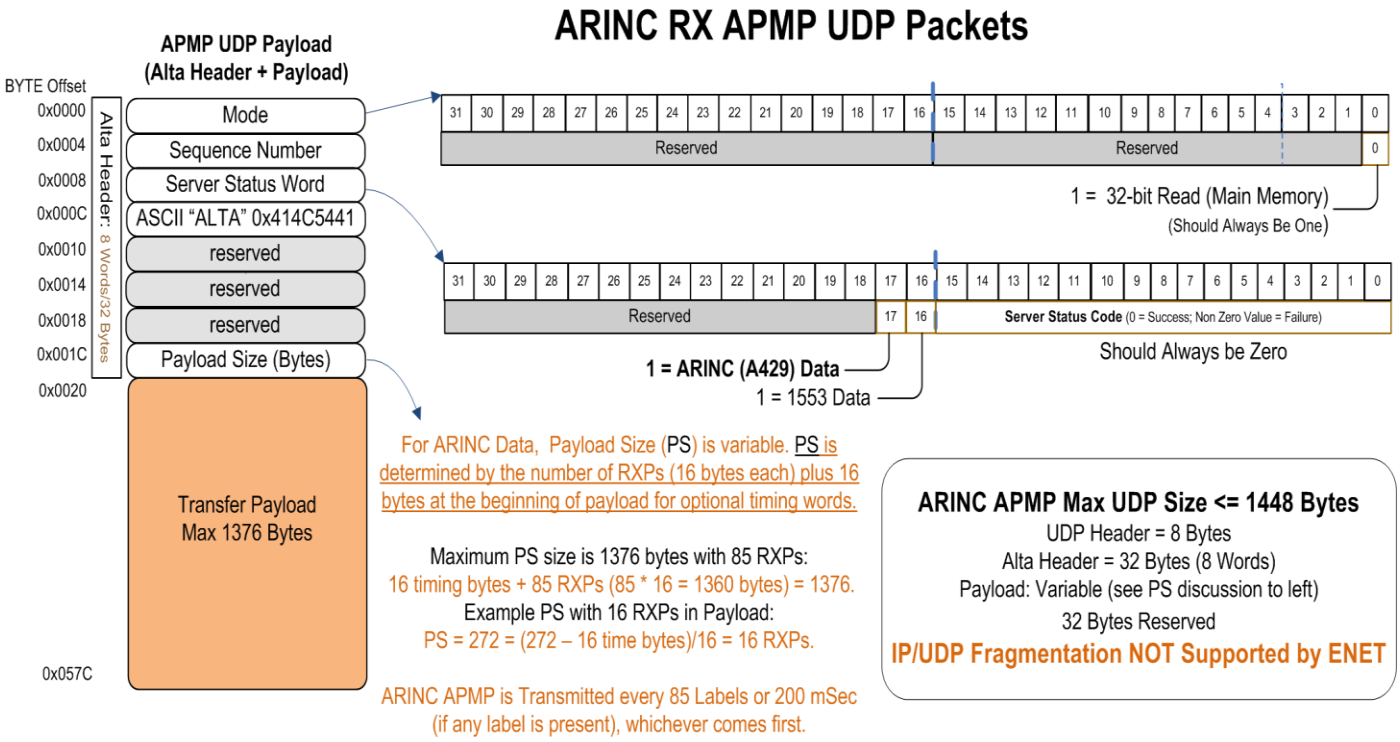


Figure RX-4: APMP UDP Packet Format

The user can select in the Root PE Control register to have reference PE and IRIG/Interval Timer time words injected as the first 4 words of the Payload at bye offset 0x0020 to 0x002C (this would provide relative and absolute time references to apply to the individual RXP times). The time formats are the SAME as if IRIG time is read from the Root PE Control register (for PE + IRIG Time) or the Read Timer in the Root Interval Timer Control Register. Please see the following bookmarks:

Root IRIG Time High & Low: 0x0024/28

Root Interval Timer: 0x004C

<~~~>

Interrupt Functions

The PE provides a linked-list interrupt queue and status registers for logging interrupt events set by the user/API in the various TX, RX, PB and SG data structures. These flag events are trapped in an interrupt queue linked-list data structure (one for each device/channel bank) to store their occurrence. The user/API will either poll or receive a hardware interrupt to signal the reading of this queue to retrieve information as to what event was capture.

The following paragraphs and Figure Int-1 detail these interrupt queue data structures.

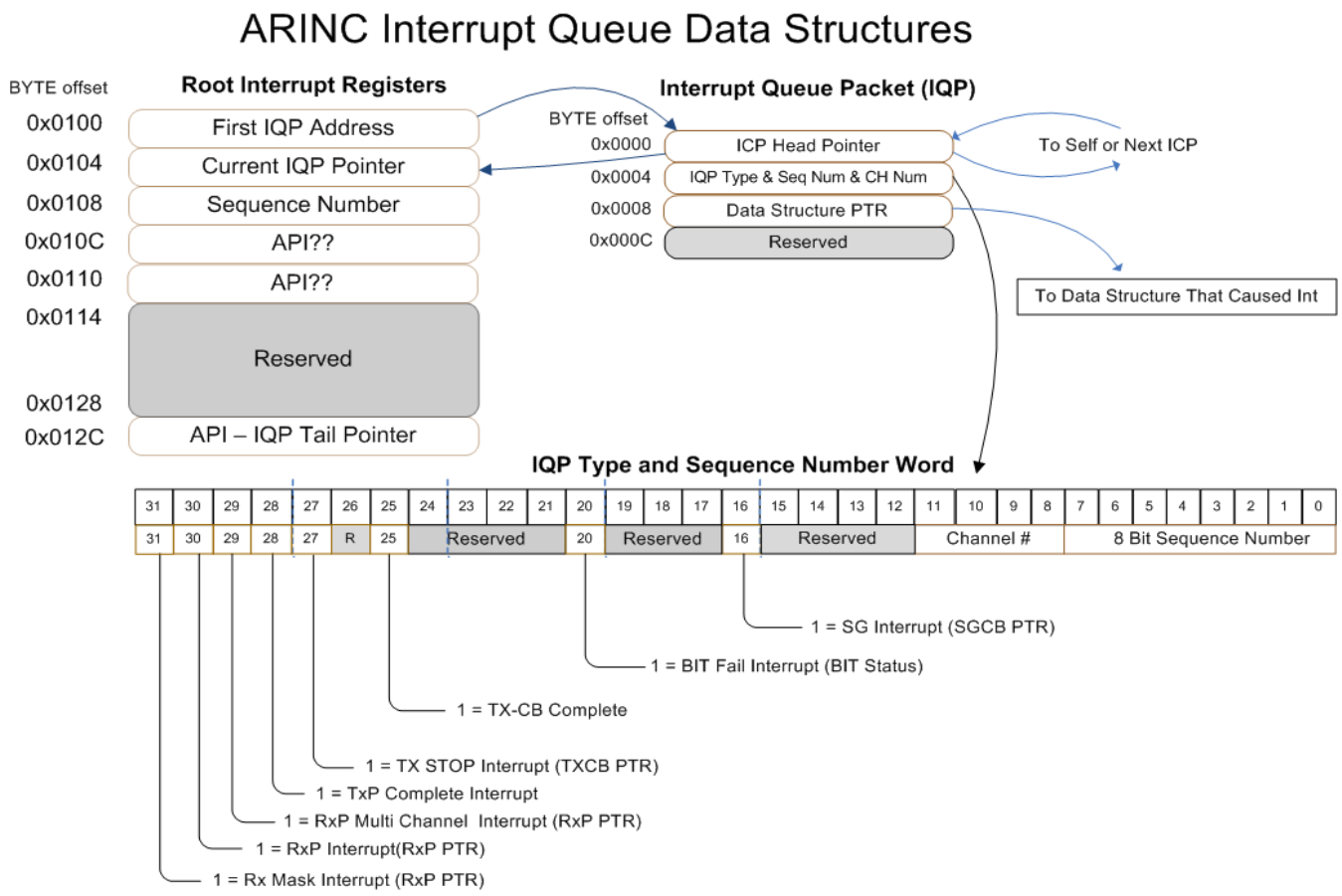


Figure Int-1: Interrupt Data Structures

Interrupt events are serviced by the user/API's application in two methods: a hardware interrupt signal to the system with an OS Interrupt Service Routine (ISR in a driver or user/API level setup) or software polling through some type of host application process timer (or aperiodically by the application). For either method, the user/API should read about the interrupt control/pending/latch bits in the Global Registers and Root PE Registers Sections. These bits control hardware signaling/clearing of the interrupt line

to the host backplane/system and provide pending information for channel/event identification.

Root Starting Interrupt Queue Packet (IQP) Address: W: 0x0100

The user/API sets this value with the address of the first IQP link of the chain. If this value is zero, then the PE will not process interrupts.

> **NOTE**: The user/API can stop interrupt processing by setting this value to a zero. Interrupt conditions will not be processed if this value is zero. The user/API should initialize all interrupt structures prior to setting this value (turning on) to the first IQP address to start/re-start interrupt processing.

Root Current IQP Address: 0x0104

The user/API can read this value to see the current location of the active IQP. The user/API should not process this IQP, but the previous unread IQP in the link-list.

Root Interrupt Sequence Number: W: 0x0108

This register is set by the user/API with the starting sequence count, which is probably zero. The PE increment this register and copy the 8 LSBs to the IQP & Sequence Number Word in an IQP.

API – Reserved: 0x010C

API – Reserved: 0x0110

The user/API should not write to these registers. They are reserved for PE interrupt temporary processing.

Reserved: 0x0114-0x0128

API – IQP Tail Pointer: 0x012C

The API uses this word to store the Interrupt Queue “tail” pointer, which points to the next Interrupt Queue entry to be processed by the API. The API reads the Current IQP Address (0x0204) to get the “head” pointer. The API then reads Interrupt Queue entries until the tail pointer equals the head pointer.

> **NOTE:** Interrupts will be posted to this queue structure within 40 µsecs of the event.

Interrupt Queue Packets (IQP) – Extended Memory

PE process events that the user/API set for detection various are queued for user/API access. The queue is a link-list of packed word data structures (Interrupt Queue Packets) that provides status to what event caused the interrupt, a running counter of interrupt events (sequence number), the address (pointer) of the data structure that

triggered the event and two reserved words. The IQP is detailed in the following paragraphs.

IQP Head Pointer: W: 0x0000

The first word of the IQP data structure is the address of the next IQP of the chain. The user/API determines how many lQP links are required to allow interrupts events to synchronize user/API application requirements. Since this is a circular chain, then length should be at least 2 links and it is recommended the user/API default to 100 links.

Interrupt Type (16 Bits) | Interrupt Sequence Number (8 Bits) | Channel # (Number - 8 Bits) : 0x0004

This register is a packed data structure where the MSB 16 bits are the Interrupt Code derived from the respective PE function (TX, RX, PB, SG) that caused the interrupt (this allows the user/API application to prepare the appropriate data structure). The bit codes are detailed in the following paragraphs.

The user/API should initialization this word to 0XFFFFFFFF.

Bits 0-7: Sequence Number

These bits are set by the PE with the lower 8 bits of the 32-bit Root Interrupt Sequence Number 0x0308. This allows the user/API to watch for possible mis-sequence/dropped interrupt events.

Bits 8-15: Channel # (Number)

These bits are set by the PE with channel number that caused the interrupt event.

Bit 16: SG Interrupt (SGCB PTR)

This bit is set by the PE to signify that the Data Structure Pointer is form a Signal Generator Control Block.

Bits 17-19: Reserved

Bit 20: BIT Fail Interrupt (BIT Status)

This bit is set by the PE to signify that the Data Structure Pointer is for the BIT Status Word.

Bits 21-24: Reserved

Bit 25: TX TxP Complete Interrupt (TX-CB Pointer)

This bit is set by the PE to signify that the Data Structure Pointer is for a TX Control Block that had a TxP complete (bit 8 of TX-CB CSR).

Bit 26: Reserved

Bit 27: TX Stop Interrupt (TX-CB Pointer)
This bit is set by the PE to signify that the Data Structure Pointer is for the last TX-CB executed.

Bit 28: TxP Complete Interrupt (TxP Pointer)
This bit is set by the PE to signify that the Data Structure Pointer is for the TxP that caused the interrupt. The Data Structure pointer value is zero for Aperiodic TXPs.

Bit 29: RxP Multi Channel Interrupt (RxP Pointer)
This bit is set by the PE to signify that the Data Structure Pointer is for the RxP of the Multi Channel RxP Data Table that caused the interrupt.

Bit 30: RxP CH Interrupt (RxP Pointer)
This bit is set by the PE to signify that the Data Structure Pointer is for the RxP that caused the interrupt.

Bit 31: RxP CH Mask Interrupt (RxP Pointer)
This bit is set by the PE to signify that the Data Structure Pointer is for the RxP that caused the interrupt with a Mask/Compare from the RX Root Channel Definition Registers (one of two).

Data Structure Pointer: W: 0x0008

This word is set by the PE with the address of the data structure that caused the interrupt event. The bits described above provide the coding of this word.

Reserved Word: 0x000C

<~~~>

Revision Information

Date	Rev	Description
10/14/08	C	Updated for New Playback Function Corrected Several Register Offset References to Figures
12/07/08	D	Updated for New Playback Time Controls, Absolute Timing Discussion.
7/16/09	D1	1. Changed Address 2. Fixed TX-CB Figure (Next TX-CB Pointer cannot point to self)
11/10/09	D2	Corrected SG Vector Time from 20 nSec to 1 Usec
2/5/10	D3	Added Interval Timer to PE Root Area – New Feature – PE Rev 0301
3/1/10	D4	Corrected SG Vector Time from 1 uSec back to 100 nSec. Changed NAICS Code to 334119 on front cover.
5/21/10	D5	Added TX-CB Control Word Bit Description: Bit 4: 1=Stop TX after TxP Complete.
11/28/10	D6	Redefined One-Shot/Aperiodic TX Channel Root Register to reflect the change from a single TXP to a single TX-CB with a list of 1-N TXPs. This manual version was not released to customers.
2/18/11	D7	Corrected Global Discretes Offset Values in Text and Figure. Added IRIG Clarification that requires +1 Second User/API adjustment and listed IRIG Decode Formats Supported.
3/14/2012	D8	Corrected Typo in TxP Table Pointer reference to TxP Number offset 0x0014.
9/10/2012	D9	Corrected RX BPW Encoding and 717 Frame Sync Words Typos. 717 Baud Rate Math Error Corrected. Added RX APMP Description.
2/6/13	E0	Changed NAICS Code to 334118 on Cover
6/24/13	E1	Minor Spelling Corrections and minor RX APMP text & figure change.
10/25/13	E2	Corrected Root TX Control Regs Drawing & ENET APMP Typo
11/11/13	E3	Added IEEE-1588 PTP Paragraphs to Global and Root PE
3/1/14	E4	Clarified Signal Capture FIFO is 512 Words