User’s Manual

RAR-USB

Abaco SYSTEMS | AMETEK

Copyrights

RAR-USB User's Manual (1500-105)

Software Revision: 1.71
Document Revision: 1.71
Document Date: 15 April 2024

Abaco Systems, Inc.
8800 Redstone Gateway SW
Huntsville, AL 35808
Main +1 866-652-2226

abaco.support@ametek.com (email)

https://www.abaco.com/products/avionics

Additional Resources

For more information, please visit the Abaco Systems website at:

www.abaco.com

Contents and Tables

Contents

Figures

Tables

CHAPTER 1

The RAR-USB

Overview

The RAR-USB is a multiple-channel USB-to-ARINC interface design available in several channel configurations supporting ARINC 429, ARINC 573/717, Avionics Discrete I/O, and IRIG time synchronization.

Features

The RAR-USB features available include:

- 64-bit 1 microsecond on-board timer
- 64-bit 1 microsecond ARINC 429 receive message time-stamps
- 1 millisecond periodic ARINC 429 message scheduler resolution
- ARINC 429 transmit message error injection
- ARINC 429 receive message error detection
- Software programmable ARINC 429 transmit and receive bit rate
- ARINC 429 receiver label filtering
- Merged mode ARINC 429 receive message buffering
- Fixed ARINC 429 Transmit Levels
- Fixed ARINC 429 Receive Threshold Levels
- IRIG-B reception supporting AM or DC/TTL input
- IRIG-B generator supporting DC/TTL output

Operating Systems Supported

The software distribution for the RAR-USB is AR-STREAM-SW, supporting the following operating systems. Installation instructions are

provided in this manual for 32-bit and 64-bit Windows operating systems, while instructions for 32-bit and 64-bit Linux are included both in this manual and in the Linux distribution file.

- Windows XP 32-bit
- Windows 7 32-bit and 64-bit
- Server 2008R2 64-bit
- Windows 8 32-bit and 64-bit
- Windows 8.1 32-bit and 64-bit
- Windows 10 32-bit and 64-bit
- Windows 11 64-bit
- Server 2012R1+R2 64-bit
- Linux Kernel 3.16.x 32-bit and 64-bit
- Linux Kernel 4.0.5 32-bit and 64-bit

Specifications

Figure 1. The RAR-USB

USB Interface

- USB 2.0 Compliant

Transmit Channels

- Up to five independent differential serial transmit channels
- Automatic ARINC 429 parity generation
- 4096 message transmit buffer for each ARINC 429 channel
- Baud rate/slew rate software-programmable for each channel
- 1024 entry message table supporting scheduled message transmission for all ARINC 429 channels
- 52K entry transmit playback buffer supporting time-synchronized transmission for all ARINC 429 channels
- 32K Entry ARINC 717 Transmit Buffer

Receiver Channels

- Up to sixteen independent, differential receive channels
- 176K receive message storage capacity through single merged receive buffer
- 64-bit, 1 µsec time-tag stored with each received message
- ARINC 429 Parity error detection

Avionics Discrete Input and Output

- Eight dedicated avionics-level discrete channels
- Output may switch to ground up to 500mA
- Fixed input threshold of 2.7 +/- 0.2 volts

IRIG Input and Output

- IRIG Time-code receiver and transmitter

Power Consumption

Table 1. Power Consumption

+5V
500 mA max

Operating Temperature

-40 to +75° C

Weight

9.3 ounces (without cable)

I/0 Connections

Figure 2. The RAR-USB

I/O Mating Connector

At publication of this document, the following mating connector was compatible with the 68-pin latched SCSI-3 connector provided on the RAR-USB. Abaco Systems supplies the cable CONSCSI3-6 for this connection, which has the same connector.

Table 2. P2 Input/Output Connector

Connector	Part No	Description	Manufacturer
P2	1-5750913-7	68 pin SCSI-3	AMP/Tyco

RAR-USB Input /Output Connector Pin-out

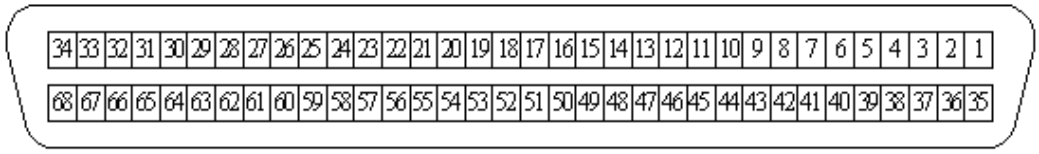


Figure 3. 68-pin I/O Connector – View Facing Connector

Table 3. RAR-USB Front Panel I/O Connections

Pin	Signal	Pin	Signal
1	TX1A	35	TX1B
2	TX2A	36	TX2B
3	TX3A	37	TX3B
4	TX4A	38	TX4B
5	TX5A [2]	39	TX5B [2]
6	Ground [1]	40	Ground [1]
7	IRIGRX+	41	IRIGRX-
8	IRIGTX	42	Ground [1]
9	DISCRETE #1	43	Ground [1]
10	DISCRETE #2	44	Ground [1]
11	DISCRETE #3	45	Ground [1]
12	DISCRETE #4	46	Ground [1]
13	DISCRETE #5	47	Ground [1]
14	DISCRETE #6	48	Ground [1]
15	DISCRETE #7	49	Ground [1]
16	DISCRETE #8	50	Ground [1]
17	Ground [1]	51	Ground [1]
18	RX1A	52	RX1B
19	RX2A	53	RX2B
20	RX3A	54	RX3B
21	RX4A	55	RX4B
22	RX5A	56	RX5B
23	RX6A	57	RX6B
24	RX7A	58	RX7B
25	RX8A	59	RX8B
26	Ground [1]	60	Ground [1]
27	RX9A	61	RX9B
28	RX10A	62	RX10B
29	RX11A	63	RX11B
30	RX12A	64	RX12B
31	RX13A	65	RX13B
32	RX14A	66	RX14B

Pin	Signal	Pin	Signal
33	RX15A	67	RX15B
34	RX16A [3]	68	RX16B [3]

Notes:

1 The ground pins are provided for shielding or as discrete I/O return lines, as required.

2 The ARINC 573/717 transmit signals are supported on the respective channel 5 (TX5A/TX5B) pins for both the BPRZ and HBP protocols.

3 The ARINC 573/717 receive signals are supported on the respective channel 16 (RX16A/RX16B) pins for both the BPRZ and HBP protocols.

The RAR-USB product configurations all have the same channel pin-out definition, adjusted based on the number of channels and protocols included. The following table describes the front panel connector pin layout for the RAR-USB. For the -J version of the RAR-USB, the ARINC 573/717 protocol support pins are assigned to receive channel 16 (RX16A/B) and transmit channel (RX5A/B).

Note: The RAR-USB ARINC 429 I/O pin assignments are pin-compatible with the CEI-520/520A and RCEI-530 I/O pin assignments.

To externally wrap ARINC signals, connect the transmitter signals to the respective receiver signals, TXnA to RXnA and TXnB to RXnB.

Optional RCONRARUSB-EC Adapter Cable

An optional adapter cable is available to provide a connection which is ARINC 429 and Discrete I/O pin-compatible with the RAR-EC/(R)CEI-715 adapter cables (CONRAR-EC, CONCEI-715, RCONCEI-715A), via the RCONRARUSB-EC cable.

RCONRARUSB-EC Adapter Cable Pin-out

The pin-out for the RCONRARUSB-EC Adapter Cable 37-pin D-Subminiature receptacle connector is shown below:

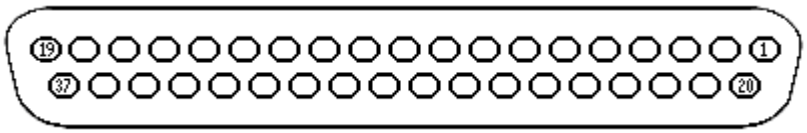


Figure 4. RCONRARUSB-EC Connector – View Facing Connector

Table 4. RCONRARUSB-EC Adapter Cable I/O Connections

Adapter Pin	SIGNAL	Adapter Pin	SIGNAL
1	RX1A	20	RX1B
2	RX2A	21	RX2B
3	RX3A	22	RX3B
4	RX4A	23	RX4B
5	RX5A	24	RX5B
6	RX6A	25	RX6B
7	RX7A	26	RX7B
8	RX16A [2]	27	RX16B [2]
9	RX8A	28	RX8B
10	TX1A	29	TX1B
11	TX2A	30	TX2B
12	TX3A	31	TX3B
13	TX4A	32	TX4B
14	Discrete #1	33	Discrete #2
15	Discrete #3	34	Discrete #4
16	TX5A [3]	35	TX5B [3]
17	Ground	36	Ground
18	IRIG TX	37	IRIG RX-
19	IRIG RX+		

Notes:

1 The ground pins are provided as Discrete I/O return lines or for shielding, as required.

2 The ARINC 573/717 receive signals are supported on the respective channel 16 (RX16A/RX16B) pins for both the BPRZ and HBP protocols.

3 The ARINC 573/717 transmit signals are supported on the respective channel 5 (TX5A/TX5B) pins for both the BPRZ and HBP protocols.

IRIG-B Signal Connections

IRIG-B time (AM or DC/TTL) may be received via signals IRIGRX+ and IRIGRX-. The following IRIG formats are accepted:

Table 5. IRIG Signal Connections

Format	Modulation Frequency	Frequency/ Resolution	Coded Expressions
B	0, 1	0, 2	0, 1, 2, 3

Upon completion of the program load, the RAR-USB initiates IRIG B002 (DC/TTL) transmission from the onboard IRIG encoder via the IRIGTX signal. The IRIGTX signal can source/sink 16 mA at valid TTL levels.

To externally wrap the IRIG generator to the IRIG receiver, connect the IRIGTX signal to IRIGRX+ input, and connect the IRIGRX- input to Ground.

CHAPTER 2

Software Installation

Software Installation under Windows

Although system resources may limit the number of boards installed on a system, the AR-STREAM-SW distribution supports up to four devices when installed under one of the operating systems documented in the section *Operating Systems Supported*. Prior to inserting the RAR-USB into a USB port, the AR-STREAM-SW software distribution must be installed on your PC.

To install the software, follow these steps:

1. Exit all programs.
1. Insert the AR-STREAM-SW CD into your CD drive.
2. If the installation does not automatically start after 10 seconds:
    - Click Start from the Windows Task Bar and select Run.
    - Use the Browse button to locate the Setup.exe file in the **Setup\Disk1** folder.
    - Double-click the file **setup.exe**. Then, click OK to launch the setup program.
3. Follow the on-screen instructions for the installation.
4. Note which device number is allocated during the installation.

Device Driver Installation under Windows

Once the software has been successfully installed, attach the RAR-USB cable to the RAR-USB device (threading the screw-locks), then insert the RAR-USB USB connector into any USB port on your PC.

Driver Installation with Windows

To complete the installation under all supported Windows operating systems, follow these steps:

With Windows 11, 10, 8.1, 8, 7, and Server 2012R1+R2/2008R2, the RAR-USB device driver installation should occur automatically. Once installation completion is acknowledged, you may continue with Installation Verification.

For 32-bit Windows XP the Windows Plug and Play hardware manager should detect the RAR-USB device, and the *Found New Hardware* dialog should automatically startup. Decline any request to query the Microsoft web site to obtain drivers for this device and continue as follows:

- When the Found New Hardware Wizard dialog is displayed for *RAR-USB Firmware Download*, select the following option:

  Install the software automatically (Recommended)

  Then select Next. Under the Completing the Found New Hardware dialog, select Finish.

- After a brief period the Found New Hardware Wizard dialog will then be displayed for *RAR-USB*, select the following option:

  Install the software automatically (Recommended)

  Then select Next. Under the Completing the Found New Hardware dialog, select Finish.

If Windows does not detect the new hardware, you should contact Abaco Systems Technical Support.

To check for proper driver installation (not necessary for any Windows version), review the device status in the Windows Device Manager as follows:

- Under Windows 7, XP and Server 2008R2, use the "Start->Run" command prompt, enter "devmgmt.msc" to open the Windows Device Manager.
- Under Windows 11, 10, 8.1, 8 and Server 2012, open the "Run" application and enter "devmgmt.msc" to open the Windows Device Manager
- Expand the Abaco Avionics Devices folder.
- Verify the device entry *RAR-USB* is shown. If this is true, the device driver was properly installed. You have completed installation of the hardware.

Multiple RAR-USB Device Installations

When multiple RAR-USB devices of the same configuration are installed on a host, the respective Device ID referenced by the application software must be manually assigned to each RAR-USB using the *USB Device Id Setup Utility*. This action assigns software/API referenced Device ID's to a specific device via the Serial Number programmed into that device, also printed on the underside of the device case.

1. From the **Abaco AR-STREAM-SW** apps/shortcut group, launch the "**AR-STREAM-SW USB Device Setup**" application. When multiple RAR-USB devices are installed, they should all appear in the Device Setup dialog, with the first device referenced by the lowest Device ID highlighted.
2. Click on "Edit Device…" to launch the "Edit Device ID" dialog.
3. Expand the Serial Number pull-down and select the Serial Number on the RAR-USB you wish to assign to this Device ID.
4. Click on the OK button to close the "Edit Device ID" dialog.
5. Highlight/select the second device.
6. Click on "Edit Device…" to launch the "Edit Device ID" dialog.
7. Expand the Serial Number pull-down and select the Serial Number on the RAR-USB you wish to assign to this Device ID (assure it is not the same serial number selected in step C).
8. Click on the OK button to close the "Edit Device ID" dialog.
9. Click on the Check Device Setup button on the Device Id Setup Utility dialog.
10. Verify the message "The current device setup is valid." appears in the Device Setup Validation Status window.
11. Click on the Apply button in the lower right corner of the Device Id Setup Utility dialog.
12. Close the "AR-STREAM-SW USB Device Setup" application.

Installation Verification under Windows

To verify the device driver is properly installed, execute the Test Configuration program.

1. Under Windows 7, XP and Server 2008R2
   a. Click Start, then Programs.
   b. Find and expand the **Abaco AR-STREAM-SW** program group
   c. Invoke the **Test RAR-USB Installation** shortcut located therein.
2. Under Windows 8, 8.1 and Server 2012R1+R2
   a. Display “Apps by name”.
   b. Find and invoke the **Test RAR-USB Installation** shortcut.
3. Under Windows 10 and 11
   a. Expand “All apps”.
   b. Scroll down and expand the **Abaco AR-STREAM-SW** application group.
   c. Invoke the **Test RAR-USB Installation** shortcut beneath.

You should verify the CONFIG LED on the top of the RAR-USB illuminates within a few seconds of invocation (and for multiple RAR-USB device installations, indicates the expected RAR-USB device referenced by the Device ID supplied). This program executes an internal wrap test on all available channels and notifies you of success or failure. If the program reports success on all channels tested, you are ready to use your RAR-USB device.

Software Installation under Linux

Abaco Systems provides the RAR-USB Linux distribution package, divided into two sections:

1. The USB support requires using the libUSB library, and can be installed or built from the source files at libusb.info. Abaco Systems takes no responsibility for its usage, therefore all technical support inquiries concerning this package must be directed to the package's admin.

   NOTE: The USB driver package does not support "sysfs"; instead, it requires using the "ceidev.conf" file.

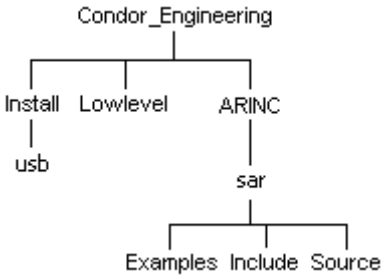
2. Second is the RAR-USB API distribution, consisting of an API library and example application demonstrating how to use the library.

Installing the RAR-USB Linux distribution requires the RAR-USB device be installed prior to execution of the following installation procedure:

1. You must log on as "root" (you may use "su")
13. Copy the Linux distribution compressed tar file (linux_rar_usb_vnnn.tgz) to the /root directory.
14. Uncompress and extract the installation file using the following:

    tar -zxvf linux_rar_usb_v*nnn*.tgz

After the tarball extraction completes, the following directory structure will be created:



Building Applications under Linux

Automatic Installation (Builds LSP and API)

Navigate to the Install directory and run the installation script by typing

./install usb

These are the configuration arguments that are accepted by the "install" script:

1. To debug the kernel device driver(s), include the option "debug_drv=<DEBUG LEVEL>" in the "./install" command line. The debug statements will be printed out to the kernel message log. The <DEBUG LEVEL> provides increasing debug information with a range of "0" (none) to "3" (all).
15. To debug the low level library, include "debug_ll" in the "./install" command line. The debug statements will be printed to stdout.
16. To build the low-level and API libraries as 32-bit libraries to run in 32-bit emulation mode for 64-bit systems, include "32bit" in the "./install" command line.

The installation is finished. Check the "install" script output and the kernel message log for any errors. If there are no errors then the device driver is

loaded into the kernel, low-level library is built, and the API distribution is ready to verify. To test the installation, navigate to the Examples directory and execute the *exapp* (or *exapp64*) application.

Manual Installation

Refer to the file *Linux_install.txt* in your distribution, section "Manual Install" concerning the manual installation of the Linux distribution and/or driver.

CHAPTER 3

RAR-USB Product Features

Overview

The RAR-USB product provides specialized features for receive message storage and time-tagging, timer usage, and transmit message scheduling. The following paragraphs document several of these features, and how they might be used in your ARINC application.

Programmable Transmit Channel Tri-State Control

The RAR-USB provides the ability to tri-state the transmitter output at the I/O connector pins using the *Transmit Disable Bit*, allowing any device connected to the respective I/O pins to drive ARINC 429/717 levels on the same bus with no adverse effect. The transmitter outputs will drive the respective transmit lines low until disabled by the host application.

Controlling the state of the *Transmit Disable Bit* is performed using the AR_SET_DEVICE_CONFIG API routine with the ARU_TX_DISABLE option; control of individual channel transmission or reception message processing is provided using the AR_SET_DEVICE_CONFIG API routine with the ARU_TX_FIFO_ENABLE and ARU_RX_FIFO_ENABLE options, respectively.

ARINC 429 Protocol Support

Several aspects of the ARINC 429 protocol are handled by the RAR-USB product.

The electrical transmission of ARINC 429 data over the bus is performed with the label field in reverse bit order. The transmit logic of the RAR-USB product automatically reverses the bit order of the ARINC 429 message label (message bits b0-b7) prior to transmission. The receiver logic of the RAR-USB also reverses the bit order of the ARINC 429

message label prior to placing the data in the respective receive buffers. This ARINC 429 label modification is fixed in the RAR-USB processing and cannot be modified by the application. For more information on the ARINC 429 protocol, see the "ARINC Tutorial" document.

ARINC 429 message parity is defined in the MSB of the ARINC 429 message. The RAR-USB transmission logic provides the capability to generate either odd or even parity based on the bit states of the first 31 bits of the ARINC 429 message. When disabled, the transmitter logic transmits the ARINC 429 message with the parity bit unaltered. When enabled, the RAR-USB product overwrites the value of the parity bit in the 32-bit user-defined message with the calculated parity value.

The RAR-USB reception logic provides the capability to detect the parity of ARINC 429 messages based on the bit states of each message. When disabled, the receiver logic provides the full ARINC 429 message as it was received, (without modification). If enabled, the receiver logic modifies the state of the parity bit (b32) to be "0" if the parity was detected as odd and "1" if the parity was detected to be even.

The bus speed for both ARINC 429 transmission and reception may be programmed to any baud rate from 3.9Kbps to 800Kbps; however, the transmitted signal slew rate doesn't provide for a viable signal beyond 150Kbps.

The API routine AR_SET_DEVICE_CONFIG provides the method to assign transmit and receive channel bus speed and parity selections for your device (using options ARU_TX_BITRATE/ ARU_RX_BITRATE and ARU_TX_PARITY/ ARU_RX_PARITY, respectively).

ARINC 573/717 Protocol Support

Several aspects of the ARINC 573/717 HBP and BPRZ protocols are handled by the RAR-USB product.

The electrical transmission of ARINC 573/717 data over the bus is performed at various bus speed/sub-frame size combinations resulting in the standard four-second frame duration. Each frame consists of four sub-frames comprised of a sub-frame sync word and subsequent data words. Each sync and data word is 12 bits long, transmitted in LSB-MSB order.

The transmit logic of a RAR-USB board relies on the application to supply the frame data in a 16-bit unsigned integer array in which only the lower 12 bits of each 16-bit element are used. The application must supply the applicable sub-frame sync words and data in the respective locations within this array for proper frame transmission.

The receiver logic of a RAR-USB board supports both raw and auto-synchronized frame data reception. With raw data frame data reception,

the data captured and provided to the application is organized in the least significant 12-bits of each element of a 16-bit unsigned integer array, based on the first detected bit transition. With auto-synchronized frame data reception, four application-provided sub-frame sync words are used by the RAR-USB ARINC 717 receive logic to synchronize reception and data logging to a detected sub-frame sequence. The sub-frame detection is based both on the provided sub-frame sync word bit patterns and the specified sub-frame size.

The API routine AR_SET_573_CONFIG provides the method to set transmit and receive channel bus speed and frame size options for your device.

RAR-USB Timers

The RAR-USB product supports two independent timers, a 64-bit one-microsecond timer and an optional IRIG timer. The one-microsecond timer is utilized for all ARINC 429 receive message time-tagging. It can be assigned to any 64-bit value by the host at any time.

Specified in microseconds from January 1st of the current year, received IRIG time is based on an external IRIG reference connected to the IRIG input of the RAR-USB device (see the section, “IRIG B Signal Connections” for the procedure to connect the RAR-USB” device to the IRIG source). If the IRIG time reference is desired, but no external IRIG source is available, the RAR-USB IRIG generator may be internally wrapped and used as the time source (see the ARU_IRIG_WRAP_ENABLE option of AR_SET_DEVICE_CONFIG); however, if the RAR-USB IRIG generator is to be used by other data collection hardware in your system, it is best to externally connect the RAR-USB IRIG output to its IRIG input. The RAR-USB IRIG generator can be reset by the host application to any desired value using the standard IRIG time format, (see AR_SET_TIME).

A user-programmable compensation to the RAR-USB IRIG time value can be defined when a consistent offset to the IRIG source time value is desired. This compensation should be used when a consistent skew in IRIG time-tagging is encountered between ARINC 429 events occurring on the RAR-USB and other IRIG time-tagged components in your system. This offset can be specified via the ARU_IRIG_SET_BIAS option of AR_SET_DEVICE_CONFIG.

An IRIG DAC threshold adjustment procedure is provided that configures the RAR-USB device IRIG receiver for optimal signal reception. This procedure is usually not necessary; however, it may be required if IRIG timing appears unstable from a known good source.

First, test the stability of the IRIG signal by invoking AR_GET_DEVICE_CONFIG with the option

ARU_IRIG_CALIBRATED. If this invocation returns a FALSE status, the adjustment should be invoked through use of the AR_SET_CONFIG routine with the ARU_IRIG_QUICK_ADJUSTMENT option. If the quick DAC adjustment is not successful, a more thorough adjustment may be performed. This adjustment is invoked through the ARU_IRIG_ADJUST_THRESHOLD option of the AR_SET_DEVICE_CONFIG routine. Execution of this IRIG adjustment may require at least one minute and should be performed only during the initialization of the board.

In addition to IRIG reception, RAR-USB product can be configured to generate IRIG time using on-board IRIG circuitry. The transmitted IRIG time value is initialized to the host calendar time by the API, and can be modified by the host application via AR_SET_TIME.

Receive Message Time-tagging and Timer Usage

The RAR-USB product time-stamps ARINC 429 received messages in the merged receive buffer with a 64-bit one-microsecond time-tag. This time-tag value is based on the on-board timer, recorded when the last bit of the 32-bit message is detected. The RAR-USB API supports multiple time-tag reference methods based on this one-microsecond timer. The active timer reference mode may be assigned by the host application by invoking the API routine AR_SET_DEVICE_CONFIG, using the ARU_RX_TIMETAG_MODE option and the selections discussed below. This assignment determines the format of the timer/time-tag value returned from all API invocations providing time-related information.

The following receive message time-tag and timer-read reference modes are available for selection:

IRIG 64-Bit Time Reference

This time reference is based on the RAR-USB IRIG receiver, with the one-second resolution extrapolated by the one-microsecond internal timer to provide a somewhat accurate estimation of a one-microsecond IRIG reference value. When the IRIG time reference is selected, all legacy receive data API routines based on a 32-bit time-tag parameter return a time-tag value with a resolution of one millisecond; while all receive data API routines supporting a 64-bit time-tag will return an IRIG timer-based time-tag.

The RAR-USB device records the internal timer value when the IRIG signal is received and decoded, (referred to as IRIG Reference Time). The API then calculates the offset between the IRIG Reference Time and the received ARINC data time-tag. Finally, the API applies that offset to the IRIG signal time value to produce an IRIG-reference message time stamp for the received data, extrapolated to provide the 1 microsecond resolution.

Internal 64-Bit One Microsecond Time Reference

This time reference is based on the RAR-USB device one-microsecond timer. When this mode is active, all legacy receive data API routines based on a 32-bit time-tag and all routines based on a 64-bit time-tag return a time-tag value with a resolution of one microsecond. The 32-bit time-tag is returned as the lower 32-bits of the 64-bit time-tag. This internal timer can be reset by the host application to any value desired, (see AR_SET_TIME).

Internal 32-Bit Twenty Microsecond Time Reference

This time reference is provided for backward compatibility to applications designed around the CEI-710 or IP-429HD-based products. When this mode is active, all legacy receive data API routines based on a 32-bit time-tag return a time-tag value with a resolution of twenty microseconds; all receive data API routines based on a single 64-bit time-tag value return a 32-bit value with a resolution of twenty microseconds (the upper 32-bits of the time-tag is zero).

Internal 32-Bit One Millisecond Time Reference

This time reference is based on a scaled version of the RAR-USB device one-microsecond timer. When this mode is active, all legacy receive data API routines based on a 32-bit time-tag/timer value return a 32-bit value with a resolution of one millisecond; all receive data API routines based on a 64-bit time-tag/timer value return a 64-bit value with a resolution of one millisecond.

CEI-x20 Compatible Time Reference

This time-tag and timer option is not available as a selection via AR_SET_CONFIG/ ARU_RX_TIMETAG_MODE; instead, this time reference mode is selected when the CEI-x20 legacy API routine AR_SET_TIMERRATE is invoked. In this mode, time references are based on a programmable time-tag resolution specified through AR_SET_TIMERRATE. When this mode is active, all receive data API routines return a time-tag/timer value based on either a 32-bit or 64-bit value scaled using the application-defined resolution. The message rate and start offset attributes assigned to scheduled message table entries are also scaled to the application-defined resolution.

Receive Message Buffering

The RAR-USB provides a single receive message storage method via the Merged Receive Buffer. This buffer provides for time-based sequentially ordered receive message buffering for multiple receive channels in a single circular buffer. The merged buffer can store up to four seconds of messages on 16 fully loaded receive channels before overflowing.

The AR-STREAM API maintains individual receive buffer and snapshot buffer storage, simulating the same buffer storage features provided with other Abaco Systems ARINC board products.

It is important to avoid a buffer overflow, as all messages received subsequent to a buffer overflow will be lost. For this reason, when using either of these circular buffer storage methods for data retrieval, the host application should periodically flush the buffer(s). The API routines AR_GETWORD*, AR_GETNEXT*, AR_GET_DATA*, and AR_GET_BLOCK* are provided to support various data retrieval methods for retrieving messages from the receive buffer(s).

Receive Buffer Entry Format

The format for the Merged Receive Buffer message entry is dependent on the protocol of the received data. The protocol data format is described as follows:

ARINC 429/575 Data Format

31	30 – 10	9 - 8	7 - 0
Parity Indication or ARINC Data MSB	ARINC Data	SDI bits or ARINC Data bits 0-1	ARINC Label (MSB – LSB)

If the Parity Enable bit is set to one in the respective Receive Channel Configuration register, the Parity Indication is set by the device to indicate the parity of the message. A Parity Indication bit value of zero indicates this message was received with ODD parity, where a Parity Indication value of one indicates the message was received with EVEN parity. If the Receive Configuration register Parity Enable bit is set to zero, this bit is not manipulated by the device; instead, it reflects the value of bit 31 as transmitted from the ARINC 429 source.

ARINC 573/717 Data Format

31 – 16	15	14	13 - 12	11 – 0
Unused	Sync Indication	Unused	Sub-frame Identification	Data Word

When the Sync Indication bit is set to one, it is an indication this word was detected as a sub-frame sync word. A Sync Indication bit value of zero indicates the message was treated as a data word. The Sub-frame Identification bit field identifies the sub-frame assignment for this word; where a value of one indicates sub-frame 1, two indicates sub-frame 2, three indicates sub-frame 3, and zero indicates sub-frame 4. The data word contains the 12-bit value designated as ARINC 717 frame data.

ARINC 429 Receive Label Filtering

The RAR-USB product provides the capability to filter received ARINC 429 messages from storage in the merged receive buffer. The filter definition for ARINC 429 message filtering is based on the combination of matching 8-bit Label value, 2-bit SDI field value, and 3-bit ESSM field value, with these fields defined within a 32-bit ARINC 429 message as follows:

eSSM	SDI	Label
30, 29, 28	9, 8	7, 6, 5, 4, 3, 2, 1, 0

Each receiver contains a separate label filter table section in which the filter definition is applied. This table is used by the RAR-USB firmware to control storage of received messages to the merged receive buffer.

Transmit Message Processing Methods

The RAR-USB product provides an individual transmit message buffer mechanization for each installed transmit channel, with three methods of invoking message transmission for the ARINC 429 protocol:

1. Burst Transmission
2. Message Scheduler
3. Transmit Playback

While the Burst Transmission method is always available for use by the host application, use of the Message Scheduler and Transmit Playback features are mutually exclusive. This is due to the potential for significant conflict of message transmission programmed by these two methods.

The method for invoking ARINC 573/717 transmission is based on burst transmission from a single large circular buffer.

ARINC 429 Burst Transmission

Direct access by the host application to individual transmit message buffers is supported for the ARINC 429 protocol. Messages are transmitted in the order they are inserted into individual transmit message buffers at the speed at which the respective bus is programmed. General methods supported by the RAR-USB API for inserting messages into transmit buffers include AR_PUTWORD and AR_PUT_429_MESSAGE for single message insertion, and AR_PUTBLOCK and AR_PUTBLOCK_MULTI_CHAN for multiple message insertion. The individual transmit message format for each of these methods is described as follows:

ARINC 429/575 Data Format

31	30 – 10	9 – 8	7 - 0
Parity Bit or ARINC Data MSB	ARINC Data	SDI bits or ARINC Data bits 0-1	ARINC Label (MSB – LSB)

If ODD or EVEN Parity is enabled for the respective Transmit Channel, bit 31 of the 32-bit data word will be overwritten by the device. The value assigned to bit 31 is based on the parity type and bit content of the remaining 31 bits. If Parity is disabled for the respective channel, bit 31 remains unchanged from the value provided.

ARINC 717 Frame Transmission

Direct access by the host application to a single ARINC 573/717 transmit message buffer is provided. Frames are transmitted in the order they are inserted into the transmit buffer at the speed at which the bus is programmed. General methods supported by the RAR-USB API for inserting messages into the ARINC 717 transmit buffers include AR_PUTWORD for single message insertion, and AR_PUT_573_FRAME for full ARINC 573/717 frame transmission. The individual transmit message format for these methods is described as follows:

ARINC 573/717 Data Format

31 – 12	11 – 0
Unused	Data Word

ARINC 573/717 data is assigned to the lower 12 bits of the 32-bit word for AR_PUTWORD or the lower 12 bits of each 16-bit array elements for AR_PUT_573_FRAME.

ARINC 429 Periodic Message Scheduling

The RAR-USB message scheduling feature supports periodic message transmission of ARINC 429 messages, with a total of 1024 message entries. It is programmed by writing message and rate information to the Message Scheduler table, supported by the API routines AR_DEFINE_MSG, AR_DEFINE_MSG_BLOCK, AR_MODIFY_MSG, and AR_MODIFY_MSG_BLOCK. As a part of the API initialization of the device, the Message Scheduler table is reset to an empty state. Once entries are defined by the host application, message scheduling is enabled by invoking the AR_GO routine.

When enabled, the Message Scheduler queries each table entry on a one millisecond basis, checking for all messages required for transmission at that particular millisecond value. The entire table is processed each millisecond, with the lowest table entry being processed first and highest table entry last. When invoking AR_STOP or AR_RESET, it is important to note the scheduler processing responds only to being disabled at the beginning of a one millisecond epoch. If the scheduler is requested to disable in the middle of creating scheduled traffic, all of the ARINC messages previously scheduled for that millisecond are loaded into the various transmit buffers before the scheduler transitions to idle.

The efficiency of the Message Scheduler is based on the number of defined messages and the frequency at which those messages are transmitted. While the Message Scheduler feature is designed to be very accurate, there are ways in which the host application definition of channel-specific message transmission scenarios may cause deviations in the periodic transmission of the messages defined therein. The most common deviation is referred to as message rate skew.

Message Rate Skew

Message rate skew is defined as the characteristic of a scheduled message appearing on the bus at a rate that is either above or below the defined rate by a significant percentage. Message rate skew typically occurs when several different message rates are defined simultaneously on the same channel, and a majority of these message rates are multiples of the other message rates. The example below illustrates this situation.

Assuming there are three groups of messages being transmitted at rates of 100 msec (referenced as block A), 200 msec (block B) and 300 msec (block C). If messages from each of the different rate groups were defined in the order of rate priority using same initial starting point of reference, the transmission of data would be defined as follows:

Time	Group
100	A
200	A,B
300	A,C

Time	Group
400	A,B
500	A
600	A,B,C
repeat…	

If we assume that each group of messages requires 10 msec to transmit, we can expand the timeline in more detail as follows:

Time	Group
100	A
200	A
210	B
300	A
310	C
400	A
410	B
500	A
600	A
610	B
620	C
700	A
800	A
810	B
900	A
910	C
etc.	

The time between the first two occurrences of the 300 msec message group (block C) is 310 msec. The time between the second and third occurrences of this group is 290 msec. Message skew like this is unpredictable as the number of different message rates increases.

The solution to this problem is to use the *start offset* feature of the message scheduler, (see the description for AR_DEFINE_MSG). In the next alternative example, the 300 msec message group was defined with a start offset of 20 msec (see note below). In this scenario, no message rate skew would occur, (as shown in the following timeline).

Note: The 20 msec offset was derived as the sum of the duration required to transmit the groups that precede this group in the scheduling order.

Time	Group
100	A
200	A
210	B
300	A
320	C
400	A
410	B
500	A

Time	Group
600	A
610	B
620	C
700	A
800	A
810	B
900	A
920	C
etc.	

In this transmission example, any start offset from 20 msec to 80 msec would suffice for the 300msec (block C) message group. A start offset greater than 90 msec would cause this message group to overlap into the next scheduled frame for the block A message group and would subsequently induce skewing for those messages.

If the three message rate groups were all even multiples of each other (e.g. 100, 200, and 400 msec) then rate skew would never occur. The good news is that, although the slowest rate messages are most susceptible to rate skew, they are also typically the most tolerant to variation in transmission time.

Since the message scheduler processes all messages in the order of their location in the schedule table, rate skew may also appear on faster rate messages defined further into the table following slower rate messages. This skew can be easily eliminated by defining faster rate messages first and slower rate messages last.

In conclusion, if the minimum rate skew is desired on all transmitted messages, you must make an a priority determination of the message loading on each channel and insure messages are scheduled not only with the fastest rates first, but taking full advantage of the start offset feature.

The AR-STREAM API provides a runtime utility routine AR_ASSIGN_SCHEDULER_START_OFFSETS, designed to read the current message scheduler channel and label rate content and assign start offsets to each defined message on a best-fit rate-priority basis. This routine should be called after all scheduled messages are defined, prior to activation of message processing on the board.

If you desire manual control of the message transmission scenario, the AR-STREAM-SW distribution provides a 32-bit Windows application in the distribution folder \Help\Start Offset Assistant called *gen_offsets.exe*. This application will generate the start offsets for the messages supplied for a set of transmit channels, when supplied in the proper scheduled message data structure input text file format. See the document *Start_Offset_Assistant.pdf* located in the same folder for a description of the application and input file format. Example input and output files are provided to demonstrate the required format.

ARINC 429 Transmit Playback Message Processing

The RAR-USB Transmit Playback feature supports timer-synchronized transmission of ARINC 429 messages, providing a circular buffer of 53,248 playback message entries. It is programmed by writing playback entries to the RAR-USB transmit playback buffer in time-sequential order. Each playback entry consists of a 32-bit ARINC 429 message, a transmit channel number, and a 64-bit 1 microsecond resolution playback time value. Once entries are written to the buffer and Transmit Playback is enabled, the Transmit Playback feature will initiate counting for the internal 1-microsecond playback timer and sequentially transmit the messages on the respective channel as each entry's playback time matches the current playback timer value.

Entries are written to the RAR-USB Transmit Playback buffer through the API routine AR_WRITE_429_TRANSMIT_PLAYBACK. Once entries are written to the Transmit Playback buffer, the Transmit Playback feature is enabled via the routine AR_SET_DEVICE_CONFIG using the item parameter option ARU_TX_PLAYBACK_ENABLE.

An example of a simple transmit playback scenario is shown below (parity, data and SDI fields are zero for the purpose of this example). This example consists of transmitting a mixture of a periodic 100 millisecond (10Hz) message having a label value of o002 on transmit channel 2 (ch index 1) with a burst of five aperiodic message blocks having a label value of o224 (0x94) once per second on channel 5 (ch index 4), for a duration of just over two seconds.

Playback Time (μsec)	Message	Channel Index
100000	0x00000002	1
200000	0x00000002	1
300000	0x00000002	1
400000	0x00000002	1
500000	0x00000002	1
600000	0x00000002	1
700000	0x00000002	1
800000	0x00000002	1
900000	0x00000002	1
910000	0x00000094	4
910000	0x00000094	4
910000	0x00000094	4
910000	0x00000094	4
910000	0x00000094	4
1000000	0x00000002	1
1100000	0x00000002	1
1200000	0x00000002	1
1300000	0x00000002	1
1400000	0x00000002	1
1500000	0x00000002	1
1600000	0x00000002	1
1700000	0x00000002	1

Playback Time (µsec)	Message	Channel Index
1800000	0x00000002	1
1900000	0x00000002	1
1910000	0x00000094	4
1910000	0x00000094	4
1910000	0x00000094	4
1910000	0x00000094	4
1910000	0x00000094	4
2000000	0x00000002	1
etc.		

In this example, once Transmit Playback is enabled, label o002 will be transmitted on each 100 millisecond interval, with label o224 burst transmitted in a burst of consecutive messages on the bus on a 1 second interval offset from the previous transmission of label o002 by precisely 10 milliseconds.

It is important to note that Playback Time values that are less than the value of the current Playback Timer result in the Transmit Playback entry being processed immediately and the message inserted in the respective transmit channel buffer.

Avionics Discrete Inputs and Outputs

The RAR-USB provides for individually configurable bi-directional Avionics Discrete Input/Output channels, used for general avionics-level I/O interfacing. Each discrete output circuit is implemented as a low side FET switch capable of sinking 500mA to ground, while the inputs are single ended, protected (50V max), with a logic threshold of approximately 2.0V. See the paragraph *Avionics Discrete I/O* for a detailed description of the Discrete Input/Output circuit.

To assign a Discrete Output state, use the AR_SET_DEVICE_CONFIG API call with the item parameter selection ARU_DISCRETE_OUT and one of the following value parameter selections:

AR_LO Discrete Output set to 1 (FET ON – conduct to Ground)
AR_HI Discrete Output set to 0 (FET OFF – tri-state)

CHAPTER 4

BusTools/ARINC™ Data Bus Analyzer

General Information

BusTools/ARINC™ is an optional ARINC 429 analysis and simulation utility which runs under Windows. It enhances the utility of an underlying ARINC 429 interface board by expanding your scope of control and by providing additional instrumentation and analytical tools. Additionally, BusTools/ARINC™ provides support for devices configured with ARINC 561, 573/717, or Commercial Standard Digital Bus (CSDB) channels.

BusTools/ARINC™ supports usage of up to four boards at the same time or independently and allows simultaneous control of all channels on each board.

Its data logging function streams data to disk or memory and replays it in a time-sequenced display. It provides multiple buffering mechanisms, including real-time display of data in engineering units. Strings of outgoing messages are generated, repeated, or automatically stepped through a sequence. Strings of incoming messages are filtered and captured for current or future analysis. A database of standard ARINC 429 translations is included. Translation among binary, hexadecimal, and engineering units is provided, as is a powerful user-defined label facility.

BusTools/ARINC Demo Software

A free demo version of BusTools/ARINC™ is available on our web site at ‘https://www.abaco.com/products/bt-arinc-bustools-software-analyzer’. The demo software operates over a simulated ARINC 429 interface board, but is otherwise identical to the full version.

CHAPTER 5

AR-STREAM-SW Software Distribution

Overview

The AR-STREAM-SW software distribution contains all of the API and example source files, libraries, and additional features necessary to support RAR-USB board installation and use of under the Windows operating system.

API Source Files

This library of utility routines provides the ability to write your own programs to interface with an RAR-USB product. They are written in C and delivered in a generic ANSI C compiler-compatible format. They can be called from other languages by adhering to the procedures defined in the applicable documentation. The API consists of the following C source files:

SAR_API.C

This file contains the streaming device specific API functionality.

GEN_ARINC_API.C

This file contains the common ARINC 429/717 API functionality provided with our existing Avionics ARINC product APIs. Most of these routines provide backward compatibility to our other Avionics software distributions at an API function and parameter definition level, limited to those features available with the RAR-USB product.

SAR_PROCESS_THREAD.C

This file contains the functionality that interfaces with the streaming device via the low-level USB driver interface.

SAR_API.H

This header file contains the majority of the API constants, data types, and function prototypes required for host application development, and should be included in all C/C++ programs that reference AR-STREAM API routines.

SAR_TYPES.H

This header file contains AR-STREAM API private/internal definitions and data structures.

SAR_ERROR.H

This header file contains the error string constant definitions utilized by the API routine AR_Get_Error, describing each of the potential error codes returned by the AR-STREAM API routines.

SAR_HW.H

This header file contains all of the API constants that define the hardware interface for the RAR-USB architecture, included in SAR_TYPES.H.

CEI_TYPES.H

This header file contains all of the type defines for the various data types used with the respective operating system and compiler; included in SAR_TYPES.H.

SAR_OS_WIN.C

This file contains the C routines that interface directly with the Abaco Systems common low-level driver interface library, CEI_Install.LIB/DLL, supporting all Windows operating systems.

SAR_OS_LNX.C

This file contains the C routines that interface directly with the Abaco Systems common Linux Support Package and libUSB library, supporting Linux kernel version 3.x.

SAR_API.DEF and SAR_API64.DEF

These are the Windows DLL export definition files for 32-bit and 64-bit Windows operating systems, respectively.

Windows Libraries

For the AR-STREAM-SW supported products, separate 32-bit and 64-bit Windows API Libraries are provided. For Windows OS target implementation, all API function prototypes are declared "_stdcall". The RAR-USB API library included in the installation is referenced as:

- sar_api_32.lib 32-bit Microsoft VS6.0 Library
- sar_api_32.dll 32-bit Microsoft VS6.0 DLL
- sar_api_64.lib 64-bit Microsoft VS2008 Library
- sar_api_64.dll 64-bit Microsoft VS2008 DLL

Included with the installation are the Abaco Systems USB Driver and Abaco Systems Common Low-level driver interface and installation verification libraries (not required for linking application programs):

- cei_install.dll 32-bit Microsoft VS6.0 DLL
- ceiusbar1_api_32.dll 32-bit Microsoft VS6.0 DLL
- cei_install64.dll 64-bit Microsoft VS2008 DLL
- ceiusbar1_api_64.dll 64-bit Microsoft VS2008 DLL

All DLLs are installed in the Windows "System" folder. The exact folder name depends on the host version of Windows operating system. The 32-bit versions of these DLLs are typically installed in either 'c:\winnt\system32' or 'c:\windows\system32' under 32-bit Windows or 'c:\windows\syswow64' under 64-bit Windows. The 64-bit versions of these DLLs will be installed in the 64-bit Windows system folder (typically 'c:\windows\system32' under 64-bit Windows).

Time-tag Structure Definition

The following API routines use the AR_TIMETAG_TYPE data structure definition in providing the timer/time-tag reference or as an initial value with a reset of the internal timer:

- AR_GET_TIME
- AR_SET_TIME
- AR_GETNEXT_XT
- AR_GETWORD_XT
- AR_GET_DATA_XT
- AR_CONVERT_TIME_TO_STRING

Under the Windows and VxWorks operating systems, the AR_TIMETAG_TYPE data structure and pAR_TIMETAG_TYPE pointer types used by these routines are defined to use 64-bit integer values, as follows:

timeTagFormat	__int64 or long long The format of the corresponding *timeTag* structure member. Valid values for this element are:

AR_TIMETAG_EXT_IRIG_64BIT

AR_TIMETAG_INT_USEC_64BIT

AR_TIMETAG_HOST_USEC_64BIT[1]

AR_TIMETAG_INT_20USEC_32BIT

AR_TIMETAG_INT_MSEC_32BIT

AR_TIMER_X20_COMPAT_32BIT

timeTag	__int64 or long long The timer-referenced time-tag, formatted as specified in the *timeTagFormat* structure member.
referenceTimeTag	__int64 or long long The original 64-bit, one microsecond RAR-USB board timer/time-stamp value reference

[1] This format returns the host OS system time value converted to have a 1 microsecond resolution, supported only by the API routine ar_get_time().

for the time value supplied in the timeTag member.

Setting the Device Time

When assigning an initial time reference, the host application may choose to set either the device 1 microsecond timer or the IRIG generator timer via invocation of AR_SET_TIME.

When AR_SET_TIME is invoked with an AR_TIMETAG_TYPE data structure parameter *timeTagFormat* member defined to be AR_TIMETAG_EXT_IRIG_64BIT, the format of the *timeTag* member is defined as a 30-bit entity of BCD-like values using the following format:

29-28	27-24	23-20	19-18	17-14	13-11	10-7	6-4	3-0
hundreds of days	tens of days	days	tens of hours	hours	tens of minutes	minutes	tens of seconds	seconds

When AR_SET_TIME is invoked with a *timeTagFormat* member defined to be AR_TIMETAG_INT_USEC_64BIT, the *timeTag* member is referenced as a 64-bit 1 microsecond timer value.

Return Status Values

The following return status values are used by the RAR-USB API routines. They are defined in the C header file SAR_API.H and are used in the following context:

C Constant	Value	Constant Definition
ARS_FAILURE	-1	Requested operation failed
ARS_NODATA	0	No data was detected or received
ARS_NORMAL	1	Normal successful completion
ARS_GOTDATA	4	Data was received
ARS_BAD_MESSAGE	5	Receipt of an invalid ARINC 429 message was detected
ARS_INVHARVAL	1003	Invalid configuration value
ARS_XMITOVRFLO	1004	Transmit buffer overflow
ARS_INVBOARD	1005	Invalid board argument
ARS_NOSYNC	1006	Transmit buffer flush failed
ARS_MEMWRERR	1013	SRAM memory test error
ARS_INVARG	1019	General invalid argument value
ARS_DRIVERFAIL	1021	Driver failed to install or uninstall the ISR

C Constant	Value	Constant Definition
ARS_WINRTFAIL	1022	Device driver open failure
ARS_CHAN_TIMEOUT	1023	Channel timeout in receive function
ARS_NO_HW_SUPRT	1024	Function not supported by specified hardware
ARS_WRAP_DATA_FAIL	1031	BIT wrap test data read-back fail
ARS_WRAP_FLUSH_FAIL	1035	BIT cannot execute external wrap test due to unknown external data reception
ARS_WRAP_DROP_FAIL	1036	BIT wrap test data not received
ARS_BOARD_MUTEX	1038	API routine failed to acquire or release a shared resource lock mechanism
ARS_NO_OS_SUPPORT	1041	There is no operating system support for the requested feature
ARS_ERR_SH_MEM_OBJ	1050	API failed to allocate a shared object (semaphore or mutex)
ARS_ERR_SH_MEM_MAP	1051	API failed to allocate a shared memory region (multi-process)
ARS_FW_NOT_SUPPORTED	1052	The firmware programmed on the board is not compatible with the API version in use.

Programming with the AR-STREAM API Interface

Communications over USB differs dramatically from motherboard or backplane accessible busses such as PCI, PCI Express, CompactPCI or PCMCIA, in that USB latency due to operating system intervention cannot be controlled by the application, API, or driver interface. While most internal bus interfaces provide direct and almost immediate access from the host application to the hardware on the device, access to a USB device is restricted by the operating system and its interaction with the associated USB device driver. Latency in the interaction between the AR-STREAM API and the RAR-USB can range from negligible to in excess of 500 milliseconds, depending on the resources and CPU time other active processes might be simultaneously requesting from the host o/s. For this reason, application requested interaction between the API and the RAR-USB is restricted to queued operations bundled via background scheduled periodic download/upload data requests in efficient, block transactions.

With this API design method, completion of any API function that relates to the modification of a device, channel, or individual message attribute

does not guarantee the RAR-USB device has received the intended programmed information; instead, completion of an API function invocation only guarantees the information has been processed by the API and will be updated on the RAR-USB device during the next set of USB block transactions. The AR-STREAM API provides the routine *ar_block_on_device_update* that blocks application execution until the device has been updated, preventing unintended system dependencies in the application. In some limited cases such as the AR_GO and AR_STOP routines, this block on execution is built into the AR-STREAM API.

Following the outline below, you can easily incorporate the RAR-USB into your application.

1. For your application to interface to any AR-STREAM-SW supported products the device must first be initialized. Invoke the AR_OPEN routine with parameters as described in the API Routines section to open a session and initialize the RAR-USB.

17. Assign the characteristics of the transmit and receive channels if the default configuration is not appropriate. This is performed with multiple invocations of either AR_SET_DEVICE_CONFIG or AR_SET_CONFIG.

18. Select the desired receive buffer mode based on individual channel/protocol usage via the routine AR_SET_DEVICE_CONFIG. Using buffered mode for multiple channels of the same protocol provides channel-specific access to received data, where merged mode provides for single receive channel access to selected channel data.

19. Once channel configuration is complete, invoke AR_GO to initiate data processing.

20. Then invoke AR_PUT_429_MESSAGE and AR_GET_429_MESSAGE to send and receive single ARINC 429 messages, respectively.

21. When communication is complete, invoke AR_STOP to suspend active data processing.

    Subsequently, you could invoke AR_GO again to restart the interface.

22. On termination of the application, invoke AR_CLOSE to release all resources acquired during initialization. It is very important that all applications invoke AR_CLOSE upon termination; otherwise, the operating system does not release the memory acquired when the API was initialized.

The example wrap program source code, contained in EXAMPLE_APPLICATION.C, is supplied with your installation. This program demonstrates the use of the API for the ARINC 429 and 717 protocols.

When calling the utility routines that return a status value, it is important to verify the returned status indicates success; otherwise, the application may

not be aware that an important function may have failed to fulfill a requested operation.

Example Routines – Summary

Example applications demonstrating various RAR-USB API features are provided in C source format, as described in the following paragraphs.

EXAMPLE_APPLICATION.C

The example source file EXAMPLE_APPLICATION.C is included with your installation. To access this example executable under the Windows operating system:

1. Click on **start**, and then **Programs**.

23. Select **AR-STREAM-SW** and then **Test RAR-USB Installation**.

Within EXAMPLE_APPLICATION.C are application-style routines demonstrating use of the API routines for the various features provided:

test_basic_arinc_429	An internal wrap test designed to demonstrate ARINC 429 API usage. This routine enables internal wrap on all 429 receive channels. It also assigns a bus speed of 12.5kbps and ODD parity to both transmit and receive channels. Ten ARINC 429 messages are sent on each transmit channel and proper reception verified on the respective receive channel.
demo_advanced_arinc_429	A demonstration of the following advanced features available with RAR-USB products: Transmit Message Scheduling Enhanced Label/SDI/ESSM Data Filtering Snapshot Message Data Acquisition Enhanced Time-tag Reset and Conversion IRIG Time-tag Selection (if installed on hardware)
demo_429_transmit_playback	A demonstration of the ARINC 429 transmit playback feature
demo_discrete_io_features	A demonstration on the use of the Discrete I/O Channels and the respective API routines
demo_irig_features	A demonstration of IRIG features requiring an external IRIG connection: IRIG DAC Threshold Adjustment

	IRIG Bias (Offset) Time Assignment IRIG Validity Determination IRIG Time Conversion and Display
test_arinc_717	An internal wrap test designed to demonstrate ARINC 573/717 API usage. This routine enables internal wrap on the ARINC 573/717 receive channel. It also assigns a bus speed 768bps, a sub-frame size of 64 words, and a BPRZ selection to the ARINC 573/717 transmit and receive channels. A frame consisting of a data pattern incrementing from $01 to $FF and sync words of $123, $224, $325, and $426 is transmitted and proper reception verified.

Dealing with Complex Message Scheduler Transmit Scenarios

Whether a transmit channel is operating at 100Kbps or 12.5Kbps, any scheduled message scenario that introduces as little as 50% bus loading is susceptible to message rate skew. The ability to assign proper offsets to the scheduled message rate definitions can become cumbersome in such situations. For these cases two options are provided:

The Start Offset Assistant utility will accept an input text file containing a C-like message scheduler structure array layout with defined channel message scenarios and generate an output text file with assigned start offset values. The contents of the output text file can then be copied as a C data structure array into the application source, to be used with the API's message scheduler support routines. See the "Start Offset Assistant" User's Manual in the folder **\Utilties\Start Offset Assistant** within this distribution for a detailed description on the use of this tool.

The AR-STREAM API also provides a utility routine that will update the contents of the existing message scheduler table with start offset values calculated in a best attempt to avoid rate skew based on the programmed bus speed for individual transmit channels. The start offset values are computed based on the message count and rates defined per transmit channel. For more details, see the routine description **ar_assign_scheduler_start_offsets** within this document.

CHAPTER 6

Program Interface Library

Overview

Abaco Systems supplies an extensive software Application Programming Interface (API) for the RAR-USB product, distributed as AR-STREAM-SW. API routines are supplied to setup the interface, configure channel attributes, and transmit and receive ARINC 429 and 717 messages.

API Routines - Summary

The routines provided in the API supporting the RAR-USB device features, (with backward compatibility to most of our legacy ARINC API's), are categorized and summarized in the following pages:

Initialization and Control Routines

ar_open	The main initialization routine acquiring the resources for the PCI memory regions and initializing the RAR-USB device.
ar_board_test	Verifies the RAR-USB data processing capabilities via internal/external data wrap.
ar_loadslv	A legacy version of the ar_open routine.

Device Control Routines

ar_go	Enables RAR-USB ARINC data processing.
ar_reset	Disables RAR-USB ARINC data processing and initializes the device to the default state.
ar_stop	Disables RAR-USB ARINC data processing.

Termination Routines

ar_close	Releases all resources for the specified device.

Receive/Transmit Channel-level Configuration Routines

ar_set_device_config	As the main channel configuration routine, it assigns ARINC 429-specific transmitter and receiver channel configuration information.
ar_get_device_config	Retrieves the value of a bit field for I/O and ARINC 429 transmitter or receiver channel configuration registers.
ar_enh_label_filter	Assigns the enhanced label filter table definition for each RAR-USB device receiver.
ar_get_config	Retrieves board-level configuration and API local attribute values.
ar_get_573_config	Retrieves the value of a bit field for an ARINC 573/717 transmitter or receiver channel configuration register.
ar_get_filter	Retrieves the specified label filter buffer entry from the enhanced label filter table.
ar_get_label_filter	Retrieves the active state of label filtering for a single label on all receivers.
ar_label_filter	Assigns ARINC 429 label values to be filtered by the specified receive channel.
ar_putfilter	Places the specified label filter buffer entry in the enhanced label filter table.
ar_set_config	Assigns board-level configuration and API local attribute values.
ar_set_573_config	Assigns ARINC 573/717 transmitter and receiver channel configuration information.

Device-level Configuration Routines

ar_get_storage_mode	Retrieves the current selected API data storage mode.
ar_set_raw_mode	Assigns both transmitter and receiver parity state on the specified ARINC 429 channel.

ar_set_storage_mode	Assigns the API data storage mode.

Receive Data Processing Routines

ar_get_429_message	Retrieves the next ARINC 429 message from a receive buffer. Optionally, it waits up to ½ second for data to become available.
ar_get_573_frame	Retrieves a specified number of ARINC 573 data words from the receive buffer.
ar_getnext	Retrieves the next message from the specified receive buffer. It waits up to ½ second for data to become available.
ar_getnextt	Retrieves the next message and a 32-bit, 20 µsec time-tag from the specified receive buffer. It waits up to ½ second for data to become available.
ar_getnext_xt	Retrieves the next message with a 64-bit, user-programmable time-tag from the specified receive buffer. It waits up to ½ second for data to become available.
ar_getword	Retrieves the next message from the specified receive buffer.
ar_getwordt	Retrieves the next message and a 32-bit, 20 µsec time-tag from the specified receive buffer.
ar_getwordt_xt	Retrieves the next message from the specified receive buffer with a 64-bit, user-programmable time-tag.
ar_get_data	Retrieves the next available data and the 64-bit, 1 µsec time-tag from a receive buffer.
ar_get_data_xt	Retrieves the next available data from a receive buffer with a 64-bit, user-programmable time-tag.
ar_getblock	Retrieves all of the available ARINC 429 messages from the requested receive buffer with 32-bit time-tags.
ar_getblock_t	Retrieves all of the available ARINC 429 messages from the requested receive buffer, with 64-bit time-tags.

ar_get_latest	Retrieves the latest message from the snapshot buffer for the specified channel/label combination.
ar_get_latest_t	Retrieves the latest message and time-tag from the snapshot buffer for the specified channel/label combination.
ar_get_snap_data	Retrieves the latest message from the API snapshot buffer for the specified channel/label/sdi combination.

Transmit Data Processing Routines

ar_assign_scheduler_start_offsets	Computes and applies best fit start offset values for each transmit channel message scenario defined in the message scheduler table.
ar_define_msg	Defines a scheduled ARINC 429 message.
ar_define_msg_block	Defines a block of scheduled ARINC 429 messages.
ar_modify_msg	Modifies an existing ARINC 429 message already defined for periodic transmission.
ar_modify_msg_block	Modifies a block of ARINC 429 messages already defined for periodic transmission.
ar_put_429_message	Places a single message in the specified ARINC 429 transmit buffer.
ar_put_573_frame	Places a specified number of ARINC 573 data words in the transmit buffer.
ar_putword	Places a single message in the specified ARINC 429 transmit buffer.
ar_putblock	Places multiple messages in a single ARINC 429 transmit buffer.
ar_putblock_multi_chan	Places multiple messages in multiple ARINC 429 transmit buffers.
ar_write_429_transmit_playback	Places transmit playback entries in the ARINC 429 transmit playback buffer

Timer-related Routines

ar_get_time	Retrieves the current hardware reference time based on the selected timer mode.

ar_get_timercntl	Retrieves the current value of the OS timer.
ar_reset_timercnt	Resets the internal timer/time-tag reference to zero.
ar_set_timerrate	Assigns the RAR-USB compatible timer resolution for use with ar_get_time() and any ARINC 429 receive data routines returning 32-bit time-tag values.
ar_set_time	Sets the internal clock/timer or IRIG time generator to an application supplied value.

Information and Status Routines

ar_get_boardname	Returns a string description of the specified device.
ar_get_boardtype	Retrieves the target device configuration.
ar_get_error	Retrieves a message string associated with a given error status code.
ar_get_rx_channel_status	Reports the current buffer state of the specified ARINC 429 receive channel.
ar_get_status	Retrieves the combined state of each receive FIFO status register Data Available bit.
ar_num_rchans	Retrieves the number of receive channels supplied by the RAR-USB device.
ar_num_xchans	Retrieves the number of transmit channels supplied by the RAR-USB device.

Utility Routines

ar_block_on_device_update	Blocks the calling application execution until the next device update has completed.
ar_clr_rx_count	Resets the counter of received messages for the specified receive channel.
ar_convert_time_to_string	Converts a standard RAR-USB 64-bit time value to character string.
ar_execute_bit	Verifies the RAR-USB operational state through various data wrap and timer tests.
ar_get_rx_count	Returns the number of messages received for the specified receive channel.

ar_set_multithread_protect	Enable/disable multithread access protection to all API routines accessing the hardware interface of the device.
ar_set_preload_config	Defines the thread setup for the calling application.
ar_sleep	Suspends the calling thread for a specified number of milliseconds.
ar_wait	Delays the calling application for the specified number of seconds.
ar_version	Retrieves the current software version number of the RAR-USB API.

AR_ASSIGN_SCHEDULER_START_OFFSETS

Syntax

CEI_INT32 ar_assign_scheduler_start_offsets (CEI_INT16 board)

Description

This routine determines the appropriate start offset values for each transmit channel message scenario as a best estimate to avoid rate skew on the respective channel. It first reads all defined messages from the message scheduler table (any message having a non-zero rate attribute), determines the appropriate start offset value for all messages on a channel-by-channel basis, and then updates the respective start offset values in the table.

This routine should be called immediately following the last invocation of AR_DEFINE_MSG or AR_DEFINE_MSG_BLOCK, and must be called prior to the invocation of AR_GO. The transmit channel bus speed for all channels referenced in the message scheduler table entries must also be assigned prior to calling this routine.

To assign start offset values to a scheduled message scenario prior to the invocation of AR_DEFINE_MSG or AR_DEFINE_MSG_BLOCK, see the Start Offset Assistant utility.

Notes:

The start offset values defined by this routine do not account for bus timing issues and/or message rate skew due to bursts of aperiodic messages invoked by the host application.

All start offset values assigned assume a continuous transmission of all messages defined in the message scheduler table.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_FAILURE	The specified device has not been initialized.
ARS_INVBOARD	An invalid *board* parameter value was provided or the specified device's session is not active.
ARS_LOCK_ACCESS_FAILED	Releasing the access lock to the API shared device interface data structure failed.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_BLOCK_ON_DEVICE_UPDATE

Syntax

CEI_INT32 ar_block_on_device_update (CEI_INT16 board)

Description

This routine will block caller execution until the next communication sequence, consisting of download from the host and upload from the device, has completed.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_FAILURE	The specified device has not been initialized.
ARS_INVBOARD	An invalid *board* parameter value was provided or the specified device's session is not active.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_BOARD_TEST

Syntax

CEI_INT16 ar_board_test (CEI_INT16 board, CEI_INT16 testType)

Description

This routine tests the ARINC 429 message buffer processing functionality of the streaming device, designed specifically for the initialization process. It performs an internal or external wrap from each ARINC 429 transmit channel matched to a respective receive channel, (unmatched receive channels are not tested).

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_FAILURE	The specified device has not been initialized.
ARS_INVBOARD	An invalid *board* parameter value was provided or the specified device's session is not active.
ARS_WRAP_DROP_FAIL	ARINC 429 wrap test data missing.
ARS_WRAP_DATA_FAIL	ARINC 429 wrap test data pattern mismatch.
ARS_WRAP_FLUSH_FAIL	Unexpected data from an external source was received during wrap test execution.
ARS_XMITOVRFLO	A transmit buffer overrun occurred.
ARS_LOCK_ACCESS_TIMEOUT	The access lock to the API shared device interface data structure could not be acquired.
ARS_LOCK_ACCESS_FAILED	Releasing the access lock to the API shared device interface data structure failed.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 testType	(input) Type of test to execute. Valid values for this parameter are: INTERNAL_WRAP (0) EXTERNAL_WRAP (1)

AR_CLR_RX_COUNT

Syntax

CEI_VOID ar_clr_rx_count (CEI_INT16 board, CEI_INT16 channel)

Description

This routine resets the API-maintained count of ARINC 429 messages received by the specified channel to zero. The AR-STREAM API maintains a count of ARINC 429 messages received over the bus interface and uploaded to the API for each channel since the device was initialized, accessed via the API routine AR_GET_RX_COUNT.

Return Value

None

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_UINT32 channel	(input) Specifies which receive channel's count value will be reset to zero. Valid range is 0 to one less than the installed receive channel count.

AR_CLOSE

Syntax

CEI_INT16 ar_close (CEI_INT16 board)

Description

This routine releases all resources acquired during the initialization of the specified device. Once this routine has been executed, invocation of other API routines results in the return of an invalid status.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_FAILURE	Failed to access a session or complete termination.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_CONVERT_TIME_TO_STRING

Syntax

void ar_convert_time_to_string (CEI_INT16 board, CEI_INT16 displayFormat, pAR_TIMETAG_TYPE timeIn, pCEI_CHAR timeString)

Description

This routine converts the 64-bit time value provided in the timeIn structure to a character string representation of date/time, format based on what is specified via the displayFormat parameter. The supplied time format (LSB resolution) must be specified in the timeIn structure member **timeTagFormat**, representing the resolution of the respective **timeTag** member data.

Return Value

none.

Arguments

CEI_INT16 board (input) Device to access. Valid range is 0-127.

CEI_INT16 displayFormat (input) Format for returned string:

AR_TD_REL_MIDNIGHT (2) AR_TD_IRIG (1)	Relative to Midnight Format and Full IRIG Format, defined as "(DDD)hh:mm:ss.uuuuuu"
AR_TD_DATE (0)	Date Format defined as "(MM/DD)hh:mm:ss.uuuuuu"

pAR_TIMETAG_TYPE timeIn (input) Source time tag structure

pCEI_CHAR timeString (output) Pointer to destination text string.

AR_DEFINE_MSG

Syntax

CEI_INT16 ar_define_msg (CEI_INT16 board, CEI_INT16 channel, CEI_INT16 rate, CEI_UINT16 start, CEI_INT32 data)

Description

This routine defines a 32-bit ARINC 429 message for periodic retransmission at the specified rate. Once defined, the message rate, content, or assigned channel may be altered via AR_MODIFY_MSG.

Return Value

Any positive value between 0 and 1023 is the unique message scheduler table entry index assigned to this message.

ARS_FAILURE	Indicates the routine encountered an uninitialized board, an invalid board/channel parameter value, or a full message scheduler table.
ARS_LOCK_ACCESS_FAILED	Either the acquisition or relinquish of the access lock to the API shared device interface data structure failed.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 channel	(input) Channel message scheduler table this routine is to access. The valid range is 0 to one less than the number of installed transmit channels.
CEI_INT16 rate	(input) Periodic transmission rate, defined in milliseconds by default. For backward compatibility to the CEI-x20 tick-timer message rate method, when AR_SET_TIMERRATE has been executed to simulate the CEI-x20 tick-timer resolution assignment within the AR-STREAM API, the rate and start parameters is scaled to the specified tick-timer resolution.
CEI_UINT16 start	(input) Offset, (in milliseconds), from the start of RAR-USB device message

	processing at which this message will begin its initial periodic transmission.
CEI_INT32 data	(input) The 32-bit ARINC 429 message to transmit.

AR_DEFINE_MSG_BLOCK

Syntax

CEI_INT16 ar_define_msg_block (CEI_INT32 numberOfEntries, pAR_SCHEDULED_MSG_ENTRY_TYPE messageEntry)

Description

This routine defines a series of 32-bit ARINC 429 messages for periodic retransmission at their specified rates. Once defined, the message rate, content, or assigned channel for any individual message scheduler table entry within this same structure may be altered via invocation of AR_MODIFY_MSG_BLOCK.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* parameter value was provided.
ARS_INVARG	An invalid *numberOfEntries* parameter value was provided.
ARS_INVHARVAL	An invalid *channel* parameter value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_FAILED	Either the acquisition or relinquish of the access lock to the API shared device interface data structure failed.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT32 numberOfEntries (input) The number of entries to define from the subsequent structure pointer parameter, messageEntry.

pAR_SCHEDULED_MSG_ENTRY_TYPE messageEntry (input)

Array of structures of message definition content, defined as follows:

unsigned int messageIndex	The unique message scheduler table entry index assigned to this message. Upon completion of this routine, the messageIndex structure member will have been updated to

	reflect the message scheduler table index assigned to the respective message.
unsigned int board	Device to access. Valid range is 0-127.
unsigned int channel	Which channel portion of the message scheduler table this routine is to access. The valid range is 0 to one less than the number of installed transmit channels.
unsigned int rate	Periodic transmission rate, defined in milliseconds by default. For backward compatibility to the CEI-x20 tick-timer message rate method, when AR_SET_TIMERRATE has been executed to simulate the CEI-x20 tick-timer resolution assignment within the RAR-USB API, the rate and start parameters will be scaled to the specified tick-timer resolution.
unsigned int start	Offset, (in milliseconds), from the start of RAR-USB device message processing at which this message will begin its initial periodic transmission.
unsigned int txCount	The total number of times this message will be transmitted. The constant value ARU_SCHED_MSG_INFINITE (0xFFFFFFFF) indicates infinite transmission of this message is requested.
unsigned int data	The 32-bit ARINC 429 message to transmit.

AR_ENH_LABEL_FILTER

Syntax

CEI_INT32 ar_enh_label_filter (CEI_INT16 board, CEI_INT16 channel, CEI_UINT16 label, CEI_UINT16 sdi, CEI_UINT16 essm, CEI_INT16 action)

Description

This routine supports the assignment of both a single entry in the enhanced label filter table for the specified receive channel and channel-wide field definitions for the entire channel filter table. The RAR-USB device enhanced label filtering feature supports the ability to both filter ARINC 429 messages and generate a hardware interrupt based on reception of a message matching the combined 8-bit label value, 2-bit SDI value, and 3-bit ESSM value.

Note: This routine should be used exclusive of the use of the legacy API routine AR_LABEL_FILTER, as any filter table value assigned with one routine supersedes a previous assignment with another.

Once message reception filtering has been enabled for a specified channel/label/sdi/essm combination, data received with matching bit field values will be discarded until label filtering for that specified message has been disabled.

The label filtering feature is disabled for all labels/sdi/essm combinations by default. Label filtering changes are effective immediately on completion of this routine.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* parameter value was provided.
ARS_INVHARVAL	Invalid *channel* parameter value.
ARS_INVARG	Invalid *label*, *sdi*, *essm*, or *action* parameter value.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_FAILED	Either the acquisition or relinquish of the access lock to the API shared device interface data structure failed.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board — (input) Device to access. Valid range is 0-127.

CEI_INT16 channel — (input) Channel label filter table this routine is to access. The valid range is 0 to one less than the installed receive channel count.

CEI_UINT16 label — (input) The *label* of interest. Valid range is 0-255. Also valid is ARU_ALL_LABELS (511), which invokes the action for all labels.

CEI_UINT16 sdi — (input) The *SDI* field value of interest. Valid range is 0-3. Also valid is ARU_ALL_SDI (4), which invokes the action for all SDI entries for the specified label.

CEI_UINT16 essm — (input) The *ESSM* field value of interest. Valid range is 0-7. Also valid is ARU_ALL_ESSM (8), which invokes the action for all ESSM entries for the specified label.

CEI_INT16 action — (input) Enable or disable filtering action for this table entry. Valid values are:

- FILTER_SEQUENTIAL (0x10) enable filtering of the respective message from the device's merged receive buffer.
- ARU_FILTER_ON (1) enable filtering of the respective message from the device's merged receive buffer.
- ARU_FILTER_OFF (0) disable filtering (default state is to not filter any labels).

AR_EXECUTE_BIT

Syntax

CEI_INT16 ar_execute_bit (CEI_INT16 board, CEI_INT16 testType)

Description

This routine performs hardware diagnostic test functionality normally associated with device-level Built-In-Test (BIT). Testing ranges from a partial SRAM pattern test to verification of ARINC 429 message wrap on transmit/receive channel pair on the specified device.

Return Value

All of these

ARS_NORMAL	Routine execution was successful.
ARS_MEMWRERR	Device SRAM test write/read/verify failure.
ARS_WRAP_DROP_FAIL	ARINC 429 wrap test data missing.
ARS_WRAP_DATA_FAIL	ARINC 429 wrap test data pattern mismatch.
ARS_WRAP_FLUSH_FAIL	Unknown external data received during wrap test execution.
ARS_XMITOVRFLO	A transmit buffer overrun occurred during wrap test execution.
ARS_INVBOARD	An invalid *board* parameter value was provided.
ARS_INVARG	Invalid *testType* parameter value.
ARS_FAILURE	The specified device has not been initialized or the timer-deviation test failed.
ARS_LOCK_ACCESS_FAILED	Either the acquisition or relinquish of the access lock to the API shared device interface data structure failed.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board — (input) Device to access. Valid range is 0-127.

CEI_INT16 testType — (input) Type of test to execute, defined as follows:

AR_BIT_BASIC_STARTUP (0) invokes a basic device initialization to a reset state (all buffers flushed and channel configurations reset).

AR_BIT_FULL_STARTUP (1) invokes device initialization, a non-destructive SRAM memory test, and an internal wrap test of all matched transmit/ receive channels, (regardless of prior invocation of the API routine AR_BYPASS_WRAP_TEST).

AR_BIT_PERIODIC (2) invokes a non-destructive SRAM memory test and a timer-deviation test, providing verification of the basic health status of the device.

AR_BIT_INT_LOOPBACK (3) invokes an internal wrap test of all matched transmit/receive channels.

AR_BIT_EXT_LOOPBACK (4) invokes an external wrap test of all matched transmit/receive channels.

AR_BIT_PARTIAL_SRAM (8)

AR_BIT_FULL_SRAM (9)

AR_BIT_SELECT_SRAM_MIN to AR_BIT_SELECT_SRAM_MAX (100 to 1123)

All of these options invoke a non-destructive pattern test of an unused 8Kb block of SRAM.

AR_GET_573_FRAME

Syntax

CEI_INT16 ar_get_573_frame (CEI_INT16 board, pCEI_INT32 numberWords, pCEI_UINT16 arincData)

Description

This function retrieves *numberWords* of ARINC 573/717 data from the ARINC 573/717 receive channel. If any data is available, the actual number of words received is indicated in the return value of *numberWords*. If auto-synchronization is configured for the ARINC 573/717 channel, this function will search the receive buffer for any occurrence of the first sub-frame sync word (defined via invocation of AR_SET_573_CONFIG with the item set to ARU_573_SYNC_WORD1) and return the specified number of words of frame data following the instance of that sync word. With automatic synchronization selected and the full frame size specified in *numberWords*, this function will wait until the full frame is received and copied to the destination array. The acquisition of an entire ARINC 573/717 frame may require up to four seconds to complete.

If the API Receive Storage Mode is set for Merged Mode buffering, all ARINC 429 receivers should be disabled and only the ARINC 573/717 receiver enabled to receive and log bus traffic in the merged buffer. If other receivers are active, messages received on those receivers will be logged along with any active ARINC 573/717 data as they are received and the content of the frame data will be corrupt.

Return Value

ARS_NODATA	No frame data was available.
ARS_GOTDATA	At least one ARINC 573/717 data word has been retrieved.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_INVHARVAL	ARINC 573 support is not available on the specified device.
ARS_INVARG	An invalid *numberWords* or *arincData* parameter was supplied.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_FAILED	Either the acquisition or relinquish of the access lock to the API shared device interface data structure failed.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.

ARS_RWR_EXECUTE_FAIL The specified device failed a communications block execution request.

Arguments

CEI_INT16 board (input) Device to access. Valid range is 0-127.

pCEI_UINT32 numberWords (input/output) As an input this specifies the number of words to retrieve from the receive buffer. As an output this indicates how many words were retrieved from the receive buffer, less than or equal to the input value of *numberWords*.

pCEI_UINT16 arincData (output) The address that is to receive the frame data. The format of each data word in the ARINC 573/717 frame is defined as follows:

15	**14**	**13 – 12**	**11 - 0**
sync word	RESERVED	subframe	data

sync word: indicates this word was detected as a sync word, where a value of 1 indicates sync word and 0 indicates data word.

subframe: identifies the sub-frame assignment for this word, where 1 indicates sub-frame 1, 2 indicates sub-frame 2, 3 indicates sub-frame 3, and 0 indicates sub-frame 4.

data: the 12-bit ARINC 573/717 data.

AR_GET_429_MESSAGE

Syntax

CEI_INT16 ar_get_429_message (CEI_INT16 board, CEI_INT16 channel, CEI_INT16 waitState, pCEI_VOID data, pCEI_VOID timetag)

Description

This routine retrieves the most recent ARINC 429 data and 32-bit time-tag from the specified channel. If no data is present in the receiver buffer, this routine attempts to retrieve data for up to one-half second. If no data is present after one-half second, a time-out status is returned. If **no wait** is specified and no data is available, the return status is so indicated.

Return Value

ARS_GOTDATA	An ARINC 429 message and its time-tag have been retrieved.
ARS_CHAN_TIMEOUT	No data available (if waitState is AR_ON).
ARS_NODATA	No data available (if waitState is AR_OFF).
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received.
ARS_INVBOARD	An invalid *board* value was provided .
ARS_INVHARVAL	Channel is not available on the device.
ARS_INVARG	A NULL *data* parameter value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which receive channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
CEI_INT16 waitState	(input) Whether or not to wait for data. A value of AR_ON specifies to wait ½ second

	for data; a value of AR_OFF specifies to return if no data is immediately available.
pCEI_VOID data	(output) The address that is to receive the data. The returned ARINC 429 data is always in normal ARINC format.
pCEI_VOID timetag	(output) The address that is to receive the 32-bit time-tag associated with the data, (resolution is programmable, default is one millisecond). If the merged receive mode is active, the upper five bits of the 32-bit time-tag word will contain the receive channel number on which the data was received. If the *timetag* parameter is NULL, time-tag information will not be provided.

AR_GETBLOCK

Syntax

CEI_INT16 ar_getblock (CEI_UINT32 board, CEI_UINT32 channel, CEI_INT32 maxMessages, CEI_INT32 offset, pCEI_INT32 actualCount, pCEI_INT32 data, pCEI_INT32 timeTags);

Description

This routine attempts to retrieve the requested number of ARINC messages from the specified receive buffer. If the *timeTags* parameter is not NULL, the 32-bit time-tag data associated with each retrieved message is also copied. The *actualCount*, *data*, and *timeTags* parameters are only valid when *Return Value* is ARS_GOTDATA or ARS_BAD_MESSAGE.

Return Value

ARS_GOTDATA	Message(s) and time-tag(s) were retrieved.
ARS_NODATA	No data was available.
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received.
ARS_INVBOARD	An invalid *board* value was provided .
ARS_INVHARVAL	Channel is not available on device.
ARS_INVARG	Invalid *maxMessages*, *actualCount*, or *data* parameter was encountered.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_UINT32 board	(input) Device to access. Valid range is 0-127.
CEI_UINT32 channel	(input) Specifies which receive channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
CEI_INT32 maxMessages	(input) The number of messages to retrieve.
CEI_INT32 offset	unused parameter, retained for legacy API support.
pCEI_INT32 actualCount	(output) The number of messages retrieved.

pCEI_INT32 data	(output) Array to store 32-bit ARINC data.
pCEI_INT32 timeTags	(output) Array to store 32-bit time-tag data, (resolution is programmable, default is one millisecond). If the merged receive mode is active, the upper five bits of the 32-bit time-tag word will contain the receive channel number on which the data was received. If the *timetag* parameter is NULL, time-tag information will not be provided.

AR_GETBLOCK_T

Syntax

CEI_INT16 ar_getblock_t (CEI_UINT32 board, CEI_UINT32 channel, CEI_INT32 maxMessages, pCEI_INT32 actualCount, pCEI_UINT32 msgChan, pCEI_INT32 data, pCEI_INT32 timeTagMsw, pCEI_INT32 timeTagLsw)

Description

This routine retrieves the available ARINC 429 messages from the requested receive channel buffer and copies them to the desired destination. If the *msgChan*, *timeTagMsw*, and *timeTagLsw* parameters are not NULL, the receive channel and 64-bit time-tag data associated with each retrieved message are also copied. The *actualCount*, *msgChan*, *data*, *timeTagMsw* and *timeTagLsw* parameters are only valid when *Return Value* is ARS_GOTDATA or ARS_BAD_MESSAGE.

Return Value

ARS_GOTDATA	Message(s) and time-tag(s) were retrieved.
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received.
ARS_NODATA	No data was available.
ARS_INVBOARD	An invalid *board* value was provided .
ARS_INVHARVAL	Channel is not available on device.
ARS_INVARG	Invalid *maxMessages*, *actualCount*, or *data* parameter was encountered.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_UINT32 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_UINT32 channel	(input) Specifies which channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
CEI_INT32 maxMessages	(input) The number of messages to retrieve.
pCEI_INT32 actualCount	(output) The number of messages retrieved.

pCEI_UINT32 msgChan	(output) Array to store the receiver channel indication, necessary for actual receive channel determination when using the Merged Receive Mode.
pCEI_INT32 data	(output) Array to store 32-bit ARINC 429 data.
pCEI_INT32 timeTagMsw	(output) Array to store the most significant 32-bits of the 64-bit time-tag data.
pCEI_INT32 timeTagLsw	(output) Array to store the least significant 32-bits of the 64-bit time-tag data.

AR_GET_BOARDNAME

Syntax

pCEI_CHAR ar_get_boardname (CEI_INT16 board, pCEI_CHAR boardName)

Description

This routine returns a character string describing the board name for the RAR-USB device. It should only be invoked after successful invocation of AR_LOADLSV.

Return Value

NULL	An uninitialized board or invalid *board* value was provided.

For any valid detected board, the return value is a character string description defined as “RAR-USB”

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
pCEI_CHAR boardName	(output) If a valid board is detected and this parameter is not NULL, the character description of that board is copied to the location referenced by this parameter. A minimum of 7 bytes of allocation is required for the destination array.

AR_GET_BOARDTYPE

Syntax

CEI_INT16 ar_get_boardtype (CEI_INT16 board)

Description

This routine returns the API/device type for the specified device. It should only be invoked after successful invocation of AR_LOADLSV.

Return Value

ARS_INVBOARD	An invalid *board* value was provided .
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

For any value less than ARS_INVBOARD and not ARS_FAILURE, the return value indicates the type of board associated with the supplied *board* value:

RAR-USB (33)

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_GET_CONFIG

Syntax

CEI_INT32 ar_get_config (CEI_INT16 board, CEI_INT16 item)

Description

This routine returns the active state of API information, board level settings, and limited ARINC 429 channel configuration register bit fields. It is provided for backward compatibility to applications based on legacy ARINC 429 API usage. The routine AR_GET_DEVICE_CONFIG is the desired routine for acquiring information regarding channel and board-level configuration.

See the ARU_* definitions in the file SAR_API.H for the most current list of parameter options supported by this routine and the values associated with those definitions.

Return Value

If the requested item is ARU_RX_CH*nn*_BIT_RATE (500-515), where *nn* is the receiver channel (01 - 32), this routine returns the current value of the channel configuration register baud rate field:

AR_HIGH	(0) high rate (100Kbs)
AR_LOW	(1) low rate (12.5Kbs)

Any other value is returned as a frequency value in Hertz.

If the requested item is ARU_TX_CH*nn*_BIT_RATE (700-704), where *nn* is the transmitter channel (01 - 05), this routine returns the current value of the channel configuration register baud rate field:

AR_HIGH	(0) high rate (100Kbs)
AR_LOW	(1) low rate (12.5Kbs)

Any other value is returned as a frequency value in Hertz.

If the requested item is ARU_RX_CH*nn*_PARITY (900-915), where *nn* is the receiver channel (01 - 32), this routine returns the current state of the specified receiver channel configuration register parity field:

AR_ODD	(0) receiver parity check enabled
AR_OFF	(8) receiver parity check disabled

If the requested item is ARU_TX_CH*nn*_PARITY (1100-1104), where *nn* is the transmitter channel (01 - 05), this routine returns the current state of the specified transmitter channel configuration register parity field:

AR_ODD	(0) odd transmitter parity
AR_EVEN	(1) even transmitter parity

If the requested item is ARU_TX_CH*nn*_SHUT_OFF (1700-1704), where *nn* is the transmitter channel (01 - 05), this routine returns the current state of the specified transmitter channel configuration register transmit disable field:

AR_ON	(7) external transmission is disabled
AR_OFF	(8) external transmission is enabled

If the requested item is ARU_TX_CH*nn*_HB_INJ (3300-3304), where *nn* is the transmitter channel (01 - 05), this routine returns the current state of the specified transmitter channel configuration register high-bit error injection field:

AR_ON	(7) 33-bit transmission is enabled
AR_OFF	(8) standard 32-bit transmission is enabled

If the requested item is ARU_TX_CH*nn*_LB_INJ (3500-3504), where *nn* is the transmitter channel (01 - 05), this routine returns the current state of the specified transmitter channel configuration register low-bit error injection field:

AR_ON	(7) 31-bit transmission is enabled
AR_OFF	(8) standard 32-bit transmission is enabled

If the requested item is ARU_TX_CH*nn*_GAP_INJ (3700-3704), where *nn* is the transmitter channel (01 - 05), this routine returns the current state of the specified transmitter channel configuration register message gap error injection field:

AR_ON	(7) 3-bit message gap is used
AR_OFF	(8) standard 4-bit message gap is used

If the requested item is ARU_RX_TIMETAG_MODE (440), this routine returns a value representing the currently selected timer/time-tag source and resolution. This value indicates the resolution of any timer-read or receive data time-tag value obtained via the API, and is defined as follows:

AR_TIMETAG_EXT_IRIG_64BIT	(0)
AR_TIMETAG_INT_USEC_64BIT	(1)
AR_TIMETAG_INT_20USEC_32BIT	(3)
AR_TIMETAG_INT_MSEC_32BIT	(4)
AR_TIMER_X20_COMPAT_32BIT	(5)

A value of AR_TIMETAG_EXT_IRIG_64BIT indicates the source is the external IRIG receiver, if connected; otherwise, if the IRIG signal is not internally wrapped this selection would be invalid.

All other values represent various timer/time-tag LSB resolution values based on the internal RAR-USB device timer.

If the requested item is ARU_ACCESS_SNAPSHOT_BUFFER (38), this routine returns the currently selected Snapshot Buffer storage mode:

ARU_LABEL_ONLY	(0) messages stored based on label
ARU_LABEL_WITH_SDI	(1) messages stored based on the combined label and SDI field values

If the requested item is ARU_IRIG_WRAP_ENABLE (441), this routine returns the current state of the IRIG Receiver internal wrap feature:

AR_ON	(7) IRIG Receiver is patched into the IRIG Generator
AR_OFF	(8) IRIG Receiver is configured for external IRIG source

If the requested item is ARU_IRIG_AVAILALBE (445), this routine returns TRUE if IRIG-B support is available, FALSE if it is not.

If the requested item is ARU_IRIG_CALIBRATED (447), this routine verifies the ability to capture consecutive IRIG time samples at a one second interval. If this results in a return status of FALSE (0), the IRIG signal is not consistent; otherwise, a return value of TRUE (1) indicates the signal is valid, or an error status (any value greater than 1) indicates a failure occurred.

If the requested item is not on this list or in the list of valid items for AR_GET_DEVICE_CONFIG, this routine will return a value of ARS_INVARG.

If the requested item is not valid for the specified device, this routine returns a value of ARS_INVHARCMD.

If the specified board is invalid or has not been initialized, this routine returns ARS_INVBOARD.

If access to the Board Lock timed-out or failed, this routine returns ARS_BOARD_MUTEX.

Arguments

CEI_INT16 board — (input) Device to access. Valid range is 0-127.

CEI_INT16 item — (input) Control function about which to return information:

- ARU_RX_CH01_BIT_RATE – ARU_RX_CH16_BIT_RATE — receiver 1 – 16 bit rate selection.
- ARU_TX_CH01_BIT_RATE – ARU_TX_CH05_BIT_RATE — transmitter 1 – 5 bit rate selection.
- ARU_RX_CH01_PARITY – ARU_RX_CH16_PARITY — receiver 1 – 16 parity state.
- ARU_TX_CH01_PARITY – ARU_TX_CH05_PARITY — transmitter 1 – 5 parity state.
- ARU_TX_CH01_SHUT_OFF – ARU_TX_CH05_SHUT_OFF — transmitter 1 – 5 enable state.
- ARU_TX_CH01_LB_INJ – ARU_TX_CH05_LB_INJ — transmitter 1 – 5 low bit error enable state.
- ARU_TX_CH01_HB_INJ – ARU_TX_CH05_HB_INJ — transmitter 1 – 5 high bit error enable state.
- ARU_TX_CH01_GAP_INJ – — transmitter 1 – 5 message

ARU_TX_CH05_GAP_INJ	gap error enable state.
ARU_IRIG_AVAILABLE	IRIG Receiver installed state.
ARU_IRIG_WRAP_ENABLE	IRIG Receiver internal wrap state.
ARU_IRIG_CALIBRATED	IRIG signal validity.
ARU_ACCESS_SNAPSHOT_BUFFER	snapshot storage mode.
ARU_FW_VERSION	Hardware Version reg. value.
ARU_CONFIGURATION	configuration of the device.
ARU_RX_TIMETAG_MODE	active timer/time-tagging mode.

AR_GET_DATA

Syntax

CEI_INT16 ar_get_data (CEI_INT16 board, pCEI_INT16 channel, pCEI_UINT32 data, pCEI_UINT32 timeTagLo, pCEI_UINT32 timeTagHi)

Description

This routine retrieves the next unread message and 64-bit time-tag from the specified receive channel. If it successfully returns data, there may or may not be more data in the buffer. It means only that there was at least one message in the buffer. Subsequent calls are required to determine if more data words are available in the buffer. If this routine returns a status value of ARS_NODATA, the buffer is empty. If neither of the *timeTagLo* or *timeTagHi* parameters are NULL, the 64-bit time-tag data associated with the retrieved message is also retrieved.

If the AR-STREAM API receive message storage mode is set for *merged* mode operation, this routine returns the next unread message from the merged receive buffer and indicates on which channel the message was received via the *channel* parameter.

Return Value

ARS_GOTDATA	A message and time-tag have been received.
ARS_NODATA	No data available.
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_INVHARVAL	Channel is not available on device.
ARS_INVARG	Invalid *data* parameter was encountered.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
pCEI_INT16 channel	(input/output) As an input, specifies which hardware receive channel this routine is to access, (see the description of the Receive Channel Select Register – 0x0040 for the list of valid hardware receive channel values) . As an output, indicates the receive channel number on which the data was received (for merged-mode channel reporting).
pCEI_UINT32 data	(output) Address that is to receive the data.
pCEI_UINT32 timeTagLo	(output) Address that is to receive the least-significant 32 bits of the 64-bit time-tag associated with the data, (resolution of the combined time-tag words is 1 μsec).
pCEI_UINT32 timeTagHi	(output) Address that is to receive the most-significant 32 bits of the 64-bit time-tag associated with the data, (resolution of the combined time-tag words is 1 μsec).

AR_GET_DATA_XT

Syntax

CEI_INT16 ar_get_data (CEI_INT16 board, pCEI_INT16 channel, pCEI_INT32 data, pAR_TIMETAG_TYPE timeTagRef)

Description

This routine retrieves the next unread message and the associated time-tag from the specified receive channel. If it successfully returns data, there may or may not be more data in the buffer. It means only that there was at least one message in the buffer. Subsequent calls are required to determine if more data words are available in the buffer. If this routine returns a status value of ARS_NODATA, the buffer is empty. The time-tag structure is required to be defined by the host application with this routine.

If the AR-STREAM API receive message storage mode is set for *merged* mode operation, this routine returns the next unread message from the merged receive buffer and indicates on which channel the message was received via the *channel* parameter.

Return Value

ARS_GOTDATA	A message and time-tag have been received.
ARS_NODATA	No data available.
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received.
ARS_INVBOARD	An invalid *board* value was provided .
ARS_INVHARVAL	Channel is not available on device.
ARS_INVARG	Invalid *data* or *timeTagRef* parameter was encountered.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
pCEI_INT16 channel	(input/output) As an input, specifies which hardware receive channel this routine is to access. As an output, indicates the receive channel number on which the data was received (for merged-mode channel reporting).
pCEI_INT32 data	(output) Address that is to receive the data.
pAR_TIMETAG_TYPE timeTagRef	(output) The address that is to receive the time-tag data structure associated with the data.

AR_GET_DEVICE_CONFIG

Syntax

CEI_INT16 ar_get_device_config (CEI_INT16 board, CEI_INT16 channel, CEI_INT16 item, pCEI_INT16 value)

Description

This routine returns the state of the device configuration register attribute based on the combined item/value parameters. It is designed to support all ARINC 429 channel configuration register bit fields available to the device.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_INVARG	The *item* argument value is not supported by this API routine.
ARS_INVHARVAL	The *channel* argument value is not supported by this device configuration.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which channel this routine is to access. Valid range is 0 to one less than the installed channel count for the specified channel type. For board-level configuration items, this parameter is not used.
CEI_INT16 item	(input) configuration register or board level attribute for which to return the current state:

ARU_RX_PARITY	receive channel parity enable.
ARU_RX_BITRATE	receive channel bit rate.
ARU_RX_FIFO_ENABLE	receive channel enable.
ARU_RECV_MODE	receiver internal wrap.
ARU_TX_BITRATE	transmit channel bit rate.

ARU_TX_PARITY	transmit channel parity select.
ARU_TX_FIFO_ENABLE	transmit channel enable.
ARU_TX_DISABLE	transmit channel transceiver disable.
ARU_TX_BIT_ERROR	transmit channel bit error enable.
ARU_TX_GAP_ERROR	transmit channel gap error enable.
ARU_FAST_SLEW_RATE	transmit channel slew rate select.
ARU_ACCESS_SNAPSHOT_BUFFER	snapshot storage mode.
ARU_IRIG_WRAP_ENABLE	IRIG receiver internal wrap state.
ARU_IRIG_AVAILALBE	IRIG receiver installed state.
ARU_IRIG_INPUT_TIME	IRIG received sample value.
ARU_IRIG_CALIBRATED	IRIG signal validity.
ARU_DEVICE_DISABLE	Device disabled state.
ARU_DISCRETE_IN	discrete input state.
ARU_RX_TIMETAG_MODE	active timer/time-tagging mode.
ARU_CHAN_COUNT_429	ARINC 429 Tx channel count.
ARU_CHAN_COUNT_573	ARINC 573/717 channel count.
ARU_CHAN_COUNT_DISC	discrete I/O channel count.
ARU_RX_FIFO_COUNT	receive FIFO buffer fill count.
ARU_TX_FIFO_COUNT	transmit FIFO buffer fill count.
ARU_RX_MSG_COUNT	receive message count.
ARU_TX_MSG_COUNT	transmit message count.
ARU_FW_VERSION	current programmed firmware vers.
ARU_CONFIGURATION	board configuration type.

pCEI_INT16 value — (output) state of the configuration register attribute:

If the requested item is ARU_RX_FIFO_ENABLE (16) or ARU_TX_FIFO_ENABLE (17), this routine will return the current value of the specified channel configuration register FIFO Enable field:

AR_ON	(7) FIFO operation enabled
AR_OFF	(8) FIFO operation disabled

If the requested item is ARU_RX_BITRATE (1) or ARU_TX_BITRATE (2), this routine will return the current value of the specified channel configuration register baud rate field:

ARU_SPEED_HIGH	(0) high rate (100Kbs)
ARU_SPEED_LOW	(1) low rate (12.5Kbs)

If the respective channel is programmed to any other bus speed value, the returned value is represented as a non-standard bus speed clock divisor value for the 16MHz device clock reference. This value and the respective baud rate may be interpreted using the following formula:

Baud Rate = 16,000,000 / (Divisor Value+2)

If the requested item is ARU_RX_PARITY (3), this routine returns the current state of the specified receive channel configuration register parity field:

AR_ ON	(7) receiver parity check enabled
AR_OFF	(8) receiver parity check disabled

If the requested item is ARU_TX_PARITY (4), this routine returns the current state of the specified transmitter channel configuration register parity field:

ARU_PARITY_ODD	(0) odd transmitter parity
ARU_PARITY_EVEN	(1) even transmitter parity
ARU_PARITY_NONE	(2) transmitter parity disabled

If the requested item is ARU_RECV_MODE (5), this routine returns the current state of the specified receive channel configuration register Internal Wrap Enable field:

AR_WRAP_ON	(0) internal wrap reception enabled
AR_WRAP_OFF	(1) internal wrap reception disabled

If the requested item is ARU_RX_MERGED_MODE (18), this routine returns the current state of the specified receive channel configuration register Merge Mode enable field:

AR_ON	(7) Merged Mode enabled
AR_OFF	(8) Merged Mode disabled

If the requested item is ARU_TX_DISABLE (10), this routine returns the current state of the specified transmit channel configuration register Transmit Disable field:

AR_ON	(7) external transmission disabled
AR_OFF	(8) external transmission enabled

If the requested item is ARU_TX_BIT_ERROR (6), this routine returns the current state of the specified transmitter channel configuration register Bit Count Hi and Low fields:

AR_LO	(0) Bit Count Low enabled
AR_HI	(1) Bit Count High enabled
AR_OFF	(8) both Bit Count Low and High disabled

If the requested item is ARU_TX_GAP_ERROR (8), this routine returns the current state of the specified transmitter channel configuration register Gap Error field:

AR_ON	(7) Gap Error enabled
AR_OFF	(8) Gap Error disabled

If the requested item is ARU_FAST_SLEW_RATE (323), this routine returns the current state of the specified transmitter channel configuration register Slew Rate field:

AR_ON	(7) Fast Slew Rate selected (1.5 µsec rise time)
AR_OFF	(8) Slow Slew Rate selected (10 µsec rise time)

If the requested item is ARU_ACCESS_SNAPSHOT_BUFFER (38), this routine returns the current state of the API snapshot storage mode:

ARU_LABEL_ONLY	(0) message storage on a label basis
ARU_LABEL_WITH_SDI	(1) message storage on a combined label/SDI basis.

If the requested item is ARU_IRIG_AVAILALBE (445), this routine returns TRUE if IRIG-B support is available, FALSE if it is not.

If the requested item is ARU_IRIG_INPUT_TIME (27), this routine returns the most recent received IRIG received sample value.

If the requested item is ARU_IRIG_WRAP_ENABLE (441), this routine returns the current state of the IRIG Receiver internal wrap feature:

AR_ON	(7) IRIG Receiver is patched into the IRIG Generator
AR_OFF	(8) IRIG Receiver is configured for external IRIG source

If the requested item is ARU_IRIG_CALIBRATED (447), this routine verifies the ability to capture consecutive IRIG time samples at a one second interval. If this results in a return status of FALSE (0), the IRIG signal is not consistent; otherwise, a return value of TRUE (1) indicates the signal is valid, or an error status (any value greater than 1) indicates a failure occurred.

If the requested item is ARU_DISCRETE_IN (14), this routine returns the current state of the specified discrete I/O channel:

AR_HI	(1) the discrete is High
AR_LO	(0) the discrete is Low

If the requested item is ARU_RX_TIMETAG_MODE (440), this routine returns a value representing the currently selected timer/time-tag source and resolution. This value indicates the resolution of any timer-read or receive data time-tag value obtained via the API, and is defined as follows:

AR_TIMETAG_EXT_IRIG_64BIT	(0)
AR_TIMETAG_INT_USEC_64BIT	(1)
AR_TIMETAG_INT_20USEC_32BIT	(3)
AR_TIMETAG_INT_MSEC_32BIT	(4)
AR_TIMER_X20_COMPAT_32BIT	(5)

A value of AR_TIMETAG_EXT_IRIG_64BIT indicates the source is the external IRIG receiver, if connected; otherwise, if the IRIG signal is not internally wrapped this selection would be invalid.

All other values represent various timer/time-tag LSB resolution values based on the internal RAR-USB device timer.

If the requested item is ARU_DEVICE_DISABLE (39), this routine returns the current value of the Global Status Register – Device Disabled bit.

If the requested item is ARU_CHAN_COUNT_429 (448), this routine returns the ARINC 429 transmit channel count detected on the board.

If the requested item is ARU_CHAN_COUNT_573 (449), this routine returns the ARINC 573/717 transmit channel count detected on the board.

If the requested item is ARU_CHAN_COUNT_ DISC (450), this routine returns the discrete output channel count detected on the board.

If the requested item is ARU_TX_FIFO_COUNT (19), this routine returns the current buffer count of messages in the specified ARINC 429 transmit FIFO awaiting transmission.

If the requested item is ARU_RX_FIFO_COUNT (28), this routine returns the current buffer count of messages in the ARINC 429 receive FIFO available to be read by the host application.

If the requested item is ARU_RX_MSG_COUNT (35), this routine returns the number of messages received on this channel since the board was last initialized.

If the requested item is ARU_TX_MSG_COUNT (36), this routine returns the number of messages transmitted on this channel since the board was last initialized.

If the requested item is ARU_FW_VERSION (20), this routine returns the current programmed firmware.

If the requested item is ARU_CONFIGURATION (21), this routine returns the Board Configuration value in the least significant 8 bits, defined as follows:

RAR-USB-1605-W	0
RAR-USB-0805-W	1
RAR-USB-0404-W	2
RAR-USB-0202-W	3
RAR-USB-0402-W	4
RAR-USB-1601-W	5
RAR-USB-0505-W	6
RAR-USB-1504J-W	7
RAR-USB-0804J-W	8
RAR-USB-0404J-W	9
RAR-USB-0202J-W	10

In addition to the value returned above, bit 8 will be set for any ARINC 717 configuration for compatibility with legacy ARINC API usage.

If the requested item is ARU_TX_PLAYBACK_ENABLE (5018), this routine returns the current state of the Transmit Playback feature:

AR_ON	(7) Transmit Playback is Enabled
AR_OFF	(8) Transmit Playback is Disabled

If the requested item is ARU_TX_PLAYBACK_COUNT (5021), this routine returns the current number of Transmit Playback message entries processed since the last Transmit Playback transition from disabled to the enabled state.

If the requested item is ARU_TX_PLAYBACK_INVALID_COUNT (5022), this routine returns the current number of Transmit Playback message entries processed containing an invalid transmit channel reference, since the last Transmit Playback transition from disabled to the enabled state..

If the requested item is ARU_TX_PLAYBACK_ACTIVE_STATUS (5023), this routine returns an indication of the state of Transmit Playback on the device, defined as follows:

TRUE	(1) Transmit Playback is actively processing entries
FALSE	(0) Transmit Playback is not actively processing entries

AR_GET_573_CONFIG

Syntax

CEI_INT16 ar_get_573_config (CEI_INT16 board, CEI_INT16 item, pCEI_INT32 value)

Description

This routine returns the state of the device configuration register attribute based on the combined item/value. It is designed to support the ARINC 573/717 configuration register attributes available to the device.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_INVARG	The *item* argument value is not supported by this API routine.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board — (input) Device this routine is to access. Valid range is 0-127.

CEI_INT16 item — (input) Configuration item about which to return information:

ARU_RECV_MODE	receiver internal wrap.
ARU_RX_BITRATE	receive channel bit rate.
ARU_RX_FIFO_ENABLE	receive channel buffer enable.
ARU_RX_MERGED_MODE	receiver merged mode enable
ARU_573_RX_AUTO_DETECT	receiver frame auto-detect enable.
ARU_573_RX_BPRZ_SELECT	receiver BPRZ/HBP selection.
ARU_TX_BITRATE	transmit channel bit rate.
ARU_TX_FIFO_ENABLE	transmit channel buffer enable.
ARU_TX_DISABLE	transmit channel external disable.
ARU_573_TX_BPRZ_SELECT	transmitter BPRZ encoder enable.
ARU_573_TX_HBP_SELECT	transmitter HBP encoder enable.
ARU_573_TX_SLEW_RATE	transmitter slew rate select.
ARU_TX_FIFO_COUNT	transmit FIFO buffer fill count.
ARU_573_SYNC_WORD1	receiver auto-detect sync word 1.
ARU_573_SYNC_WORD2	receiver auto-detect sync word 2.
ARU_573_SYNC_WORD3	receiver auto-detect sync word 3.
ARU_573_SYNC_WORD4	receiver auto-detect sync word 4.

pCEI_INT32 value	(output) The address that receives the state of the item requested:

If the requested item is ARU_RECV_MODE (5), this routine returns the current state of the ARINC 573/717 receive channel Internal Wrap Enable:

AR_WRAP_ON	(0) internal wrap reception enabled
AR_WRAP_OFF	(1) internal wrap reception disabled

If the requested item is ARU_RX_FIFO_ENABLE (16) or ARU_TX_FIFO_ENABLE (17), then this routine will return the current value of the ARINC 573/717 channel FIFO Enable:

AR_ON	(7) FIFO operation enabled
AR_OFF	(8) FIFO operation disabled

If the requested item is ARU_TX_DISABLE (10), this routine returns the current state of the ARINC 717 transmit channel configuration register Transmit Disable field:

AR_ON	(7) external transmission disabled
AR_OFF	(8) external transmission enabled

If the requested item is ARU_RX_MERGED_MODE (18), this routine returns the current state of the ARINC 573/717 receive channel Merge Mode Enable:

AR_ON	(7) Merge mode enabled
AR_OFF	(8) Merge mode disabled

If the requested item is ARU_573_RX_AUTO_DETECT (301), this routine returns the current state of the ARINC 573/717 receive channel Auto-synchronization Enable:

AR_ON	(7) ARINC 573/717 frame auto-detection enabled
AR_OFF	(8) ARINC 573/717 frame auto-detection disabled

If the requested item is ARU_573_RX_BPRZ_SELECT (302), this routine returns the current state of the ARINC 573/717 receive channel Encoding Enable:

AR_ON	(7) ARINC 573/717 BPRZ encoding enabled
AR_OFF	(8) ARINC 573/717 HBP encoding enabled

If the requested item is ARU_RX_BITRATE (1) or ARU_TX_BITRATE (2), this routine returns the current state of the ARINC 573/717 channel Baud Rate/Subframe Size selection (ranging from 0 to 7):

ARU_573_RATE_SIZE_384_32	384 bps, 32 word sub-frame
ARU_573_RATE_SIZE_768_64	768 bps, 64 word sub-frame
ARU_573_RATE_SIZE_1536_128	1536 bps, 128 word sub-frame
ARU_573_RATE_SIZE_3072_256	3072 bps, 256 word sub-frame
ARU_573_RATE_SIZE_6144_512	6144 bps, 512 word sub-frame

ARU_573_RATE_SIZE_12288_1024	12288 bps, 1024 word sub-frame
ARU_573_RATE_SIZE_24576_2048	24576 bps, 2048 word sub-frame
ARU_573_RATE_SIZE_49152_4096	49152 bps, 4096 word sub-frame

If the requested item is ARU_573_TX_BPRZ_SELECT (313), this routine returns the current state of the ARINC 573/717 transmit channel Encoding Enable:

AR_ON	(7) ARINC 573/717 BPRZ encoding enabled
AR_OFF	(8) ARINC 573/717 BPRZ encoding disabled

If the requested item is ARU_573_TX_HBP_SELECT (314), this routine returns the current state of the ARINC 573/717 transmit channel Encoding Enable:

AR_ON	(7) ARINC 573/717 HBP encoding enabled
AR_OFF	(8) ARINC 573/717 HBP encoding disabled

If the requested item is ARU_573_TX_SLEW_RATE (305) this routine returns the current state of the ARINC 573/717 transmit channel Slew Rate selection:

ARU_573_TX_SLEW_1PT5	(1) 1.5µsec rise time
ARU_573_TX_SLEW_10PT0	(0) 10.0µsec rise time

If the requested item is ARU_TX_FIFO_COUNT (19), this routine returns the current buffer count of messages in the ARINC 717 transmit FIFO awaiting transmission.

If the requested item is ARU_573_SYNC_WORD1 (307), ARU_573_SYNC_WORD2 (308), ARU_573_SYNC_WORD3 (309), or ARU_573_SYNC_WORD4 (310), this routine returns the 12-bit value for the respective receiver sub-frame sync word.

AR_GET_ERROR

Syntax

pCEI_CHAR ar_get_error (CEI_INT16 status)

Description

Most of the API routines return status values, a majority of which indicate an error condition. When supplied with such an error value, this routine returns a pointer to a message string describing the error.

Review the section, “Return Status Values”, for the current list of possible error codes and their explanations.

Return Value

A pointer to the error message character string. An allocation of at least 40 characters should be provided when copying the character string.

Arguments

CEI_INT16 status	(input) a status value returned by any of the API utilities.

AR_GETFILTER

Syntax

CEI_INT32 ar_getfilter (CEI_UINT32 board, CEI_UINT32 channel, pCEI_CHAR filterTable)

Description

This routine returns the contents of a single channel label filter table from the device. Each receive channel has a separate section within the label filter table, used by the firmware to control storage of received labels in the device Merged Receive Buffer. Each element of the filter table consists of a bit field defined for compatibility with the CEI-x20 product line as follows:

FILTER_SEQUENTIAL 0x10 If CLEAR add label to Sequential receive buffer

The filter buffer for a single channel is defined as follows:

filterTable[MAX_ESSM][MAX_SDI][MAX_LABEL]

and accessed as:

filterTable[eSSM][SDI][label]

where the bits of the ARINC word are split up as follows:

eSSM	SDI	label
30, 29, 28	9, 8	7, 6, 5, 4, 3, 2, 1, 0

To modify entries in the label filter table, refer to the API routines AR_ENH_LABEL_FILTER, AR_PUTFILTER, and AR_LABEL_FILTER.

Return Value

ARS_NORMAL Routine execution was successful.

ARS_INVARG Invalid *channel* or *filterTable* parameter.

ARS_INVBOARD An invalid *board* value was provided.

ARS_FAILURE The specified device has not been initialized.

ARS_LOCK_ACCESS_TIMEOUT The shared data structure access lock was not acquired.

ARS_RWR_INSERT_REQ_FAIL The specified device failed a communications packet initialization request.

ARS_RWR_EXECUTE_FAIL The specified device failed a communications block execution request.

Arguments

CEI_UINT32 board	(input) Device to access. Valid range is 0-127.
CEI_UINT32 channel	(input) Specifies which receive channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
pCEI_CHAR filterTable	(input) Array to receive the contents of the specified channel's label filter table. This array must have a minimum allocation of 8Kbytes.

AR_GET_IRIG_TIME_SET

Syntax

CEI_INT16 ar_get_irig_time_set (CEI_INT16 board, PTR_TIME_TAG_TYPE irigSampleTime, PTR_TIME_TAG_TYPE irigTimeStamp, PTR_TIME_TAG_TYPE boardTime)

Description

When measuring the signal presence and accuracy of the IRIG sample, comparing the board time and IRIG sample time provides a convenient method to validate IRIG signal integrity.

This routine returns the current IRIG Sample Time 64-bit one microsecond correlated value, the associated IRIG sample 64-bit time-stamp, and the current value of the device's internal 64-bit timer, all captured from the same host interface snapshot uploaded from the device.

Return Value

ARS_NORMAL Routine execution was successful.

ARS_INVBOARD An invalid *board* value was provided.

ARS_FAILURE The specified device has not been initialized.

ARS_LOCK_ACCESS_TIMEOUT The shared data structure access lock was not acquired.

ARS_RWR_INSERT_REQ_FAIL The specified device failed a communications packet initialization request.

ARS_RWR_EXECUTE_FAIL The specified device failed a communications block execution request.

Arguments

CEI_INT16 board (input) Device to access. Valid range is 0-127.

PTR_TIME_TAG_TYPE irigSampleTime (output) 64-bit IRIG Sample Time-of-Year as a 64-bit, one-microsecond correlated value.

PTR_TIME_TAG_TYPE irigTimeStamp (output) IRIG sample received time-stamp, 64-bit, one-microsecond resolution.

PTR_TIME_TAG_TYPE boardTime (output) Current board timer value, 64-bit, one-microsecond resolution.

AR_GET_LABEL_FILTER

Syntax

CEI_INT16 ar_get_label_filter (CEI_INT16 board, CEI_UINT16 label)

Description

This routine returns the active state of label filtering for the specified label value on all channels, with a bitwise indication for each of the installed receive channels.

Return Value

Given the routine is supplied with a valid board and label value, the return value indicates the active state of label filtering for the specified label on each receive channel. The indication is provided via bitwise state, where the label filter state on receive channel zero (zero-referenced) is indicated via b0 as "1" to indicate the label is filtered and "0" to indicate either the label is not filtered or the receive channel is not installed. Subsequent bits in the value indicate the label filter state for the respective receive channel.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_UINT16 label	(input) Specifies which label to query. Valid range is 0 to 255.

AR_GET_LATEST

Syntax

CEI_VOID ar_get_latest (CEI_INT16 board, CEI_INT16 channel, CEI_UINT16 label, pCEI_VOID data, pCEI_CHAR seq_num)

Description

In support of backward compatibility to previous ARINC product APIs, this routine returns the latest ARINC 429 message received for the specified channel/label combination from the AR-STREAM API's simulated snapshot buffer. Invocation of this routine is only valid when the API Receive Message Buffering Mode is *buffered*/*individual*, and will return an error if the Buffering Mode is *merged*.

If the *label* parameter value requested is either 256 or the value ARU_ALL_LABELS (511), this routine treats the *data* parameter as an array reference and returns the most recent received ARINC message for all 256 valid ARINC labels for the specified channel, in successive *data* array elements. This function assumes that the caller has allocated at least 1024 bytes for *data* when used in this mode.

If no message has been received for the specified channel/label since the last initialization of the device, a data value of zero will be returned.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which receive channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
CEI_UINT16 label	(input) The label value of interest.
pCEI_VOID data	(output) Location to store the 32-bit ARINC 429 message.
pCEI_CHAR seq_num	(output) Unsupported legacy parameter.

AR_GET_LATEST_T

Syntax

CEI_INT32 ar_get_latest_t (CEI_INT16 board, CEI_INT16 channel, CEI_UINT16 label, pCEI_UINT32 data, TIME_TAG_TYPE * timeTag)

Description

This routine returns the latest ARINC 429 message and time-stamp received for the specified channel/label combination from the AR-STREAM API's simulated snapshot buffer. Invocation of this routine is only valid when the API Receive Message Buffering Mode is *buffered*/*individual*, and will return an error if the Buffering Mode is *merged*.

If no message has been received for the specified channel/label since the last initialization of the device, a data value of zero is returned for the message and time-stamp. If the timeTag parameter is NULL, no time-stamp information is returned.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	Invalid *label* or null *data* parameter.
ARS_INVHARVAL	Channel is not available on device.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which receive channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
CEI_UINT16 label	(input) The label value of interest.
pCEI_UINT32 data	(output) Location to store the ARINC 429 message.
TIME_TAG_TYPE * timeTag	(output) The address that is to receive the 64-bit message time-stamp, the format of

which is determined by the current API time-tag format.

AR_GETNEXT

Syntax

CEI_INT16 ar_getnext (CEI_INT16 board, CEI_INT16 channel, pCEI_VOID destination)

Description

This routine retrieves the next unread message from the specified receive channel. If no message is present in the receiver FIFO buffer upon invocation, this routine polls the buffer waiting for the presence of a received message for up to one-half second. If no message is present after one-half second, a time-out status is returned.

Return Value

ARS_GOTDATA	A message has been retrieved.
ARS_CHAN_TIMEOUT	No message was available or received.
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received on the specified channel.
ARS_INVARG	Invalid *destination* parameter.
ARS_INVHARVAL	Channel is not available on device.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
pCEI_VOID destination	(output) The address that is to receive the message.

AR_GETNEXTT

Syntax

CEI_INT16 ar_getnextt (CEI_INT16 board, CEI_INT16 channel, pCEI_VOID destination, pCEI_VOID timetag)

Description

This routine retrieves the next unread message and scaled 32-bit time-stamp from the specified receive channel. If no message is present in the receiver FIFO buffer when invoked this routine polls the buffer waiting for the presence of a received message for up to one-half second. If no message is present after one-half second, a time-out status is returned.

If the *timetag* parameter is not NULL, the 32-bit translation of the 64-bit message time-stamp will be returned, scaled to the active legacy 32-bit time-tag mode, (1 millisecond resolution by default).

Return Value

ARS_GOTDATA	A message has been retrieved.
ARS_CHAN_TIMEOUT	No message was available or received.
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received.
ARS_INVARG	Invalid *destination* parameter.
ARS_INVHARVAL	Channel is not available on device.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
pCEI_VOID destination	(output) The address that is to receive the message.
pCEI_VOID timetag	(output) The address that is to receive the 32-bit time-tag associated with the data,

(resolution is programmable). If the merged receive mode is active for the specified channel, the upper five bits of the 32-bit time-tag word contain the receive channel number on which the data was received.

AR_GETNEXT_XT

Syntax

CEI_INT16 ar_getnext_xt (CEI_INT16 board, CEI_INT16 channel, pCEI_UINT32 data, pAR_TIMETAG_TYPE timeTagRef)

Description

This routine retrieves the next unread message and the associated time-tag structure from the specified receive channel. If no message is present in the receiver FIFO buffer when invoked, this routine polls the buffer waiting for the presence of a received message for up to one-half second. If no message is present after one-half second, a time-out status is returned. The time-tag structure is required to be defined by the host application with this routine.

Return Value

ARS_GOTDATA	A message has been retrieved.
ARS_CHAN_TIMEOUT	No message was available or received.
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received.
ARS_INVARG	Invalid *data* parameter.
ARS_INVHARVAL	Channel is not available on device.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.

pCEI_UINT32	data	(output) The address that is to receive the message.
pAR_TIMETAG_TYPE	timeTagRef	(output) The address that is to receive the time-tag data structure associated with the message.

AR_GET_RX_CHANNEL_STATUS

Syntax

CEI_UINT32 ar_get_rx_channel_status (CEI_INT16 board, CEI_INT16 channel, pCEI_INT32 channelStatus, pCEI_INT32 messageCount)

Description

This routine returns the current status of the specified receive channel buffer. If either an ARINC 429 protocol error or buffer overflow bit was set in the receive channel buffer status register, it is cleared on return from this routine.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An uninitialized board or invalid *board* value was provided.
ARS_INVARG	Invalid *channelStatus* or *messageCount* parameter.
ARS_INVHARVAL	ARINC 429 channel is not available on the device.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
pCEI_INT32 channelStatus	(output) Location to store the bitwise representation of the current receiver buffer status bits. The Status Register Bit assignments are defined as follows:

b0 - AR_BUFFER_MSG_AVAILABLE (1)

Set indicates at least one message is ready to read from the buffer. Clear indicates the buffer is empty.

b1 - AR_INVALID_MSG_DETECTED (2)

Set indicates at least one ARINC 429 message protocol error was detected since either this routine was previously invoked or a message was last retrieved from the buffer. Clear indicates no protocol error has been encountered.

b2 - AR_BUFFER_OVERFLOW_DETECTED (4)

Set indicates the respective receive channel encountered a message buffer overflow since this routine was previously invoked. Clear indicates no buffer overflow has been encountered.

pCEI_INT32 messageCount — (output) Location to store the number of messages currently available in the respective receive buffer, acquired from the respective receive channel status register. This value will only be valid if b0 in the *channelStatus* return value is set, and has a valid range of 1 - 2047.

AR_GET_RX_COUNT

Syntax

CEI_UINT32 ar_get_rx_count (CEI_INT16 board, CEI_INT16 channel)

Description

This routine returns a count of the number of ARINC 429 messages received on each channel since the device was last initialized (see AR_LOADSLV). Both tracking ARINC 429 message count and invocation of this routine are only valid when the API Receive Message Buffering Mode is *buffered*/*individual*, and will return an error if the Buffering Mode is *merged*.

If the API routine AR_CLR_RX_COUNT has been invoked by the host application prior to this routine's invocation, the API will reset the message count for the respective receive channel.

Return Value

Current count of ARINC 429 messages received on the specified channel.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which receive channel this routine is to access.

AR_GET_SNAP_DATA

Syntax

CEI_INT32 ar_get_snap_data (CEI_INT16 board, CEI_INT16 channel, CEI_UINT16 label, CEI_UINT16 sdi, pCEI_UINT32 data)

Description

This routine returns the latest ARINC 429 message received for the specified channel/label combination from the AR-STREAM API's simulated snapshot buffer. Invocation of this routine is only valid when the API Receive Message Buffering Mode is *buffered*/*individual*, and will return an error if the Buffering Mode is *merged*.

If no message has been received for the specified channel/label since the last initialization of the device, a data value of zero is returned for the message.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	Invalid *label, sdi,*or null *data* parameter.
ARS_INVHARVAL	Channel is not available on device.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized or the AR-STREAM API Receive Message Buffering Mode is defined as *merged*.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which receive channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
CEI_UINT16 label	(input) The label value of interest.
CEI_UINT16 sdi	(input) The SDI value of interest.
pCEI_UINT32 data	(output) Location to store 32-bit ARINC429 message.

AR_GET_SNAP_DATA_T

Syntax

CEI_INT32 ar_get_snap_data_t (CEI_INT16 board, CEI_INT16 channel, CEI_UINT16 label, CEI_UINT16 sdi, pCEI_UINT32 data, PTR_TIME_TAG_TYPE timeTag)

Description

This routine returns the latest ARINC 429 message and its associated time-stamp, received for the specified channel/label combination from the AR-STREAM API's simulated snapshot buffer. Invocation of this routine is only valid when the API Receive Message Buffering Mode is *buffered*/*individual*, and will return an error if the Buffering Mode is *merged*.

If no message has been received for the specified channel/label since the last initialization of the device, a data value of zero is returned for the message.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	Invalid *label*, *sdi*,or null *data* parameter.
ARS_INVHARVAL	Channel is not available on device.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized or the AR-STREAM API Receive Message Buffering Mode is defined as *merged*.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which receive channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
CEI_UINT16 label	(input) The label value of interest.
CEI_UINT16 sdi	(input) The SDI value of interest.

pCEI_UINT32 data	(output) Location to store 32-bit ARINC429 message.
PTR_TIME_TAG_TYPE timeTag	(output) The address that is to receive the 64-bit one microsecond resolution time-tag associated with the ARINC 429 message.

AR_GET_STATUS

Syntax

CEI_UINT32 ar_get_status (CEI_INT16 board, pCEI_UINT16 state)

Description

This routine returns the state of the FIFO Data Available bit for up to 16 receivers in a bitwise 16-bit value. Invocation of this routine is only valid when the API Receive Message Buffering Mode is *buffered*/*individual*, and will return a value of zero if the Buffering Mode is *merged*.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An uninitialized board or invalid *board* value was provided .

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
pCEI_UINT16 state	(output) Location to store the receiver FIFO status. The Status Register Bit Assignments are defined as follows, ("1" indicates Data Available, “0” indicates No Data Available):

b0 - ARINC 429 Receiver 1

b1 - ARINC 429 Receiver 2

…

b14 - ARINC 429 Receiver 15

b15 - ARINC 429 Receiver 16

AR_GET_STORAGE_MODE

Syntax

CEI_INT16 ar_get_storage_mode (CEI_INT16 board, pCEI_INT16 mode)

Description

This routine is designed to provide compatibility with the legacy ARINC device APIs. It returns the current state of the AR-STREAM API Receive Message Buffering Mode. When the API Receive Message Buffering Mode is *buffered*, each receiver is assigned an independent circular buffer for storage of received messages. When the buffering mode is *merged*, all messages remain in a single merged buffer as uploaded from the RAR-USB device. Each receive data API routine processes the active storage mode internally, acquiring data from the appropriate buffer. This routine should be used in conjunction with the AR_SET_STORAGE_MODE routine.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An uninitialized board or invalid *board* value was provided.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
pCEI_INT16 mode	(output) The address that is to receive the state of the current API storage mode. Valid return values for this parameter are:

ARU_BUFFERED	(0)	buffered receive mode
ARU_MERGED	(2)	merged receive mode

AR_GET_TIME

Syntax

CEI_INT16 ar_get_time (CEI_INT16 board, CEI_INT16 format, pAR_TIMETAG_TYPE timeTag)

Description

This routine returns the current time/timer value scaled from either the RAR-USB device internal 64-bit timer or the most recently received IRIG timer reference, as specified via the *format* parameter. If the requested timer reference is IRIG, the return structure will also contain the IRIG Sample timer-referenced time-stamp assigned when the sample was received by the device.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	An invalid format parameter value was provided.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board — (input) Device to access. Valid range is 0-127.

CEI_INT16 format — (input) Time format requested. Valid options are:

AR_TIMETAG_EXT_IRIG_64BIT	0
AR_TIMETAG_INT_USEC_64BIT	1
AR_TIMETAG_HOST_USEC_64BIT	2
AR_TIMETAG_INT_20USEC_32BIT	3
AR_TIMETAG_INT_MSEC_32BIT	4
AR_TIMER_X20_COMPAT_32BIT	6

pAR_TIMETAG_TYPE timeTag

(output) Current device timer or translated IRIG sample value. The *timeTag.timeTag* structure member will be defined as follows

based on the supplied *format* parameter value:

AR_TIMETAG_EXT_IRIG_64BIT - 64-bit IRIG sample time in microseconds since beginning of current year. The returned *timeTag.R.referenceTimeTag* structure member will contain the board internal timer-referenced time-stamp assigned when the last bit of the IRIG sample was processed by the RAR-USB IRIG receiver.

AR_TIMETAG_INT_USEC_64BIT - 64-bit internal board timer in microseconds.

AR_TIMETAG_HOST_USEC_64BIT - 64-bit host operating system time scaled to have a 1 microsecond resolution.

AR_TIMETAG_INT_20USEC_32BIT - 32-bit internal board timer in microseconds.

AR_TIMETAG_INT_MSEC_32BIT - 32-bit internal board timer scaled to have a 20 microsecond resolution.

AR_TIMER_X20_COMPAT_32BIT - 32-bit internal board timer scaled to have a 1 millisecond resolution.

AR_GET_TIMERCNTL

Syntax

CEI_UINT32 ar_get_timercntl (CEI_INT16 board)

Description

This routine is provided for legacy support of the CEI-x20 ARINC API, returning the current 32-bit, 1 millisecond resolution time reference value based on the current application-specified timer mode, (specified through AR_SET_CONFIG using the attribute ARU_RX_TIMETAG_MODE). If the current timer mode is assigned to any 64-bit timer, the least-significant 32-bits of the internal device timer will be returned (this applies to IRIG, host, or internal timer). If the current timer mode is assigned to either of the 32-bit CEI-x20 API compatibility or 20 microsecond (IP-AVIONICS) resolution modes, the respective 32-bit adjusted timer value will be returned.

Return Value

The 32-bit timer value.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_GETWORD

Syntax

CEI_INT16 ar_getword (CEI_INT16 board, CEI_INT16 channel, pCEI_VOID destination)

Description

This routine retrieves the next unread message from the specified receive channel. If it successfully returns data message, there may or may not be more messages in the buffer. It only means there was at least one message in the buffer. Subsequent calls would be required to determine if more messages are available in the buffer. If this routine returns a status value of ARS_NODATA, the buffer is empty.

The *channel* value passed to this routine corresponds to the ARINC 429 receive channel index, starting with zero. If the *channel* value is set to 32 and an ARINC 573/717 receiver exists, it is used as the designated receive channel buffer.

Return Value

ARS_GOTDATA	A message has been retrieved.
ARS_NODATA	No data available.
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received on the specified channel.
ARS_INVARG	A null *destination* parameter value was provided.
ARS_INVHARVAL	Channel is not available on device.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which receive channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
pCEI_VOID destination	(output) The address that is to receive the message. The format of the message value is dependent on the protocol assigned to the respective receive channel (see Receive Message Buffering).

AR_GETWORDT

Syntax

CEI_INT16 ar_getwordt (CEI_INT16 board, CEI_INT16 channel, pCEI_VOID destination, pCEI_VOID timetag)

Description

This routine retrieves the next unread message from either the Merged Receive Buffer or the API individual receive buffer for the specified ARINC 429 receive channel. If it successfully returns data message, there may or may not be more messages data in the buffer. It means only that there was at least one message in the buffer. Subsequent calls are required to determine if more messages are available in the buffer. If this routine returns a status value of ARS_NODATA, the buffer is empty.

Return Value

ARS_GOTDATA	A message has been retrieved.
ARS_NODATA	No data available.
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received on the specified channel.
ARS_INVARG	A null *data* parameter value was provided.
ARS_INVHARVAL	Channel is not available on device.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
pCEI_VOID destination	(output) The address that is to receive the retrieved ARINC 429 message.

pCEI_VOID timetag	(output) The address that is to receive the 32-bit time-tag associated with the data, (resolution is programmable). If the merged receive mode is active for the specified channel, the upper five bits of the 32-bit time-tag word contain the receive channel number on which the data was received.

AR_GETWORD_XT

Syntax

CEI_INT16 ar_getword_xt (CEI_INT16 board, CEI_INT16 channel, pCEI_VOID data, pAR_TIMETAG_TYPE timeTagRef)

Description

This routine retrieves the next unread ARINC 429 message and the associated time-tag structure from the specified receive channel. If it successfully returns a message, there may or may not be more messages in the buffer. It means only that there was at least one message in the buffer. Subsequent calls are required to determine if more messages are available in the buffer. If this routine returns a status value of ARS_NODATA, the buffer is empty. The time-tag structure is required to be defined by the host application with this routine.

Return Value

ARS_GOTDATA	A message has been retrieved.
ARS_NODATA	No data available.
ARS_BAD_MESSAGE	An invalid length ARINC 429 message was received on the specified channel.
ARS_INVARG	A null *data* or *timeTagRef* parameter value was provided.
ARS_INVHARVAL	Channel is not available on device.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device this routine is to access. Valid range is 0-127.
CEI_INT16 channel	(input) Specifies which channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
pCEI_VOID data	(output) The address that is to receive the 32-bit ARINC 429 message.
pAR_TIMETAG_TYPE timeTagRef	(output)

The address that is to receive the time-tag data structure associated with the message.

AR_GO

Syntax

CEI_INT16 ar_go (CEI_INT16 board)

Description

This routine enables the message scheduler and all receive message processing on the RAR-USB device.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_HAS_ERROR_OCCURRED

Syntax

CEI_INT16 ar_has_error_occurred (CEI_INT16 board, CEI_INT16 statusCode, pCEI_INT16 state)

Description

Some legacy ARINC API routines return valid bitwise and/or numeric values that may conflict with the values of error codes returned when error conditions are encountered. This routine provides the method to verify if an error condition was actually encountered when the return status from one of these routines matches the value of a valid error code.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	The *statusCode* parameter value is not a valid AR-STREAM API error code or the state parameter is NULL.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 statusCode	(input) Any valid error code returned by an AR-STREAM API routine.
pCEI_INT16 state	(output) Indicates if the specified statusCode value has been returned by any API function since the last initialization/reset of the specified board. TRUE if the supplied statusCode value has been encountered and returned FALSE if the supplied statusCode value has not been encountered.

AR_INITIALIZE_API

Syntax

CEI_INT16 ar_initialize_api (CEI_INT16 board)

Description

This routine is provided for backward compatibility with other ARINC 429 API libraries and should not be invoked prior to AR_LOADSLV. It initializes the API and device host interface to an initial state.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_INITIALIZE_DEVICE

Syntax

CEI_INT16 ar_initialize_device (CEI_INT16 board)

Description

This routine is provided for backward compatibility with other ARINC 429 API libraries and should not be invoked prior to AR_LOADSLV. It initializes the API and device host interface to an initial state. The default setup of the device following execution of this routine is:

- ARINC 429 Transmitter FIFOs disabled, speed set for 100Kbps with the fast slew rate and ODD parity enabled.
- ARINC 429 Receiver FIFOs disabled, speed set for 100Kbps, parity checking enabled, and internal wrap disabled.
- ARINC 717 Transmitter and Receiver FIFOs disabled, set for HBP encoding at 384BPS (32-word subframe), auto-detect enabled, and Internal Wrap disabled.
- Message Scheduler disabled, no messages defined.
- Transmit Playback disabled and playback buffer empty.
- All receive label filtering disabled.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_LABEL_FILTER

Syntax

CEI_INT16 ar_label_filter (CEI_INT16 board, CEI_INT16 channel, CEI_UINT16 label, CEI_INT16 action)

Description

RAR-USB devices support the ability to filter ARINC 429 messages by the 8-bit label value. Once filtering has been enabled for a specified channel/label combination, data received with that label value would be discarded until label filtering for the specified label has been disabled. Label filtering is disabled for all labels by default.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	An invalid *label* or *action* value was provided.
ARS_INHARVAL	The specified *channel* does not support label filtering.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_FAILED	Either the acquisition or relinquish of the access lock to the API shared device interface data structure failed.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 channel	(input) channel label filter table this routine is to access. The valid range is 0 to one less than the installed receive channel count.
CEI_UINT16 label	(input) The label of interest. Valid range is 0-255. Also valid is ARU_ALL_LABELS (511), which invokes the action for all labels on the specified channel.
CEI_INT16 action	(input) Enable or disable filtering for this combination of board/channel/label. Valid values are:

ARU_FILTER_ON (1) or FILTER_SEQUENTIAL (0x10) enables filtering on this label

ARU_FILTER_OFF (0) disables filtering (default state is to not filter any labels).

AR_LOADSLV

Syntax

CEI_INT16 ar_loadslv (CEI_INT16 board, CEI_UINT32 base_seg, CEI_INT32 base_port, CEI_UINT16 ram_size)

Description

This routine opens a session with the specified RAR-USB device and initializes the API. The default setup of the API and the device following execution of this routine is:

- ARINC 429 Transmitter FIFOs disabled, speed set for 100Kbps with the fast slew rate and ODD parity enabled.
- ARINC 429 Receiver FIFOs disabled, speed set for 100Kbps, parity checking enabled, and internal wrap disabled.
- ARINC 717 Transmitter and Receiver FIFOs disabled, set for HBP encoding at 384BPS (32-word subframe), auto-detect enabled, and Internal Wrap disabled.
- Message Scheduler disabled, no messages defined.
- Transmit Playback disabled and playback buffer empty.
- All receive label filtering disabled.
- API simulated Receive Mode set to individual channel buffering

If any portion of the initialization fails or the device is not detected, a status other than ARS_NORMAL is returned. If the return status is ARS_DRIVERFAIL, an invocation of AR_GET_ERROR should supply an explanation of the installation error condition.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_DRIVERFAIL	The device driver failed to open a session with the device, either because the device is not properly installed in the host system or a resource conflict is inhibiting device driver initialization.
ARS_FAILURE	Failed to access the driver interface library.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_UINT32 base_seg	(input) This parameter is ignored, (supplied for ARINC API compatibility only).
CEI_INT32 base_port	(input) This parameter is ignored, (supplied for ARINC API compatibility only).
CEI_UINT16 ram_size	(input) This parameter is ignored, (supplied for ARINC API compatibility only).

AR_MODIFY_MSG

Syntax

CEI_INT16 ar_modify_msg (CEI_INT16 board, CEI_INT16 channel, CEI_INT16 msgNumber, CEI_INT16 rate, CEI_INT32 data)

Description

This routine modifies an existing 32-bit ARINC message for periodic retransmission, originally created through use of the AR_DEFINE_MSG or AR_DEFINE_MSG_BLOCK API routines.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_INVARG	An invalid *msgNumber* value was provided.
ARS_INVHARVAL	An invalid *channel* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 channel	(input) Channel message scheduler table this routine is to access. The valid range is 0 to one less than the number of installed transmit channels.
CEI_INT16 msgNumber	(input) The unique message scheduler table entry index assigned to this message, as returned for the respective message from the routine AR_DEFINE_MSG or AR_DEFINE_MSG_BLOCK.

CEI_INT16 rate	(input) Periodic transmission rate, defined in milliseconds. A rate value of zero will disable message transmission for this message scheduler table entry and make this entry available for reuse on the next invocation of AR_DEFINE_MSG or AR_DEFINE_MSG_BLOCK.
CEI_INT32 data	(input) The updated 32-bit ARINC message to transmit.

AR_MODIFY_MSG_BLOCK

Syntax

CEI_INT16 ar_modify_msg_block (CEI_INT32 numberOfEntries, pAR_SCHEDULED_MSG_ENTRY_TYPE messageEntry)

Description

This routine provides a method to modify the channel assignment or rate and data values on a series of 32-bit ARINC messages previously defined for periodic retransmission via AR_DEFINE_MSG_BLOCK.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* structure member value was provided.
ARS_INVARG	An invalid *messageIndex* structure member value was provided.
ARS_INVHARVAL	An invalid *channel* structure member value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT32 numberOfEntries (input) The number of entries to modify using the subsequent structure pointer parameter, messageEntry.

pAR_SCHEDULED_MSG_ENTRY_TYPE messageEntry (input)

array of structures of message definition content, defined as follows:

CEI_UINT32 messageIndex	The unique message scheduler table entry index assigned to this message. This messageIndex structure member will have been defined in a previous invocation of AR_DEFINE_MSG_BLOCK.
CEI_UINT32 board	Device to access. Valid range is 0-127.

CEI_UINT32 channel	Which channel portion of the message scheduler table this routine is to access. The valid range is 0 to one less than the number of installed transmit channels.
CEI_UINT32 rate	Periodic transmission rate, in milliseconds. A rate value of zero will disable message transmission.
CEI_UINT32 start	Not supported during message modification.
CEI_UINT32 txCount	Not supported during message modification.
CEI_UINT32 data	The 32-bit ARINC message to transmit.

AR_NUM_RCHANS

Syntax

CEI_INT16 ar_num_rchans (CEI_INT16 board)

Description

This routine retrieves the number of receive channels installed on the specified device.

Return Value

Any non-zero value	number of installed receive channels.
Zero	An uninitialized board or invalid *board* value was provided.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_NUM_XCHANS

Syntax

CEI_INT16 ar_num_xchans (CEI_INT16 board)

Description

This routine retrieves the number of transmit channels installed on the specified device.

Return Value

Any non-zero value	number of installed transmit channels.
Zero	An uninitialized board or invalid *board* value was provided.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_OPEN

Syntax

CEI_INT16 ar_open (CEI_INT16 board)

Description

This routine opens a session with the specified RAR-USB device and initializes the API. The default setup of the API and the device following execution of this routine is:

- ARINC 429 Transmitter FIFOs disabled, speed set for 100Kbps with the fast slew rate and ODD parity enabled.
- ARINC 429 Receiver FIFOs disabled, speed set for 100Kbps, parity checking enabled, and internal wrap disabled.
- ARINC 717 Transmitter and Receiver FIFOs disabled, set for HBP encoding at 384BPS (32-word sub-frame), auto-detect enabled, and Internal Wrap disabled.
- Message Scheduler disabled, no messages defined.
- Transmit Playback disabled and playback buffer empty.
- All receive label filtering disabled.
- API simulated Receive Mode set to individual channel buffering

If any portion of the initialization fails or the device is not detected, a status other than ARS_NORMAL is returned. If the return status is ARS_DRIVERFAIL, an invocation of AR_GET_ERROR should supply an explanation of the installation error condition.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_DRIVERFAIL	The device driver failed to open a session, either because the device is not properly installed in the host system or a resource conflict is inhibiting device driver initialization.
ARS_FAILURE	Failed to access the driver interface library.
ARS_RWR_INSERT_REQ_FAIL	The device failed a communications initialization request.
ARS_RWR_EXECUTE_FAIL	The device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_PUT_429_MESSAGE

Syntax

CEI_INT16 ar_put_429_message (CEI_INT16 board, CEI_INT16 channel, CEI_INT32 data)

Description

This routine places the supplied message data in the specified transmit channel burst transmit buffer. When this routine returns, the data has not necessarily been sent, it has only been placed in the respective transmit buffer. If messages are present in the respective device transmit buffer ahead of it, this data will be transmitted in turn. If the specified transmit buffer is full, an overflow status is returned.

Note: Since ARINC 429 transmit data rates are relatively slow, almost any host can generate transmit data at a much faster rate than data is transmitted.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_XMITOVRFLO	A transmit buffer overrun occurred.
ARS_INVHARVAL	An invalid *channel* value was provided.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 channel	(input) ARINC 429 transmit channel this routine is to access. The valid range is 0 to one less than the number of installed transmit channels.
CEI_INT32 data	(input) ARINC 429 message to transmit in standard ARINC 429 format.

AR_PUT_573_FRAME

Syntax

CEI_INT16 ar_put_573_frame (CEI_INT16 board, CEI_UINT32 numberWords, pCEI_UINT32 transmitCount, pCEI_INT16 arincData)

Description

This routine attempts to transfer *numberWords* of ARINC 573/717 data from the *arincData* source to the device ARINC 573/717 transmit buffer. The amount of data transferred to the transmitter is based on what is available in the buffer, with the actual number of words transferred indicated in the return value of *transmitCount*.

Note: Since ARINC 573/717 transmit data rates are relatively slow, almost any host can generate transmit frame data at a much faster rate than frame data is actually transmitted.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_XMITOVRFLO	A transmit buffer overrun occurred.
ARS_INVARG	Either of the *transmitCount* or *arincData* parameters were NULL.
ARS_INHARVAL	The specified *board* does not support the ARINC 573/717 protocol.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_UINT32 numberWords	(input) Number of words to copy from the source 573 frame to the transmit buffer.
pCEI_UINT32 transmitCount	(output) Indicates how many words were copied from the source 573 frame to the transmit buffer, either less than or equal to the value of *numberWords*.

pCEI_INT16 arincData — (output) Pointer to the array of ARINC 573/717 frame data. The format of each data word in the source ARINC 573/717 frame is defined as follows:

15 – 12	**11 - 0**
RESERVED	data

data: the 12-bit ARINC 573/717 data.

AR_PUTBLOCK

Syntax

CEI_INT32 ar_putblock (CEI_UINT32 board, CEI_UINT32 channel, CEI_INT32 maxMessages, CEI_INT32 offset, pCEI_INT32 data, pCEI_INT32 actualCount)

Description

This routine transfers the array of ARINC 429 messages to the specified transmit channel buffer. When this routine returns, the data has not been transmitted, it has only been placed in the transmit buffer. If other data is in the transmit buffer ahead of it, this data is transmitted in turn.

Note: Since ARINC 429 transmit data rates are relatively slow, almost any host can generate transmit data at a much faster rate than data is transmitted.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_XMITOVRFLO	A transmit buffer overrun occurred.
ARS_INVHARVAL	An invalid *channel* value was provided.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_UINT32 board	(input) Device to access. Valid range is 0-127.
CEI_UINT32 channel	(input) ARINC 429 transmit channel this routine is to access. The valid range is 0 to one less than the number of installed transmit channels.
CEI_INT32 maxMessages	(input) The number of messages to transmit.

CEI_INT32 offset	Unused legacy parameter.
pCEI_INT32 data	(input) Array supplying 32-bit ARINC data values.
pCEI_INT32 actualCount	(output) The number of messages copied to the transmit buffer.

AR_PUTBLOCK_MULTI_CHAN

Syntax

CEI_INT32 ar_putblock_multi_chan (CEI_UINT32 board, CEI_INT32 maxMessages, pCEI_UINT32 channels, pCEI_INT32 data, pCEI_INT32 actualCount)

Description

This routine transfers messages from the *data* array source to the channel transmit buffer corresponding to the respective transmit channel element of the *channels* array. When this routine returns, the data has not necessarily been transmitted, it has only been placed in the respective transmit buffer(s). If other data is in the transmit buffer ahead of it, this data will be transmitted in turn.

Note: Since ARINC 429 transmit data rates are relatively slow, almost any host can generate transmit data at a much faster rate than data is transmitted.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_XMITOVRFLO	A transmit buffer overrun occurred.
ARS_INVHARVAL	An invalid *channel* value was provided.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_UINT32 board	(input) Device to access. Valid range is 0-127.
CEI_INT32 maxMessages	(input) The number of messages to transmit.
pCEI_UINT32 channels	(input) Array supplying the ARINC 429 transmit channel on which this routine is to transmit the respective ARINC 429 data. The transmit channel index in each element of this array corresponds directly to the ARINC 429 message defined in the respective element of the *data* array. The valid range for each element of this array is 0 to one less than the number of installed transmit channels.
pCEI_INT32 data	(input) Array supplying 32-bit ARINC data values.
pCEI_INT32 actualCount	(input) The number of messages transmitted.

AR_PUTFILTER

Syntax

CEI_INT32 ar_putfilter (CEI_UINT32 board, CEI_UINT32 channel, pCEI_CHAR filterTable)

Description

This routine assigns an entire channel portion of the label filter table for the specified receive channel. Each receive channel has a separate area in the device label filter table, which is used by the firmware to control storage of received labels. Each element of the filter table consists of a single bit field defined for compatibility with legacy ARINC 429 product lines as follows:

FILTER_SEQUENTIAL 0x10 If CLEAR add label to circular receive buffer

The filter buffer for a single channel is defined as follows:

filterTable[MAX_ESSM][MAX_SDI][MAX_LABEL]

and accessed as follows in the array referenced by *filterTable*:

filterTable[eSSM][SDI][label]

where the bits of the ARINC 429 message are defined as follows:

eSSM	SDI	label
30, 29, 28	9, 8	7, 6, 5, 4, 3, 2, 1, 0

To write individual label filter table elements, refer to the API routines AR_ENH_LABEL_FILTER and AR_LABEL_FILTER.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVHARVAL	An invalid *channel* value was provided.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_UINT32 board	(input) Device to access. Valid range is 0-127.

CEI_UINT32 channel	(input) Specifies which receive channel this routine is to access. Valid range is 0 to one less than the installed receive channel count.
pCEI_CHAR filterTable	(input) Array containing the contents of the specified channel's label filter table. This array must have an allocation of 8Kbytes.

AR_PUTWORD

Syntax

CEI_INT16 ar_putword (CEI_INT16 board, CEI_INT16 channel, CEI_INT32 arincdata)

Description

This routine places the supplied message data in the specified transmit channel burst transmit buffer. When this routine returns, the data has not necessarily been sent, it has only been placed in the respective transmit buffer. If messages are present in the respective device transmit buffer ahead of it, this data will be transmitted in turn. If the specified transmit buffer is full, an overflow status is returned.

The channel value passed to this routine corresponds to the ARINC 429 transmit channel index, starting with zero. If that value is set to 32 and an ARINC 573/717 transmitter exists, it is used as the designated transmit channel buffer.

Note: Since ARINC transmit data rates are relatively slow, almost any host can generate transmit data at a much faster rate than data is transmitted.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_XMITOVRFLO	A transmit buffer overrun occurred.
ARS_INVHARVAL	An invalid *channel* value was provided.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 channel	(input) Transmit channel this routine is to access. The valid range is 0 to one less than the installed transmit channel count.
CEI_INT32 arincdata	(input) 32-bit ARINC 429 message to transmit.

AR_RESET

Syntax

CEI_INT16 ar_reset (CEI_INT16 board)

Description

This routine reinitializes the device to the same initial state as that following an invocation of AR_LOADSLV.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_RESET_TIMERCNT

Syntax

CEI_VOID ar_reset_timercnt (CEI_INT16 board)

Description

This routine is designed to provide compatibility with the legacy ARINC 429 product lines. It resets the RAR-USB device internal one-microsecond timer to zero.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_SET_CONFIG

Syntax

CEI_INT16 ar_set_config (CEI_INT16 board, CEI_INT16 item, CEI_UINT32 value)

Description

This routine provides a means to define general device configuration attributes, as well as limited individual channel configuration attributes. It is provided for backward compatibility to legacy ARINC 429 product based applications. AR_SET_DEVICE_CONFIG is the desired routine for defining channel and board-level configuration items.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	The *item* argument value is not supported by this API routine.
ARS_INVHARVAL	The *item* argument value is not supported by this device configuration.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 item	(input) Attribute about which to set information:

ARU_XMIT_RATE	transmit rate for all transmitters.
ARU_RECV_RATE	receive rate for all receivers.
ARU_PARITY	parity for all transmitters and receivers.
ARU_INTERNAL_WRAP	enables internal wrap mode for all receivers.
ARU_RX_CH01_BIT_RATE – ARU_RX_CH16_BIT_RATE	receiver 1 - 16 bit rate.

ARU_TX_CH01_BIT_RATE –
ARU_TX_CH05_BIT_RATE transmitter 1 - 5 bit rate.

ARU_RX_CH01_PARITY –
ARU_RX_CH16_PARITY receiver 1 - 16 parity.

ARU_TX_CH01_PARITY –
ARU_TX_CH05_PARITY transmitter 1 - 5 parity.

ARU_TX_CH01_SHUT_OFF –
ARU_TX_CH05_SHUT_OFF transmitter 1 - 5 disable.

ARU_TX_CH01_LB_INJ – transmitter 1 - 5 low bit
ARU_TX_CH05_LB_INJ error enable.

ARU_TX_CH01_HB_INJ – transmitter 1 - 5 high bit
ARU_TX_CH05_HB_INJ error enable.

ARU_TX_CH01_GAP_INJ – transmitter 1 - 5
ARU_TX_CH05_GAP_INJ message gap error enable.

ARU_RX_TIMETAG_MODE the timer/time-tag source and resolution

ARU_ACCESS_SNAPSHOT_BUFFER snapshot storage mode

ARU_IRIG_WRAP_ENABLE enables IRIG receiver internal wrap

ARU_IRIG_INPUT_THRESHOLD sets the IRIG DAC threshold

ARU_IRIG_ADJUST_THRESHOLD invokes an IRIG DAC auto-adjustment procedure

ARU_IRIG_QUICK_ADJUSTMENT invokes an IRIG DAC auto-adjustment procedure

ARU_IRIG_SET_BIAS assigns an offset to the board IRIG time value

CEI_UINT32 value (input) the value to set the item.

If the specified item is ARU_XMIT_RATE (1) or ARU_RECV_RATE (2), valid value parameter selections are:

AR_HIGH (0) high rate (100Kbs)
AR_LOW (1) low rate (12.5Kbs)
Any other value specifies a frequency value in Hertz.

If the specified item is ARU_RX_CH*nn*_BIT_RATE (500-515), where *nn* is the receiver channel (01 - 15), valid value parameter selections are:

AR_HIGH (0) high rate (100Kbs)

AR_LOW (1) low rate (12.5Kbs)
Any other value specifies a frequency value in Hertz.

If the specified item is ARU_TX_CH*nn*_BIT_RATE (700-704), where *nn* is the transmitter channel (01 - 05), valid value parameter selections are:

AR_HIGH (0) high rate (100Kbs)
AR_LOW (1) low rate (12.5Kbs)
Any other value specifies a frequency value in Hertz.

Note

Any specified transmit bus frequency below 15KHz will be assigned to a slow slew rate.
Any specified transmit bus frequency above 15KHz will be assigned to a fast slew rate.

If the specified item is ARU_PARITY (3), the value parameter specifies the parity selection for all transmit and receive channels.

AR_ODD (0) odd transmit parity and receive parity detect enabled
AR_EVEN (1) even transmit parity and rx parity detect enabled
AR_OFF (8) transmit parity and receive parity detect disabled
AR_RAW (0x2000) transmit parity and rx parity detect disabled

If the specified item is ARU_RX_CH*nn*_PARITY (900-915), where *nn* is the receiver channel (01 - 15), valid value parameter selections are:

AR_ODD (0) receiver parity detection enabled
AR_OFF (8) receiver parity detection disabled
AR_RAW (0x2000) receiver parity detection disabled

If the specified item is ARU_TX_CH*nn*_PARITY (1100-1104), where *nn* is the transmitter channel (01 - 05), valid value parameter selections are:

AR_ODD (0) odd transmitter parity
AR_EVEN (1) even transmitter parity
AR_OFF (8) transmitter parity disabled
AR_RAW (0x2000) transmitter parity disabled

If the requested item is ARU_TX_CH*nn*_SHUT_OFF (1700-1704), where *nn* is the transmitter channel (01 - 05), valid value parameter selections are:

AR_ON (7) external transmission is disabled
AR_OFF (8) external transmission is enabled

For the RAR-PCIE and RAR15-XMC-XT boards, disabling external transmission also causes the transmit pins to switch to a tri-state condition; for all other boards the transmit pins will switch to a null condition.

If the requested item is ARU_TX_CH*nn*_HB_INJ (3300-3304), where *nn* is the transmitter channel (01 - 05), valid value parameter selections are:

AR_ON (7) 33-bit transmission is enabled
AR_OFF (8) standard 32-bit transmission is enabled

If the requested item is ARU_TX_CH*nn*_LB_INJ (3500-3504), where *nn* is the transmitter channel (01 - 05), valid value parameter selections are:

AR_ON (7) 31-bit transmission is enabled
AR_OFF (8) standard 32-bit transmission is enabled

If the requested item is ARU_TX_CH*nn*_GAP_INJ (3700-3704), where *nn* is the transmitter channel (01 - 05), valid value parameter selections are:

AR_ON (7) 3-bit message gap is used
AR_OFF (8) standard 4-bit message gap is used

If the specified item is ARU_INTERNAL_WRAP (4), valid value parameter selections are:

AR_WRAP_ON	(0) internal wrap enabled
AR_WRAP_OFF	(1) internal wrap disabled

If the specified item is ARU_RX_TIMETAG_MODE (440), valid value parameter selections represent the timer/time-tag source and resolution. This item specifies the resolution of any timer-read or receive data time-tag value obtained via the API, with value selections defined as follows:

AR_TIMETAG_EXT_IRIG_64BIT	(0)
AR_TIMETAG_INT_USEC_64BIT	(1)
AR_TIMETAG_INT_20USEC_32BIT	(3)
AR_TIMETAG_INT_MSEC_32BIT	(4)

A value of AR_TIMETAG_EXT_IRIG_64BIT selects the source as the external IRIG receiver, if connected; otherwise, if the IRIG signal is not internally wrapped this selection would be invalid. All other values represent various timer/time-tag LSB resolution values based on the internal RAR-USB device timer.

If the specified item is ARU_ACCESS_SNAPSHOT_BUFFER (38), a valid value parameter for selecting the API Snapshot Buffer storage mode is:

ARU_LABEL_ONLY	(0) messages stored based on label
ARU_LABEL_WITH_SDI	(1) messages stored based on the combined label and SDI field values

If the specified item is ARU_IRIG_WRAP_ENABLE (441), valid value parameter selections are:

AR_ON (7) IRIG receiver internal wrap enabled
AR_OFF (8) IRIG receiver internal wrap disabled

If the specified item is ARU_IRIG_INPUT_THRESHOLD (442), the value parameter specifies the IRIG receiver threshold voltage in millivolts.

The items ARU_IRIG_ADJUST_THRESHOLD (443) and ARU_IRIG_QUICK_ADJUSTMENT (444) invoke the IRIG DAC auto-adjustment procedure. This procedure will determine the low and high threshold values at which the incoming IRIG signal is present. It then determines the best threshold level for the IRIG DAC, and returns the value to the application in place of a returned status (failures are indicated via return value of ARS_FAILURE). This execution of this adjustment should require less than one second.

If the specified item is ARU_IRIG_SET_BIAS (446), a valid value parameter consists of an offset to the board-supplied IRIG time specified in milliseconds. The bias time range is +/-32.768 seconds.

AR_SET_DEVICE_CONFIG

Syntax

CEI_INT16 ar_set_device_config (CEI_INT16 board, CEI_INT16 channel, CEI_INT16 item, CEI_INT16 value)

Description

This is the recommended routine to define the general device and ARINC 429 channel configuration attributes.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	The *item* argument value is invalid.
ARS_INVHARVAL	The *value* or *channel* argument value is invalid.
ARS_INVBOARD	The *board* argument value is invalid.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board — (input) Device to access. Valid range is 0-127.

CEI_INT16 channel — (input) Specifies which channel this routine is to access. Valid range is 0 to one less than the installed channel count for the respective channel type.

CEI_INT16 item — (input) Specifies the configuration attribute to define:

ARU_RX_BITRATE	receive rate for specified channel.
ARU_TX_BITRATE	transmit rate for specified channel.
ARU_RX_PARITY	receive parity for specified channel.
ARU_TX_PARITY	transmit parity for specified channel.
ARU_RX_FIFO_ENABLE	receive channel FIFO enable.
ARU_TX_FIFO_ENABLE	transmit channel FIFO enable.
ARU_TX_DISABLE	transmit channel transceiver disable.
ARU_TX_GAP_ERROR	transmit message gap error enable.
ARU_TX_BIT_ERROR	transmit message size error enable.

ARU_FAST_SLEW_RATE	transmit channel slew rate select.
ARU_RECV_MODE	receive channel internal wrap mode.
ARU_RX_MERGED_MODE	board receive buffer mode select.
ARU_ACCESS_SNAPSHOT_BUFFER	snapshot storage mode.
ARU_RX_TIMETAG_MODE	timer/time-tag source and resolution.
ARU_DISCRETE_OUT	sets a discrete output state.
ARU_IRIG_WRAP_ENABLE	enables IRIG receiver internal wrap.
ARU_IRIG_INPUT_THRESHOLD	sets the IRIG DAC threshold.
ARU_IRIG_ADJUST_THRESHOLD	both invoke IRIG DAC
ARU_IRIG_QUICK_ADJUSTMENT	auto-adjustment procedures.
ARU_IRIG_SET_BIAS	assigns an offset to IRIG time.
ARU_TX_PLAYBACK_ENABLE	controls transmit playback

CEI_INT16 value (input) the value to set the specified item.

If the requested item is ARU_RX_BITRATE (1) or ARU_TX_BITRATE (2), valid value parameter selections are:

ARU_SPEED_HIGH	(0) high rate (100Kbs)
ARU_SPEED_LOW	(1) low rate (12.5Kbs)

Any other value assigns a non-standard bus speed, and is translated as a divisor for the 16MHz device clock reference. This value and the respective baud rate may be interpreted using the following formulas:

Baud Rate = 16,000,000 / (Value+2)

Value = (16,000,000 / Desired Baud Rate) - 2

Note

Any non-standard transmit bus speed value resulting in a baud rate below 15KHz will be assigned to a slow slew rate. Any non-standard transmit bus speed value resulting in a baud rate at or above 15KHz will be assigned to a fast slew rate.

If the requested item is ARU_RX_PARITY (3), valid value parameter selections are:

AR_ON	(7) receiver parity enabled
AR_OFF	(8) receiver parity disabled

If the requested item is ARU_TX_PARITY (4), valid value parameter selections are:

ARU_PARITY_ODD	(0) odd transmitter parity
ARU_PARITY_EVEN	(1) even transmitter parity
ARU_PARITY_NONE	(2) transmitter parity disabled

If the requested item is ARU_RECV_MODE (5), valid value parameter selections are:

AR_WRAP_ON	(0) internal wrap enabled
AR_WRAP_OFF	(1) internal wrap disabled

If the requested item is ARU_RX_FIFO_ENABLE (16) or ARU_TX_FIFO_ENABLE (17), valid value parameter selections are:

AR_ON (7) Rx/Tx FIFO operation enabled
AR_OFF (8) Rx/Tx FIFO operation disabled

If the requested item is ARU_TX_DISABLE (10), valid value parameter selections are:

AR_ON (7) external transmission disabled
AR_OFF (8) external transmission enabled

Disabling external transmission also causes the transmit pins to switch to a tri-state condition.

If the requested item is ARU_TX_GAP_ERROR (8), valid value parameter selections are:

AR_ON (7) transmit message gap error enabled
AR_OFF (8) transmit message gap error disabled

If the requested item is ARU_TX_BIT_ERROR (6), valid value parameter selections are:

AR_LO (0) Low Bit Error operation is enabled
AR_HI (1) High Bit Error operation is enabled
AR_OFF (8) bit errors are disabled on this transmitter

If the requested item is ARU_FAST_SLEW_RATE (323), valid value parameter selections are:

AR_ON (7) Fast Slew Rate selected (1.5 µsec rise time)
AR_OFF (8) Slow Slew Rate selected (10 µsec rise time)

If the requested item is ARU_RX_MERGED_MODE (18), valid value parameter selections are:

AR_ON (7) board receive buffering mode set to merged
AR_OFF (8) board receive buffering mode set to individual

If the requested item is ARU_ACCESS_SNAPSHOT_BUFFER (38), valid value parameter selections are:

ARU_LABEL_ONLY (0) message storage on a label basis
ARU_LABEL_WITH_SDI (1) message storage on a label/sdi basis

If the specified item is ARU_RX_TIMETAG_MODE (440), valid value parameter selections represent the timer/time-tag source and resolution. This item specifies the resolution of any timer-read or receive data time-tag value obtained via the API, with value selections defined as follows:

AR_TIMETAG_EXT_IRIG_64BIT (0)
AR_TIMETAG_INT_USEC_64BIT (1)
AR_TIMETAG_INT_20USEC_32BIT (3)
AR_TIMETAG_INT_MSEC_32BIT (4)

A value of AR_TIMETAG_EXT_IRIG_64BIT selects the source as the external IRIG receiver, if connected; otherwise, if the IRIG signal is not internally wrapped this selection would be invalid. .

All other values represent various timer/time-tag LSB resolution values based on the internal RAR-USB device timer.

If the specified item is ARU_DISCRETE_OUT (12), valid value parameter selections are:

AR_HI (1) Discrete Out set to 0 (FET OFF – tri-state)
AR_LO (0) Discrete Out set to 1 (FET ON – conduct to Ground)
(see paragraph *Avionics Discrete I/O* for the Discrete circuit diagram)

If the specified item is ARU_IRIG_WRAP_ENABLE (441), valid value parameter selections are:

AR_WRAP_ON (0)	IRIG receiver internal wrap enabled
AR_WRAP_OFF (1)	IRIG receiver internal wrap disabled

If the specified item is ARU_IRIG_INPUT_THRESHOLD (442), the value parameter specifies the IRIG receiver threshold voltage in millivolts.

The items ARU_IRIG_ADJUST_THRESHOLD (443) and ARU_IRIG_QUICK_ADJUSTMENT (444) invoke the IRIG DAC auto-adjustment procedure. This procedure will determine the low and high threshold values at which the incoming IRIG signal is present. It then determines the best threshold level for the IRIG DAC, and returns the value to the application in place of a returned status (failures are indicated via return value of ARS_FAILURE). This execution of this adjustment should require less than one second.

If the specified item is ARU_IRIG_SET_BIAS (446), the API assigns an offset to the board IRIG time value calculation for any time and time-tag retrieval.

If the specified item is ARU_TX_PLAYBACK_ENABLE (5018), the API will assign the state of transmit playback feature on the RAR-USB and mutually exclude the message scheduler feature:

AR_ON (7) transmit playback enabled, msg scheduling disabled
AR_OFF (8) transmit playback disabled, msg scheduling enabled

AR_SET_573_CONFIG

Syntax

CEI_INT16 ar_set_573_config (CEI_INT16 board, CEI_INT16 item, CEI_INT32 value)

Description

This routine provides the method for manipulating the ARINC 573/717 channel configuration attributes.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	The *item* argument value is invalid.
ARS_INVHARVAL	The *value* argument value is invalid.
ARS_INVBOARD	The *board* argument value is invalid.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board — (input) Device to access. Valid range is 0-127.

CEI_INT16 item — (input) Specifies the configuration attribute to define:

ARU_RECV_MODE	receiver internal wrap.
ARU_RX_MERGED_MODE	receiver merge mode enable.
ARU_RX_BITRATE	receive channel bit rate.
ARU_RX_FIFO_ENABLE	receive channel FIFO enable.
ARU_TX_BITRATE	transmit channel bit rate.
ARU_TX_FIFO_ENABLE	transmit channel FIFO enable.
ARU_573_RX_AUTO_DETECT	data frame auto-detect enable.
ARU_573_RX_BPRZ_SELECT	receiver BPRZ/HBP selection.
ARU_573_TX_BPRZ_SELECT	transmit BPRZ encoding enable.
ARU_573_TX_HBP_SELECT	transmit HBP encoding enable.
ARU_573_TX_SLEW_RATE	transmit slew rate select.
ARU_573_SYNC_WORD1	receiver auto-detect sync word 1.
ARU_573_SYNC_WORD2	receiver auto-detect sync word 2.

ARU_573_SYNC_WORD3	receiver auto-detect sync word 3.
ARU_573_SYNC_WORD4	receiver auto-detect sync word 4.

CEI_INT32 value (input) The state to assign to the specified configuration item:

If the requested item is ARU_RECV_MODE (5), valid value parameter selections are:

AR_WRAP_ON (0) = internal wrap enabled
AR_WRAP_OFF (1) = internal wrap disabled

If the requested item is ARU_RX_MERGED_MODE (18), valid value parameter selections are:

AR_ON	(7) board receive buffering mode set to merged
AR_OFF	(8) board receive buffering mode set to individual

If the requested item is ARU_RX_FIFO_ENABLE (16) or ARU_TX_FIFO_ENABLE (17), valid value parameter selections are:

AR_ON	(7) FIFO operation enabled
AR_OFF	(8) FIFO operation disabled

If the configuration item is ARU_RX_BITRATE (1) or ARU_TX_BITRATE (2), valid item values are one of the following (0-7):

ARU_573_RATE_SIZE_384_32	384 bps, 32 word sub-frame
ARU_573_RATE_SIZE_768_64	768 bps, 64 word sub-frame
ARU_573_RATE_SIZE_1536_128	1536 bps, 128 word sub-frame
ARU_573_RATE_SIZE_3072_256	3072 bps, 256 word sub-frame
ARU_573_RATE_SIZE_6144_512	6144 bps, 512 word sub-frame
ARU_573_RATE_SIZE_12288_1024	12288 bps, 1024 word sub-frame
ARU_573_RATE_SIZE_24576_2048	24576 bps, 2048 word sub-frame
ARU_573_RATE_SIZE_49152_4096	49152 bps, 4096 word sub-frame

If the configuration item is ARU_573_RX_AUTO_DETECT (301), valid item values are one of the following:

AR_ON	(7) ARINC 573/717 frame auto-detection enabled
AR_OFF	(8) ARINC 573/717 frame auto-detection disabled

If the configuration item is ARU_573_RX_BPRZ_SELECT (302), valid item values are one of the following:

AR_OFF	(7) ARINC 573/717 HBP reception enabled
AR_ON	(8) ARINC 573/717 BPRZ reception enabled

If the configuration item is ARU_573_TX_BPRZ_SELECT (313), valid item values are one of the following:

AR_OFF (7) ARINC 573/717 BPRZ transmission disabled
AR_ON (8) ARINC 573/717 BPRZ transmission enabled

If the configuration item is ARU_573_TX_HBP_SELECT (314), valid item values are one of the following:

AR_OFF (7) ARINC 573/717 HBP transmission disabled
AR_ON (8) ARINC 573/717 HBP transmission enabled

If the configuration item is ARU_573_TX_SLEW_RATE (315), valid item values are one of the following:

ARU_573_TX_SLEW_1PT5 (1) = fast (1.5µsec rise time)
ARU_573_TX_SLEW_10PT0 (0) = slow (10.0µsec rise time)

If the configuration item is ARU_573_SYNC_WORD1 (307), ARU_573_SYNC_WORD2 (308), ARU_573_SYNC_WORD3 (309), or ARU_573_SYNC_WORD4 (310), a valid item value is any 12-bit non-zero value.

AR_SET_MULTITHREAD_PROTECT

Syntax

CEI_INT16 ar_set_multithread_protect (CEI_INT16 board, CEI_INT16 state)

Description

This routine controls the use of mutex/semaphore protection around all global/shared API data structure accesses performed within the API routines. This type of thread protection should be enabled for any multi-threaded application or reentrant API usage.

Return Value

ARS_NORMAL	routine was successful.
ARS_INVARG	An invalid *state* value was provided.
ARS_INVBOARD	An invalid *board* value was provided.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 state	(input) Multi-thread protection setting, valid values are defined as follows:

AR_ON (7) enables mutex/semaphore protection.

AR_OFF (8) disables mutex/ semaphore protection.

AR_SET_ PRELOAD_CONFIG

Syntax

CEI_INT16 ar_set_preload_config (CEI_INT16 board, CEI_INT16 item, CEI_UINT32 value)

Description

This routine is a legacy routine providing a method to induce API thread protection when executing multi-threaded applications with your RAR-USB device. Call this routine before calling AR_LOADSLV to update the value of a particular pre-load API operational configuration setting. This routine should not be called subsequent to any invocation of AR_LOADSLV.

If *item* is ARU_CONCURRENCY_MODE, the *value* parameter specifies the API concurrency mode. One of three modes may be selected: AR_CONC_NONE or AR_CONC_MULTITHRD, (the AR-STREAM-SW API does not support multi-process operation so the value parameter option AR_CONC_MULTIPROC is not allowed).

The default concurrency mode, AR_CONC_NONE, provides no multi-thread protection to the device and no multi-process API support. The user application must ensure that only one thread is calling into the API at any given time, and only a single process may interface with a particular board.

If AR_CONC_MULTITHRD concurrency mode is selected, thread protection for access to global/shared data structures is provided internally within the API. The user application may call into the API from multiple threads, but all threads must belong to a single process. The main user application thread should initialize the board with a call to AR_LOADSLV before other threads attempt to call into the API. This mode is supported on all operating systems supported by the RAR-USB software distribution.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_INVARG	An invalid *item* or *value* parameter was provided.
ARS_BOARD_MUTEX	Creation of the Board Lock mechanism failed.
ARS_NO_OS_SUPPORT	The item selection not supported with the host operating system.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 item	(input) Attribute about which to set information, currently limited to a single option, ARU_CONCURRENCY_MODE.
CEI_UINT32 value	(input) the value to set the specified item.
AR_CONC_NONE	no multi-thread or multi-process support (default).
AR_CONC_MULTITHRD	multi-thread concurrency mode (see Description section for details).

AR_SET_RAW_MODE

Syntax

CEI_INT16 ar_set_raw_mode (CEI_INT16 board, CEI_INT16 direction, CEI_INT16 channel, CEI_INT16 control)

Description

This routine is designed to provide compatibility with legacy ARINC 429 product APIs. AR_SET_DEVICE_CONFIG is the recommended routine for manipulating the channel parity selection.

Each transmit and receive channel can be configured to run in *raw* mode, where parity assignment and detection is disabled. When raw mode is selected, every 32-bit ARINC word is transmitted or received with the parity bit (msb) unchanged. This differs from a standard ARINC 429 data transfer in which the message parity is always calculated. Raw mode is typically used for older ARINC specifications such as ARINC 575.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVHARVAL	An invalid *channel* parameter was provided.
ARS_INVARG	An invalid *direction* or *control* parameter was provided.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 direction	(input) The type of channel specified in the channel argument (transmit or receive). Valid values to select transmit channels are: TRANSMIT_CHANNEL (0) ARU_XMIT (34) Valid values to select receive channels are:

RECEIVE_CHANNEL (1)
ARU_RECV (35)

CEI_INT16 channel — (input) Specifies which channel this routine is to access. Valid range is 0 to one less than the installed channel count for the respective channel type.

CEI_INT16 control — (input) Enables or disables raw mode.

AR_ON (7) enable "raw" mode, parity is disabled
AR_OFF (8) disable "raw" mode, parity assignment and/or checking is enabled

AR_SET_ STORAGE_MODE

Syntax

CEI_INT16 ar_set_storage_mode (CEI_INT16 board, CEI_INT16 mode)

Description

The RAR-USB device stores received ARINC messages in a single large merged circular buffer; however the AR-STREAM API will store uploaded ARINC messages in either individual channel-indexed circular buffers or in a single merged circular buffer. This routine allows you to select the universal receive message buffer mode for all receive channels on the device, as either BUFFERED or MERGED.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	An invalid *mode* parameter was provided.
ARS_INVBOARD	An invalid *board* value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 mode	(input) The type of receive data storage mode to assign. Valid values are:

ARU_BUFFERED	(0) store messages in individual buffers accessed by channel number
ARU_MERGED	(2) store messages in a merged buffer

AR_SET_TIME

Syntax

CEI_INT16 ar_set_time (CEI_INT16 board, pAR_TIMETAG_TYPE timeTag)

Description

This routine assigns a value to the specified RAR-USB device internal timer or IRIG time generator based on an application-supplied time format and value.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVARG	An invalid *direction* or *control* parameter was provided.
ARS_INVBOARD	An invalid *timeTag.timeTagFormat* structure member value was provided.
ARS_FAILURE	The specified device has not been initialized.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board — (input) Device to access. Valid range is 0-127.

pAR_TIMETAG_TYPE timeTag

(input) The 64-bit device timer or 30-bit IRIG time generator value to assign to the respective hardware. Valid options for the *timeTagFormat* structure member are:

AR_TIMETAG_EXT_IRIG_64BIT (0)

AR_TIMETAG_INT_USEC_64BIT (1)

To assign a 30-bit IRIG Day/Time value, the *timeTag* structure member should be defined as a 30-bit value using the following bit field format:

29-28	27-24	23-20	19-18	17-14	13-11	10-7	6-4	3-0
hundreds of days	tens of days	days	tens of hours	hours	tens of minutes	minutes	tens of seconds	seconds

To assign a 64-bit internal timer value, the *timeTag* structure member should be defined as a 64-bit 1 microsecond resolution time value.

The *timeTagRef* structure member is not used by this routine.

See the section titled *Time-tag Structure Definition* for more information on the AR_TIMETAG_TYPE data structure.

AR_SLEEP

Syntax

CEI_VOID ar_sleep (CEI_UINT32 sleep_ms)

Description

This routine suspends execution of the calling thread for the specified number of milliseconds. The method used to implement this operation is the C run-time library function *Sleep*. The accuracy of this operation is dependent upon the accuracy of the underlying operating system call.

Return Value

None

Arguments

CEI_INT32 sleep_ms	(input) Sleep duration, in milliseconds.

AR_SET_TIMERRATE

Syntax

CEI_VOID ar_set_timerrate (CEI_INT16 board, CEI_INT16 rate)

Description

This routine assigns the API internal timer reference resolution for compatibility with applications based on the CEI-x20 product family device timer and time-tag operation. When you invoke this routine, the RAR-USB API sets the current timer usage and time-tag reporting mode to the "CEI-x20 compatibility mode". In this mode, all scheduled message rate and start offset values and receive message time-stamp values are referenced in terms of the resolution value assigned in the "rate" parameter instead of the standard one millisecond (for scheduled message rate/offset) or one microsecond (for receive message time-stamps).

The actual RAR-USB hardware device time-tag reference timer resolution is not programmable; rather, it is a fixed one microsecond resolution.

The RAR-USB message scheduler minimum rate resolution is fixed at a one millisecond resolution. As a result, any timer rate assignment having a resolution that is not divisible by, or is less than, one millisecond, coupled with an attempt to define a message scheduler entry rate or start offset value that is not divisible by one millisecond results in that value being assigned to the nearest 1 millisecond value below the supplied value.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.
CEI_INT16 rate	(input) Resolution of the RAR-USB-emulated timer operation, specified as a tick-timer value having a resolution of 250 nanoseconds.

AR_STOP

Syntax

CEI_INT16 ar_stop (CEI_INT16 board)

Description

This routine disables all receive and scheduled transmit message processing on the device.

Return Value

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	The *board* argument value is invalid.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board	(input) Device to access. Valid range is 0-127.

AR_VERSION

Syntax

CEI_VOID ar_version (pCEI_CHAR verstr)

Description

This routine retrieves the current software version number of the AR-STREAM API.

Arguments

pCEI_CHAR verstr	(output) String representation of the API Version number consisting of up to 10 characters.

AR_WAIT

Syntax

CEI_VOID ar_wait (CEI_FLOAT nsecs)

Description

This routine delays the calling application by the specified number of seconds. The method used to implement this operation is the C run-time library function *Sleep*. The accuracy of this operation is dependent upon the accuracy of the underlying operating system call.

Arguments

CEI_FLOAT nsecs	(input) Number of seconds to delay.

AR_WRITE_429_TRANSMIT_PLAYBACK

Syntax

CEI_INT32 ar_write_429_transmit_playback (CEI_UINT32 device, pCEI_UINT32 entryCount, PTR_AR_TX_PLBK_BFR_TYPE playbackMsgSet, pCEI_UINT32 transmitStatus)

Description

This routine writes a transmit playback message set to the specified device. Once entries are written to the device, transmit playback must be enabled via AR_SET_DEVICE_CONFIG using the item parameter value ARU_TX_PLAYBACK_ENABLE. It is important to note this routine does not validate the supplied transmit playback time value or transmit channel in the *playbackMsgSet* structure array.

ARS_NORMAL	Routine execution was successful.
ARS_INVBOARD	The *board* argument value is invalid.
ARS_FAILURE	The specified device has not been initialized.
ARS_LOCK_ACCESS_TIMEOUT	The shared data structure access lock was not acquired.
ARS_RWR_INSERT_REQ_FAIL	The specified device failed a communications packet initialization request.
ARS_RWR_EXECUTE_FAIL	The specified device failed a communications block execution request.

Arguments

CEI_INT16 board — (input) Device to access. Valid range is 0-127.

pCEI_UINT32 entryCount — (input/output) As an input, specifies the number of playback entries to write to the playback buffer; as an output indicates how many playback entries were written to the playback buffer.

PTR_AR_TX_PLBK_BFR_TYPE playbackMsgSet (input) Reference to the playback message set to write to the playback buffer, defined as follows:

TIME_TAG_TYPE timeToTransmit – the playback time at which this message should be transmitted, in microseconds.

CEI_UINT32 transmitChannel – the transmit channel on which to transmit this message, with a valid range from 0 to one less than the number of installed transmit channels.

CEI_UINT32 messageData – the 32-bit ARINC 429 message to transmit.

CHAPTER 7

RAR-USB Hardware

Overview

This chapter describes specific hardware features of the RAR-USB product.

Power

The RAR-USB is a bus-powered USB device and requires no more than 500mA of USB supply current.

LEDs

The RAR-USB has four LED indicators, as shown in Figure 5, and described below.

The POWER LED (**PWR**) is illuminated whenever the unit has power.

The CONFIG LED (**CONFIG**) is illuminated when the driver has initialized the unit and configured the on-board FPGA.

The TRANSMIT ACTIVE LED (**ACTIVE)** flashes on/off when ARINC 429 messages are actively transmitting on any channel.

The RECEIVE ACTIVE LED (**ACTIVE)** flashes on/off when ARINC 429 messages are actively being received on any channel.

Figure 5. The RAR-USB LEDs

Optional Mounting Kit

An RAR-USB Mounting Kit is included with your RAR-USB device. It contains two pairs of mounting brackets and two sets of four screws and seals. The 4.5 mm seal should be used with the 18mm screw, while the 7.5mm seal is used with the 22mm screw.

To mount the brackets to the RAR-USB, first decide on the bracket configuration that best suits your mounting requirements. Figure 6 shows the bracket orientation and dimensions for the two mounting configurations available.

Once you have determined your desired mounting configuration, individually remove the screw cover located on the outer portion of each of the two face plates and replace it with the desired size seal/screw and the respective bracket. Brief assembly instructions are included within the mounting kit.

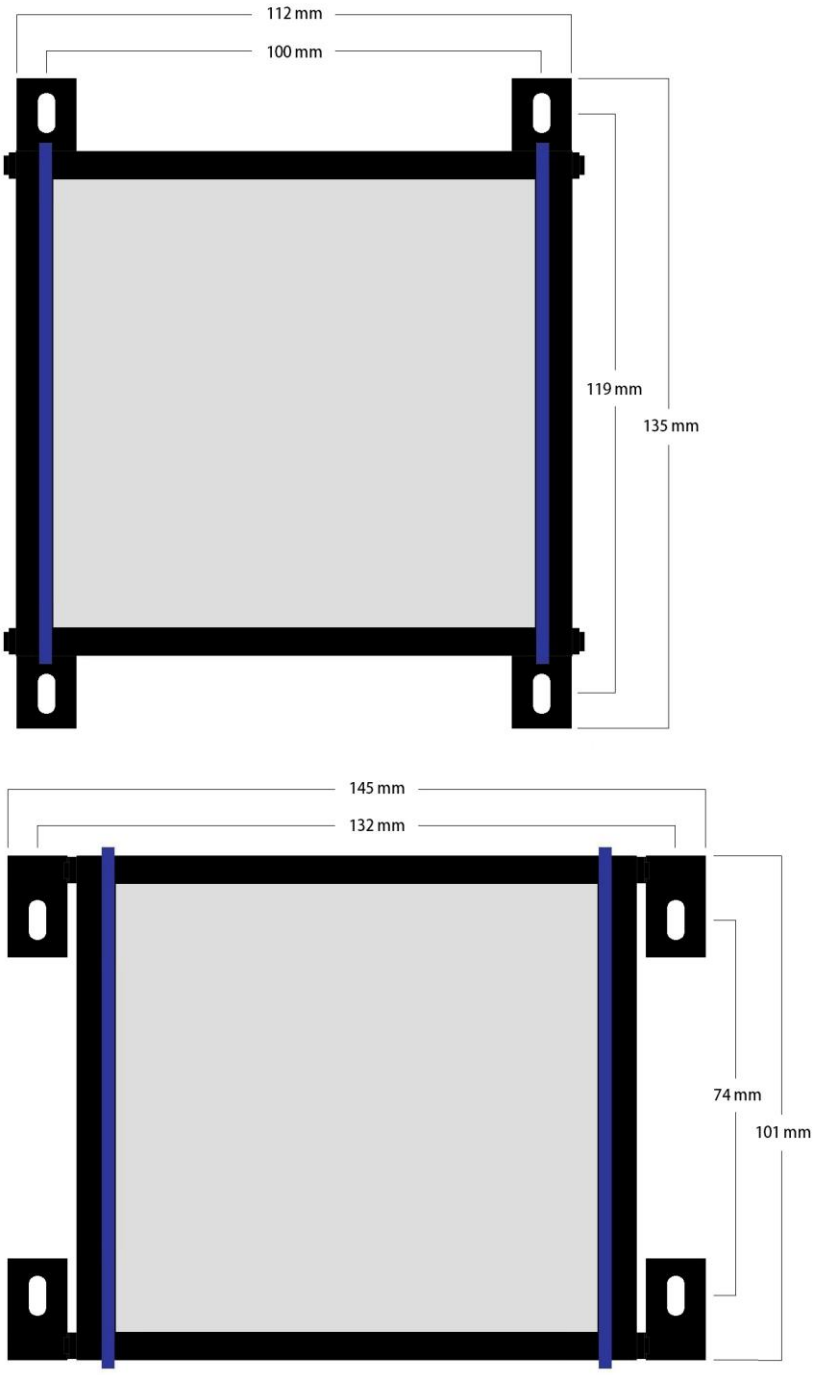


Figure 6. RAR-USB Mounting Options

IRIG DAC Register

The IRIG DAC Control Register determines the IRIG Receiver voltage threshold. The optimal threshold for a DC level IRIG signal is the midpoint between the upper and lower voltage levels of the IRIG signal. An appropriate level for an amplitude modulated (AM) encoded IRIG signal is at the 80% point between the upper and lower voltage levels of the IRIG signal. The 80% value is acceptable for a DC signal and should be used if the host application does not know which encoding (DC or AM) is used. The desired IRIG DAC value can be determined using the formula:

$$\text{IRIG DAC value} = (128 + ((256/3.3)*(4.99/22.1) * V_{IRIG\_Threshold}))$$

Where $V_{IRIG\_Threshold}$ is the value in Volts for the IRIG receive threshold relative to the input pins.

Avionics Discrete I/O

The RAR-USB provides eight bi-directional individually configurable Avionics Discrete I/O channels, used for general avionics-level I/O interfaces. The discrete outputs are low side n-channel FET switches capable of sinking 500mA, while the inputs are single ended, protected (50V max), with a logic threshold of approximately 2.0 V. The basic circuit for a discrete I/O channel is shown below:

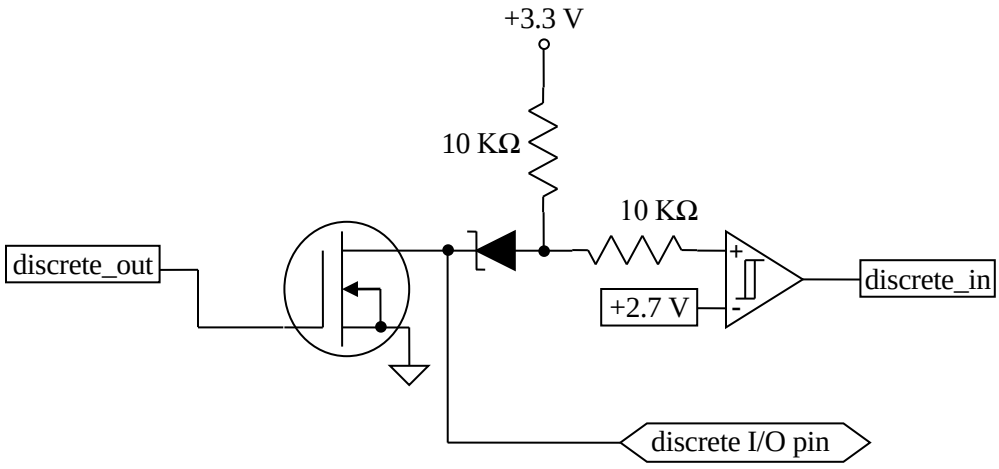


Figure 7. The RAR-USB Discrete I/O Circuit

Discrete Outputs

The discrete output channels have the following truth-table functionality:

Discrete Out	Discrete I/O pin
1	FET ON [conduct to Ground]
0	FET OFF [tri-state]

To assign Discrete Output states using the AR_SET_DEVICE_CONFIG API call with the item parameter value ARU_DISCRETE_OUT, valid value parameter selections are:

AR_LO Discrete Out set to 1 (FET ON – conduct to Ground)
AR_HI Discrete Out set to 0 (FET OFF – tri-state)

Discrete Inputs

When disconnected from any external signal, Discrete In reflects the value of Discrete Out. When the FET is on, reading Discrete In should return a "0". When the FET is off, reading Discrete In generally returns a "1" (because of the weak 22 KΩ pull-up resistor) but the load attached to the discrete I/O pin must also be taken into consideration.

The discrete input channels have the following truth-table functionality:

Discrete I/O pin	Discrete In
> 2.0 VDC	1
< 2.0 VDC	0

When a Discrete Input is connected to an external circuit, reading the state via AR_GET_DEVICE_CONFIG invocation with the item parameter ARU_DISCRETE_IN will result in one of the following return values:

AR_HI Discrete Input is 1
AR_LO Discrete Input is 0