ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *DDC DD-42924i5-300* at our website: *Click HERE*

Hardware/Software Manual
For the DD-42924i5 ARINC 429
PCI Interface Card
MN-42924i5-001

The information provided in this User's Guide is believed to be accurate; however, no responsibility is assumed by Data Device Corporation for its use, and no license or rights are granted by implication or otherwise in connection therewith.

Specifications are subject to change without notice.



105 Wilbur Place, Bohemia, New York 11716-2482

For Technical Support - 1-800-DDC-5757 ext. 7771
Headquarters - Tel: (631) 567-5600 ext. 7435, Fax: (631) 567-7358
Southeast - Tel: (703) 450-7900, Fax: (703) 450-6610
West Coast - Tel: (714) 895-9777, Fax: (714) 895-4988
United Kingdom - Tel: +44-(0)1635-811140, Fax: +44-(0)1635-32264
Ireland - Tel: +353-21-341065, Fax: +353-21-341568
France - Tel: +33-(0)1-41-16-3424, Fax: +33-(0)1-41-16-3425
Germany - Tel: +49-(0)8141-349-087, Fax: +49-(0)8141-349-089
Japan - Tel: +81-(0)3-3814-7688, Fax: +81-(0)3-3814-7689
World Wide Web – http://www.ddc-web.com

All rights reserved. No part of this User's Guide may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Data Device Corporation.



DATA DEVICE CORPORATION
REGISTERED TO ISO 9001
FILE NO. A5976

REV C March, 2004
© 2000 Data Device Corp.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RECORD OF CHANGE

Revision	Date	Pages	Description
A	3/2000	All	Original Issue
B	5/2002	All	Software update
C	3/2004	All	Figure 5, Removed all ref to ReadConfiguration() function

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table of Contents

SOFTWARE LICENSE AND POLICIES VII

TRADEMARKS VIII
UPDATE POLICY VIII

HOW TO USE THIS MANUAL IX

TEXT USAGE IX
SYMBOLS AND ICONS IX
SPECIAL HANDLING AND CAUTIONS X
TRADEMARKS X

INTRODUCTION 1

SYSTEM REQUIREMENTS 1
TECHNICAL SUPPORT 2

HARDWARE INSTALLATION 3

WINDOWS 95/98/2000 4
WINDOWS NT 4.0 8
HARDWARE CONFIGURATION 9
DDC 429 CARD MANAGER 10
CARD PINOUTs 13
LOOPBACK CONNECTOR 15

SOFTWARE INSTALLATION 16

SOFTWARE OVERVIEW 17

DLL HIGHLIGHTS 17
DD-42976S0, “ARINC 429 PCI CARD TOOLBOX” 18
DD-42999SO, “ARINC 429 COMMERCIAL AVIONICS UTILITIES” SOFTWARE SUITE (OPTIONAL) 22

PROGRAMMER’S REFERENCE 25

TYPE AND STRUCTURE DEFENITIONS 25
FUNCTION PROTOTYPES 25
INITIALIZATION 25
MULTIPLE CARD FUNCTIONS 26
DISCRETES 27
TRANSMITTING 29
RECEIVING 30
TIMERS AND TIMESTAMPS 32
GENERATING PC INTERRUPTS 35
DIAGNOSTIC FUNCTIONS 35
SHUTDOWN 36
USING PARITY AS DATA 36

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

STATUS INQUIRY FUNCTIONS 36
UTILITY FUNCTIONS 37

DLL HIGH-LEVEL FUNCTION DEFINITIONS 38

ADDLABEL 39
ADDLABELGROUP 40
CLEAR_FIFO 42
CLEARPCINTERRUPT 43
CLOSE429MSGWINDOW 44
CLOSECHANNEL 45
CLOSECHANNELALL 46
COMPOSEARINCWORD 47
CREATE429MSGWINDOW 49
DECOMPOSEARINCWORD 50
DELETEALLLABEL 52
DELETELABEL 53
DELETELABELGROUP 54
DISABLEGLOBALCARDTOPCINT 56
DISPLAY 57
ENABLEGLOBALCARDTOPCINT 58
ENABLEPARITYASDATA 61
GET429ABSOLUTETIME 63
GET429TIMEDIFFERENCE 65
GET429TIMERRESOLUTION 66
GETCARDCURRENTTIME 67
GETCHANNELSPEED 69
GETCHANNELSTATUS 70
GETCHIPSPEED 73
GETCURRENTCARD 74
GET_EXTENDED_SCHEDULED_LABEL_VALUE 75
GETPENDINGARINCWORDS 77
GETSCHEDULEDLABELVALUE 78
GETTESTRESULTSTRUCTADDR 80
INITIALIZE 81
KILL_EXTENDED_SCHEDULED 83
KILL_SCHEDULED 85
KILL_SCHEDULED_ALL 87
MODIFY_MAILBOX_MODE 88
OPENCHANNEL 90
P429T_VERSTR 93
P429TB_VERSION 94
P429TBVERSION 95
PANIC 96
READDISCRETE 97
READDISCRETEQUEUE 98
RECEIVE_FIFO 100

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RECEIVE_FIFO_BLOCK 102
RECEIVE_MAILBOX 104
RECEIVE_MAILBOX_EXTENDED 106
RUNDIAGNOSTIC 108
SETCHANNELPARITY 111
SETCHANNELSPEED 112
SETCHIPSPEED 114
SETCURRENTCARD 116
SETDETAILEDDIAGNOSTIC 117
SETDISPLAYFUNCTION 118
SETERRORFUNCTION 119
SETINITIALABSOLUTETIME 120
SHUTDOWN 121
STATUSTOASCII 122
TOOLBOXREADCONFIGFILE 124
TRANSMIT_EXTENDED_SCHEDULED 125
TRANSMIT_FIFO 127
TRANSMIT_FIFO_BLOCK 129
TRANSMIT_SCHEDULED 131
WRITEDISCRETE 133
ZEROCARDCLOCK 134

DLL LOW-LEVEL FUNCTION DEFINITIONS 135

ARINCTOTRANSCEIVER 136
ISBOARDPRESENT 137
LOADER 138
STARTCARD 139
TIMEDELAY 141
TOOLBOXREADCONFIGINFO 142
TRANSCEIVERTOARINC 143

APPENDIX A 144

TYPE DEFINITIONS 144

APPENDIX B 154

ACRONYMS 154

INDEX 155

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

List of Figures

Figure 1 - DD-42924i5 PCI Card Block Diagram 3
Figure 2 - DD-42924i5-300 Pinout 13
Figure 3 - Loopback Connector - P/N DDC-70140-1 15
Figure 4 - DD-42999S0 22
Figure 5 - Discrete Schematic 27
Figure 6 - Mailbox and FIFO Reception 31

List of Tables

Table 1 - DD-42924i5-300 Pinouts 14

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SOFTWARE LICENSE AND POLICIES

THIS PROGRAM LICENSE AGREEMENT IS DISPLAYED CONSPICUOUSLY ON THE PACKAGE SO THAT YOU CAN READ IT BEFORE OPENING THIS PACKAGE. RETAINING POSSESSION OF THE PACKAGE AND USING THE PROGRAMS AND MATERIALS CONTAINED IN IT INDICATES THAT YOU HAVE AGREED TO THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE WITH THEM, YOU SHOULD PROMPTLY RETURN THE PACKAGE UNOPENED TO DDC AND YOUR MONEY WILL BE REFUNDED.

LICENSE. Data Device Corporation, ("DDC") grants a license to the individual and/of corporate licensee to use the Program Product (the set of computer programs and machine readable and printed materials supplied with this package for a term beginning on the date you accept this Agreement and continuing until termination. This license agreement is automatically terminated if you violate any of the terms and conditions of this License Agreement. You may terminate this agreement at any time by destroying all copies, modifications, and original materials supplied with the Program Product and destroying any portions of the DDC Program Product merged in any form.

USE OF THE PROGRAM PRODUCT. You may use the Program Product only on the Card with which the product was purchased. You may copy the program for backup or transfer to any machine. Reverse compiling and disassembling of programs in object format are prohibited. Title to the program is not transferred to you by this license. Any attempted sublicense, assignment, or other transfer of the programs or the rights and obligations of this Agreement without the prior written consent of DDC shall be void.

RESTRICTIONS. You may not use, copy, modify, merge, or transfer the Program Product or any portion or modification thereof in whole or in part, except as expressly provided in this Agreement.

LIMITED WARRANTY. DDC warrants the distribution diskette(s) upon which the Program Product is provided to be free from defects in materials and workmanship under normal use for a period of one year from the date of purchase by the original purchaser. FOR BREACH OF ANY WRITTEN OR IMPLIED WARRANTY ON THIS PRODUCT, THE CONSUMER IS LIMITED TO THE REPLACEMENT OF THE DEFECTIVE DISKETTE(S).

NOTE: SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES SO

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

DDC MAKES NO OTHER WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AND NO WARRANTY THAT THE PROGRAM PRODUCT IS FREE OF ERRORS. DDC and its authorized dealers shall not be liable for the cost of any repair of correction required for defective DDC Program Product material except as stated in this paragraph.

GENERAL TERMS. This is the only agreement between User and DDC regarding this Program Product and it may be modified only by a written agreement between those parties. This agreement cannot be modified by purchase orders, advertising or other representation by any person. If any provision of this agreement shall be held invalid, illegal, or unenforceable, then the validity, legality and enforceability of the remaining provisions shall not be affected or impaired thereby. This agreement shall be governed by the laws of the State of New York. By opening or keeping the package, you acknowledge that you have read this agreement, understand it and agree to be bound by its terms and conditions, you acknowledge that you have read this agreement, understand it and agree to be bound by its terms and conditions.

TRADEMARKS

Trademarks mentioned in this manual are the property of their respective owners.

UPDATE POLICY

We will update the product as new functions are developed or major bugs are fixed, to the extent of the warranty period. We will send you further information, as it becomes available.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

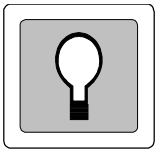
HOW TO USE THIS MANUAL

This manual uses typographical and iconic conventions to assist the reader in understanding the content. This section will define the text formatting and icons used in the rest of the manual. This manual is formatted with a 'Scholar Margin' where many tips, symbols or icons will be located.

TEXT USAGE

- **BOLD**–text that is written in bold letters indicates important information and table, figure, and chapter references.
- ***BOLD ITALIC***–will designate DDC Part Numbers.
- `Courier New`–is used to indicate code examples.
- <…> - Indicates user entered text or commands.

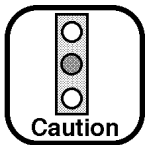
SYMBOLS AND ICONS



The Idea/Tip icon will be used to identify a handy bit of supplementary information that may be useful to the user.



The Note icon signifies important supplementary information that will be useful to the user.



The Caution icon identifies important information that presents a possibility of damage to the product if not heeded.

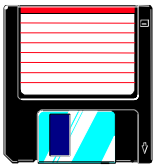


Much stronger than a Caution, the Warning icon presents information pertaining to hazards that will cause damage to the product and possible injury to the user.



The Reference icon indicates that there is related material in this manual or in another specified document.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



The Disk Icon describes information that is related to software.

SPECIAL HANDLING AND CAUTIONS

The ***DD-42924i5-300*** uses state-of-the-art components, and proper care should be used to ensure that the device will not be damaged by Electrical Static Discharge (ESD), physical shock, or improper power surges and that precautions are taken to avoid electrocution.



- Turn off power to the PC and unplug from wall.
- **NEVER** insert or remove card with power turned on.
- Ensure that standard ESD precautions are followed. As a minimum, one hand should be grounded to the power supply in order to equalize the static potential.
- Do not store disks in environments exposed to excessive heat, magnetic fields or radiation.

TRADEMARKS

All trademarks are the property of their respective owners.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

INTRODUCTION

The ***DD-42924i5*** provides full, intelligent interfacing between the ARINC 429 Data Bus and a PCI Bus. The ***DD-42924i5*** is packaged on a half-size IBM PC/AT printed circuit board with a PCI bus interface. On-board Interrupt Mask and Interrupt Status Registers support flexible operation for both interrupt and polling applications. Other features of the board include:

- Plug-and-Play (PnP) compatible for easy installation
- Intel 80C186 microprocessor operating at 20 MHz
- 256 Kbytes of on-board Program RAM
- 256 Kbytes of dual-port shared static RAM
- Two FIFOs (one input and one output)
- Five discrete inputs and five discrete outputs
- Programmable event interrupts to PC
- Designed in a modular format, providing up to four ARINC 429 receivers and up to two ARINC 429 transmitters

Application software has the capability to simulate the outputs of various airborne systems, receive inputs from these systems, and provide bus analyzer functions. Included with each ***DD-42924i5*** PCI Card is the DDC Toolbox library with example source code that allows users to easily write their own application software for the card.

WHAT IS INCLUDED IN THIS MANUAL?

This manual contains a detailed installation guide for the ***DD-42924i5*** PCI Card and a basic overview of the software supplied with the card. Toolbox Library, DLL, and Driver installations for Windows® 95, 98, Windows NT®, and Windows 2000 will be covered in this manual.



The library software provides a level of abstraction such that it is not necessary to understand the operation of the chip set. This manual provides an introduction to the library.

SYSTEM REQUIREMENTS

An IBM compatible PC with an 80386 processor or better is required. The PC must be configured with the Windows 95, 98, NT, or 2000 operating system, math co-processor, monitor, and disk drive as required by the software. A cable assembly is also required.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TECHNICAL SUPPORT

In the event that problems arise beyond the scope of this manual you can get in touch with DDC by calling:

Customer support:

1-800-DDC-5757, ext. 7257 or 7435

Headquarters:

1-631-567-5600, ext. 7257 or 7435

Regional Offices:

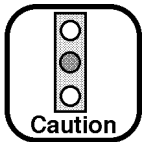
Southeast:
(703) 450-7900 Fax: (703) 450-6610
West Coast:
(714) 895-9777 Fax: (714) 895-4988
Europe:
44 (1635) 811140 Fax: 44 (1635) 32264
Asia/Pacific:
81 (3) 3814-7688 Fax: 81 (3) 3814-7689

Or by faxing 1-631-567-5758 to the attention of DATA BUS Applications.

DDC also has an Internet World Wide Web site, which allows customers to easily download new revisions of software and documentation. The Internet address is **www.ddc-web.com**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HARDWARE INSTALLATION





The ***DD-42924i5*** card is a PCI Target device and may be inserted into any PCI 2.1, or higher, compatible slot. When installing the card, the following should be observed:

- **NEVER** insert or remove the card with the power turned on.
- **ALWAYS** take proper precautions to guard against static damage. Use a wrist strap if available, or ensure proper static grounding by touching the power supply cover **WITH POWER OFF**.
- Insert the card at a slight angle so that the connectors first protrude from the rear opening and then gently press the card into the motherboard connector. Secure with proper hardware.
- Make sure that adjacent cabling and wiring do no hinder the airflow around the card.

The ***DD-42924i5*** contains jumpers to select the output discrete Voltage level. This card is designed as a Plug-and-Play device and as such, there are no jumpers or switches to be set for address and interrupt selection. Figure 1 shows a block diagram for the 2 x 4 PCI Card.

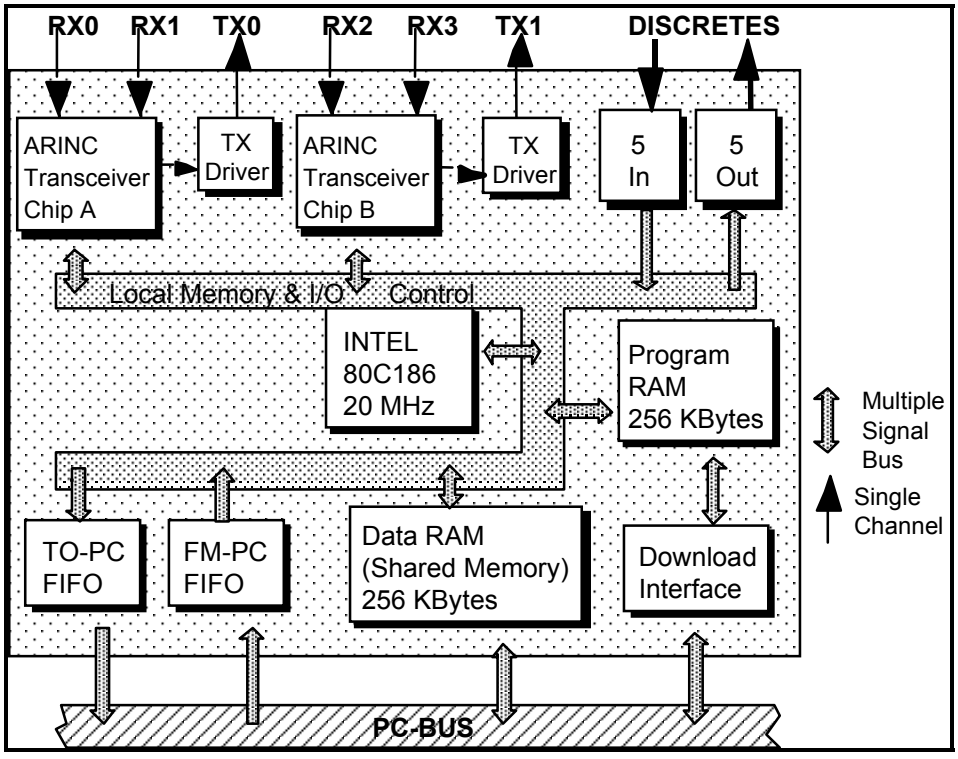


Figure 1 - DD-42924i5 PCI Card Block Diagram

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

WINDOWS 95/98/2000

For a Windows 95/98 or Windows 2000 operating system, continue with the following installation instructions. If using another operating system, skip to the relevant installation section.

1) Install the card as described in the previous **HARDWARE INSTALLATION** section.

2) Turn on the computer.

3) The **Add New Hardware Wizard** window will appear after boot-up:



4) Click on the **Next** button.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



5) Click on **Search for the best driver for your device**.

6) Click on the **Next** button.

7) Insert the Data Bus Software CD or, if using floppy disks, disk 1 into the appropriate drive.

8) Select the Checkbox for the appropriate drive type option, or select the **Specify a location** option to browse for the driver.



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



9) Once the driver has been located, click on the **Next** button to install the device driver.



10) Click on the **Finish** button to confirm the installation.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



11) Click on the **Finish** button to close the Add New Hardware Wizard.

12) Continue configuring the device with the **DDC 429 Card Manager**. (see page 9)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

WINDOWS NT 4.0

For a Windows NT 4.0 operating system, continue with the following installation instructions. If using another operating system, skip to the relevant installation section.

1) Install the card as described in the previous **HARDWARE INSTALLATION** section.

2) Turn on the computer.

3) Insert the Data Bus Software CD or, if using floppy disks, disk 1 into the appropriate drive.

4) Click on the Windows **Start** button.

5) Click on **Run**.

6) In the **Open** window type ***<drive>:\Setup.exe***.

7) Click **OK.**

8) Follow any instructions that may appear on the screen. The computer will need to be rebooted for the device driver to run.

9) Once rebooted, proceed with the **DDC 429 Card Configuration** instructions. (see page 9).

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

HARDWARE CONFIGURATION

The ***DD-42924i5*** is a PCI device, and as such does not require any jumpers or switches to set the Base address or interrupt values. The job of configuration for Plug-and-Play PCI configuration is performed by the PC BIOS. During the initial power on boot process, the BIOS performs an enumeration of the PCI bus and provides a configuration in the system that satisfies the card requirements. The card provides the BIOS with details of how much memory it requires, the interrupts that it supports, and any other operating parameters that the system needs to know by way of configuration registers built into the card. These registers are configured at the factory to contain the optimum values for the operation of the ***DD-42924i5***. The ***DD-42924i5*** PCI card and software drivers allow for shared interrupts, thus simplifying the installation and reducing the risk of device conflicts.

Once the card has been properly installed, its configuration information is stored to the Windows Registry. The card information stored in the Registry is access and controlled through the use of the **DDC 429 Card Manager**. The **DDC 429 Card Manager** may be used to view the status of installed cards, check current software driver revisions, and map an installed card to a device number. In order to access an installed card you **must** assign it a device number.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

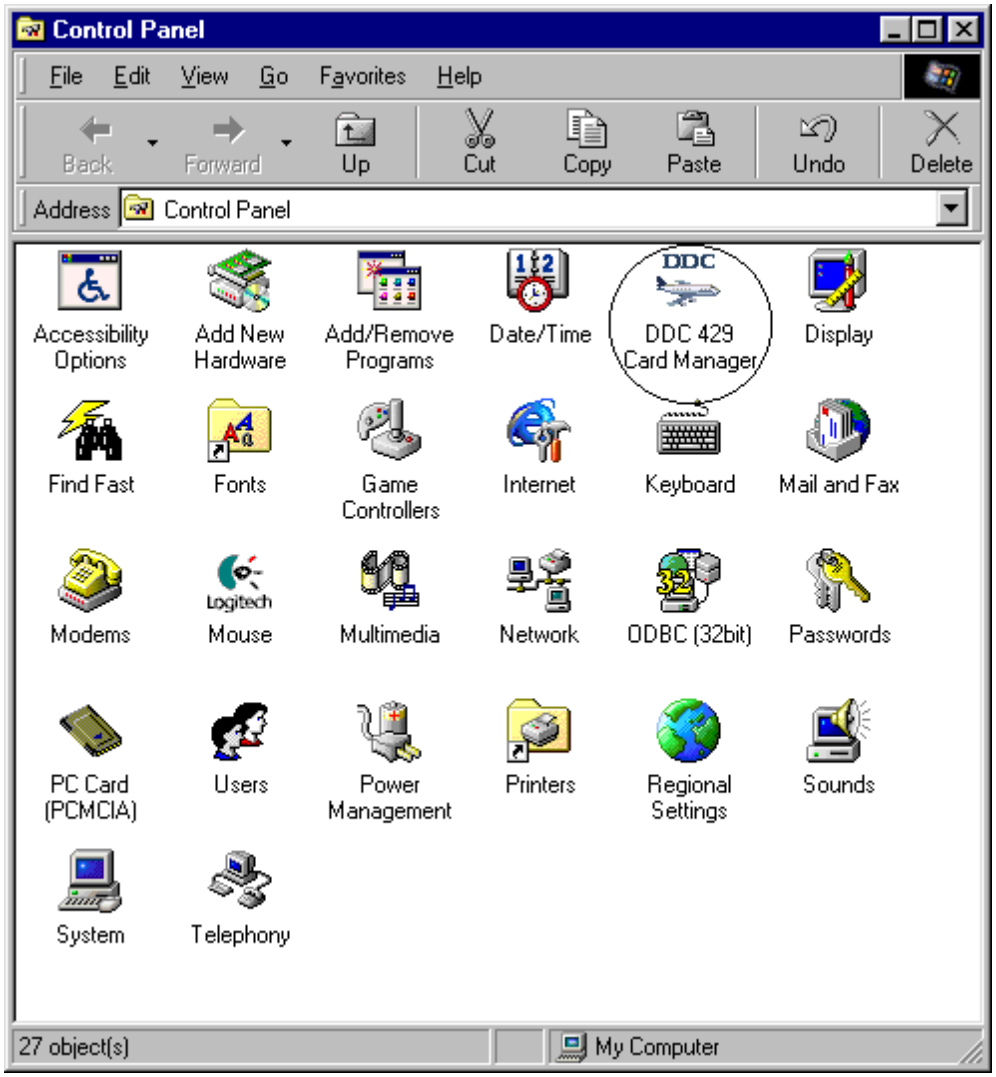
DDC 429 CARD MANAGER

Once the hardware has been installed, the device should be configured. Device configuration is performed through the use of the **DDC 429 Card Manager** located in the **Windows Control Panel**.

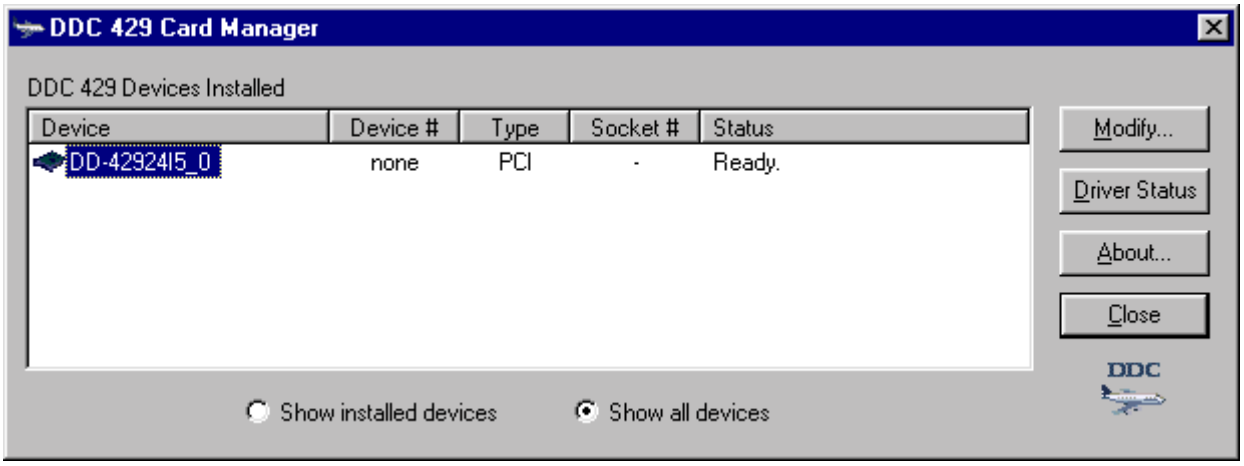
1) Turn on the computer.

2) Open the **Windows Control Panel** by clicking on the **Start** button. Then click on the **Settings** option. Finally, click on **Control Panel**.

3) Double-click on the **DDC 429 Card Manager** icon (circled in the screen image below).

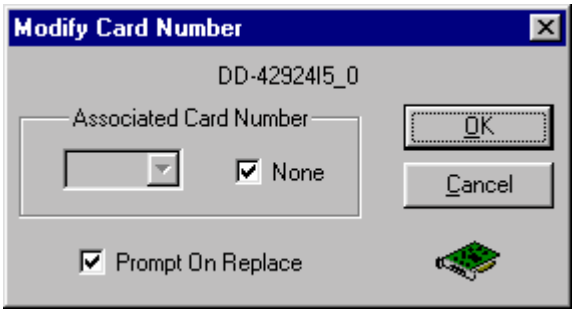


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



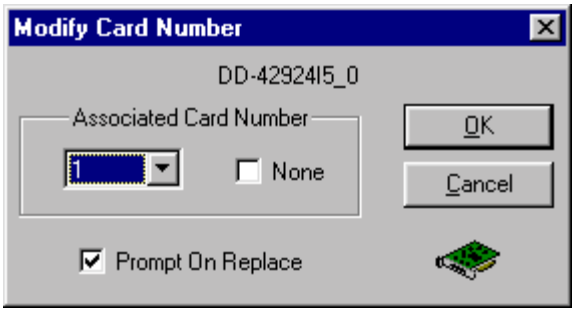
4) Select an installed Device (i.e., **DD-42924i5_0**) and click the **Modify** button.

5) The **Modify Card Number** window will appear:



6) Unselect the **None** option by clicking on its check box.

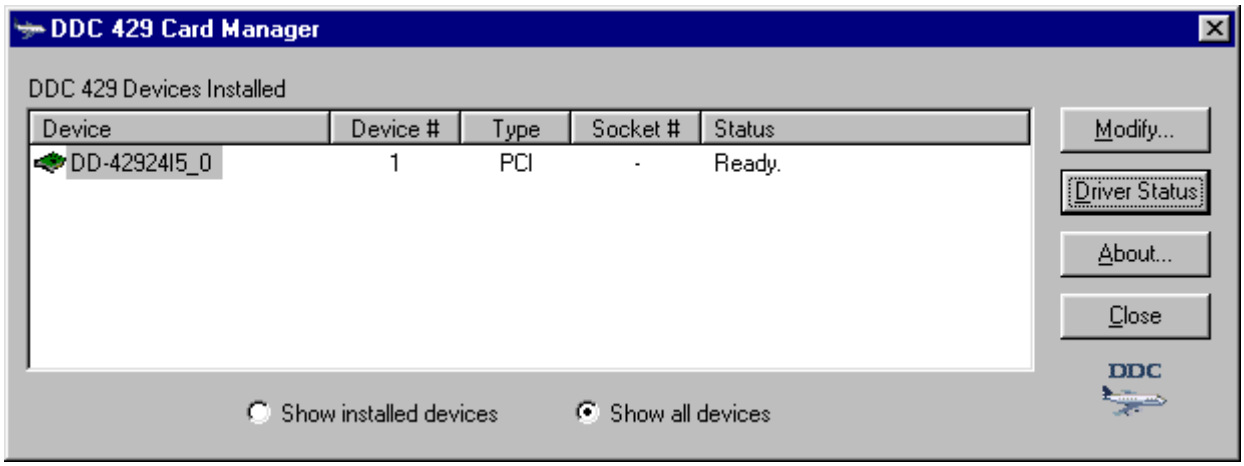
7) Next, select a device number to associate the card with by clicking on the list box and choosing a device number.



8) When finished, click on the **OK** button.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The **DDC 429 Card Manager** window should display the device mapped to a device number. The card status should show ready, such as in the window displayed below.



9) Click on the **Close** button to complete card configuration. The card is now ready for use.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CARD PINOUTS

This section describes the pinouts for the card. The card has one connector, a DB-37 #(DPC-37S). The pinouts for this connector are shown in Figure 2 and Table 1 on the following pages.

Note: *All ground returns are connected to the chassis ground.*

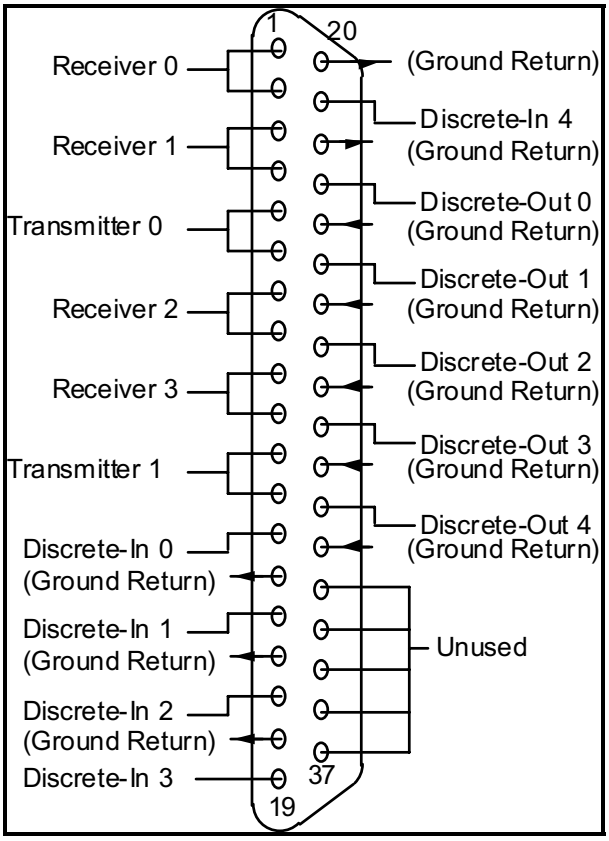


Figure 2 - DD-42924i5-300 Pinout

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Table 1 - DD-42924i5-300 Pinouts

Pin Number	Description
1	ARINC 429 receiver 0 A
2	ARINC 429 receiver 0 B
3	ARINC 429 receiver 1 A
4	ARINC 429 receiver 1 B
5	ARINC 429 transmitter 0 A
6	ARINC 429 transmitter 0 B
7	ARINC 429 receiver 2 A
8	ARINC 429 receiver 2 B
9	ARINC 429 receiver 3 A
10	ARINC 429 receiver 3 B
11	ARINC 429 transmitter 1 A
12	ARINC 429 transmitter 1 B
13	Discrete-in 0
14	Ground return for DI0
15	Discrete-in 1
16	Ground return for DI1
17	Discrete-in 2
18	Ground return for DI2
19	Discrete-in 3
20	Ground return for DI3
21	Discrete-in 4
22	Ground return for DI4
23	Discrete-out 0
24	Ground return for DO0
25	Discrete-out 1
26	Ground return for DO1
27	Discrete-out 2
28	Ground return for DO2
29	Discrete-out 3
30	Ground return for DO3
31	Discrete-out 4
32	Ground return for DO4
33-37	Unused

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LOOPBACK CONNECTOR

The wiring inside the loopback connectors supplied with the ***DD-42924i5*** PCI card is illustrated in Figure 3.

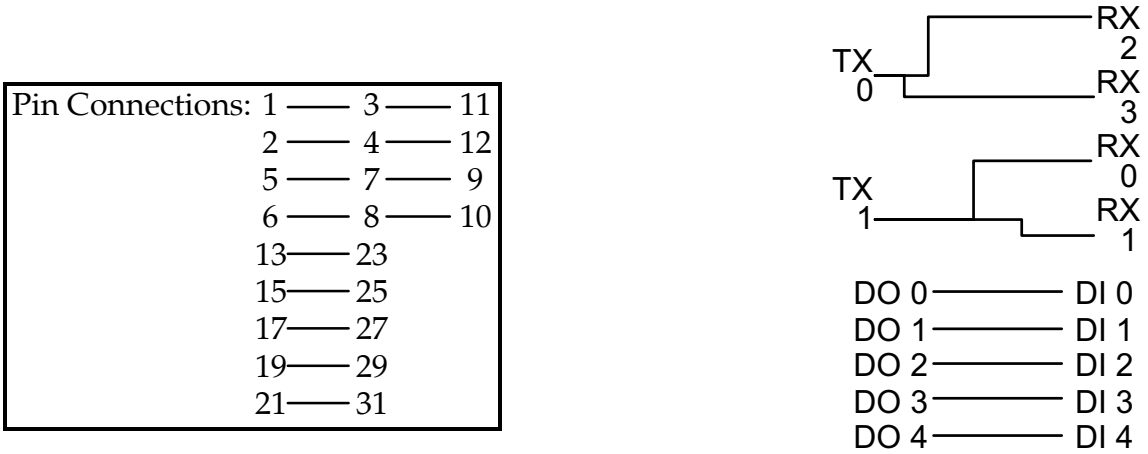


Figure 3 - Loopback Connector - P/N DDC-70140-1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SOFTWARE INSTALLATION

The ***DD-42924i5*** PCI Card is supplied with Runtime Libraries for Windows 95/98, Windows NT, and Windows 2000 and demonstration software for Windows 95/98, Windows NT, and Windows 2000.
The basic installation is listed below.



Note: *If installing under Windows NT and you have already completed the* ***Hardware Installation*** *and* ***Hardware Configuration*** *Sections, you do not need to run the* ***setup.exe*** *program again. The software libraries have already been installed to your system.*

- Run the **setup.exe** program located on the ***DD-42976S0*** installation disk.
- Supply the necessary information during setup.

This software is a runtime library that provides the user with a hardware abstraction layer for the DDC ***DD-42924i5*** PCI series ARINC 429 Card. This software layer includes routines that dramatically reduce software development time by providing a high level interface to the PCI card. There are eighteen demo programs included with the library to demonstrate the full capabilities of the card

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SOFTWARE OVERVIEW

DLL HIGHLIGHTS

The DDC ARINC 429 Win32 DLL is a collection of routines that will allow a Windows Win32 programmer access to the following features of the ***DD-42924i5*** PCI Card, in a simple, easy to understand format:

TRANSMITTING

- Queued transmission on any channel
- Broadcast (Scheduled) word transmission on any channel

RECEIVING

- Queued receiving on any channel (with label filtering and time stamping)
- “Mailbox” receiving on any channel

MISCELLANEOUS

- Programmable ARINC bus speed (High or Low)
- Supports up to 8 ***DD-42924i5*** PCI Cards per PC

There are several software applications which are used in conjunction with ***DD-42924i5*** PCI Cards. Some of these applications may be included with your card.

If you did not receive the application software described on the next page with your card, please contact DDC customer service.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DD-42976S0, "ARINC 429 PCI CARD TOOLBOX"

This software is a runtime library that provides the user with a hardware abstraction layer for the DDC PCI series ARINC 429 Card. This software layer includes routines that dramatically reduce software development time by providing a high level interface to the PCI card. There are eighteen demo programs included with the library to demonstrate the full capabilities of the card. The demos are located in the EXE directory, and can be run by typing in the appropriate filename at the Toolbox Command Prompt. The source code for the demos can also be viewed by opening the "C" code file for the corresponding demo.

DEMO ONE

This program demonstrates the writing and reading of discretes. The logical step of comparing what was read by the discrete inputs with what was written to the corresponding discrete outputs is left as an optional exercise for the programmer. The loopback connector is required for this demo.

DEMO TWO

This program uses the filtering system to look for specific labels and displays the data when it is received on receivers R0-R3. After initializing receiver R0 to receive words in FIFO filtering mode, the program sets up filters for the following labels: 160, 170, and 180 on receiver R0; 150 on receiver R1; 190 on receiver R2; 190 and 200 on receiver R3. This demo requires an external source of ARINC words to receivers R0-R3. If no source is provided, Receive_FIFO will return QUEUE_IS_EMPTY.

DEMO THREE

This program tests the capability of the card to receive words on receiver R0 with time stamping but no filtering. Words received on receiver R0 will be displayed on the screen. If no words are received, QUEUE_IS_EMPTY will be displayed on the screen. This demo requires an external source of ARINC words to receiver R0. If no source is provided, Receive_FIFO will return QUEUE_IS_EMPTY.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DEMO FOUR

This program tests the basic FIFO reception with time stamping. Receiver R3 is enabled to receive words displayed in the following format: the label in octal, the data word in hex, the absolute time in milliseconds, and the relative time in milliseconds. The relative time is the time since a word with the same label was received. Transmission is initiated from transmitter T0. If the user presses 'F' or 'f' during the reception loop, Clear_FIFO will be called for receiver R3 to empty the FIFO. This demo requires the loopback connector in order to connect receiver R3 to transmitter T0.

DEMO FIVE

This program tests the basic mailbox reception of the card. Receive_Mailbox is called looking for labels 160, 170, and 180(decimal), if their respective mailboxes are not empty, the word will be displayed. The word is displayed in the following format: the label in decimal, and the data in decimal and hex. If the mailbox is empty, BIN_IS_EMPTY is displayed on the screen. This demo requires the loopback connector.

DEMO SIX

This program tests basic FIFO transmission on transmitters T0 and T1. The Transmit_FIFO function is used for transmission on transmitter T0 and the Transmit_FIFO_Block function is used for transmission on transmitter T1. After each successful call to the Transmit_FIFO function, a '?' will be displayed on the screen. After each successful call to the Transmit_FIFO_Block function, a '!' will be displayed on the screen. This means that ten '?' followed by a '!' will be displayed. To verify correct transmission on transmitter T0 or transmitter T1 an external device would have to be set up to receive ARINC 429 words.

DEMO SEVEN

This program tests complete scheduled transmissions on transmitter T0. An initial schedule of transmission is set up on labels 1-7, then deletes the scheduled transmission on labels 3, 4, 1, 7, and all the rest. To verify correct transmission on transmitter T0, an external device would have to be set up to receive ARINC 429 words.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DEMO EIGHT

This program tests simultaneous scheduled and FIFO transmission on transmitters T0 and T1. If the transmission is successful then a '.,' will be displayed on the screen. Since Initialize was called with a timer resolution of TEN_MILLI_SEC, the scheduled transmissions on transmitter T1 which are supposed to be 10, 12, 14 and 18 msec will actually occur at 10 msec intervals. To verify correct transmission on transmitter T0, an external device would have to be set up to receive ARINC 429 words.

DEMO NINE

This program tests scheduled transmission and FIFO reception on two cards. This demo expects two cards to be installed in the PC, and requires a loopback connector for each card.

DEMO TEN

This program tests all card capabilities. It calls a 'RunDiagnostic' with ROM_FIFO_CHECK, MEMORY_CHECK, and LOOPBACK_CHECK, does a FIFO loopback check, and does 186 timer interrupts. This demo requires the loopback connector.

DEMO ELEVEN

This program tests all of the same operations as demo ten does, but goes through the loop three times. This demo requires the loopback connector.

DEMO TWELVE

This program demonstrates how the toolbox can be used to send an interrupt to the PC once every ten milliseconds. Once the interrupt generation feature is enabled, the main loop waits for one second and then displays the total number of interrupts that occurred. This demo requires no input to the card.

DEMO THIRTEEN

This program demonstrates how the toolbox can be used to send an interrupt to the PC upon reception of an ARINC word. This demo requires an external source to transmit ARINC words to receiver R0.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DEMO FOURTEEN

This program demonstrates the transmission of words in the “parity as data” mode of operation. This demo transmits four words using this mode.

DEMO FIFTEEN

This program demonstrates the GetChannelSpeed, GetChipSpeed, GetChannelStatus, Get429Timer Resolution, and P429TB_version functions.

DEMO SIXTEEN

This program provides a demonstration of the ReadDiscreteQueue toolbox function.

DEMO SEVENTEEN

This program demonstrates how the toolbox can send and respond to two card’s interrupts simultaneously with independent ISR’s or a common routine. The interrupt generation interval is ten milliseconds for card one and twenty milliseconds for card two. The main loop prints out the total number of interrupts that occurred per card. This demo requires no input to the card.

DEMO EIGHTEEN

This program demonstrates how the card’s library can transmit and receive data at user specified speeds. The program transmits data from both transmitters at user specified data rates and does a single mailbox reception on receiver zero and receiver two to verify proper loopback operation. The supplied loopback connector must be connected for this demo to work properly. The message transmission rate is 6msec and the infinite reception loop can be exited by pressing ‘8’ or ‘Q’ on the keyboard.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DD-42999SO, "ARINC 429 COMMERCIAL AVIONICS UTILITIES" SOFTWARE SUITE (OPTIONAL)

This software package has two software programs included, which are described on the next page.

Some of the common features are listed below:

- Simple-to-use Windows Interface
- Supports Windows 95/98/NT/2000
- Compatible with DDC's hardware (PCI, ISA, PCMCIA, PMC, and VME (MXI))
- One Convenient Package includes the ARINC 429 Data Bus Analyzer, and the ARINC 615 Data Loader
- Provides solutions for all your Avionics testing, simulation, and monitoring needs



Figure 4 - DD-42999S0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ARINC 429 DATA BUS ANALYZER

The ARINC 429 Data Bus Analyzer is DDC's advanced, easy-to-use, Windows-based software application solution for data analysis and simulation on an ARINC 429 data-bus. When combined with one of DDC's hardware cards, the ARINC 429 Data Bus Analyzer provides flexibility for the capture, simulation, manipulation, and display of ARINC 429 data on the bus. This software application features real-time engineering unit conversion, data filtering, programmable triggering, discrete input and output triggering capabilities, and raw-data time stamping. This software enables engineers and maintenance personnel to easily test and maintain avionics systems.

This program allows easy access to data analysis of ARINC 429 bus traffic.

Some features include:

- User friendly 32-bit Windows interface
- Reception and transmission of ARINC 429 words
- Supports 32 receive channels and 32 transmit channels
- Embedded engineering unit database and editing environment
- User defined data filters and triggers
- Storage of time-stamped ARINC 429 words to disk
- Provides discrete triggering capabilities
- Filtering and triggering with real time raw data conversion in user definable engineering units
- Chronological and Monitor data display formats
- Data searching and sorting engines
- Saves Analyzer settings to configuration files

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ARINC 615 DATA LOADER

The ARINC 615 Data Loader is DDC's advanced, easy-to-use, Windows-based software application solution for a portable ARINC 615 Data Loader. This software application provides all the functionality of a portable ARINC 615 Data Loader. When combined with one of DDC's hardware cards, the ARINC 615 Data Loader can be used to upload and download information to and from ARINC 615 compatible devices. This software application features discrete output triggering capabilities, simultaneous loading of multiple units, multiple user modes, raw-data time stamping capabilities, and advanced history logging which logs all of the received and transmitted signals as seen by the ARINC 615 Data Loader to a file.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PROGRAMMER'S REFERENCE

This chapter describes the operation of the ***DD-42924i5*** PCI card.

The ***DD-42924i5*** PCI card uses the Intel 80C186 microprocessor operating at 20 MHz. There are 256 Kbytes of Program RAM that is used for program storage and execution, booting the microprocessor and for simple diagnostic tests. There are 256 Kbytes of Data RAM. The card also contains two FIFOs, one for input and one for output, used for communicating commands and data to the card program. These are each 512 Bytes in capacity and are called the TO-PC FIFO and the FM-PC FIFO. Refer to Figure 1, which shows the block diagram for the ***DD-42924i5*** PCI card.

The card provides for up to four ARINC 429 receivers and up to two ARINC 429 transmitters. The board also supports five discrete inputs and five discrete outputs. The programmable ARINC 429 characteristics are controlled by the 80C186 microprocessor. When receiving ARINC words, the card may be programmed to send interrupts to the PC, or the PC can poll the ***DD-42924i5*** PCI card through the Data RAM.

TYPE AND STRUCTURE DEFENITIONS

The type, structure, and enumeration declarations are in a standard C header file called **TYPES429.H**. The user should take some time to look at this file and become familiar with its contents. This file will need to be included in your source code whenever you need to declare a structure to be passed to one of our library functions, or when you want to use one of the defined types. Header 'STDWNDEF.H' provides general and Windows specific type definitions. It too, is included in 'TYPES429.H'.

FUNCTION PROTOTYPES

The function prototypes for all functions in the DLL are in a standard C header file called **PROTO429.H**. There is no need to include this file in your source code. It is done automatically in **TYPES429.H**.

INITIALIZATION

In order for the library to function properly, some internal variables must be initialized, and the software that runs on the ***DD-42924i5*** PCI card must be loaded. In addition, the desired speed must be selected, and the transmit and receive channels must be initialized. The

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

following sections describe the functions that do this and how to use them.

INITIALIZING THE DLL FUNCTIONALITY

Prior to any other function calls from the DLL, the function **Initialize** must be called. This function must be called once for each ***DD-42924i5*** PCI card installed in the PC. It loads the executable hex file to the card, and runs power-up diagnostics (as directed by the user). When use of the card is no longer needed, the **Shutdown** function must be called in order to release system resources used by the card. Undesirable system behavior could result if this is not followed.

Prior to using a receiver or transmitter the **OpenChannel** function must be called. The **OpenChannel** function tells the toolbox what type of reception or transmission you plan to use on the channel (FIFO, Mailbox, Scheduled), and what parity the channel will be. You may call **OpenChannel** twice and set the transmitter channel for both FIFO and scheduled transmission. This means that you can set up a transmitter to handle both FIFO and scheduled transmissions at the same time. However, it is not possible to set up FIFO and Mailbox reception at the same time.

To disable a transmit or receive channel, the user must call the function **CloseChannel** or **CloseChannelAll**. This will stop all transmitting or receiving for a specified channel or all channels.

SETTING CHANNEL SPEED

There are two functions used to set the speed on each of the receive or transmit channels. The **SetChipSpeed** function and the **SetChannelSpeed** function should be used. This is normally done before calling the **OpenChannel** routine. All channels default to high speed. So if the user never calls these functions, all channels will operate at high speed.

MULTIPLE CARD FUNCTIONS

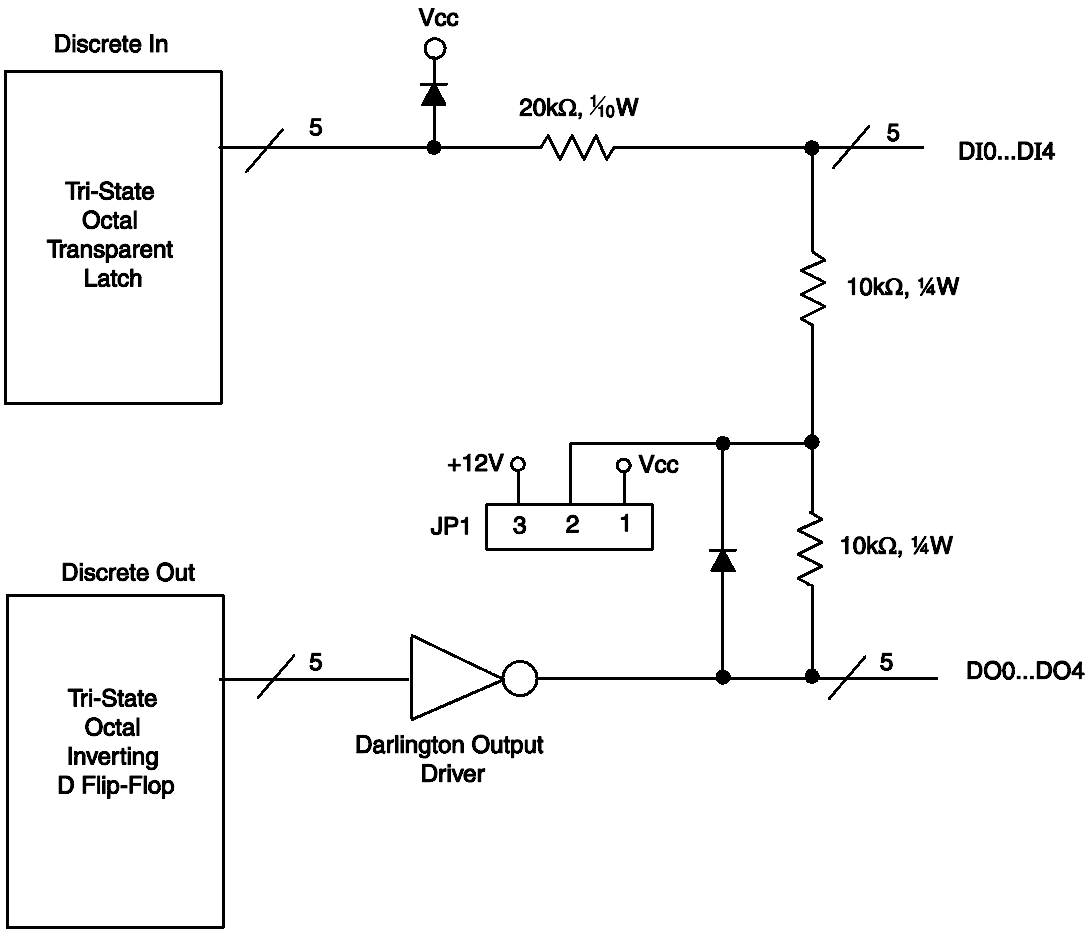
All DLL operations take place only on the currently active ***DD-42924i5*** PCI Card. If using more than one card, choose which card is active with the **SetCurrentCard** routine. **GetCurrentCard** will return the active card. The cards are numbered from one to eight if eight cards are being utilized. For example, when initializing the cards, the user

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

should set the current card to one, initialize it, set the current card to two, initialize it, etc. The current card is initially set to one.

DISCRETES

The ***DD-42924i5*** PCI card contains 5 CMOS compatible discrete inputs and 5 discrete outputs. The voltage level for the output discretes can be either +5 Volts or +12 Volts. Both output voltages are connected through a 10K Ohm pull-up resistor. See Figure 5 for a discrete schematic. Each output can sink up to 100 mA of current. Each input accepts any level between 3.5 and 40 volts as a "true" condition, and 0 to 1.35 volts as "false" condition. All inputs are CMOS compatible.



NOTE: The jumper is located on the top edge of the card near the front panel D-type connector.

Figure 5 - Discrete Schematic

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The discretes provide the additional capability to control external devices or indicate internal events, etc. Within the I/O word, bit 0 corresponds to discrete 0.



Note: *If a discrete input is read and is not connected to an actual signal (floating input), the discrete will be read as* ON.

READING AND WRITING DISCRETES

There are two basic routines for handling the discrete inputs and outputs on the card. The **WriteDiscrete** routine is used for setting the value (voltage) of a discrete output, and the **ReadDiscrete** routine is used for reading a discrete input.

TRANSMITTING

Before trying to transmit on the ***DD-42924i5*** PCI Card, the user should first have called the **Initialize** and **OpenChannel** functions described on page 26. Once the **OpenChannel** function has been called to define a transmitter, the library will set up a transmit FIFO queue that is 534 32-bit ARINC words in size. The DLL provides four functions for transmission of ARINC words, **Transmit_FIFO**, **Transmit_FIFO_Block**, **Transmit_Scheduled**, and **Transmit_Extended_Scheduled**. All these functions use the ArincRawWordType structure definition containing an ARINC label and its data. This structure is described further in **Section A.2** of Appendix A.



Note: *The ArincRawWordType structure definition is different for Microsoft and Borland C. For Microsoft C programmers this structure can be useful because it allows access to the bit fields in an ARINC word as fields in a C structure. For Borland C users the* ***ComposeArincWord*** *and* ***DecomposeArincWord*** *routines are provided.*

The function **Transmit_FIFO** takes a single ARINC word and transmits it once over the specified channel.

The function **Transmit_FIFO_Block** is very similar to **Transmit_FIFO**, except that it takes an array of ARINC words to transmit over a channel.

The function **Transmit_Scheduled** will transmit an ARINC word repeatedly at a particular update rate. The function **Kill_Scheduled**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

will stop transmission of an ARINC word that was previously set up by **Transmit_Scheduled**.

The function **Transmit_Extended_Scheduled** will transmit an ARINC word, with a unique SDI, repeatedly at a particular update rate. The function **Kill_Extended_Scheduled** will stop transmission of an ARINC word with a unique SDI that was previously set up by **Transmit_Scheduled**.

The **Get_Extended_Scheduled_Label_Value** routine is used to get the status of a label, specified by SDI, that has been previously configured for scheduled transmission.

RECEIVING

Before trying to receive on the ***DD-42924i5*** PCI Card, the user should first have called the **Initialize** and **OpenChannel** functions. Once the **OpenChannel** function has been called to define a receiver, the library will set up a receive FIFO queue that is 534 32-bit ARINC words in size. If more than 534 ARINC data words are sent to the receiver without calling one of the functions to receive the data (i.e., Receive_FIFO), then the next time the user attempts to dequeue the ARINC data using one of the receiver functions, the function will return DATA_WAS_LOST. The DLL provides ten functions associated with ARINC word reception, **AddLabel**, **AddLabelGroup**, **DeleteLabel**, **DeleteLabelGroup**, **DeleteAllLabel**, **Receive_Mailbox**, **Receive_FIFO**, **Receive_FIFO_Block**, **GetPendingArincWords**, and **Clear_FIFO**.

There are two basic methods of receiving ARINC data, the mailbox method, and the FIFO method. In the mailbox method, only the last received value for any label is stored by the DLL, and can be requested at any time by using the **Receive_Mailbox** routine. The mailbox method has two modes available: Extended and Normal. The **Receive_Mailbox** routine uses Normal mode.
Receive_Mailbox_Extended uses the Extended mode of operation allowing the user to check for only the last received value for any label, according to a specific SDI. The **Modify_Mailbox_Extended** routine may be used to switch between mailbox modes. For the FIFO method, all the labels that are being filtered through a single channel are placed in a queue, and can be dequeued through the **Receive_FIFO** and **Receive_FIFO_Block** routines. (See Figure 6)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

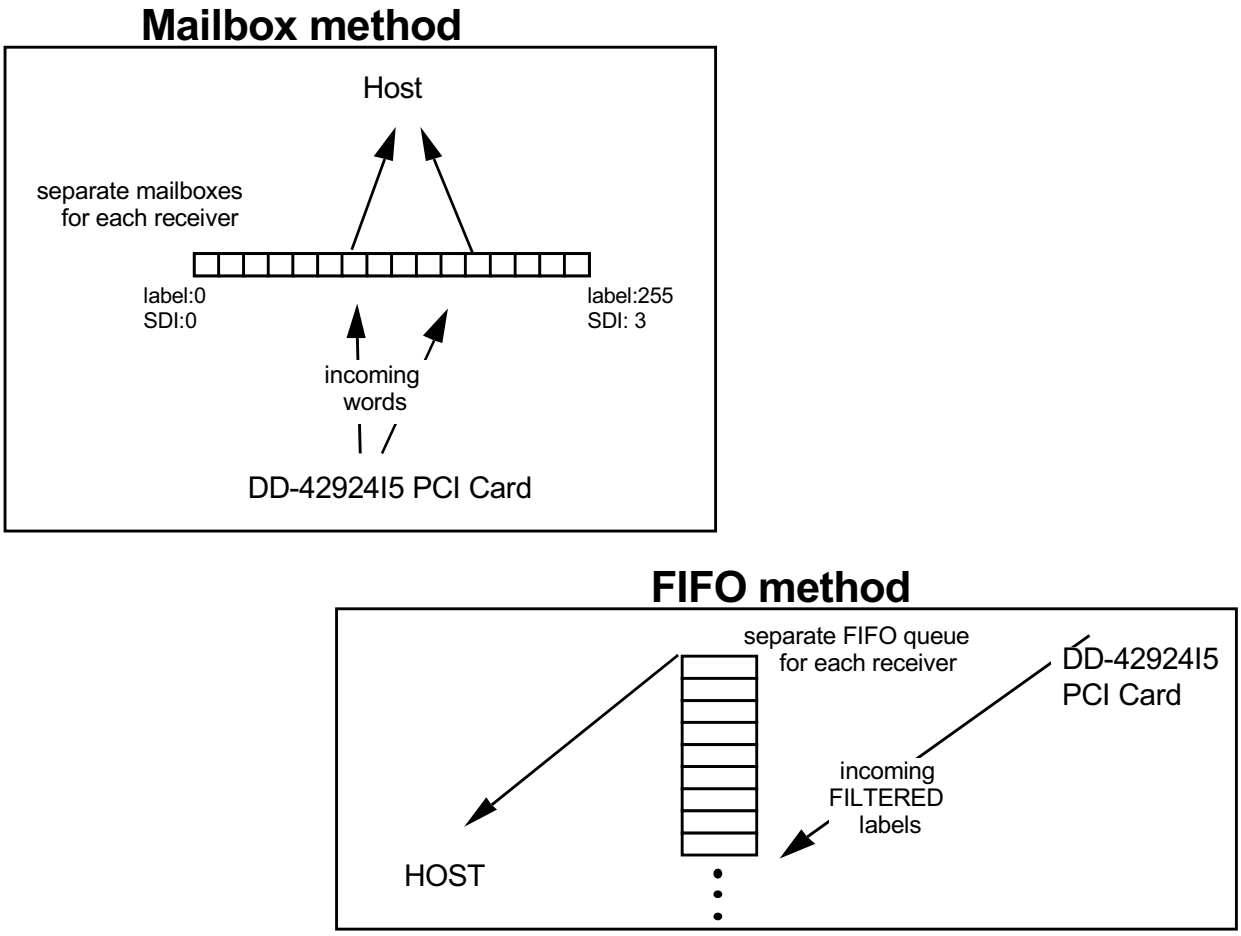


Figure 6 - Mailbox and FIFO Reception

When a receiver is set for FIFO queuing, filtering can be set to either on or off through the **OpenChannel** function. If the filtering feature is on, the functions **AddLabel**, **AddLabelGroup**, **DeleteLabel**, **DeleteLabelGroup**, and **DeleteAllLabel** can be used to add or delete labels from the current list of labels being filtered. The list of filter labels is empty until the first **AddLabel** or **AddLabelGroup** function is called (i.e., no words will be received).

For all receive data, the parity bit (bit #32) represents the parity as received by the ***DD-42924i5*** PCI card. A parity of zero means the parity of the received word was ODD, a one means it was EVEN.

RECEIVE_MAILBOX

The data returned from the **Receive_Mailbox** routine corresponds to the last data element received for the specified label. The software on

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

the ***DD-42924i5*** PCI card throws away any words found to have incorrect parity (there is no way for the software to determine what the original label was).

RECEIVE_FIFO AND RECEIVE_FIFO_BLOCK

The data returned from the **Receive_FIFO** routine will be the first word in the queue for the specified receiver channel. The software on the ***DD-42924i5*** PCI card will store any words found to have incorrect parity in the FIFO the same as if they had correct parity. It is up to the user to check the words returned from the **Receive_FIFO** routine to see that they had the correct parity.

The data returned from the **Receive_FIFO** function is a pointer to the ArincWordType union of structures. The purpose of this union is to handle the different types of data that can be returned from a receiver, based on how the **OpenChannel** had the KeepTimeInfo parameter set (i.e., whether timestamping was turned on or off). The ArincWordType union is described further in **Section A.2** of Appendix A.

The **Receive_FIFO_Block** routine is very similar to the **Receive_FIFO** routine, except that it dequeues a list of words from the queue for the specified receiver channel.

TIMERS AND TIMESTAMPS

When the DLL **Initialize** function is called, the user may configure a parameter called "Timer Resolution." The value given by this parameter affects system operation in a number of important ways.

The DLL uses the timer inside the 80186 microprocessor on the ***DD-42924i5*** PCI card to determine when to transmit scheduled data, what the time stamp should be for received data, and when to generate periodic interrupts to the PC. The "Timer Resolution" (an integer number of milliseconds) determines how often the timer interrupts the software code running on the card causing the software to transmit scheduled transmissions and perform periodic functions.

When using scheduled transmit data, the Update Rate for each transmitted word can only be an integer multiple of the "Timer Resolution." For example, if the "Timer Resolution" = 10ms, and the user requests an Update Rate of 12 ms, the word will be transmitted every 10 ms.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

When the DLL is being used to generate periodic interrupts to the PC (See **EnableGlobalCardToPCInt** function), the DLL can only generate periodic interrupts at integer multiples of the "Timer Resolution." For example, if the "Timer Resolution" = 10 ms, and the user requests a periodic interrupt every 12 ms, the interrupt will actually occur every 10 ms. Therefore, if the "Timer Resolution" is set to a low number, (two milliseconds for example), the scheduled transmission and periodic interrupts will be accurate to two milliseconds, but the performance of the DLL will decrease due to the fact that it will have to handle the timer interrupt more frequently. This means that the DLL may not be capable of handling all channels when they are fully loaded. For this reason, the user is given the capability to decrease the "Timer Resolution" to either 10 or 20 milliseconds. This will decrease the accuracy of scheduled transmissions and periodic interrupts, but will allow the DLL to handle higher bus loading (since it will be handling fewer timer interrupts).

TIME STAMP CALCULATIONS

Time stamp resolution for received data is initially set to zero. Time stamping begins when one of the following conditions occurs:

1. The **OpenChannel** function is called to start FIFO reception with time stamps.

2. The **OpenChannel** function is called to allow scheduled transmission.

3. The **EnableGlobalCardToPCInt** function is called to start periodic PC interrupts.

4. The **ZeroCardClock** function is called.

This time stamp can be reset to zero using the **ZeroCardClock** function. The current value of the timer can be read using the **GetCardCurrentTime** function. This is useful when the user wishes to synchronize multiple ***DD-42924i5*** PCI cards, or synchronize the time stamps with other equipment and timers. The time stamps on received words are given in a 32-bit portion HardwareStamp, and a 16-bit portion SoftwareStamp. The SoftwareStamp represents the number of times the 80186 timer has generated an interrupt (once every "Timer Resolution"). The HardwareStamp represents the value read from the 80186 internal timer (which increments every 0.20µ seconds). To get the time elapsed (in milliseconds) from a time stamp on a received word, use the following equation:

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Elapsed time = (HardwareStamp x 0.00020 milliseconds)
+ (SoftwareStamp x TimerResolution milliseconds)



16-bit not supported

Note: *On previous versions of the toolbox for the ISA cards, the hardware stamp was a 16-bit variable, as opposed to the 32-bit variable currently used for the PCI card. This change must be taken into consideration when porting software originally developed for the ISA card(s) to the new PCI card. Since the SoftwareStamp is a 16-bit unsigned short, it can represent 0..65,535. When it overflows it causes Time Stamp "roll-over" (the elapsed time suddenly goes from a large number to approximately 0). Since the value of the "Timer Resolution" affects how often the SoftwareStamp is incremented, it also affects how soon it will overflow. When the "Timer Resolution" is set to 2 ms, the timer will roll-over every 131.07 seconds (approximately 2.1845 minutes). When the "Timer Resolution" is set to 20 ms, the timer will roll-over every 1310.7 seconds (approximately 22 min). The time calculation functions described in the next section helps the user account for this "roll-over."*

TIME CALCULATION FUNCTIONS

There are three routines available to calculate time stamps on received words. These functions operate on received words only, and do NOT change the value of the timer used to time stamp data on the ***DD-42924i5*** PCI card. **Get429AbsoluteTime** returns the time difference between the supplied ARINC word and the timer clock enable time. **SetInitialAbsoluteTime** allows the user to set an additional offset (in milliseconds) for the timer clock. **Get429TimeDifference** returns the time difference between the two supplied ARINC words. **Get429AbsoluteTime** is designed to be called once for every word received in a stream of data. Since the time stamp can only represent times from 0 to 130 seconds (assuming a timer resolution of 2 msec), there is a "roll-over" (or overflow) that occurs approximately every 130 seconds. **Get429AbsoluteTime** has code which looks for this "roll-over" and compensates for it. It is possible to "fool" this code into thinking a "roll-over" has occurred when in fact it hasn't. If **Get429AbsoluteTime** were called twice with the same arguments, but at different times, unintended results like the following might occur:

if

T_1 = Get429AbsoluteTime (ArincWord$_1$);

T_2 = Get429AbsoluteTime (ArincWord$_2$);

...

T_{n-1} = Get429AbsoluteTime (ArincWord$_{n-1}$);

T_n = Get429AbsoluteTime (ArincWord$_1$);

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

then

$T_1 == T_n$ is not necessarily true

To avoid this, call **Get429AbsoluteTime** sequentially, i.e.:

T_1 = Get429AbsoluteTime (ArincWord$_1$);
T_2 = Get429AbsoluteTime (ArincWord$_2$);
...
T_{n-1} = Get429AbsoluteTime (ArincWord$_{n-1}$);
T_n = Get429AbsoluteTime (ArincWord$_n$);

GENERATING PC INTERRUPTS

The ***DD-42924i5*** PCI card is capable of generating PC Interrupts. The DLL and Driver can generate interrupts in one of two ways:

- A frequency-driven approach, where the PC is interrupted every "X" milliseconds.

- A reception-driven approach, where the PC is interrupted every time an ARINC word is received by one of a particular set of receivers, e.g., if the set of receivers is R0 & R2, a word received by either R0 or R2 will cause an interrupt to be sent to the PC.

The programmer can choose which mode to set up a card for; choosing the frequency to interrupt the PC in the frequency-driven case, or the particular set of receivers in the reception-driven case. The card must be properly enabled and disabled in order to ensure proper system operation.

See the descriptions for **EnableGlobalCardToPcInt** and **DisableGlobalCardToPcInt**.

DIAGNOSTIC FUNCTIONS

In addition to running diagnostics on start up using the **Initialize** function, there are routines that will run diagnostics at any time and routines that will determine the level of details of failure. The **RunDiagnostic** routine runs a particular diagnostic test and returns a pass/fail status. If an error occurred, specific return codes provide details pertaining to which test failed. For the **ROM_FIFO** test and **MEMORY** test, a data structure is updated with an address. Expected

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

data and actual data offer further details as to specifics associated with each test.

SHUTDOWN

There is a routine called **Shutdown** that allows the PC program to close all the channels and halt the program running on ***DD-42924i5*** PCI Card and release system resources. This function must be used when exiting a PC program.

USING PARITY AS DATA

The ***DD-42924i5*** PCI cards support a transmission mode called "ParityAsData". In this mode the data in the parity bit is preserved and transmitted (instead of being over-written by the parity computed on the Transceiver chip). This is done automatically by the software running on the card, and is transparent to the toolbox user. This feature operates on both FIFO and scheduled transmission data. See the function description for **EnableParityAsData**.

STATUS INQUIRY FUNCTIONS

There are several routines that return status or state information for the DLL.

GetChannelStatus will get the state of a channel that was previously initialized using the **OpenChannel** routine.
GetChipSpeed will get the current chip speed setting (used in conjunction with the **SetChipSpeed** function). The **GetChannelSpeed** function will get the current channel speed setting (used in conjunction with the **SetChannelSpeed** function).

The **GetPendingArincWords** routine will return the number of ARINC words waiting to be transmitted or the number of words in the receiver FIFO for a specified channel.

The **GetScheduledLabelValue** routine will return the status of a label that was previously setup for scheduled transmission.

The **GetTimerResolution** routine will return the resolution used for received word time stamping and scheduled transmissions that was passed to the **Initialize** routine.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

UTILITY FUNCTIONS

There are utility functions called **ComposeArincWord** and **DecomposeArincWord** that can be used to compose and decompose ARINC words from the label, data, SSM and Parity.

There is also a utility function called **StatusToAscii** that will take a status as returned from one of the DLL routines, and convert it to a string containing that status.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DLL HIGH-LEVEL FUNCTION DEFINITIONS

This chapter contains high-level DLL functions. High-level functions are defined as common instruction calls that will be utilized for board operations.

A detailed description of each DLL function, containing information about the routine's functionality, prototype, a formal parameter list, possible errors encountered, return codes, and example code will be contained in this section.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

AddLabel

Specifies a label that needs to be collected off of the ARINC bus.

PROTOTYPE

#include "types429.h"

STATUS **AddLabel** (ChannelType *Receiver*, LabelType *Label*);

PARAMETERS

Receiver	(input parameter) *ChannelType* value representing the desired receive channel number (*R0, R1, R2, R3*).
Label	(input parameter) *LabelType* value (*UCHAR*), label filter to be enabled.

DESCRIPTION

When a receive channel has been configured via the **OpenChannel** function, using the FIFO method with filtering is enabled, the **AddLabel** routine can be used to add a label to the set of currently active label filters. If the label was previously configured on the specified channel for filtering, the routine will have no effect and PC_SUCCESS will be returned. If the channel was not configured for filtering and FIFO, the routine will return BAD_ARGUMENT.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
if (AddLabel (R2, 190) == PC_SUCCESS)
  printf ("Label 190 successfully added to receive channel #2");
```

SEE ALSO

DeleteLabel()
DeleteAllLabel()
OpenChannel()
DeleteLabelGroup()
AddLabelGroup()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

AddLabelGroup

Specifies a group of labels that need to be collected off of the ARINC bus.

PROTOTYPE

#include "types429.h"

STATUS **AddLabelGroup** (ChannelType *Receiver*,
LabelType FAR **Labels*,
USHORT *NumOfLabels*);

PARAMETERS

Receiver	(input parameter) Receive channel number (*R0, R1, R2, R3*).
Label	(input parameter) Pointer to an array of label filters to be enabled.
NumOfLabels	(input parameter) Total number of labels to be added.

DESCRIPTION

The **AddLabelGroup** function works similarly to the **AddLabel** routine, except that it enables a group of labels to the selected receiver. Only when a receive channel has been configured via the **OpenChannel** function, using the FIFO method with filtering is enabled, can the **AddLabelGroup** routine be used to add a group of labels to the set of currently active labels. If the label was previously configured on the specified channel for filtering, the routine will have no effect and PC_SUCCESS will be returned. If the channel was not configured for filtering and FIFO, the routine will return BAD_ARGUMENT.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
LabelType R0_Labels[ ] = {0xaa, 160, 170, 180};              // a group of labels
if (AddLabelGroup (R0, R0_Labels, 4) == PC_SUCCESS)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
printf ("Label Group 'R0_Labels' successfully added to receive channel #0");
```

SEE ALSO

DeleteLabel()	DeleteLabelGroup()
DeleteAllLabel()	AddLabelGroup()
AddLabel()	Receive_FIFO()
OpenChannel()	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Clear_FIFO

Clears a receive channel FIFO queue

PROTOTYPE

#include "types429.h"

STATUS **Clear_FIFO** (ChannelType *Receiver*);

PARAMETERS

Receiver	(input parameter) Receive channel FIFO to be cleared (*R0, R1, R2, R3*).

DESCRIPTION

The **Clear_FIFO** routine allows the programmer to clear the Receiver's FIFO of all currently stored ARINC words. Each receiver has its own FIFO, so the user must specify which receiver FIFO to clear. If the channel is currently receiving data, the FIFO may contain data immediately after it is cleared, due to new data having been placed in it. If the channel hasn't been opened for FIFO reception, BAD_ARGUMENT will be returned.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
if (Clear_FIFO (R2) == PC_SUCCESS)
  printf ("Receive channel #2 FIFO queue has been cleared");
```

SEE ALSO

Receive_FIFO()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ClearPCInterrupt

Obsolete function.

PROTOTYPE

#include "types429.h"

STATUS **ClearPCInterrupt** (SHORT *CardNumber*);

PARAMETERS

CardNumber	(input parameter) Card number (*1-8*).

DESCRIPTION

Allows the PC to clear an "interrupt" sent from the 429 card.



Note: *Obsolete function. Nothing is performed if called this function is provided only for backward software compatibility reasons with older DDC-429 CLASSIC ISA boards. It serves no function with the current libraries or products.*

RETURN VALUE

PC_SUCCESS	operation successful

EXAMPLE

None

SEE ALSO

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Close429MsgWindow

Obsolete. This function has no effect when called.

PROTOTYPE

#include "types429.h"

VOID **Close429MsgWindow** (*VOID*);

PARAMETERS

None

DESCRIPTION

Obsolete, this function is provided only for backward software compatibility reasons with older DDC-429 CLASSIC ISA boards. It serves no function with the current libraries or products.

RETURN VALUE

VOID

EXAMPLE

None

SEE ALSO

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CloseChannel

Disables an active channel.

PROTOTYPE

#include "types429.h"

STATUS **CloseChannel** (ChannelType *Channel*);

PARAMETERS

Channel	(input parameter) Channel to be disabled (*R0, R1, R2, R3, T0, T1*).

DESCRIPTION

When a channel has been set up for transmitting or receiving using **OpenChannel**, the **CloseChannel** routine can be used to disable the channel. It can be reopened later by using **OpenChannel**. If the channel is not legal, BAD_ARGUMENT is returned.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
if (CloseChannel (T1) == PC_SUCCESS)
  printf ("Transmit channel #1 has been closed");
```

SEE ALSO

CloseChannelAll()	OpenChannel()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CloseChannelAll

Disables all active channels.

PROTOTYPE

#include "types429.h"

STATUS **CloseChannelAll** (VOID);

PARAMETERS

None

DESCRIPTION

When a series of channels have been set up for transmitting or receiving using **OpenChannel**, the **CloseChannelAll** routine can be used to disable all the active channels. They can later be reopened using **OpenChannel**. (PC_FAILURE will be returned if the host PC stops communicating with the DDC-429 Card.)

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

```
if (CloseChannelAll ( ) == PC_SUCCESS)
  printf ("All channels have been closed");
```

SEE ALSO

CloseChannel() OpenChannel()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ComposeArincWord

Formats data for transmission onto the ARINC 429 bus.

PROTOTYPE

#include "types429.h"

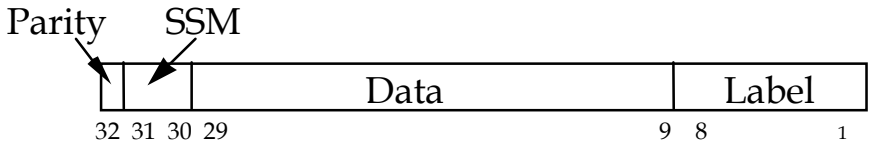
VOID **ComposeArincWord** (ArincRawWordType FAR **AW*,
LabelType *Label*,
ULONG *Data*,
UCHAR *SSM*);

PARAMETERS

AW	(output parameter) This pointer is a reference to the returned composed ARINC word.
Label	(input parameter) ARINC label to be added to the composed word.
Data	(input parameter) ARINC data word to be added to the composed word.
SSM	(input parameter) ARINC sign status matrix to be added to the composed word.

DESCRIPTION

The **ComposeArincWord** routine will take a label, data, and SSM, and compose an ARINC word so that can be passed to the DLL transmission routines. This word is composed according to the standard ARINC 429 format as shown below:



Note: *This routine is provided because Turbo C does not allow 32-bit fields. Microsoft C users can directly manipulate the fields in the ARINC word as they are defined in TYPES429.H.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURN VALUE

VOID

EXAMPLE

```
ArincRawWordType        TxAW[1];            // Raw data array for transmitter.
unsigned char           txSSM = 3;          // Sign Status Matrix
ULONG       txData =  0x1FFFFF;             // Transmit data.
UCHAR       txLabel = O130;                 // Octal Label number

ComposeArincWord (TxAW[1], txLbel,  txData, txSSM );
```

SEE ALSO

DecomposeArincWord()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Create429MsgWindow

Obsolete. This function has no effect when called.

PROTOTYPE

#include "types429.h"

BOOL **Create429MsgWindow** (SHORTINT *nHorizontal*,
SHORTINT *nVertical,*
SHORTINT *nWidth*
SHORTINT *nHeight*);

PARAMETERS

nHorizontal	(input parameter) Horizontal window position.
nVertical	(input parameter) Vertical window position.
nWidth	(input parameter) Width of window.
nHeight	(input parameter) Height of window.

DESCRIPTION

Obsolete, this function is provided only for backward software compatibility reasons with older DDC-429 CLASSIC ISA boards. It serves no function with the current libraries or products.

RETURN VALUE

TRUE	Function executed successfully
FALSE	Register window class failure

EXAMPLE

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DecomposeArincWord

Breaks raw ARINC data format to individual components.

PROTOTYPE

#include "types429.h"

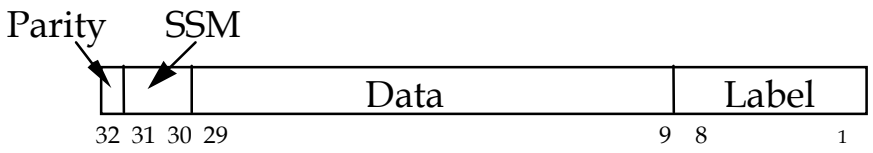
VOID **DecomposeArincWord** (ArincRawWordType FAR **AW*,
LabelType FAR **Label*,
ULONG FAR **Data*,
UCHAR FAR **SSM*,
ParityType FAR **Parity*);

PARAMETERS

AW	(input parameter) This pointer is a reference to the ARINC word to be decomposed.
Label	(output parameter) ARINC label returned.
Data	(output parameter) ARINC data word returned.
SSM	(output parameter) ARINC sign status matrix returned.
Parity	(output parameter) Pointer to word parity (0=FALSE, 1=TRUE) returned. If the expected parity was received then the following bit will be 1.

DESCRIPTION

The **DecomposeArincWord** routine will take an ARINC word that was returned from one of the DLL receive routines and break it down into its label, data, SSM, and parity. This word is decomposed according to the standard ARINC 429 format as shown below:



Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



Note: *This routine is provided because Turbo C does not allow 32-bit fields. Microsoft C users can directly manipulate the fields in the ARINC word as they are defined in TYPES429.H.*

RETURN VALUE

VOID

EXAMPLE

```
ArincRawWordType        RxAW[1];    // Raw data array for transmitter.
ULONG       RxData;                 // Transmit data.
UCHAR       RxLabel;                // Octal Label number
unsigned char       RxSSM;          // Sign Status Matrix
ParityType          R0Parity;       // Word Parity

DecomposeArincWord (&RxAW[1],  &RxLabel, &RxData, &RxSSM, &R0Parity);
```

SEE ALSO

ComposeArincWord () OpenChannel()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DeleteAllLabel

Removes all labels to be collected off the ARINC bus from active filtering.

PROTOTYPE

#include "types429.h"

STATUS **DeleteAllLabel** (ChannelType *Receiver*);

PARAMETERS

Receiver	(input parameter) Receive channel number the labels are being deleted from (*R0, R1, R2, R3*).

DESCRIPTION

When a receiver has been configured, via **OpenChannel,** with the FIFO method of receiving and filtering is enabled, the **DeleteAllLabel** routine can be used to delete all the labels in the set of currently filtered labels for a channel. If the channel was not set up for filtering and FIFO, the routine will return BAD_ARGUMENT. If no labels were previously set up on the specified channel for filtering, the routine will have no effect and PC_SUCCESS will be returned.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
if (DeleteAllLabel(R2 == PC_SUCCESS)
  printf ("All labels from R2's filter have been deleted");
```

SEE ALSO

DeleteLabel()	DeleteLabelGroup()
AddLabel ()	AddLabelGroup()
OpenChannel()	ReceiveFIFO()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DeleteLabel

Removes a label to be collected off the ARINC bus from active filtering.

PROTOTYPE

#include "types429.h"

STATUS **DeleteLabel** (ChannelType *Receiver*, LabelType *Label*);

PARAMETERS

Receiver	(input parameter) Receive channel number the labels are being deleted from (*R0, R1, R2, R3*).
Label	(input parameter) Label filter to be deleted.

DESCRIPTION

When a receiver has been configured, via **OpenChannel,** with the FIFO method of receiving and filtering is enabled, the **DeleteLabel** routine can be used to delete a label from the set of currently filtered labels. If the channel was not set up for filtering and FIFO, the routine will return BAD_ARGUMENT. If the label being deleted was not previously set up on the specified channel for filtering, the routine will have no effect and PC_SUCCESS will be returned.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
if (DeleteLabel (R0, 180) == PC_SUCCESS)
  printf ("Label 180 (decimal) has been deleted from R0's filter");
```

SEE ALSO

DeleteLabel()	DeleteLabelGroup()
AddLabel ()	AddLabelGroup()
OpenChannel()	ReceiveFIFO()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DeleteLabelGroup

Removes a group of labels to be collected off the ARINC bus from active filtering.

PROTOTYPE

#include "types429.h"

STATUS **DeleteLabelGroup** (ChannelType *Receiver*,
LabelType FAR **Labels*,
USHORT *NumOfLabel*);

PARAMETERS

Receiver	(input parameter) Receive channel number the labels are being deleted from (*R0, R1, R2, R3*).
Label	(input parameter) Pointer to an array of label filters to be deleted.
NumOfLabel	(input parameter) Number of labels in the array.

DESCRIPTION

The **DeleteLabelGroup** function works almost exactly like the **DeleteLabel** routine, except that it deletes a list of labels from the currently filtered labels for a receiver. The **DeleteLabelGroup** routine should only be used when a receiver has been configured, via **OpenChannel,** with the FIFO method of receiving and filtering is enabled. If the channel was not configured for filtering or FIFO, the routine will return BAD_ARGUMENT. If any of the labels being deleted were not previously set up on the specified channel for filtering, the routine will have no effect on those labels and PC_SUCCESS will be returned.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

EXAMPLE

```
LabelType     R3_LabelsToDelete[ ] = {190,200};        // A group of labels
                                                          (decimal)

printf("Delete a group of 2 labels from R3's filter\n");
  if(DeleteLabelGroup(R3,R3_LabelsToDelete, 2) == PC_SUCCESS)
        printf (“The Label Group has been deleted from R3's filter”);
```

SEE ALSO

DeleteLabel()
AddLabel()
OpenChannel()
Receive_FIFO_Block()
DeleteAllLabel ()
AddLabelGroup()
ReceiveFIFO()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DisableGlobalCardToPcInt

Halts periodic interrupts from the card.

PROTOTYPE

#include "types429.h"

STATUS **DisableGlobalCardToPcInt** (VOID);

PARAMETERS

None

DESCRIPTION

The **DisableGlobalCardToPcInt** routine stops the card from generating interrupts to the PC and terminates the interrupt threads.



Note: *It is imperative that the* ***DisableGlobalCardToPcInt*** *function be called when interrupts are no longer in use, when the card is being disabled with the shutdown function, or when the application is about to exit. The Enable/Disable must be adhered to in order to prevent system instability.*

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

```
DisableGlobalCardToPcInt();          // Disable card interrupts
```

SEE ALSO

EnableGlobalCardToPcInt ()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Display

Obsolete. This function has no effect when called.

PROTOTYPE

#include "types429.h"

STATUS **Display** (LPSTR StrToDisplay);

PARAMETERS

StrToDisplay	(input parameter) String containing message to be displayed

DESCRIPTION

Obsolete, this function is provided only for backward software compatibility reasons with older DDC-429 CLASSIC ISA boards. It serves no function with the current libraries or products.

RETURN VALUE

VOID

EXAMPLE

None

SEE ALSO

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

EnableGlobalCardToPcInt

Enables periodic interrupts from the card to the PC.

PROTOTYPE

#include "types429.h"

STATUS **EnableGlobalCardToPcInt** (SHORTINT *Interval*,
SHORTINT *UNUSED*,
SHORTINT *UNUSED*,
FARPROC *PC_ReceiverIntRoutine*);

PARAMETERS

Interval (input parameter)
Frequency of interrupts being generated.

UNUSED (input parameter)
This is an unused parameter. It is included in the function
As a place-holder for legacy software.

PC_ReceiverIntRoutine

(input parameter)
The user supplied interrupt handler routine.

DESCRIPTION

The **EnableGlobalCardToPcInt** routine will install the PC_ReceiverIntRoutine as an interrupt handler for the currently active card. If the PC_ReceiverIntRoutine is NULL, the **EnableGlobalCardToPcInt** routine will cause the toolbox to generate interrupts, but not install an interrupt routine. The **EnableGlobalCardToPcInt** routine features two different ways for the ***DD-42924i5*** card to generate interrupts to the PC: (1) interrupt the PC every <Interval> milliseconds, or (2) interrupt the PC whenever an ARINC word is received. Which mode is active is determined by the value of Interval:

If (Interval >= 0) then
Frequency-driven Interrupt Mode is enabled and the Interval = the frequency at which to interrupt the PC in milliseconds

else (Interval < 0)
Reception-driven Interrupt Mode is enabled and the least significant 4 bits of Interval, function as a bit mask for R0-R3

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

bit 0 corresponds to R0	R0_INT_ENABLED = 0x8001 (as in types429.h)
bit 1 corresponds to R1	R1_INT_ENABLED = 0x8002 (as in types429.h)
bit 2 corresponds to R2	R2_INT_ENABLED = 0x8004 (as in types429.h)
bit 3 corresponds to R3	R3_INT_ENABLED = 0x8008 (as in types429.h)

By setting more than one bit, interrupts can be enabled from more than one receiver, e.g., 0x8003 will interrupt the PC if a word is received at *R0* or *R1*.

In Frequency-driven mode, the Interval will be truncated to a multiple of the *TimerResolution* (specified by the **Initialize** routine). If the Interval < TimerRes., then Interval is set to TimerRes.



Note: *The* ***DisableGlobalCardToPcInt*** *function must be called when interrupts are no longer being used, when the card is being disabled with the shutdown function, or when the application is about to exit. The Enable/Disable must be adhered to in order to prevent system instability.*

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

```
// the following 2 lines are an example of a user defined global interrupt service
        routine.

void WINAPI IRQ_InterruptRoutine (DWORD card)      // 'card' number is passed
  NumOfInterrupts++;

// Enable timed interrupts from DD-42924i5 to PC with a 10mS period,
// and tell it to run the ISR defined above.

if(Code1 = EnableGlobalCardToPcInt(    10,
                                       (SHORTINT)NULL,
                                       (SHORTINT)NULL,
                                       (FARPROC)IRQ_InterruptRoutine) !=
                                            PC_SUCCESS)
{
        printf("\nPeriodic interrupt enable error.\n");
        return 1;
}
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SEE ALSO

DisableGlobalCardToPcInt ()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

EnableParityAsData

Allows the ARINC parity bit to be used as a data bit.

PROTOTYPE

#include "types429.h"

STATUS **EnableParityAsData** (ChannelType *Transmitter*,
PC_BOOLEAN *Switch*);

PARAMETERS

Transmitter	(input parameter) Transmit channel number to enable (*T0, T1*).
Switch	(input parameter) Parameter is used to enable or disable function (*PC_TRUE=ON, PC_FALSE=OFF*).

DESCRIPTION

The **EnableParityAsData** function sets the mode for the given transmitter channel. A PC_TRUE value means use "ParityAsData" mode, and a PC_FALSE value means use "normal" mode. The user can put different channels into different modes, and go back and forth between modes on any channel of their choosing in real-time. This feature operates on both FIFO and scheduled transmission data.

Upon initialization all transmitters default to normal mode, so the **EnableParityAsData** function is only necessary if "ParityAsData" mode is desired.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
// Set ParityAsData mode to TRUE on T0

EnableParityAsData(T0, PC_TRUE);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SEE ALSO

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Get429AbsoluteTime

Returns the difference between a Time Stamped ARINC word and the current card clock.

PROTOTYPE

#include "types429.h"

float **Get429AbsoluteTime** (register ArincWordType FAR **AW*);

PARAMETERS

AW	(input parameter) Pointer to a time stamped ARINC word. This must be declared as a *register* variable of type *ArincWordType*.

DESCRIPTION

Get429AbsoluteTime will return the difference in milliseconds between a time stamped ARINC word and the current card's clock enable time. **See Timers and Timestamps Section on Page 32.**

RETURN VALUE

float	Absolute time in milliseconds

EXAMPLE

```
// in this function, the relative time stamp which is calculated is the number of
// milliseconds since that same label was last received.

static ArincWordType PrevAW[256];      // holds previous words indexed by label
static ArincWordType AW;               // latest received ARINC word

#ifdef _MSC_VER
   fprintf (stdout, "%-7o 0x%06lX %10.2f %10.2f%s",
   ((int)AW->SR.fld.Label), ((long)AW->SR.fld.Data), Get429AbsoluteTime(AW),
   Get429TimeDifference (PrevAW+AW->SR.fld.Label, AW);

#else /* Borland compiler */
   Data[0] = '\0';
   PutDataInHexFormat (Data, AW->SR.fld.Data, 3);
   fprintf (stdout, "%-7o 0x%s %10.2f %10.2f%s",
   ((int)AW->SR.fld.Label), Data, Get429AbsoluteTime(AW),
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
Get429TimeDifference (PrevAW+AW->SR.fld.Label, AW);
#endif
```

SEE ALSO

Get429TimeDifference()	GetCardCurrentTime()
OpenChannel()	SetInitialAbsoluteTime()
ZeroCardClock()	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Get429TimeDifference

Return the difference between two time stamped ARINC words.

PROTOTYPE

#include "types429.h"

float **Get429TimeDifference** (ArincWordType FAR **PrevAW*,
ArincWordType FAR **CurAW*);

PARAMETERS

PrevAW	(input parameter) Pointer to the previous time stamped ARINC word.
CurAW	(input parameter) Pointer to the current time stamped ARINC *w*ord.

DESCRIPTION

Get429TimeDifference will return the time difference in milliseconds between the two supplied ARINC words as a floating-point number. See **See Timers and Timestamps Section on Page 32**.

RETURN VALUE

float	Absolute time in milliseconds

EXAMPLE

See **Get429AbsoluteTime** function for example.

SEE ALSO

Get429AbsoluteTime()	GetCardCurrentTime()
OpenChannel()	SetInitialAbsoluteTime()
ZeroCardClock()	

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Get429TimerResolution

Returns the current Timer resolution.

PROTOTYPE

#include "types429.h"

ResolutionType **Get429TimerResolution** (VOID);

PARAMETERS

None

DESCRIPTION

The **Get429TimerResolution** routine will return the resolution in milliseconds used for received word time stamping, frequency driver interrupts, and scheduled transmissions that was passed to the Initialize routine.

RETURN VALUE

This function returns the current Timer Resolution in number of milliseconds.

EXAMPLE

```
printf("The Card has been Initialized with a timer resolution of %d millisec\n",
                                        (int)Get429TimerResolution());
```

SEE ALSO

Initialize()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetCardCurrentTime

Returns the current hardware and software timer values of the card.

PROTOTYPE

#include "types429.h"

STATUS **GetCardCurrentTime** (USHORT FAR **SoftwareTime*,
ULONG FAR **HardwareTime*);

PARAMETERS

SoftwareTime	(output parameter) Pointer returns the board's current software timer value.
HardwareTime	(output parameter) Pointer returns the board's current hardware timer value.

DESCRIPTION

GetCardCurrentTime will return the current SoftwareTime and HardwareTime counters of the current DDC-429 Card's timer. See **See Timers and Timestamps Section on Page 32**.



Note: *The hardware stamp has changed from a 16-bit value for the ISA cards to a 32-bit value for the new PCI card. This change must be considered when porting software originally developed for the ISA cards to the new PCI card.*

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

```
unsigned short SoftwareTime;
unsigned long  HardwareTime;
GetCardCurrentTime(&SoftwareTime, &HardwareTime);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SEE ALSO

Get429AbsoluteTime()
Get429TimeDifference()
OpenChannel()
SetInitialAbsoluteTime()
ZeroCardClock()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetChannelSpeed

Returns current channel speed.

PROTOTYPE

#include "types429.h"

STATUS **GetChannelSpeed** (ChannelType *Channel*, SpeedType FAR **Speed*);

PARAMETERS

Channel	(input parameter) The desired channel (*R0, R1, R2, R3, T0, T1*) to return its current speed.
Speed	(output parameter) A pointer that returns the current speed of the selected channel.

DESCRIPTION

The **GetChannelSpeed** routine will return the current speed for a given channel.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument

EXAMPLE

```
SpeedType   Speed;          // return value for GetChannelSpeed
char        Buffer[40];     // STATUS word converted to ASCII text buffer

GetChannelSpeed(T0, &Speed);        //displays the integer value for Speed
   printf("T0 is set for STATUS %d\n", Speed));

StatusToAscii (Speed, Buffer);      //converts STATUS to a string
printf("T0 is set for %s\n", Buffer);       //displays the string value for Speed
```

SEE ALSO

SetChannelSpeed() SetChipSpeed()
GetChipSpeed()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetChannelStatus

Returns a current channel's configuration.

PROTOTYPE

#include "types429.h"

STATUS **GetChannelStatus** (ChannelType *Channel*,
PC_BOOLEAN FAR **Filter*,
PC_BOOLEAN FAR **KeepTimeInfo*,
PC_BOOLEAN FAR **FIFO*,
ParityType FAR **Parity*,
PC_BOOLEAN *Scheduled*);

PARAMETERS

Channel (input parameter)
Desired channel to check (*R0, R1, R2, R3, T0, T1*)

Filter (output parameter)
Pointer returns a PC_BOOLEAN type value of either *PC_TRUE* or *PC_FALSE*.
PC_TRUE is returned if label filters are in use, else *PC_FALSE* is returned.



Note: *This parameter only applies when FIFO is set to PC_TRUE.*

KeepTimeInfo (output parameter)
Pointer returns a PC_BOOLEAN type value of either *PC_TRUE* or *PC_FALSE*.
PC_TRUE is returned when time stamping of the incoming data is enabled, else *PC_FALSE* is returned.



Note: *This parameter only applies when FIFO is set to PC_TRUE.*

FIFO (output parameter)
Pointer returns a PC_BOOLEAN type value of either *PC_TRUE* or *PC_FALSE*.
PC_TRUE is returned when the received data is enqueued

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

in a FIFO. *PC_FALSE* is returned when only the last received data word of each label is stored in its associated mailbox position.

Parity (output parameter)
Pointer returns a PC_ ParityType value of either *EVEN_PARITY* if even data parity is enabled or *ODD_PARITY* if odd data parity is enabled

Scheduled (input parameter)
This parameter applies to transmit channels only.
The entries of ChannelStatus for scheduled transmission are above the FIFO (T0-Tn) at locations (Tn+1-T2n).
The user must pass PC_BOOLEAN type value to define if the channel was configured as scheduled.

PC_TRUE is passed if the transmit channel has been configured for scheduled data transmissions.
PC_*FALSE* is passed if the transmit channel has been configured in FIFO mode.

DESCRIPTION

GetChannelStatus will get the state of a channel that was previously initialized using the **OpenChannel** routine. Because the user can open a channel for both scheduled and FIFO transmission, the user must specify for transmitters whether the FIFO or scheduled set-up parameters are desired. For more information on what each of the parameters mean, For more detail see the function **OpenChannel**.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument

EXAMPLE

```
ParityType      Parity;
PC_BOOLEAN      Filter,  TimeStampingOn,  FIFO;
PC_BOOLEAN      Scheduled = PC_TRUE;
        // Scheduled = PC_TRUE only if the channel was previously configured for
                scheduled transmissions

GetChannelStatus(T0, &Filter, &TimeStampingOn, &FIFO, &Parity, Scheduled);

                // Filter has no meaning for Transmitters - it is ignored
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SEE ALSO

OpenChannel()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetChipSpeed

Returns a transceiver chip's current speed.

PROTOTYPE

#include "types429.h"

STATUS **GetChipSpeed** (ChipType *ChipId*, SpeedType FAR **Speed*);

PARAMETERS

ChipId	(input parameter) The desired transceiver chip to return its current speed. (*CHIP_A*, *CHIP_B*)
Speed	(output parameter) A pointer that returns the current speed of the selected transceiver chip.

DESCRIPTION

The **GetChipSpeed** routine will return the current speeds set for either Transceiver chip on the ***DD-42924i5*** card. This routine takes two parameters, a chip select and a pointer to the speed. For a discussion of which chips control which transmitters and receivers, see the **Setting Channel Speed Section on Page 26**.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument

EXAMPLE

```
SpeedType   Speed;          // return value for GetChipSpeed
char        Buffer[40];     // STATUS word converted to ASCII text buffer

GetChipSpeed(CHIP_A, &Speed);
StatusToAscii (Speed, Buffer);                  //converts STATUS to a string
  printf("Chip A (T0, R0 & R1) is set for %s\n", Buffer));
```

SEE ALSO

GetChannelSpeed() SetChannelSpeed()
SetChipSpeed()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetCurrentCard

Returns the currently selected card number.

PROTOTYPE

#include "types429.h"

UCHAR **GetCurrentCard** (VOID);

PARAMETERS

None

DESCRIPTION

The **GetCurrentCard** routine will return the active ***DD-42924i5*** card. The cards are numbered from one to eight, for support of up to eight cards.

RETURN VALUE

This function returns an *unsigned char* representing the currently selected card number (1-8). This card number is the value that corresponds to the device number in the control panel.

EXAMPLE

```
// This function will display the currently selected card

static char* CardArray[ ] = {"Card1","Card2", "Card3", "Card4"};
   fprintf(stdout,"%s: ",CardArray [GetCurrentCard( ) ]);
```

SEE ALSO

SetCurrentCard()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Get_Extended_Scheduled_Label_Value

Gets the status of a label that has been previously configured for scheduled transmission specified by label and SDI.

PROTOTYPE

#include "types429.h"

STATUS **Get_Extended_Scheduled_Label_Value**
(ChannelType *Transmitter*,
LabelType *Label*,
SDI_TYPE *SDI*,
ArincRawWordType FAR **ArincLabel*,
USHORT FAR **UpdateRate*);

PARAMETERS

Transmitter	(input parameter) Transmit channel being queried (*T0, T1*)
Label	(input parameter) Label being queried
SDI	(input parameter) The ARINC SDI to filter by. (*SDI_00, SDI_01, SDI_10, SDI_11, SDI_NONE*)
ArincLabel	(output parameter) Pointer to the ARINC label and associated data being transmitted on the given schedule
UpdateRate	(output parameter) Pointer to the Number of milliseconds between transmissions (_ timer resolution passed to the *Initialize* function and _ 60,000)

DESCRIPTION

The **Get_Extended_Scheduled_Label_Value** routine will return the value in the data word for a given label that was previously setup for scheduled transmission, based on a unique SDI bit. If no transmission is scheduled for the supplied Label, BAD_ARGUMENT is returned.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	no transmission was scheduled for the supplied label
PC_FAILURE	operation failed

EXAMPLE

```
SDI_TYPE            SDI = SDI_10;
ArincRawWordType    ARW;
USHORT UpdateRate;

if (GetScheduledLabelValue (T0, 3, SDI, &ARW, &UpdateRate) ==
                                                    PC_SUCCESS)
  sprintf (“SDI 2, Label 3 has an UpdateRate of %d msec\n",UpdateRate);
```

SEE ALSO

Transmit_Scheduled()	GetScheduledLabelValue()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetPendingArincWords

Returns the number of words in the FIFO waiting to be transmitted or received.

PROTOTYPE

#include "types429.h"

SHORT **GetPendingArincWords** (ChannelType *Channel*);

PARAMETERS

Channel	(input parameter) Channel being queried (*R0, R1, R2, R3, T0, T*1).

DESCRIPTION

The **GetPendingArincWords** routine will return the number of words waiting to be transmitted, or received, in the FIFO for any channel. This does not include scheduled transmissions.



Note: *As soon as the* ***GetPendingArincWords*** *routine returns, the card may begin adding, or removing, data to and from the queue. The number of pending words is therefore most accurate immediately after being returned from this routine.*

RETURN VALUE

This function returns a *short* corresponding to the number of ARINC words currently waiting in a transmit or receive FIFO. If a value less than 0 is returned this represents that an invalid function argument was passed in the parameter list. If a 0 is returned, then the queue is empty.

EXAMPLE

```
short    NumberOfWordsInFIFO;

NumberOfWordsInFIFO = GetPendingArincWords(R0);
fprintf(stdout, "R0's FIFO currently contains %d words\n",
                                                NumberOfWordsInFIFO);
```

SEE ALSO

Receive_FIFO() Transmit_FIFO()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetScheduledLabelValue

Returns label data from word that was configured for scheduled transmission.

PROTOTYPE

#include "types429.h"

STATUS **GetScheduledLabelValue** (ChannelType *Transmitter*,
LabelType *Label*,
ArincRawWordType FAR **ArincLabel*,
USHORT FAR **UpdateRate*);

PARAMETERS

Transmitter	(input parameter) Transmit channel being queried (*T0, T1*)
Label	(input parameter) Label being queried
ArincLabel	(output parameter) Pointer to the ARINC label and associated data being transmitted on the given schedule
UpdateRate	(output parameter) Pointer to the Number of milliseconds between transmissions (_ timer resolution passed to the *Initialize* function and _ 60,000)

DESCRIPTION

The **GetScheduledLabelValue** routine will return the value in the data word for a given label that was previously setup for scheduled transmission. If no transmission is scheduled for the supplied Label, BAD_ARGUMENT is returned.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	no transmission was scheduled for the supplied label

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

PC_FAILURE	operation failed

EXAMPLE

```
ArincRawWordType        ARW;
USHORT UpdateRate;

if (GetScheduledLabelValue (T0, 3, &ARW, &UpdateRate) == PC_SUCCESS)
  sprintf (“Label 3 has an UpdateRate of %d msec\n",UpdateRate);
```

SEE ALSO

Transmit_Scheduled() Get_Extended_Scheduled_Label_Value()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

GetTestResultStructAddr

Returns an indication of which self-test failed, if any.

PROTOTYPE

#include "types429.h"

TestResultStructType* **GetTestResultStructAddr** (VOID);

PARAMETERS

None

DESCRIPTION

The **TestResults** routine indicates which test failed. The structure members contain more details as to specific failing sequences or locations. Refer to the header file for explanations of failure codes.

RETURN VALUE

This function returns the address of a structure of type *TestResultStructType*.

EXAMPLE

```
// Get the address of the library 'TestResultStruct'
// for detailed analysis of diagnostics.

TestResultStructType      *TestResults;      // Pointer to test result structure.

TestResults = GetTestResultStructAddr();
```

SEE ALSO

RunDiagnostic()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Initialize

Initializes the card for operation.

PROTOTYPE

#include "types429.h"

STATUS **Initialize** (LPSTR *HexFile*,
LPSTR *Unused*,
FARPROC *Unused*,
FARPROC *Unused* ,
DiagnosticType *Diagnostic*,
ResolutionType *TimerResolution*,
CardType *Card*);

PARAMETERS

HexFile	(input parameter) The hex file name for the card executable program file. The default file name is "pciexec.hex" for the 2x4 pci cards.
Diagnostic	(input parameter) Type of diagnostic to perform at initialization.
TimerResolution	(input parameter) Resolution used for scheduled transmissions, frequency driver interrupts, and receiver time stamping.
Card	(input parameter) The card model to be initialized.

DESCRIPTION

Prior to any other function calls by the DLL, the **Initialize** function must be called. This will set up the internal variables for the Toolbox libraries, allocate system resources, download software to the ***DD-42924i5*** PCI Card, and initialize all channels to high speed.

This function utilizes seven parameters. The first parameter is a hex file name for the card executable program file. The second, third, and fourth parameters are no longer used, pass PC_NULL to all of these parameters. The fifth parameter allows the user to define what type of diagnostic tests to run at startup. The sixth argument designates the timer resolution. The final parameter specifies the model of DDC-429 Card that will be used. You must call shutdown

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

for each call to initialize to free system resources pertaining to a particular card when the card is no longer needed.

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

```
STATUS Code;

if ((Code = Initialize("pciexec.hex",
                       (LPSTR) PC_NULLF,
                       (FARPROC) PC_NULLF,
                       (FARPROC) PC_NULLF,
                       NO_CHECK,              // don't perform diagnostics
                       TEN_MILLI_SEC,         // timer resolution
                       TWO_TX_FOUR_RX_PCI_CARD
                                              // model number of DD-42924i5 card
     )) != PC_SUCCESS )
  {
                                              // convert returned STATUS code to string
          char Buf[40];
          StatusToAscii(Code, Buf);
          printf("Initialize FAILED returning %s\n", Buf);
          exit(0);
  }
```

SEE ALSO

Shutdown ()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Kill_Extended_Scheduled

Terminates transmission of an ARINC word that was previously set up by **Transmit_Extended_Scheduled** command with a unique SDI bit.

PROTOTYPE

#include "types429.h"

STATUS **Kill_Extended_Scheduled** (ChannelType *Transmitter*,
SDI_TYPE *SDI*,
LabelType *Label*);

PARAMETERS

Transmitter	(input parameter) The transmitter channel on which the transmission is being stopped (*T0, T1*)
SDI	(input parameter) The ARINC SDI to filter by. (*SDI_00, SDI_01, SDI_10, SDI_11, SDI_NONE*)
Label	(input parameter) The label being removed from transmission.

DESCRIPTION

The function **Kill_Extended_Scheduled** will stop transmission of an ARINC word that was previously set up by **Transmit_Extended_Scheduled**, given a unique SDI. If there was no scheduled transmission on the specified channel for the specified label, the routine will return PC_SUCCESS. If the channel is not a legal transmitter, or if the transmitter is not opened for scheduled transmission, the routine will return BAD_ARGUMENT.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
// Kill scheduled transmission with  label  3 and SDI 2
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
SDI_TYPE                SDI = SDI_10;
ArincRawWordType        ARW;

Kill_Scheduled(T0, SDI, 3);

if(GetScheduledLabelValue(T0, 3, &ARW, &UpdateRate) != PC_SUCCESS)
  printf("Label 3 has been deleted\n");
```

SEE ALSO

Kill_Scheduled_All() Transmit_Scheduled()
Kill_Scheduled()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Kill_Scheduled

Terminates transmission of an ARINC word that was previously set up by Transmit_Scheduled command.

PROTOTYPE

#include "types429.h"

STATUS **Kill_Scheduled** (ChannelType *Transmitter*, LabelType *Label*);

PARAMETERS

Transmitter	(input parameter) The transmitter channel on which the transmission is being stopped (*T0, T1*)
Label	(input parameter) The label being removed from transmission.

DESCRIPTION

The function **Kill_Scheduled** will stop transmission of an ARINC word that was previously set up by **Transmit_Scheduled**. If there was no scheduled transmission on the specified channel for the specified label, the routine will return PC_SUCCESS. If the channel is not a legal transmitter, or if the transmitter is not opened for scheduled transmission, the routine will return BAD_ARGUMENT.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
// Kill scheduled transmission with label  3

ArincRawWordType        ARW;

Kill_Scheduled(T0, 3);

if(GetScheduledLabelValue(T0, 3, &ARW, &UpdateRate) != PC_SUCCESS)
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
printf("Label 3 has been deleted\n");
```

SEE ALSO

Kill_Scheduled_All()
Kill_Extended_Scheduled()
Transmit_Scheduled()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Kill_Scheduled_All

Terminates all scheduled transmissions on a given channel.

PROTOTYPE

#include "types429.h"

STATUS **Kill_Scheduled_All** (ChannelType *Transmitter*);

PARAMETERS

Transmitter	(input parameter) The transmitter channel on which the transmission is being stopped (*T0, T1*)

DESCRIPTION

The **Kill_Scheduled_All** routine allows the programmer to stop all scheduled transmissions on a given transmitter.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
// kill all scheduled transmissions on channel T0

Kill_Scheduled_All(T0);
```

SEE ALSO

Kill_Scheduled() Transmit_Scheduled()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Modify_Mailbox_Mode

Modifies the current mode of operation for the Receive Mailbox.

PROTOTYPE

#include "types429.h"

STATUS **Modify_Mailbox_Mode** (ChannelType *Channel*,
Mailbox_Mode_Type *Mode*);

PARAMETERS

Channel (input parameter)
The receive channel (*R0, R1, R2, R3*) from which the FIFO is being read.

Mode (input parameter)
The desired Mailbox Mode for receiver operation. (*MBOX_NORMAL, MBOX_EXTENDED*)

DESCRIPTION

The **Modify_Mailbox_Mode** routine should be used when a receiver channel has been configured via the **Receive_Mailbox_Extended** function and the **Receive_Mailbox** operation is now desired, or visa-versa. The difference between the Mailbox modes is that the Extended mode filters its operation by a unique SDI bit, the Normal mode does not. This function may only be used if the **OpenChannel** command FIFO parameter set to PC_FALSE.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
// This routine switched the receiver Mailbox mode from NORMAL to EXTENDED
// operation for receive channel 2

Modify_Mailbox_Mode (R2, MBOX_EXTENDED);
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SEE ALSO

Receive_Mailbox() Receive_Mailbox_Extended()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

OpenChannel

Configures a channel for operation.

PROTOTYPE

#include "types429.h"

STATUS **OpenChannel** (ChannelType *Channel*,
PC_BOOLEAN *Filter*,
PC_BOOLEAN KeepTimeInfo,
PC_BOOLEAN *FIFO*,
ParityType *Parity*);

PARAMETERS

RECEIVER CONFIGURATION

Channel (input parameter)
Channel to open (*R0, R1, R2, R3*)

Filter (input parameter)
PC_TRUE is used if label filters are in desired, else *PC_FALSE* should be set.



Note: *This parameter only applies when FIFO is set to PC_TRUE.*

KeepTimeInfo (input parameter)
PC_TRUE is used when time stamping of the incoming data is desired, else *PC_FALSE* should be set.



Note: *This parameter only applies when FIFO is set to PC_TRUE.*

FIFO (input parameter)
PC_TRUE is used when the received data is to be enqueued in a FIFO.
PC_FALSE is used when only the last received data word of each label is to be stored in its associated mailbox position.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Parity (input parameter)
EVEN_PARITY is to be used if even data parity is desired, or *ODD_PARITY* is to be used if odd data parity is desired. If the desired parity as set by this parameter is received by the card, the 32nd bit of the receiver ARINC data word will be set to a value of 1 to indicate that the parity of the received ARINC data word mathces the receiver's parity parameter setting.

TRANSMITTER CONFIGURATION

Channel (input parameter)
Channel to open (*T0, T1*)

Filter (input parameter)
Not applicable for transmit channels.

KeepTimeInfo (input parameter)
Not applicable for transmit channels.

FIFO (input parameter)
PC_TRUE is used when non-scheduled transmissions are desired.
PC_FALSE is used when to scheduled transmissions are desired.

Parity (input parameter)
EVEN_PARITY is to be used if even data parity is desired, or *ODD_PARITY* is to be used if odd data parity is desired.

DESCRIPTION

To initialize a transmit or receiver channel, you must first call the function named **OpenChannel**. This must be called prior to using other functions which set up, send or receive from the channel. Once the OpenChannel function has been called to define a transmitter, the library will set up a transmit FIFO queue that is 534 32-bit ARINC words in size.

Transmitting: **OpenChannel** may be called twice for a single transmitter and set the FIFO parameter to PC_TRUE and then PC_FALSE. This means that you can set up a transmitter to handle both FIFO and scheduled transmissions at the same time. Once the OpenChannel function has been called to define a receiver, the library will set up a receive FIFO queue that is 534 32-bit ARINC words in size. If more than 534 ARINC data words are sent to the receiver without calling one of the functions to receive the data (i.e., Receive_FIFO), then the next time the user attempts to dequeue the ARINC data using one of the receiver functions, the function will return DATA_WAS_LOST.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Receiving: The data returned from a FIFO receiver will depend on how the receiver is initialized with the **OpenChannel** routine. If the KeepTimeInfo parameter is PC_TRUE, a time stamp will be stored for each word received in the ArincStampedRawWordType structure using the HardwareStamp and SoftwareStamp fields. The data returned is placed in a union of two possible structures called ArincWordType.

```
typedef union{
        ArincRawWordType            R;
        ArincStampedRawWordType     SR;
}ArincWordType;
```

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
INTERNAL_ERROR	an internal card error has occurred
PC_FAILURE	operation failed

EXAMPLE

```
// Channel T1 will be opened in FIFO mode using ODD Parity
//  without time stamping

If  OpenChannel (T1, PC_FALSE, PC_FALSE, PC_TRUE, ODD_PARITY)  ==
                                                        PC_SUCCESS )
   printf ("T1 is opened for FIFO Transmission ... \n");
```

SEE ALSO

CloseChannel() Receive_FIFO()
Receive_FIFO_Block()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

P429T_VerStr

Returns the current DLL name and software toolbox version.

PROTOTYPE

#include "types429.h"

LPSTR **P429T_VerStr** (VOID);

PARAMETERS

None

DESCRIPTION

The **P429T_VerStr** routine returns a pointer to a string containing the current DLL and software toolbox version number of the DD-42976SO-300, e.g., a pointer to a string containing a value of "PC429.DLL - Ver 1.0".

RETURN VALUE

This function returns a *char* * that points to string containing the current software toolbox version number.

EXAMPLE

```
printf("You have compiled with the : %s\n",
        P429T_VerStr());
```

SEE ALSO

P429TB_version() P429TBVersion()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

P429TB_version

Returns the current software toolbox version.

PROTOTYPE

#include "types429.h"

LPSTR **P429TB_version** (VOID);

PARAMETERS

None

DESCRIPTION

The **P429TB_version** routine returns a pointer to a string containing the version number of the DD-42976SO-300 Toolbox, e.g., a pointer to a string containing a value of "1.0".

RETURN VALUE

This function returns a *char* * that points to string containing the current software toolbox version number.

EXAMPLE

```
printf("You have compiled with PC 429 Toolbox version: %s\n",
        P429TB_version());
```

SEE ALSO

P429T_VerStr() P429TBVersion()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

P429TBVersion

Returns the current DLL name and software toolbox version.

PROTOTYPE

#include "types429.h"

BOOL **P429TBVersion** (LPSTR *lpVerNbr*);

PARAMETERS

lpVerNbr	(output parameter) The current DLL and software Toolbox version being used.

DESCRIPTION

The **P429TBVersion** routine outputs a LPSTR type variable *lpVerNbr* that returns a pointer to a string containing the current DLL and software toolbox version number of the DD-42976SO-300, e.g., a pointer to a string containing a value of "PC429.DLL - Ver 1.0".

RETURN VALUE

TRUE	Function executed successfully

EXAMPLE

```
LPSTR  lpVerNum;              // String pointer containing current DLL version

P429TBVersion(&lpVerNum);
printf("You have compiled with the : %s\n", lpVerNum);
```

SEE ALSO

P429TB_version() P429T_VerStr()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Panic

Obsolete. This function has no effect when called.

PROTOTYPE

#include "types429.h"

VOID **Panic** (LPSTR *Str*);

PARAMETERS

Str	(input parameter) Str contains an error message string.

DESCRIPTION

Obsolete, this function is provided only for backward software compatibility reasons with older DDC-429 CLASSIC ISA boards. It serves no function with the current libraries or products.

RETURN VALUE

VOID

EXAMPLE

None

SEE ALSO

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ReadDiscrete

Read the current value of a discrete input.

PROTOTYPE

#include "types429.h"

STATUS **ReadDiscrete** (DiscreteType *Discrete*, DiscreteValue FAR ***Value*);

PARAMETERS

Discrete	(input parameter) Discrete being read from (*DI0, DI1, DI2, DI3, DI4*).
Value	(output parameter) Pointer returns a *DiscreteValue* type representing the current value of a discrete input (*PC_ON, PC_OFF*).

DESCRIPTION

The **ReadDiscrete** routine is used for getting the value of one of the discrete inputs on the ***DD-42924i5*** PCI Card.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
// This example will read the current value of Discrete input 0

DiscreteValue Value;
ReadDiscrete(DI0, &Value);
printf ("The current value of DI0 is %d", Value);
```

SEE ALSO

WriteDiscrete()	ReadDiscreteQueue()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ReadDiscreteQueue

Returns the current values of all five discrete inputs in packed form.

PROTOTYPE

#include "types429.h"

STATUS **ReadDiscreteQueue** (StampedDiscreteType Far * *Values*);

PARAMETERS

Values	(output parameter) Stamped discrete structure to be updated.

DESCRIPTION

The Toolbox checks the state of the input discretes on a periodic basis, and whenever the input discretes change state, they are time-stamped and placed in a queue.

The **ReadDiscreteQueue** routine dequeues a time-stamped discrete sample from the discrete queue. The discrete sample contains a time-stamp and a packed word containing the state of all 5 of the input discretes.
If the operation is successful, the function returns PC_SUCCESS and parameter "value" is updated with the discrete sample. If the discrete queue is empty, the function returns QUEUE_IS_EMPTY status. If the queue has overflowed, the status DATA_WAS_LOST is returned indicating that the user has not been dequeueing the discrete samples fast enough.

RETURN VALUE

PC_SUCCESS	operation successful
DATA_WAS_LOST	data has not been collected from the discrete *q*ueue quickly enough, and a overflow has occurred
QUEUE_IS_EMPTY	the discrete queue is empty

EXAMPLE

```
// This routine will read the stamped discrete queue

StampedDiscreteType StampedDiscretes;
STATUS Code;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
switch(Code = ReadDiscreteQueue(&StampedDiscretes))
        {
        case PC_SUCCESS:
                PrintStampedDiscrete(StampedDiscretes);
                        // PrintStampedDiscrete is a dummy to print the queue
                break;

        case QUEUE_IS_EMPTY:
                printf("\nQueue is empty");
                break;

        case DATA_WAS_LOST:
                {
                char Buf[40];
                StatusToAscii(Code, Buf);
                printf("Error Code %s\n", Buf);
                }
        break;
        }
```

SEE ALSO

ReadDiscrete()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Receive_FIFO

Reads the next ARINC data word from the FIFO.

PROTOTYPE

#include "types429.h"

STATUS **Receive_FIFO** (ChannelType *Receiver*,
ArincWordType FAR * *ArincLabel*)

PARAMETERS

Receiver	(input parameter) The receive channel (*R0, R1, R2, R3*) from which the FIFO is being read.
ArincLabel	(output parameter) Pointer to an ARINC word being read from the FIFO.

DESCRIPTION

The **Receive_FIFO** routine should be used when the receiver channel has been setup with the **OpenChannel** command FIFO parameter set to PC_TRUE. With this routine, the data returned will be the first word in the FIFO queue for the specified receiver channel. (See **Receiving Section on Page 30**.) If **Receive_FIFO** is called when the receiver was not configured for FIFO, QUEUE_IS_EMPTY will be returned. If **Receive_FIFO** is not called often enough, the FIFO will overflow. When this happens, the next time **Receive_FIFO** is called, it will return DATA_WAS_LOST.
For discussion of how to use data returned in ArincWordType union, see discussion under **OpenChannel**.

RETURN VALUE

PC_SUCCESS	operation successful
DATA_WAS_LOST	data has not been collected from the FIFO quickly enough, and a FIFO overflow has occurred
QUEUE_IS_EMPTY	the receiver being called is not configured for FIFO
BAD_ARGUMENT	invalid argument

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

EXAMPLE

```
// check for reception on R0

ArincWordType      AW;
STATUS             Code;

Code=Receive_FIFO(R0, &AW);
printf("R0: ");
switch (Code) {
   case PC_SUCCESS:
        PrintArincWord2(&(AW.R), "\n");  // a dummy routine to print ARINC words
        break;

   case DATA_WAS_LOST:
        PrintArincWord2(&(AW.R), ", DATA_WAS_LOST\n");
        break;

   default:   printf(Code);
}
```

SEE ALSO

OpenChannel()	Receive_FIFO_Block()
Receive_Mailbox()	SetChannelSpeed()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Receive_FIFO_Block

Reads a block ARINC data words from the FIFO.

PROTOTYPE

#include "types429.h"

STATUS **Receive_FIFO_Block** (ChannelType *Receiver*,
ArincWordType FAR * *ArincLabels*,
USHORT *MaxNumofLabels*,
USHORT FAR * *LabelsReceived*);

PARAMETERS

Receiver	(input parameter) The receive channel (*R0, R1, R2, R3*) from which the FIFO is being read.
ArincLabels	(output parameter) Pointer to an array of ARINC words read from the FIFO.
MaxNumofLabels	(input parameter) Maximum number of words to be placed into the array.
LabelsReceived	(output parameter) Pointer to the total number of ARINC words dequeued from the FIFO.

DESCRIPTION

The **Receive_FIFO_Block** routine should be used when the receiver channel has been configured with the **OpenChannel** command FIFO parameter set to PC_TRUE. With this routine, the data returned will be the first group of words (oldest) in the FIFO queue for the specified receiver channel. If **Receive_FIFO_Block** is called when the receiver was not configured for FIFO, QUEUE_IS_EMPTY will be returned. If **Receive_FIFO** is not called often enough, the FIFO will overflow. When this happens, the next time **Receive_FIFO** is called, it will return DATA_WAS_LOST. If words are received but not as many as specified in the MaxNumofWords parameter, then the WordsReceived will be < MaxNumofWords and the **Receive_FIFO_Block** routine will return QUEUE_IS_EMPTY. For discussion of how to use data returned in ArincWordType union, see discussion under **OpenChannel** on **Page 90**.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURN VALUE

PC_SUCCESS	operation successful
DATA_WAS_LOST	data has not been collected from the FIFO quickly enough, and a FIFO overflow has occurred
QUEUE_IS_EMPTY	the receiver being called is not configured for FIFO or the total number of words dequeued Was less than the *MaxNumofWords* parameter
BAD_ARGUMENT	invalid argument

EXAMPLE

```
// This example will return the error STATUS in the variable ‘Code’
// It will read channel R0 and place a maximum of 8 ARINC words into array ‘AW’

Code=Receive_FIFO_Block(R0, AW, 8, &WordsReceived);
```

SEE ALSO

OpenChannel()	Receive_FIFO()
Receive_Mailbox()	SetChannelSpeed()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Receive_Mailbox

Returns the last received data word for a specific label.

PROTOTYPE

#include "types429.h"

STATUS **Receive_Mailbox** (ChannelType *Receiver*,
LabelType *Label*,
ArincRawWordType FAR * *ArincWord*);

PARAMETERS

Receiver	(input parameter) The receive channel (*R0, R1, R2, R3*) from which the Mailbox is being read.
Label	(input parameter) The ARINC label being queried.
ArincWord	(output parameter) Pointer to a structure of received ARINC words.

DESCRIPTION

The **Receive_Mailbox** routine should be used when the receiver channel has been configured with the **OpenChannel** command FIFO parameter set to PC_FALSE. With this routine, the data returned will be the last one received for the specified label. If no new data has been received since the previous call to **Receive_Mailbox**, or if **Receive_Mailbox** is called when the receiver is not configured for mailbox, BIN_IS_EMPTY will be returned.

RETURN VALUE

PC_SUCCESS	operation successful
BIN_IS_EMPTY	the receiver being called is not configured for Mailbox operation
BAD_ARGUMENT	invalid argument

EXAMPLE

// This routine checks receiver 0 for a received label of 160 (decimal)

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
ArincWordType      AW;
STATUS             Code;

if ((Code=Receive_Mailbox(R0, 160, &AW)) == PC_SUCCESS)
  PrintArincWord(&AW);          // a dummy routine to print ARINC words
else
  printf (Code);
```

SEE ALSO

Receive_FIFO()	Receive_FIFO_Block()
OpenChannel()	SetChannelSpeed()
Receive_Mailbox_Extended()	Modify_Mailbox_Mode()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Receive_Mailbox_Extended

Returns the last received data word for a specific label filtering by SDI.

PROTOTYPE

#include "types429.h"

STATUS **Receive_Mailbox_Extended** (ChannelType *Receiver*,
SDI_TYPE *SDI*,
LabelType *Label*,
ArincRawWordType FAR * *ArincWord*);

PARAMETERS

Receiver	(input parameter) The receive channel (*R0, R1, R2, R3*) from which the Mailbox is being read.
SDI	(input parameter) The ARINC SDI to filter by. (*SDI_00, SDI_01, SDI_10, SDI_11, SDI_NONE*)
Label	(input parameter) The ARINC label being queried.
ArincWord	(output parameter) Pointer to a structure of received ARINC words.

DESCRIPTION

The **Receive_Mailbox_Extended** routine should be used when the receiver channel has been configured with the **OpenChannel** command FIFO parameter set to PC_FALSE. With this routine, the data returned will be the last one received for the specified label. If no new data has been received since the previous call to **Receive_Mailbox_Extended**, or if **Receive_Mailbox_Extended** is called when the receiver is not configured for mailbox, BIN_IS_EMPTY will be returned.

RETURN VALUE

PC_SUCCESS	operation successful
BIN_IS_EMPTY	the receiver being called is not configured for Mailbox operation

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

BAD_ARGUMENT	invalid argument

EXAMPLE

```
// This routine checks receiver 0 for a received label of 160 (decimal)
// and an SDI of 3

SDI_TYPE        SDI = SDI_11;
ArincWordType   AW;
STATUS          Code;

if ((Code=Receive_Mailbox(R0, SDI, 160, &AW)) == PC_SUCCESS)
  PrintArincWord(&AW);          // a dummy routine to print ARINC words
else
  printf (Code);
```

SEE ALSO

Receive_Mailbox()	Modify_Mailbox_Mode()
Receive_FIFO()	Receive_FIFO_Block()
OpenChannel()	SetChannelSpeed()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RunDiagnostic

Perform card diagnostic tests.

PROTOTYPE

#include "types429.h"

TEST_RESULT **RunDiagnostic** (DiagnosticType *Diagnostic*,
CHAR FAR* *ConfigFile*,
CHAR FAR * *HexFile*,
CardType *Card*);

PARAMETERS

Diagnostic (input parameter)
Diagnostic test to be performed:

NO_CHECK	: Do not perform any tests.
ROM_FIFO_CHECK	: Test ROM and FIFOs.
MEMORY_CHECK	: Test dual port / program RAM.
LOOPBACK_CHECK	: Checks transceiver internally.
SELF_TEST	: Performs a loopback test on the ROM_FIFO.

ConfigFile (input parameter)
This parameter must pass a *NULL*, it is not a supported parameter for the ***DD-42924i5*** card.

HexFile (input parameter)
A pointer to the hex file name for the card executable program file PCIEXEC.HEX.

Card (input parameter)
The card model being tested.
(The ***DD-42924i5*** card type is *TWO_TX_FOUR_RX_CARD*)

DESCRIPTION

The **RunDiagnostic** routine allows the user to specify a particular test to be run and returns a detailed status. Certain tests update a data structure internal to the library, which is provided to offer details about a particular failure, failure location, data for memory tests, discrete input/output lines etc. A pointer to this structure is obtained by calling GetTestResultStruct().

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com



WARNING: Once this routine is called, the user must execute the entire process of initialization afterwards, starting with the call to **Initialize**. The diagnostic tests may destroy the initial configuration of the DDC-429 Card toolbox.



Note: *The MEMORY_CHECK takes several minutes to run.*

RETURN VALUE

This function will return a value of type *TEST_RESULT*. The following is a list of all possible returned values:

TEST_PASSED,
TEST_FAILED,
READ_CONFIG_FAILED,
STUCK_AT_FAULT_RAM_TEST_FAILED,
COUPLING_FAULT_RAM_TEST_FAILED,
INITIALIZE _FAILED,
BOOT_CODE_VERSION_READ_FAILED,
I186_WRITE_PCI_READ_SHARED_RAM_TEST_FAILED,
I186_WRITE_PCI_READ_PROGRAM_RAM_TEST_FAILED,
I186_RAM_TEST_FAILED,
BOOT_CODE_DISCRETES_TEST_FAILED,
STATUS_REGISTER_TEST_FAILED,
LOADER_FAILED,
R0_OPEN_FAILED,
R1_OPEN_FAILED,
TO_OPEN_FAILED,
R2_OPEN_FAILED,
R3_OPEN_FAILED,
T1_OPEN_FAILED,
CHIP_A_LOW_SPEED_INTERNAL_LOOPBACK_FAILED,
CHIP_A_HIGH_SPEED_INTERNAL_LOOPBACK_FAILED,
CHIP_B_LOW_SPEED_INTERNAL_LOOPBACK_FAILED,
CHIP_B_HIGH_SPEED_INTERNAL_LOOPBACK_FAILED.

EXAMPLE

```
// This example performs a MEMORY_CHECK diagnostic test

TEST_RESULT      Result;                 // Result of 'RunDiagnostic()'

Result = RunDiagnostic(    MEMORY_CHECK,
                           NULL,
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
                                "pciexec.hex",
                                TWO_TX_FOUR_RX_PCI_CARD);

    if( Result != TEST_PASSED ) {
            printf( "           MEMORY_CHECK diagnostic FAILED\n" );

    switch(Result) {
            case   READ_CONFIG_FAILED : {
                   printf("\n\tCould not read configuration information.\n");
                   break;
                   }

            case   STUCK_AT_FAULT_RAM_TEST_FAILED : {
                   printf("\n\t\"Stuck at fault\" test failed.\n");
                   printf("\t\tAddress:\t0x%x", TestResults->Address);
                   printf("\n\t\tExpect:\t0x%x", TestResults->Expect);
                   printf("\n\t\tActual:\t0x%x", TestResults->Actual);
                   break;
                   }

            case   COUPLING_FAULT_RAM_TEST_FAILED : {
                   printf("\n\t\"Coupling fault\" test failed.\n");
                   printf("\t\tAddress:\t0x%x", TestResults->Address);
                   printf("\n\t\tExpect:\t0x%x", TestResults->Expect);
                   printf("\n\t\tActual:\t0x%x", TestResults->Actual);
                   break;
                   }

            default         : {
                   printf("\nUnrecognized error");
                   break;
                   }
            }
    }
```

SEE ALSO

Initialize() GetTestResultStruct ()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SetChannelParity

Sets channel parity generation to ODD or EVEN.

PROTOTYPE

#include "types429.h"

STATUS **SetChannelParity** (ChannelType *Channel*, ParityType *Parity*);

PARAMETERS

Channel (input parameter)
Channel on which to set the parity (*R0, R1, R2, R3, T0, T1*).

Parity (input parameter)
Parity to set the channel to:

EVEN_PARITY	generate even parity.
ODD_PARITY	generate odd parity.

DESCRIPTION

The **SetChannelParity** routine will set the current parity generation for a given channel.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument

EXAMPLE

```
// set channel R0 for Even Parity generation

SetChannelParity(R0, EVEN_PARITY);
```

SEE ALSO

OpenChannel() EnableParityAsData()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SetChannelSpeed

Sets the speed of a given channel.

PROTOTYPE

#include "types429.h"

STATUS **SetChannelSpeed** (ChannelType *Channel*, SpeedType *Speed*);

PARAMETERS

Channel (input parameter)
Channel on which to set the speed (*R0, R1, R2, R3, T0, T1*).

Speed (input parameter)
Speed to set the channel to:

SELF_TEST_LOW_SPEED,
SELF_TEST_HIGH_SPEED,
LOW_SPEED_ALL
HI_SPEED_ALL
LOW_SPEED_R_HI_SPEED_T
HI_SPEED_R_LOW_SPEED_T

DESCRIPTION

The **SetChannelSpeed** routine will set the speed for a given channel. This routine takes two parameters: a channel and a speed.



Note: *Paired receivers (on same Transceiver chip) must always have the same speed. If they are assigned different speeds, both receivers will be set to the last assigned speed value. The receivers are paired as follows: R0 with R1, and R2 with R3.*

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
SetChannelSpeed(R0, HI_SPEED_ALL);   // set channel R0 to High Speed
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SEE ALSO

GetChannelSpeed()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SetChipSpeed

Sets the current speed of the selected ARINC transceiver.

PROTOTYPE

#include "types429.h"

STATUS **SetChipSpeed** (ChipType *ChipId*, SpeedType *Speed*);

PARAMETERS

ChipId (input parameter)
Chip on which to set the speed (*CHIP_A, CHIP_B*).

Speed (input parameter)
Speed to set the channel to:

SELF_TEST_LOW_SPEED,
SELF_TEST_HIGH_SPEED,
LOW_SPEED_ALL
HI_SPEED_ALL
LOW_SPEED_R_HI_SPEED_T
HI_SPEED_R_LOW_SPEED_T

DESCRIPTION

The **SetChipSpeed** function is used to set the chip speed for each of the transceiver chips on the ***DD-42924i5*** PCI card. While this routine is generally called before calling the **OpenChannel** routine, it can be called to change the speed of a chip at any time. This routine takes two parameters, a chip and a speed.

If using the self-test speeds, the Transceiver chips will be set up for a wrap-around test, so anything you transmitted on T0 will be received by R0 and R1, and anything transmitted on T1 will be received by R2 and R3. The chip speed defaults to high speed upon initialization.



Note: *When* ***SetChipSpeed*** *is called, all pending transmit and received data will be cleared.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
PC_FAILURE	operation failed

EXAMPLE

```
// CHIP_A controls channels R0, R1, T0.
// setting CHIP_A speed to HI_SPEED_ALL, sets all affected channels
// to high speed

SetChipSpeed(CHIP_A, HI_SPEED_ALL);
```

SEE ALSO

GetChannelSpeed() GetChipSpeed()
SetChannelSpeed()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SetCurrentCard

Sets the active card that is to take toolbox commands.

PROTOTYPE

#include "types429.h"

STATUS **SetCurrentCard** (UCHAR *CardNum*);

PARAMETERS

CardNum	(input parameter) Current Card number to be activated corresponding to the device numbers in the control panel.

DESCRIPTION

SetCurrentCard controls which DDC-429 Card is active. The cards are numbered from one to eight, for support of up to eight cards.



Note: *The old ISA library indexed cards from zero, not one. Take this into consideration when porting software originally developed for the ISA cards, to the new PCI card.*

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument

EXAMPLE

```
// set the current card to 1.
// Any following Toolbox commands will be issues to card 1.

SetCurrentCard(1);
```

SEE ALSO

GetCurrentCard()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SetDetailedDiagnostic

Obsolete. This is an unsupported function.

PROTOTYPE

#include "types429.h"

VOID **SetDetailedDiagnostic** (PC_BOOLEAN *Flag*);

PARAMETERS

Flag	(input parameter) Specifies whether or not the diagnostics will display detailed messages.

DESCRIPTION

Obsolete function, the **SetDetailedDiagnostic** routine has no effect when called.

RETURN VALUE

VOID

EXAMPLE

None

SEE ALSO

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SetDisplayFunction

Obsolete. Overwrites the default output handling routine.

PROTOTYPE

#include "types429.h"

VOID **SetDisplayFunction** (FARPROC *lpfnDisplayFunction*);

PARAMETERS

lpfnDisplayFunction	(input parameter) Pointer to a user defined message handling function.

DESCRIPTION

Obsolete function, the **SetDisplayFunction** routine has no effect when called.

RETURN VALUE

VOID

EXAMPLE

None

SEE ALSO

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SetErrorFunction

Obsolete. Overwrites the default error handling routine.

PROTOTYPE

#include "types429.h"

VOID **SetErrorFunction** (FARPROC *lpfnErrorFunction*);

PARAMETERS

lpfnErrorFunction	(input parameter) Pointer to a user defined error handling function.

DESCRIPTION

Obsolete function, the **SetErrorFunction** routine has no effect when called.

RETURN VALUE

VOID

EXAMPLE

None

SEE ALSO

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SetInitialAbsoluteTime

Sets an initial time (in milliseconds) for the timer clock.

PROTOTYPE

#include "types429.h"

VOID **SetInitialAbsoluteTime** (USHORT *TimeInMilliSeconds*);

PARAMETERS

TimeInMilliSeconds	(input parameter) The timer clock enable time value.

DESCRIPTION

SetInitialAbsoluteTime allows the user to set an initial time (in milliseconds) for the timer clock enable time. This routine can be used before **Get429AbsoluteTime** to initialize the timer clock to a value greater than 0 milliseconds. The **SetInitialAbsoluteTime** routine is NOT card sensitive, i.e., it will set the initial time for receive data from ALL the cards.

RETURN VALUE

VOID

EXAMPLE

```
// sets the initial start of the board timer to 100 milliseconds

SetInitialAbsoluteTime(100);
```

SEE ALSO

Get429AbsoluteTime() Get429TimeDifference()
OpenChannel()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Shutdown

Forces the host PC to halt execution of all ***DD-42924i5*** operations.

PROTOTYPE

#include "types429.h"

STATUS **Shutdown** (VOID);

PARAMETERS

None

DESCRIPTION

The **Shutdown** routine forces the PC program to halt the executive running on the DDC-429 PC Card, and to release system resources. When a particular card is no longer in use, this function must be called to free memory and interrupt resources. An initialize/shutdown paradigm must be adhered to in order to prevent undesirable system behavior. This function must be used when exiting a PC program.

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

```
// terminate all card processing

Shutdown();
```

SEE ALSO

Initialize()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

StatusToAscii

Converts a given functions returned *status* and converts it to ASCII text.

PROTOTYPE

#include "types429.h"

STATUS **StatusToAscii** (STATUS *Code*, LPSTR *Str*);

PARAMETERS

Code (input parameter)
Status returned by a Toolbox function.

Str (output parameter)
Pointer to a string containing the ACSII text conversion of the Status code. The string will contain one of the following responses:

PC_SUCCESS
QUEUE_IS_FULL
QUEUE_IS_EMPTY
SCHEDULED_IS_FULL
BIN_IS_EMPTY
BAD_ARGUMENT
INTERNAL_ERROR
TIME_OUT
DATA_WAS_LOST
PC_FAILURE

DESCRIPTION

The **StatusToAscii** routine will take a status as returned from one of the Toolbox routines, and convert it to a string containing that status. This routine would be used if the user wanted to print the status of a particular routine.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument

EXAMPLE

```
// This example checks the current channel speed of T0
// It returns the STATUS code in both decimal and converted String form
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
SpeedType   Speed;              // return value for GetChannelSpeed
char        Buffer[40];          // STATUS word converted to ASCII text buffer

GetChannelSpeed(T0, &Speed);          //displays the integer value for Speed
   printf("T0 is set for STATUS %d\n", Speed));

StatusToAscii (Speed, Buffer);        //converts STATUS to a string
printf("T0 is set for %s\n", Buffer);  //displays the string value for Speed
```

SEE ALSO

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ToolBoxReadConfigFile

Obsolete. This function has no effect when called.

PROTOTYPE

#include "types429.h"

STATUS **ToolBoxReadConfigInfo** (LPLONG *loadAddr*,
LPLONG *dpramAddr*,
LPLONG *ioAddr*,
LPINT *Unused*)

PARAMETERS

loadAddr	(input parameter) Host PC memory segment address the ***DD-42924i5*** Card's program RAM is located. (0xA000 to 0xD000)
dpramAddr	(input parameter) Host PC memory segment address the ***DD-42924i5*** Card's shared RAM is located. (0xA000 to 0xD000)
ioAddr	(input parameter) Base IO address of card (i.e., 0x300, 0x310, etc.)

DESCRIPTION

Obsolete, this function is provided only for backward software compatibility reasons with older DDC-429 CLASSIC ISA boards. It serves no function with the current libraries or products.

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Transmit_Extended_Scheduled

Configures a channel to transmit an ARINC word repeatedly at a particular update rate over a specified channel with a unique SDI bit.

PROTOTYPE

#include "types429.h"

STATUS **Transmit_Extended_Scheduled**
(ChannelType *Transmitter*,
USHORT *UpdateRate*,
SDI_TYPE *SDI*,
ArincRawWordType FAR * *ArincWord*);

PARAMETERS

Channel	(input parameter) Channel to transmit through (*T0, T1*).
UpdateRate	(input parameter) The number of milliseconds between transmissions (_ timer resolution passed to Initialize and _ 60,000).
SDI	(input parameter) The ARINC SDI to filter by. (*SDI_00, SDI_01, SDI_10, SDI_11, SDI_NONE*)
ArincWord	(input parameter) Pointer to an array of ARINC data to be transmitted.

DESCRIPTION

The function **Transmit_Extended_Scheduled** will transmit an ARINC word repeatedly at a particular update rate over a specified channel. This transmission is unique based on the SDI bit. The word transmitted is the structure ArincRawWordType as defined in the header file TYPES429.H. This structure is useful for C programmers because it allows access to the bit fields in an ARINC word as fields in a C structure. This structure is described further in **Section A.2** in Appendix A. In order to use this function, **OpenChannel** must have been called with the FIFO parameter set to PC_FALSE. If it was not, then BAD_ARGUMENT is returned. A maximum of 256 labels per channel can be scheduled at any one time. The UpdateRate should be a multiple of the timer resolution passed to the **Initialize** routine. If not, it will be truncated to the largest integer multiple that is less than the UpdateRate specified.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
SCHEDULED_IS_FULL	occurs when the FIFO, for a channel that is configured with scheduled transmits, is full. May also occur if the channel was configured with the FIFO parameter set to PC_FALSE.

EXAMPLE

```
// This example transmits a single ARINC word every 10 milliseconds
// The Transmitted data word will have an SDI of 3

ArincRawWordType AW;
SDI_TYPE         SDI = SDI_11;

ComposeArincWord(AW,    1, 0x1FFFFF, 3);          // Data: 7FFFFF
OpenChannel(T0, PC_FALSE, PC_FALSE, PC_FALSE, EVEN_PARITY);

Transmit_Scheduled(T0,  10, SDI, &AW[0]);
```

SEE ALSO

Transmit_FIFO() Transmit_FIFO_Block()
Transmit_Scheduled()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Transmit_FIFO

Transmits a single word through the FIFO.

PROTOTYPE

#include "types429.h"

STATUS **Transmit_FIFO** (ChannelType *Transmitter*,
ArincRawWordType FAR * *ArincWord*);

PARAMETERS

Channel	(input parameter) Channel to transmit through (*T0, T1*).
ArincWord	(input parameter) Pointer to the ARINC data to be transmitted.

DESCRIPTION

This function **Transmit_FIFO** takes a single ARINC word and transmits it once over the specified channel. The word transmitted is the structure ArincRawWordType as defined in the header file TYPES429.H. This structure is useful for C programmers because it allows access to the bit fields in an ARINC word as fields in a C structure. This structure is described further in **Section A.2** in Appendix A. In order to use this function, **OpenChannel** must have been called with the FIFO parameter set to PC_TRUE. If it was not, then BAD_ARGUMENT will be returned.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument
QUEUE_IS_FULL	FIFO queue is full

EXAMPLE

```
// This example composes a single ARINC word
// then transmits it through the FIFO

ArincRawWordType        AW;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
ComposeArincWord(AW,    1, 0x1FFFFF, 3);
OpenChannel(T0, PC_FALSE, PC_FALSE, PC_TRUE, EVEN_PARITY);

if((Transmit_FIFO(T0, AW)) == PC_SUCCESS) {
printf ("Single data word transmitted through FIFO \n");
```

SEE ALSO

Transmit_FIFO_Block() Transmit_Scheduled()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Transmit_FIFO_Block

Transmits a block of words through the FIFO.

PROTOTYPE

#include "types429.h"

STATUS **Transmit_FIFO_Block** (ChannelType *Transmitter*,
ArincRawWordType FAR * *ArincWords*,
USHORT *NumOfElem*,
USHORT FAR * *NumOfElemTransmitted*);

PARAMETERS

Channel (input parameter)
Channel to transmit through (*T0, T1*).

ArincWord (input parameter)
Pointer to an array of ARINC data to be transmitted.

NumOfElem (input parameter)
The number of ARINC words in the array.

NumOfElemTransmitted

(output parameter)
A pointer to the actual number of ARINC words transmitted.

DESCRIPTION

The function **Transmit_FIFO_Block** is very similar to **Transmit_FIFO**, except that it takes an array of ARINC words to transmit over a specified channel. The words transmitted are each an ArincRawWordType structure as defined in the header file TYPES429.H. This structure is useful for C programmers because it allows access to the bit fields in an ARINC word as fields in a C structure. This structure is described further in **Section A.2** in Appendix A. In order to use this function, **OpenChannel** must have been called with the FIFO parameters set to PC_TRUE. If it was not, then BAD_ARGUMENT is returned.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

QUEUE_IS_FULL	FIFO queue is full

EXAMPLE

```
// This example composes multiple ARINC words
//  then transmits them in a block through the FIFO

ArincRawWordType AW[20];

ComposeArincWord(AW,    1, 0x1FFFFF, 3);
ComposeArincWord(AW+1,  2, 0x0AAAAA, 2);  /* 1010 */
ComposeArincWord(AW+2,  3, 0x155555, 1);  /* 0101 */
ComposeArincWord(AW+3,  4, 0x0AAAAA, 3);
ComposeArincWord(AW+4,  5, 0x1ABCDE, 0);

OpenChannel(T1, PC_FALSE, PC_FALSE, PC_TRUE, EVEN_PARITY);

if((Transmit_FIFO_Block(T1, AW+5, 5, &NumOfElemTransmitted)) ==
                                                    PC_SUCCESS) {
printf ("Data word block transmitted through FIFO \n");
```

SEE ALSO

Transmit_FIFO()	Transmit_Scheduled()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Transmit_Scheduled

Configures a channel to transmit an ARINC word repeatedly at a particular update rate over a specified channel.

PROTOTYPE

#include "types429.h"

STATUS **Transmit_Scheduled** (ChannelType *Transmitter*,
USHORT *UpdateRate*,
ArincRawWordType FAR * *ArincWord*);

PARAMETERS

Channel	(input parameter) Channel to transmit through (*T0, T1*).
UpdateRate	(input parameter) The number of milliseconds between transmissions (_ timer resolution passed to Initialize and _ 60,000).
ArincWord	(input parameter) Pointer to an array of ARINC data to be transmitted.

DESCRIPTION

The function **Transmit_Scheduled** will transmit an ARINC word repeatedly at a particular update rate over a specified channel. The word transmitted is the structure ArincRawWordType as defined in the header file TYPES429.H. This structure is useful for C programmers because it allows access to the bit fields in an ARINC word as fields in a C structure. This structure is described further in **Section A.2** in Appendix A. In order to use this function, **OpenChannel** must have been called with the FIFO parameter set to PC_FALSE. If it was not, then BAD_ARGUMENT is returned. A maximum of 256 labels per channel can be scheduled at any one time. The UpdateRate should be a multiple of the timer resolution passed to the **Initialize** routine. If not, it will be truncated to the largest integer multiple that is less than the UpdateRate specified.

RETURN VALUE

PC_SUCCESS	operation successful
BAD_ARGUMENT	invalid argument

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SCHEDULED_IS_FULL	occurs when the FIFO, for a channel that is configured with scheduled transmits, is full. May also occur if the channel was configured with the FIFO parameter set to PC_FALSE.

EXAMPLE

```
// This example transmits a single ARINC word every 10 milliseconds

ArincRawWordType AW;

ComposeArincWord(AW,    1, 0x1FFFFF, 3);                // Data: 7FFFFF
OpenChannel(T0, PC_FALSE, PC_FALSE, PC_FALSE, EVEN_PARITY);

Transmit_Scheduled(T0,  10, &AW[0]);
```

SEE ALSO

Transmit_FIFO() Transmit_FIFO_Block()
Transmit_Extended_Scheduled()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

WriteDiscrete

Sets the value of a discrete output.

PROTOTYPE

#include "types429.h"

STATUS **WriteDiscrete** (DiscreteType *DiscreteNum*, DiscreteValue *Value*);

PARAMETERS

DiscreteNum	(input parameter) Discrete being written to (*DO0, DOI1, DO2, DO3, DO4*).
Value	(input parameter) The value to set a discrete output to (*PC_ON, PC_OFF*).

DESCRIPTION

The **WriteDiscrete** routine is used for setting the value of one of the discrete outputs on the ***DD-42924i5*** PCI card.

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

```
// This example sets Discrete Output 0 to 'OFF'

DiscreteValue Value=PC_OFF;

WriteDiscrete(DO0, PC_ON);
printf  ("Discrete output DO0 is set to %d", Value);
```

SEE ALSO

ReadDiscrete()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ZeroCardClock

Resets a cards internal time stamp clock to zero.

PROTOTYPE

#include "types429.h"

STATUS **ZeroCardClock** (VOID);

PARAMETERS

None

DESCRIPTION

ZeroCardClock allows the user to reset a ***DD-42924i5*** Card's internal time stamp clock to zero. When this function executes, schedule transmissions, periodic interrupts (scheduled through the **EnableGlobalCardToPcInt** routine), and FIFO TimeStamping may show signs of disruption because this function temporally disables the timer interrupts. Any disruption will be temporary, and everything should begin operating again as soon as the function returns.

If the receive FIFOs contain data when the **ZeroCardClock** function is called, there may be some data with time stamps relative to the old time, and some relative to the new time. This will cause the **Get429AbsoluteTime** function to malfunction. To avoid this, be sure to call the **Clear_FIFO** function after the **ZeroCardClock** call.

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

```
ZeroCardClock();
printf ("The card time stamp clock has been initialized to zero");
```

SEE ALSO

GetCardCurrentTime()
Get429AbsoluteTime()
Get429TimeDifference()
OpenChannel()
SetInitialAbsoluteTime().

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

DLL LOW-LEVEL FUNCTION DEFINITIONS

This chapter contains low-level DLL functions. Low-level functions are defined as hardware specific or special instruction calls that are needed for board operations. Low-level functions exist in order to facilitate board programming outside of the normal “High-level” software environment, when a more direct method of hardware programming is required.

A detailed description of each DLL function, containing information about the routine’s functionality, prototype, a formal parameter list, possible errors encountered, return codes, and example code will be contained in this section.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ArincToTransceiver

Converts a data word from ARINC format to Transceiver chip format.

PROTOTYPE

#include "types429.h"

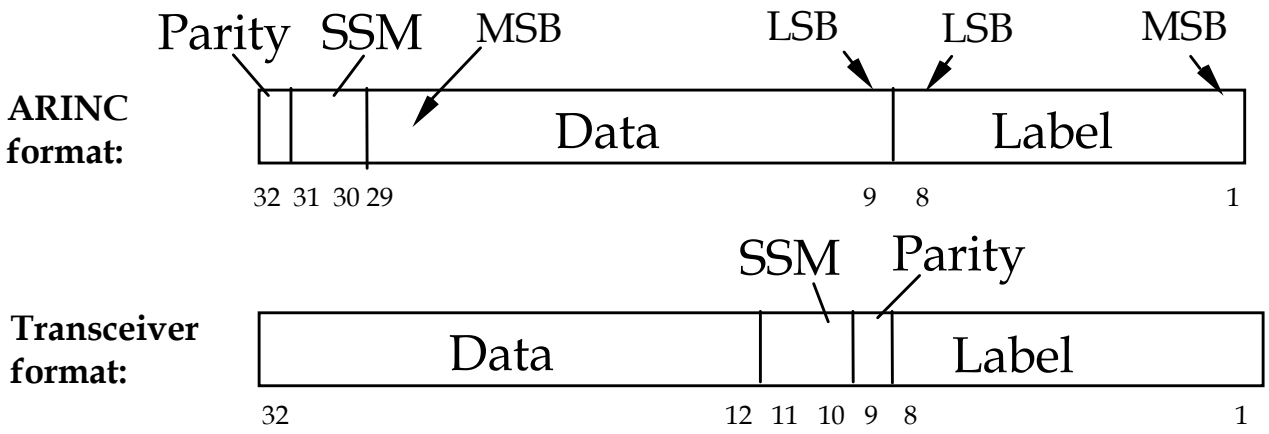
ULONG **ArincToTransceiver** (ULONG *Data*);

PARAMETERS

Data	(input parameter) 32-bit ARINC word to convert.

DESCRIPTION

This function **ArincToTransceiver** will convert an ARINC word to the format used by the Transceiver chips on the ***DD-42924i5*** Card. This function would be used when writing board code that will write words to the Transceiver chips directly.



RETURN VALUE

unsigned long	32-bit word converted to Harris Transceiver format

EXAMPLE

None.

SEE ALSO

TransceiverToArinc()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

IsBoardPresent

Checks to see if a ***DD-42924i5*** PCI card is presently installed.

PROTOTYPE

#include "types429.h"

PC_BOOLEAN **IsBoardPresent** (USHORT *SharedRam*);

PARAMETERS

SharedRam	(input parameter) Segment address base dual port RAM.

DESCRIPTION

This function **IsBoardPresent** will detect whether a ***DD-42924i5*** Card is present in the PC bus.



Note: *This is done automatically by the* ***Initialize*** *function.*

RETURN VALUE

PC_TRUE	a card was found
PC_FALSE	a card was not found

EXAMPLE

None.

SEE ALSO

Initialize()	Loader() StartCard()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Loader

Forces the host PC to halt execution of all ***DD-42924i5*** operations.

PROTOTYPE

#include "types429.h"

STATUS **Loader** (LPSTR *LoadFile*,
WORD *PC_Base*,
WORD *IO_Base*);

PARAMETERS

LoadFile (input parameter)
String containing name of hex file to be loaded.

PC_Base (input parameter)
Host PC memory segment address the ***DD-42924i5*** Card's program RAM is located. (0xA000 to 0xD000)

IO_Base (input parameter)
Base IO address (in PCI memory window) of card.

DESCRIPTION

This function **Loader** will load the ***DD-42924i5*** Card with an Intel formatted executable hex file. This function would be needed if a user had written a non-standard board library and needed to program the card manually.



Note: *The hex file used is loaded automatically by the Initialize function.*

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

None.

SEE ALSO

Initialize()
IsBoardPresent()
StartCard()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

StartCard

Forces the ***DD-42924i5*** into operational mode.

PROTOTYPE

#include "types429.h"

STATUS **StartCard** (USHORT *CardIOAddr*,
ResolutionType *TimerResolution*);

PARAMETERS

CardIOAddr (input parameter)
Base IO address (in PCI memory window) of card.

TimerResolution (input parameter)
The resolution used for scheduled transmissions and receiver time stamping. Available resolution:

TWO_MILLI_SEC
TEN_MILLI_SEC
TWENTY_MILLI_SEC

DESCRIPTION

The function **StartCard** will start the code executing on the ***DD-42924i5*** Card by making the EPROM jump to 0x00000 in the DDC-429 Card RAM. The program should then jump to location 0x00410 where the main program code is loaded. This allows the first 0x400 bytes to be reserved for interrupt vectors.

This function starts the code that was downloaded to the ***DD-42924i5*** PCI board. This means the utility that was used to convert the executable file to the "ROMable" hex file must locate the entry point of the program to 0x00410. This function would be used when the loader has been called by the user rather than using the standard **Initialize** function.



Note: *The* ***Initialize*** *function performs* ***StartCard*** *automatically.*

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

EXAMPLE

None.

SEE ALSO

Initialize()
IsBoardPresent()
Loader()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TimeDelay

Delays host PC processing of all ***DD-42924i5*** operations for a given period.

PROTOTYPE

#include "types429.h"

VOID **TimeDelay** (LONG *Wait*);

PARAMETERS

Wait	(input parameter) The number of milliseconds to delay processing. Accurate to within 1 millisecond.

DESCRIPTION

The **TimeDelay** function will delay processing on the PC for a given length of time. This would be used while waiting for the card to respond to ROM-FIFO commands.

RETURN VALUE

VOID

EXAMPLE

```
// wait 5 milliseconds

TimeDelay(5);
```

SEE ALSO

None

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

ToolBoxReadConfigInfo

Returns the values for the currently selected card from the Kernel driver

PROTOTYPE

#include "types429.h"

STATUS **ToolBoxReadConfigInfo** (LPLONG *progAddr*,
LPLONG *dpramAddr*,
LPLONG *ioAddr*,
LPINT *NULL*);

PARAMETERS

progAddr	(output parameter) Segment address for base program RAM.
dpramAddr	(output parameter) Segment address base dual port RAM.
ioAddr	(output parameter) Base IO address (in PCI memory window) of card.

DESCRIPTION

The function **ToolBoxReadConfigFile** will return the values for the currently selected card from the Kernel driver. This function is needed if the user needs to get card info. **Initialize** uses this function automatically.

RETURN VALUE

PC_SUCCESS	operation successful
PC_FAILURE	operation failed

EXAMPLE

USHORT ProgRam, DPRAM, CardIO, IRQ;
ToolBoxReadConfigInfo ((LPINT) & ProgRam, (LPINT) & DPRAM, (LPINT) & CardIO, (LPINT) NULL;

SEE ALSO

Initialize()	IsBoardPresent()
Loader()	StartCard()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TransceiverToArinc

Converts a data word from Transceiver chip format to ARINC format.

PROTOTYPE

#include "types429.h"

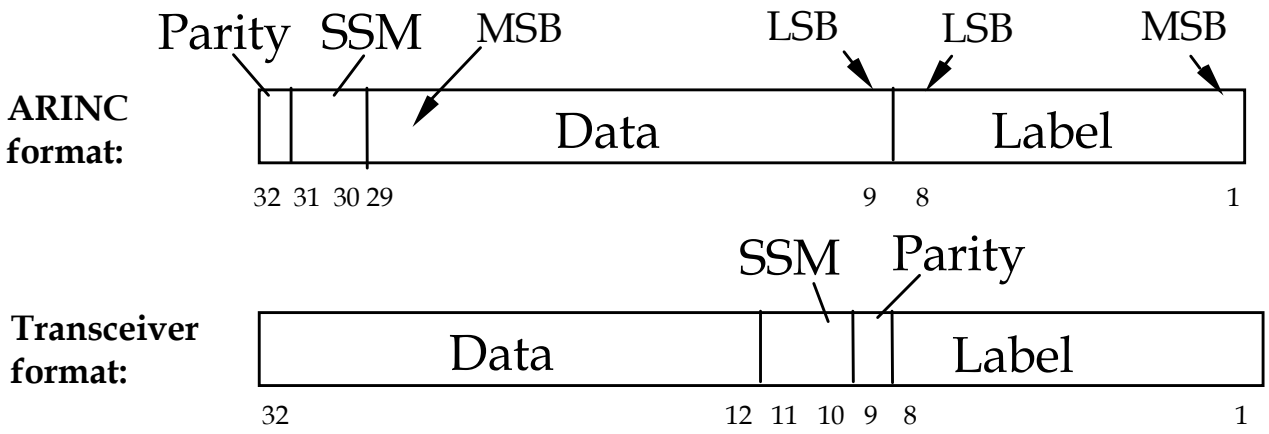
ULONG **TransceiverToArinc** (ULONG *Data*);

PARAMETERS

Data	(input parameter) 32-bit Harris word to convert.

DESCRIPTION

This function **TransceiverToArinc** will convert a word in the format used by the Transceiver chips to standard ARINC format. This function would be used when writing board code that will read words from the Transceiver chips directly.



RETURN VALUE

unsigned long	32-bit word converted to ARINC format

EXAMPLE

None.

SEE ALSO

ArincToTransceiver()

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX A

TYPE DEFINITIONS

The following section contains the structure and enumeration definitions that are used by the Windows DLL. This information is contained in TYPES429.H. **Section A.1** contains the enumeration's in alphabetical order, and **Section A.2** contains the structures, also in alphabetical order.

A.1 ENUMERATION/CONSTANT DEFINITIONS

```
#define PC_NULLF ((void (*)()) 0)

          // Bit Mask for EnableGlobalCardToPcInt()
#define R0_INT_ENABLED        0x8001   /* ARINC word received on R0 */
#define R1_INT_ENABLED        0x8002   /* ARINC word received on R1 */
#define R2_INT_ENABLED        0x8004   /* ARINC word received on R2 */
#define R3_INT_ENABLED        0x8008   /* ARINC word received on R3 */

#define TWO_MILLI_SEC         2
#define TEN_MILLI_SEC         10
#define TWENTY_MILLI_SEC      20    /* Max permissible resolution */

typedef enum {
  TWO_TX_FOUR_RX_CARD,
  FOUR_TX_FOUR_RX_CARD,
  EIGHT_TX_FOUR_RX_CARD,
  ONE_TX_TWO_RX_CARD,
  TWO_TX_FOUR_RX_HALF_CARD,
  ONE_TX_TWO_RX_HALF_CARD,
  TWO_TX_FOUR_RX_PCI_CARD,
  INVALID_CARD,
FORCE_TO_2_BYTES3=32767              /* Required for LabWindows */
} CardType;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
typedef enum {
    R0,     /* receiver  0 */
    R1,     /* receiver  1 */
    R2,     /* receiver  2 */
    R3,     /* receiver  3 */
    T0,     /* transmitter 0 */
    T1,     /* transmitter 1 */
    T2,     /* transmitter 2 */
    T3,     /* transmitter 3 */
    T4,     /* transmitter 4 */
    T5,     /* transmitter 5 */
    T6,     /* transmitter 6 */
    T7,     /* transmitter 7 */
    INVALID_CHANNEL,
    FORCE_TO_2_BYTES8=32767
} ChannelType;

typedef enum { CHIP_A, CHIP_B,
    FORCE_TO_2_BYTES11=32767        /* Required for LabWindows */
ChipType;

typedef enum {
    NO_CHECK,                   /* do not run diagnostics */
    ROM_FIFO_CHECK,             /* test ROM and FIFOs  */
    MEMORY_CHECK,               /* checks the dual port and program RAM */
                                /* checks for stuck at faults and coupling faults */
    LOOPBACK_CHECK,             /* checks transceiver internally */
    SELF_TEST                   /* run ROM_FIFO test and LOOPBACK */
    FORCE_TO_2_BYTES12=32767    /* Required for LabWindows */
} DiagnosticType;

typedef enum {
    DI0,    /* discrete input 0 */
    DI1,    /* discrete input 1 */
    DI2,    /* discrete input 2 */
    DI3,    /* discrete input 3 */
    DI4,    /* discrete input 4 */
    DO0,    /* discrete output 0 */
    DO1,    /* discrete output 1 */
    DO2,    /* discrete output 2 */
    DO3,    /* discrete output 3 */
    DO4,    /* discrete output 4 */
    INVALID_DISCRETE,
    FORCE_TO_2_BYTES9=32767         /* Required for LabWindows */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
} DiscreteType;
typedef enum {
 PC_OFF,
 PC_ON
 FORCE_TO_2_BYTES5=32767            /* Required for LabWindows */
 } DiscreteValue;


//   This type is used to change the mode of the mailbox for
allocating
//  slots for mailbox slots.


typedef enum {
  MBOX_NORMAL,      /* label is used to determine mailbox slot */
  MBOX_EXTENDED     /* sdi are and label used to determine mailbox
slot */
} Mailbox_Mode_Type;

typedef enum {
PC_FALSE,
PC_TRUE,
FORCE_TO_2_BYTES4=32767            /* Required for LabWindows */
} PC_BOOLEAN;

typedef enum {
EVEN_PARITY,
ODD_PARITY,
FORCE_TO_2_BYTES7=32767            /* Required for LabWindows */
} ParityType;

//  This type enumerates the values used for specifying the sdi for
a
//  label.

typedef enum {
  SDI_00,
  SDI_01,
  SDI_10,
  SDI_11,
  SDI_NONE
} SDI_Type;

typedef enum {
   SELF_TEST_LOW_SPEED,
   SELF_TEST_HIGH_SPEED,
   LOW_SPEED_ALL,
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    HI_SPEED_ALL,
    LOW_SPEED_R_HI_SPEED_T,          /* low speed receivers, high speed transmitters
*/
    HI_SPEED_R_LOW_SPEED_T,          /* high speed receivers, low speed transmitters
*/
    INVALID_SPEED,                   /* invalid speed  */
    FORCE_TO_2_BYTES7=32767          /* Required for LabWindows */
   } SpeedType;

typedef enum {
     PC_SUCCESS,
     QUEUE_IS_FULL,
     QUEUE_IS_EMPTY,
     SCHEDULED_IS_FULL,          /* same as QUEUE_IS_FULL, but applies only to
                                 */
                                 /* transmitters whose FIFO' flag are set to false
                                  */
                                 /* (transmitters which are configured as scheduled)
                                 */
     BIN_IS_EMPTY,               /* same as QUEUE_IS_EMPTY, but applies only
                                  to  */
                                 /* receivers whose FIFOs' flag are set to false  */
                                 /* (receivers in which only the most recent data */
                                 /* are kept, i.e., Mailbox mode)          */
     BAD_ARGUMENT,
     INTERNAL_ERROR,
     TIME_OUT,                   /* happens if there are some board problems */
                                 /*  Error handling routine would be called
                                automatically. */
     DATA_WAS_LOST,              /* happens when a receiver's FIFO in DPRAM */
                                 /* overflows and data which hasn't been dequeued
                                */
                                 /* (by the PC) is over- written (by the card).  */
     PC_FAILURE                  /* to handle cases where the card or a file is not
                                  */
                                 /*  present,  etc.  -  i.e.,  when  the
                                INTERNAL_ERROR */
                                 /* doesn't really apply. */
     FORCE_TO_2_BYTES10=32767    /* Required for LabWindows */

} STATUS;

typedef enum {
  TEST_PASSED,                       // This is the original configuration
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  TEST_FAILED,                          // along with the LabWindows enumerator
below.
  READ_CONFIG_FAILED,                   // Error reading card information from the
                                           registry.
  STUCK_AT_FAULT_RAM_TEST_FAILED,              // RAM test failure.
  COUPLING_FAULT_RAM_TEST_FAILED,              // RAM test failure.
  INITIALIZE_FAILED,                           // Call to 'Initialize ()' failed.
  BOOT_CODE_VERSION_READ_FAILED,     // 186 failed to boot.  Couldn't read the
                                        bootcode
                                      // version information from the TO-PC
                                        FIFO after
                                      // reset.

  I186_WRITE_PCI_READ_SHARED_RAM_TEST_FAILED, //Preliminary shared RAM
                                                 test failed.
  I186_WRITE_PCI_READ_PROGRAM_RAM_TEST_FAILED, // Preliminary program
                                                  RAM
                                                 // test failed.

  I186_RAM_TEST_FAILED,                  //Bootcode RAM test failed.
  BOOT_CODE_DISCRETES_TEST_FAILED,    //Bootcode walking '1's discretes
                                        loopback
                                      //test failed.

  STATUS_REGISTER_TEST_FAILED,           //Status register/FIFO tests failed.
  LOADER_FAILED,                         //Failed to load exec onto card.
  R0_OPEN_FAILED,          //'OpenChannel(Receiver0)' call in internal loopback test
                             failed.
  R1_OPEN_FAILED,          //'OpenChannel(Receiver1)' call in internal loopback test
                             failed.
  T0_OPEN_FAILED,          //'OpenChannel(Transmitter0)' call in internal loopback
                             test failed.
  R2_OPEN_FAILED,          //'OpenChannel(Receiver2)' call in internal loopback test
                             failed.
  R3_OPEN_FAILED,          //'OpenChannel(Receiver3)' call in internal loopback test
                             failed.
  T1_OPEN_FAILED,          //'OpenChannel(Transmitter0)' call in internal  loopback
                             test failed.

  CHIP_A_LOW_SPEED_INTERNAL_LOOPBACK_FAILED,    //Loopback test failed for
                                                   chip A @ low //speed.
  CHIP_A_HIGH_SPEED_INTERNAL_LOOPBACK_FAILED,   //Loopback test failed for
                                                   chip A @ high //speed.
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
  CHIP_B_LOW_SPEED_INTERNAL_LOOPBACK_FAILED,     //Loopback test failed //for
                                                   chip B @ low //speed.
  CHIP_B_HIGH_SPEED_INTERNAL_LOOPBACK_FAILED,  //Loopback test failed //for
                                                   chip B @ high //speed.
  FORCE_TO_2_BYTES14=32767                  //Required for LabWindows
} TEST_RESULT;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

A.2 STRUCTURE DEFINITIONS



Note 1: *In the next few unions, i8, i16, and i32 simply allow the data to be referenced as a group of bytes rather than the standard ARINC label and data.*



Note 2: *The defined string _MSC_VER is defined within the Microsoft C compiler, so all compilers will choose the proper parts of the following structures automatically.*

```
typedef UCHAR LabelType;


//  Each receiver FIFO can hold any of the following 2 types defined below:
//  ArincRawWordType:
//   a raw ArincStampedRawWordType:
//   a raw Stamped packet


/*  Turbo C 2.0 and indeed Borland C/C++ 3.0 support bitfields, but up to a
  a maximum of 16 bits.  This is why we need a separate data structure for
  Borland's C compilers.  - l 04-28-93
*/


typedef union {
  UCHAR i8[4];
  USHORT i16[2];
  ULONG i32;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#ifdef _MSC_VER
//  Microsoft supports 32-bit bitfields, so we have the following for MSC.
  struct {
    ULONG Label :8;
    ULONG Data  :24;
  } fld;
#else
/*
 *  Turbo C 2.0 and Borland C/C++ 3.0 support bitfields, but only up to a
 *  a maximum of 16 bits.  This is why we need the following data structure'
 *  for Borland's C/C++ compilers.
*/
  struct {
    UCHAR Label;
    UCHAR Data[3];
  } fld;
#endif
} ArincRawWordType;


typedef union {
  UCHAR i8[8];
  USHORT i16[4];
  ULONG i32[2];
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#ifdef _MSC_VER
//  Microsoft supports 32-bit bitfields, so we have the following for MSC.
  struct {
    ULONG Label :8;
    ULONG Data  :24;
    USHORT SoftwareStamp;
    ULONG HardwareStamp;
  } fld;
#else
/*
 *  Turbo C 2.0 and Borland C/C++ 3.0 support bitfields, but only up to a
 *  a maximum of 16 bits.  This is why we need the following data structure'
 *  for Borland's C/C++ compilers.
*/
  struct {
    UCHAR Label;
    UCHAR Data[3];
    USHORT SoftwareStamp;
    ULONG HardwareStamp;
  } fld;
#endif
   } ArincStampedRawWordType;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
typedef union {
  ArincRawWordType        R;
  ArincStampedRawWordType SR;
} ArincWordType;


/* added the following type 20-Mar-95 */
/* Description:
//  This type defines a time-stamped snapshot of the discrete inputs.
//  The discrete values 5..1 are stored in bits 4..0 of the object
//  PackedDiscreteState.  These values are recorded on a change of
//  state.
*/


typedef struct {
  USHORT PackedDiscreteState;
  USHORT SoftwareStamp;
  ULONG HardwareStamp;
} StampedDiscreteType;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX B

ACRONYMS

ACARS	Aircraft Communication, Addressing and Reporting System
ACMS	Aircraft Condition Monitoring System
ARINC	Aeronautical Radio Inc.
ATE	Automated Test Equipment
CGA	Color Graphics Adapter
CMC	Central Maintenance Computer
CMOS	Complementary Metal Oxide Semiconductor
CS	Chip Set
DADC	Digital Air Data Computer
DOS	Disk Operating System
DPRAM	Dual-Port Random Access Memory
EFIS	Electronic Flight Instruments System
EGA	Enhanced Graphics Adapter
EIU	Electronic Interface Unit
EPROM	Erasable Programmable Read Only Memory
FIFO	First In First Out Memory
FMC	Flight Management Computer
GND	Ground (electrical)
I/O	Input/Output
IDT	Integrated Device Technology
IDU	Integrated Display Unit
IRS	Inertial Reference Unit
LRU	Line Replaceable Unit
MCDU	Multi-Control Display Unit
MHZ	Megahertz
N/A	Not available
PC	Personal Computer
RAM	Random Access Memory
ROM	Read Only Memory
RX	Receiver
TTL	Transistor-Transistor Logic
TX	Transmitter
UUT	Unit Under Test
VDC	Voltage Direct Current
VGA	Video Graphics Adapter

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

INDEX

AddLabel 28, 29, 37, 38
AddLabelGroup ... 28, 29, 37, 38, 39, 50, 51, 53
ArincRawWordType 123, 125, 129
ArincToTransceiver 135, 142
BAD_ARGUMENT 120
BIN_IS_EMPTY 102, 104, 120
caution icon ix
Clear_FIFO 19, 28, 41, 132
ClearPCInterrupt 40
Close429MsgWindow 42
CloseChannel 25, 43, 90
CloseChannelAll 25, 43, 44
ComposeArincWord 27, 35, 45
Create429MsgWindow 47
DATA_WAS_LOST 120
DecomposeArincWord 27, 35, 46, 48
DeleteAllLabel 28, 29, 37, 39, 50, 53
DeleteLabel 28, 29, 37, 39, 50, 51, 52, 53
DeleteLabelGroup 28, 29, 37, 39, 50, 51, 52, 53
DisableGlobalCardToPcInt 33, 54, 57
discrete input 26
discrete output 26
discretes 26
disk icon x
Display 155
DisplayFunction 79
Dual-Port RAM 24
electrical static discharge x
EnableGlobalCardToPcInt 31, 33, 56, 132
EnableParityAsData 34, 59
ESD x
FM-PC FIFO 24
Get_Extended_Scheduled_Label_Value .. 72
Get429AbsoluteTime 32, 33, 61, 63, 65, 118, 132
Get429TimeDifference .. 32, 62, 63, 65, 118, 132
Get429TimerResolution 78
GetCardCurrentTime 31, 62, 63, 64, 132
GetChannelSpeed 21, 34, 66, 70, 111, 113
GetChannelStatus 21, 34, 67, 68
GetChipSpeed 21, 34, 66, 70, 113
GetCurrentCard 25, 71, 114
GetPendingArincWords 28, 34, 74
GetScheduledLabelValue 34, 72, 73, 75
GetTestResultStruct 77, 106, 108
GetTimerResolution 34
header file 24
headquarters 2
icon
  caution ix
  disk x
  idea/tip ix
  note ix
  reference ix
  warning ix
idea/tip icon ix
Initialize ... 20, 25, 27, 28, 30, 33, 34, 57, 72, 75, 78, 79, 107, 108, 123, 129, 136, 137, 138
installation
  hardware 3
  software 16
INTERNAL_ERROR 120
IsBoardPresent 136, 137, 139, 141
jumper blocks 3
Kill_Extended_Scheduled 81
Kill_Scheduled 27, 28, 81, 82, 83, 84, 85
Kill_Scheduled_All 82, 84, 85
Loader 136, 137, 139, 141
loopback connector 15
Modify_Mailbox_Mode 86
note icon ix
OpenChannel .. 25, 27, 28, 29, 30, 31, 34, 37, 38, 43, 44, 49, 50, 51, 52, 62, 63, 65, 68, 69, 86, 88, 89, 90, 98, 99, 100, 101, 102, 103, 104, 105, 112, 118, 123, 125, 127, 129, 132, 149
P429TB_version 21, 91, 92, 93
Panic 94
ParityAsData 34, 59
PC_FAILURE 120
PC_SUCCESS 120
Plug-and-Play Configuration 3, 9
Program RAM 24
PROTO429.H 24

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

QUEUE_IS_EMPTY 120
QUEUE_IS_FULL 120
Read Only Memory..... 155
ReadConfiguration..... 136, 137, 139, 141
ReadDiscrete 27, 95, 97, 131
ReadDiscreteQueue 21, 96
Receive_FIFO 18, 28, 30, 39, 41, 74, 90, 98, 99, 100, 101, 103, 105
Receive_FIFO_Block 28, 30, 90, 99, 100, 103, 105
Receive_Mailbox 19, 28, 29, 86, 99, 101, 102
Receive_Mailbox_Extended 28, 86, 104
receiver..... 70
Receiving 17
ROM 155
RunDiagnostic..... 20, 33, 106
SCHEDULED_IS_FULL 120
SetChannelSpeed 25, 34, 66, 70, 99, 101, 103, 105, 109, 110, 113
SetChipSpeed..... 25, 34, 66, 70, 112
SetCurrentCard 25, 71, 114
SetDetailedDiagnostic..... 115
SetDisplayFunction..... 116
SetErrorFunction..... 117
SetInitialAbsoluteTime .. 32, 62, 63, 65, 118, 132
Shutdown 25, 34, 119
StartCard 136, 137, 138, 141
StatusToAscii..... 35, 120
Structure Definitions..... 151
technical support 2
TestResults..... 77
TIME_OUT..... 120
TimeDelay..... 140
ToolBoxReadConfigFile..... 122, 141
ToolBoxReadConfigInfo 122, 141
TO-PC FIFO 24
TransceiverToArinc 135, 142
Transmit_Extended_Scheduled 28, 123
Transmit_FIFO 19, 27, 74, 124, 125, 126, 127, 128, 130
Transmit_FIFO_Block 27, 124, 126, 127, 130
Transmit_Scheduled.... 27, 73, 76, 81, 82, 83, 84, 85, 123, 126, 128, 129
transmitter 70, 89
Transmitting..... 17, 27
TYPES429.H 24, 123, 125, 127, 129, 145
warning icon..... ix
web site 2
WriteDiscrete 27, 131
ZeroCardClock..... 31, 62, 63, 65, 132

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP