# EMBC1000-PCI429EI-42

(Simplified V1.5.2)

Orbita Control Engineering Co., Ltd.

Addr: Orbita TechPark, 1 BaishaRoad, Zhuhai, Guangdong, China, 519080 Tel: +86 756 3391979 Fax: +86 756 3391980 Web: <u>www.myorbita.net</u>



First published in 2008 by Orbita Control Engineering Co. Ltd. Zhuhai, China

© Orbita Control Engineering Co. Ltd.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior permission, in writing, from Orbita Control Engineering Co. Ltd.(thereafter called "Orbita").

## **User's Manual Information**

This document contains the simplified User Manual of EMBC1000-PCI429EI-42 card. Orbita reserves the rights to make changes in the products or specifications contained in this document in order to supply the best possible products. Orbita does not assume responsibility for errors that may appear in this manual.

Orbita also assumes no responsibility for the use of this device beyond the product specifications. Applications for any target hardware connections contained in this publication are for illustration purposes only and Orbita makes no representation or warranty that such applications will be suitable for the use specified without further testing or modification according to the target hardware specifications.

The software associated with the shipped device shall not be used for other purpose except as stated in the terms of the software license agreement, or with special permission from Orbita.

## **Special Notes**

EMBC, EIPC, S698, OBT429, OBT1553B, ORION are registered trademarks of Orbita Control Engineering Co. Ltd.

Microsoft, Windows XP, Windows 2000 are registered trademarks of Microsoft Corporation. All other products mentioned in this User's Manual are trademarks or registered trademarks of their respective manufacturers.

## Application of EMBC1000-PCI429EI-42

EMBC1000-PCI429EI-42 is a PCI based card that provides new levels of performance and flexibility for systems interfacing to ARINC429 data bus, including data transmission, data reception, real-time data display, data recording and replay, data post analysis, etc..



# CONTENTS

CHAPTER 1 OVERVIEW	1
1.1 About EMBC1000-PCI429EI-42	1
1.2 Applications	2
1.3 Characteristics	2
1.4 System Requirements	3
1.5 Special Handling and Care	3
CHAPTER 2 GETTING STARTED	4
2.1 ARINC429 BUS INTERFACE	4
2.2 ELECTRIC PROPERTIES OF ARINC429 BUS CONNECTIONS	5
2.3 Resources on CD-ROM	5
CHAPTER 3 ERROR INJECTION AND DETECTION	7
3.1 Parity Error	7
3.2 GAP ERROR	7
3.3 Short Word Error	7
3.4 Long Word Error	8
CHAPTER 4 OPERATIONS AND SETUP	9
4.1 HARDWARE AND ITS DRIVERS INSTALLATION	9
4.1.1 Hardware installation	9
4.1.2 Board oriented drivers Installation	9
4.1.3 LIB oriented drivers Installation	
4.2 GET STARTED WITH THE APPLICATION SOFTWARE	
4.3 PARAMETERS SETUP	
4.3.1 Rx Channel Parameter Setup	
4.3.2 Tx Channel Parameter Setup	
4.4 DATA TRANSMISSION OPERATIONS	24
4.5 DATA RECEIVING OPERATIONS	
4.6 ERROR INJECTION AND ERROR DETECTION OPERATION	
4.6.1 Transmit error injection	
4.6.2 Automatic error detection in receive channels	
CHAPTER 5 DATA ANALYSIS SOFTWARE	35
CHAPTER 6 DEVELOP YOUR OWN APPLICATION SOFTWARE	
6.1 Application Programming Interface	
6.2 EXAMPLE SOURCE CODE	
6.3 API FUNCTION DESCRIPTION	
CHAPTER 7 PRODUCT ORDERING INFO	57
CHAPTER 8 RELATED PRODUCTS	
Orbita Control Engineering Co., Ltd.	- ii-



APPENDIX A A DINC/20 PROTOCOL INTRODUCTION	50
APPENDIA A: ARINC429 PROTOCOL INTRODUCTION	<b>39</b>



# CHAPTER 1 OVERVIEW

# 1.1 About EMBC1000-PCI429EI-42



Figure 1-1 EMBC1000-PCI429EI-42 Card

EMBC1000-PCI429EI-42 is a PCI card designed specifically to interface with ARINC429 data bus target hardware or systems, including data transmission, data receiving, event monitoring, remote control, real-time data display, data recording and replay, data post analysis of ARINC429 target hardware or systems.

Designed with 4 independently programmable receive (Rx) channels and 2 independently programmable transmit (Tx) channels, and automatic error detection in receive channels and error injection in transmit channels. EMBC1000-PCI429EI-42 card operates in 100k/50k/48k/12.5kbps rates with software configurable. This including other advanced designs make it an ideal tool to perform data acquisition, data analysis, remote control or event monitoring over the ARINC429 bus.

EMBC1000-PCI429EI-42 card comes with drivers software, API (Application Programming Interface) library and user oriented application software, running under Windows 2000/XP. The user oriented application software has been designed with the capabilities of simulating the outputs of various airborne systems, receiving inputs from these systems, and providing bus analyzer functions. API library is also provided together with example source code (Visual C++), which allow users to easily develop



their own application software or project.

EMBC1000-PCI429EI-42 card is of PCI Plug-and-Play (PnP) compatible for easy installation (any PCI 2.1, or higher, compatible slot), and multiple such cards can work together inside one PC or workstation.

## **1.2 Applications**

EMBC1000-PCI429EI-42 card is well suited for all types of ground support work (development, manufacturing test, on-site maintenance, etc.), as well as on-board data acquisition. LRU developers find that this card provides easy access for simulating and/or testing new systems prior to use with actual flight systems. Avionics maintenance and validation teams enjoy on-site testing and analysis with this card.



Figure 1-2 Wide Applications

## **1.3 Characteristics**

EMBC1000-PCI429EI-42 card features with the following characteristics:

- Comply with 32-bit 33Mhz PCI bus speed;
- PCI Plug-and-Play (PnP) compatible for easy installation;
- Up to 4 programmable ARINC429 Rx channels;
- Up to 2 programmable ARINC429 Tx channels;
- Support 32/25 bit Word length;
- 1024 bytes FIFO for each Rx channel;
- 1024 bytes FIFO for each Tx channel;
- Programmable baud rate (12.5Kbps、48Kbps、50Kbps、100Kbps);
- Transmitting capability: Queued (FIFO) transmission to any external channel; Scheduled label and SDI transmission to any channel;



Recurrent support(Time gap: 1 ~ 99,999,999 us);

- Receiving capability: Queued (FIFO) reception from any external channel; "Mailbox" type reception from any channel; Filtering of receive labels
- Automatic error detection in receive channels: Parity error; Gap error (less than 4 bit times); Short word error (time out error); Long word error (more than 32/25 bits);
- Transmit error injection: Parity error (make the parity bit to be opposite); Gap error (make gap to be 2 bit times); Short word error (30/23 bits); Long word error (34/27 bits);
- Up to 256 multiple cards in one system supported;
- Complete set of drivers for Windows XP/2000;
- Complete API and ANSI C DLL library for user's design and integration;
- Well configured application software and source codes provided;
- PCI Modular design: 130mm x85mm;
- Working temperature [-40°C, +85°C];

## **1.4 System Requirements**

An IBM compatible PC equipped with PCI bus, Pentium processor, math co-processor, hard disk drive, CD-ROM drive, monitor, Windows XP or Windows 2000, etc.. shall be required for installing the PCI card and the associated software.

A cable assembly is required to interface to the ARINC 429 bus target hardwar or other discrete channels.

## 1.5 Special Handling and Care

Since EMBC1000-PCI429EI-42 card uses state-of-the-art components and connectors, properly handlings and cares must be taken to ensure that the device will not be damaged by Electrical Static Discharge (ESD), physical shock, or improper power surges.



- > Turn off power to the PC completely;
- > NEVER insert or remove card with power turned on;
- Ensure that standard ESD precautions are taken. At least, one hand should be grounded to the power supply in order to eliminate static potentials;
- Do not store the card in environment exposed to excessive heat, magnetic fields or radiation.



# CHAPTER 2 GETTING STARTED

## 2.1 ARINC429 bus Interface

One front-panel DB25 male connector (it has been referred to "P801" inside the card) is used for implementation of ARINC429 bus signal interface between ARINC429 target hardware and the PCI bus.







Pin No	Descriptions	Pin Number	Descriptions
1	Receive channel 1 A	14	Receive channel 1 B
2	GND	15	Receive channel 2 A
3	Receive channel 2 B	16	Receive channel 3 A
4	Receive channel 3 B	17	GND
5	Receive channel 4 A	18	Receive channel 4 B
6	NC	19	NC
7	GND	20	NC
8	NC	21	NC
9	NC	22	GND
10	NC	23	NC
11	Transmit channel 2 A	24	Transmit channel 2 B
12	GND	25	Transmit channel 1 A
13	Transmit channel 1 B		

Table 2-1	DB25 n	nale	Connector	Pinouts
-----------	--------	------	-----------	---------

Note: Per ARINC-429 specification, "A-Side" signal is "+" (positive) with respect to ground, while "B-Side" signal is "-" (negative) with respect to ground.

## 2.2 Electric properties of ARINC429 bus connections

For the ARINC429 receive (Rx) channels, when you connect with any target hardware, the max input voltage of any signals shall be:  $\pm 30$ VDC.

As for the ARINC429 tranmit (Tx) channels, their bus data signals are standrard outputs:+5V±5% for High Voltage and-5V±5% for Low Voltage. If the ARINC429 tranmit (Tx) channel works under full load, the max resistance and capacitance impedance of the load is:  $400\Omega/30,000$ pF; while the max resistance and capacitance impedance of the load is:  $4000\Omega/10,000$ pF when the ARINC429 tranmit (Tx) channel works only under half load.

## 2.3 Resources on CD-ROM

The CD-ROM includes: Directory: G:\ (assume G:)

\ApplicationSoftware EMBC1000-PCI429EI.exe wdapi901.dll

data anaysis executable file DLL file for executable file



\doc

wdapi901.lib LIB file for executable file Note: EMBC1000-PCI429EI.exe, wdapi901.dll, wdapi901.lib must be kept in the same directory.

EMBC1000-PCI429EI-42USERMANU	AL(SV1.5. 2).pdf	User manual
\ driver		
\boarddriver EMBC1000-PCI429EI.sys EMBC1000-PCI429EI-42_device.	Board oriented	l drivers
LIBdriver EMBC1000-PCI429.inf EMBC1000-PCI429.sys wd901.cat	LIB oriented d	ivers
\UserDesign \APILibrary PCI_LINK3.dll PCI_LINK3.lib wdapi901.dll wdapi901.lib	For user's dev	elopment use
\sample VC++ sample		
\DataAnalysis DataConvert.exe Source File.txt Target File.txt		
\training LookBack.avi TX1_SEND.avi TX2_SEND.avi DataAnalysis.avi LIBdriver_install.avi LB_ERROR_INJECTION.avi boarddriver_install.avi	Video files to show	the operations



# **CHAPTER 3 ERROR INJECTION AND DETECTION**

EMBC1000-USB429EI-42 is a device which has the ability to inject errors into the transmit channel and detect the errors from receive channels.

## 3.1 Parity Error

If you enable the parity error injection, the parity bit in the nomal word will be chang to negative artificially. The **Figure 3-1** show you the definition of the Parity Error in the 32bits word. The Parity Error in the 25bits word is defined in the same way.

	Transfer Sequences (LSB Firstly)										
32	31	30	29	28	27	26	25	24	23	22~1	
Ρ	D	D	D	D	D	D	D	D	D		
	Normal (D - Data bit; P – Parity bit)										
ND	-	-	<b>D</b>	-	<b>D</b>		<b>D</b>	<b>D</b>			
NP	D	D	D	D	D	D	D	U			

Parity Error (D - Data bit; NP - Negative Parity bit)

#### Figure 3-1 Parity Error(32bit)

## 3.2 Gap Error

According to the ARINC429 protocol, the gaps between two words is 4 Null bits at least, in EMBC1000-USB429EI-42, the way to inject Gap error into transmit data is to make the Gap time between two words is 2 Null bits, Shown in **Figure 3-2**:



Figure 3-2 Gap Error(32bit)

## 3.3 Short Word Error

ShortWord Error is also called "Break Error", means the data was send not enough 32/25 bits, just like the data frame is broken. In EMBC1000-USB429EI-42, the way to inject the shortword error into the transmit data is cut down two bits which replace by two NULL bit, Shown in **Figure 3-3**:





Figure 3-3 Short Word Error(32bit)

# 3.4 Long Word Error

ILong Word Error is also called "Long Frame Error". It means the word sending contain 34/27 bits. If you enable the Long Wordd error injection, the 2 bits "00" will be add to nomal word(32 bits or 25 bits). The **Figure 3-4** show you the definition of the Long Word Error in the 32bits word. The Long Word Error in the 25bits word is defined in the same way.

	Transfer Sequences (LSB Firstly)										
34	33	32	31	30	29	28	27	26	25	24~1	
N	Ν	D	D	D	D	D	D	D	D		
	Normal (D - Data bit; N – Null bit)										
AD	AD AD D D D D D D D										
	Long Word Error (D - Date bit: AD - Additional Date bit)										

Figure 3-4 Long Word error(32bit)



# **CHAPTER 4 OPERATIONS AND SETUP**

## 4.1 HARDWARE and its drivers INSTALLATION

#### 4.1.1 Hardware installation

EMBC1000-PCI429 card is designed to be inserted directly into any PCI 2.1, or higher, compatible slot of PC.

**Note:** When installing the card, the following cautions must be taken:

- 1) NEVER insert or remove the card with the power turned on.
- 2) 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.
- 3) Insert the card gently into the motherboard slot. Secure with proper hardware.
- 4) Make sure that adjacent cabling and wiring do not hinder the airflow around the card.

Once the PCI card gets installed into your PC and proper connections are made between ARINC429 target hardware and PCI card via the DB25 connector, then the hardware installation gets basically done. Congratulations!

**NOTE:** Please be advised that you should assign a number to each PCI card (we called **Board Number** in our setup software) you inserted into the system according to the sequential order of the PCI slots in your system. If only ONE PCI card be inserted in your system, then the **Board Number** is 0. If there are more than one PCI card installed into your system, you should assign a number (valid 0-255) to each board according to the location of each PCI slot in the system. When the application software is used, it must be configured with each board, though multiple boards can work together.

However, before you could use this card, you need to install drivers software from the provided CD-ROM onto your PC: **board oriented drivers and LIB oriented drivers**, which are designed to run under Windows XP or Windows 2000.

#### 4.1.2 Board oriented drivers Installation

Once the card gets installed and the PC is turned ON, the PC shall perform automatic detection for the card inserted. If the drivers have not been installed yet, it will treat the card to be a new hardware and, it will report to you right away by bringing up "Add New Hardware Wizard" to guide you for the installation of the required drivers.





Figure 4-1 Add New Hardware Wizard window

Click on "Install from a list or specific location" and then click on the Next button, you will asked to choose:

'lease choose your search ar	nd installation options.
O Search for the best driver in	these locations.
Use the check boxes below paths and removable media	to limit or expand the default search, which includes loc . The best driver found will be installed.
Search removable me	edia (floppy, CD-ROM)
Include this location i	n the search:
D:\d\Libdriver	Browse
⊙ <u>D</u> on't search. I will choose t	he driver to install.
Choose this option to select the driver you choose will be	the device driver from a list. Windows does not guaran e the best match for your hardware.

Figure 4-2 choose the driver to install



Choose "Don't search.I will choose the driver to install", then click on the **Next** button, you will get:

Found New Hardware Wizard
Select the device driver you want to install for this hardware.
Select the manufacturer and model of your hardware device and then click Next. If you have a disk that contains the driver you want to install, click Have Disk.
Model
EMBC1000-PCI429-42 Driver
Have Disk
< <u>Back</u> <u>Next</u> >Cancel

Figure 4-3 select the .inf file from disk

click on "Have Disk... "button, you will enter the next window:

Ins	tall F	rom Disk
S Mc	H.	Insert the manufacturer's installation disk, and then OK make sure that the correct drive is selected below. Cancel
		Copy manufacturer's files from:
		A:\ <u>B</u> rowse

Figure 4-4 open the .inf file



Browse the CD-ROM, open the EMBC1000-PCI429EI\_device.inf from CD\_ROM (**G:**\ **driver**\ **boarddriver**) and click on **OK** button, you will get:

Found New Hardware Wizard	
Select the device driver you want to in	stall for this hardware.
Select the manufacturer and model of you have a disk that contains the hiver you Show <u>compatible hardware</u>	ur hardware device and then click Next. If you want to install, click Have Disk.
Model	n l
EMBC1000-PCI429EI-42 Driver	
This driver is not digitally signed! <u>Tell me why driver signing is important</u>	Have Disk
	< <u>Back</u> Nyt> Cancel

Figure 4-5 open the .inf file

Click on **Next** button to have the driver installed.



Figure 4-6 finish the board drivers installation

Click on **Finish** button to complete this installation procedure. Congratulations! Now you have installed the board drivers onto your PC successfully.



#### 4.1.3 LIB oriented drivers Installation

From your desktop of Windows XP, enter into the **Control Panel** and then open **Add Hardware**, you will get:



Figure 4-7 open Add Hardware

Click on **Next** button, you enter:

Is the hardware connected?			E ST
Have you already connected this hardware	e to your computer?	,	
Yes, I have already connected the h	hardware		
No, I have not added the <u>hardware</u>	yet		
	< <u>B</u> ack	<u>N</u> ext >	Cancel

Figure 4-8 Add Hardware Wizard



Click on "Yes,I have already connected the hardware",and then click on **Next** button, you will be asked:

Add Hardware Wizard
The wizard can help you install other hardware
The wizard can search for other hardware and automatically install it for you. Or, if you know exactly which hardware model you want to install, you can select it from a list.
What do you want the wizard to do?
$\bigcirc$ Search for and install the hardware automatically (Recommended)
Install the hardware that I manually select from a list (Advanced)
<pre>&lt;<u>Back</u> Cancel</pre>

Figure 4-9 Add Hardware Wizard

Click on "install the hardware that I manually select from a list",and then click on **Next** button, you will get:

From the list below, select the type of hardware you a	re installing
If you do not see the hardware category you want, click Sho	w All Devices.
Common <u>h</u> ardware types:	
Show All Devices	~
😼 Display adapters	
IDE ATA/ATAPI controllers	
IEEE 1394 Bus host controllers	
Imaging devices	
Infrared devices	
<b>■#</b> Jungo	
Modems	~
- Parti	Neut > Causal
< <u>B</u> ack	

Figure 4-10 select the type of hardware



Click on **Next** button, you enter the window below.

Select the n	nanufacture	er an	d model of your hardware device and then click Next. If you	
have a disk	that conta	ins tł	ne driver you want to install, click Have Disk.	
Manufacturer		~	Model	
Standard Infrared Port) (Standard Modem Types) (Standard MTP-Compliant Dev			Serial Cable using IrDA Protocol	
(Standard port tupes	) •)			
(Standard port types	2			

Figure 4-11 select the .inf file from disk

Click on Have Disk... button, you will see:

Add Hardwa Select th	are Wizard e device driver you want to install for this hardware.
	I From Disk Insert the manufacturer's installation disk, and then make sure that the correct drive is selected below. Cancel
Ma (St) (St) (St) (St)	Copy manufacturer's files from:
	A.Y

Figure 4-12 open the .inf file



Open the EMBC1000-PCI429.inf from CD-ROM (G:\ driver\ LIBdriver) and then click on **OK** button, you enter:

Select the manufacturer and model o have a disk that contains the driver y	f your hardware device a ou want to install, click H	nd then click Next. If you ave Disk.
Model		
orbita EMBC1000-PCI429		
This driver is not digitally signed		Have Disk.

Figure 4-13 open the .inf file

Click on Next button, you get:

Add Hardware Wizard	
The wizard is ready to install your hardware	
Hardware to install:	
orbita EMBC1000-PCI429	
To start installing your new hardware, click Next.	
< <u>B</u> ack	Vext> Cancel

Figure 4-14 install hardware



Click on **Next** button to have the LIB drivers installed.



Figure 4-15 finish the LIBdriver installing

Click on **Finish** button to complete this installation procedure. Congratulations! Now you have installed the LIB drivers onto your PC successfully.

#### HINT: Check if the hardware drivers get installed properly

Yes, you can simply open the **Device Manager** to check. If both board drivers and LIB drivers gets installed properly, they shall be displayed under device group **Jungo** as below:



Figure 4-16 hardware install successfully



## 4.2 Get started with the Application Software

Once the HARDWARE INSTALLATION gets done successfully, EMBC1000-PCI429EI card is ready to use.

Each PCI card should be assigned a number (we called **Board Number** in our setup software) according to the sequential order of the PCI slots in your system. If only ONE PCI card be inserted in your system, then the **Board Number** is 0. If there are more than one PCI card installed into your system, you should assign a proper number (valid 0-255) to each board according to the location of each PCI slot in the system. When the application software is used, it must be configured with each board with the specified Board Number, though multiple boards can work together.

Double click on the application software EMBC1000-PCI429EI.exe (G:\ ApplicationSoftware), you will see the software startup Logo:



Figure 4-17 software startup Logo

After about 10 seconds, the "Select Board Number" window will popup. The "Board Number" means the application software will only opration this card. When multiple cards in one system, the user should open multiple application software to operate the cards. The user should get the value of "Board Number" from the PCI slot which the card inserted. If only one card in the system, the "Board Number" is 0.

Now you should enter the "Board Number" (possible assignment value range: 0 to 255) of the PCI card. Click on OK button after the number gets entered.

×
ec.)
1

Figure 4-18 Select Board Number



If the "Board Number" you entered is incorrect or the hardware drivers have not been installed properly, you will observe the Error information window as below. In this case, you should double check if the PCI card is in proper working mode or if the card is numbered properly.



Figure 4-19 Error prompt

If Board Number is properly specified and the PCI card works well with the system, the application software will be launched and its main window will be opened as below.



Figure 4-20 main window of the application software



## 4.3 Parameters Setup

#### 4.3.1 Rx Channel Parameter Setup

Click the "RX-1234", you will enter the parameters setup page of Rx channels, shown in **Figure 4-2**1.

MBC1000-PCI429EI-42	Set	up Rx channel F	Parameters	
-1234   TX-1   TX-2   About	EM			)
Parameter Setting		Receiv	ve Data	
Word Length 25 - BIT	RX-1	RX-2	RX-3	RX-4
Baud rate 100 💌 Kbps				
Parity check disable 💌				
SDI decode disa				a .
Save As				4
Label check disal Savejn:	SAVE	←	• 🗈 📸 📰 •	
LABEL 00				
LI 00 Sa	ave the parame	eters as RX_SE	TTING.txcfg file	e
12 00 V SM				
14 00 M Sar				
15 00 F Sa		/		
16 00 🔽 Sar				
L7 00 F Se File name	RX_SETTING		Save	
Select Chann Save as h	pe: All Files(*.rxcfg)		Cancel	
₩ RX-1 ₩ RX-2		- E	1	
₩ RX-3 ₩ RX-4	RX-1 Start	RX-2 Start	RX-3 Start	RX-4 Start
	RX-1 Clear	RX-2 Clear	RX-3 Clear	RX-4 Clear
Sum Sutting Total Sutting	RX-1 Save	RX-2 Save	RX-3 Save	RX-4 Save
Dave Setting Load Setting	Error Detection	Error Detection	Error Detection	Error Detection
Annly Setting	GAP :	GAP:	GAP:	GAP:
whith secting	S_W: I. W	S_W:	S_W: L W:	S_W: L W:
				1.77-7.07

Figure 4-21 Rx channel parameters setup

Now you are free to set the Rx channel parameters, such as: Word length, Baudrate, Parity check, SDI check, Lable check, etc.. After changing the parameter, you must press the "Apply setting" button to make it valid, and the configuration can be saved into a data file by pressing the "Save Setting" button, and an existing configuration can be loaded from a file by pressing the "load setting" button.

Note: User must press the "Apply setting" button to enable the changes to parameters.

#### Possible assignment value for each parameter

Under the receive (Rx) channel parameter setup window, the content of each parameter



can be selected by pulling down the respective menu bar. The contents of each parameter are listed below.

Menu Item	Possible assignment value	Default
Word Length	32, 25 bit	32
Baud Rate	12.5, 48, 50, 100 Kbps	12.5
Parity Check	Disable, odd, even	Disable
SDI Decode	Disable, 00,01,10,11 (binary)	Disable
Label Check	Disable, enable	Disable
L1	0~0xFF	00
L2	0~0xFF	00
L3	0~0xFF	00
L4	0~0xFF	00
L5	0~0xFF	00
L6	0~0xFF	00
L7	0~0xFF	00

Buttons on Rx parameters setup page description:

Apply Setting

"Slect channel": Select a singel Rx channel or multiple

channles which you want to change the parameters.

Save Setting "Save Setting": button: Save the Rx configuration into a txt file.

Load Setting Load Setting": button: Load Rx configuration from a exiting txt file.

"Apply Setting": button: Enabel the parameter changes.

If the Rx parameters setup done, you can observe it from the pulldown bar:

RX-1:	32bit	Rate:12.5Kbps	Parity:disable	SDI:disable	LABEL: disable	•
RX-1:	32bit	Rate: 12, 5Kbps	Parity:disable	SDI: disable	LABEL: disable	
RX-2:	32bit	Rate: 12.5Kbps	Parity: disable	SDI: disable	LABEL: disable	
RX-3:	32bit	Rate: 12.5Kbps	Parity: disable	SDI: disable	LABEL: disable	
RX-4:	32bit	Rate: 12.5Kbps	Parity:disable	SDI:disable	LABEL: disable	6

Figure 4-22 Parameter setting display

Once the Rx channel parameters get setup, each Rx channel will perform data receiving operations strictly according to the data format defined in the setup.



## 4.3.2 Tx Channel Parameter Setup

Click the "TX-1" button, you will enter the parameters setup page of the first Tx channel channel, shown in Figure 4-23.

EMBC1000-PCI429EI-42           Tx-1 channe           RX-1234 TX-1 TX-2 About EM	I Parameters setup
Parameter Setting Word length 25 III Baud rate 50 Kbps Parity check even I Begin Data(H)	Send data Single add SDI(B) SSM(B) LAB(H) Data(H) Bulky add Increment(H) Number(D) Add
Word gap Save As Work Save jn: C SAVE	?×       ▼ ← € ☆ Ⅲ▼   Delete
Error In Parity Gap	ameters as TX1_SETTING.txcfg
Repeti     File name:     TX1_SETTING       Image: Time gap:     All Files(*.txcfg)	Cancel Send
Save Setting Load Setting	Stop
All: 0	Words Send Num: 5 Words

Figure 4-23 setup transmit channels' parameter

In this window, you are free to change the Tx channel parameter such as: word length, baud rate, parity, word gap, repetition mode, error injiection and the work mode such as: "Normal mode" or "Loopback mode". In the loopback mode, EMBC1000-PCI429EI can only receive the data from the internal Tx channels of the device, data from external target will be ignored. The configuration can be saved into a data file by pressing the "Save Setting" button, and an existing configuration can be loaded from a file by pressing the "load setting" button.

In the Repetition Mode, if "enable" is selected, the current data in "send data" area will be tramsitted repeatedly. The interval time between 2 sequential words is specified by "Time gap" (ranged from 1 to 99,999,999 us), the repetition transmit will be terminated by press the "stop" button in "send data" area. In the Repetition Mode, if "disable" is selected, the current data will only be sent one time.



**Note**: User must click on the "Apply Setting" button to to make any parameter changes effecticve.

Click on the "TX-2", you will enter the parameters setup window for transmit channel (Tx-2). The setting for this channel is identical to TX-1's.

#### Possible assignment value for each parameter

Under the trasimit (Tx) channel parameter setup window, the content of each parameter can be selected by pulling down the respective menu bar. The contents of each parameter are listed below.

Menu Item	Possible assignment value	Default
Word Length	32, 25	32
Baud Rate	12.5, 48, 50, 100	12.5
Parity Check	Disable, odd, even	Disable
Word Gap	5 ~ 255	10
Work Mode	Normal, Loopback	Normal
Repetition Mode *	Disable, Enable	Disable
Error injection **	Parity/Gap/Short Word/Long Word Error	No error injecting
Time gap ***	1 ~ 99,999,999 us	100 us

 Table 4-2 Possible Assignment Value For Each Tx Parameter

\* The max number of data can be transmitted under Repetition Mode is 1024.

\* The max number of data can be transmitted under Repetition Mode is 1024.

\*\* You are free to inject the errors:

- Parity error: the parity bit of the data will be opposite to the bit you defined.
- Gap error: set the word gap to be 2 bit times;
- Short word error: only send 30/23 bits in one word(32/25bit).

Long word error: send 34/27 bits in one word(32/25bit)..

Note: only of those error(gap error, short word error, long word error) can be injected at the same time.

\*\*\* Refers to the time gap between two Repetition transmits. Time gap can only be assigned a value when Repetition Mode is under "Enable".



## 4.4 Data Transmission Operations

After the parameter setup is finished, the user can add the data into the "Send data" area for data transmission operations (by pressing "Send" button under the data transmit window).

The user can add the data into the "Send data" area, either in single or bulk mode.

In single mode, there is a dedicated data entry area under "Single add", where one may simply put a digital number (in Hex) into the field of "WORD(H)", or he may define the following detailes to compose a WORD to be sent: PAR(B), SDI(B), SSM(B), LAB(H), Data(H), which represents Parity, SDI check bits, SSM bits, Lable check and Data.

Field	Range
WORD(HEX)	0x0000000~0xFFFFFFF
PAR(Binary)	0,1
SDI (Binary)	00,01,10,11
SSM (Binary)	00,01,10,11
LAB(HEX)	0x0~0xFF
DATA(HEX)	0x000000~0x7FFFF

 Table 4-3
 The fields definition of Tx data

In bulk mode, there is a dedicated data entry area under "Bulky add", where one may simply create a base digital number (in Hex, here we defined it to "Begin Data"), and define the increment and the toal number of data in the respective field, then it will automatically generate a set of data.

It is also possible for the user to use single mode entry method repeatedly to generate a set of data (bulk data) to tranmit.

For any data added into this field, it can be saved into a data file by pressing the "Save" button, and an existing data file can be loaded in pressing the "Load" button.

The user can press "Clear" button to clear all the data which have addedd in the "Send data" area. If you want to delete one word from the data added in the "Send data" area, you should select it and then press "Delete" button.

Each time, the total number of WORDs added shall counted and be displayed in the "All" field. And the total number of WORDs transmitted via this transmit channel shall be counted and displayed in the "Send Num" field.

Data transmission can be terminated any time by pressing "Stop" button.



## 1) Add single Tx data in WORD field

enter a single data wit	h WORD	Press ADD to add sing	le data, and
PV-1234 TX-1 TV-2 About FM		the attributes to be	created and
Describer Contrine	$\setminus$ /	displayed automatically.	
Parameter Setting		Single add	
Word length 32 BIT	WORD (H) PAR (B	) SDI(B) SSM(B) LAB(H) Data 10 11 78 091A	(H) .2 Add
Baud rate 12.5 💌 Kbps	-	Bulky add	
Parity check disable	Begin Data(H)	Increment (H) Number (D)	Add
Word gap 10 BITs	WORD PAR	SDI SSM LAB DATA	
Work Mode © Normal C Loop	12345678 0	10 11 18 09182	Clear
Error Injection			
🗌 🗖 Parity 🗖 Short Word			
🗖 Gap 🧖 Long Word			Save
Repetition			Load
€ disable € enable			
Time gap: 100 us			Send
			Stop
Save Setting Load Setting			
Apply Setting	All: 1	Words Send Num:	Words
The total number of wor	rds added		Exit

Figure 4-24 Add single data with WORD field



## 2) Add single data in attribute fields

	Press ADD to generate a WORD
Enter a single data with a	and to add it into the Send data
RX-1234 TX-1 TX-2 About EM	area and to display it in the area
Parameter Setting Word length 32 • BIT Baud rate 12.5 • Kbps Parity check disable • Word gap 10 BITs	Send data       Single add       WORD (H)     PAR (B)       1     01       01     01       11     11       11     11       11     11       11     11       11     11       11     11       11     11       11     11       11     11       11     11       11     11       11     11       Add
Work Mode       Image: Normal       Image: Normal <td>A single word is added into</td>	A single word is added into
□ Gap □ Long Word	the "Send data" area, and is displayed here.
© disable O enable Time gap: 100 us	SendStop
Save Setting Load Setting A Apply Setting	11: 1 Words Send Num: Words
The total number of wor	

Figure 4-25 Add single data with attribute fields



#### 3) Add Bulk data



Figure 4-26 Add Bulky data

#### Attention: about the Data Format

The standard ARINC429 Data Word Format is different from the format of the data in the "Send data" or "Receive data" area, show in **Figure 4-27** and Figure 3-28.





32-bit ARINC429 word transfer order (LSB first)



Figure 4-27 Mapping of ARINC429 Data in 32-bit Format



25-bit ARINC429 word transfer order (LSB first)

Figure 4-28 Mapping of ARINC429 Data in 25-bit Format

Device word display and add order



## 4.5 Data Receiving Operations

Data Receiving Operation is quite simple. Once the Rx channel parameter setup gets done, then the user can simply press "RX-1 Start"/"RX-2 Start"/"RX-3 Start"/"RX-4 Start" button to enable the RX1/RX2/RX3/RX4 channel to start to receive data from the connected ARINC429 target hardware. It is also very important that the communication protocols between target hardware and this card should be defined the same.

Please be advised that "RX-1 Start"/"RX-2 Start"/"RX-3 Start"/"RX-4 Start" buttons can be toggled. Once you press "RX-1 Start"/"RX-2 Start"/"RX-3 Start"/"RX-4 Start" once, it will be toggled into "RX-1 Stop"/"RX-2 Stop"/"RX-3 Stop"/"RX-4 Stop".

To terminate Data Receiving operation, simply pressing "RX-1 Stop"/"RX-2 Stop"/"RX-3 Stop"/"RX-4 Stop" buttons again and, all buttons will then be toggled to "RX-1 Start"/"RX-2 Start"/"RX-3 Start"/"RX-4 Start" mode.

The data received and the total number of the data received will be displayed in the "Receive Data" area. It is advised that the "Receive Data" area only display 3000 words max. When more than 3000 words received, only the last 3000 words can be displayed.

The user can press "RX-1 Save"/"RX-2 Save"/"RX-3 Save"/"RX-4 Save" button to create a .txt data file to save all data received before pressing "RX-1 Start"/"RX-2 Start"/"RX-3 Start"/"RX-4 Start" button. Once the .txt data file created, then all data received will be saved into this .txt file for later analysis. The .txt data file can not be created if Data Receiving operation is still in progress.

The user can clear the data displayed on the "Receive Data" area by pressing "RX-1 Clear"/"RX-2 Clear"/"RX-3 Clear"/"RX-4 Clear" button.

The user can enable the automatic error detection function by selecting the button--"Error Detection". And the error will be display on the "Receive Data" area, and the total number of the error detected will be display on the "Error Detection" area. The user can clear them by pressing "RX-1 Clear"/"RX-2 Clear"/"RX-3 Clear"/"RX-4 Clear" button.



-1234 TX-1 TX-2	2 About E	w1			
Parameter Set	ting		Recei	ve Data	
Word Length 25	- BIT	RX-1	RX-2	RX-3	RX-4
Baud rate  100	▼ Kbps				
Parity check disa	ble -				
,					
SDI decode  disa	Save As			25	
Label check disa	Save in:	SAVE	•		
LABEL (H)					1
	CORX_SET	ING.rxcrg			
12 00 V Sa					
L3 00 🔽 Sa					
14 00 🕅 Sa					
15 00 🔽 Sa					
17 00 V Sa	Contract of the second				
Li j 00 jø 5a	File <u>n</u> ame:	R1		Save	
Select Chann	Save as type	E All Files(*.txt)		Cancel	
	КХ-2	1	1	1	<b>4</b>
🔽 RX-1		RX-1 Start	RX-2 Start	RX-3 Start	RX-4 Start
▼ RX-1 ▼ ▼ RX-3 ▼	RX-4				
₩ RX-1 ₩ ₩ RX-3 ₩	RX-4	RX-1 Clear	RX-2 Clear	RX-3 Clear	RX-4 Clear
RX-1 RX-1 RX-1 RX-1 RX-1 RX-1 RX-1 RX-1	RX-4	RX-1 Clear RX-1 Save	RX-2 Clear RX-2 Save	RX-3 Clear RX-3 Save	RX-4 Clear RX-4 Save
V RX-1 V V RX-3 V Save Setting Load	EX-4	RX-1 Clear RX-1 Save Z Error Detection	RX-2 Clear RX-2 Save Fror Detection	RX-3 Clear RX-3 Save V Error Detection	RX-4 Clear RX-4 Save
IV RX-1 IV IV RX-3 IV Save Setting Load	Setting	RX-1 Clear RX-1 Save ✓ Error Detection PAR: GAP:	RX-2 Clear RX-2 Save Fror Detection PAR: GAP:	RX-3 Clear RX-3 Save Frror Detection PAR: GAP:	RX-4 Clear RX-4 Save F Error Detection PAR: GAP:

Figure 4-29 Create a .txt file to save Data Received



Parameter	Setting	i Car			Receiv	e Data			
Word Length	25 <b>•</b> BIT	RX-1		RX-2		RX-3		RX-4	
		12340078	~	12340078	~	12340078	~	12340078	~
Baud rate	100 × Khng	12340089		12340089		12340089		12340089	
1266 9 C CO 1	Kops	1234009A		1234009A		1234009A		1234009A	
		123400AD		123400AD		123400AD		123400AD	
Farity check	disable 💌	123400CD		123400CD		123400CD		123400CD	
		123400DE		123400DE		123400DE		123400DE	
SDI decode	disable -	123400EF		123400EF		123400EF		123400EF	
	,	12340100		12340100		12340100		12340100	
		12340111		12340111		12340112		12340112	
Label check	disable 🗾	12340133		12340133		12340133		12340133	
		12340144		12340144		12340144		12340144	
LABEI	L 00	12340155		12340155		12340155		12340155	
L1 00		12340166		12340166		12340166		12340166	
	-	12340111		123401188		12340111		123401188	
12 00 1	✓ Same as L1	12340199		12340199		12340199		12340199	
L3 00	▼ Same as L1	123401AA		123401AA		123401AA		123401AA	
14 00 1	7	123401BB		123401BB		123401BB		123401BB	
LA   00	♥ Same as L1	123401CC		12340100 12340100		12340100		12340100	
L5 00 [	🗸 Same as Li	123401EE		123401EE		123401EE		123401EE	
L6 00 I	▼ Same as L1	123401FF		123401FF		123401FF		123401FF	
17 000	-	12340010		12340010		12340010		12340010	
<b>PU</b> 00 1	₩ Same az L1	12340021		12340021		12340021		12340021	
k		12340032		12340032	comm.(	12340032	-	12340032	
Select	Channel	12340043	~	12340043	~	12340054	~	12340054	~
RX-1	<b>▼</b> RX-2	1000		1000		1000		1000	
₩ RX-3	₩ RX-4	RX-1 Sto	PP ]	RX-2 Sto	P	RX-3 Stop		RX-4 Stop	p
		RX-1 Cle	ar	RX-2 Clea	ur 🛛	RX-3 Clear	r	RX-4 Clea	ar
		RE-1 Sat	70	RX-2 Sav	e.	RX-3 Save		RX-4 Sav	2
Save Setting	Load Setting	P Error Det	ection	🔽 Error Det	ection	F Error Deter	tion	V Error Dete	ctior
		PAR: 0		PAR: 0		PAR: 0		PAR: 0	
1.1.1.1.1	and the second	GAP: 0		GAP: 0		GAP: 0		GAP: 0	
Apply S	etting	S W: O		S #: 0		SW:0		S W: 0	

Figure 4-30 Data Receiving operations(Disable automatic error detection )

# 4.6 Error injection and error detection Operation

#### 4.6.1 Transmit error injection

The user can enable the error injection when she doing the tx channel parameters setup. The user can selet parity error, gap error, short word error, long word error(**Figure 4-35**) to inject into the data transmit.

You are free to inject the errors:

- 1. Parity error: the parity bit of the data will be opposite to the bit you defined;
- 2. Gap error: set the word gap to be 2 bit times;
- 3. Short Word error: only send 30/23 bits in one word(32/25bit);
- 4. Long Word error: send 34/27 bits in one word(32/25bit).

**Note**: only of those error(gap error, short word error, long word error) can be injected at the same time.





#### Step about data receiving with automatic error detection:

- 1. Do parameter setup,just as the Word Length, Baud Rate, Parity Check, Work Mode, Error Injection, Repetition and so on;
- 2. Press the "Apply setting" button to enable the changes to parameters;
- 3. Add the data you want to send;
- 4. Press the "Send" buttons to send the data you added.



Figure 4-35 Error injection operation

**Note**: User must press the "Apply Setting" button to to make the error injection effecticve.

#### 4.6.2 Automatic error detection in receive channels

Some error ( such as: **Parity Error**, **Gap Error**, **Short Word Error**, **Long Word Error**) can be automatic detected when you enable the automatic error detection by select the button-- "Error Detection".



#### Step about data receiving with automatic error detection:

- 1. Do parameter setup,just as the Word Length, Baud Rate, Parity Check, SDI Decode, Label Check and so on;
- 2. Select the channel which you want to it's parameters;
- 3. Press the "Apply setting" button to enable the changes to parameters;
- 4. Select the the "Error Detection" button to enable the error detection function, each channel has its own "Error Detection" button;
- 5. Press the "RX-1 Start"/"RX-2 Start"/"RX-3 Start"/"RX-4 Start" buttons to begin the Data Receiving operation.

Once some error detect from the receiving data, they will be display on the "Receive Data" area(just as: PE, GE, SE, LE, PG), and the total number of the error detected will be display on the "Error Detection" area. The user can clear them by pressing "RX-1 Clear"/"RX-2 Clear"/"RX-3 Clear"/"RX-4 Clear" button.

Error	Error description
PE	Parity Error
GE	Gap Error
SE	Short Word Error
LE	Long Word Error
PG	Parity and Gap Error

#### Table 4-4 Error detect form Receiving data



😻 EMBC1000-PCI429EI-42		Error Display		
RX-1234   TX-1   TX-2   About EM		7		
Parameter Setting		Receive	e Data	
Word Length 32 T BTT	RX-1	RX-2	RX-3	RX-4
123	45778 PE	12345778 PE	12345778 PE	12345778 PE
Baud rate 100 - Kbps 123	45744 PE	12345744 PE 12345910 PE	12345744 PE 12345910 PE	12345744 PE 12345910 PE
123	459DC PE	123459DC PE	123459DC PE	123459DC PE
Parity check odd - 123	45778 PG	12345980 FE 12345778 PG	12345980 FE	12345940 FE
ETT 1 1 1: 12 123	45744 PG	12345744 PG 12345910 PG	12345744 PG 12345910 PG	12345744 PG 12345910 PG
SDI decode disable 123	459DC PG	123459DC PG	123459DC PG	123459DC PG
Label check disable V 48D	14179 SE	48D14179 SE	48D14179 SE	48D14179 SE
	14189 SE 16021 SE	48D14189 SE 48D16021 SE	48D14189 SE 48D16021 SE	48D14189 SE 48D16021 SE
LABEL (H) 48D	160ED SE	48D160ED SE	48D160ED SE	48D160ED SE
L1 00 123	45678 LE	12345678 LE	12345678 LE	12345678 LE
12 00 🔽 Same as Li 123	45644 LE 45810 LE	12345644 LE 12345810 LE	12345644 LE 12345810 LE	12345644 LE 12345810 LE
L3 00 📈 Same as L1 123	458DC LE	123458DC LE	123458DC LE	123458DC LE
L4 00 🔽 Same as L1 123	45678	12345678	12345678	12345678
L5 00 🔽 Same as L1 123	45644	12345644 12345810	12345644 12345810	12345644 12345810
LS 00 🔽 Same as L1 123	458DC	123458DC 12345848	123458DC 12345848	123458DC 12345848
L7 00 🔽 Same as L1	100/10		12010000	
Salast Channel				
I I I I I I I I I I I I I I I I I I I	25	25	25	25
🔽 RX-3 🔽 RX-4	RX-1 Stop	RX-2 Stop	RX-3 Stop	RX-4 Stop
	RX-1 Clear	RX-2 Clear	RX-3 Clear	RX-4 Clear
	RX-1 Save	RX-2 Save	EX-3 Save	RX-4 Save
Dave Setting Load Setting Er	ror Detection	F Error Detection	V Error Detection	F Error Detection
A PAR: GAP:	10 4	PAR: 10 GAP: 4	PAR: 10 GAP: 4	PAR: 10 GAP: 4
Apply Setting	5	S_W: 5	S_W: 5	S_W: 5 I W: 5
Lan.	odd SDT-d	isable LABFLidisable		
ameters setup	oud DDI.u	ISADIC LADIS. GISADI		

Figure 4-32 error detection operation



# **CHAPTER 5 Data Analysis Software**

The ARINC429 data received can be analyzed with the provided **Data Analysis** software. The software will deal with the data either saved in the .txt data file or in the data field entered on-line.

Double click on the software Data Convert.exe(**G:\DataAnalysis**), you will see the main window showed in **Figure 5-1**.

-File Operation		Da	ta Operation		
Source File	Word Length Begin Data()	Seclet: 🛛	32 🗾	Bit er(D)	
Target File	₩ N				Conver
Convert	WORD	PAR	SDI SSM	LAB	DATA
Convert number:					

Figure 5-1 main window of Data Convert

The software will deal with the data either saved in the .txt data file or in the data operation field entered on-line.

#### 1) Data Conversion from a Data File

In this way, user can analyze the ARINC429 data saved in the \*.txt file (assume Source File.txt).

Click the "Source File" button to open the data file.

The ARINC429 data saved in the Source File are in Hex, the first line is always the



definition of the word length (25 or 32 bit), following are the data (each word per line) , shown in **Figure 5-3.** 

Fi S	le Operation	1	Word Length	Dat: Seclet: 🛉 32	a Operatio	n Bit	
	Open Look in: C SuorceFile TargetFile	32_25chan	9	•	⇔ Ē ↔	<b>?</b> ►	nvert ATA
Con	File <u>n</u> ame: Files of <u>type</u> :	SuorceFile				<u>O</u> pen Cancel	
	Bit						4

#### Figure 5-2 open a source file

Source	e File - N	lotep	ad	▶	
<u>Eile E</u> dit	F <u>o</u> rmat	<u>V</u> iew	Help		
Lie Lui wordlen 0000000 0000000 0000000 0000000 000000	rginal gth:321 1 2 3 4 5 5 6 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9	bit	Ūeih		
0000001 0000001 0000001	7 8 9				~

#### Figure 5-3 Source File



Then, you need to create a target file to save such conversion. Click the **"Target File**" button to open a \*.txt file (assume Target File.txt) which saved the results of the analysis (actually, conversion results so far), shown in **Figure 5-4.** 

See Data Convert	X
File Operation Data Operation Data Operation Bit	
Open ?X	
Look jn: 🗀 32_25chang 💌 🖛 🗈 💣 🏢 🗸 avert	
E SuorceFile	[
Cont	
File <u>n</u> ame: TargetFile <u>Open</u>	
Word Files of type: *.txt Cancel	
Bit	
Exit	

#### Figure 5-4 open Target File

Click the "**Convert**" button, when the analysis (conversion) is completed, the "Convert **number**" area will show the total number of the words analysed. And then you can open the Target File(Target File.txt) to study the results, shown in **Figure 5-5** and **Figure 5-6**.



Rile Oremetics	- V	lata Ozanatiaz	
file Operation	n n	ata Operation	
Source File	Word Length Seclet:	32 💌 Bit	າາ
Target File			Convert
Convert	WORD PAR	SDI SSM	LAB DATA
Stop Convert number:	32_25change X Complete!		
121 'ord Length:		1	



🚺 Tar	get l	File - I	Notep	ad			×
<u>File E</u>	dit F	ormat	⊻iew	Help			
WOR 00000 00000 00000 00000 00000 00000 0000	D D 0001 0002 0003 0004 0005 0006 0007 0008 00000 0000 00000 0000 0000 0000 0000 0000 0000 0000 0000	PAR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SDI 00 00 00 00 00 00 00 00 00 00 00 00 00	SSM 00 00 00 00 00 00 00 00 00 00 00 00 00	LAB 01 02 03 04 05 06 07 08 00 00 00 00 00 00 00 11 12 13 14 15 16 17 18	DATA 000000 00000 00000 00000 00000 00000 0000	
00000	019	0	00	00	19	00000	~

Figure 5-6 Target File



#### 2) Data Conversion in Data Operation field

There is a dedicated data entry area under "**Data Operation**". One may simply create a set of data by using the base digital number (in Hex, here we termed it to be "**Begin Data**"), defining the increment(accept 0 to 0xFFFFF) and the toal number(accept 0 to 65535) of data in the respective field.

Once the data is configured, you may click on the "Convert" button, it will automatically generate a set of data and perform conversion over them automatically. The attributes such as PAR, SDI, SSM, LAB, DATA, etc. will be generated and displayed in the "Data Display Area", shown in Figure 5-7.

-File Uperation	5		Data Ope	ration —			
Source File	Word Length S Begin Data(He	Seclet: ex) Incr	32 ement (He:	▼ Bi x) Number	t (D)		
Target File	12345678		.23	100		Convert	
Convert	WORD	PAR	SDI	SSM	LAB	DATA	~
convert	12345678	0	10	11	78	091A2	
	1234579B	1	10	11	9B	091A2	
stop (	123458BE	0	11	00	BE	091A2	_
-~P	123459E1	1	11	00	E1	091A2	
	12345B04	1	11	01	04	091A2	
	12345C27	0	11	10	27	091A2	
	12345D4A	1	11	10	4A	091A2	
	12345E6D	0	11	11	6D	091A2	
ert number:	12345F90	1	11	11	90	091A2	
	123460B3	0	00	00	B3	091A3	
121	123461D6	1	00	11	D6	091A3	
	123462F9	0	00	0	F9	091A3	
an an agus an	1234641C	0	00	10	YC.	091A3	
ength:	1234653F	1	00	10	$\langle \rangle$	091A3	
	12346662	0	00	11	Z	Q1A3	
25 Bit	12346785	1	00	11	Det	- Diaute	
		10 <b>-</b> 12			Data	a Disdia	VA

Figure 5-7 data conversion in Data Operation field





## **CHAPTER 6 Develop Your own Application Software**

To allow the user to develop his own application software or project, EMBC1000-PCI429EI card comes with drivers software, API (Application Programming Interface) library and user oriented application software, running under Windows 2000 or Windows XP. The user oriented application software has been designed with the capabilities of simulating the outputs of various airborne systems, receiving inputs from these systems, and providing bus data analysis functions. API library is also provided together with example source code (Visual C++), which allow users to easily develop their own application software or project based on the real world applications.

# **6.1 Application Programming Interface**

When you begin to develop your own application software for this PCI card, you should finish the settings below in you project(build in Visual C++ 6.0):

Copy PCI\_LINK3.dll, PCI\_LINK3.lib, wdapi901.dll, wdapi901.lib (you can find them in CD-ROM\UserDesign\APILibrary) to your project working directory.

Add the PCI\_LINK3.lib and wdapi901.lib to the project: Project $\rightarrow$ Setting $\rightarrow$ Link, shown in **Figure 6-1**:

roject Settings	?
Settings For: Win32 Debug 🔹	General   Debug   C/C++ Link   Resources   M
EMBC1000 - PCI429 - 42 - e Source Files EMBC1000 - PCI429 - EMBC1000 - PCI429 - EMBC1000 - PCI429 - StdAfx.cpp Header Files EMBC1000 - PCI429 - EMBC1000 - PCI420 - EMBC1000 - PCI420 - EMBC1000 - EMBC100 - EMBC100 - EMBC100 - EMBC100 - EMBC100 - EMBC100 - EMBC100 -	Category:       General       Reset         Output file name:       Debug/EMBC1000 - PCI429 - 42 - example.exe         Object/library modules:       PCI_LINK3.lib wdapi901.lib         Image: PCI_LINK3.lib wdapi901.lib       Ignore all default libraries         Image: Link incrementally       Generate mapfile         Image: Project Options:       PCI_LINK3.lib wdapi901.lib /nologo
x b	/subsystem:windows /incremental:yes /pdb:"Debug/EMBC1000 - PCI429 - 42 - example.pdb"
	OK Cancel

Figure 6-1 Add the PCI\_LINK3.lib and wdapi901.lib to the project



Edit the head file, refer to the source code below:

```
// EMBC1000-PCI429-42-exampleDlg.h : header file
#if !defined(AFX_EMBC1000PCI42942EXAMPLEDLG_H__5ACC0538_C64A_4316_8
DDD_9A1CA8B3A7CF_INCLUDED_)
#define
AFX EMBC1000PCI42942EXAMPLEDLG H 5ACC0538 C64A 4316 8DDD 9A1C
A8B3A7CF INCLUDED
#if MSC VER > 1000
#pragma once
#endif // _MSC_VER > 1000
typedef void * WDC DEVICE HANDLE;
// CEMBC1000PCI42942exampleDlg dialog
class CEMBC1000PCI42942exampleDlg : public CDialog
{
// Construction
public:
   CEMBC1000PCI42942exampleDlg(CWnd* pParent = NULL); //standard
constructor
// Dialog Data
   //{{AFX DATA(CEMBC1000PCI42942exampleDlg)
   enum { IDD = IDD EMBC1000PCI42942EXAMPLE DIALOG };
   //}}AFX DATA
   // ClassWizard generated virtual function overrides
   //{{AFX VIRTUAL(CEMBC1000PCI42942exampleDlg)
   protected:
   virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV support
   //}}AFX VIRTUAL
// Implementation
protected:
   HICON m_hlcon;
   // Generated message map functions
   //{{AFX MSG(CEMBC1000PCI42942exampleDlg)
   virtual BOOL OnInitDialog();
   afx_msg void OnSysCommand(UINT nID, LPARAM IParam);
   afx msg void OnPaint();
   afx msg HCURSOR OnQueryDraglcon();
   afx_msg void OnInit();
   afx_msg void OnTX1Init();
   afx_msg void OnTX2Init();
   afx msg void OnRX1Init();
   afx_msg void OnRX2Init();
   afx msg void OnRX3Init();
   afx_msg void OnRX4Init();
```



```
afx msg void OnSendDataTX2();
    afx_msg void OnSendDataTX1();
    afx msg void OnClose();
    afx msg void OnRxread();
    afx msg void OnRx1read();
    afx_msg void OnRx2read();
    afx_msg void OnRx3read();
    afx msg void OnRx4read();
    //}}AFX_MSG
    DECLARE MESSAGE MAP()
};
#pragma comment(lib,"PCI_LINK3.lib")
extern "C" _declspec(dllexport) WDC_DEVICE_HANDLE DeviceFindAndOpen(int
Board num);
extern "C" declspec(dllexport) DWORD OBT429TOPCI LibInit(void);
extern "C" _declspec(dllexport)
                                   int RX_DATA( WDC_DEVICE_HANDLE hDev,
unsigned int RX N, unsigned int RX DATA[256], unsigned int fifo leve);
extern "C" _declspec(dllexport)
                                   int TX1_SEND(WDC_DEVICE_HANDLE hDev,
unsigned int txdata[64], int fifo leve);
extern "C" declspec(dllexport)
                                   int TX2_SEND(WDC_DEVICE_HANDLE hDev,
unsigned int txdata[64], int fifo_leve);
extern "C" _declspec(dllexport) void RX1_INIT(WDC_DEVICE_HANDLE hDev,
              unsigned int enable,
                                      // (0, 1)
              unsigned int word length,
                                              // (25, 32)
              unsigned int parity_select,
                                          // (0,1,2)
              unsigned int label enable,
                                          // (0,1)
              unsigned int sdi,
                                     // (0,1)
              unsigned int scaler,
                                              // (0~65535)
              unsigned int label1,
              unsigned int label2,
              unsigned int label3,
              unsigned int label4,
              unsigned int label5,
              unsigned int label6,
               unsigned int label7
              );
extern "C" _declspec(dllexport) void RX2_INIT(WDC_DEVICE_HANDLE hDev,
               unsigned int enable,
                                      // (0, 1)
                                              // (25, 32)
               unsigned int word length,
              unsigned int parity select,
                                          // (0,1,2)
              unsigned int label_enable, // (0,1)
              unsigned int sdi,
                                      // (0,1)
              unsigned int scaler,
                                              // (0~65535)
               unsigned int label1,
```



unsigned int label2,
unsigned int label3,
unsigned int label4,
unsigned int label5,
unsigned int label6,
unsigned int label7
);
extern "C" _declspec(dllexport) void RX3_INIT(WDC_DEVICE_HANDLE hDev,
unsigned int enable, // (0, 1)
unsigned int word_length, // (25, 32)
unsigned int parity_select, // (0,1,2)
unsigned int label_enable, // (0,1)
unsigned int sdi, // (0,1)
unsigned int scaler, //
unsigned int label1,
unsigned int label2,
unsigned int label3,
unsigned int label4,
unsigned int label5,
unsigned int label6,
unsigned int label7
);
extern "C" _declspec(dllexport) void RX4_INIT(WDC_DEVICE_HANDLE hDev,
unsigned int enable, // (0, 1)
unsigned int word_length, // (25, 32)
unsigned int parity_select, // (0,1,2)
unsigned int label_enable, // (0,1)
unsigned int sdi, // (0,1)
unsigned int scaler, // (0~65535)
unsigned int label1,
unsigned int label2,
unsigned int label3,
unsigned int label4,
unsigned int label5,
unsigned int label6,
unsigned int label7
);
extern "C" declspec(dllexport) void TX1 INIT(WDC DEVICE HANDLE hDev,
unsigned int enable, unsigned int word_length, unsigned int parity, unsigned int scaler,
unsigned int tgap, unsigned int mode);
extern "C" _declspec(dllexport) void TX2_INIT(WDC_DEVICE_HANDLE hDev,
unsigned int enable, unsigned int word_length, unsigned int parity, unsigned int scaler,
unsigned int tgap, unsigned int mode);
extern "C" _declspec(dllexport) void RX_ENABLE(WDC_DEVICE_HANDLE



hDev, unsigned int num, unsigned int enable); extern "C" \_declspec(dllexport) void TX\_ENABLE(WDC\_DEVICE\_HANDLE hDev, unsigned int num, unsigned int enable); extern "C" \_declspec(dllexport) void DeviceClose(WDC\_DEVICE\_HANDLE hDev); extern "C" declspec(dllexport) void OBT429 RESET(WDC DEVICE HANDLE hDev); //{{AFX\_INSERT\_LOCATION}} // Microsoft Visual C++ will insert additional declarations immediately before the previous line. #endif // !defined(AFX\_EMBC1000PCI42942EXAMPLEDLG\_H\_\_5ACC0538\_C64A\_4316\_8D DD 9A1CA8B3A7CF INCLUDED )

Now the API setup is done. When you build your project, VC++ will link the APIs automatically and add them to your project.

## 6.2 Example Source Code

This example source code will show you how to use the API. Detailed info is given in CD-ROM (G:\ UserDesign\sample).

# 6.3 API Function Description

Name	DWORD OBT429TOPCI_LibInit(void)
Function	Initialize the card
Parameter	Void
Return Value	DWORD

1. Card Initialization Function: OBT429TOPCI\_LibInit()

#### 2. Getting the handle: DeviceFindAndOpen()

Name	WDC_DEVICE_HANDLE DeviceFindAndOpen(int Board_num)						
Function	Get the handle of card						
Parameter	Board_num: the number of the PCI slot which the card insertedpossible assignment value : 0 to 255						
Return Value	WDC_DEVICE_HANDLE hDevX						

3. Reset the card: OBT429\_RESET()



Name	void OBT429_RESET(WDC_DEVICE_HANDLE hDev)
Function	Reset the card (Note: After resetting, all channels' parameters will be the default value.)
Parameter	hDev: handle of the card
Return Value	void

## 4. Set the parameters of RX-1: RX1\_INIT()

	void RX1_INIT	(WDC_DEVICE_HANDLE hDev,			
	unsigned int er	nable, unsigned int word_length,			
	unsigned int parity_select, unsigned int label_enable,				
	unsigned int sdi,unsigned int scaler,				
Name	unsigned int fif	o_half_level, unsigned int label1,			
	unsigned int la	bel2, unsigned int label3,			
	unsigned int la	bel4, unsigned int label5,			
	unsigned int la	bel6, unsigned int label7			
	)				
Function	Set the first red	ceive channel's parameters			
	hDev:	handle of the card			
	enable:	enable the receive channel(RX1)			
		possible assignment value : 0:disable 1: enable			
	word_length:	word length			
		possible assignment value : 0:32bits 1: 25bits			
	parity_select:	set parity check			
		possible assignment value: 0:disable 1: odd 2: even			
	label_enable:	enable the label check			
		possible assignment value : 0:disable 1: enabl			
	sdi:	set SDI check			
Parameter		possible assignment value : 0:disable 1: SDI=00			
		2: SDI=01 3: SDI=10 4: SDI=11			
	Scaler:	set Buad rate			
	possible assigi	nment value : 0:12.5Kbps 1: 48Kbps			
		2: 100Kbps 3: 50Kbps			
	fifo_half_level:	set fifo_half level			
		possible assignment value: 0~1023			
	label1:	possible assignment value: 0~0xFF			
	label2:	possible assignment value: 0~0xFF			
	label3:	possible assignment value: 0~0xFF			
	label4:	possible assignment value: 0~0xFF			



	label5:	possible assignment value: 0~0xFF
	label6:	possible assignment value: 0~0xFF
	label7:	possible assignment value: 0~0xFF
Return	waid	
Value	νοια	

## 5. Set the parameters of RX-2: RX2\_INIT()

	void RX2_INIT(WD0				
	unsigned int enable,	unsigned int word_length,			
	unsigned int parity_select, unsigned int label_enable,				
	unsigned int sdi, unsigned int scaler,				
Name	unsigned int fifo hal	f level, unsigned int label1,			
	unsigned int label2, unsigned int label3.				
	unsigned int label4,	unsigned int label5.			
	unsigned int label6,	unsigned int label7			
	)				
Function	Set the second rece	ive channel's parameters			
	hDev: hand	le of the card			
	enable: enab	le the receive channel(RX2)			
	poss	sible assignment value: 0:disable 1: enable			
	word_length: wor	dlength			
	poss	sible assignment value: 0:32bits 1: 25bits			
	parity_select: set p	parity check			
	poss	sible assignment value: 0:disable 1: odd 2: even			
	label_enable: enab	ble the label check			
	poss	sible assignment value: 0:disable 1: enabl			
	sdi: set s	SDI check			
	pos	sible assignment value: 0:disable 1: SDI=00			
Darameter		2: SDI=01 3: SDI=10 4: SDI=11			
Parameter	Scaler: set	Buad rate			
	possible assignment	t value: 0:12.5Kbps 1: 48Kbps			
		2: 100Kbps 3: 50Kbps			
	fifo_half_level: set	fifo_half level			
	pos	sible assignment value: 0~1023			
	label1: pos	sible assignment value: 0~0xFF			
	label2: pos	sible assignment value: 0~0xFF			
	label3: pos	sible assignment value: 0~0xFF			
	label4: pos	sible assignment value: 0~0xFF			
	label5: pos	sible assignment value: 0~0xFF			
	label6: pos	sible assignment value: 0~0xFF			
	label7: pos	sible assignment value: 0~0xFF			
Return Value	void				



## 6. Set the parameters of RX-3: RX3\_INIT()

	void RX3_INIT(WDC_DEVICE_HANDLE hDev, unsigned int enable, unsigned int word_length, unsigned int parity_select,unsigned int label_enable, unsigned int sdi,unsigned int scaler,					
Name	unsigned int fifo_half_level, unsigned int label1,					
	unsigned int label2, unsigned int label3,					
	unsigned int label4, unsigned int label5,					
	unsigned int labelo, unsigned int label?					
	)					
Function	Set the third receive channel's parameters					
	hDev: handle of the card					
	enable: enable the receive channel(RX3)					
	possible assignment value: 0:disable 1: enable					
	word_length: word length					
	possible assignment value: 0:32bits 1: 25bits					
	parity_select: set parity check					
	possible assignment value: 0:disable 1: odd 2: even					
	label_enable: enable the label check					
	possible assignment value: 0:disable 1: enabl					
	sdi: set SDI check					
	possible assignment value: 0:disable 1: SDI=00					
Parameter	2. 301-01 3. 301-10 4. 301-11					
	l nossible assignment value: 0:12 5Kbns 1: 48Kbns					
	2: 100Kbps 3: 50Kbps					
	fifo half level: set fifo half level					
	possible assignment value: 0~1023					
	label1: possible assignment value: 0~0xFF					
	label2: possible assignment value: 0~0xFF					
	label3: possible assignment value: 0~0xFF					
	label4: possible assignment value: 0~0xFF					
	label5: possible assignment value: 0~0xFF					
	label6: possible assignment value: 0~0xFF					
	label7: possible assignment value: 0~0xFF					
Return	void					
Value	VOIU					

7. Set the parameters of RX-4: RX4\_INIT()



	void RX4_INIT	(WDC_DEVICE_HANDLE hDev,
	unsigned int er	hable, unsigned int word length,
	unsigned int pa	arity select, unsigned int label enable,
	unsigned int so	li.unsigned int scaler.
Name	unsigned int fif	o half level unsigned int label1.
	unsigned int la	bel2 unsigned int label3
	unsigned int la	bel4 unsigned int label5
	unsigned int la	bel6 unsigned int label7
	)	
Function	Set the fourth r	receive channel's parameters
	hDev:	handle of the card
	enable:	enable the receive channel(RX4)
		possible assignment value: 0:disable 1: enable
	word_length:	word length
		possible assignment value: 0:32bits 1: 25bits
	parity_select:	set parity check
		possible assignment value: 0:disable 1: odd 2: even
	label enable:	enable the label check
	_	possible assignment value: 0:disable 1: enabl
	sdi:	set SDI check
		possible assignment value: 0:disable 1: SDI=00
_		2: SDI=01 3: SDI=10 4: SDI=11
Parameter	Scaler:	set Buad rate
	possible assigr	nment value: 0:12.5Kbps 1: 48Kbps
		2: 100Kbps 3: 50Kbps
	fifo half level:	set fifo half level
		possible assignment value: 0~1023
	label1:	possible assignment value: 0~0xFF
	label2:	possible assignment value: 0~0xFF
	label3:	possible assignment value: 0~0xFF
	label4:	possible assignment value: 0~0xFF
	label5:	possible assignment value: 0~0xFF
	label6:	possible assignment value: 0~0xFF
	label7:	possible assignment value: 0~0xFF
Return		
Value	void	

8. Set the parameters of TX-1: TX1\_INIT()



Name	void TX1_INIT	(WDC_DEVICE_HANDLE hDev,
	unsigned int enable, unsigned int word_length,	
	unsigned int parity, unsigned int scaler,	
	unsigned int fifo half level, unsigned int tgap,	
	unsigned int m	ode
	)	
Function	Set the first tra	nsmit (TX1) channel's parameters
	hDev:	handle of the card
	enable:	enable the transmit channel(TX1)
		possible assignment value: 0:disable 1: enable
	word_length:	word length
		possible assignment value: 0:32bits 1: 25bits
	parity_select:	set parity check
		possible assignment value: 0:disable 1: odd 2: even
Parameter	Scaler:	set Buad rate
Falametei		possible assignment value: 0:12.5Kbps 1: 48Kbps
		2: 100Kbps 3: 50Kbps
	fifo_half_level:	set fifo_half level
		possible assignment value: 0~1023
	tgap:	set the word gap
		possible assignment value: 4 to 255
	mode:	select the work modle
		possible assignment value: 0: Loop 1: Normal
Return	Void	
Value	VOIU	

## 9. Set the parameters of TX-2: TX2\_INIT()

Name	void TX2_INIT(WDC_DEVICE_HANDLE hDev, unsigned int enable, unsigned int word_length, unsigned int parity, unsigned int scaler, unsigned int fifo_half_level, unsigned int tgap, unsigned int mode )
Function	Set the second transmit (TX2) channel's parameters



	hDev:	handle of the card
	enable:	enable the transmit channel(TX2)
		possible assignment value: 0:disable 1: enable
	word_length:	word length
		possible assignment value: 0:32bits 1: 25bits
	parity_select:	set parity check
		possible assignment value: 0:disable 1: odd 2: even
Deremeter	Scaler:	set Buad rate
Parameter		possible assignment value: 0:12.5Kbps 1: 48Kbps
		2: 100Kbps 3: 50Kbps
	fifo_half_level:	set fifo_half level
		possible assignment value: 0~1023
	tgap:	set the word gap
		possible assignment value: 4 to 255
	mode:	select the work modle
		possible assignment value: 0: Loop 1: Normal
Return	void	
Value	νοια	

## 10. Send data via the TX1: TX1\_SEND()

Name	int TX1_SEND(WDC_DEVICE_HANDLE hDev, unsigned int txdata[1024], int fifo_leve )	
Function	Send dat a from the first transmit (TX1) channel	
Parameter	hDev:handle of the cardtxdata[1024]:the array for the words which are going to sendfifo_leve:the number of the words which are going to sendpossible assignment value:1 to 1024	
Return Value	<ul><li>0: means sending data faild</li><li>1: means sending data successful</li><li>Note: The return value should be used to judge if sending data successfulor not.</li></ul>	

## 11. Send data via the TX2: TX2\_SEND()

Name	int TX2_SEND(WDC_DEVICE_HANDLE hDev, unsigned int txdata[1024], int fifo_leve )
Function	Send dat a from the second transmit (TX2) channel



Parameter	hDev: handle of the card txdata[1024]: the array for the words which are going to send fifo_leve: the number of the wordswhich are going to send possible assignment value: 1 to 1024
Return Value	0: means sending data faild 1: means sending data successful Note: The return value should be used to judge if sending data successfulor not.

#### 12. Receive data from RX-1, RX-2, RX-3, RX-4: RX\_DATA()

Name	int RX_DATA( WDC_DEVICE_HANDLE hDev, unsigned int RX_N, unsigned int RX_DATA[1024], unsigned int RX_ER_STA[1024], unsigned int fifo_leve )
Function	Receive data throug 4 receive(RX1/RX2RX3/RX4) channels
Parameter	hDev:handle of the cardRX_N:select the receive channelpossible assignment value:1: select RX12:select RX23: select RX3RX_DATA[1024]:the array for the words which receving from outside.RX_ER_STA[1024]:the array for the error detect from the receving channal shen enable the error detection.fifo_leve:the number of the wordswhich are going to send possible assignment value:1to 1024
Return Value	The number of the data receive

#### 13、Releas card: DeviceClose ()

Name	void DeviceClose(WDC_DEVICE_HANDLE hDev)	
Name	Close the card and release the handle of the card.	
Parameter	hDev: handle of the card	
Return Value	void	



14. Check if the receive (RX1/RX2/RX3/RX4) channel's FIFO is empty or not:: RX\_FIFO\_STATUS ()

Name	int RX_FIFO_STATUS(WDC_DEVICE_HANDLE hDev,
	unsigned int RX_N
	)
Function	Check if the receive (RX1/RX2/RX3/RX4) channel's FIFO is empty or
	not
	hDev: handle of the card
Paramotor	RX_N: select the receive channel
Falameter	possible assignment value: 1: select RX1 2: select RX2
	3: select RX3 4: select RX4
Return Value	0: means the channel's FIFO is empty
	1: means the channel's FIFO is full
	2: means the channel's FIFO is half_full (the number of the data in fifo
	is bigger than the "fifo_half_level" you seted.)
	Others: means some mistake, such as the TX_N is invalid.

#### 15. Check if the transmit(TX1/2) channel's FIFO is empty or not: TX\_FIFO\_STATUS ()

Name	int TX_FIFO_STATUS(WDC_DEVICE_HANDLE hDev, unsigned int TX_N )
Function	Check if the transmit(TX1/TX2) channel's FIFO is empty or not
Parameter	hDev: handle of the card TX_N: select the transmit channel possible assignment value: 1: select TX1 2: select TX2
Return Value	<ul> <li>0: means the channel's FIFO is empty</li> <li>1: means the channel's FIFO is empty</li> <li>2: means the channel's FIFO is half_full (the number of the data in fifo is bigger than the "fifo_half_level" you seted.)</li> <li>Others: means some mistake.such as the TX N is invalid.</li> </ul>

#### 16、Reset the any channel's FIFO: FIFO\_RESET ()

Name	void FIFO_RESET(WDC_DEVICE_HANDLE hDev, unsigned int CHANNEL )
Function	Reset the any channel's(RX1/RX2/RX3/RX4/TX1/TX2) FIFO



Parameter	hDev: CHANNEL:	handle of the card select channel possible assignment value: 1: select RX1 2: select RX2 3: select RX3 4: select RX4 5: select TX1 6: select TX2
Return Value	void	

#### 17、Enable the receive(RX!/RX2/RX3/RX4) channel: RX\_ENABLE ()

Name	void RX_ENABLE( WDC_DEVICE_HANDLE hDev, unsigned int num,unsigned int enable )						
Function	Enable the receive(RX!/RX2/RX3/RX4) channel						
Parameter	hDev: handle of the card num: select the receive channel possible assignment value: 1: select RX1 2: select RX2 3: select RX3 4: select RX4 enable: enable the receive channel possible assignment value: 0:disable 1: enable						
Return Value	void						

## 18、Enable the transmit(TX1/TX2) channel: TX\_ENABLE ()

Name	void TX_ENABLE( WDC_DEVICE_HANDLE hDev, unsigned int num,unsigned int enable )
Function	Enable the transmit(TX1/TX2) channel
Parameter	hDev: handle of the card num: select the transmit channel possible assignment value: 1: select TX1 2: select TX2 enable: enable the transmit channel possible assignment value: 0:disable 1: enable
Return Value	void



19. Enable Automatic send repetition in the transmit(TX1/TX2) channel:

#### TX\_RE\_ENABLE ()

Name	void TX_RE_ENABLE(WDC_DEVICE_HANDLE hDev, unsigned int num,unsigned int enable )					
Function	Enable Automatic send repetition in the transmit(TX1/TX2) channel					
Parameter	hDev: handle of the card num: select the transmit channel possible assignment value: 1: select TX1 2: select TX2 enable: enable or disable possible assignment value: 0:disable 1: enable					
Return Value	void					

 $20\,{}_{\times}$  Fill fifo when enable the automatic send repetition in the transmit (TX1/TX2) channel:TX\_RE\_FILLFIFO ()

Name	Void TX_RE_FILLFIFO(WDC_DEVICE_HANDLE hDev, unsigned int num,unsigned int txdata[1024], int fifo_level )						
Function	Fill fifo when enable the automatic send repetition in the transmit (TX1/TX2) channel						
Parameter	hDev:       handle of the card         num:       select the transmit channel         possible assignment value:       1: select TX1 2: select TX2         txdata [1024]:       the array for the words which are going to send         automatically       possible assignment value:         fifo_leve:       the number of the words which are going to send         automatically       possible assignment value:         fifo_leve:       the number of the words which are going to send         automatically       possible assignment value:         1: enable       the number of the words which are going to send						
Return Value	void						



21. Set the time gap when enable Automatic send repetition in the transmit(TX1/TX2) channel: TX\_RE\_SETTIME ()

Name	void TX_RE_SETTIME(WDC_DEVICE_HANDLE hDev, unsigned int num,unsigned int timegap )						
Function	Set the time gap when enable Automatic send repetition in the transmit(TX1/TX2) channel						
Parameter	hDev: handle of the card num: select the transmit channel possible assignment value: 1: select TX1 2: select TX2 timegap: Refers to the time gap between two Repetition transmits possible assignment value:1~99,999,999 us						
Return Value	void						

22、Read back the total number of the word have be send in the transmit(TX1/TX2) channel:TX\_RE\_ SEND ()

Name	unsigned int TX_RE_SEND(WDC_DEVICE_HANDLE hDev, unsigned int TX_N )
Function	Read back the total number of the word have be send in the transmit(TX1/TX2) channel
Parameter	hDev: handle of the card TX_N : select the transmit channel possible assignment value: 1: select TX1 2: select TX2
Return Value	unsigned int ,the total number of the word have be send

#### 23、Inject error in the transmit(TX1/TX2) channel: TX\_ER\_INJ()

Name	void TX_ER_INJ(WDC_DEVICE_HANDLE hDev, unsigned int TX_N,unsigned int error )
Function	Inject error in the transmit(TX1/TX2) channel



	hDev: TX_N :	handle of the card select the transmit channel possible assignment value: 1: select TX1 2: select TX2							
	error:	error type							
		possible assignment value: 1: enable gap error							
Doromotor		2: enable parity error							
Parameter		3: enable short word error							
		4: enable long word error							
		11: disable gap error							
		12: disable parity error							
		13: disable short word error							
		14: disable long word error							
Return Value	void								

## 24、Enable/disable error detection in RX-1, RX-2, RX-3, RX-4: RX\_ER\_DETECT\_EN ()

Name	void RX_ER_DETECT_EN(WDC_DEVICE_HANDLE hDev, unsigned int RX_N,unsigned int en )						
Function	Enable/disable error detection in the Rx(RX-1/RX-2/RX-3/RX-4) channel						
Parameter	hDev: handle of the card RX_N : select the receive channel possible assignment value: 1: select RX1 2: select RX2 3: select RX3 4: select RX4 en: enable/disable error detection possible assignment value: 1: enable 2: disable						
Return Value	void						



# CHAPTER 7 PRODUCT ORDERING INFO

Product	Rx	Тх	Baud rate				Software support	
Number	Channel	Channel	100K	50 <b>K</b>	48 <b>K</b>	12. <b>5K</b>		
EMBC1000- PCI429EI-42	4	2	V	V	V	√	Windows 2000 or Windows XP based drivers and application software	



# CHAPTER 8 RELATED PRODUCTS

Product	Interface	Error Injection And Detection	Rx Channel	Tx Channel	Baud rate				Software support
Number					100K	50K	48K	12.5K	Sonware support
EMBC1000- PCI429-42	PCI		4	2	V	V	V	V	Windows 2000 or Windows XP based drivers and application software
EMBC1000- USB429-42	USB		4	2	V	V	V	V	Windows 2000 or Windows XP based drivers and application software
EMBC1000- USB429EI-42	USB	V	4	2	V	V	V	V	Windows 2000 or Windows XP based drivers and application software



## Appendix A: ARINC429 Protocol Introduction

ARINC429 is an international standard for Digital Information Transfer System (DITS). It is application-specific for commercial and transport aircraft. It ignores the complexities of different manufacturers' avionics system interfaces and supplies uniform plat form for system communication.

Based on the requirements of ARINC Specification 429, digital information is transmitted by wires in unidirectional data bus, differential coupling or twisted pairs. So ARINC429 is serial communication. The ARINC429 standard supports High, Low, and Null states.

ARINC data words are always 32 or 25 bits in length. Transmission of sequential words is separated by at least four Null bits. Each ARINC word contains a parity bit, 8-bit label. The label words are quite important in ARINC429 and identify the data type and the parameters associated with it, such as latitude data, longitude data. The rest data bits of the word are divided into different fields based on the label. For making communication completely standardized and to avoid conflicts, all of the flight functions have been given labels and data formats.

When a 32-bit ARINC word is transmitted, each word contains:

- Parity : bit32
- SSM : bit31~30, Sign Status Matrix
- Data : bit29~11
- SDI : bit10~9, Source Destination Identifiers
- Label : bit8~1

The 32-bit ARINC Word typically use the format shown in Table A-1 which includes five primary fields, namely P (parity), SSM, Data, SDI, and Label. Attention, ARINC convention numbers the bits from 1 (LSB) to 32 (MSB), not from 0 to 31 as usually.

The order of 32-bit data word transmitted on ARINC bus is as follows (LSB first): Label(8)- Label(7)- Label(6)- Label(5)- Label(4)- Label(3)- Label(2)- Label(1)- SDI(1)- SDI(2)- Data(1)- Data(2)- Data(3)- Data(4)- Data(5)- Data(6)- Data(7)- Data(8)- Data(9)- Data(10)- Data(11)- Data(12)- Data(13)- Data(14)- Data(15)- Data(16)- Data(17)- Data(18)- Data(19)- SSM(1)- SSM(2)- Parity. The least significant bit of each byte, except the label, is transmitted first, and the label is transmitted ahead of the data in each case.



PARITY	SSM	DATA	SDI	LABEL
32	31~30	29 ~ 11	10~9	8~ 1

Table A-1. 32-bit ARINC Data Word Format

When a 25-bit ARINC word is transmitted, each word contains:

- Parity : bit25
- Data : bit24~9
- Label : bit8~1

The 25-bit ARINC Word typically use the format shown in Table A-2. Attention, ARINC convention numbers the bits from 1 (LSB) to 25 (MSB), not from 0 to 25 as usually.

PARITY	DATA	SDI	LABEL
25	24 ~ 9	10~9	8~ 1

 Table A-2.
 25-bit ARINC Data Word Format

The order of 25-bit data word transmitted on ARINC bus is as follows (LSB first): Label(8)- Label(7)- Label(6)- Label(5)- Label(4)- Label(3)- Label(2)- Label(1)-Data(1) -Data(2) -Data(3) -Data(4) -Data(5) -Data(6) -Data(7) -Data(8) -Data(9) -Data(10) -Data(11) -Data(12) -Data(13) -Data(14) -Data(15) -Data(16) –Parity. The least significant bit(LSB) of each word except the label is transmitted first, and the label is transmitted with MSB first.