ECDR-GC314-XXX-FS
Software User Manual

This manual covers the Echotek ECDR-GC314-FS family of high-speed digital receiver boards

ECDR-GC314-PMC-FS
ECDR-GC314-PCI-FS
ECDR-GC314-PCI-R++OT

REVISION 1.0

ECHOTEK CORPORATION
555 Sparkman Drive
Suite 400
Huntsville, AL 35816
Phone: (256) 721-1911
Fax: (256) 721-9266
E-Mail: Sales@echotek.com
Web Site: http://www.echotek.com

Copyright statement:

Disclaimer:

The information in this document has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Echotek Corporation makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Furthermore, Echotek Corporation reserves the right to make changes to any product herein to improve reliability, function or design, without obligation of Echotek Corporation to notify any person of such revision or changes. Echotek Corporation does not assume any liability arising out of applications or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

Revision History

Rev.	Date	Chapter/Section	Change/Addition
0.4	3/11/03	Preliminary Release	
0.5	07/18/03	Preliminary Release	Updated document to reflect current API development.

Table of Contents

List of Tables

Chapter 1 Introduction

This chapter contains a brief introduction to Echotek Corporation's ECDR-GC314-FS driver code, a list of system requirements and technical support contact information.

Features

The ECDR-GC314-FS driver code is designed for use with Echotek's ECDR-GC314-FS family of high-speed digital receiver boards, which consists of the following boards:

- ECDR-GC314-PMC-FS
- ECDR-GC314-PCI-FS
- ECDR-GC314-R++OT

The ECDR-GC314-FS driver code is designed to support operation under the Windows NT/2000, Linux and VxWorks Operating Systems. This driver is supplied as a statically-linked library for all supported operating systems. The source code is also supplied, along with workspace and project files that can be used to compile the code. The ECDR-GC314-FS driver uses the Jungo WinDriver product and provides the necessary run-time components.

For complete descriptions of supported function calls, refer to *Chapter 4 Application Programming Interface (API)*.

Before Getting Started

Before installing the ECDR-GC314-FS driver code, ensure that your system meets the following requirements:

- The ECDR-GC314-XXX-FS board is installed in your system.

  If the ECDR-GC314-XXX-FS is configured for use with an external clock source, connect the clock input to the ECDR-GC314-XXX-FS. The input clock should have an amplitude range of between -4 dBm and +4 dBm. Please refer to the *ECDR-GC314-XXX-FS User Manual* for exact specifications.

Any clock source with amplitude outside this range may cause the board to malfunction and may damage the board.

- The analog input connected to the ECDR-GC314-XXX-FS should have an amplitude less than or equal to +5 dBm.
-

Any input with amplitude exceeding +5 dBm causes the Analog-to-Digital converter to saturate and may damage the board.

Technical Support

If you need additional technical information or assistance, use the following contact information:

E-Mail: rich@echotek.com

Telephone: 256.721.1911 Ext. 161

Facsimile: 256.721.9266 Ext. 161

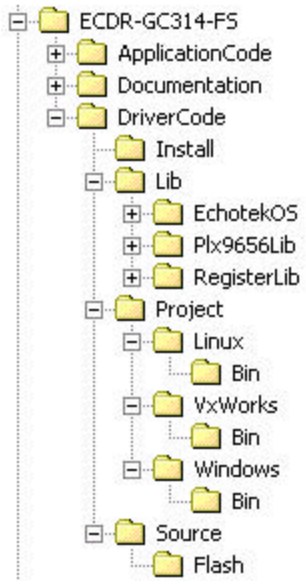
Address: Echotek Corporation
555 Sparkman Drive, Suite 400
Huntsville, Al 35816

Chapter 2
Building and Linking the Driver

This chapter contains information for building and linking the ECDR-GC314-FS driver for all available development environments.

ECDR-GC314-FS Driver Directory Structure

The ECDR-GC314-FS driver is delivered with the following directory structure:



The ECDR-GC314-FS driver directory contains the following subdirectories:

- *DriverCode* – Contains all driver source code, Echotek code Libraries, Installation files and Project files required to compile and build the driver for the supported operating systems. Each operating system has a *Project* subdirectory containing all files needed to compile and build the driver code for the operating system.

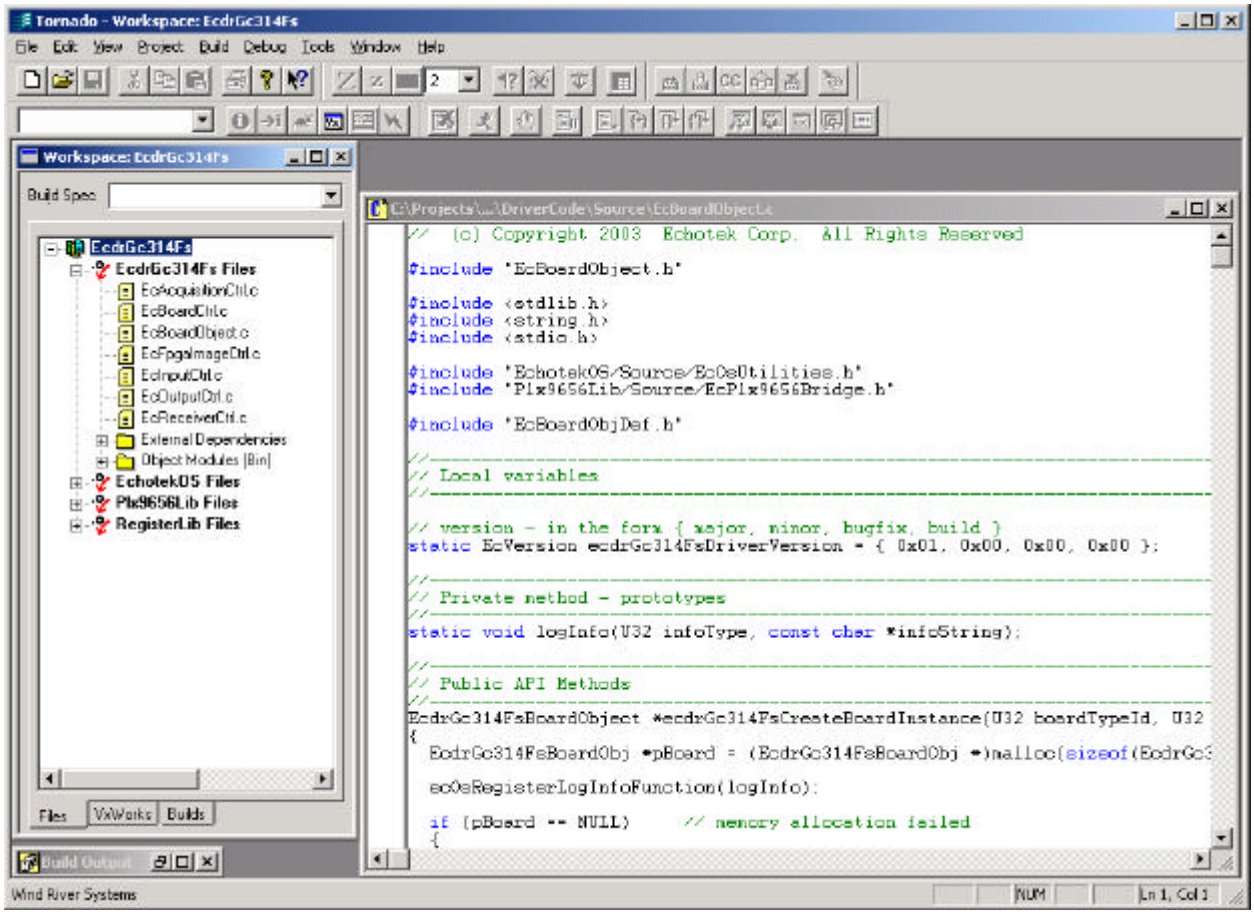
- *Documentation* – Contains user manuals, test files and other supporting documents detailing the operation and usage of the Echotek ECDR-GS314-FS family of digital receiver boards.

- *ApplicationCode* – Contains all application source code, which demonstrates the functionality of the ECDR-GC314-FS boards and may be referenced by users to guide them. The application code is detailed in the *ECDR-GC314-FS Demo Code User Manual*.

Building the Driver for VxWorks

The ECDR-GC314-FS driver was developed using the Tornado II IDE (Integrated Development Environment). All required workspace and project files are included in the *Project/VxWorks* subdirectory. The ECDR-GC314-FS driver code is delivered as a static library object that can be linked with a user's application or system code. The source code and project files are also provided in case a user needs to modify the driver code for a specific application.

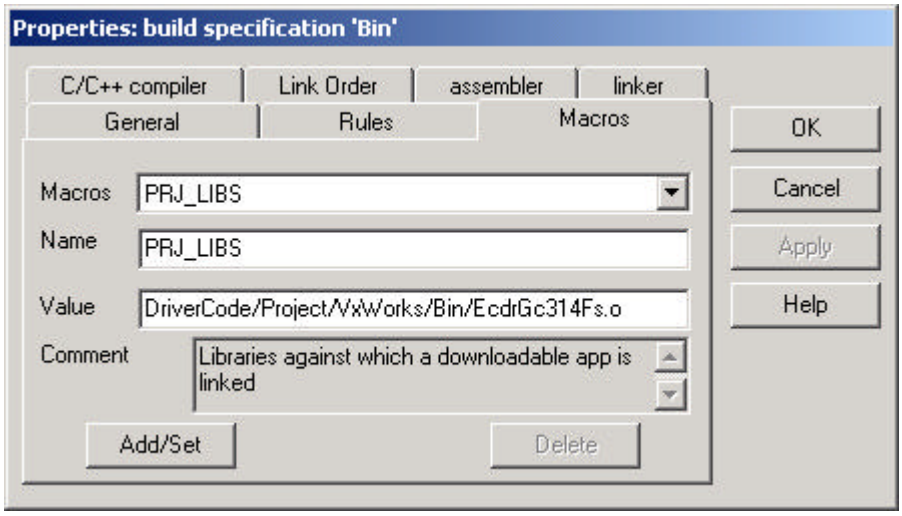
The following shows the Tornado II IDE with the ECDR-GC314-FS driver workspace files.



The EcdrGc314Fs workspace is made up of several projects. The main project is the EcdrGc314Fs project and the others are Echotek code library projects. The output of each project is linked together to form the ECDR-GC314-FS driver static library object (**EcdrGc314Fs.o**), which is in the *Project/VxWorks/Bin* subdirectory.

The driver code is delivered already compiled and linked into the library object. The following steps can be used to link the ECDR-GC314-FS driver library object to a user application.

1. Under the **Builds** setting of the **User Application** project, add the **ECDR-GC314-FS/DriverCode/Project/VxWorks/Bin/EcdrGc314Fs.o** file to the **PRJ_LIBS** macro.

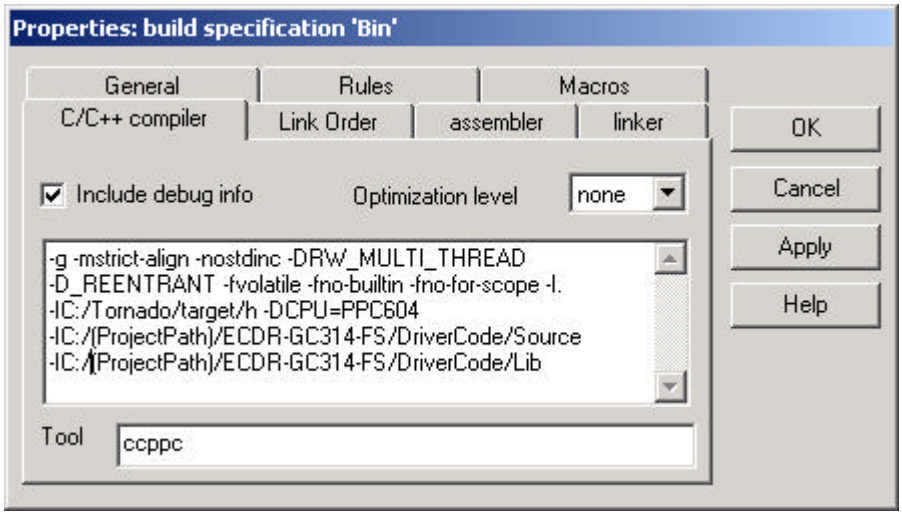


2. Next, click **C/C++ compiler** and type the following paths to specify your C/C++ compiler rules:

   -IC:(*ProjectPath*)/ECDR-GC314-FS/DriverCode/Source

   -IC:(*ProjectPath*)/ECDR-GC314-FS/DriverCode/Lib

   where **(*ProjectPath*)** is the application code root path.



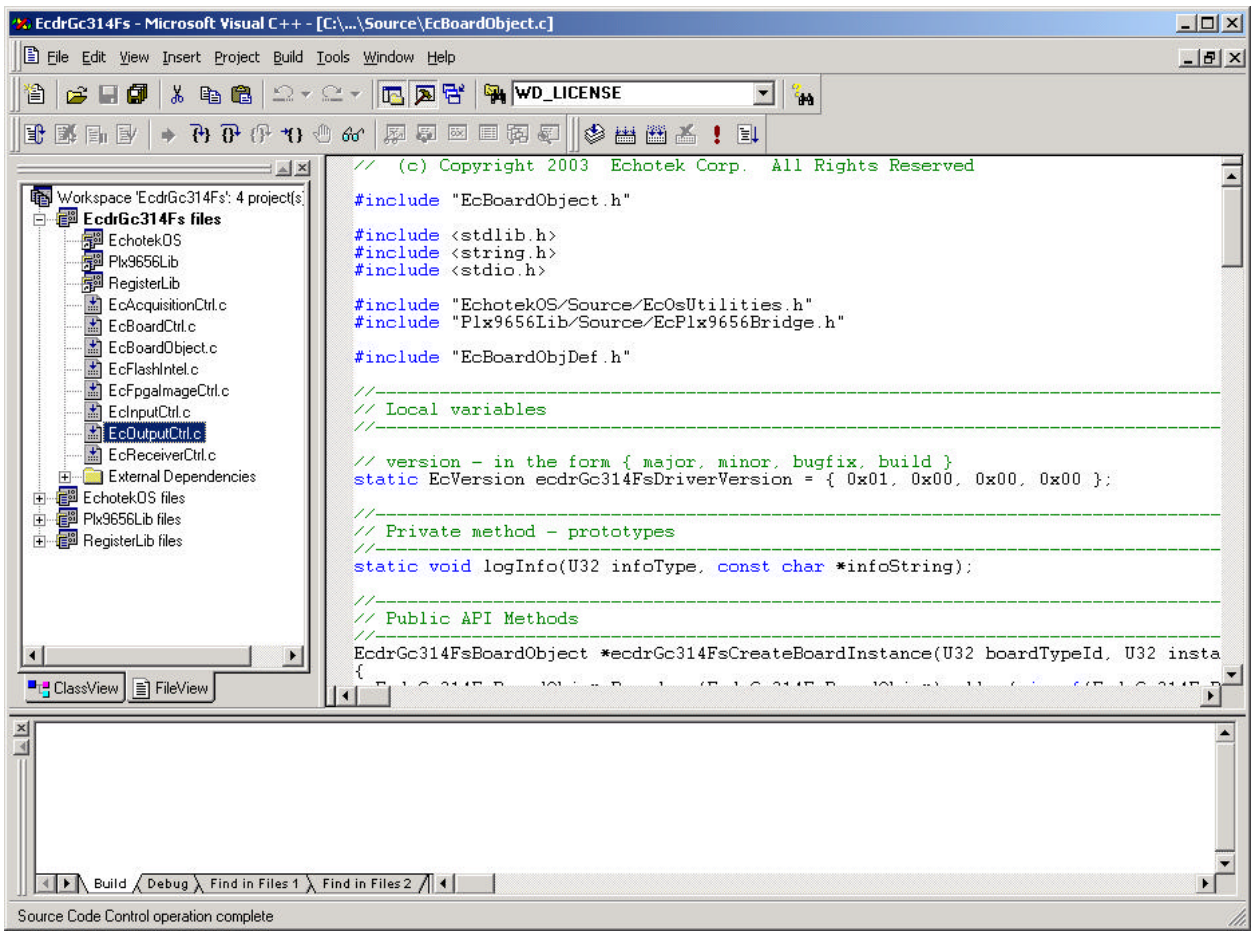
3. Finally, compile and link with your code.

The application code delivered with the ECDR-GC314-FS driver code can be used as an example of the linking procedure.

Building the Driver for Windows

The ECDR-GC314-FS driver was developed using the Microsoft Visual C++ 6.0 IDE (Integrated Development Environment). All required workspace and project files are included in the *Project/Windows* subdirectory. The ECDR-GC314-FS driver code is delivered as a static library object that can be linked with a user's application or system code. The source code and project files are delivered incase a user needs to modify the driver code for a specific application.

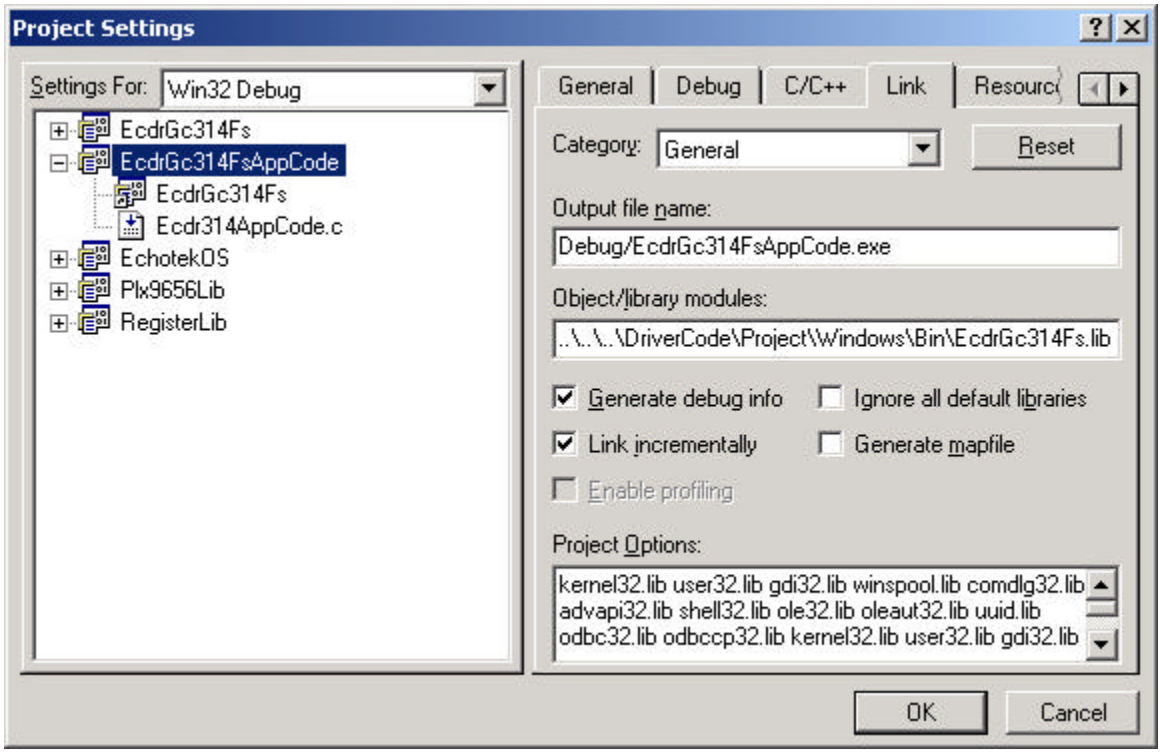
The following shows the Visual C++ IDE with the ECDR-GC314-FS driver workspace files.



The EcdrGc314Fs workspace is made up of several projects. The main project is the EcdrGc314Fs project and the others are Echotek code library projects. The output of each project is linked together to form the ECDR-GC314-FS driver static library object (**EcdrGc314Fs.lib**), which is found in the *Project/Windows/Bin* subdirectory.

The driver code is delivered already compiled and linked into the library object. The following steps can be used to link the ECDR-GC314-FS driver library object to a user application.

1. Under the **Project Setting** of the **User Application** project, add the ***ECDR-GC314-FS*\DriverCode\Project\Windows\Bin\EcdrGc314Fs.lib** file to the **Object/library modules**.

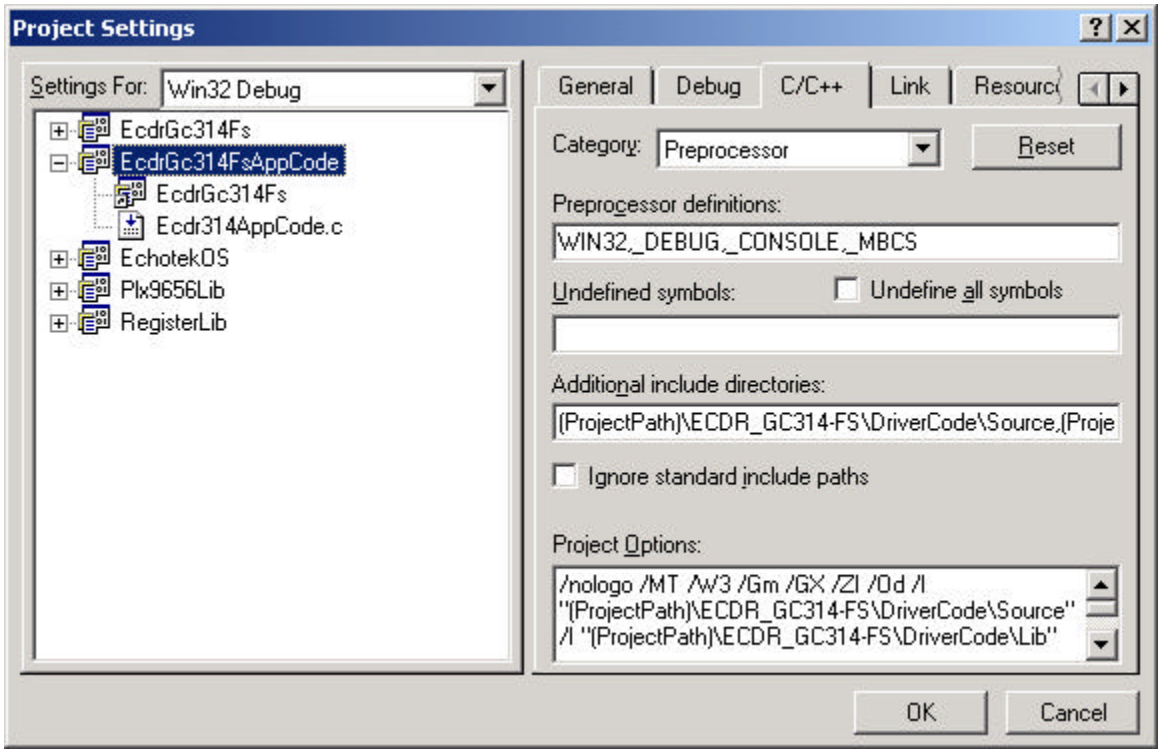


2. Next, click **C/C++** and type the following paths into the **Additional include directories** field to specify your C/C++ compiler rules:

   (*ProjectPath*)\ECDR-GC314-FS\DriverCode\Source

   (*ProjectPath*)\ECDR-GC314-FS\DriverCode\Lib

   where **(*ProjectPath*)** is the application code root path.



3. Finally, compile and link with your code.

The application code delivered with the ECDR-GC314-FS driver code can be used as an example of the linking procedure.

Building the Driver for Linux

TBD

Chapter 3 Installing the Driver

This chapter includes steps for installing the ECDR-GC314-FS driver.

Installing the ECDR-GC314-FS Driver for Windows

The Echotek ECDR-GC314-FS driver code uses the WinDriver product by Jungo within the Windows environment. The following procedure installs the WinDriver run-time components and the ECDR-GC314-FS driver components necessary for use with Windows.

The *Install* subdirectory contains the following files:

- **windrvr.sys**
- **windrvr.vxd**
- **wd_virtual.inf**
- **EcdrGc314Fs.inf**
- **wdreg_gui.exe**

Windows 98/Me and 2000/XP

NOTE:

This section refers to distribution of SYS files. Due to the limitations of Windows 98/Me, **WINDRVR.SYS** cannot be loaded dynamically on these operating systems, but instead requires the rebooting of your computer. In lieu of rebooting, you may use **WINDRVR.VXD** and follow the installation instructions for Windows 95 below.

Installing the driver on the target computer

Follow the instructions below in order to properly install your driver on the target computer:

- **Preliminary steps:**
  - To avoid rebooting, ensure that no PCI/USB devices are currently registered to work with WinDriver before attempting to install the driver - i.e. ensure that no INF files pointing to **windrvr.sys** are currently installed for any of the PCI/USB devices on the PC. (This may be relevant, for example, when upgrading a driver developed with an earlier version of WinDriver).

    To do this, uninstall all PCI/USB devices that are registered to work with WinDriver from the **Device Manager** (**Properties > Uninstall**). If this action is not performed, **WDREG_GUI** will fail when the **reload** or **remove** command is used (see instructions below). **WDREG_GUI** will then inform the user that s/he must first uninstall all devices currently registered to work with WinDriver or reboot the PC in order to successfully execute the command.

- To prevent Windows from automatically installing old INF files for PCI/USB devices that you wish to handle with WinDriver, it is also recommended that any backup INF files be deleted that Windows may have created for these PCI/USB devices.
  - On Windows 2000/XP, these files are stored in the *\%windir%\inf* directory and are named **oem*.inf**.
  - On Windows 98/Me, these files are stored in the *\Windows\inf\other* directory.

NOTE:

You can search for the device's vendor ID and device/product ID in the backup INF directory to locate the relevant files(s) for your device(s).

- **Installing WinDriver's kernel module:**

3. Copy **windrvr.sys** to the Windows installation directory on the target computer:
   - On Windows 2000 – *WINNT\system32\drivers*
   - On Windows 98/Me/XP – *Windows\system32\drivers*

TIP!

The command ***%windir%\system32*drivers** points to the Windows installation directory, regardless of the operating system. %windir% is equivalent to typing the Windows base directory (WINNT or Windows).

4. Copy **wd_virtual.inf** to a temporary directory on the target computer (e.g., *c:\tmp*). You can delete this file from the temporary directory after completing the installation process.

5. Use the utility **wdreg_gui** to install WinDriver's kernel module on the target computer. From the command line, type:

   \> **wdreg_gui -inf <full path to wd_virtual.inf> reload**

   For example, if **wd_virutal.inf** has been copied to the *c:\tmp* directory on the target computer, type the following:

   \> **wdreg_gui -inf c:\tmp\wd_virtual.inf reload**

NOTE:

You must type the **full path** to the INF file when using **wdreg_gui**.

NOTE:

wdreg_gui is an interactive utility. If the utility fails, it will display a message instructing the user on how to resolve the problem. In some cases, the user may be asked to reboot the computer.

CAUTION:

Ensure that you do not overwrite **windrvr.sys** with an older, existing version of the file in the Windows driver directory (*%windir%\system32\drivers*). You should compare the timestamp on the two versions of the files and not overwrite a newer version with an older version.

- **Installing the EcdrGc314Fs.inf** (updating Windows **Device Manager**):
  - On Windows 2000/XP, you can use the **wdreg_gui** utility to automatically load the INF file and update Windows **Device Manager**. To automatically install your INF file and update Windows **Device Manager**, run **wdreg_gui** with the **loadinf** option:

    \> **wdreg_gui -inf <full path to INF file> loadinf**

    For example, if the INF file **my_inf.inf** has been copied to the *c:\tmp* directory on the target computer, run:

    \> **wdreg_gui -inf c:\tmp\my_inf.inf loadinf**

    On Windows 98/Me, skip this step and follow the manual INF installation instructions below.

  - On Windows 98/Me, install the INF file manually using Windows' **Add New Hardware Wizard** or **Upgrade Device Driver Wizard.**

NOTE:

To prevent Windows from automatically detecting and installing an INF file previously installed for the device, remove the backup INF file that Windows created from the *Windows\inf\other* directory before installing the new INF file that you created. (You can search for the device's vendor ID and device/product ID in the backup INF directory to locate the relevant files(s) for your device(s)). This is particularly relevant when upgrading from a previous WinDriver version.

Ensure that you do not overwrite **windrvr.sys** with an older, existing version of the file in the Windows driver directory (*%windir%\system32\drivers*). You should compare the timestamp on the two versions of the files and not overwrite a newer version with an older version.

Windows 95 and NT 4.0

Installing the driver on the target computer

Follow the instructions below in order to properly install your driver on the target computer:

- **Installing WinDriver's kernel module:**

1. Copy the **windrvr.sys** (Windows NT) or **windrvr.vxd** (Windows 95) file to the *Windows* installation directory on the target computer:
   - On Windows NT target computers – Copy **windrvr.sys** to *WINNT\system32\drivers*.
   - On Windows 95 target computers – Copy **windrvr.vxd** to *Windows\system\VMM32*.

2. Use the **wdreg_gui** utility to add **windrvr.sys** / **windrvr.vxd** to the list of device drivers that Windows loads upon booting:

    - Windows NT/95 – Use the following installation command:

        \> **wdreg_gui install**

    - Windows 98/Me (when installing **windrvr.vxd**) – Use the `-vxd` flag in the installation command:

        \> **wdreg_gui -vxd install**

    By default, **wdreg_gui** installs **windrvr.sys** on Windows NT/98/Me/2000/XP and **windrvr.vxd** on Windows 95.

Installing the ECDR-GC314-FS Driver for Linux

The Echotek ECDR-GC314-FS driver code uses the WinDriver product by Jungo within the Linux environment. The following procedure installs the WinDriver run-time components and the ECDR-GC314-FS driver components necessary for use with Linux.

TBD

Installing the ECDR-GC314-FS Driver for VxWorks

There is no installation procedure for the ECDR_GC314-FS driver code for the VxWorks operating system environment. Simply follow the Building and Linking instructions from *Chapter 2* to use the ECDR-GC314-FS driver code and include the file **EcdrGc314FsApi.h** in the user application code.

Chapter 4
Application Programming Interface (API)

This chapter includes the API calls supported by the ECDR-GC314-FS driver.

Overview of the ECDR-GC314-FS API

The ECDR-GC314-FS API provides all function calls necessary to configure, control, and utilize all functionality of ECDR-GC314-FS boards. The function calls are divided into several groups related to various aspects of board configuration or operation. This chapter lists all function calls, along with descriptions and valid parameters.

The EcdrGc314Fs driver's API is defined in the **EcdrGc314FsApi.h** header file. Simply include this file in the user application code when calling the driver code function calls. Each ECDR-GC314-FS board in a system must be 'created' and initialized using the `ecdrGc314FsCreateBoardInstance` function call, which returns a board pointer that must then be passed into each function to reference the specific board.

Most functions return an `EcStatus` value that indicates the success or failure of the function call. The valid values for EcStatus are :

- `EC_OK` – Indicates successful operation of the function
- `EC_ERROR` – Indicates a failure to complete the desired function
- `EC_INVALID_PARAMETER` – Indicates that one or more parameters were not valid or out of range

The ECDR-GC314-FS software provides sample application code that uses the API to perform most functions. Refer to the application code for examples of the program flow using the ECDR-GC314-FS driver API.

The ECDR-GC314-FS boards are made up of multiple Receiver Blocks that control all aspects of data collection. The complete details of the board architecture are described in the *ECDR-GC314-FS Hardware User Manual*. Most ECDR-GC314-FS driver API function calls take the parameters and channel of the specific Receiver Block the function is to act upon. A unique feature of the ECDR-GC314-FS driver API is the ability to OR together the valid Receiver Block values and channel to enable functions to act on more that one identical Receiver Block and/or channel at the same time. Thus, a user can configure multiple Receiver Blocks and/or channels with the same configuration using one set of function calls.

Board Initialization and Configuration Function Calls

The *Board Initialization and Configuration* group of function calls relate to board initialization and configuration. These function calls create the existence of the board, initialize the driver code to control a specific instance of an ECDR-GC314-FS board and initialize and configure the hardware of the specified ECDR-GC314-FS board.

ecdrGc314FsCreateBoardInstance (Create)

This function call constructs an instance of a board object containing all information necessary to use the board. The required memory for the board object is allocated during the `ecdrGc314FsCreateBoardInstance` function call and the freed during the `Delete` function call. The `ecdrGc314FsCreateBoardInstance` function call takes two parameters:

- The specific type ID that uniquely identifies the board
- The instance of the specified board that you are going to control (i.e. which board)

This function call will also initialize the board with the appropriate configuration.

Synopsis

```
#include "EcdrGc314FsApi.h"
EcdrGc314FsBoardObject *ecdrGc314FsCreateBoardInstance
                        (U32 boardTypeId, U32 instance);
```

Return Values

Returns a pointer to *`EcdrGc314FsBoardObject`*, or `NULL` if an error occurs.

Parameters

Parameters

Parameters	Description

`boardTypeId`	ID of the specific type of EcdrGc314Fs board used.
	Valid choices for *`boardTypeId`* are:
	`ECDR_GC314_PMC_FS`
	`ECDR_GC314_PCI_FS`
	`ECDR_GC314_RPPOT`

`instance`	Specified board instance to control (where zero (0) is the first board instance).

ecdrGc314FsDeleteBoardInstance (Delete)

All created instances of a board must be deleted when it is no longer needed, or at the end of program execution.

Synopsis

```
#include "EcdrGc314FsApi.h"
void ecdrGc314FsDeleteBoardInstance(EcdrGc314FsBoardObject
                                    *pBoardObject);
```

Return Values

void

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.

ecdrGc314FsGetDriverVersion

This function call returns the version of the EcdrGc314Fs driver. There are two characteristics of this function call:

- The return is a U32 that represents the version in the form *0xXXYYZZBB*, where *XX* is major, *YY* is minor, *ZZ* is bugfix and BB is the build.
- The other characteristic returns the version by copying the string *XX.YY.ZZ Build BB* into the ***sPtr*** char pointer passed in by the caller. If the caller passes in a NULL pointer, only the U32 is returned. The passed-in char buffer *MUST* be at least 22 characters in length.

Synopsis

```
#include "EcdrGc314FsApi.h"
U32 ecdrGc314FsGetDriverVersion(char *sPtr, int maxLen);
```

Return Values

U32 representing the version, in the form *0xXXYYZZBB* (see description above).

Parameters

Parameters	Description
sPtr	Pointer to the character buffer into which this function call will copy the version string *XX.YY.ZZ Build BB*. If a NULL pointer is passed in, only the U32 is returned. The passed in char buffer *MUST* be at least 22 characters in length.
maxLen	The maximum number of characters that can be written into the char buffer pointed to by ***sPtr***.

ecdrGc314FsRegisterLogInfoFunction

This function call registers a function pointer to a logging facility that enables the EcdrGc314Fs driver code to display information (log errors, stats, etc.) to the user. The registered function provides a user-defined mechanism to display the information of the driver code. The driver code calls this registered function passing in the information to display. The registered function must perform all desired formatting of the passed in string (ie. *`*infostring`* contains no `\n` characters).

The prototype for the `logInfoFuncPtr` is `void logInfoFunction(U32 infoType, const char *infoString)`, where `infoType` is used to indicate the type of information:

- `EC_CRITICAL_ERROR`
- `EC_MAJOR_ERROR`
- `EC_MINOR_ERROR`
- `EC_ALERT_INFO`
- `EC_GENERAL_INFO`
- `EC_DEBUG_INFO`

Synopsis

```
#include “EcdrGc314FsApi.h”

void ecdrGc314FsRegisterLogInfoFunction(VOID_FUNCPTR
                                        pLogInfoFunc);
```

Return Values

void

Parameters

Parameters	Description

pLogInfoFunc	Pointer to the function that implements a logging function. The prototype for the *pLogInfoFunc* is: `void logInfoFunction(U32 infoType, const char *infoString),` where *infoType* is used to indicate the type of information. Valid choices for *infoType* are:
	`EC_CRITICAL_ERROR`
	`EC_MAJOR_ERROR`
	`EC_MINOR_ERROR`
	`EC_ALERT_INFO`
	`EC_GENERAL_INFO`
	`EC_DEBUG_INFO`

ecdrGc314FsUnregisterLogInfoFunction

This function call disables the EcdrGc314Fs driver code's logging facility.

Synopsis

```
#include "EcdrGc314FsApi.h"

void ecdrGc314FsUnregisterLogInfoFunction(void);
```

Return Values

void

Parameters

void

ecdrGc314FsGetFpgaVersion

This function call provides the version of the Echotek ECDR-GC314-XXX-FS Altera Stratix design programmed into the FPGA. There are two characteristics of this function call:

- The return is a U32 that represents the version in the form *0xXXYYZZBB*, where *XX* is major, *YY* is minor, *ZZ* is bugfix and *BB* is the build.
- The other characteristic returns the version by copying the string *XX.YY.ZZ Build BB* into the ***sPtr*** char pointer passed in by the caller. If the caller passes in a NULL pointer, only the U32 is returned. The passed in char buffer *MUST* be at least 22 characters in length.

Synopsis

```
#include "EcdrGc314FsApi.h"

U32 ecdrGc314FsGetFpgaVersion(EcdrGc314FsBoardObject
                              *pBoardObject, char *sPtr,
                              int maxLen);
```

Return Values

U32 representing the version, in the form *0xXXYYZZBB* (see description above).

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
sPtr	Pointer to the character buffer into which this function call will copy the version string *XX.YY.ZZ Build BB*. If a NULL pointer is passed in, only the U32 is returned. The passed-in char buffer *MUST* be at least 22 characters in length.
maxLen	The maximum number of characters that can be written into the char buffer pointed to by ***sPtr***.

ecdrGc314FsResetBoard

This function call enables a user to reset a specific EcdrGc314Fs board.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsResetBoard(EcdrGc314FsBoardObject
                               *pBoardObject);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	The EcdrGc314FsBoardObject pointer returned by the Create function call for the specific board instance.

ecdrGc314FsSelectClockSource

This function call selects either the on-board oscillator or the external clock source (the Front panel connector) as the source for the data clocking of the board. No *clockSource* is initially selected. This function call must be made before any configuration can be performed on the board.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSelectClockSource
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 clockSource);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
clockSource	Valid choices are: EC_INTERNAL_CLOCK EC_EXTERNAL_CLOCK

ecdrGc314FsEnableExtTrigger

This function call enables the External Trigger (TRG) signal. This signal is used as both a trigger source and/or a sync source for the Acquisition Control logic and the Receiver Control logic. The initial state of the External Trigger signal is disabled.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsEnableExtTrigger(EcdrGc314FsBoardObject
                                     *pBoardObject);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.

ecdrGc314FsDisableExtTrigger

This function call disables the External Trigger (TRG) signal. This signal is used as both a trigger source and/or a sync source for the Acquisition Control logic and the Receiver Control logic. The initial state of the External Trigger signal is disabled.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsDisableExtTrigger
         (EcdrGc314FsBoardObject *pBoardObject);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.

ecdrGc314FsEnableLbTerm

This function call enables termination of the Local Bus cycle after the specified termination count. The termination count represents the number of PCI clock cycles that must pass, without a local bus response, before the Local Bus termination occurs. The initial state of the Local Bus Termination is disabled.

NOTE:

When assigning this value, ensure that enough time is allowed for a Local Bus transfer to complete.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsEnableLbTerm
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 terminationCount);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
terminationCount	Number of PCI clock cycles that must pass, without a Local Bus response, before the Local Bus termination occurs. This is a 32-bit count.

ecdrGc314FsDisableLbTerm

This function call disables the termination of the Local Bus cycle. The initial state of the Local Bus Termination is disabled.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsDisableLbTerm(EcdrGc314FsBoardObject
                                  *pBoardObject);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.

ecdrGc314FsRegisterLbTermCallback

This function call enables a user to register an event callback function that will be executed when the Local Bus cycle is terminated. This function call enables the event notification (interrupt). The prototype of the callback function must be `void function(void*)`.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsRegisterLbTermCallback
         (EcdrGc314FsBoardObject *pBoardObject,
          void *pFunction, void *pFuncArgument);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
pFunction	Pointer to the callback function that will be executed when a Local Bus cycle is terminated.
pFuncArgument	Pointer to any data that a user wants to pass to the callback function when the event occurs.

ecdrGc314FsUnregisterLbTermCallback

This function call enables a user to unregister an event callback function. The `Unregister` function call disables the event notification (interrupt).

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsUnregisterLbTermCallback
            (EcdrGc314FsBoardObject *pBoardObject);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

ecdrGc314FsLoadFpgaImage

This function call enables a user to load an FPGA image file into the onboard Flash memory. Refer to the *ECDR-GC314-XXX-FS Hardware Users Manual* for complete details about the FPGA images and their Flash partitioning. Once images are loaded into Flash memory, another function call enables a user to reconfigure the FPGA with these images.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsLoadFpgaImage(EcdrGc314FsBoardObject
                                  *pBoardObject, U32
                                  imageNumber, U32 *pBuffer,
                                  U32 numBytes);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.
imageNumber	Number of the Flash partition into which an image is to be loaded. Each partition may contain one FPGA image.
pBuffer	Pointer to the buffer that contains the FPGA image data.
numBytes	Number of bytes of FPGA image data to be loaded.

ecdrGc314FsReconfigureFpga

This function call enables a user to reconfigure the FPGA with the specified image file that has been previously loaded into the onboard Flash memory. Refer to the *ECDR-GC314-XXX-FS Hardware Users Manual* for complete details on the FPGA images and their Flash partitioning.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsReconfigureFpga(EcdrGc314FsBoardObject
                                    *pBoardObject, U32
                                    imageNumber);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
imageNumber	Number of the FPGA image in Flash memory to be programmed into the FPGA.

Input Control Function Calls

The *Input Control* group of function calls select and configure the source of data to the Receiver Blocks. There are four Receiver Blocks, consisting of the Echotek Internal Wideband Receiver (IWB) and three TI (Texas Instruments) GC4016 Receivers.

ecdrGc314FsSelectIwbInputSource

This function call selects the input channel (i.e. one of the front panel connectors) as the input to the IWB. An all 0s (zeros) input can also be connected as the input channel of the IWB.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSelectIwbInputSource
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 inputChannel);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
inputChannel	The Input Channel that is to be connected to the IWB. Valid choices are: EC_INPUT_CHANNEL_1 EC_INPUT_CHANNEL_2 EC_INPUT_CHANNEL_3 EC_INPUT_ALL_ZEROS

ecdrGc314FsSelectGc4016InputSource

This function call selects the input source to the TI GC4016 receivers. The available input sources are the three input channels of digitized A/D data, or the output of the IWB. When the input channels (CH1, CH2, CH3) are selected, each input channel is connected to the input channels of each GC4016 (A, B and C, respectively, with D grounded). When IWB data is selected, the 16-bits of I and Q data are connected to input A and B of each TI GC4016 receiver, with input C and D grounded.

IMPORTANT: The GC4016s must be configured to operate in the appropriate mode to match the input data connected to the GC4016 chips.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSelectGc4016InputSource
        (EcdrGc314FsBoardObject *pBoardObject,
         U32 inputSource);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
inputSource	The input source to be connected to the TI GC4016 receivers. Valid choices are: EC_SOURCE_INPUT_CHANNELS EC_SOURCE_IWB_IANDQ_DATA

ecdrGc314FsInputChanOverrange

This function call indicates whether the specified input channel's A/D input has exceeded the acceptable range (i.e. if the A/D has been saturated). The TRUE/FALSE over range status result is returned using the passed-in Boolean pointer.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsInputChanOverrange
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 inputChannel, Boolean *pOverrangeStatus);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
inputChannel	The input channel to check for A/D over range status. Valid choices are: EC_INPUT_CHANNEL_1 EC_INPUT_CHANNEL_2 EC_INPUT_CHANNEL_3
pOverrangeStatus	Pointer to the Boolean variable in which to hold the resulting TRUE/FALSE value.

Sync Control Function Calls

To support synchronized operations across more than one ECDR-GC314-XX-FS, two sync buses are implemented and are connected to all Receiver Blocks and their associated control logic.

There are three ways a sync signal can be generated onto these sync buses:

- By the onboard sync signal generation logic
- By the sync master of the multi-board sync bus
- By the external trigger (the external trigger is only capable of generating sync on Sync Bus 1).

Each control logic and Receiver Block capable of receiving the sync signal on the sync buses has an associated sync mask in the **Global Sync Mask** register. A sync mask is simply a sync enable that, when set, enables any sync occurring on the sync bus to be received by the associated logic.

Two sets of masks are available for each control logic:

- A one-time mask – The one-time mask enables the reception of one sync signal from the sync bus before it goes inactive. A user must set the one-time mask again if another sync signal is desired. For some operations, a periodic sync signal is expected from external source(s) or from onboard RTS syncs generation logic. In this case, it can be difficult to determine when to arm the sync mask to catch the periodic sync signal at the right time. Furthermore, the act of rearming the sync mask(s) after each sync can cause unnecessary bus traffic in a system.
- A continuous mask – The continuous sync mask is provided to most logic that might require a periodic sync signal. This sync mask, when set, stays set until it is cleared.

Two types of sync signals can be generated on the sync buses:

- Pulsed – A pulsed sync signal goes active for four clocks before being de-asserted.
- Latched – A latched sync will stay active until being de-asserted manually.

A special type of pulsed sync, the RTS sync, can also be generated. A user can specify an RTS time (a 32-bit value) so that a pulsed sync is generated onto the sync bus whenever this time matches the onboard free running RTS.

ecdrGc314FsSetGlobalSyncMask

This function call enables a user to set sync masks that globally affect the module.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetGlobalSyncMask
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 syncMaskSelect);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
syncMaskSelect	Selects the sync mask to enable. Valid choices are: EC_SYNC1_IWB_SIA_ONE_TIME EC_SYNC1_IWB_SIA_CONTINUOUS EC_SYNC1_IWB_SIB_ONE_TIME EC_SYNC1_IWB_SIB_CONTINUOUS EC_SYNC1_GC4016_SIA_ONE_TIME EC_SYNC1_GC4016_SIA_CONTINUOUS EC_SYNC1_GC4016_SIB_ONE_TIME EC_SYNC1_GC4016_SIB_CONTINUOUS EC_SYNC1_CLEAR_RTS_COUNTER EC_SYNC1_LATCH_RTS_COUNTER EC_SYNC2_IWB_SIA_ONE_TIME EC_SYNC2_IWB_SIA_CONTINUOUS EC_SYNC2_IWB_SIB_ONE_TIME EC_SYNC2_IWB_SIB_CONTINUOUS EC_SYNC2_GC4016_SIA_ONE_TIME EC_SYNC2_GC4016_SIA_CONTINUOUS EC_SYNC2_GC4016_SIB_ONE_TIME EC_SYNC2_GC4016_SIB_CONTINUOUS EC_SYNC2_CLEAR_RTS_COUNTER EC_SYNC2_LATCH_RTS_COUNTER

ecdrGc314FsClearGlobalSyncMask

This function call enables a user to clear sync masks that globally affect the module.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsClearGlobalSyncMask
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 syncMaskSelect);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
syncMaskSelect	Selects the sync mask to clear. Only the continuous sync mask can be cleared. This function has no effect if one-time sync mask is specified. Valid choices are: EC_SYNC1_IWB_SIA_CONTINUOUS EC_SYNC1_IWB_SIB_CONTINUOUS EC_SYNC1_GC4016_SIA_CONTINUOUS EC_SYNC1_GC4016_SIB_CONTINUOUS EC_SYNC2_IWB_SIA_CONTINUOUS EC_SYNC2_IWB_SIB_CONTINUOUS EC_SYNC2_GC4016_SIA_CONTINUOUS EC_SYNC2_GC4016_SIB_CONTINUOUS

ecdrGc314FsSetSyncMask

These function calls enable a user to set sync masks that affect a particular receiver channel.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetSyncMask
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel,
          U32 syncMaskSelect);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to configure. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D

Parameters	Description
syncMaskSelect	Selects the sync mask to enable. Valid choices are: EC_SYNC2_DATA_ACQ_CONTINUOUS EC_SYNC2_DATA_ACQ_ONE_TIME EC_SYNC1_DATA_ACQ_CONTINUOUS EC_SYNC1_DATA_ACQ_ONE_TIME EC_SYNC2_CLEAR_FIFO EC_SYNC1_CLEAR_FIFO EC_SYNC2_CLEAR_FRAME_COUNTER EC_SYNC1_CLEAR_FRAME_COUNTER EC_SYNC2_UPDATE_CHANNEL_ID EC_SYNC1_UPDATE_CHANNEL_ID

ecdrGc314FsClearSyncMask

These function calls enable a user to clear sync masks that affect a particular receiver channel.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsClearSyncMask
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel,
          U32 syncMaskSelect);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to configure. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D
syncMaskSelect	Selects the sync mask to clear. Only the continuous sync mask can be cleared. This function has no effect if one-time sync mask is specified. Valid choices are: EC_SYNC2_DATA_ACQ_CONTINUOUS EC_SYNC1_DATA_ACQ_CONTINUOUS

ecdrGc314FsEnableSync

This function call enables a user to set generation of sync signals on either/both sync buses. When a pulsed sync is selected, the sync logic generates a four-clocks-wide sync signal onto the associated sync bus. A latched sync causes the sync signal to be asserted until being cleared. The RTS sync enable does not cause a sync signal to be generated immediately. Instead, it allows the RTS syncing logic to generate sync signals whenever the current timestamp value matches the pre-programmed Sync1/Sync2 RTS time. This sync is periodic and will continue to be generated until it is disabled. The External Trigger enable only generates a sync to Sync Bus 1 whenever an external trigger has been detected.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsEnableSync
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 syncSelect);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
syncSelect	Selects the sync to generate. Valid choices are: EC_SYNC1_PULSE EC_SYNC1_LEVEL EC_SYNC1_RTS EC_SYNC1_EXT_TRIGGER EC_SYNC2_PULSE EC_SYNC2_LEVEL EC_SYNC2_RTS

ecdrGc314FsDisableSync

This function call enables a user to disable the generation of sync signals on either/both sync buses.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsDisableSync
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 syncSelect);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
syncSelect	Clears the selected sync signal. Valid choices are: EC_SYNC1_LEVEL EC_SYNC1_RTS EC_SYNC1_EXT_TRIGGER EC_SYNC2_LEVEL EC_SYNC2_RTS

ecdrGc314FsSetSyncBusMode

This function call enables a user to set up the sync bus mode of the board. In a single board configuration, the *syncBusMode* is always EC_SYNC_BUS_MASTER. In multi-board configuration, only one board can be configured as EC_SYNC_BUS_MASTER; other boards must be configured as EC_SYNC_BUS_SLAVE.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetSyncBusMode
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 syncBusMode);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
syncBusMode	Selects the sync bus mode. Valid choices are: EC_SYNC_BUS_MASTER EC_SYNC_BUS_SLAVE

ecdrGc314FsSetSync1RtsTime

The following function call enables the setting of the sync 1 RTS time to generate a sync signal onto the sync buses. The RTS sync enable must be enabled via ecdrGc314FsEnableSync() before the RTS syncing logic will generate syncs onto the sync buses.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetSync1RtsTime
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 rtsSyncTimeSelect);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
rtsSyncTimeSelect	The time when a sync signal is to be generated on Sync Bus 1.

ecdrGc314FsSetSync2RtsTime

The following function call enables the setting of the sync 2 RTS time to generate a sync signal onto the sync buses. The RTS Sync enable must be enabled via `ecdrGc314FsEnableSync()` before the RTS syncing logic will generate syncs onto the sync buses.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetSync2RtsTime
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 rtsSyncTimeSelect);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.
rtsSyncTimeSelect	The time when a sync signal is to be generated on Sync Bus 2.

ecdrGc314FsGetSync1RtsTime

This function call enables a user to retrieve the current sync 1 RTS time.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsGetSync1RtsTime
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 *pRtsSyncTime);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
pRtsSyncTime	Stores the current SYNC 1/SYNC 2 sync 1 RTS time.

ecdrGc314FsGetSync2RtsTime

This function call enables a user to retrieve the current sync 2 RTS time.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsGetSync2RtsTime
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 *pRtsSyncTime);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
**pRtsSyncTime*	Stores the current sync 2 RTS time.

Frame Control Function Calls

The *Frame Control* function calls provide capabilities to control data framing and latching/clearing timestamps on the ECDR-GC314-XXX-FS module.

ecdrGc314FsSetFramingMode

This function call selects the data framing header mode when data framing is enabled.

Two modes are available:

- EC_FRAME_RTS_AND_CHID – Frame header contains both the RTS and Channel ID
- EC_FRAME_RTS_ONLY – Frame header contains only the RTS

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetFramingMode
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 framingMode);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
framingMode	Selects the desired framing mode. Valid choices are: EC_FRAME_RTS_AND_CHID EC_FRAME_RTS_ONLY

ecdrGc314FsGetTimestamp

This function call retrieves the contents of either of two relative timestamp counters:

- EC_SYNCHRONIZED_RTS – This counter contains the free-running RTS value
- EC_IMMEDIATE_RTS – This counter contains the synchronized or "latched" timestamp value

The synchronized RTS updates only when the LATCHED_RTS_COUNTER sync mask (sync1 or sync2) is set and the corresponding sync has been generated.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsGetTimestamp
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 rtsSelect, U32 *pRtsValue);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
rtsSelect	Selects the timestamp counter to read from. Valid choices are: EC_SYNCHRONIZED_RTS EC_IMMEDIATE_RTS
pRtsValue	Stores the timestamp value read from the board.

ecdrGc314FsSetFrameSize

This function call sets the fame size for a particular channel.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetFrameSize
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel, U32 frameSize);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
PboardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
ReceiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
Channel	The channel of the Receiver Block to configure. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D
FrameSize	Desired frame size.

ecdrGc314FsGetFrameSize

This function call retrieves the frame size for a particular channel.

Synopsis

#include “EcdrGc314FsApi.h”

```
EcStatus ecdrGc314FsGetFrameSize
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel,
          U32 *pFrameSize);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to configure. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D
pFrameSize	Stores the retrieved frame size.

ecdrGc314FsEnableDataFraming

These function calls enable a user to set the data framing capability on a channel basis. The frame size of the channel being enabled must be set by calling ecdrGc314FsSetFrameSize() prior to enabling data framing. If the frame size is not set, this function call will return EC_ERROR.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsEnableDataFraming
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to configure. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D

ecdrGc314FsDisableDataFraming

This function call enables a user to disable the data framing capability on a channel basis.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsDisableDataFraming
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to configure. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D

ecdrGc314FsSetChId

This function call enables a user to set the channel ID for any particular channel.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetChId
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel, U32 chId);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to configure. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D
chId	The channel ID for the specified channel.

ecdrGc314FsGetChId

This function call enables a user to retrieve the channel ID for any particular channel.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsGetChId
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel, U32 *pChId);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to configure. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D
pChId	Stores the retrieved channel ID.

ecdrGc314FsGetFrameCount

These function calls allow users to retrieve the current frame count for any particular channel.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsGetFrameCount
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel,
          U32 *pFrameCount);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to configure. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D
pFrameCount	Stores the retrieved channel frame count.

Receiver Control Function Calls

The *Receiver Control* group of function calls give access to each of the four Receiver Blocks' configuration registers. The Receiver Blocks are made up of the IWB and the three TI GC4016 receivers.

Each Receiver Block has two synchronization signal inputs:

- Sync Input A (SIA)
- Sync Input B (SIB)

These signal inputs are used internally to synchronize various aspects of the receiver logic as set in the Receiver Block registers. The synchronization process completes when the sync signal transitions from an asserted state to a released state. Refer to the *Echotek Internal Wideband Receiver Module User Manual* and the *TI GC4016 User Manual* for complete details about the synchronization and synchronization signal inputs.

ecdrGc314FsWriteRcvrConfig (Write)

This function call writes to the configuration registers of the specified Receiver Block. The registers of the Receiver Blocks may be separated into two types of registers:

- Global registers – Registers that control overall operation of the receiver.
- Channel registers – Registers that control an individual channel and that are set per channel.

For global registers and for Receiver Blocks that only have one channel (the IWB Block), the channel argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsWriteRcvrConfig
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel,
          U32 registerOffset, U32 data);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

receiverBlock	The Receiver Block to configure. Valid choices are:
	`EC_IWB_RECEIVER`
	`EC_GC4016_RECEIVER_1`
	`EC_GC4016_RECEIVER_2`
	`EC_GC4016_RECEIVER_3`

channel	The channel of the Receiver Block to configure. Valid choices are:
	`EC_GC4016_CHANNEL_A`
	`EC_GC4016_CHANNEL_B`
	`EC_GC4016_CHANNEL_C`
	`EC_GC4016_CHANNEL_D`

Parameters	Description
registerOffset	The register offset of the configuration register to which writing will occur. Valid choices are dependent on the Receiver Block being configured. The two Register Offset tables define the register offsets available for the IWB and the TI receivers.
data	The value to write to the configuration register.

ecdrGc314FsReadRcvrConfig (Read)

This function call reads the configuration registers of the specified Receiver Block. The registers of the Receiver Blocks may be separated into two types of registers:

- Global registers – Registers that control overall operation of the receiver.
- Channel registers – Registers that control an individual channel and that are set per channel.

For global registers and for Receiver Blocks that only have one channel (the IWB Block), the channel argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsReadRcvrConfig
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel,
          U32 registerOffset, U32 *pData);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to query. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D

Parameters	Description
registerOffset	The register offset of the configuration register from which to read. Valid choices are dependent on the Receiver Block being configured. The two Register Offset tables define the register offsets available for the IWB and the TI receiver.
pData	Pointer to variable to return register value.

The IWB Block Registers Offset table defines the valid choices for the *registerOffset* value in the READ and WRITE receiver configuration function calls. The IWB Block Data Value Macros table defines helper Macros that ease the programming of the IWB registers, and are used for the data value in the WRITE configuration function call. Refer to the *Echotek Wideband Receiver Module User Manual* for complete details on the registers.

The GC4016 Receiver Block Register Offset table defines the register offsets of the TI GC4016 Receiver Block. Refer to the *TI GC4016 User Manual* for complete details on the registers.

Echotek Internal Wideband Receiver (IWB) Registers Offset
EC_IWB_GAINDEC(*filterSet*) Where *filterSet* is the filter set to configure (0-63).
EC_IWB_SEL
EC_IWB_FREQ
EC_IWB_PHASE
EC_IWB_FREQSYNC
EC_IWB_RECSYNC

Table 4-1: IWB Block Register Offsets

Echotek Internal Wideband Receiver (IWB) Data Value Macros
EC_IWB_GAINDEC_VAL(*fineGain, coarseGain, firDecimation*) Where *fineGain* is an 8 bit value, *coarseGain* is a 3 bit value and valid choices for *firDecimation* are: EC_IWB_FIR_NO_DECIMATION EC_IWB_FIR_DECIMATE_BY_2 EC_IWB_FIR_DECIMATE_BY_3 EC_IWB_FIR_DECIMATE_BY_4
EC_IWB_FREQSYNC_VAL(*freqSync, phaseSync*) Where the valid choices for *freqSync* and *phaseSync* are: EC_IWB_SYNC_DISABLE EC_IWB_SIA_ENABLE EC_IWB_SIB_ENABLE EC_IWB_SIA_SIB_ENABLE EC_IWB_SYNC_ALWAYS
EC_IWB_RECSYNC_VAL(*ncoSync, firSync, flushSync*) Where the valid choices for *ncoSync* , *firSync*, and *flushSync* are: EC_IWB_SYNC_DISABLE EC_IWB_SIA_ENABLE EC_IWB_SIB_ENABLE EC_IWB_SIA_SIB_ENABLE

Table 4-2: IWB Block Data Value Macros

TI Global Registers Offset	TI Channel Registers Offset
EC_GC4016_GLOBAL_RESET	EC_GC4016_PHASE
EC_GC4016_STATUS	EC_GC4016_FREQ_LO_WORD
EC_GC4016_PAGE	EC_GC4016_FREQ_HI_WORD
EC_GC4016_CHECKSUM	EC_GC4016_CH_RESET
EC_GC4016_GEN_SYNC	EC_GC4016_FREQ_SYNC
EC_GC4016_COUNT_SYNC	EC_GC4016_NCO_SYNC
EC_GC4016_COUNTER	EC_GC4016_ZPAD_MODE_CTL
EC_GC4016_N_CHAN_OUT	EC_GC4016_FLUSH_SYNC
EC_GC4016_N_MULT	EC_GC4016_DEC_RATIO
EC_GC4016_FILTER_SEL	EC_GC4016_CIC_SCALE
EC_GC4016_FINAL_SHIFT	EC_GC4016_SPLIT_IQ
EC_GC4016_CHAN_MAP	EC_GC4016_CFIR
EC_GC4016_ADD_TO	EC_GC4016_PFIR
EC_GC4016_RESAMP_CLK_DIV	EC_GC4016_INPUT_CTL
EC_GC4016_RATIO_MAP	EC_GC4016_PEAK_CTL
EC_GC4016_RATIO_0	EC_GC4016_PEAK_CNT
EC_GC4016_RATIO_1	EC_GC4016_FINE_GAIN
EC_GC4016_RATIO_2	
EC_GC4016_RATIO_3	
EC_GC4016_TRISTATE_CTL	
EC_GC4016_OUTPUT_FMT	
EC_GC4016_OUTPUT_MODE	
EC_GC4016_OUTPUT_FRAME_CTL	
EC_GC4016_OUTPUT_WORD_SIZE	
EC_GC4016_OUTPUT_CLK_CTL	
EC_GC4016_SERIAL_MUX_CTL	
EC_GC4016_OUTPUT_TAG_A	
EC_GC4016_OUTPUT_TAG_B	
EC_GC4016_OUTPUT_TAG_C	
EC_GC4016_OUTPUT_TAG_D	
EC_GC4016_REV_MASK	
EC_GC4016_MISC	

Table 4-3: TI Receiver Block Register Offsets

ecdrGc314FsWriteRcvrCoeff

This function call writes the filter coefficients for the specified Receiver Block. A Receiver Block is comprised of one or more filters that may have one or more sets of coefficients. Refer to the *Echotek Wideband Receiver Module User Manual* for complete details on IWB filter configuration and coefficient register sets. Refer to the *TI GC4016 User Manual* for complete details on GC4016 filter configuration and coefficient registers. For Receiver Blocks with only one channel (the IWB Block) or for filters that affect all channels, the ***channel*** argument is ignored. For filters that have multiple sets of coefficients (the IWB Block), the coefficient set is selected via the ***filterSet*** argument of the specific filter selection.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsWriteRcvrCoeff
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel, U32 filter,
          U32 numCoeff, U32 *pData);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to configure. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D

Parameters	Description
filter	The filter being configured. Valid choices are Receiver Block dependent: EC_IWB_FIR(*filterSet*, *firDecimation*) EC_GC4016_CFIR EC_GC4016_PFIR EC_GC4016_RESAMPLER where *filterSet* is coefficient set to configure (0-63), and valid choices for *firDecimation* are: EC_IWB_FIR_NO_DECIMATION EC_IWB_FIR_DECIMATE_BY_2 EC_IWB_FIR_DECIMATE_BY_3 EC_IWB_FIR_DECIMATE_BY_4
numCoeff	The number of coefficients to configure.
pData	Pointer to the data buffer containing *numCoeff* of values to configure into the specified coefficient set.

ecdrGc314FsReadRcvrCoeff

This function call reads the filter coefficients for the specified Receiver Block. A Receiver Block is comprised of one or more filters that may have one or more sets of coefficients. Refer to the *Echotek Wideband Receiver Module User Manual* for complete details on IWB filter configuration and coefficient register sets. Refer to the *TI GC4016 User Manual* for complete details on the GC4016 filter configuration and coefficient registers. For Receiver Blocks with only one channel (the IWB Block) or for filters that affect all channels, the ***channel*** argument is ignored. For filters that have multiple sets of coefficients (the IWB Block), the coefficient set is selected via the argument ***filterSet*** of the specific filter selection.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsReadRcvrCoeff
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel, U32 filter,
          U32 numCoeff, U32 *pData);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to query. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D

Parameters	Description

filter	The filter being configured. Valid choices are Receiver Block dependent:
	`EC_IWB_FIR(`*`filterSet`*`,`
	`  `*`firDecimation`*`)`
	`EC_GC4016_CFIR`
	`EC_GC4016_PFIR`
	`EC_GC4016_RESAMPLER`
	Where *`filterSet`* is coefficient set to configure (0-63), and valid choices for *`firDecimation`* are:
	`EC_IWB_FIR_NO_DECIMATION`
	`EC_IWB_FIR_DECIMATE_BY_2`
	`EC_IWB_FIR_DECIMATE_BY_3`
	`EC_IWB_FIR_DECIMATE_BY_4`

numCoeff	The number of coefficients to read.
pData	Pointer to the data buffer large enough for numCoeff of values to read into from the specified coefficient set.

Acquisition Control Function Calls

The *Acquisition Control* group of function calls control how and when data is acquired by the board. These function calls provide individual configuration of the acquisition modes of each of the Receiver Blocks (the IWB and the three TI GC4016 receivers).

ecdrGc314FsSelectAcquisitionMode

This function call selects the acquisition mode of each of the specified Receiver Blocks. Refer to the *ECDR-GC314-XXX-FS Hardware Users Manual* for complete details about the acquisition modes.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSelectAcquisitionMode
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel,
          U32 acquisitionMode);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D
acquisitionMode	The acquisition mode of the Receiver Block. Valid choices are: EC_ACQUISITION_DISABLED EC_ACQUISITION_GATE EC_ACQUISITION_BURST EC_ACQUISITION_CONTINUOUS

ecdrGc314FsSetBurstCount

This function call sets the burst count of each Receiver Blocks' acquisition control logic. The burst count is the number of data words (32-bit) captured when the Receiver Block is in burst acquisition mode. The maximum burst count is 16384 for the IWB Block and 4096 for the three GC4016 Receiver Blocks.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetBurstCount
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel,
          U32 burstWordCount);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D
burstWordCount	Number of words to capture in burst mode.

ecdrGc314FsSetDataSkipCount

This function call sets the Data Skip count of each Receiver Blocks' acquisition control logic. The Data Skip count is the number of data words (32-bit) skipped after the trigger is received but before data is captured. The Data Skip count is applicable in all acquisition modes. The maximum Data Skip count is 16384 for the IWB Block and 4096 for the three GC4016 Receiver Blocks.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetDataSkipCount
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel,
          U32 skipWordCount);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D
skipWordCount	Number of words to skip before capturing data.

ecdrGc314FsStopContinuousAcquisition

This function call stops data acquisition for an individual Receiver Block configured for continuous acquisition mode.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsStopFreerunAcquisition
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to configure. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_GC4016_CHANNEL_A EC_GC4016_CHANNEL_B EC_GC4016_CHANNEL_C EC_GC4016_CHANNEL_D

ecdrGc314FsRegisterAcquisitionDoneCallback

This function call enables a user to register an event callback function that will be executed when the data acquisition is completed. The `Register` function call enables the event notification (interrupt). The prototype of the callback function must be `void function(void*)`.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsRegisterAcquisitionDoneCallback
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel, void *pFunction,
          void *pFuncArgument);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

receiverBlock	The Receiver Block to configure. Valid choices are:
	`EC_IWB_RECEIVER`
	`EC_GC4016_RECEIVER_1`
	`EC_GC4016_RECEIVER_2`
	`EC_GC4016_RECEIVER_3`

channel	The channel of the Receiver Block to query. Valid choices are:
	`EC_GC4016_CHANNEL_A`
	`EC_GC4016_CHANNEL_B`
	`EC_GC4016_CHANNEL_C`
	`EC_GC4016_CHANNEL_D`

pFunction	Pointer to the callback function that will be executed when data acquisition is completed.
pFuncArgument	Pointer to any data that the a wants to pass to the callback function when the event occurs.

ecdrGc314FsUnregisterAcquisitionDoneCallback

This function call enables a user to unregister an event callback function. The `Unregister` function call disables the event notification (interrupt).

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsUnregisterAcquisitionDoneCallback
        (EcdrGc314FsBoardObject *pBoardObject,
         U32 receiverBlock, U32 channel);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

receiverBlock	The Receiver Block to configure. Valid choices are:
	`EC_IWB_RECEIVER`
	`EC_GC4016_RECEIVER_1`
	`EC_GC4016_RECEIVER_2`
	`EC_GC4016_RECEIVER_3`

channel	The channel of the Receiver Block to query. Valid choices are:
	`EC_GC4016_CHANNEL_A`
	`EC_GC4016_CHANNEL_B`
	`EC_GC4016_CHANNEL_C`
	`EC_GC4016_CHANNEL_D`

Output Control Function Calls

The *Output Control* group of function calls control the FIFOs that data is collected into and collected data gets to the end user. All EcdrGc314Fs boards can perform DMA transfers. The functions to configure the DMA transfers are included in this section. The *Output Control* functions also provide the function calls that enable a user to be notified of events (interrupts) related to data output.

The procedure to perform DMA transfers is to create a DMA transfer list, which consists of one or more DMA transfer requests. Each request is defined as a transfer of a number of data words from the FIFOs to a user-defined memory location. Once a DMA transfer list is created, additional DMA transfer request can then be added to the list (chaining the transfers together). Finally, the DMA transfers contained in the list are started. Once all the transfers in a list are complete, the user-defined event callback function is executed.

ecdrGc314FsEnableReceiverBypass

This function call enables the Receiver Block bypass. If the IWB bypass is enabled, raw A/D data is collected in the IWB Block FIFO.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsEnableIwbBypass
        (EcdrGc314FsBoardObject *pBoardObject);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.

ecdrGc314FsDisableReceiverBypass

This function call disables the Receiver Block bypass. If the IWB bypass is disabled, the output data of the IWB (I and Q data) is collected in the IWB Block FIFO.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsDisableIwbBypass
         (EcdrGc314FsBoardObject *pBoardObject);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.

ecdrGc314FsSetFifoMode

This function call sets the operation mode for the FIFOs that collect the data from the three GC4016 Receiver Blocks. This mode of operation *MUST* match the configuration of the specific Receiver Block (if Receiver Block `EC_GC4016_RECEIVER_1` is set for Narrowband Mode, then the FIFO mode for Receiver Block `EC_GC4016_RECEIVER_1` must also be set for Narrowband Mode). Refer to the *ECDR-GC314-XXX-FS Hardware Users Manual* for complete details about the FIFO modes.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetFifoMode
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 fifoMode);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

receiverBlock	The Receiver Block to configure. Valid choices are:
	`EC_GC4016_RECEIVER_1`
	`EC_GC4016_RECEIVER_2`
	`EC_GC4016_RECEIVER_3`

fifoMode	The FIFO mode of the Receiver Block. Valid choices are:
	`EC_FIFO_MODE_NARROWBAND`
	`EC_FIFO_MODE_WIDEBAND`
	`EC_FIFO_MODE_EXTRA_WIDEBAND`

ecdrGc314FsClearFifo

This function call clears the specified FIFO. Depending on the FIFO mode, the FIFOs may be combined to create a larger FIFO for a particular receiver channel. Clearing all data for a particular receiver channel may require one or more `ecdrGc314FsClearFifo` commands. Refer to the *ECDR-GC314-XXX-FS Hardware Users Manual* for complete details on FIFO modes. For Receiver Blocks that only have one channel (the IWB Block), the *`channel`* argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsClearFifo
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
`pBoardObject`	Pointer to `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

`receiverBlock`	The Receiver Block to configure. Valid choices are:
	`EC_IWB_RECEIVER`
	`EC_GC4016_RECEIVER_1`
	`EC_GC4016_RECEIVER_2`
	`EC_GC4016_RECEIVER_3`

`channel`	The channel of the Receiver Block to query. Valid choices are:
	`EC_ GC4016_CHANNEL_A`
	`EC_ GC4016_CHANNEL_B`
	`EC_ GC4016_CHANNEL_C`
	`EC_ GC4016_CHANNEL_D`

ecdrGc314FsSetFifoLevel

This function call sets the level indicator for each FIFO. The level specifies a threshold at which a user is interested in knowing when data in the FIFO is equal to or exceeds the amount. The maximum level for the IWB Block is 16383 and, for the three GC4016 Receiver Blocks, is 1023. For the GC4016 Receiver Blocks, the level may be a multiple of 1023 based on the FIFO mode of operation. Refer to the *ECDR-GC314-XXX-FS Hardware Users Manual* for complete details about the FIFO modes. For Receiver Blocks that only have one channel (the IWB Block), the *channel* argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsSetFifoLevel
        (EcdrGc314FsBoardObject *pBoardObject,
         U32 receiverBlock, U32 channel, U32 fifoLevel);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to query. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_ GC4016_CHANNEL_A EC_ GC4016_CHANNEL_B EC_ GC4016_CHANNEL_C EC_ GC4016_CHANNEL_D
fifoLevel	The value of the FIFO level to program into the FIFO.

ecdrGc314FsGetFifoStatus

This function call returns the status of the specified FIFO. The possible states are:

- EC_FIFO_EMPTY – The FIFO contains no data
- EC_FIFO_NOT_EMPTY – The FIFO contains at least one data word
- EC_FIFO_LEVEL – The number of data words is greater than or equal to the value set in the FIFO level
- EC_FIFO_FULL – The FIFO is full.

FIFO overflows are indicated in the events of the Ecdr-Gc314-Fs driver. The status is returned using the pointer `U32 *fifoStatus`. For Receiver Blocks that only have one channel (the IWB Block), the `channel` argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsGetFifoStatus
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel,
          U32 *pFifoStatus);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

receiverBlock	The Receiver Block to query. Valid choices are:
	`EC_IWB_RECEIVER`
	`EC_GC4016_RECEIVER_1`
	`EC_GC4016_RECEIVER_2`
	`EC_GC4016_RECEIVER_3`

Parameters	Description
channel	The channel of the Receiver Block to query. Valid choices are: EC_ GC4016_CHANNEL_A EC_ GC4016_CHANNEL_B EC_ GC4016_CHANNEL_C EC_ GC4016_CHANNEL_D
pFifoStatus	Pointer to a variable containing the FIFO status return value. The possible values of *pFifoStatus* are: EC_FIFO_EMPTY EC_FIFO_NOT_EMPTY EC_FIFO_LEVEL EC_FIFO_FULL

ecdrGc314FsRegisterFifoOverflowCallback

This function call enables a user to register an event callback function that will be executed if the specified FIFO is overflowed. The `Register` function call enables the event notification (interrupt). The prototype of the callback function must be `void function(void*)`. For Receiver Blocks that only have one channel (the IWB Block), the *`channel`* argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsRegisterFifoOverflowCallback
        (EcdrGc314FsBoardObject pBoardObject,
         U32 receiverBlock, U32 channel, void *pFunction,
         void *pFuncArgument);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
`pBoardObject`	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

`receiverBlock`	The Receiver Block to configure. Valid choices are:
	`EC_IWB_RECEIVER`
	`EC_GC4016_RECEIVER_1`
	`EC_GC4016_RECEIVER_2`
	`EC_GC4016_RECEIVER_3`

`channel`	The channel of the Receiver Block to configure. Valid choices are:
	`EC_ GC4016_CHANNEL_A`
	`EC_ GC4016_CHANNEL_B`
	`EC_ GC4016_CHANNEL_C`
	`EC_ GC4016_CHANNEL_D`

`pFunction`	Pointer to the callback function that will be executed if the specified FIFO is overflowed.
`pFuncArgument`	Pointer to any data that a user wants to pass to the callback function when the event occurs.

ecdrGc314FsUnregisterFifoOverflowCallback

This function call enables a user to unregister an event callback function. The `Unregister` function call disables the event notification (interrupt).

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsUnregisterFifoOverflowCallback
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

receiverBlock	The Receiver Block to configure. Valid choices are:
	`EC_IWB_RECEIVER`
	`EC_GC4016_RECEIVER_1`
	`EC_GC4016_RECEIVER_2`
	`EC_GC4016_RECEIVER_3`

channel	The channel of the Receiver Block to configure. Valid choices are:
	`EC_ GC4016_CHANNEL_A`
	`EC_ GC4016_CHANNEL_B`
	`EC_ GC4016_CHANNEL_C`
	`EC_ GC4016_CHANNEL_D`

ecdrGc314FsRegisterFifoLevelCallback

This function call enables a user to register an event callback function that will be executed when the number of data words in the specified FIFO is equal to or exceeds the level amount. The `Register` function call enables the event notification (interrupt). The prototype of the callback function must be `void function(void*)`. For Receiver Blocks that only have one channel (the IWB Block), the *`channel`* argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsRegisterFifoLevelCallback
        (EcdrGc314FsBoardObject *pBoardObject,
         U32 receiverBlock, U32 channel,
         void *pFunction, void *pFuncArgument);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
`pBoardObject`	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

`receiverBlock`	The Receiver Block to configure. Valid choices are:
	`EC_IWB_RECEIVER`
	`EC_GC4016_RECEIVER_1`
	`EC_GC4016_RECEIVER_2`
	`EC_GC4016_RECEIVER_3`

`channel`	The channel of the Receiver Block to configure. Valid choices are:
	`EC_ GC4016_CHANNEL_A`
	`EC_ GC4016_CHANNEL_B`
	`EC_ GC4016_CHANNEL_C`
	`EC_ GC4016_CHANNEL_D`

`pFunction`	Pointer to the callback function that will be executed if the specified FIFO's Level is reached.
`pFuncArgument`	Pointer to any data that a user wants pass to the callback function when the event occurs.

ecdrGc314FsUnregisterFifoLevelCallback

This function call enables a user to unregister an event callback function. The `Unregister` function call disables the event notification (interrupt).

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsUnregisterFifoLevelCallback
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

receiverBlock	The Receiver Block to configure. Valid choices are:
	`EC_IWB_RECEIVER`
	`EC_GC4016_RECEIVER_1`
	`EC_GC4016_RECEIVER_2`
	`EC_GC4016_RECEIVER_3`

channel	The channel of the Receiver Block to configure. Valid choices are:
	`EC_ GC4016_CHANNEL_A`
	`EC_ GC4016_CHANNEL_B`
	`EC_ GC4016_CHANNEL_C`
	`EC_ GC4016_CHANNEL_D`

ecdrGc314FsReadFifoData

This function call reads data from the specified FIFO and copies it into the buffer pointed to by *pBuffer*. The number of words to copy is specified by *wordCount*. For Receiver Blocks that only have one channel (the IWB Block), the *channel* argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsReadFifoData
         (EcdrGc314FsBoardObject *pBoardObject,
          U32 receiverBlock, U32 channel, U32 *pBuffer,
          U32 wordCount);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
receiverBlock	The Receiver Block to query. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_ GC4016_CHANNEL_A EC_ GC4016_CHANNEL_B EC_ GC4016_CHANNEL_C EC_ GC4016_CHANNEL_D
pData	Pointer to the data buffer to copy data into from the specified FIFO.
wordCount	Number of data words to copy from the specified FIFO.

ecdrGc314FsCreatePciDmaTransferList

This function call creates a DMA transfer list containing one or more DMA transfer requests. The DMA transfer list initializes with the parameters of the first request. Additional DMA transfers request can then be added to the list (chaining the transfer request together) using the AddPciDmaTransfer function call. For Receiver Blocks that only have one channel (the IWB Block), the *channel* argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsCreatePciDmaTransferList
         (EcdrGc314FsBoardObject *pBoardObject,
          DmaTransferList **pDma, U32 receiverBlock,
          U32 channel, U32 wordCount, U32 **pBuffer);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
pDma	Pointer to the *DmaTransferList* pointer variable that a user keeps to identify this specific DMA transfer list.
receiverBlock	The Receiver Block to query. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_ GC4016_CHANNEL_A EC_ GC4016_CHANNEL_B EC_ GC4016_CHANNEL_C EC_ GC4016_CHANNEL_D

Parameters	Description
wordCount	Number of data words to copy from the specified FIFO.
pBuffer	Pointer to a U32 pointer used to pass back the address of the buffer into which the data for this transfer request will be stored.

ecdrGc314FsAddPciDmaTransfer

This function call adds a DMA transfer request to the specified DMA transfer list. This function call will chain the transfer request to the other request in the list. For Receiver Blocks that only have one channel (the IWB Block), the *channel* argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsAddPciDmaTransfer
         (EcdrGc314FsBoardObject *pBoardObject,
          DmaTransferList *pDma, U32 receiverBlock,
          U32 channel, U32 wordCount, U32 **pBuffer);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
pDma	Pointer to the *DmaTransferList* to which this request is to be added (chained).
receiverBlock	The Receiver Block to query. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_ GC4016_CHANNEL_A EC_ GC4016_CHANNEL_B EC_ GC4016_CHANNEL_C EC_ GC4016_CHANNEL_D

Parameters	Description
wordCount	Number of data words to copy from the specified FIFO.
pBuffer	Pointer to a U32 pointer used to pass back the address of the buffer into which the data for this transfer request will be stored.

ecdrGc314FsStartPciDmaTransfer

This function call starts the DMA transfers specified in the passed-in DMA transfer list, and continues transferring data until the last request in the list is completed.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsStartPciDmaTransfer
         (EcdrGc314FsBoardObject *pBoardObject,
          DmaTransferList *pDma);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
pDma	Pointer to the *DmaTransferList* to which this request is to be added (chained).

ecdrGc314FsDeletePciDmaTransferList

This function call deletes the specified DMA transfer list and all list-associated memory. All buffers used for data transfers will be freed.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsDeletePciDmaTransferList
         (EcdrGc314FsBoardObject *pBoardObject,
          DmaTransferList *pDma);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
pDma	Pointer to the *DmaTransferList* to be deleted.

ecdrGc314FsRegisterPciDmaTransferDoneCallback

This function call enables a user to register an event callback function that will be executed when the PCI DMA transfer is completed. The `Register` function call enables the event notification (interrupt). The prototype of the callback function must be `void function(void*)`.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsRegisterPciDmaTransferDoneCallback
         (EcdrGc314FsBoardObject *pBoardObject,
          void *pFunction, void *pFuncArgument);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.
pFunction	Pointer to the callback function executed when a PCI DMA transfer is completed.
pFuncArgument	Pointer to any data that a user wants to pass to the callback function when the event occurs.

ecdrGc314FsUnregisterPciDmaTransferDoneCallback

This function call enables a user to unregister an event callback function. The `Unregister` function call disables the event notification (interrupt).

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsUnregisterPciDmaTransferDoneCallback
        (EcdrGc314FsBoardObject *pBoardObject);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

ecdrGc314FsCreateRwDmaTransferList

This function call creates a DMA transfer list containing one or more DMA transfer requests. The DMA transfer list initializes with the parameters of the first request. Additional DMA transfers request can then be added to the list (chaining the transfer request together) using the AddRwDmaTransfer function call. For Receiver Blocks that only have one channel (the IWB Block), the *channel* argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsCreateRwDmaTransferList
        (EcdrGc314FsBoardObject *pBoardObject,
         DmaTransferList **pDma, U32 receiverBlock,
         U32 channel, U32 wordCount, U32 destRoute,
         U32 destAddr);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
pDma	Pointer to the *DmaTransferList* pointer variable that a user keeps to identify this specific DMA transfer list.
receiverBlock	The Receiver Block to query. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_ GC4016_CHANNEL_A EC_ GC4016_CHANNEL_B EC_ GC4016_CHANNEL_C EC_ GC4016_CHANNEL_D

Parameters	Description
wordCount	Number of data words to copy from the specified FIFO.
destRoute	The Raceway route code of the destination memory.
destAddr	The Raceway address of the destination memory.

ecdrGc314FsAddRwDmaTransfer

This function call adds a DMA transfer request to the specified DMA transfer list. This function call will chain the transfer request to the other request in the list. For Receiver Blocks that only have one channel (the IWB Block), the *channel* argument is ignored.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsAddRwDmaTransfer
         (EcdrGc314FsBoardObject *pBoardObject,
          DmaTransferList *pDma, U32 receiverBlock,
          U32 channel, U32 wordCount, U32 destRoute,
          U32 destAddr);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
pDma	Pointer to the *DmaTransferList* to which this request is to be added (chained).
receiverBlock	The Receiver Block to query. Valid choices are: EC_IWB_RECEIVER EC_GC4016_RECEIVER_1 EC_GC4016_RECEIVER_2 EC_GC4016_RECEIVER_3
channel	The channel of the Receiver Block to query. Valid choices are: EC_ GC4016_CHANNEL_A EC_ GC4016_CHANNEL_B EC_ GC4016_CHANNEL_C EC_ GC4016_CHANNEL_D

Parameters	Description
wordCount	Number of data words to copy from the specified FIFO.
destRoute	The Raceway route code of the destination memory.
destAddr	The Raceway address of the destination memory.

ecdrGc314FsStartRwDmaTransfer

This function call starts the DMA transfers specified in the passed-in DMA transfer list, and continues transferring data until the last request in the list is completed.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsStartRwDmaTransfer
         (EcdrGc314FsBoardObject *pBoardObject,
          DmaTransferList *pDma);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
pDma	Pointer to the *DmaTransferList* to which this request is to be added (chained).

ecdrGc314FsDeleteRwDmaTransferList

This function call deletes the specified DMA transfer list and all list-associated memory. All buffers used for these data transfers will be freed.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsDeleteRwDmaTransferList
         (EcdrGc314FsBoardObject *pBoardObject,
          DmaTransferList *pDma);
```

Return Values

EC_OK or EC_ERROR

Parameters

Parameters	Description
pBoardObject	Pointer to the EcdrGc314FsBoardObject returned by the Create function call for the specific board instance.
pDma	Pointer to the *DmaTransferList* to be deleted.

ecdrGc314FsRegisterRwDmaTransferDoneCallback

This function call enables a user to register an event callback function that will be executed when the Raceway (RACE++) DMA transfer is completed. The `Register` function call enables the event notification (interrupt). The prototype of the callback function must be `void function(void*)`.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsRegisterRwDmaTransferDoneCallback
         (EcdrGc314FsBoardObject *pBoardObject,
          void *pFunction, void *pFuncArgument);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.
pFunction	Pointer to the callback function that will be executed when a Raceway DMA transfer is completed.
pFuncArgument	Pointer to any data that a user wants to pass to the callback function when the event occurs.

ecdrGc314FsUnregisterRwDmaTransferDoneCallback

This function call enables a user to unregister an event callback function. The `Unregister` function call disables the event notification (interrupt).

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsUnregisterRwDmaTransferDoneCallback
        (EcdrGc314FsBoardObject *pBoardObject);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.

ecdrGc314FsRegisterRwErrorCallback

This function call enables a user to register an event callback function that will be executed if a RACEway (RACE++) read error or timeout occurs. The `Register` function call enables the event notification (interrupt). The prototype of the callback function must be `void function(void*)`.

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsRegisterRwErrorCallback
         (EcdrGc314FsBoardObject *pBoardObject,
          void *pFunction, void *pFuncArgument);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the *EcdrGc314FsBoardObject* returned by the `Create` function call for the specific board instance.
pFunction	Pointer to the callback function that will be executed when a Raceway Error occurs.
pFuncArgument	Pointer to any data that a user wants to pass to the callback function when the event occurs.

ecdrGc314FsUnregisterRwErrorCallback

This function call enables a user to unregister an event callback function. The `Unregister` function call disables the event notification (interrupt).

Synopsis

```
#include "EcdrGc314FsApi.h"

EcStatus ecdrGc314FsUnregisterRwErrorCallback
        (EcdrGc314FsBoardObject *pBoardObject);
```

Return Values

`EC_OK` or `EC_ERROR`

Parameters

Parameters	Description
pBoardObject	Pointer to the `EcdrGc314FsBoardObject` returned by the `Create` function call for the specific board instance.