bc635PCI
Developer's Kit
8500-0076

User's Guide
(October, 1999)
Rev E

bc635PCI
DEVELOPER'S KIT

TABLE OF CONTENTS

SECTION **PAGE**

CHAPTER ONE
INTRODUCTION

CHAPTER TWO
INSTALLATION

CHAPTER THREE
LIBRARY DEFINITIONS

This Page Intentionally Left Blank.

CHAPTER ONE
INTRODUCTION

1.0 GENERAL

The bc635PCI Developer's Kit is designed to provide a suite of tools useful for application development that access the bc635PCI Time and Frequency Processor features. This provides an interface between the bc635PCI and applications developed for Windows 95™, and Windows NT™ environments. In addition to the interface library, an example program is provided, complete with source code, in order to provide a better understanding of the kit features and benefits.

1.1 FEATURES

The salient features of the Developer's Kit include:

- Interface library with access to all bc635PCI features.
- Example programs, with source, utilizing the interface library.
- User's Guide providing a library definition.

1.2 OVERVIEW

The Developer's Kit was designed to provide an interface to the bc635PCI Time and Frequency Processor in Windows 95™ and Windows NT™ 32 bit environments. The example programs were developed under Microsoft Visual C++ v6.0. The example programs provide sample code which exercise the interface library as well as examples of converting many of the ASCII format data objects passed to and from the device into a binary format suitable for operation and conversion. The example programs were developed using discrete functions for each operation that allows the developer to clip any useful code and use it in their own applications. A resource file is included with interface dialogs to allow the operator of a program to set any configurable parameters for operating the bc635PCI hardware. Applications programs developed using the 32 bit interface library are binary compatible with both Windows 95™ and Windows NT™. This is made possible by the use of the Blue Waters Systems' WinRT package as a hardware abstraction layer.

This Page Intentionally Left Blank.

CHAPTER TWO
INSTALLATION

2.0 HARDWARE AND DRIVER INSTALLATION

Install the bc635PCI hardware and the appropriate driver before proceeding to the Software Developer's Kit Installation. Please refer to the "bc635PCI Time & Frequency Processor User's Guide" for Hardware and Driver Installation.

2.1 SOFTWARE DEVELOPER'S KIT INSTALLATION

Installation of the Developer's Kit is handled by the installer program.

- Use the setup.exe program on the Developer's Kit to install the kit.
- Use the compiled example program "bc635cpp.exe" to test the system.

2.2 CONFIGURATION

Directory structures are created in the specified location. These structures contain all required files to develop Windows 95™ and Windows NT™ 32 bit user applications.

2.2.1 32 BIT APPLICATION

Top Level Directory

UNINST ISU		29,014	03-25-98	2:53p	Uninst.isu
DOCUMENTATION		<DIR>	03-25-98	2:53p	Documentation
EXAMPLE PROGRAMS		<DIR>	03-25-98	2:53p	Example Programs
UTILITITY PROGRAMS		<DIR>	03-25-98	2:53p	Utility Programs

Documentation

TOC	DOC	22,016	03-25-98	11:02a	Toc.doc
CHAPTER1	DOC	37,888	03-25-98	2:52p	Chapter1.doc
CHAPTER2	DOC	53,248	03-25-98	2:52p	Chapter2.doc
CHAPTER3	DOC	64,000	02-23-98	5:22p	Chapter3.doc
FC	DOC	19,456	03-25-98	10:53a	Fc.doc

Example Programs

HARDWA~1	<DIR>		03-25-98	2:53p	Hardware Libraries
TRAY PR~1	<DIR>		03-25-98	2:53p	Tray Program
DEMO PR~1	<DIR>		03-25-98	2:53p	Demo Program

Example Programs\Demo Program\Mfc

RES	<DIR>		03-25-98	2:53p	Resource Files
BC635CPP	DSP	5,834	06-02-98	11:23a	Bc635cpp.dsp
BC_INT	H	1,044	05-26-98	11:52a	Bc_int.h
BC_INT	LIB	4,454	06-02-98	11:16a	Bc_int.lib
BC635CPP	CPP	4,099	05-18-98	5:55p	Bc635cpp.cpp
BC_DRVR	H	253	02-13-98	10:52a	Bc_drvr.h
BC635CPP	DSW	539	05-18-98	6:46p	Bc635cpp.dsw
BC635CPP	H	1,388	05-18-98	5:55p	Bc635cpp.h
BC635CPP	RC	50,224	05-27-98	10:02a	Bc635cpp.rc
BC635C~1	CPP	1,782	05-18-98	5:55p	Bc635cppDoc.cpp
BC635C~1	H	1,505	05-18-98	5:55p	Bc635cppDoc.h
BC635C~2	CPP	33,718	07-17-98	6:46a	Bc635cppView.cpp
BC635C~2	H	4,366	05-27-98	9:57a	Bc635cppView.h
BC637PCI	H	10,623	05-27-98	10:27a	Bc637pci.h
BC637PCI	LIB	35,188	06-02-98	11:23a	Bc637pci.lib
DLGFILE	CPP	2,753	05-19-98	2:25p	DlgFile.cpp
DLGSPE~1	H	3,773	05-26-98	12:05p	DlgSpecial.h
DLGFILE	H	1,221	05-19-98	12:19p	DlgFile.h
DLGHELP	CPP	1,720	05-27-98	9:59a	DlgHelp.cpp
DLGHELP	H	1,265	05-27-98	9:46a	DlgHelp.h
DLGOSC~1	CPP	8,287	05-26-98	7:15p	DlgOscillator.cpp
DLGOSC~1	H	3,718	05-26-98	3:25p	Dlgoscillator.h
DLGPCI	CPP	13,918	05-26-98	7:10p	DlgPci.cpp
DLGPCI	H	4,771	05-26-98	6:59p	DlgPci.h
DLGSIG~1	CPP	14,286	05-26-98	9:45p	DlgSignals.cpp
DLGSIG~1	H	5,618	05-23-98	5:45p	DlgSignals.h
DLGSPE~1	CPP	9,061	06-01-98	2:23p	DlgSpecial.cpp
STDAFX	H	928	05-18-98	5:55p	StdAfx.h
DLGTIME	H	6,390	05-27-98	10:19a	DlgTime.h
DLGTIM~1	CPP	8,593	05-27-98	10:16a	DlgTimecode.cpp
DLGTIM~1	H	3,714	05-27-98	10:19a	DlgTimecode.h

MAINFRM	CPP	2,908	07-15-98	5:12p	MainFrm.cpp
MAINFRM	H	1,600	05-26-98	9:47a	MainFrm.h
README	TXT	3,810	05-18-98	5:55p	ReadMe.txt
RESOURCE	H	10,765	05-27-98	10:02a	Resource.h
STDAFX	CPP	206	05-18-98	5:55p	StdAfx.cpp
DLGTIME	CPP	15,999	05-27-98	10:15a	DlgTime.cpp

Example Programs\Datum BC635 DemoConfig Program\Mfc\Res

BC635C~1	ICO	1,078	05-18-98	5:55p	Bc635cppDoc.ico
BC635CPP	RC2	400	05-18-98	5:55p	Bc635cpp.rc2
BC635C~2	ICO	1,078	05-18-98	5:55p	Bc635cpp1.ico
BC635CPP	ICO	766	04-07-98	11:06a	Bc635cpp.ico

Example Programs\Tray Program\Mfc

RES	<DIR>		03-25-98	2:53p	Resource Files
DAT_BRD	C	705	02-13-98	12:37p	Dat_brd.c
DAT_BRD	H	74	02-13-98	12:37p	Dat_brd.h
DAT_CFG	H	153	02-13-98	12:37p	Dat_cfg.h
DAT_CLK	H	819	06-01-98	11:21a	Dat_clk.h
DAT_GPS	C	4,976	02-13-98	12:37p	Dat_gps.c
DAT_GPS	H	769	02-13-98	12:37p	Dat_gps.h
DAT_NET	C	3,873	05-30-98	2:48p	Dat_net.c
DAT_NET	H	313	04-08-98	10:13a	Dat_net.h
DAT_REG	C	10,287	02-13-98	12:37p	Dat_reg.c
DAT_REG	H	1,304	02-13-98	12:37p	Dat_reg.h
DAT_RES	H	2,060	06-01-98	11:08a	dat_res.h
DAT_TYM	C	5,735	07-17-98	1:00a	Dat_tym.c
DAT_TYM	H	78	05-30-98	12:28p	Dat_tym.h
NTPC	H	386	11-14-97	8:47a	Ntpc.h
BC637PCI	H	9,974	05-12-98	12:04p	Bc637pci.h
TRAYTI~1	H	2,014	05-31-98	7:05p	TrayTimeCppDlg.h
SHEETS~1	CPP	6,595	07-06-98	10:53a	sheetsetup.cpp
SHEETS~1	H	4,321	06-02-98	10:32a	sheetsetup.h
SOMEFU~1	C	3,490	07-06-98	5:52p	SomeFunctions.c
STDAFX	CPP	209	05-27-98	12:47p	StdAfx.cpp
STDAFX	H	928	05-27-98	12:47p	StdAfx.h
TRAYTI~1	CPP	2,898	05-30-98	10:14p	TrayTimeCpp.cpp
TRAYTI~1	DSP	5,477	07-17-98	9:07a	TrayTimeCpp.dsp
TRAYTI~1	DSW	545	05-27-98	12:47p	TrayTimeCpp.dsw
TRAYTI~2	H	1,387	05-27-98	12:47p	TrayTimeCpp.h
TRAYTI~1	RC	9,100	07-06-98	5:53p	TrayTimeCpp.rc
TRAYTI~2	CPP	9,580	07-06-98	10:53a	TrayTimeCppDlg.cpp
RESOURCE	H	1,768	07-06-98	5:51p	resource.h

Example Programs\Datum Tray Time Utility Program\Mfc\Res

BC637PCI	OBJ	78,953	03-25-98	1:21p	bc637pci.obj
BC_INT	OBJ	48,971	02-09-98	12:11p	bc_int.obj
TRAYTIMECPP	ICO	1,000	07-25-98	12:11p	Traytimecpp.ico
TRAYTIMECPP	RC2	1,000	07-25-98	12:11p	Traytimecpp.rc2

Example Programs\Hardware Libraries

MICROS~1	<DIR>		03-25-98	2:53p	Microsoft

Example Programs\Hardware Libraries\Microsoft

BC637PCI	LIB	35,000	07-25-98	1:51p	Bc637pci.lib
BC637PCI	DLL	58,000	07-25-98	1:51p	Bc637pci.dll
BC_INT	LIB	5,000	07-25-98	1:51p	bc_int.lib
BC_INT	DLL	27,000	07-25-98	1:51p	bc_int.dll

Utility Programs

BC635CPP	EXE	331,778	02-09-98	12:11p	Bc635cpp.exe
TRAYTIMECPP	EXE	304,150	03-25-98	1:21p	Traytimecpp.exe
BC635CPP	HLP	384,778	02-09-98	12:11p	Help File
BC635CPP	CNT	4,124	02-09-98	12:11p	Help File

2.3 TEST INSTALLATION

Use the MFC precompiled example program version supplied in the Developer's Kit to test the installation.

If a device open error is received, the hardware interface was not installed or configured properly. Verify that the correct driver was installed according to the guidelines in the "bc635PCI Time & Frequency Processor User's Guide".

2.4 PROJECT CREATION

2.4.1 Microsoft Visual C++

You can easily rebuild Bc635cpp.exe and TrayTimeCpp.exe by opening the corresponding project file with Microsoft Visual C++ 5/6. If you want to use Bc637pci.dll and Bc_int.dll in your own MFC project, follow the instructions below:

1) Insert Bc635.lib and Bc_int.lib into your project.

2) If building a new project similar to Bc635cpp, you don't need to change the default settings of the project.

3) If building a new project similar to TrayTimeCpp, you may need to change the project settings:

   a) For both debug version and release version, go to "C/C++" tab, select "Precompiled Headers" category and then check "Not using precompiled headers" button. Next, go to the Link tab, select "General category" and add "ws2_32.lib" to "Object/Library Module" edit box.

   b) For release version, Link tab, select "Customize" category and then check "Force File Output" box.

This Page Intentionally Left Blank.

CHAPTER THREE
LIBRARY DEFINITIONS

3.0 GENERAL

The interface library provides functions for each of the programming packets supported by the bc635PCI Time and Frequency Processor. In addition, functions are provided to both read and write individual registers and dual port RAM locations on the card. To understand the usage and effects of each of these functions, please refer to the User's Guides provided with the hardware.

3.1 FUNCTIONS

bcStartPCI	
Prototype	int bcStartPCI (int devno);
Packet	N/A
Input Parameter	Device Number (0-3)
Returns	RC_OK On Success RC_ERROR On Failure
Description: This opens the underlying hardware layer. Use 0 if only one card in the system.	

bcStopPCI	
Prototype	int bcStopPCI (void);
Packet	N/A
Input Parameter	None
Returns	RC_OK On Success RC_ERROR On Failure
Description: Closes the underlying hardware layer.	

bcGetReg	
Prototype	int bcGetReg (UINT offset, ULONG *data);
Packet	N/A
Input Parameter	Offset = 0 based offset of requested register.. Data = pointer to unsigned long to return value requested.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Returns the contents of the requested register.	

Note: This command operates on the LCA registers (status/config registers).

bcSetReg	
Prototype	int bcSetReg (UINT offset, ULONG *data)
Packet	N/A
Input Parameter	Offset = 0 based offset of requested register. Data = pointer to unsigned long value to be set.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Sets the contents of the requested register.	

Note: This command operates on the LCA registers (status/config registers).

bcGetDPReg	
Prototype	int bcGetDPReg (UINT offset, UCHAR *data);
Packet	N/A
Input Parameter	Offset = 0 based offset of requested register. Data = pointer to unsigned char to return value requested.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Returns the contents of the requested register.	

Note: This command operates on the Dual Port Memory (packet interface).

bcSetDPReg	
Prototype	int bcSetDPReg (UINT offset, UCHAR *data)
Packet	N/A
Input Parameter	Offset = 0 based offset of requested register. Data = pointer to unsigned char value to be set.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Sets the contents of the requested register.	

Note: This command operates on the Dual Port Memory (packet interface).

bcReadBinTime	
Prototype	int bcReadBinTime (ULONG *major, ULONG *min, UCHAR *stat);
Packet	N/A
Input Parameter	major = unsigned long pointer to store major time (Unix format). min = unsigned long pointer to store microseconds. stat = unsigned char to store status bits.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Latches and returns time captured from the time registers.	

bcReadDecTime	
Prototype	int bcReadDecTime (struct tm *ptm, ULONG *min, UCHAR *stat);
Packet	N/A
Input Parameter	ptm = pointer to tm struct to store major time (calendar format). min = pointer to unsigned long to store microseconds. stat = pointer to unsigned char to store status bits.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Latches and returns time captured from the time registers.	

bcReadEventTime	
Prototype	int bcReadEventTime (ULONG *maj, ULONG *min);
Packet	N/A
Input Parameter	maj = pointer to unsigned long to store major time (Unix format). min = pointer to unsigned long to store microseconds.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Latches and returns time captured caused by an external event.	

bcSetBinTime	
Prototype	int bcSetBinTime (ULONG newtime);
Packet	0x12
Input Parameter	newtime = unsigned long time value to set (Unix format).
Returns	RC_OK On Success RC_ERROR On Failure
Description: Set the major time buffer.	

bcSetBCDTime	
Prototype	int bcSetBCDTime (struct tm stm);
Packet	0x12
Input Parameter	stm = tm struct containing new time values to set.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Set the major time buffer.	

bcSetYear	
Prototype	int bcSetYear (INT year);
Packet	0x13
Input Parameter	year = int value of new year (1990-2036).
Returns	RC_OK On Success RC_ERROR On Failure
Description: Set the current year value.	

bcSetMode	
Prototype	int bcSetMode (UCHAR mode);
Packet	0x10
Input Parameter	UCHAR mode: Sets the TFP operating mode. ***Note***: The following are defined in the "Bc637pci.h" header file; #define MODE_IRIG 0x00 #define MODE_FREE 0x01 #define MODE_1pps 0x02 #define MODE_RTC 0x03 #define MODE_GPS 0x06
Returns	RC_OK On Success RC_ERROR On Failure
Description: Sets the operating mode of the board.	

bcCommand	
Prototype	int bcCommand (int command);
Packet	0x1A
Input Parameter	int command = requested command action ***Note***: The following are defined in the "Bc637pci.h" header file. #define CMD_WARMSTART 0x01
Returns	RC_OK On Success RC_ERROR On Failure
Description: Sends reset command to the board.	

bcSetDac	
Prototype	int bcSetDac (int dacval);
Packet	0x24
Input Parameter	int dacval = new d/a value to modify frequency of internal oscillator. Allowed values 0x0000 - 0xffff.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Set new dac value.	

Note: This command is not required for standard operation of the device. Be sure to understand the effects of this operation before utilizing this command.

bcSetHbt	
Prototype	int bcSetHbt (char mode, int cnt1, int cnt2);
Packet	0x14
Input Parameter	char mode = requested mode int cnt1 = divisor 1 int cnt2 = divisor 2
Returns	RC_OK On Success RC_ERROR On Failure
Description: Program a periodic output (synchronous or asynchronous to 1pps).	

bcSetPDelay	
Prototype	int bcSetPDelay (long int delay);
Packet	0x17
Input Parameter	long int delay = propagation delay (-9999999 to +9999999 100ns steps)
Returns	RC_OK On Success RC_ERROR On Failure
Description: Program a propagation delay into the timing engine to account for delays introduced by long cable runs.	

Note: Usage of a propagation delay value with an absolute value larger than 1 millisecond (or 10000 steps) requires first that the user disable jamsynchs. Refer to the hardware manual for more information.

bcSetTcIn	
Prototype	int bcSetTcIn (UCHAR format, UCHAR type);
Packet	0x15, 0x16
Input Parameter	Unsigned char format = time code format. Unsigned char type = modulation type of time code. ***Note***: The following are defined in the "Bc637pci.h" header file; format #define TCODE_IRIG_A 'A' #define TCODE_IRIG_B 'B' #define TCODE_IEEE 'I' type #define TCODE_MOD_AM 'M' #define TCODE_MOD_DC 'D'
Returns	RC_OK On Success RC_ERROR On Failure
Description: Sets time code type and format for operating mode 0 (time code mode).	

bcSetClkSrc	
Prototype	int bcSetClkSrc (UCHAR which);
Packet	0x20
Input Parameter	Unsigned char which = which clock source (internal | external). ***Note***: The following are defined in the "Bc637pci.h" header file; #define CLK_INT 'I' /* Use on-board clock */ #define CLK_EXT 'E' /* Use external clock */
Returns	RC_OK On Success RC_ERROR On Failure
Description: Sets the 10MHz-clock source for the board.	

Note: This command is not required for standard operation of the device. Be sure to understand the effects of this operation before utilizing this command

bcSetGenCode	
Prototype	int bcSetGenCode (UCHAR format);
Packet	0x1B
Input Parameter	Unsigned char format = time code format. ***Note***: The following are defined in the "Bc637pci.h" header file; #define TCODE_GEN_B 'B' #define TCODE_GEN_I 'I'
Returns	RC_OK On Success RC_ERROR On Failure
Description: Sets the time code generator format.	

bcSetLocOff	
Prototype	int bcSetLocOff (INT Offset, UCHAR half)
Packet	0x1D
Input Parameter	INT Offset = hours from input time source. (-16 - +16). UCHAR half = half hour increment 0: No Half Hour increment 1: Add Half Hour increment
Returns	RC_OK On Success RC_ERROR On Failure
Description: Programs the board to operate at an offset from UTC.	

bcReqOscData	
Prototype	int bcReqOscData (OscData *pdata);
Packet	0x19
Input Parameter	pdata = pointer to OscData structure. The structure is defined in the "Bc637pci.h" header file.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Returns oscillator data from the board.	

Note: See the header file for OscData structure to understand values being returned.

bcReqTimeData	
Prototype	int bcReqTimeData (TimeData *pdata);
Packet	0x19
Input Parameter	pdata = pointer to TimeData structure. The structure is defined in the "Bc637pci.h" header file.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Returns time data from the board.	

Note: See the header file for TimeData structure to understand values being returned.

bcReqTimeCodeData	
Prototype	int bcReqTimeCodeData (TimeCodeData *pdata);
Packet	0x19
Input Parameter	pdata = pointer to TimeCodeData structure. The structure is defined in the "Bc637pci.h" header file.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Returns timecode data from the board.	

Note: See the header file for TimeCodeData structure to understand values being returned.

bcReqOtherData	
Prototype	int bcReqOtherData (OtherData *pdata);
Packet	0x19
Input Parameter	pdata = pointer to OtherData structure. The structure is defined in the "Bc637pci.h" header file.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Returns other data from the board.	

Note: See the header file for OtherData structure to understand values being returned.

bcReqVerData	
Prototype	int bcReqOscData (VerData *pdata);
Packet	0x19
Input Parameter	pdata = pointer to VerData structure. The structure is defined in the "Bc637pci.h" header file.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Returns version data from the board.	

Note: See the header file for VerData structure to understand values being returned.

bcReqManufData	
Prototype	int bcReqManufData (ManufData *pdata);
Packet	0x19
Input Parameter	pdata = pointer to ManufData structure. The structure is defined in the "Bc637pci.h" header file.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Returns manufacture data from the board.	

Note: See the header file for ManufData structure to understand values being returned.

bcSetGain	
Prototype	int bcSetGain (int gain);
Packet	0x25
Input Parameter	int gain = oscillator control filter gain value.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Modify the internal oscillator frequency control algorithm.	

Note: This command is not required for standard operation of the device. Be sure to understand the effects of this operation before utilizing this command.

bcSetGenOff	
Prototype	int bcSetGenOff (INT Offset, UCHAR half)
Packet	0x1C
Input Parameter	INT offset = hours from input time source. (-16 - +16). UCHAR half = half hour increment 0: No Half Hour increment 1: Add Half Hour increment
Returns	RC_OK On Success RC_ERROR On Failure
Description: Programs the board time code generator to operate at an offset from UTC.	

bcSetJam	
Prototype	int bcSetJam (INT jam_ctl);
Packet	0x21
Input Parameter	INT jam_ctl = 0 or 1 0 = Jam-Synchs Disabled 1 = Jam-Synchs Enabled
Returns	RC_OK On Success RC_ERROR On Failure
Description: Modify the internal oscillator frequency control algorithm.	

Note: This command is not required for standard operation of the device. Be sure to understand the effects of this operation before utilizing this command.

bcSetDis	
Prototype	int bcSetDis (INT dis_ctl);
Packet	0x23
Input Parameter	INT dis_ctl = 0 or 1 0 = Oscillator Disciplining Disabled 1 = Oscillator Disciplining Enabled (default)
Returns	RC_OK On Success RC_ERROR On Failure
Description: Modify the internal oscillator frequency control algorithm.	

Note: This command is not required for standard operation of the device. Be sure to understand the effects of this operation before utilizing this command.

bcSetLocalFlag	
Prototype	int bcSetLocalFlag (UCHAR local_flag);
Packet	0x40
Input Parameter	UCHAR local_flag = 0 or 1 ***Note***: The following are defined in the "Bc637pci.h" header file; #define LOCAL_FLAG_DIS (0) (default) #define LOCAL_FLAG_ENA (1)
Returns	RC_OK On Success RC_ERROR On Failure
Description: Enables or disables the local time offset entered using bcSetLocOff( ) function.	

bcSetDayLightFlag	
Prototype	int bcSetDayLightFlag (UCHAR dlight_flag);
Packet	0x41
Input Parameter	UCHAR dlight_flag = 0 or 1 ***Note***: The following are defined in the "Bc637pci.h" header file; #define DAY_LIGHT_DIS (0) (default) #define DAY_LIGHT_ENA (1)
Returns	RC_OK On Success RC_ERROR On Failure
Description: This command only applies Decoding IEEE 1344 Time Code. If the dlight_flag is enabled, the TFP adjusts its time by one hour.	

bcYearAutoInc	
Prototype	int bcYearInc (UCHAR year_inc);
Packet	0x42
Input Parameter	UCHAR year_inc = 0 or 1 ***Note***: The following are defined in the "Bc637pci.h" header file; #define YEAR_AUTO_DIS (0) #define YEAR_AUTO_ENA (1) (default)
Returns	RC_OK On Success RC_ERROR On Failure
Description: This commands the TFP to enable or disable the auto incrementing of the Year at the beginning of each year. The Year variable is stored into the EEPROM for reference.	

bcSetTmFmt	
Prototype	int bcSetTmFmt (INT tm_fmt);
Packet	0x11
Input Parameter	INT tm_fmt = 0 or 1 1 = Binary Time Format (Default) 0 = Decimal Time Format
Returns	RC_OK On Success RC_ERROR On Failure
Description: Modify the format of the major time data returned by the board.	

Note: This command is not required for standard operation of the device. Be sure to understand the effects of this operation before utilizing this command.

bcSetUtcCtl	
Prototype	int bcSetUtcCtl (INT utc_ctl);
Packet	0x33
Input Parameter	INT utc_ctl = 0 or 1 0 = UTC Format (Default) 1 = GPS Format
Returns	RC_OK On Success RC_ERROR On Failure
Description: Modify the time base in GPS mode. This command determines whether the board will correct the received GPS time for leap second offset and events.	

Note: This command is not required for standard operation of the device. Be sure to understand the effects of this operation before utilizing this command.

bcSetLeapEvent	
Prototype	int bcSetLeapEvent (CHAR flag, ULONG levent);
Packet	0x1E
Input Parameter	CHAR flag = -1, 0, 1 -1 = Deletion of 1 second 0 = Disable 1 = Insertion of 1 second ULONG levent = Unix time since January, 1st 1970
Returns	RC_OK On Success RC_ERROR On Failure
Description: This command can be used in modes other than GPS mode for inserting or deletion of one leap second.	

Note: This command is not required when the TFP is in GPS mode since the TFP automatically handles Leap seconds insertion or deletion.

bcAdjustClock	
Prototype	int bcAdjustClock (LONG clkvalue);
Packet	0x29
Input Parameter	LONG clkvalue = 0x80000000 to 0x7FFFFFFF
Returns	RC_OK On Success RC_ERROR On Failure
Description: This command advance/retard the TFP internal clock. The TFP can adjust its clock up to 100 milliseconds per each second. Each count is equal to 10 microseconds.	

Note: This command is not required for standard operation of the device. Be sure to understand the effects of this operation before utilizing this command.

bcSpecialBoot	
Prototype	int bcSpecialBoot (INT sp_boot);
Packet	0xFB
Input Parameter	INT sp_boot; ***Note***: The following are defined in the "Bc637pci.h" header file; #define NORMAL_BOOT 0x0000 #define SPECIAL_BOOT 0x0100
Returns	RC_OK On Success RC_ERROR On Failure
Description: In Special Boot Configuration, the TFP ignores additional resets after power-on. ***Note:*** This command is not required for standard operation of the device. Be sure to understand the effects of this operation before utilizing this command	

bcSetPciCard	
Prototype	int bcSetPciCard (UCHAR setting, ULONG password, INT data);
Packet	0xFE
Input Parameter	***Note***: The following are defined in the "Bc637pci.h" header file; UCHAR setting = module settings #define MODEL_ID 0x04 /* IRIG or GPS Model */ #define CRYSTAL_ID 0x03 /* Standard or Oven crystal */ ULONG password; Datum will supply the password INT data; For Model ID #define SET_BC635 0x0635 #define SET_BC637 0x0637 For Crystal ID #define STD_CRYSTAL 0x0002 #define MTI_CRYSTAL 0x0014
Returns	RC_OK On Success RC_ERROR On Failure
Description: This command sets the manufacture settings of the module. To change any of these settings, a password is required. This command is mainly used for field upgrades.	

Note: This command is not required for standard operation of the device. Be sure to understand the effects of this operation before utilizing this command.

bcForceJam	
Prototype	int bcForceJam (void);
Packet	0x22
Input Parameter	None
Returns	RC_OK On Success RC_ERROR On Failure
Description: This command forces the TFP to Jam-Sync on the next rising edge of the 1PPS output. The Jam-Sync bit must be enabled before using this command.	

bcSyncRtc	
Prototype	int bcSyncRtc (void);
Packet	0x27
Input Parameter	None
Returns	RC_OK On Success RC_ERROR On Failure
Description: This command forces the TFP to Synchronize the RTC time to current time.	

bcDisRtcBatt	
Prototype	int bcDisRtcBatt (void);
Packet	0x28
Input Parameter	None
Returns	RC_OK On Success RC_ERROR On Failure
Description: This command disconnects the RTC IC from the Battery after power is turned off. Upon power on, the TFP automatically connects the RTC IC to the battery.	

bcGPSReq	
Prototype	int bcGPSReq (GpsPkt *ptr);
Packet	0x31
Input Parameter	GpsPkt *ptr This structure commands information detailing the packet to retrieve and the buffer where the data will be stored.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Retrieve a data packet from the GPS receiver. Refer to the bc637PCI User's Guide for more details regarding this command. (See PACKET 0x31 definition.)	

bcGPSSnd	
Prototype	int bcGPSSnd (GpsPkt *ptr);
Packet	0x30
Input Parameter	GpsPkt *ptr This structure commands information detailing the packet to send and the buffer where the data is stored.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Send a data packet from the GPS receiver. Refer to the bc637PCI User's Guide for more details regarding this command. (See PACKET 0x30 definition.)	

bcGPSMan	
Prototype	int bcGPSMan (GpsPkt *ptrIn, GpsPkt *ptrOut);
Packet	0x32
Input Parameter	GpsPkt *ptrIn This structure commands information detailing the packet to send and the buffer where the data is stored. .GpsPkt *ptrOut This structure commands information detailing the packet to retrieve and the buffer where the data will be stored.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Manually send and retrieve data packets from the GPS receiver. Refer to the bc637PCI User's Guide for more details regarding this command. (See PACKET 0x32 definition.)	

bcGPSOperMode	
Prototype	int bcGPSReq (UCHAR static);
Packet	0x34
Input Parameter	UCHAR static = 0 or 1 ***Note***: The following are defined in the "Bc637pci.h" header file; #define GPS_NONE_STATIC (0) #define GPS_STATIC (1) (default)
Returns	RC_OK On Success RC_ERROR On Failure
Description: By default, the TFP directs the GPS receiver to Static Mode of Operation after the TFP is tracking to GPS. This Command allows the user to disable this feature. See Packet 2C in the GPS Manual for detail description on this feature. This function should only be used when the TFP is in GPS Mode of Operation.	

bcStartInt	
Prototype	int bcStartInt (HWND hWnd, INT dev_no, INT int_mode);
Packet	N/A
Input Parameter	HWND hWnd = Window handle to receive interrupt messages. INT dev_no = bc635PCI device to open (same used in bcStartPCI). INT int_mode = type of interrupt. ***Note***: The following are defined in bc_int.h #define BC_INT_ONE_SHOT 1 #define BC_INT_RECURRING 2
Returns	RC_OK On Success RC_ERROR On Failure
Description: Start the interrupt thread. This thread will send a message to the program using the window handle passed in. The two allowed messages are; #define WM_INT_DYING 0x7026 #define WM_INT_DETECTED 0x7025	

bcStopInt	
Prototype	int bcStopInt (void);
Packet	N/A
Input Parameter	None
Returns	RC_OK On Success RC_ERROR On Failure
Description: Stop the interrupt thread. This thread will send a message to the program using the window handle passed in. The two allowed messages are; #define WM_INT_DYING 0x7026 #define WM_INT_DETECTED 0x7025	

bcSetInts	
Prototype	int bcSetInts (ULONG *mask, INT *latch);
Packet	N/A
Input Parameter	ULONG *mask = pointer to mask to load into INTERRUPT MASK register. ***Note***: The following are defined in the header file; #define INT_EVENT (1<<0) #define INT_HBEAT (1<<1) #define INT_STROBE (1<<2) #define INT_1pps (1<<3) #define INT_DPA (1<<4) INT *latch = 0 or 1 0 = do not latch time in event registers when interrupt detected. 1 = latch time in event registers when interrupt detected.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Enable one or more interrupt sources. This thread will send a message to the program using the window handle passed in. The two allowed messages are; #define WM_INT_DYING 0x7026 #define WM_INT_DETECTED 0x7025	

Note: Refer to the bc635PCI User's Guide for more information regarding allowed values for the INTERRUPT MASK.

bcReqInts	
Prototype	int bcReqInts (ULONG *mask, INT *latch);
Packet	None
Input Parameter	ULONG *mask = pointer to mask to load from INTERRUPT MASK register. INT *latch = value of current latch setting.
Returns	RC_OK On Success RC_ERROR On Failure
Description: Query the currently enabled interrupts.	

Note: Refer to the bc635PCI User's Guide for more information regarding allowed values for the INTERRUPT MASK.

This Page Intentionally Left Blank.