bc635/637PCI-U
Linux Developer's Kit

User's Guide

Rev 05 February 2005

CHAPTER ONE

INTRODUCTION

1.0 GENERAL

The bc635/637PCI-U Developer's Kit is designed to provide a suite of tools useful in the development of applications which access features of the bc635/637PCI-U Time and Frequency Processor. This kit has been designed to provide an interface between the bc635/637PCI-U and applications developed for Linux 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 features of the bc635/637PCI-U.
- 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 bc635/637PCI-U Time and Frequency Processor in the Linux OS environment. The example program 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 program was developed using discrete functions for each operation, which allows the developer to clip any useful code and use it in their own applications.

CHAPTER TWO

INSTALLATION

2.0 HARDWARE INSTALLATION

Installation of PCI boards is quite a bit simpler than in most bus architectures due to two factors:

- Geographical addressing, which eliminates the need for DIP switches and jumpers normally required to select a “base address” or interrupt level for plug-in modules.

- Auto configuration, which allows the host computer to read the device ID and other configuration information directly from the Configuration Registers.

The only thing the user has to do is pick a vacant PCI slot and plug the bc635/637PCI-U Time and Frequency Processor (TFP) into it and install the software. Be sure to consult the user documentation that came with your particular workstation for any specific PCI card installation instructions.

2.1 SOFTWARE INSTALLATION

Because the bc635/637PCI-U driver is a KLM, the Linux source code including versions.h must be installed on the system for the driver to install correctly. To install the Software driver and the sample program follow the following steps:

1. Create a directory /usr/bin/bc635pci

```
/usr/bin> mkdir bc635pci
```

2. Make bc635pci your active directory

```
/usr/bin> cd bc635pci
```

3. Extract the file BCPCIVxxx.tgz
Substitute the xxx with the actual version number
```
/usr/bin/bc635pci> tar xvzf BCPCIVxxx.tgz
```

5. Compile Driver

```
/usr/bin/bc635pci> make
```

6. Become super user

```
/usr/bin/bc635pci> su
```

7. Install Driver

```
/usr/bin/bc635pci> make install
```

8. Configure Driver

```
Change the user and group ids and give read/write permissions to the
device file /dev/windrvr6 depending on how you wish to allow users to
access hardware through the device.
```

2.2 TEST INSTALLATION

Rebuild the sample test program to verify that the software installation was successful.

1. Make sample your active directory

```
/usr/bin/bc635pci> cd sample
```

2. Rebuild the sample code

```
/usr/bin/bc635pci/sample> make
```

3. Run the sample program

```
/usr/bin/bc635pci/sample> ./pcidemo
```

Note: If a device open error is received, do the following:

1. Restart the computer

2. Make drvr your active directory

```
/usr/bin/bc635pci> cd drvr
```

3. Run this command

The installer should have created a folder called "LINUX.x.x.x.x" under the bc635pci path.

Substitute the x.x.x.x with the actual folder name, and run the command bellow

```
Linux Kernel 2.4.x and lower:

/usr/bin/bc635pci/drvr> ./wdreg LINUX.x.x.x.x/windrvr6.o

Linux Kernel 2.6.x and higher:

/usr/bin/bc635pci/drvr> ./wdreg LINUX.x.x.x.x/windrvr6.ko
```

Note: You need to run this command every time you power-up the machine. You can add this command to a batch file to automatically run every time you power-up the machine.

4. Run the sample program

```
/usr/bin/bc635pci/sample> ./pcidemo
```

2.3 UNINSTALL INSTRUCTIONS

Note: You must be logged in as root in order to uninstall.

1. **Uninstall the driver service**

   - Do a /sbin/lsmod to check if the WinDriver module is in use by any application or by other modules. Make sure no programs are using WinDriver. If any application or module is using WinDriver, close all applications and do a /sbin/rmmod to remove any module using WinDriver.

   - Run the command "/sbin/rmmod windrvr6"

   - rm -rf /dev/windrvr6 (Remove the old device node in the /dev directory.)

2. **Delete the bc635pci installation directory**

Use the command rm -rf /usr/bin/bc635pci

CHAPTER THREE

LIBRARY DEFINITIONS

3.0 GENERAL

The interface library provides functions for each of the programming packets supported by the bc635/637PCI-U 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	BC_PCI_HANDLE bcStartPci (BYTE Interrupts);
Packet	N/A
Input Parameter	BYTE Interrupts: Open device with/without interrupts support. The allowed values are defined in the 'bcuser.h' file: #define ENABLE_INTERRUPTS 1 #define DISABLE_INTERRUPTS 0
Returns	BC_PCI_HANDLE hBC_PCI On Success NULL On Failure
Description: This opens the underlying hardware layer. The return handle is used with the rest of the functions.	

bcStopPci	
Prototype	void bcStopPci (BC_PCI_HANDLE hBC_PCI);
Packet	N/A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function
Returns	None

Description: Closes the underlying hardware layer and releases any used resources.

bcReadReg	
Prototype	BOOL bcReadReg (BC_PCI_HANDLE hBC_PCI, DWORD Offset, PDWORD Data);
Packet	N/A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function Offset = See defined Registers offsets in the "bcuser.h" header file. Data = pointer to unsigned long to return value requested.
Returns	TRUE On Success FALSE On Failure
Description: Returns the contents of the requested register.	

bcWriteReg	
Prototype	BOOL bcWriteReg (BC_PCI_HANDLE hBC_PCI, DWORD Offset, DWORD Data);
Packet	N/A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function Offset = See defined Registers offsets in the "bcuser.h" header file. Data = unsigned long value to be set.
Returns	TRUE On Success FALSE On Failure
Description: Sets the contents of the requested register.	

bcReadBinTime	
Prototype	BOOL bcReadBinTime (BC_PCI_HANDLE hBC_PCI, PDWORD major, PDWORD min, PBYTE stat);
Packet	N/A

Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function 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	TRUE On Success FALSE On Failure
Description: Latches and returns time captured from the time registers.	

bcReadDecTime	
Prototype	BOOL bcReadDecTime (BC_PCI_HANDLE hBC_PCI, struct tm *ptm, PDWORD ulpMin, PBYTE pstat);
Packet	N/A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function ptm = pointer to tm stuct to store major time (calendar format). ulpMin = pointer to unsigned long to store microseconds. pstat = pointer to unsigned char to store status bits.
Returns	TRUE On Success FALSE On Failure
Description: Latches and returns time captured from the time registers.	

bcSetBinTime	
Prototype	BOOL bcSetBinTime (BC_PCI_HANDLE hBC_PCI, DWORD newtime);
Packet	0x12
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function newtime = unsigned long time value to set (Unix format).
Returns	TRUE On Success

	FALSE On Failure
Description: Set the major time buffer.	

bcSetDecTime	
Prototype	BOOL bcSetDecTime (BC_PCI_HANDLE hBC_PCI, struct tm);
Packet	0x12
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function tm = tm struct containing new time values to set.
Returns	TRUE On Success FALSE On Failure
Description: Set the major time buffer.	

bcReqYear	
Prototype	BOOL bcReqYear (BC_PCI_HANDLE hBC_PCI, PDWORD year);
Packet	0x19
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function year = pointer to unsigned long value of new year (1990-2036).
Returns	TRUE On Success FALSE On Failure
Description: Request the current year value.	

bcSetYear	
Prototype	BOOL bcSetYear (BC_PCI_HANDLE hBC_PCI, DWORD year);
Packet	0x13
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function year = value of new year (1990-2036).
Returns	TRUE On Success

	FALSE On Failure
Description: Set the current year value.	

bcReadEventTime	
Prototype	BOOL bcReadEventTime (BC_PCI_HANDLE hBC_PCI, PDWORD maj, PDWORD min, PBYTE stat);
Packet	N/A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function maj = pointer to unsigned long to store major time (Unix format). min = pointer to unsigned long to store microseconds. stat = unsigned char to store status bits.
Returns	TRUE On Success FALSE On Failure
Description: Latches and returns time captured caused by an external event.	

bcSetStrobeTime	
Prototype	BOOL bcSetStrobeTime (BC_PCI_HANDLE hBC_PCI, DWORD dMaj, DWORD dMin);
Packet	N/A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function dMaj = unsigned long value for strobe major time. dMin = unsigned long value for strobe minor time.
Returns	TRUE On Success FALSE On Failure
Description: Set the strobe time.	

bcReqTimeFormat

Prototype	BOOL bcReqTimeFormat (BC_PCI_HANDLE hBC_PCI, PBYTE timeformat);
Packet	0x19
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function timeformat = pointer to unsigned char value for time format. The allowed values are defined in the 'bcuser.h' file: enum { FORMAT_DECIMAL = 0x00 }; enum { FORMAT_BINARY = 0x01 };
Returns	TRUE On Success FALSE On Failure
Description: Request current time format	

bcSetTimeFormat	
Prototype	BOOL bcSetTimeFormat (BC_PCI_HANDLE hBC_PCI, BYTE tmfmt);
Packet	0x11
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function tmfmt = unsigned char value for time format. The allowed values are defined in the 'bcuser.h' file: enum { FORMAT_DECIMAL = 0x00 }; enum { FORMAT_BINARY = 0x01 };
Returns	TRUE On Success FALSE On Failure
Description: Set time format	

bcSetMode	
Prototype	void bcSetMode (BC_PCI_HANDLE hBC_PCI, BYTE mode);
Packet	0x10
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function mode = unsigned char value for new operating mode. The allowed values are defined in the 'bcuser.h' file: enum { MODE_IRIG = 0x00 }; enum { MODE_FREE = 0x01 }; enum { MODE_1PPS = 0x02 }; enum { MODE_RTC = 0x03 }; enum { MODE_GPS = 0x06 };
Returns	None
Description: Sets the operating mode of the board.	

bcSetLocOff	
Prototype	BOOL bcSetLocOff (BC_PCI_HANDLE hBC_PCI, USHORT offset, BYTE half);
Packet	0x1D
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function Offset = hours from input time source. (-16 - +16). half = half hour increment 0: No Half Hour increment 1: Add Half Hour increment
Returns	TRUE On Success FALSE On Failure

Description: Programs the board to operate at an offset from UTC.

bcSetGenOff	
Prototype	BOOL bcSetGenOff (BC_PCI_HANDLE hBC_PCI, USHORT offset, BYTE half);
Packet	0x1C
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function offset = hours from input time source. (-16 - +16). half = half hour increment 0: No Half Hour increment 1: Add Half Hour increment
Returns	TRUE On Success FALSE On Failure
Description: Programs the board time code generator to operate at an offset from UTC.	

bcSetPropDelay	
Prototype	BOOL bcSetPropDelay (BC_PCI_HANDLE hBC_PCI, long value);
Packet	0x17
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function value = long data for propagation delay.
Returns	TRUE On Success FALSE On Failure
Description: Sets the propagation delay offset.	

bcSetHbt	
Prototype	BOOL bcSetHbt (BC_PCI_HANDLE hBC_PCI, BYTE mode, USHORT n1, USHORT n2);

Packet	0x14
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function mode = unsigned char value for the heartbeat mode. n1 = unsigned short value for heartbeat counter 1. n2 = unsigned short value for heartbeat counter 2. The allowed values are defined in the 'bcuser.h' file: enum { PERIODIC_SYNC = 0x01 }; enum { PERIODIC_NOSYNC = 0x00 };
Returns	TRUE On Success FALSE On Failure
Description: Sets the heartbeat counters and mode.	

bcSetTcIn	
Prototype	BOOL bcSetTcIn (BC_PCI_HANDLE hBC_PCI, BYTE TcIn);
Packet	0x15
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function TcIn = unsigned char value for time code input. The allowed values are defined in the 'bcuser.h' file: enum { TCODE_IRIG_A = 0x41 }; enum { TCODE_IRIG_B = 0x42 }; enum { TCODE_IEEE = 0x49 }; enum { TCODE_NASA = 0x4E };
Returns	TRUE On Success FALSE On Failure

Description: Sets the input time code format.

bcSetTcInMod	
Prototype	BOOL bcSetTcInMod (BC_PCI_HANDLE hBC_PCI, BYTE TcInMod);
Packet	0x16
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function TcInMod = unsigned char value for time code input modulation. The allowed values are defined in the 'bcuser.h' file: enum { TCODE_MOD_AM = 0x4D }; enum { TCODE_MOD_DC = 0x44 };
Returns	TRUE On Success FALSE On Failure
Description: Sets the input time code modulation.	

bcSetGenCode	
Prototype	BOOL bcSetGenCode (BC_PCI_HANDLE hBC_PCI, BYTE GenTc);
Packet	0x1B
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function GenTc = unsigned char value for the time code output. The allowed values are defined in the 'bcuser.h' file: enum { TCODE_IRIG_B = 0x42 }; enum { TCODE_IEEE = 0x49 };
Returns	TRUE On Success FALSE On Failure

Description: Sets the output time code format.

bcSetLeapEvent	
Prototype	BOOL bcSetLeapEvent (BC_PCI_HANDLE hBC_PCI, char flag, DWORD leapevt);
Packet	0x1E
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function flag = char value for the leap event flag. leapevt = unsigned long value for the leap event time.
Returns	TRUE On Success FALSE On Failure
Description: Sets the leap event time.	

bcSetClkSrc	
Prototype	BOOL bcSetClkSrc (BC_PCI_HANDLE hBC_PCI, BYTE clk);
Packet	0x20
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function clk = unsigned char value for the clock source. The allowed values are defined in the 'bcuser.h' file: enum { CLK_INT = 0x49 }; enum { CLK_EXT = 0x45 };
Returns	TRUE On Success FALSE On Failure
Description: Sets the clock source, Internal/External.	

bcSetDac	
Prototype	BOOL bcSetDac (BC_PCI_HANDLE hBC_PCI, USHORT dac);
Packet	0x24
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function dac = unsigned short value for the DAC.
Returns	TRUE On Success FALSE On Failure
Description: Sets the DAC value	

bcSetGain	
Prototype	BOOL bcSetGain (BC_PCI_HANDLE hBC_PCI, short gain);
Packet	0x25
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function gain = short value for the Gain.
Returns	TRUE On Success FALSE On Failure
Description: Sets the GAIN.	

bcSetJam	
Prototype	BOOL bcSetJam (BC_PCI_HANDLE hBC_PCI, BYTE jam);
Packet	0x21
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function jam = unsigned char value for enabling/disabling jam-sync. The allowed values are defined in the 'bcuser.h' file: enum { JAM_SYNC_ENA = 0x01 }; enum { JAM_SYNC_DIS = 0x00 };
Returns	TRUE On Success

	FALSE On Failure
Description: Sets the Jam-Sync.	

bcSetGpsTmFmt	
Prototype	BOOL bcSetGpsTmFmt (BC_PCI_HANDLE hBC_PCI, BYTE gpsfmt);
Packet	0x33
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function gpsfmt = unsigned char value for gps time format. The allowed values are defined in the 'bcuser.h' file: enum { GPS_TIME_FMT = 0x01 }; enum { UTC_TIME_FMT = 0x00 };
Returns	TRUE On Success FALSE On Failure
Description: Sets the GPS time format.	

bcSetGpsOperMode	
Prototype	BOOL bcSetGpsOperMode (BC_PCI_HANDLE hBC_PCI, BYTE gpsmode);
Packet	0x34
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function gpsmode = unsigned char value for gps mode. The allowed values are defined in the 'bcuser.h' file: enum { GPS_STATIC = 0x01 }; enum { GPS_NONE_STATIC = 0x00 };
Returns	TRUE On Success

	FALSE On Failure
Description: Sets the GPS operating mode.	

bcSetLocalOffsetFlag	
Prototype	BOOL bcSetLocalOffsetFlag (BC_PCI_HANDLE hBC_PCI, BYTE flagoff);
Packet	0x40
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function flagoff = unsigned char value for enabling/disabling local offset time. The allowed values are defined in the 'bcuser.h' file: enum { LOCAL_OFF_ENABLE = 0x01 }; enum { LOCAL_OFF_DISABLE = 0x00 };
Returns	TRUE On Success FALSE On Failure
Description: Sets the local offset flag.	

bcSetYearAutoIncFlag	
Prototype	BOOL bcSetYearAutoIncFlag (BC_PCI_HANDLE hBC_PCI, BYTE yrinc);
Packet	0x42
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function yrinc = unsigned char value for enabling/disabling year auto-increment flag. The allowed values are defined in the 'bcuser.h' file: enum { YEAR_AUTO_ENA = 0x01 };

	enum { YEAR_AUTO_DIS = 0x00 };
Returns	TRUE On Success FALSE On Failure
Description: Sets the year auto increment flag.	

bcAdjustClock	
Prototype	BOOL bcAdjustClock (BC_PCI_HANDLE hBC_PCI, long cval);
Packet	0x29
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function cval = long value for adjusting the clock
Returns	TRUE On Success FALSE On Failure
Description: Advance/Retard clock value.	

bcCommand	
Prototype	void bcCommand (BC_PCI_HANDLE hBC_PCI, BYTE cmd);
Packet	0x1A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function cmd = unsigned char value for software reset. The allowed value is defined in the 'bcuser.h' file: enum { CMD_WARMSTART = 0x01 };
Returns	None
Description: Software reset.	

bcForceJam	

Prototype	BOOL bcForceJam (BC_PCI_HANDLE hBC_PCI);
Packet	0x22
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function
Returns	TRUE On Success FALSE On Failure
Description: Forces a Jam-Sync.	

bcSyncRtc	
Prototype	BOOL bcSyncRtc (BC_PCI_HANDLE hBC_PCI);
Packet	0x27
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function
Returns	TRUE On Success FALSE On Failure
Description: Sync RTC clock with current time.	

bcDisRtcBatt	
Prototype	BOOL bcDisRtcBatt (BC_PCI_HANDLE hBC_PCI);
Packet	0x28
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function
Returns	TRUE On Success FALSE On Failure
Description: Disable battery.	

bcReqSerialNum	
Prototype	BOOL bcReqSerialNum (BC_PCI_HANDLE hBC_PCI, PDWORD serial);
Packet	0xFE
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function

	serial = pointer to unsigned long value for serial number of the board.
Returns	TRUE On Success FALSE On Failure
Description: Request serial number of the board.	

bcReqHardwarFab	
Prototype	BOOL bcReqHardwarFab (BC_PCI_HANDLE hBC_PCI, PWORD fab);
Packet	0xF5
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function fab = pointer to unsigned short value for the hardware fab.
Returns	TRUE On Success FALSE On Failure
Description: Request hardware fab of the board.	

bcReqAssembly	
Prototype	BOOL bcReqAssembly (BC_PCI_HANDLE hBC_PCI, PWORD num);
Packet	0xF4
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function num = pointer to unsigned short value for the assembly number.
Returns	TRUE On Success FALSE On Failure
Description: Request assembly number of the board.	

bcReqOscData	
Prototype	BOOL bcReqOscData (BC_PCI_HANDLE hBC_PCI, OscData *pdata);
Packet	0x19
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function

	pdata = pointer to OscData structure. The structure is defined in the "bcuser.h" header file.
Returns	TRUE On Success FALSE On Failure
Description: Request Oscillator data of the board.	

bcReqTimeCodeData	
Prototype	BOOL bcReqTimeCodeData (BC_PCI_HANDLE hBC_PCI, TimeCodeData *pdata);
Packet	0x19
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function pdata = pointer to TimeCodeData structure. The structure is defined in the "bcuser.h" header file.
Returns	TRUE On Success FALSE On Failure
Description: Request time code data of the board.	

bcReqTimeData	
Prototype	BOOL bcReqTimeData (BC_PCI_HANDLE hBC_PCI, TimeData *pdata);
Packet	0x19
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function pdata = pointer to TimeData structure. The structure is defined in the "bcuser.h" header file.
Returns	TRUE On Success FALSE On Failure
Description: Request time data of the board.	

bcReqOtherData	
Prototype	BOOL bcReqOtherData (BC_PCI_HANDLE hBC_PCI, OtherData *pdata);
Packet	0x19
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function pdata = pointer to OtherData structure. The structure is defined in the "bcuser.h" header file.
Returns	TRUE On Success FALSE On Failure
Description: Request other data of the board.	

bcReqVerData	
Prototype	BOOL bcReqVerData (BC_PCI_HANDLE hBC_PCI, VerData *pdata);
Packet	0x19
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function pdata = pointer to VerData structure. The structure is defined in the "bcuser.h" header file.
Returns	TRUE On Success FALSE On Failure
Description: Request version data of the board.	

bcReqModel	
Prototype	BOOL bcReqModel (BC_PCI_HANDLE hBC_PCI, ModelData *pdata);
Packet	0x19
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function pdata = pointer to ModelData structure. The structure is defined in the "bcuser.h" header file.
Returns	TRUE On Success

	FALSE On Failure
Description: Request model data of the board.	

bcGPSReq	
Prototype	BOOL bcGPSReq (BC_PCI_HANDLE hBC_PCI, GpsPkt *pktout);
Packet	0x31
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function
	pktout = structure commands information detailing the packet to retrieve and the buffer
Returns	TRUE On Success
	FALSE On Failure
Description: Retrieve a data packet from the GPS receiver. Refer to the User's Guide for more details regarding this command. (See packet 0x31 definition)	

bcGPSSnd	
Prototype	BOOL bcGPSSnd (BC_PCI_HANDLE hBC_PCI, GpsPkt *pktin);
Packet	0x30
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from "bcStarPci"
	pktin = structure commands information detailing the packet to send and the buffer.
Returns	TRUE On Success
	FALSE On Failure
Description: Send a data packet to the GPS receiver. Refer to the User's Guide for more details regarding this command. (See packet 0x30 definition)	

bcGPSMan	
Prototype	BOOL bcGPSMan (BC_PCI_HANDLE hBC_PCI, GpsPkt *pktin,
	GpsPkt *pktout);
Packet	0x32

Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function pktin = structure commands information detailing the packet to send and the buffer. pktout = structure commands information detailing the packet to retrieve and the buffer
Returns	TRUE On Success FALSE On Failure
Description: Manually send and retrieve data packets from the GPS receiver. Refer to the User's Guide for more details regarding this command. (See Packet0x32 definition)	

bcStartInt	
Prototype	BOOL bcStartInt (BC_PCI_HANDLE hBC_PCI);
Packet	N/A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function
Returns	TRUE On Success FALSE On Failure
Description: Start the interrupt thread. This thread will execute bcShowInt() function every time an interrupt is detected.	

bcStopInt	
Prototype	void bcStopInt (BC_PCI_HANDLE hBC_PCI);
Packet	N/A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function
Returns	None
Description: Stops the interrupt thread.	

bcSetInt	
Prototype	BOOL bcSetInt (BC_PCI_HANDLE hBC_PCI, BYTE IntVal);
Packet	N/A

Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function IntVal = unsigned char value for selecting the interrupt source. The allowed values are defined in the 'bcuser.h' file: enum { INTERRUPT_EVENT = 0x01 }; enum { INTERRUPT_PERIODIC = 0x02 }; enum { INTERRUPT_STROBE = 0x04 }; enum { INTERRUPT_1PPS = 0x08 }; enum { INTERRUPT_GPS = 0x10 };
Returns	TRUE On Success FALSE On Failure
Description: Enable one-interrupt sources.	

bcReqInt	
Prototype	BOOL bcReqInt (BC_PCI_HANDLE hBC_PCI, PBYTE Ints);
Packet	N/A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function Ints = pointer on unsigned char value for current used interrupt. The allowed values are defined in the 'bcuser.h' file: enum { INTERRUPT_EVENT = 0x01 }; enum { INTERRUPT_PERIODIC = 0x02 }; enum { INTERRUPT_STROBE = 0x04 }; enum { INTERRUPT_1PPS = 0x08 }; enum { INTERRUPT_GPS = 0x10 };
Returns	TRUE On Success

	FALSE On Failure
Description: Query the current enabled interrupt.	

bcShowInt	
Prototype	void bcShowInt (BC_PCI_HANDLE hBC_PCI);
Packet	N/A
Input Parameter	BC_PCI_HANDLE hBC_PCI : Handle returned from 'bcStarPci' function
Returns	None
Description: This function is used as an interrupt service routine. The user can add any code in this function to perform tasks once an interrupt is detected.	