

bc635PCI

Linux Developer's Kit

User's Guide

**bc635PCI
LINUX DRIVER**

TABLE OF CONTENTS

| SECTION | PAGE |
|---|-------------|
| CHAPTER ONE | |
| INTRODUCTION | |
| 1.0 General..... | 1-1 |
| 1.1 Features..... | 1-1 |
| 1.2 Overview..... | 1-1 |
| CHAPTER TWO | |
| INSTALLATION | |
| 2.0 Hardware and Driver Installation | 2-1 |
| 2.1 Software Developer's Kit Installation | 2-1 |
| 2.3 Test Installation..... | 2-2 |
| CHAPTER THREE | |
| LIBRARY DEFINITIONS | |
| 3.0 General..... | 3-1 |
| 3.1 Functions..... | 3-1 |

This Page Intentionally Left Blank.

INTRODUCTION

1.0 GENERAL

The bc635PCI Developer's Kit is designed to provide a suite of tools useful in the development of applications which access features of the bc635PCI Time and Frequency Processor. This kit has been designed to provide an interface between the bc635PCI 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 bc635PCI.
- 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 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.

This Page Intentionally Left Blank.

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 Datum bc635PCI 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

To install the Software driver and the sample program follow the following steps:

1. Create a directory /usr/bin/datum

```
/usr/bin> mkdir datum
```

2. Make datum your active directory

```
/usr/bin> cd datum
```

3. Extract the file DATUMPCI.tgz

```
/usr/bin/datum> tar xvzf DATUMPCI.tgz
```

4. Make drvr your active directory

```
/usr/bin/datum> cd drvr
```

5. Install Driver

```
/usr/bin/datum/drvr> make install IS_REGISTERED=1
```

6. Back up one directory

```
/usr/bin/datum/drvr> cd ..
```

CHAPTER THREE

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/datum> cd sample
```

2. Rebuild the sample code

```
/usr/bin/datum/sample> make
```

3. Run the sample program

```
/usr/bin/datum/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/datum> cd drvr
```

3. Run this command

The installer should have created a folder called "LINUX.x.x.x.x" under the datum path. Substitute the x.x.x.x with the actual folder name, and run the command below

```
/usr/bin/datum/drvr> ./wdreg LINUX.x.x.x.x/windrvr.o
```

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/datum/sample> ./pcidemo
```


CHAPTER THREE**LIBRARY DEFINITIONS****3.0 GENERAL**

The interface library provides functions for each of the programming packets supported by the bc635/7PCI 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 struct 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> Sets the GAIN. | |

3-8

| 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> Sets the GPS operating mode. | |

3-9

| 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> Request Oscillator data of the board. | |

3-12

| 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. | |

3-13

| 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> 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 |
| <i>Description:</i> Stops the interrupt thread. | |

3-15

| 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 }; |

| | |
|---|---|
| | <pre>enum { INTERRUPT_PERIODIC = 0x02 }; enum { INTERRUPT_STROBE = 0x04 }; enum { INTERRUPT_1PPS = 0x08 }; enum { INTERRUPT_GPS = 0x10 };</pre> |
| Returns | TRUE On Success FALSE On Failure |
| <i>Description:</i> 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: <pre>enum { INTERRUPT_EVENT = 0x01 }; enum { INTERRUPT_PERIODIC = 0x02 }; enum { INTERRUPT_STROBE = 0x04 }; enum { INTERRUPT_1PPS = 0x08 }; enum { INTERRUPT_GPS = 0x10 };</pre> |
| Returns | TRUE On Success FALSE On Failure |
| <i>Description:</i> 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 |
| <i>Description:</i> 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. | |