bc620AT
Developer's Kit

8500-0086
User's Guide
November, 1998

CHAPTER ONE
INTRODUCTION

1.0 GENERAL

The bc620AT Developer's Kit is designed to provide a suite of tools useful in the development of applications which access features of the Datum bc620AT Time & Frequency Processor. This kit has been designed to provide an interface between the bc620AT and applications developed for Windows 95™, and Windows NT™ environments. In addition to the interface DLL, two example programs are 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 bc620AT.
- Hardware driver for Windows NT™ and VxD for Windows 95™
- Example programs, with source, utilizing the interface library.
- Console application to configure registry keys.
- User's Guide providing a library definition.

1.2 OVERVIEW

The Developer's Kit was designed to provide an interface to the bc620AT Time & Frequency Processor in the 32-bit environments of Windows 95™ and Windows NT™ . The example programs were developed under Microsoft Visual C ++ 5.0. The example programs provides sample code which exercise the interface DLL 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 which 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 bc620AT hardware. Application programs developed using the 32-bit interface DLL 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. A discrete 32-bit console application is provided in the Developer's Kit which can be distributed to end users to configure registry keys to access the hardware interface.

CHAPTER TWO
INSTALLATION

2.0 GENERAL

Installation of the Developer's Kit is handled by the installer program. Following the installation, the user must set up the appropriate hardware driver and registry key information for the operating system. The following steps are required for a full system installation.

- Use the setup.exe program on the Developer's Kit to install the kit.
- Copy the appropriate hardware driver to the system location.
- Use the supplied registry utility to configure the registry keys.
- Use the compiled example programs to test the system.

Note: A reboot is necessary after configuring the registry entries for the first time.

2.1 CONFIGURATION

Directory structures are created in the specified location. These structures contains all required files to develop 32-bit user applications. In addition, copies of the hardware driver files and configuration utilities are provided for redistribution with user-developed 32-bit applications.

Directory of dist\…\Example Programs\Bc620atDemoCpp
This directory contains all the files for rebuilding the example program.

Directory of dist\…\Example Programs\Bc620atClockCpp
This directory contains all files for rebuilding the example program.

Directory of dist\…\Example Programs\Hardware Libraries
This directory contains compiled dll and lib files.

Directory of dist\…\Hardware Drivers
This directory contains Windows 95 and WinNT Drivres.

Directory of dist\…\Utility Programs
This directory contains three .exe programs

Directory of dist\…\Documentation
This directory contains the manual for the software developer's kit.

2.2 HARDWARE DRIVER INSTALLATION

A hardware driver handles the underlying I/O space access in the Developer's Kit routines. A service is used for Windows NT™ and a virtual device driver for Windows 95™.

Copy the appropriate file for the host platform from the Developer's Kit util subdirectory into the defined location.

Platform	File	Location
Windows NT™	WINRT.SYS	\windir\SYSTEM32\DRIVERS
Windows 95™	WRTDEV0.VXD	\windir\SYSTEM\VMM32

2.3 BOARD ADDRESS CONFIGRATION

Use the supplied registry utility bcreg.exe to configure the registry keys. The keys differ with the host OS. The utility will determine the correct operating system and create and/or modify the appropriate register keys.

The registry utility needs to know the base address set on the bc620AT hardware and an interrupt level, if any interrupt jumpers were set. The command syntax can be queried by executing the program with no parameters.

bcreg 0x300 0
In this example, the base address is set to hex 300 and the interrupt is ignored. A sample of the output from the command is shown below.

```
C:> bcreg 0x300 0
Using Windows 95
Using base address 0x300
Interrupt disabled
Registry info set-up
```

If this key were being set up for the first time, a message would be displayed indicating that the system must be rebooted before the changes will take effect.

2.4 TEST INSTALLATION

Use the compiled version of the example program 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 above.

If the device opens but “?????” are displayed instead of valid time values in the main window, the hardware interface was not configured correctly. Verify the base address of the installed bc620AT and use the registry utility in the utils subdirectory to reconfigure the driver. If the error persists, an address conflict may exist with some other piece of hardware in the system. Try changing the hardware address of the bc620AT and reconfiguring the driver before executing the example program again.

2.5 PROJECT CREATION

You can easily rebuild Bc620atDemoCpp.exe and bc620atClockCpp.exe by opening the corresponding project file with Visual C++ 5.0.
If you want to use bcutil.dll in your own MFC project, you may follow the instructions below:

1) Insert bcutil.lib into your project.
2) If building a new project similar to bc620atDemoCpp, you don't need to change the default settings of the project.
3) If building a new project similar to bc620atClockCpp, 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 "bcutil.lib.lib" to "Object/Library Module" edit box.

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

CHAPTER THREE
LIBRARY DEFINITIONS

3.0 GENERAL

The interface library provides functions for each of the programming packets supported by the bc620AT Time and Frequency Processor with the exception of the GPS packet "J." In addition, functions are provided to both read and write individual registers 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

Note: Library functions bcOpen and bcClose are not applicable for 16-bit applications.

bcOpen	
Prototype	int bcOpen (int devno);
Packet	N/A
Input Parameter	Device Number ***Note***: This value must be set to 0.
Returns	RC_OK on Success RC_ERROR on Failure
Description: This opens the underlying hardware layer. The developer's kit currently only supports one hardware device per application.	

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

bcGetByte	
Prototype	int bcGetByte (unsigned char page, int offset, unsigned char *value);
Packet	N/A

Input Parameter	page = bc620AT Page Register
	offset = 0 Based Offset of Requested Register
	value = 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.	

bcSetByte	
Prototype	int bcSetByte (unsigned char page, int offset, unsigned char *value);
Packet	N/A

Input Parameter	page = bc620AT Page Register
	offset = 0 Based Offset of Requested Register
	value = Pointer to Unsigned Char to Value to be Set

Returns	RC_OK on Success RC_ERROR on Failure
Description: Sets the contents of the requested register.	

bcReadTime	
Prototype	int bcReadTime (unsigned char *sout);
Packet	N/A
Input Parameter	unsigned char pointer to output string. This string will be filled with eight bytes corresponding to TIME0-TIME7. ***Note***: This array is NOT null terminated.
Returns	RC_OK on Success RC_ERROR on Failure
Description: Latches and returns time captured from the time registers.	

bcSetMode	
Prototype	int bcSetMode (unsigned char mode);
Packet	A
Input Parameter	unsigned char indicating requested operating mode. ***Note***: The following are defined in bcutil.h #define MODE_IRIG 0x00 #define MODE_FREE 0x01 #define MODE_1pps 0x02 #define MODE_RTC 0x03 #define MODE_GPS 0x04
Returns	RC_OK on Success RC_ERROR on Failure
Description: Sets the operating mode of the bc620AT.	

bcSetTime	
Prototype	int bcSetTime (char *day, char *hour, char *min, char *sec);
Packet	B
Input Parameter	char *day = Julian day number (Jan 1 = 001) [3 characters] char *hour = hour [2 characters] char *min = minute [2 characters] char *sec = second [2 characters] ***Note***: These are fixed length fields passed exactly as given to the bc620AT. It is not necessary to null terminate the arrays.
Returns	RC_OK on Success RC_ERROR on Failure
Description: Set the major time buffer.	

bcCommand	
Prototype	int bcCommand (int command);
Packet	C
Input Parameter	int command = requested command action ***Note***: The following are defined in bcutil.h #define CMD_WARMSTART 0x01 #define CMD_COLDSTART 0x02 #define CMD_JAM 0x03 #define CMD_NO_JAM 0x04 #define CMD_SYNC_RTC 0x05
Returns	RC_OK on Success RC_ERROR on Failure
Description: Sends command to the bc620AT.	

bcSetDac	
Prototype	int bcSetDac (int dacval);
Packet	D
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	F
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	G
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 one millisecond (or 10000 steps) requires first that the user disable jamsynchs. Refer to the hardware manual for more information.

bcSetTcIn	
Prototype	int bcSetTcIn (int format, int type);
Packet	H
Input Parameter	int format = time code format int type = modulation type of time code ***Note***: The following are defined in bcutil.h <u>format</u> #define TCODE_IRIG_A 0x00 #define TCODE_IRIG_B 0x01 #define TCODE_2137 0x02 #define TCODE_NASA36 0x03 #define TCODE_XR3 0x04 <u>type</u> #define TCODE_MOD_AM 0x10 #define TCODE_MOD_DC 0x11
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 (int which);
Packet	I
Input Parameter	int which = which clock source (internal | external) ***Note***: The following are defined in bcutil.h #define CLK_INT 0x00 #define CLK_EXT 0x01
Returns	RC_OK on Success RC_ERROR on Failure
Description: Sets the 10MHz clock source for the bc620AT.	

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 (int format);
Packet	K
Input Parameter	int format = time code format ***Note***: The following are defined in bcutil.h #define GEN_IRIG_B 0x00 #define GEN_IRIG_H_DC 0x01
Returns	RC_OK on Success RC_ERROR on Failure
Description: Sets the time code generator format.	

bcSetRTC	
Prototype	int bcSetRTC (char *year, char *month, char *mday, char *hour, char *min, char *sec);
Packet	L
Input Parameter	char *year = year (1980-2079)[4 characters] char *month = month (Jan = 1) [2 characters] char *mday = month day (e.g. Jan 1 = 01) [2 characters] char *hour = hour [2 characters] char *min = minute [2 characters] char *sec = second [2 characters] ***Note***: These are fixed length fields passed exactly as given to the bc620AT. It is not necessary to null terminate the arrays.
Returns	RC_OK on Success RC_ERROR on Failure
Description: Set the time in the Real Time Clock chip.	

Note: This does not effect the time in the time buffers unless the bc620AT is operating in RTC mode (Mode Three). The time in the RTC chip is initialized to Jan 1, 1900 each time the hardware is reset and this time is NOT used in any other mode of operation.

bcSetLocOff	
Prototype	int bcSetLocOff (int offset);
Packet	M
Input Parameter	int offset = hours from input time source. (-11 - +12)
Returns	RC_OK on Success RC_ERROR on Failure
Description: Programs the bc620AT to operate at an offset from UTC.	

Note: The function is only valid when the bc620AT is operating in GPS mode (Mode Four).

bcRequest	
Prototype	int bcRequest (unsigned char reqno, char *sout);
Packet	O
Input Parameter	unsigned char reqno = requested data packet char *sout = buffer to data packet requested. ***Note***: The following are defined in bcutil.h #define REQ_RTC_TIME 0x00 #define REQ_DAC_VALUE 0x01 #define REQ_LEAP_SEC 0x02 #define REQ_PROG_DATA 0x03 #define REQ_MOD_VER 0x04 #define REQ_YEAR 0x05
Returns	RC_OK on Success RC_ERROR on Failure
Description: Returns requested data packet from bc620AT.	

Note: Return packets two (leap seconds) and five (year) are only valid when the bc620AT is operating in GPS mode (Mode Four).

bcSetPath	
Prototype	int bcSetPath (int path);
Packet	P
Input Parameter	int path = requested path value (in lower 8 bits) ***Note***: The following are defined in bcutil.h #define PATH_DIAG_OFF (1<<0) #define PATH_DIAG_ON (0<<0) #define PATH_LEAP_ON (1<<1) #define PATH_LEAP_OFF (0<<1) #define PATH_JAM_OFF (1<<2) #define PATH_JAM_ON (0<<2) #define PATH_DISC_OFF (1<<3) #define PATH_DISC_ON (0<<3) #define PATH_ECHO_ON (1<<4) #define PATH_ECHO_OFF (0<<4) #define PATH_TIME_GPS (1<<5) #define PATH_TIME_UTC (0<<5) #define PATH_DC_MOVING (1<<6) #define PATH_DC_STATIC (0<<6) #define PATH_FMT_BIN (1<<7) #define PATH_FMT_BCD (0<<7)
Returns	RC_OK on Success RC_ERROR on Failure
Description: Sets path bits which modify default operation of the bc620AT. Refer to the hardware manual for more information on the use and effects of this function.	

Note: While this command works for all revisions of the bc620AT, some firmware versions return the path value incorrectly in request packet 0 - 3 (programmable data). Please contact the factory for a firmware upgrade if you encounter problems reading back path data with nibble values higher than nine. This does not affect operation of the device.

bcSetGain	
Prototype	int bcSetGain (int gain);
Packet	Q
Input Parameter	int gain = digital to analog converter value for disciplining internal oscillator independent of selected reference source.
Returns	RC_OK on Success RC_ERROR on Failure
Description: Modifies the internal oscillator frequency.	

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);
Packet	R
Input Parameter	int offset = hours from input time source. (-11 - +12)
Returns	RC_OK on Success RC_ERROR on Failure
Description: Programs the bc620AT time code generator to operate at an offset from UTC.	

Note: The function is only valid when the bc620AT is operating in GPS mode (Mode Four).