PC03XT
Developer's Kit

8500-0085

User's Guide

Rev. A

October, 1998

CHAPTER ONE
INTRODUCTION

1.0 GENERAL

The PC03XT Developer's Kit is designed to provide a suite of tools useful in the development of applications which access features of the Datum PC03XT Time Code Reader. This kit has been designed to provide an interface between the PC03XT 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 features of the PC03XT.
- Hardware driver for Windows NT™ and VxD for Windows 95™
- Example program, with source, utilizing the interface library.
- Console application to configure registry keys.
- Manual providing a library definition.

1.2 OVERVIEW

The Developer's Kit was designed to provide an interface to the PC03XT Time Code Reader in the 32-bit environments of Windows 95™ and Windows NT™ . The interface library is provided in object format suitable for linking with either Borland's Professional C compiler or with Microsoft Visual C++. Please contact the factory if you require an alternate format. The example program provides sample code which exercises 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. A resource file is included with interface dialogs to allow the operator of a program to set any configurable parameters for operating the PC03XT hardware. Applications programs developed using the 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. 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 program to test the system.

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

2.1 SOFTWARE DEVELOPER'S KIT INSTALLATION

Run the SETUP.EXE program to install 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 PC03XT hardware and an interrupt level, if any interrupt jumpers were set. The command syntax can be queried by executing the program with no parameters.

Example: -
A:\>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.

```
A:\> bcreg 0x300 0
Using Windows 95 or NT
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 "?????" or "ffffffff" are displayed instead of valid time values in the main window, the interface was not configured correctly. First try using the Time Code command to set up the board for proper operation. Verify the base address of the installed PC03XT 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 PC03XT and reconfiguring the driver before executing the example program again.

2.5 PROJECT CREATION AND STRUCTURE

For VC ++ 5.0 user. To rebuild the example programs from the supplied source files, go to the Pc03xtDemoCpp or Pc03xtTrayTimeCpp subdirectory, open the project file, pull down "Build" menu and click "Build All".
A directory structure is created in the specified location. This structure contains all required files to develop user applications. In addition, copies of the hardware driver files and configuration utilities are provided for redistribution with user-developed applications.

Directory of dist\Documintation
CHAPTER 1
CHAPTER 2

CHAPTER 3

Directory of dist\Utils

This directory contains hardware drivers and configuration utilities.

BCREG	EXE	Registry configuration utility
WINRT	SYS	Windows NT™ hardware layer
WRTDEV0	VXD	Windows 95™ hardware layer

Directory of dist\Sample Programs\Hardware Library

This directory contains the DLL's used in the development of both Pc03xt Demo and Pc03xt Tray Time example programs.

BC_IO	LIB
BC_IO	DLL

The above DLL is already copied to Windows system subdirectory by the installation automatically.

Directory of dist\Sample Programs\Pc03xtDemoCpp

This directory contains the files used in the development of the Pc03xt Demo example program.

BCTIME	C	Windows™ Time Display Routine
PC03XT	C	Hardware interface library source code
BC_IO	H	Hardware interface file
PC03XT	H	Hardware interface file
BC_ERR	H	Error return code definitions
BCCMD	H	Command Processor
BCTIME	H	Time Display
BC_IO	LIB	Interface Library for BC_IO.DLL

The above files are necessary to create a new project similar to the sample program and have to be inserted into the new project. Under 'Project' tab, choose 'Settings'. Select 'C/C ++' tab and in the 'Category' field, select "Precompiled Headers" and click on "Not using precompiled headers". The rest is typical MFC SDI files, among them, Dlg*.CPP and Dlg*.H that defines the dialog classes used in the sample program.

Directory of dist\Sample Programs\Pc03xtTrayTimeCpp

This directory contains the files used in the development of the Pc03xt Tray Time example utility.

DAT_BRD	C	
DAT_NET	C	
DAT_REG	C	
DAT_TYM	C	
PC03XT	C	Hardware interface library source code
SomeFunctions	C	Accessory functions
DAT_BRD	H	
DAT_NET	H	

DAT_REG	H	
DAT_TYM	H	
DAT_CFG	H	
DAT_CLK	H	
DAT_GPS	H	
PC03XT	H	Hardware interface file
BC_IO	H	Interface Library for BC_IO.DLL
BC_ERR	H	Error return code definitions
BC_IO	LIB	Interface Library for BC_IO.DLL

The above files are necessary to create a new project similar to the sample program and have to be inserted into the new project. Under ‘Project’ tab, choose ‘Settings’. Select ‘C/C ++’ tab and in the ‘Category’ field, select “Precompiled Headers” and click on “Not using precompiled headers”. Next, under “Link” tab, select “Customize” category and check “Force file output” box. Finally, under “Link” tab of “Project Settings”, select “General” category and add “wsock32.lib” to “Objet/Library Modules” edit box. The rest is typical MFC dialog application files, among them , PropertySheetPage.H/CPP files that defines the property sheet and property pages used in the sample which could be reused.

2.6 SYSTEM CLOCK UTILITY (PC03XT TRAY TIME)

This utility is designed to operate under Win95™ and Win NT™ v4.0. This is a system tray utility that will query the Pc03xt and set the system clock on a periodic basis.

1) Double click on the **“Pc03xtTrayTimeCPP.exe”** to install.

2) A small world icon will show up on the lower right portion of the desktop (where the clock appears), click on that icon and it will display a window (Datum Tray Time).

3) Click setup

4) Choose 1 minute or any other value for the interval update.

5) Check the Status: -

If it says *"Waiting for the board to acquire time"* then the time on the host computer is not synchronized to the Pc03xt time yet

If it says *"Set Clock OK"* then the synchronizing process is taking effect.

6) Drag the program into your startup group to have it run automatically at boot.

CHAPTER THREE
LIBRARY DEFINITIONS

3.0 GENERAL

The interface library provides access to all functions supported by the PC03XT Time Code Reader. 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 Operation and Technical manuals provided with the hardware.

3.1 FUNCTIONS

bcOpen	
Prototype	int bcOpen (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.	

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 (INT offset, unsigned char *value);
Packet	n/a
input parameter	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 (INT offset, unsigned char value);
Packet	n/a
input parameter	offset = 0 based offset of requested register value = unsigned char 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 14 bytes . NOTE: This array is NOT null terminated.
returns	RC_OK on success RC_ERROR on failure
Description: Latches and returns time captured from the board.	

bcSetTcFormat	
Prototype	int bcSetTcIn (UCHAR format);
Packet	none
input parameter	UCHAR format = time code format NOTE: The following are defined in the header file <u>format</u> #define TC_IRIG_A 0x00 #define TC_IRIG_B 0x01 #define TC_IRIG_G 0x02 #define TC_2137 0x03 #define TC_XR3 0x04 #define TC_NASA36 0x05 #define TC_DEF_B 0x06
returns	RC_OK on success RC_ERROR on failure
Description: Sets time code format.	

bcSetTcData	
Prototype	int bcSetTcIn (UCHAR channel, UCHAR direction);
Packet	none
input parameter	UCHAR channel = time code input channel (1-4) UCHAR direction = time code direction (forward or reverse) NOTE: The following are defined in the header file <u>channel</u> #define TC_CH_1 0x00 #define TC_CH_2 0x01 #define TC_CH_3 0x02 #define TC_CH_4 0x03 <u>direction</u> #define TC_FORWARD 0x00 #define TC_REVERSE 0x01
returns	RC_OK on success RC_ERROR on failure
Description: Sets time code input channel and direction.	

bcFIFONotEmpty	
Prototype	BOOL bcFIFONotEmpty (void);
Packet	n/a
input parameter	none
returns	TRUE if data available in FIFO. FALSE is FIFO is empty
Description: Check for availability of FIFO data for time and message packets.	

bcSetTrigger	
Prototype	INT bcSetTrigger (UCHAR sense, UCHAR onoff);
Packet	n/a
input parameter	unsigned char sense = external trigger edge (positive/negative) unsigned char onoff = enable or disable NOTE: The following are defined in the header file <u>sense</u> #define TC_TRIG_POS 0x00 #define TC_TRIG_NEG 0x01 <u>onoff</u> #define TC_TRIG_OFF 0x00 #define TC_TRIG_ON 0x01
returns	RC_OK on success RC_ERROR on failure
Description: Sets up external time capture trigger.	

bcSetPkt	
Prototype	INT bcSetPacket (UCHAR packet)
Packet	n/a
input parameter	unsigned char packet = type of packet for requests NOTE: The following are defined in the header file packet #define TIME_PKT 0x00 #define MSG_PKT 0x01
returns	RC_OK on success RC_ERROR on failure
Description: Sets up the type of data returned by a packet request.	

bcReqPkt	
Prototype	INT bcReqPacket (void)
Packet	n/a
input parameter	None
returns	RC_OK on success RC_ERROR on failure
Description: Requests a packet be loaded into the output FIFO. Note that up to 300 microseconds may elapse before data is ready to be read from FIFO. See the bcFIFONotEmpty routine for details on checking data availability.	

BcReadMsg	
Prototype	int bcReadMsg (unsigned char *sout);
Packet	n/a
input parameter	unsigned char pointer to output string. This string will be filled with 15 bytes . NOTE: This array is NOT null terminated.
returns	RC_OK on success RC_ERROR on failure
Description: Latches and returns msg packet.	

BcResetFIFO	
Prototype	int bcResetFIFO (void);
Packet	n/a
input parameter	None
returns	RC_OK on success RC_ERROR on failure
Description: Resets the output FIFO, removing any pending data.	

bcResetBrd	
Prototype	int bcResetBrd (void);
Packet	n/a
input parameter	none
returns	RC_OK on success RC_ERROR on failure
Description: Resets the board. NOTE: This command is not required for normal operation. Be sure to understand the effects of this command before using.	

bcResetInt	
Prototype	int bcResetInt (void);
Packet	n/a
input parameter	none
returns	RC_OK on success RC_ERROR on failure
Description: Resets the interrupt flip-flop to clear a pending interrupt.	