# PM2355

# FRAME RELAY PORT CARD

# DEVELOPMENT KIT

# PLATFORM

**RELEASED**

**ISSUE 2: JANUARY 2001**

## REVISION HISTORY

| Issue No. | Issue Date | Details of Change |
|---|---|---|
| Issue 2 | January 2001 | Minor updates to directory structure Figure 7. and Table 3 |
| Issue 1 | December 2000 | Document created |

## CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# 1 INTRODUCTION

The Frame Relay Port Card Development Kit Platform is the hardware platform on which the PMC-Sierra Frame Relay Port Card Development Kit board and application software can be tested and verified. The purpose of this development platform is to help PMC-Sierra's customers in reducing their application software development cycle.

The Frame Relay Port Card Development Kit consists of:

- Frame Relay Port Card Development Kit Board
- Frame Relay Port Card Development Kit software and documentation on the accompanying CD-ROM.

Please note that:

- The Pentium-based PC is not supplied by PMC-Sierra. The PC platform and its hardware and software configuration are the user's responsibility.
- The WindRiver Tornado 2 integrated development environment is not supplied by PMC-Sierra. If the user wants to continue development on the VxWorks OS it is the user's responsibility to purchase and configure the Tornado 2 development environment.
- The source code on the Frame Relay Port Card Development Kit Platform CD-ROM is distributed "as is". There is no backward compatibility with the latest software releases and the distributed source code shall be used for informational purposes only.

## 1.1 Purpose

This document provides a detailed software specification for the Frame Relay Port Card Development Kit Platform.

## 1.2 Scope

This document describes hardware and software requirements for the Frame Relay Port Card Development Kit Platform without going into the specifics of the PMC-Sierra Frame Relay Port Card Development Kit board. It describes the Frame Relay Port Card Development Kit Platform in the level of detail required to

configure and run the system. This document also contains a detailed software installation procedure.

## 1.3 Application

This document is designed for embedded systems development based on the functionality of the PMC-Sierra devices included in this design, namely the PM7384 FREEDM™-84P672, PM8315 TEMUX, PM5342 SPECTRA-155, and PM4314 QDSX.

## 1.4 Documentation

The following documents are included on the distribution CD-ROM.

**Table 1 Reference Documents on Distribution CD-ROM**

| PRODUCT NUMBER | DESCRIPTION | DOCUMENT NUMBER |
|---|---|---|
| PM7384 | FRAME ENGINE AND DATALINK MANAGER 84P672 (FREEDM™-84P672) Data Sheet | PMC-1990445 |
| PM7384 | FREEDM-84P672 Programmer's Guide | PMC-1990715 |
| PM7384 | FREEDM-84P672 Revision C Device Errata | PMC-2000953 |
| PM7384 | Answers to Frequently Asked Questions Regarding the FREEDM-32P672, FREEDM-84P672, FREEDM-32A672 and FREEDM-84A672 | PMC-2001119 |
| PM7484 PM8315 | Density Solutions with TEMUX and FREEDM-84 | PMC-1991202 |
| PM7384 | FREEDM-84P672 Frame Engine and Datalink Manager Short Form Data Sheet | PMC-1991024 |
| PM7384 | FREEDM-84P672 PCI Bus Utilization and Latency Analysis | PMC-1990863 |
| PM8315 | High Density T1/E1 Framer with Integrated VT/TU Mapper and M13 Multiplexer Telecom Standard Product Data Sheet | PMC-1981125 |
| PM8315 | High Density T1/E1 Framer with Integrated VT/TU Mapper and M13 Multiplexer Telecom Standard Product Register Descriptions | PMC-1900495 |
| PM8315 | Technical Overview of the High Density T1/E1 Framer with Integrated VT/TU Mapper and M13 Multiplexer Telecom Standard Product | PMC-1981141 |
| PM8315 | PM8315 TEMUX High Density T1/E1 Framer with Integrated VT/TU Mapper and M13 Mux Short Form Data Sheet | PMC-1971264 |
| PM8315 | Frame Relay Port Card Reference Design | PMC-1990533 |
| PM8315 | TEMUX Programmer's Guide | PMC-1991268 |
| PM8315 | Answers to Frequently Asked Questions Regarding the TEMUX | PMC-2000800 |
| PM5342 | SONET/SDH Payload Extractor/Aligner Telecom Standard Product Data Sheet | PMC-1970133 |

| | | |
|---|---|---|
| PM5342 | SONET/SDH Payload Extractor/Aligner Errata | PMC-2001737 |
| PM5342 | Suggestions for SPECTRA-155 PECL Terminations | PMC-1970938 |
| PM4314 | QUAD T1/E1 Line Interface Device Telecom Standard Product Data Sheet | PMC-1950857 |
| PM4314 | QUAD T1/E1 Line Interface Unit Short Form Data Sheet | PMC-1941034 |

# 2 GLOSSARY

| | |
|---|---|
| FRPC | Frame Relay Port Card |
| HOST | Used interchangeably with the Remote Command Machine |
| Target Machine | Hardware platform where FRPC Development Kit Board resides. |
| BSP | Board Support Package |
| OS | Operating System |
| BIOS | Basic Input Output System |
| I/F | Interface |
| S/W | Software |
| H/W | Hardware |
| APS | Automatic Protection Switching |
| Freedm | FREEDM™-84P672 |
| Spectra | SPECTRA-155 |

# 3 GENERAL DESCRIPTION

## 3.1 FRPC Development Kit Platform Architecture

The full development platform consists of two basic components:

- Target Machine
- Remote Command Machine

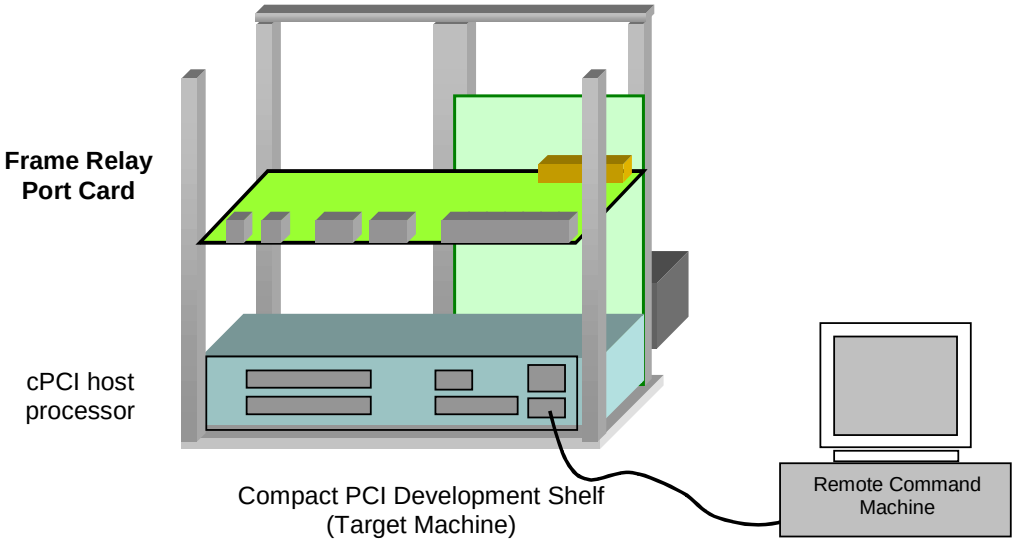


**Figure 1 Frame Relay Port Card System Configuration**

The target machine acts as the hardware support for the PMC-Sierra FRPC Development Kit Board.

To extend the testing and development capabilities of the basic system, the Remote Command Machine has been added. The communication link is serial RS232. The Remote Command Machine runs user applications (such as HostApp), simple terminal applications, or is used with some more powerful tools (such as TCL) for testing and development that is not supported on the Target Machine.

In order to establish communication with the target machine, the user must connect the remote and target machines via an RS-232 cable.

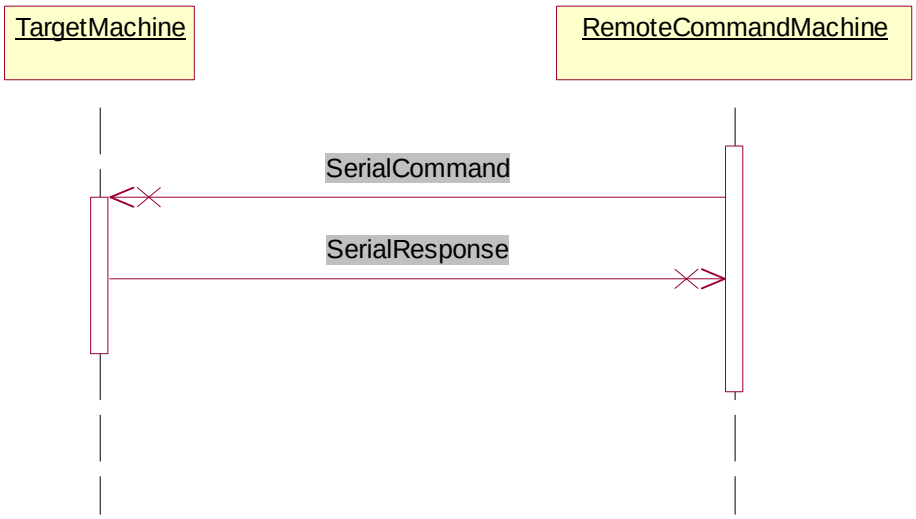


**Figure 2 Communication between command and target machines**

Using a serial connection, the user can send a predefined set of commands (supported by the application software supplied by PMC-Sierra or the user) and have the commands executed on the target system. The application software will send responses back to the Remote Command Machine.

# 4 TARGET MACHINE HARDWARE CONFIGURATION

The following represents the minimum hardware requirements for the target system:

- CompactPCI chassis with Intel Pentium based single board computer (minimum 233MHz). System was developed and tested on CPV5000 SBC.
- Two available serial communication ports (required for communication with the Remote Command Machine and microprocessor controlled part of the system) capable of sending and receiving data at a minimum of 115200 bps and 38400 bps respectively.
- Standard 1.44 MB Floppy Disk Drive
- IDE Hard Disk Drive. Recommended size is 2.1GB.
- Keyboard
- VGA Video Adapter
- VGA Compatible Monitor

**Network card is optional**. PMC-Sierra does not provide network support software for the target platform. It is up to user to add networking capabilities to the target system.

**Video Graphic Capabilities**. There is no requirement for any graphical capabilities of the video card on the target machine since the system will run in text mode.

# 5 SOFTWARE

## 5.1 Target Application – Design

The purpose of this section is to introduce specific software design concepts that will help the user understand the software related discussions later in the document, while also providing basic knowledge to those users who want to add their own code to the existing application.

The application code may be divided into four major software layers, as shown in Figure 3. These layers are:

1. Platform Support Layer
2. Framework
3. Device Interface Layer (Application specific layer)
4. Device Drivers

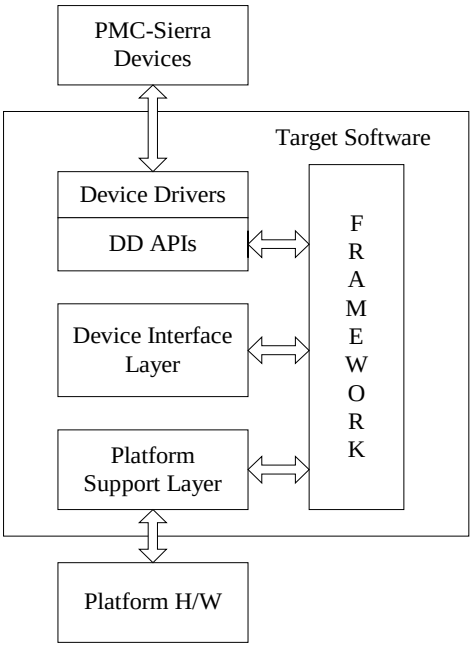


**Figure 3 Target Software**

## 5.2 Framework

The framework is a software wrapper that provides infrastructure to the rest of the application code. The framework is designed and implemented to be platform and application independent. Together with platform support and application specific modules, the framework is incorporated in the application executable (e.g. frpc.out for VxWorks platform or main.s19 in the case of MC68340).

Infrastructure services provided by the framework consist of the following functions:

- Scheduling.
- Event processing and notification.
- Timer services.
- Command parsing and distributing.
- Data block transfer and processing.
- Inter and intra board communication services.

In the FRPC application, the same framework code runs on both the VxWorks and MC68340 platforms.

The framework sources are located on the \dev\codebase directory of the distribution CD.

## 5.3 Platform Support Layer

Platform support code consists of low level s/w routines that provide the framework with necessary porting functions. The standard set of platform support routines include: Clock handler, Semaphores, Serial Port drivers and various interrupt handlers.

Source files that belong to this layer of code are located on the distribution CD under the \dev\platform directory.

## 5.4 Device Interface Layer

This is application specific layer of the code that deals with the PMC-Sierra devices present in the system. Typically there is one device interface per each type of PMC-Sierra device, hence there is one handler for the FREEDM™-

84P672device, one for TEMUX, SPECTRA-155 etc. The device interface modules provide command and event handlers and other higher level routines needed to implement application requirements.

The device interface files can be found in the \dev\if directory.

## 5.5 Device Driver Layer

For an in depth description of the device drivers, please refer to the corresponding device driver specification provided on the distribution CD.

In the software implementation of the FRPC, only the FREEDM™-84P672 device driver is being used. This is due to the fact that reference design drivers are implemented for VxWorks platforms only and FREEDM™-84P672 is the only device operating in this environment. Low level routines for other PMC-Sierra devices provide only basic I/O routines and are available through the platform support code.

The FREEDM™-84P672 device driver code can be found in the \dev\drv\freedm-ng\ directory.

## 5.6 Software Device Interface Concept

The concept of the software device interface, as implemented in this design represents a software abstraction for a physical device (e.g. chip, COM ports, timers) or an abstraction for a s/w module that provides some functionality to the rest of the system (e.g. board, command parser, loader). Our system consists of two target platforms:

- Board0 – Master Target. CompactPCI based Pentium platform that executes the framework and the application code under VxWorks OS.
- Board1 – Node Target. Microprocessor MC68340 platform that executes the framework with the application code.

As mentioned previously, the PMC-Sierra devices present on the Frame Relay Port Card are the FREEDM™-84P672, three TEMUX devices, two SPECTRA-155s and the QDSX. The FREEDM™-84P672 device exists in the PCI address space and is controlled by the Pentium processor through the PCI bus. The other PMC-Sierra devices (TEMUX#1,2,3, QDSX, working and protection SPECTRA-155s) are memory mapped in the microprocessor address space and can only be controlled by the on board microprocessor MC68340.

Communication between the host application and the target system is established through one of the host COM ports and COM1 on the master target. Communication between master target (Board0) and the node target (Board1) runs over COM2 on the master target and COM1on the node. Commands that the host sends to the node target are first processed on the master target and then streamed to Board1. Responses that the node sends to the host application are received by the master first and then dispatched to the host.

From a software perspective, all the devices are grouped according to the board in which they logically belong. For example, the FREEDM™-84P672 device is physically present on the FRPC but it can only be accessed through the VxWorks board, hence it belongs to Board0. Conversely, all other devices (TEMUX, Working SPECTRA-155 etc) belong to Board1 since they are controlled using the MC68340. The following figure shows FRPC system as viewed by the software.

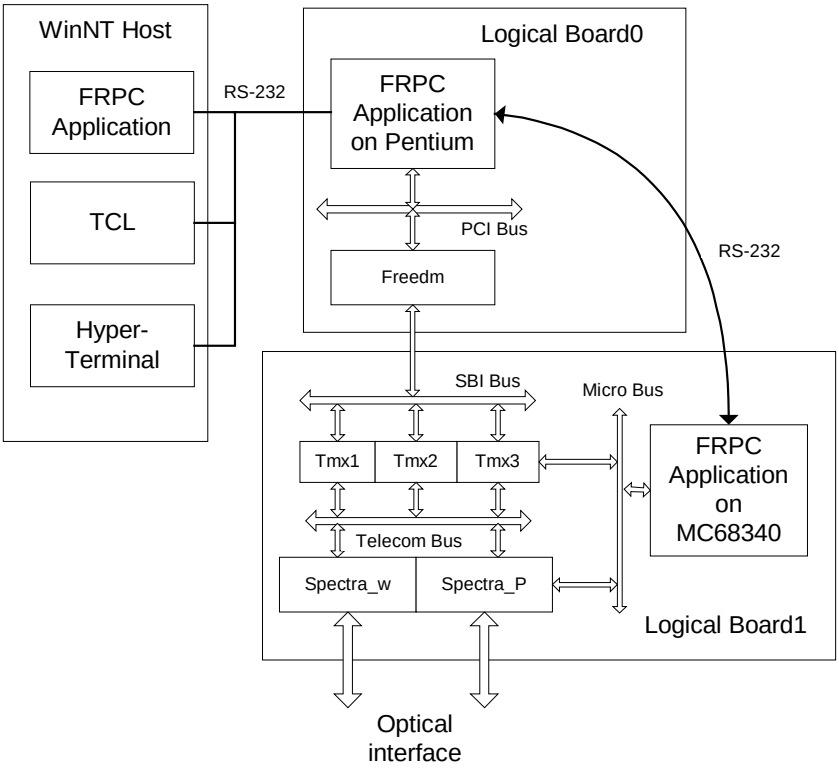


**Figure 4 FRPC Elements and Connections**

The following software devices are located on logical board0:

- Board0 – VxWorks target. This abstraction is needed since file and block downloads and block executions go through it.

- Board1 – Device name that represents interface for the next logical board in the chain. Logical boards in our system are chained using serial interfaces. Access to a remote logical board (e.g. logical board1) goes through logical Board0.
- Com1 – Communication port connecting target machine to the host (remote command machine). Port is set to run at 115Kbps.
- Com2 – Communication port connecting board0 to board1. Port is set to run at 38.4Kbps. Data rate is set to a maximum matching speed.
- Cmd – command parser interface.
- Freedm – freedm device handler.

The following software devices are located on logical board1:

- Board1 – 68K target
- Com1 – Communication port connecting board1 to board0. Port is set to run at 38.4kbps.
- Cmd – Command parser interface.
- Temux1, Temux2, Temux3 – The TEMUX device handlers.
- Spectra_W, Spectra_P – The SPECTRA-155 device handlers.
- Micro – Microprocessor device needed to control Automatic Protection Switching.

### 5.6.1 Command Handling

Handlers for every logical device in the system register the set of commands it supports to the framework Registration takes place at the handler's initialization time. The framework keeps a per device list of commands and whenever new command line is received, the framework tries to match command name with names from the command list for that particular device. If a valid command is found, the framework invokes the registered handler. Appendix A gives the complete list of commands with explanations.

### 5.6.2 Downloadable Data Blocks

Data blocks are software structures implemented for passing larger chunks of data between nodes in the system. Block transfer guarantees consistency of delivered data, yet does not guarantee delivery. No assumptions on the nature of the transferred data are made. A block may be loaded to target system memory, or sent to a file if the system has a hard drive. Block transfer to remote nodes is also implemented.

The primary use of blocks is as follows:

- Downloading complete script files that would execute on the target machine (as oppose to executing scripts on the host and sending commands one at the time). Commands contained in a block can be executed by issuing ‘exec’ command (e.g. exec boardN block blockName param). Once downloaded, data block may be executed any number of times with a different device name as a parameter. Scripting language provided locally on target is explained in the next section.

- Blocks are used when downloading files from host to target. This capability proves to be very useful for uploading target executable code built on host.

- Blocks are also used to enable the host application to transmit large messages via the optical interface.

### 5.6.3 Scripts in Blocks

The scripting processor interprets the full command set, listed in Appendix A. Scripts may only contain command lines. Comments and blank lines are not permitted (an empty line is considered as an end of file marker).

Examples:

write temux1 01 48

write temux1 1200 4A

### 5.6.4 Repetitive Commands

This feature helps make smaller and more efficient scripts. Every repetitive command maintains an iteration counter that can be used in expressions in that

line. The symbol used in expressions to refer to the iteration counter is '%' (during execution, % is replaced by the value of the counter).

The syntax is shown below:

*Command devName rep[start,stop,step] <expression-1> <expression-2>…<expression-N>*

where:

*Command* – Device generic command such as read or write.

*devName* – Represents one of the recognized device names, such as temux1, temux2, temux3, spectra_w, spectra_p, qdsx, or freedm.

*rep[start,stop,step]* – Represents the starting and stopping value of the iteration counter (denoted by %), as well as the counter increment.

<*expression-N*> - simple mathematical expression that uses basic operations "+ - * / ". Space character is argument separator and may not be part of an expression.

Counter parameters start/stop/step are decimal digits while other parameters follow the rules described in Appendix A. The iteration counter starts at START, incrementing by STEP (step may be negative) until the counter reaches the STOP value. Arguments in the repetitive command are separated with spaces. Since *rep[x,y,z]* and *expressions* are considered to be arguments, they may not contain space characters.

**Examples:**

```
/*writes 00 to TEMUX#1 registers 0x9c, 11c,…141c
write temux1 rep[1,28,1] 001c+80*% 00

 /*writes C0 to TEMUX#1 registers 1240, 1242,…, 1252
write temux1 rep[0,12,2] 1240+% C0

/*  reads from TEMUX#2 registers: 0x41, 0x81, …,0x181 */
read temux2 rep[1,6,1] 40*%+1

/*  writes 2 to FREEDM™-84P672 registers 0x01, 0x11, 0x21, …0x91 */
write freedm rep[0,9,1] 10*%+1 2
```

## 5.7 Host Applications

Since communication to the target machine is established using standard RS-232 protocol, any host application supporting this protocol may be used.

As a part of this SDK, the Windows based application “HostApp” is provided. It is a custom built GUI to showcase the functionality of the Frame Relay Port Card. The majority of this section is dedicated to introducing its features.

During the development phase of this product, other standard Windows packages had also been used for various reasons. One of the most frequently used applications was TCL. The details on how to set up the TCL environment are provided the paragraph 5.8.

### 5.7.1 HostApp Description

HostApp is a Windows based GUI that enables users to conveniently send commands and graphically present the results.

Standard Windows file utility My Computer may be used to locate and start execution of the application file HostApp.exe. Figure 5 depicts the application main window.

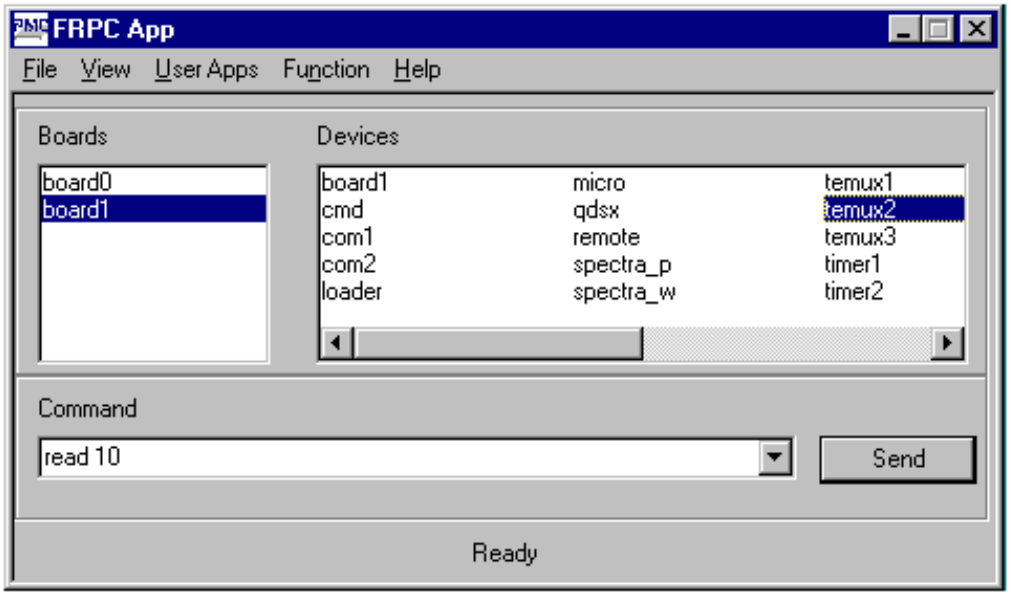


**Figure 5 HostApp Main Window**

In order to view the results of device interaction (for example, a read), the user must open the Terminal Window, by selecting the Terminal Window option in the View menu.

As shown in the figure above, device Temux2 on logical Board1 is selected. If the user wants to execute any of Temux2's commands (e.g. read from register 0x10) it is sufficient to enter "read 10" and the command line will be automatically completed with the device name and the reference to the appropriate board (command is sent upon clicking the Send button).

For example, to read temux1 register 01, select board1 in the Boards selection pane, and temux1 in the Devices selection pane. Type "read 01" in the Command dialog box, and click Send. The results of this read operation can be viewed in the Terminal Window.

Automatic device name insertion can be avoided by typing in the full command line. Note that to use this alternative method, the appropriate board (example, board0 for FREEDM-84P762 access, board1 for Temux1 access) must be selected in the Boards selection pane. In addition, board0 or board1 must be selected in the Devices selection pane. Then type the full command (eg read temux1 01) in the Command dialog box.

The following is a list of the features supported by this application:

- List of devices in the system grouped by boards (as shown on Figure 5)
- Command entry that relates command to selected board and device.
- Terminal Window – displays data received over the communication port.
- Debug Window – host application control and debugging information.
- Block and file load and execute – menu driven aid for downloading and executing blocks and files.
- Message control. Enables message composing and transmitting through the selected channel, receive and transmit message display setup, loop-back control and statistics display. The statistics window will be periodically updated.
- APS monitor. Automatic Protection Switching control command activates one of the available SPECTRA-155 devices and initializes signal monitoring. In the case of signal loss on the active SPECTRA-

155, APS automatically switches to the protection SPECTRA-155. The host monitor shows real-time information on the currently active device.

- Real-time register Watch feature.
- System Self-test

### 5.7.2 HostApp Functions

This section describes the functionality provided by the Frame Relay Port Card HostApp. The main application window is shown in Figure 5.

Drop Down menus:

**File**

Open – function to open system description file App.def.

Properties – communication port setup.

Exit

**View**

Debug – host application debug window.

Terminal – terminal receiver window. Shows all characters received from the target system.

**Function**

Block

Load – file menu to select file(s) to be downloaded as a script block to the memory on the board selected in the board window.

Execute – block selection menu. Dir command will first be executed to display the list of loaded blocks. One or more blocks from the displayed list may then be selected for execution.
Dir – displays the list of blocks loaded on the selected board.
Delete - block selection menu. Dir command will first be executed to display the list of loaded blocks. One or more blocks from the displayed list may then be selected for deletion.

File - these functions are available for boards containing a disk drive.

Load – file menu to select file(s) to be downloaded to the disk drive of the board selected in the board window.

Dir – displays the list of files present on the current directory of the selected board. The list will include "." and "..' symbols for current and parent directories.

Message

Composer – Edit window for composing outgoing message. User has to specify the transmission channel and may set repeat count so that the same message gets transmitted more then once. This feature is intended for demonstrating APS capability. Composed message will first be downloaded to the target platform, and then submitted to the FREEDM™-84P672 device driver for transmission.

Settings

Rx Channel – Packet display settings for the selected receive channel. Received data may be displayed on either the host or target monitor. If host display mode is specified, a read only display window will be displayed. The RxChannel field at the bottom of that window is also read only, and editing it will have no effect on the channel displayed. To suppress data display, set toggle switch to off.

Tx Channel – Packet display settings for the selected transmit channel.

Loop-back – Setup for different loop-back modes. The FREEDM™-84P672 device driver supports channel loop-back, diagnostic channel loop-back, and link loop-back.

Packet Statistics

Channel – Requests display of the packet statistics for the selected channel. If the periodic field is set, channel statistics will periodically be updated.

Link – Requests display of the packet statistics for the selected link. If the periodic field is set, link statistics will periodically be updated.
Other - Requests display of other available statistics. If the periodic field is set, various other statistics will periodically be updated.

Register Watch – Window that displays and periodically updates register values. Values are updated every 5sec, unless a different

value for the period is given. All watches on the same device are parts of the same watch interface, hence a period change for one of the monitored registers will affect the others. Watches for each device are independently updated.

Self-test – This function is primarily added to enable fast system testing. Self-test configures the path for the packet loop-back at the optical interface. The test will be successful if the packet transmitted over the specified channel is received unchanged on the same channel. The user is prompted to select a file (generally Setup.ini) that contains the list of blocks that are to be loaded and executed on the target platforms. Blocks are processed in the order they appear in the file. Upon execution of the last block, the system waits a couple of seconds to receive notification that test is OK and displays appropriate message.

Note: Test requires that one fibre optic cable be connected between the optical receive and transmit outputs of the Working SPECTRA-155.

Automatic Protection Switching (APS) – A function to monitor the currently active SPECTRA-155. Switching between working and protection devices is done automatically as soon as interrupt handling for both SPECTRA-155s is enabled.

## 5.8 TCL Command Processor

In order for TCL to communicate with the target system, the user needs to setup various serial port parameters to the values the target computer expects. The current configuration is 115200bps, 8bits, 1stop bit.

There are no special requirements for the commands the TCL processor sends to the logical Board1 as long as the syntax described in Appendix A is used.

Note that commands directed to logical Board1 first have to be recognized by Board0's parser prefix and then sent to Board1. In order for this parser to recognize commands sent to Board1, "send remote" has to be placed at the beginning of each command line.

**Example:**

```
/*command to read from register 0x15 on temux2 */
send remote read temux2 15
```

## 5.9 HyperTerminal

All the rules that the TCL processor needs to follow in order to communicate with the target system apply for Hyper-Terminal as well. For details on how to set Hyper-Terminal communication parameters, please refer to Appendix I.

# 6 INSTALLATION PROCEDURE

The software for the FRPC consists of four major parts. VxWorks Kernel files and Pentium BSP boot floppy disk, application code running on the VxWorks target, application code running on the Motorola MC68340 microprocessor (located in EPROM), and the host application (typically HostApp).

## 6.1 Software on the Remote Command Machine

Paragraph 5.7 details requirements for the host application software. This section explains installation procedure for the HostApp host application.

### 6.1.1 Files

The FRPC host application, HostApp, consists of 5 files. Files are located on the distribution CD in the **\Dev\Platform\Win** directory. These files are: HostApp.exe, HostApp.ini, App.def and two Board.h files.

**HostApp.exe** – this file contains application executable code. Its main purpose is to make command entry easier and more intuitive, while providing necessary display support so that the user knows what is happening on the target system. The HostApp program is designed so that it is easy to add more application specific functionality.

**HostApp.ini** – The standard ini file that contains application configuration information (com-port settings, etc.) and user setup (location of the target system definition file, recent selections, etc.). This file is automatically created and updated by the application program.

**App.def** – A target system definition file (ASCII format) with references to board description files. Includes file(s) board.h used to build the target applications by reading board configuration information.

**Board.h** – One per target board with code executing on it. This is a standard .H file that contains a list of software devices on this board. The term “software device” is explained in the section “FRPC Software Devices”.

#### 6.1.1.1 Installation

The host code installation process is quite simple. Application executable HostApp.exe, ini file HostApp.ini. Definition file App.def may be located anywhere in the system, since it can be located using the File/Open menu. Once found, the

location of this file will be kept in the initialization file hostApp.ini. If definition file is located in the same directory with the previous two, it will be automatically opened. H files could be located elsewhere as long as app.def points to the correct location.

### 6.1.2 Blocks

Blocks are files containing scripts to be executed on the target system (details are given in the Paragraph 5.6.2 and the Appendix B). The purpose of the blocks provided with this SDK is in enabling users to quickly setup target systems. In order to execute block it needs to be downloaded from the host to the target. Though blocks may be downloaded from the distribution CD it is recommended that the user copy them to a dedicated directory on the host computer. Normally blocks reside in the target development sub tree **\dev\app\frpc\blocks** but this is not a requirement since the standard Windows file menu is provided to locate them in the run time.

## 6.2 Target Application Code: VxWorks Platform

### 6.2.1 Target Machine Software Configuration

The software that is required for system startup and operation must be located on the target machine hard drive (with the exception of the boot disk).

Since VxWorks is a real-time operating system, there are not too many hard disk formats that this OS can support. One that users might find easy to install and use is FAT-16. This is also a format that operating systems like MS-DOS and MS Windows can recognize.

It is assumed that the user is using a FAT-16 disk format.

### 6.2.2 Hard Disk Formatting

In order to use the hard drive and install the new software, the hard disk must first be formatted.

The fastest way to do this is to use MS-DOS version 6 or higher, and follow the MS-DOS installation procedure from the Microsoft Corporation DOS User’s Manual.

The MS-DOS operating system has been chosen because of its fast boot time, fast installation time and small size. MS-DOS does not have to be installed**,** but

the user may find some MS-DOS external commands useful during the development process (such as the DOSKEY command that repeats the commands on the command prompt). MS-DOS is required for automatic software installation on the target system.

The file system on the hard drive must be FAT-16.

VxWorks only recognizes the primary FAT-16 partition. The other partitions (if installed) may be used for other purposes.

### 6.2.3 Target Platform Directory Structure

A minimal fixed directory structure has to be maintained on the target machine as shown in Figure 6.

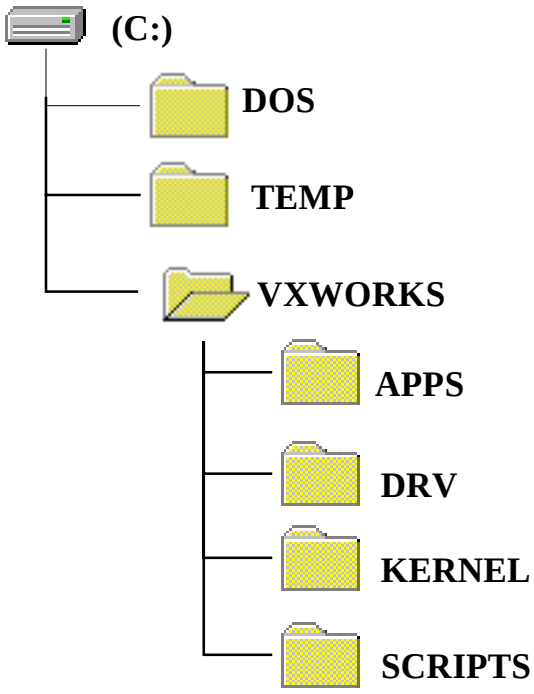


**Figure 6 Target Platform Directory Structure**

**Table 2 Directory Structure on the Target Machine Hard Drive**

| Directory Item | Comment |
|---|---|
| (C:) | Root directory of the FAT-16 file system. |

| Directory Item | Comment |
|---|---|
| /DOS | Directory where the Disk Operating System (DOS) might be installed. |
| /TEMP | Temporary directory. |
| /VXWORKS | Root directory of the VxWorks operating system. |
| /VXWORKS/APPS | Directory where application code is stored. |
| /VXWORKS/DRV | Directory where PMC-Sierra software chip drivers and supporting binaries should be installed. |
| /VXWORKS/ KERNEL | Directory where VxWorks kernel and VxWorks supporting binaries should be installed. |
| /VXWORKS/ SCRIPTS | Directory where VxWorks shell script files should be installed. The most important script is `startup.vxs` which executes automatically during the system startup. This script must be located in this directory. |

The VxWorks directory structure will be created during automatic software installation.

### 6.2.4 Software Installation on the Target Platform

Once the target platform has been prepared properly, the binaries supplied by PMC-Sierra have to be installed onto the target platform hard drive, and the source code supplied by PMC-Sierra has to be installed onto the software development platform. For the automatic installation, a MS-DOS CD-ROM driver must be installed properly on the target machine prior to installation. If the user wishes to set up the target platform manually, a CD-ROM is not required.

### 6.2.5 Making the VxWorks Boot Floppy

The target system boots up off the VxWorks boot floppy. All the necessary files and utilities necessary to make a VxWorks boot floppy are located on the FRPC distribution CD-ROM.

The following steps should be taken in order to make a bootable VxWorks floppy:

1. Put the FRPC distribution CD-ROM into your host machine CD-ROM drive. Open Windows Explorer and change the CD-ROM directory to \DEV\PLATFORM\VXWORKS\BSP_PENTIUM.

2. Put the blank floppy disk into your floppy drive and label it as “VxWorks Hard Disk Boot”. Make sure the floppy disk is not write protected.

3. Double-click on the mvxwbd.bat file. This action will open an MS-DOS window and start making the bootable VxWorks floppy.

4. Once the MS-DOS window has been closed, the VxWorks bootable floppy disk is ready and the user can remove it from the host machine floppy drive. The file bootrom.sys should exist on the bootable floppy.

Note: the MS-DOS window may not close automatically after the MS-DOS task has been finished. In that case the MS-DOS box will have “Finished…” at its top left corner. The user should close the window manually.

### 6.2.6 Software Installation Steps

The assumption is that the target VxWorks platform is equipped with a disk drive.

The first step is to format the hard drive, and install MS-DOS on it. MS-DOS is used primarily to simplify the file transfer procedure. It is recommended that MS-DOS version 6.21 be installed.

Once the MS-DOS system is installed, we need to create the VxWorks directory structure:

```
cd \ - position to the root directory
md vxworks
cd vxworks
md kernel
md scripts
md apps
```

Files necessary for the VxWorks target system installation are located in the appropriate directories starting from **\Dev\Platform\VxWorks**. If the target system is not equipped with the CD-ROM drive the user needs to copy files to floppy disk first. Now add the provided files to the appropriate directories:

```
cd \vxworks\kernel
copy a:\vxworks .
copy a:\vxworks.sym .
cd ..\scripts
copy a:\vxconfig.txt
cd ..\apps
copy a:\frpc.out .
```

Everything is set up now. Place the VxWorks boot floppy disk into the floppy drive and reboot the system. When the boot-up sequence finishes, the FRPC application will be loaded and automatically started.

## 6.3 Target Application Code: MC68340 Microprocessor Platform

The complete code that runs on the microprocessor will be loaded in the EPROM provided with the board, so no installation is needed. For the instructions on how to rebuild application code for the microprocessor platform, please refer to Appendix D.

## 6.4 PMC-Sierra Development Kit Distribution CD-ROM

The Development Kit distribution CD-ROM contains all the necessary documentation and software required for the target platform. Once installed the target system is fully functional and ready to run.

**Table 3 Contents of the Development Kit Distribution CD-ROM**

| Directory Item | Comment |
|---|---|
| (D:) | Root directory of the distribution CD-ROM. |
| /DOC | Location of all the required documentation. |
| /DEV | Root directory of the distributed software. |
| /DEV/APP/ FRPC | Root directory for the application specific FRPC software. |
| /DEV/APP/FRPC/ 68340 | Application specific code running on MC68340 microprocessor. |
| /DEV/APP/FRPC/ BLOCKS | Script files downloadable to one of the hardware platforms. |
| /DEV/APP/FRPC/ VxWorks | Directory where the VxWorks operating system and its supporting binaries is located. |
| /DEV/ CODEBASE | Framework source code. Application and platform independent. |
| /DEV/DRV | Root directory containing device driver code. |
| /DEV/DRV/ FREEDM_NG | The FREEDM-84P672 device driver source code. |
| /DEV/DRV/IF | Device and application specific source code. Commands and other device specific functions are located in this directory. |
| /DEV/ PLATFORM | Platform support code. |
| /DEV/ PLATFORM/68340 | MC68340 platform support code (semaphores, clocks, com handler). |
| /DEV/ PLATFORM/ VxWorks | VxWorks platform support code. |

| Directory Item | Comment |
|---|---|
| /DEV/ PLATFORM/ VxWorks/KERNEL | VxWorks kernel files. |
| /DEV/ PLATFORM/ VxWorks/BSP | Location of the add-on source code for the Tornado 2 development system Board Support Package (BSP). |
| /DEV/ PLATFORM/WIN | Host application code running on Windows platform. |

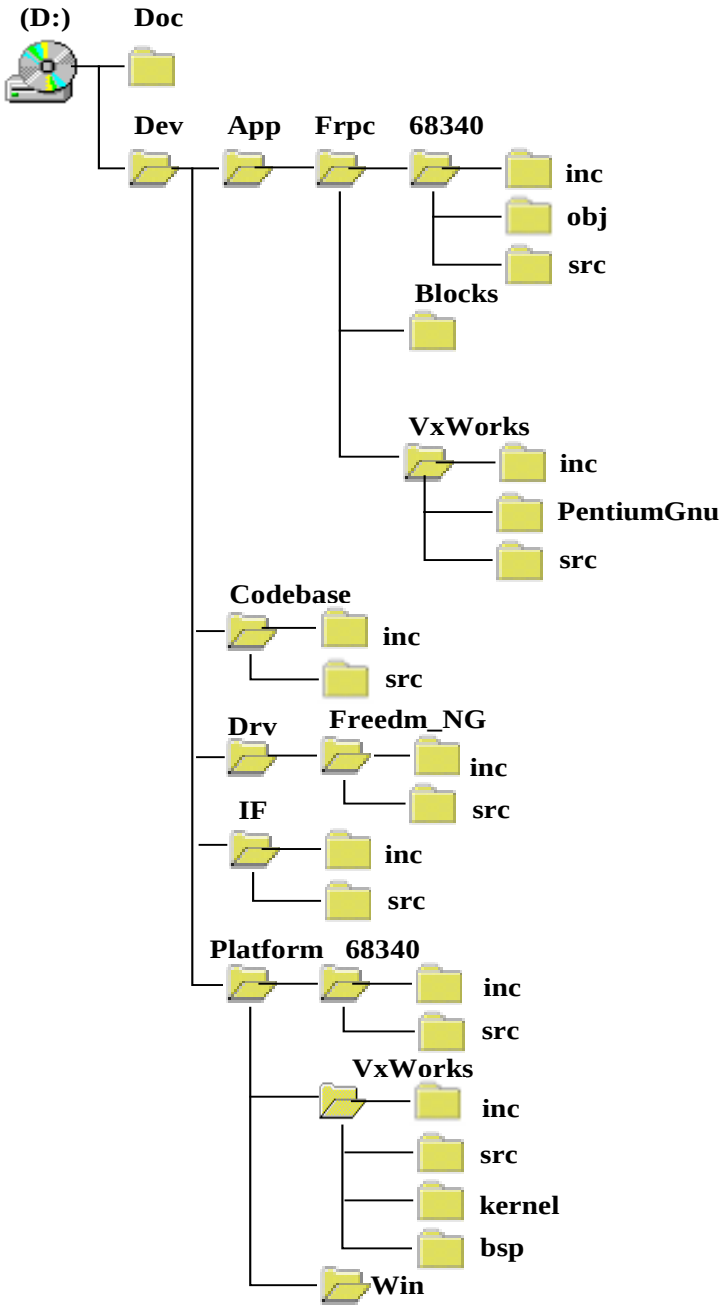


**Figure 7 FRPC Distribution CD-ROM Directory Structure**

# 7 TARGET SYSTEM STARTUP

This section provides the list of steps necessary to connect all the components of the Frame Relay Port Card development platform and start up the application software on the target platform.

- Plug a CPV5000 SBC motherboard into slot 0 of the target cPCI chassis.
- Plug the Frame Relay Port Card into one of the available cPCI slots.
- Connect the two black serial “Datalink” connectors into the COM1 and COM2 ports of the CPV5000.
- Connect one of the available COM ports on the host PC to COM1 on the target motherboard. This connection has to be made using a NULL modem cable from the host PC COM port to the DB-9 connector on the black “Datalink” connector.
- Connect COM2 on the target motherboard to the available COM port on the Frame Relay Port Card using the provided serial adapter (RJ-45 to DB-9). Note that the CPV5000 is configured as a DCE, while the combination of the Frame Relay Port Card serial port and the provided RJ-45 to DB-9 adapter form a DTE.

Once the described connections are made and software installation on the target system is finished, the Frame Relay Port Card development platform is ready to run. The user should ensure that the BIOS bootup sequence on the target machine is “A: then C:”.

In order to run the system the user has to do the following:

- Insert the VxWorks boot floppy into the target machine floppy drive.
- Reboot the target platform (by either switching the power off and on, or pressing the reset button).

After VxWorks finishes its boot up sequence, the VxWorks Shell command prompt should show up on the terminal platform.

If there is no monitor attached to the target platform, the user can see when that the boot sequence is finished from the Terminal Window on the Remote Command Machine.

The following lines appear when the target is ready (“>>” is ready prompt):

```
>> Serial channel - 115200-8-N-1
newboard
>>
```

# 8 OPERATIONS

This section describes how to configure the Frame Relay Port Card for the following modes of operation:

1. T1
2. E1
3. DS-3
4. Unchannelised DS-3
5. T1 Serial Clock and Data via QDSX

Other modes of operation are supported by the Frame Relay Port Card, however scripts for these modes are not included on the distribution CD-ROM.

Note that the following configurations can also be achieved by executing the appropriate setup.ini as part of a system self test. For example, to configure the board for T1 mode, select Self Test from the Function menu (from within HostApp) and then select the file setup_t1.ini. Similarly, to configure the board for E1 mode, select the file setup_e1.ini.

## 8.1 T1 Mode

The following scripts (located on the distribution CD-ROM) must be loaded and executed (in the order provided) to configure the board for T1 mode of operation.

1. temuxconfig_all_T1.txt
2. temux_sbi.txt
3. frd_start_t1.txt
4. frd_fix.txt
5. frdprov_chall_t1.txt
6. frd_stst.txt *

* - Only need to load if running the self test.

## 8.2 E1 Mode

The following scripts (located on the distribution CD-ROM) must be loaded and executed (in the order provided) to configure the board for E1 mode of operation.

1. temuxconfig_all_E1.txt
2. temux_sbi.txt
3. frd_start_E1.txt
4. frd_fix.txt
5. frdprov_chall_E1.txt
6. frd_stst.txt *

* - Only need to load if running the self test.

## 8.3 DS-3 Mode

The following scripts (located on the distribution CD-ROM) must be loaded and executed (in the order provided) to configure the board for DS-3 mode of operation.

1. config_alltmx_nlb_ds3.txt
2. temux_sbi.txt
3. frd_start_ds3.txt
4. frd_fix.txt
5. frdprov_chall_T1.txt
6. frd_stst.txt *

* - Only need to load if running the self test.

## 8.4 Unchannelized DS-3 Mode

The following scripts (located on the distribution CD-ROM) must be loaded and executed (in the order provided) to configure the board for unchannelized DS-3 mode of operation.

1. config_alltmx_nlb_ds3.txt
2. config_alltmx_nlb_unds3.txt
3. temux_sbi_ds3.txt

4. frdinit.txt
5. freedmds3.txt
6. freedm_sbi_uds3.txt
7. frdprov_chall_unds3.txt
8. frd_fix.txt
9. frd_stst.txt *

* - Only need to load if running the self test.

## 8.5 T1 Serial Clock and Data via QDSX

The following script (located on the distribution CD-ROM) must be loaded and executed to configure TEMUX3 for T1 serial clock and data mode of operation.

1. temux_qdsx.txt

# 9 APPENDIX A: COMMAND SET

This section is provided to assist those developing their own host applications (i.e. own scripts using TCL). HostApp combined with the provided command block takes care of most of the syntax issues.

The majority of commands supported by the FRPC software follows the common syntax as shown below.

*command devName <parameter list>*

The following are legal values for the parameter *devName*:

temux1
temux2
temux3
spectra_w
spectra_p
qdsx
freedm

The exceptions are the command handler commands where device name is omitted.

The command set may be divided into the following three groups:

General commands - available on every system running framework.

Device generic commands – same syntax and functionality regardless of the device or system. Commands that come in this group are: read, write, wwrite and watch.

Device specific commands – all other commands.

## 9.1 General Commands

### 9.1.1 Command Handler

**Exit** - s/w shutdown command.
**Syntax**: exit

**List** - The result of this command is a list of all the commands available on the particular logical board.
**Syntax**: list

**Identify** - Request for identification from the particular target system. Unique identification for each logical board is provided at compile time.

**Syntax**: *identify*

**Prompt** - Toggle to suppress or enable prompt (“>>”) display.

**Syntax**: *prompt*

### 9.1.2 Communication Port Control

**Echo** – Controls character echo coming from target device

**Syntax**: *echo comN on|off*

**Example**: echo com1 on

**Fctrl** – Flow control function is not implemented.

### 9.1.3 Block Handling

**Load** – This command loads a block from the file blockName on the host to memory on board-N. The block will be appended to the target list of available blocks

**Syntax**: *load boardN block blockName <size>*

**Example**: load board0 block frd_start.txt 62

**Exec** - Sends a request to the scripting processor to execute commands from block blockName.

**Syntax**: *exec boardN block blockName*

**Example**: exec board0 block frd_start.txt

**Del** - Deletes block blockName from memory and removes it from the list of available blocks.

**Syntax**: *del boardN block blockName*

**Example**: del board0 block frd_start.txt

**Dir** – Returns a list of currently loaded blocks.

**Syntax**: *dir boardN block*

**Example**: dir board0 block

### 9.1.4 File Handling

**Load** – This command loads a file from the file fileName on the host to the current directory on the logical boardN. This command is only supported on systems with disk drives.

**Syntax**: *load boardN file fileName <size>*

**Example**: load board0 file frpc.out 324567

**Dir** - List of files in the pathName directory.
**Syntax**: *dir boardN file pathName*
**Example**: dir board0 file /vxworks

**Del** - Removes a file from the directory with the specified name. This command is disabled since it may cause unintended effects on system integrity if misused.
**Syntax**: *del board-N file fullFileName*

### 9.1.5 Device Generic Commands

**Read** - Reads data from register regNum on device devName.
**Syntax**: *read devName <regNum>*
**Params**: *<regNum> - hex*
**Example**: read temux1 10
read spectra_p 0

**Write** - Writes <data> to <regNum> on device devName.
**Syntax**: *write devName <regNum> <data>*
**Params**: *<regNum> - hex, <data> - hex.*
**Example**: write freedm 10 5
write spectra_w 32 8

**Wwrite** - Writes <data> to <regNum> and waits for BUSY bit to get reset. Maximum wait is 1ms and if BUSY bit stays set, the command fails.
**Syntax**: *wwrite devName <regNum> <data>*
**Params**: <regNum> - hex, <data> - hex.
**Example**: wwrite temux3 20 10

**Watch** - Adds the watch request for <regNum> on the specified device. Values are updated every 5sec, unless specified. All watches on the same device are parts of the same watch interface, therefore a period change for one of the monitored registers will affect the others. Watches on other devices are updated independently.
**Syntax**: watch devName <regNum> <add|del|period> <period_sec>
**Params**: <regNum> - hex, <period> decimal.
**Example**: watch freedm 40 add 5
watch freedm period 20
watch freedm del

### 9.1.6 Board Specific Commands

The following is the list of commands with explanations for board0.

#### 9.1.6.1 Board0

Note 1: Prior to issuing any commands to the FREEDM™-84P672, the device must be properly initialized. Please refer to the script frd_start.txt for further information.

Note 2: Transmitting messages shorter than the CRC size (for example, one byte) are illegal.

Remote - This command enables the user to send commands to devices on remote board. E.g. send remote write freedm <regNum> <data>

**Syntax**: *send remote <command recognized by devices on the remote node>*

##### 9.1.6.1.1 Freedm

**Device generic commands** – all four generic commands apply to the FREEDM™-84P672 device.

**Init** - Command to initialize the FREEDM™-84P672 device driver for the device devID on the board plugged into the specified cPCI slot.

**Syntax**: *init freedm <slotDevID>*

**Params**: <slotDevID> - hex value. This is an eight bit value where the upper four bits decode the slot number and lower four bits equal the ID of the FREEDM™-84P672 device on that board. Since there is only one FREEDM™-84P672 device on the FRPC, the deviceID nibble is set to 0. Note that the FREEDM™-84P672 device driver expects the first slot to be 0.

**Example**: init freedm 0x20 /* this command will initialize FREEDM™-84P672 device on the board plugged into the third cPCI slot */

**Remove** – Closes the connection to the FREEDM™-84P672 device driver and de-allocates all memory blocks allocated to that driver.

**Syntax**: *remove freedm*

**Set** - Command to load parameters necessary to initialize the FREEDM™-84P672 device. <LinkFmt> specifies link data type (FREEDM_T1_FRAMED, FREEDM_E1_FRAMED, FREEDM_T1_UNSTR, FREEDM_E1_UNSTR. The Default value is FREEDM_T1_UNSTR). Initialization vector <initVec> contains

parameters to setup registers, buffers and rx/txLinkInfo. If omitted s/w provides default vector.

**Syntax**: *set freedm <linkType> <initVec>*
**Params**: <initVec> - for details see the FREEDM™-84P672 Programmer's Guide and driver Api source file frd_api.c.
**Example**: set freedm t1_framed 0

**Reset** – freedm device s/w reset. This command is automatically executed when remove freedm command is issued.

**Syntax**: *reset freedm*

**Activate** – Installs Interrupt Service Routine and Deferred Processing Routine that handles the FREEDM™-84P672 device interrupts.

**Syntax**: *activate freedm*

**Deactivate** – Removes Interrupt Service Routine and Deferred Processing Routine.

**Syntax**: *deactivate freedm*

**Tx** – Transmits <data[]> through channel <chan>. Note that this command assumes that the full datapath, including channel provisioning, has been performed.

**Syntax**: *tx freedm <chan> <data[]>*
**Params**: <chan> - decimal, <data> - decimal
**Example**: see paragraph 9.6.

**Selftest** – This command has the same effect as Tx with the addition that the transmitted data will be recorded and checked against incoming data stream on the same channel.

**Syntax**: *selftest freedm <chan> <data[]>*
**Params**: <chan> - decimal, <data> - decimal
**Example**: see paragraph 9.7.

**Txblock** – This function enables large blocks of data to be send by the host and output to <chan>.

**Syntax**: *txblock freedm <chan> <blockSize>*
**Params**: <chan> - decimal, <blockSize> - decimal

**Rxprov** – Provisions the FREEDM™-84P672 receive channel <chan> to one or more time slots on the specified link. The Time slot value is entered as a hexadecimal number while all other values are decimal numbers. RHDL parameters are: offset, prov, strip, delin, priority, invert, crc and s7bit. For further information , please refer to section 9.5, the FREEDM™-84P672 Data Sheet, and the FREEDM™-84P672 Programmer's Guide.

**Syntax**: *rxprov freedm <chan> <link> <time slot> <numBlocks> <xferSize> <rhdl params[0-7]>*
**Params**: <timeslot> - hex, <chan>, <link>, <numBlocks>, <xferSize>, <rhdl> - decimal
**Example**: see paragraph 9.5.

**Txprov** – provisions the FREEDM™-84P672 transmit channel <chan> to one or more time slots on the specified link. Time slot value is entered as a hexadecimal number while all other values are decimal numbers. THDL parameters are: offset, prov, delin, priority, invert, dfcs, crc, s7bit, trans, idle, level and flag. . For further information , please refer to section 9.5, the FREEDM™-84P672 Data Sheet, and the FREEDM™-84P672 Programmer's Guide.

**Syntax**: *rxprov freedm <chan> <link> <time slot> <numBlocks> <xferSize> <thdl params[0-10]>*
**Params**: <timeslot> - hex, <chan>, <link>, <numBlocks>, <xferSize>, <thdl> - decimal
**Example**: see paragraph 9.5.

**Unprov** – Unprovisions the specified channel.
**Syntax**: *unprov freedm rx|tx <chan>*
**Params**: <chan> - decimal
**Example**: unprov freedm rx 0

**Info** – Displays provisioning information for a given channel.
**Syntax**: *info freedm get rxprov|txprov <chan>*
**Params**: <chan> - decimal
**Example**: info freedm get rxprov 1

**Test** – Performs selected (register or memory) test.
**Syntax**: *test freedm reg|mem*
**Example**: test freedm reg

**Stats** – This function displays packet statistics for a given link or channel.
**Syntax**: *stats freedm link|chan|other <chan|link> periodic|del|reset*
**Params**: <chan> - decimal value.
<link> - decimal value.
periodic – requires periodical updates (on every 5 seconds).
del – removes previously set periodic watch.
reset – resets statistic values on a selected channel.
**Example**: stats freedm chan 2 periodic
stats freedm chan 2 reset
stats freedm chan 2 del

**Loop** – Sets different loop-back modes (chan, allchan, diagchan, diaglink).
Chan – Software loopback on the given channel pair in the host PCI memory.
Allchan – Special case of the above where each channel's receiver is looped to the transmitter of that same channel.
Diagchan – Diagnostic loop-back mode which sets CDLBEN bit for the selected channel causing all data transmitted on that channel to be written into the same channels receiver stream.
Diaglink – Puts the selected link into line loop-back mode where receive link data from the SBI PISO is looped back into the SBI SIPO.
**Syntax**: *loop freedm chan <rx> <tx>*
*loop freedm allchan*
*loop freedm diagchan <chan>*
*loop freedm diaglink <link>*
**Params**: <chan>, <link> - decimal

Please refer to the FREEDM™-84P672 Programmers Guide and FREEDM-NG Driver Manual for detailed explanation regarding different loopback modes.

**Disp** – Sets the display mode for the FREEDM™-84P672 packets. The contents of the received or transmitted packet may be displayed on the VxWorks console if rx|tx switch is selected, or sent up to the host in case host option is selected.
**Syntax**: *disp freedm rx|tx|host <chan> on|off*
**Params**: <chan> - decimal

### 9.1.6.2 Board1

#### 9.1.6.2.1 Micro

**Device generic commands** – all four generic commands apply to the microprocessor device.

**Monitor** – Starts up APS (Automatic Protection Switching) monitoring. In the case of a loss of signal (LOS) alarm on the currently active SPECTRA-155, the data path will be switched to the other one.
**Syntax**: *monitor micro on|off*

#### 9.1.6.2.2 Spectra_w/Spectra_p

**Device generic commands** – All four generic commands apply to the SPECTRA_155s.

**SetInt** – Activates/deactivates interrupt handling for the selected SPECTRA-155 device.

**Syntax**: *setint spectra_w/spectra_p on|off*
**Example**: setint spectra_w on
setint spectra_w off

#### 9.1.6.2.3 QDSX

**Device generic commands** – All four generic commands apply to QDSX.

#### 9.1.6.2.4 Temux1/2/3

**Device generic commands** – All four generic commands apply to all three TEMUXs.

**Iwrite** – Indirect write command.
**Syntax**: *iwrite temuxN <slice> rpsc|tpsc <reg> <val>*
**Params**: <slice> - decimal value in the range [1,28], <reg>,<val> - hex
**Example**: iwrite temux3 12 tpsc 4 20

**Iread** - Indirect read command.
**Syntax**: *iread temuxN <slice> rpsc|tpsc <reg>*
**Params**: <slice> - decimal value range in the [1,28], <reg>,<val> - hex
**Example**: iread temux2 7 rpsc 4

**SetInt** – Activates/deactivates interrupt handling for the selected TEMUX device.
**Syntax**: *setint temuxN on|off*
**Example**: setint temux3 on
setint temux1 off

# 10 APPENDIX B: BLOCK FILES DESCRIPTION

## 10.1 temuxconfig_all_T1.txt

This script configures the Working and Protection SPECTRA-155, and all TEMUX devices for T1 mode.

### 10.1.1 Working and Protection SPECTRA-155 Configuration

The following is a basic description of the configuration performed. Please refer to the script and the device register description for a complete description.

- Receive and Transmit line side operating mode set to STS-3 (STM-1/AU3).
- Path AIS, REI/RDI not inserted over DROP bus.
- LDC1J1V1 indicates only C1 and J1 bytes.

### 10.1.2 TEMUX #1- TEMUX#3

The following is a basic description of the configuration performed. Please refer to the script and the device register description for a complete description.

- High Density T1/E1 Framer Mode
- SBI system side interface
- T1/E1 Mapper line side interface
- Telecom Add and Drop SPE number (for example, 1 for TEMUX #1).
- TU processing in the RTDM and TTMP enabled for all tributaries in each TUG2 for TUG3 #1 (TEMUX #1) through TUG3 #3 (TEMUX #3)
- All 28 COMET slices configured for T1, ESF mode.
- Egress tributary clocking over the SBI bus is configured for master mode.

## 10.2 temux_sbi.txt

The script temux_sbi.txt enables each SBI ADD and DROP bus tributary and configures the TEMUX EXSBI block for clock master mode.

## 10.3 frd_start.txt

This script first initializes the FREEDM™-84P672 device driver for the device on the board in the third cPCI slot. It then loads the default initialization vector (this vector is part of the driver code and initializes various driver data structures) and sets up the FREEDM™-84P672 device for T1_FRAMED mode. Finally, the command sets up and activates the driver interrupt routine.

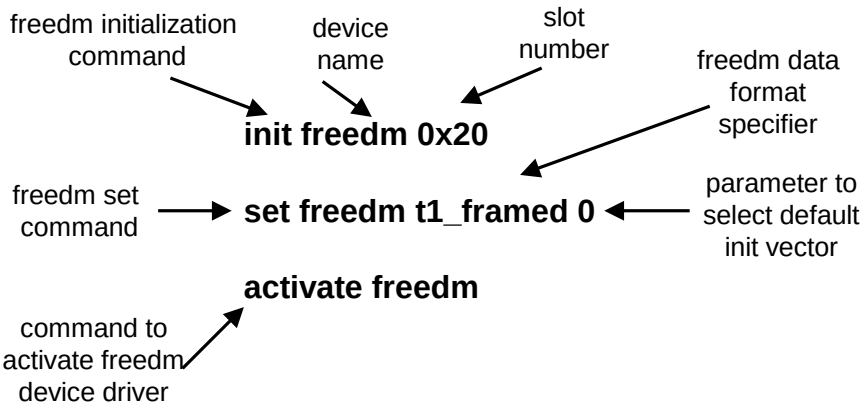


**Figure 8 frd_start Command Format**

## 10.4 frd_fix.txt

Please refer to section 2.5 of FREEDM™-84P672 Revision C device errata (PMC-2000953).

## 10.5 frdprov_chall_T1.txt

This script provisions 84 receive and transmit channels within the FREEDM™-84P672. The following example illustrates how the script is used to provision receive and transmit channels.

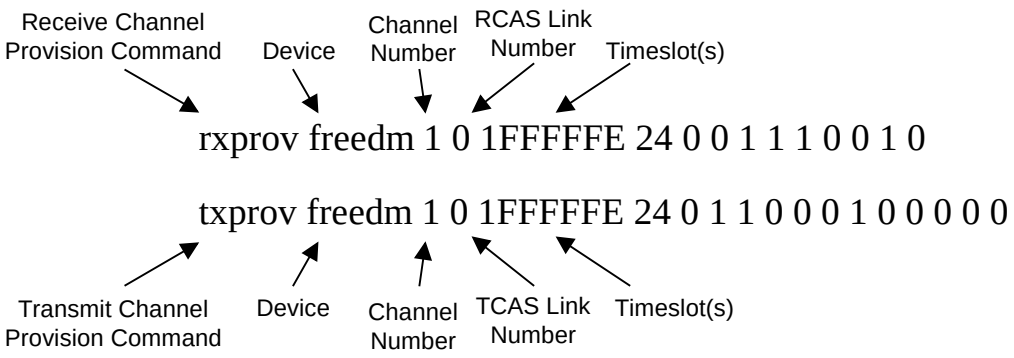


**Figure 9 RXPROV, TXPROV Command Format**

Valid channel numbers are from 0 through 1023. Valid link numbers are 0 through 83 for T1 mode, and 0 through 62 for E1 mode.

The value for the timeslot field is determined in the following way:

| 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |

**Figure 10 RXPROX, TXPROV Timeslot Value Determination**

The top row of the above drawing represents timeslots from 0 through 24 (note that timeslot 0 is only valid for unframed mode of operation). The bottom row represents whether that particular timeslot is provisioned for that channel (1) or not (0). Therefore, for the above example, the value in the timeslot field should be 0x192918E. The value 0x1FFFFFE represents all timeslots (1 through 24) of the specified T1 link provisioned for the given channel number.

Please refer to the FREEDM™-84P672 Programmers Guide and FREEDM-NG Driver Manual for further information regarding the remaining parameters.

### 10.6 frd_tx.txt

This script is used to transmit specified message on the specified channel, as indicated in the following example:

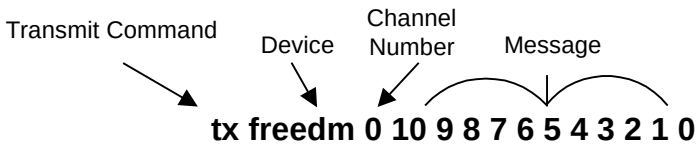


**Figure 11 Transmit Command Format**

## 10.7 frd_stst.txt

This is a script that puts the FREEDM™-84P672 interface code in the self-test mode. Self-test works the same way as the Tx command with the addition that transmitted packet gets recorded and checked against the next received packet on that same channel. Self-test is successful when these packets are identical.

# 11 APPENDIX D: HOW TO REBUILD THE MC68340 APPLICATION

Application executable is provided in the EPROM with the FRPC and it's copy may also be found on the distribution CD in the \dev\app\frpc\68340\obj directory. The program main.obj is the version of the executable as used by the BDM development system while main.s19 contains the same code in EPROM format.

In order to rebuild code for the microprocessor, the user needs Introl development tools revision 3.10.

Makefile can be found in the directory \dev\app\frpc\68340. All files addressed by the makefile are located at the same relative locations so user needs only to modify the base directory. To rebuild the application, open the DOS box, go to the directory where makefile resides, and enter **make** command from the DOS command prompt. As a result, all of the object files and executables will be created and placed in the current directory. File main.s19 can be downloaded to an EPROM.

The Frame Relay Port Card application code for the microprocessor can be build to run in the debugging environment with the BDM controller, or as a standalone system with the executable code loaded in EPROM. The only difference between these two versions is in the code starting address provided to the loader.

For the BDM based system, the base address has to be set to 0x0000 E000.

When the code executes from EPROM, the starting address has to be set to 0x0010 0000.

Loader information is located in the files **c68.ld** (BDM version) and **c68_epr.ld** (EPROM version). The makefile is set to create both versions automatically. The BDM code can be found in the **main.obj** file while the EPROM code is in **main.s19**.

Please refer to the Frame Relay Port Card Reference Design (PMC-1990533) for information regarding jumper configuration.

# 12 APPENDIX E: HOW TO BUILD VXWORKS KERNAL AND BSP PACKAGE

This section covers the requirements for rebuilding the VxWorks kernel and the Pentium BSP package.

> Note: Kernel files (VxWorks and VxWorks.sym ) are provided on the distribution CD and may be found in the \dev\platform\vxworks\kernel directory. BSP_Pentium is also provided as a part of the FRPC SDK and is located on the boot floppy.

First, the user has to install the Tornado 2 Development Platform. Then move the following files:

> sysLib.c, sysPci.c, sysPci.h and Config.h
>
> (since these files already exist on the BSP directory user may find it safer to take original files and modify them according to the ones found on the CD)

to the appropriate BSP_Pentium directory.

The following steps are used to create a VxWorks bootable floppy that will contain the BSP_Pentium files.

1. Edit Config.h and modify DEFAULT_BOOT_LINE according to Config.h provided on the distribution CD. This line provides information on where to find the kernel files (VxWorks and VxWorks.sym) and which script file (VxConfig.txt see next paragraph) should be executed once boot sequence finishes.

2. Make sure Config.h includes prjParams.h (located on the BSP directory and available on the distribution CD).

3. Start up Tornado Development environment.

4. From the Tornado Project menu, select the Build option and then select “Build Boot ROM”.

5. From the build menu select BSP_pentium (left had side window) and bootrom_uncmp as an image to build. This will result in the creation of the bootrom_uncmp file that will be placed on the BSP_Pentium directory.

6. To create a VxWorks bootable floppy disk, place an empty floppy in the floppy disk drive and run mvxwbd.bat (either from MyComputer or from DOS prompt).

7. When the mvxwbd.bat batch file finishes, remove the floppy disk and make it read only to prevent other users from corrupting its content.

The following steps are used to generate a VxWorks kernel.

1. From within the Tornado environment, create a new project.

2. Select option "Create Bootable VxWorks image".

3. Base project on the /Tornado/target/config/BSP_pentium board support package.

4. Place new project in existing directory and let Tornado create files and generate dependencies. Files placed in the project directory are: linkSyms.c, prjConfig.c, romInit.s, romStart.c, sysALib.s and usrAppInit.c). As an external dependency, prjComps.h should be included.

5. From the Tornado Project menu, select the Add/Include option and add sysLib.c file from BSP directory.

6. Regenerate dependencies.

7. From the Tornado Project menu, select the Build option and then from the drop down menu, select the build item. As a result, Tornado will add a new subdirectory Default and place all the object files and new kernel files VxWorks and VxWorks.sym in the directory.

8. Copy these two files to the target machine on c:\vxworks\kernel directory.

Everything is now ready to start up the VxWorks target system. Place the boot floppy disk in the target floppy disk drive, and reboot the target computer.

Note: The Kernel files may be placed elsewhere in the system (e.g. network drive) as long as they are available to the target machine during boot time and if the file Config.h contains the DEFAULT_BOOT_LINE line modified to point to that location.

# 13 APPENDIX F: HOW TO REBUILD THE VXWORKS APPLICATION

To rebuild the VxWorks application, the user needs to have the Tornado 2.0 development platform installed on the host computer.

The first step is to copy all of the files from the dev directory on the distribution CD to the local host. The complete directory structure should be restored as on the CD.

The directory \dev\app\frpc\vxworks already contains the workspace and project files (frpc.wsp and frpc.wpj). Since the application workspace is created from within Tornado, all resources are added through the Project Add/Include menu and contain absolute references. The easiest way to recreate the building environment is to create new workspace and add resources from their current locations.

1. Remove PENTIUMgnu directory and files frpc.wsp, frpc.wpj, prjObj.lst and makefile from the \dev\app\frpc\vxworks directory (these files will be recreated once new project is added).

2. Start the Tornado Development Environment.

3. Select New Project from the File drop down menu.

4. Select the option “Create downloadable application modules for VxWorks”.

5. Enter Frpc for the project name, select \dev\app\frpc\vxworks for the location directory and enter \dev\app\frpc\vxworks\frpc.wsp as a workspace name.

6. Double click on Next> option and check box that selects Toolchain as a base for the new project. From the drop-down menu on the right hand side, select PENTIUMgnu as a base.

7. Select Next> option again, check if all the entered parameters are correct and press the Finish button. This will create an empty project. Now the source files need to be added.

8. From the Project drop down menu select Add/Include option and add the following files:

   - \dev\app\frpc\vxworks\src\cbmain.c

   - \dev\codebase\src*.c (all .c files from this directory)

   - \dev\drv\freedm_ng\drv\src*.c (all .c files from this directory).

- \dev\if\src\freedmif.c
- \dev\platform\vxworks\src*.c (all .c files from this directory).

1. The next step is to set compiler options so that it is able to find all of the included files.
2. From the Workspace window (that should be visible as soon as workspace is created), click on the Builds tab.
3. Open FRPC Builds (click on '+'). PENTIUMgnu should be the only item underneath FRPC builds.
4. Right click on PENTIUMgnu and select Properties from the drop down menu.
5. New window "Properties: build specification 'PENTIUMgnu' will appear. Select "C/C++ compiler" tab. Edit window underneath the tabs should contain following references:

   - g -mpentium -ansi -nostdinc -DRW_MULTI_THREAD
   - D_REENTRANT -fvolatile -nostdlib -fno-builtin -fno-defer-pop
   - ID:/Tornado/target/h
   - ID:/data/work/dev/app/frpc/vxworks/inc
   - ID:/dev/codebase/inc
   - ID:/dev/platform/vxworks/inc
   - ID:/dev/if/inc
   - ID:/dev/drv/freedm_ng/drv/inc
   - ID:/Tornado/target/config/all -ID:/Tornado/target/config/bsp_pentium
   - ID:/Tornado/target/h/drv/pci/ -DCPU=PENTIUM

Note: Drive name and full directory path for both system and user include files should reflect locations on the host platform (e.g. C:/data/work/dev/app/frpc/vxworks/inc).

1. Select Apply option.
2. Activate "Workspace:FRPC" window and select Files tab from the bottom of that window.

3. Select Build item from Build menu. Make sure the focus is set to one of the listed FRPC files otherwise Build item cannot be activated. Tornado should now be able to rebuild the complete application. Object files and executable – FRPC.OUT will be created in the PENTIUMgnu subdirectory.

Note: Before making any changes to the FRPC Application, make sure the original application may be recreated by following the procedure explained in the previous text.

To activate new application code, restart the target platform in DOS mode (remove boot floppy and restart computer) and copy frpc.out to the directory C:\VxWorks\Apps

# 14 APPENDIX G: CREATING VXWORKS BOOT FLOPPY

All the files needed to create a VxWorks boot floppy disk are located on the BSP directory To create VxWorks bootable floppy disks, place an empty floppy in the floppy disk drive and run mvxwbd.bat (either from MyComputer or from DOS prompt).

When mvxwbd.bat batch file finishes, remove the floppy disk and make it read only to prevent other users from corrupting its content.

# 15 APPENDIX H: CONTENTS OF THE VXWORKS.TXT STARTUP SCRIPT

VXWORKS.TXT is a special file: it is executed automatically upon the startup of the system. The name of the file is arbitrary and is specified in the default boot line during the compilation of the VxWorks boot code.

```
##############################################################################
# 06 Jun 2000
# File: vxconfig.txt
# Script file for VxWorks boot from the HDD. This assumes that default
# boot line points to the HDD.
##############################################################################

# Give HDD a name.
usrAtaConfig(0,0,"c:")

# Give FDD a name.
usrFdConfig(0,0,"a:")

# Load FRPC Application.
ld 1,0,"frpc.out"

# Start FRPC Application
cbmain
```

**Figure 12 Contents of the VXWORKS.TXT Startup Script**

This script will attach the drive letters to the floppy and hard disks, load the FRPC application, and start its execution.

# 16 APPENDIX I: HYPERTERMINAL CONFIGURATION

There are many terminal programs on the market. The example in this document will show (step by step) how to properly configure Windows HyperTerminal.

After opening HyperTerminal the first step is to configure the properties of the current session.

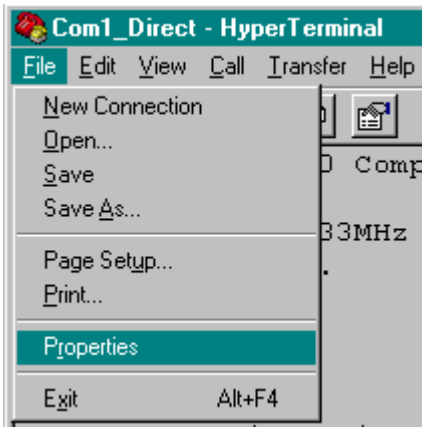


**Figure 13 Setting up the HyperTerminal Properties**

After clicking on Properties, the HyperTerminal “Properties” dialog box will show up.

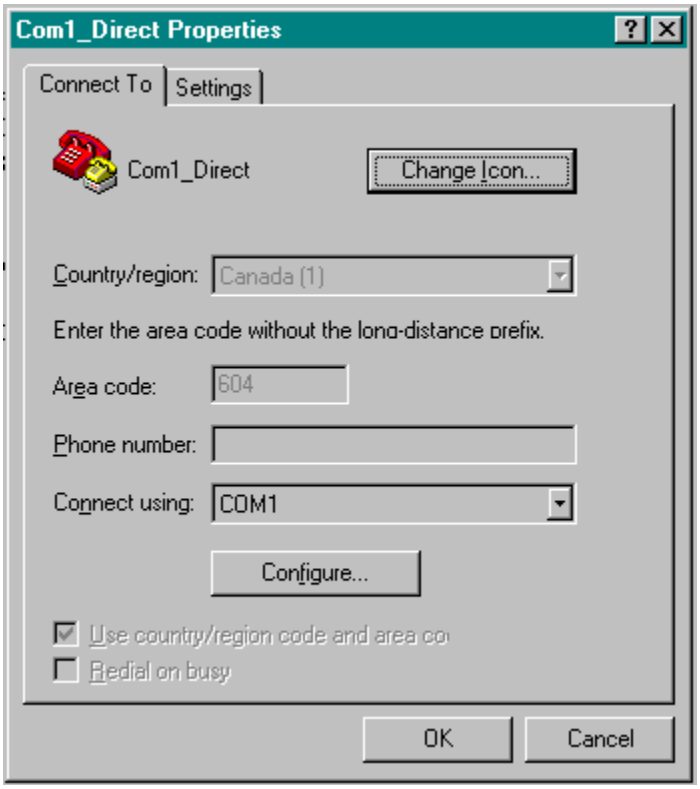


**Figure 14 HyperTerminal Properties Box**

We will assume here that COM1 is the port that is free to use for communication between the host system and the target machine. If COMx is the communication port that is free, the user should select it from the dropdown list and click on the Configure button.

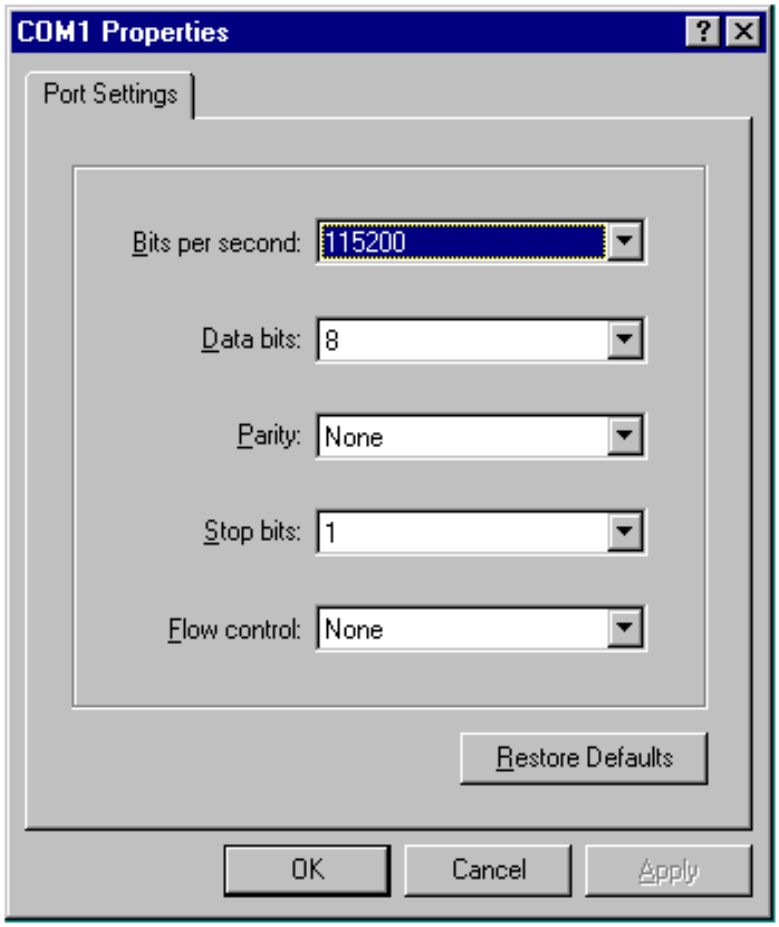


**Figure 15 COM Port Settings**

The “Port Settings” dialog will show up. The serial port configuration should be exactly the same as shown on Figure 12. After setting up the port, click OK. The COMx properties dialog box will disappear.

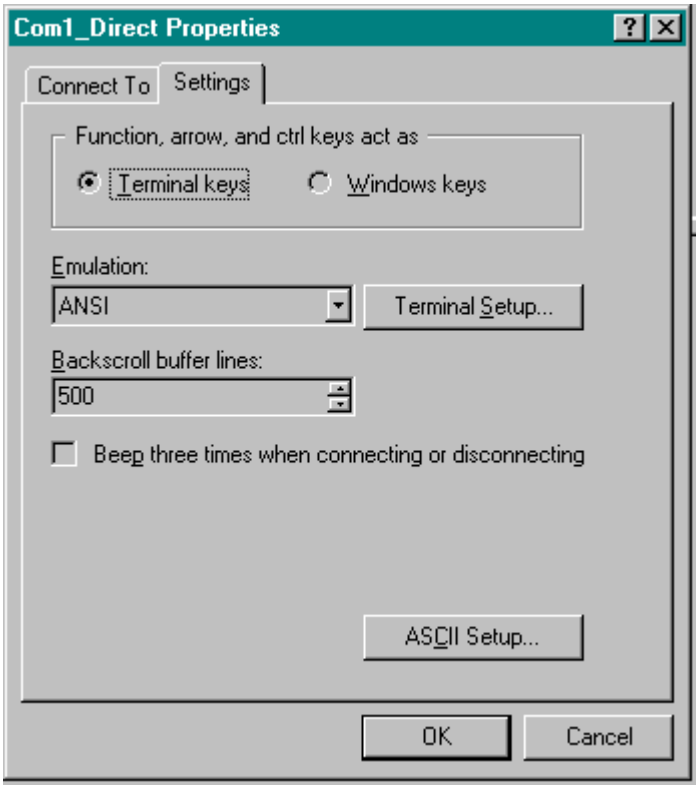


**Figure 16 HyperTerminal Settings Dialog**

Click on the “Settings” tab and select “Terminal keys”, “ANSI” Emulation and enter “500” Backscroll buffer lines.

Click on the “ASCII Setup…” button and set ASCII parameters as shown on the following figure.

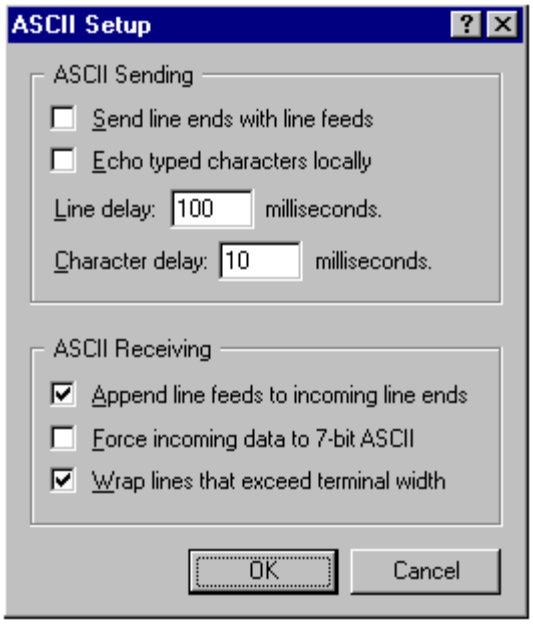


**Figure 17 ASCII Setup Dialog**

Once setup has been completed click OK and the “ASCII Setup” dialog box will disappear.

Click OK on the session properties dialog box.

The user must save the properties in order for them to come into effect Otherwise, HyperTerminal will keep the old settings. After the terminal configuration is complete, the Remote Command Machine is ready to communicate with the target system.

## NOTES

## CONTACTING PMC-SIERRA, INC.

PMC-Sierra, Inc.
105-8555 Baxter Place Burnaby, BC
Canada V5A 4V7

Tel: (604) 415-6000

Fax: (604) 415-6200

Document Information: document@pmc-sierra.com
Corporate Information: info@pmc-sierra.com
Application Information: apps@pmc-sierra.com
(604) 415-4533
Web Site: http://www.pmc-sierra.com



PMC-2002057 (R2) Issue date: January 2001