

PROMJet

In-Circuit Emulator

User's Manual



EmuTec Inc.

Tel: 425.357.5200

Fax: 425.357.5201

Website: www.emutec.com

Email: support@emutec.com

LIMITED WARRANTY

EmuTec Inc. (EmuTec) warrants PROMJet against defects in material and workmanship for a period of One year from the date of purchase. During this warranty period, EmuTec will repair, or at its option, replace PROMJet at no charge when furnished with proof of retail purchase. This warranty applies only to the original retail purchaser and is not transferable.

This warranty does not apply if PROMJet has been damaged by accident, abuse or misapplication, or as a result of service or modification other than by EmuTec.

EmuTec IS NOT RESPONSIBLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES RESULTING FROM BREACH OF ANY EXPRESS OR IMPLIED WARRANTY, INCLUDING DAMAGE TO PROPERTY AND, TO THE EXTENT PERMITTED BY LAW, DAMAGES FOR PERSONAL INJURY. THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. PROMJet IS NOT INTENDED FOR USE IN LIFE SUPPORT EQUIPMENT OR IN MISSION CRITICAL APPLICATIONS.

Some states do not allow limitation of implied warranties, or the exclusion or limitation of incidental or consequential damages, so that the above limitations or exclusions may not apply to you. This warranty gives you specific legal rights and you may also have other rights that vary from state to state.

Note to the user: The technical information contained in this manual is based on information available at the time of publication and is subject to change without notice. Although every reasonable effort has been made to include accurate information, the statements in this manual are not warranties. EmuTec makes no warranty or claims as to the accuracy, completeness, or suitability for any particular purpose of the technical information provided.

This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without the prior written consent of EmuTec.

This product is protected by United States and foreign laws. You should not buy or use this product if you are a competitor or working as an agent for a competitor to EmuTec. You should not disassemble or reverse engineer the technology incorporated in this product. If you do not agree to the terms of this agreement, please do not use this product or install the software. Promptly return the product to the place of purchase for a full refund.

PROMJet is a registered trademark of EmuTec Inc.

Table of Contents

LIMITED WARRANTY	2
TABLE OF CONTENTS.....	3
PACKING LIST	5
INTRODUCTION	6
SYSTEM REQUIREMENTS.....	7
HARDWARE INSTALLATION	8
OPERATING INSTRUCTIONS FOR WINDOWS.....	11
USING PROMJET WINDOWS SOFTWARE	12
SETTING UP PROMJET AND FILE PARAMETERS	12
WINDOWS SOFTWARE OPERATION	18
<i>Downloading a Data File to PROMJet.....</i>	<i>18</i>
<i>Uploading a Data File from PROMJet.....</i>	<i>18</i>
<i>Comparing a Data File with PROMJet</i>	<i>18</i>
<i>Filling PROMJet with a Value</i>	<i>19</i>
<i>Editing PROMJet memory using binary editor.....</i>	<i>19</i>
<i>Editing data files using binary editor</i>	<i>20</i>
<i>Setting PROMJet configuration.....</i>	<i>20</i>
<i>Trigger logic Setup</i>	<i>20</i>
<i>Using PROMJet to Reset the Target Processor</i>	<i>21</i>
<i>Recording a Script File</i>	<i>21</i>
<i>Executing a Script File form Windows or DOS.....</i>	<i>21</i>
<i>Updating PROMJet Firmware</i>	<i>22</i>
OPERATING INSTRUCTIONS FOR LINUX	23
LINUX SOFTWARE OPERATION.....	23
EXAMPLES.....	28
USING PROMJET WITH FTP SOFTWARE	29
PROMJET ACCESS AND COMMUNICATION LIBRARY.....	32
HOST ACCESS FUNCTIONS (USB/LPT).....	33
HOST ACCESS FUNCTIONS (ETHERNET)	36
HOST COMMUNICATION FUNCTIONS (USB/LPT).....	38
HOST COMMUNICATION FUNCTIONS (ETHERNET)	40
TARGET COMMUNICATION FUNCTIONS.....	41
APPENDIX	44
TROUBLE SHOOTING TIPS	44
WRITE-BY-READING MEMORY ACCESS	45
SPI INTERFACE SIGNALS AND FUNCTIONS.....	47
<i>Data Read Command:.....</i>	<i>47</i>
<i>Data Read Fast Command:</i>	<i>48</i>
<i>Read Status Register Command:</i>	<i>48</i>
<i>Data Program Command:.....</i>	<i>48</i>
<i>Read ID Command:</i>	<i>48</i>

<i>Read PROMJet ID Command:</i>	49
LPC/FWH INTERFACE SIGNALS AND FUNCTIONS	50
<i>Memory Read Command:</i>	50
<i>Memory Write Command:</i>	51
<i>Read manufacturer and device ID Command:</i>	51
<i>Read general-purpose Inputs Command:</i>	51
<i>LPC Read Cycle:</i>	52
<i>LPC Read Cycle:</i>	52
<i>LPC Write Cycle:</i>	52
<i>FWH Read Cycle:</i>	53
<i>FWH Write Cycle:</i>	53
INTERFACE SIGNALS AND PINOUT	54
BINARY AND HEX FILE FORMATS.....	55
<i>Binary Format</i>	55
<i>Intel Hex Format</i>	55
<i>Motorola S Record Format</i>	55
IDENTIFYING YOUR PROMJET	56
ELECTRICAL AND PHYSICAL SPECIFICATIONS	57

Packing List

Each PROMJet package should include a 6-foot USB MINI-B cable to connect PROMJet to the Host PC. It also comes with Windows and Linux software and this user's manual.

Optional items, such as parallel-port adapter kit with power supply adapter, DIP, PLCC, TSOP, PSOP, SSOP or BGA adapters and extension cable can be ordered separately. Please contact EmuTec or your local representative for a list of available options.

Please check that the items received match your order. Contact EmuTec or your local representatives immediately if any items are missing or if items were damaged during shipment.

Introduction

PROMJet™ is a very advanced development tool for embedded systems. It is an ultra compact, high performance EPROM, FLASH and memory emulator that facilitates software development by eliminating the need for programming flash memory during an application's development cycle. With features such as high speed RAM (down to 25ns access time), ultra fast download speed (2Mbit/S), jumper-less configuration for parallel, LPC, FWH and SPI flash memory, Windows software, PROMJet is the best choice among all other emulators for a very affordable price.

PROMJet replaces the FLASH memory of the system under development allowing the user to load, examine, modify, view and patch the program code directly into PROMJet's emulation memory from a PC or workstation. The target system recognizes PROMJet as if it were an actual flash chip plugged into the target socket. Because PROMJet functions independently of microprocessor type, it may be used to develop code in virtually any target system that uses a DIP, PLCC, TSOP or PSOP JEDEC EPROM or FLASH. An external /WE signal allows target monitors to modify the ROM memory e.g. to set breakpoints or change power-up values. An external RESET and /RESET output signals can be used to reset the target system when necessary. PROMJet is cascade-able to emulate wider data bus (up to 128 bit) or multiple-ROM targets. PROMJet also supports **serial** FLASH memory interfaces for **SPI** and **LPC/FWH**.

Adding the **In-Circuit-Emulation** (ICE) option to PROMJet enables the host computer to access the emulation memory while the target processor is **running** without arbitration or wait-state signals. This allows the user to dynamically modify process control variables, look-up tables, loop counts, timing constants or force conditions to simulate external stimulus in real time without stopping the target system. It can be also used to establish a communication channel between the host computer and the target system via the dual-port memory. Furthermore, by adding the **Write-By-Reading** (WBR) option with the ICE feature, the target processor can modify PROMJet contents **only** by reading specific memory locations inside PROMJet. This feature allows the target to communicate with the host by reading PROMJet. The trigger circuit inside PROMJet allows the user to watch for an access to a memory location by the target CPU.

The ICE and WBR options are the key to true non-intrusive and transparent memory-based debugging. It enables PROMJet to work with software debuggers allowing the user to debug an embedded application from a standard PC or a workstation using parallel or Ethernet connection. PROMJet does not use any system resources (IO-addresses or UART) or any memory outside the monitor area. Using PROMJet instead of a serial connection frees the serial port for your application eliminating any conflicts between the RTOS and the debugger. PROMJet also provides the full ROM space during debugging so that your application can be executed from the location as in the final configuration eliminating the need to relocate the code after debugging. PROMJet provides nearly all the functionality of an ICE for a variety of processors at a fraction of the cost with no additional CPU pods.

System Requirements

PROMJet requires a PC running MS Windows 95/XP/Vista or higher with one free USB or parallel port (optional parallel kit required).

To use the networking capabilities of the software, your system must have a networking adapter compatible with windows.

To use PROMJet with Ethernet, you must use a special Ethernet adapter available from EmuTec for PROMJet. The adapter connects to PROMJet via the supplied 8-pin modular cable and to the Ethernet via a 10/100BASE-T connector.

The Ethernet adapter can also use FTP client software on any host system (PC, Mac, Linux or UNIX) to download, upload or set PROMJet size and mode. Contact EmuTec for more information regarding the Ethernet adapter.

Hardware Installation

This chapter explains how to install PROMJet to be used with a host computer. The installation is very simple and does not take more than few minutes to complete. Before starting the installation, make sure that you have all the components you need to connect PROMJet to the host computer (USB cable, RJ45 cable, parallel port connector or an Ethernet adapter) and any adapters needed to connect PROMJet to the target system such as 32-position PLCC or 48-pin TSOP adapters. Make sure the target system power is turned off and, if using the parallel port connection, verify that also your PC is turned off. Start the installation by plugging the supplied USB cable or parallel-port connector to the PC. If using an Ethernet adapter, use the supplied RJ45 cable to connect PROMJet to the adapter.

Connect PROMJet to the target system and **verify that pin 1 of PROMJet is connected to pin 1 of the target system connector or socket**. Finally, connect the other side of the cable to the appropriate PROMJet connection (USB or Parallel).

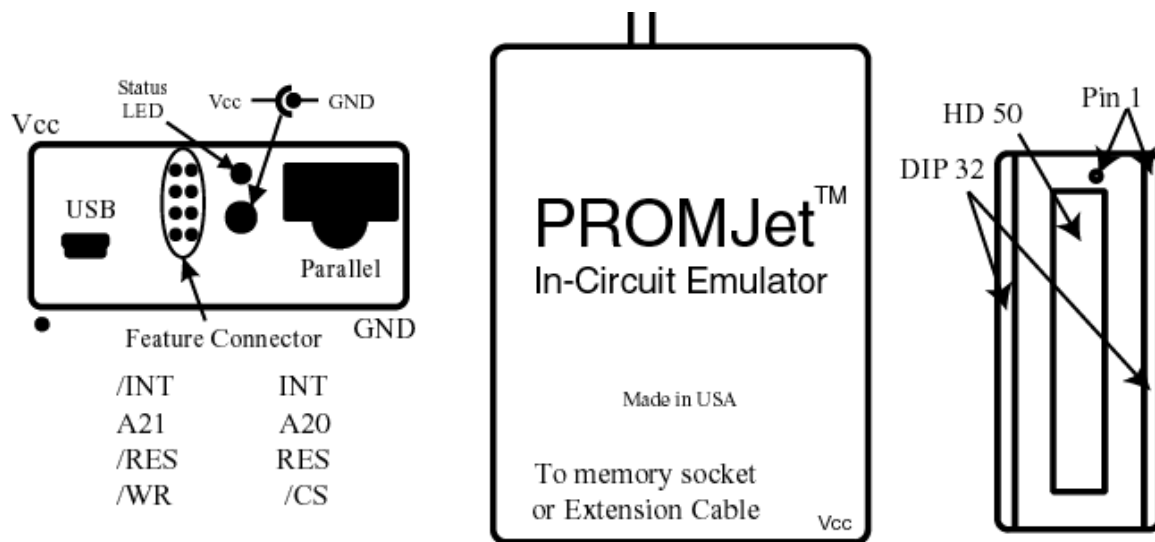


Figure 1: PROMJet top, side and bottom view.

After installing and connecting PROMJet, please turn on the power to the host PC or the Ethernet adapter and then to PROMJet if using external power supply. **MAKE SURE** that PROMJet is powered on **BEFORE** you power your target system. Failure to follow this sequence will cause damage to your PROMJet. That does not apply if PROMJet uses parasite power from the target system.

PROMJet has the capability of resetting the target system via an open collector Reset and /Reset signal. Use the /RES signal if an active low reset is needed; otherwise use the RES signal. **Please make sure that the target system's reset signal driver is disconnected or it has an open-collector output before connecting PROMJet's reset signal.** The reset signal will be **optionally** activated via the software while accessing PROMJet from the host PC (see software operation). The reset output becomes also active when the target is turned off (and PROMJet is powered via an external power

supply) or if the configuration word of PROMJet is not correct due to power failure. The status LED will change its color from green (normal operation) to red (reset status) while the reset signal is active.

When using a PROMJet **without** the ICE option, **the user must reset the target processor when accessing PROMJet's memory**. Specifying the reset option while executing the software (see below) can do this. When using a PROMJet-ICE, the **user has to reset the target processor only when downloading a new program**. It is not required to reset the target processor when modifying some of the data bytes on the fly or viewing PROMJet contents via the full-screen editor.

The target system can read from or write to PROMJet. Since the write signal is not available through the EPROM socket, this signal must be supplied via the /WR signal in the feature connector (see Figure 1). If the /CS signal of the EPROM socket (pin 22) is not active while the target system is writing to PROMJet, a second /CS signal must be connected to the /CS input of the external connector to select PROMJet. The /CS signal of the EPROM socket and the /CS signal of the external connector are anded to generate an internal /CS signal for PROMJet. The target system can use either one to read or write PROMJet. The second /CS signal can also be used to emulate another memory device occupying a different address space **but sharing the same address and data bus**; in this case, the /CS signal of the second device must be connected to the /CS signal of PROMJet.

If PROMJet is configured to emulate FLASH memory (see software operation), the /WR signal will be supplied through pin 31 of the FLASH socket (see the appendix for pinout information). In this case the A19/WR signal in the feature connector will serve as address 19 (A19). Please **do not** use this feature if you are connecting the PROMJet using the 50-pin High-density connector since all adapters are designed to work with the connector in EPROM mode.

If PROMJet is featured with the In-Circuit Emulation (ICE) option, it allows the host computer to access the emulation memory while the target processor is running without arbitration or wait-state signals. This allows the user to modify system variables in real time or make real-time watches without stopping the target system. This feature is also used to establish a virtual communication channel to connect a monitor program running on the target processor with a software debugger running on the host without using any target system resources (IO-addresses or UART) or any memory outside the monitor area.

If PROMJet is featured with the Write-By-Reading option, the target processor can also write to a 256-byte memory block, selected by the software on the host processor, without using the /WR signal. To write to the memory block, the target processor has to read from a specific 4-byte word inside the 256-byte memory block in a specific sequence. While reading from PROMJet, the target processor uses its address lines A3 to A0 to supply the data to PROMJet. To enable this mode, the user has first to set the address of this 4-byte word (see software operation for more information). The start address of the 256-byte memory block will be calculated by resetting the least significant byte of the 4-byte address (e.g. if the word address is 0xFD64, the 256 byte block will start at 0xFD00). See the appendix for more information regarding this feature.

When debugging 16 bit or wider targets and using multiple PROMJets, you have to make sure that the first PROMJet in the chain (the one closest to the PC) has the ICE option. Most debuggers require only the first PROMJet in the chain to have the ICE option. Please check the documentation that comes with your debugger software for more information. If you would like to access PROMJet's memory

while the target is running, only PROMJets with the ICE option can be accessed without stopping the target system.

PROMJet has the capability of generating an interrupt request to interrupt the target processor while running with a software debugger. Use the /INT signal if an active low interrupt signal is needed; otherwise use the INT signal. **Please make sure that the target system's interrupt signal driver is disconnected or it has an open-collector output before connecting PROMJet's interrupt signal.** This signal can only be activated via a device driver or a DLL that works with a debugger or a software library. It cannot be activated via the PROMJet software.

A20 and A21 are address input signals for EPROM/FLASH sizes higher than 8Mbit when using the 32-pin DIP connection. To emulate a 16 or a 32 MBit memory device, connect these signals with A20 and A21 of your target processor.

If PROMJet is featured with the Variable-Voltage option, it can then be used to emulate any memory that has a supply voltage between 1.8 and 5V. If PROMJet is used in a low-voltage system, the user **MUST** use the external 5Volt power supply from the top of PROMJet or the USB cable.

To emulate a 16-bit memory device, you can either use one 16-bit PROMJet or two 8-bit PROMJet devices. If you are using a single device, you should connect it to the target system using the 50-pin High-density connector on the PROMJet bottom. The 32-pin DIP connector supports only 8-bit devices.

If you are having problems using PROMJets due to physical limitations such as vertical clearance or having a 16 or 32 bit system that has the EPROM sockets very close to each other, you can use an extension cable (available from EmuTec) to connect PROMJet to the socket of the target system.

EmuTec also offers a 50-pin Flex-circuit cable to extend the 50-pin header. There is a wide variety of adapters (over 50) to connect PROMJet to other DIP, PLCC, TSOP, PSOP, SSOP and BGA devices. These adapters connect mostly via the 50-pin high-density connector on the bottom of PROMJet. When using such an adapter, please make sure to align pin 1 of the adapter to pin 1 of the PROMJet connector. It is normally identified by a white circle next to the 50-pin header. If the adapter has more than one connector, it should have more than one circle to identify pin 1 for each connector. The other end of the adapter will plug into a special connector on the target board (supplied by EmuTec) or soldered down to the memory device pads (such as TSOP pads) on the target PCB.

Operating Instructions for Windows

The windows software is used to operate PROMJet from a host PC running Windows 2000/XP or higher via a USB, parallel port or Ethernet connection. The software enables the user to download, upload, compare and setup PROMJet mode and size.

The software must be first installed into the host computer before it can be used with PROMJet. To do this, insert the supplied disk into the floppy drive of your PC. From the Start pull-up menu in Windows, choose RUN. In the Run dialog box type {drive letter}:\setup and then press OK. You can also use the Browse button in the dialog box to check the location of the setup program. The setup program will start and will guide the user throughout the installation. After completing the installation, the setup program will create a PROMJet program group for all installed PROMJet files.

The software has two different operating modes. The first one is a direct operating mode where the user inputs PROMJet parameters and data file information and then executes the desired function by pressing its button in the dialog box. When the user exits the software by pressing the EXIT button, the software will store all the information entered so that it will display it the next time it starts. So if the user is always downloading the same data file for a specific project, all what he has to do is to press the Download key even after exiting and restarting the software.

The second operating mode is the script execution mode. This mode is more suitable for executing series of commands. For example if the user would like to download multiple data files to one or multiple PROMJets, he has to record this series of operations only one time and then executes it every time he needs to download the files. This mode can be also called from within a DOS window under Windows by typing PROMJET <Script File> from the command line or a batch file. The software will start in script mode and will automatically run the script file without displaying the main dialog box except in the case of an error. This mode is suitable when the user would like to download a file after calling the compiler from a batch file. The script mode is described in more details in the Using Scripting Language section below.

The software does support up to 16 PROMJets connected via USB or parallel ports. The user can operate the PROMJets in any suitable configuration. For example if the user has 8 units he can either configure them as a single memory bank for a 64 bit target, or as two banks for a 32 bit target, or as four banks for a 16 bit target, or as eight banks for an 8 bit target. He can also choose different data-width combinations for different processors in a single target system (e.g. one 32 bit bank, one 16 bit bank and two 8 bit banks).

Before operating PROMJet, make sure that the unit is supplied with power either from the USB cable or from an external power supply and is connected to the PC via the provided USB, parallel port connector or Ethernet adapter.

Using PROMJet Windows Software

PROMJet has two device drivers. One is for the parallel port and the other is for the USB connection. Before using PROMJet software under Windows, the user must install PROMJet Device Driver. To do so, you **must** log into Windows as a system administrator. PROMJet software will install the parallel port device driver when it runs the first time and will display a message stating the success of the installation. The first time the user plugs PROMJet in the USB port of a PC, Windows will ask for the location of the device driver. It is installed in the PROMJet main directory, which is by default if the user did not change it during installation, C:\Program Files\EmuTec\PROMJet\.

Setting up PROMJet and File Parameters

This section explains the usage and the functionality of the various parameters of PROMJet windows software. The user has to adjust the values of these parameters before using PROMJet. In this Chapter we will explain the usage of each edit box and function button. The main dialog box of the software is displayed in figure 2 below.

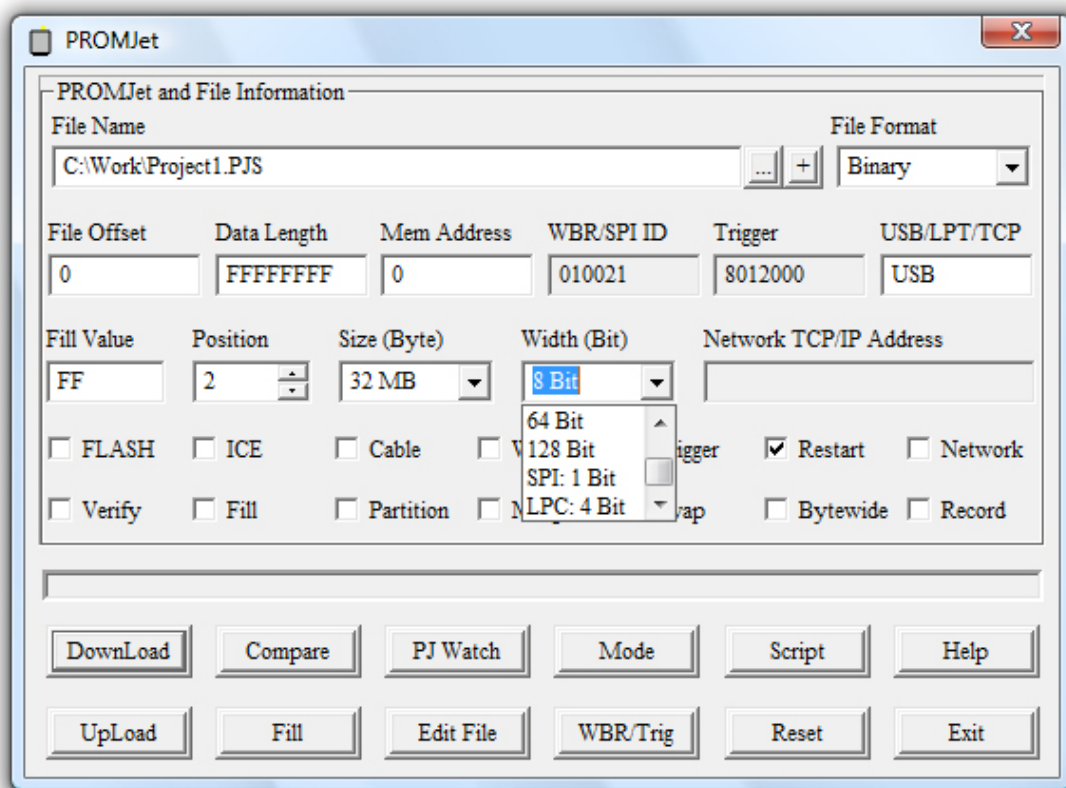


Figure 2: PROMJet software main dialog box.

File Name

This edit box is used to enter the name of the data file to be transferred between PROMJet and the PC. The user can either type the name of the file in the field or press the button on the right side of the field to start the familiar windows open-file dialog box. This dialog box can navigate the user through the available disk drives and data files in the system. If the user needs to create a new file, he has to navigate using the dialog box until reaching the desired sub-directory and then type the name of the new file in the appropriate edit box. The user has then to press the Open button to confirm the choice. PROMJet windows software accepts long file and directory names. Use the '+' button on the right side of the file name edit box to select multiple files. This is useful when downloading multiple files to PROMJet.

File Format

The user can choose the format of the data file by simply pulling down this box and clicking the desired format. PROMJet software currently reads and writes Binary, Intel Extended Hex, and Motorola S-Record data files. The dialog box can also specify script as a file format. By choosing this format, the software assumes that the file name and path in the File Name edit box is a script file and it will attempt to execute it if the user facilitates the Script button.

File Offset

An unsigned offset value to be subtracted from the start address of a data file while reading it or added to the start address while writing a hexadecimal data file (Intel or Motorola). This offset can take an UNSIGNED hexadecimal value between 0 and 0xFFFFFFFF. Such an offset can be used to shift the start address of a file to be adjusted to the memory start address of PROMJet (e.g. if the target system has PROMJet in location 0xE0000 and an Intel hex file is used with a start address of 0xE0000, specifying E0000 as an offset will result in downloading the data to address 0 of PROMJet).

Data Length

This field specifies the length of data to be read from or written to PROMJet or a data file. This is a hexadecimal value that can vary from 0 to 0xFFFFFFFF.

Memory Address

Sets the start address of PROMJet to the specified value between 0 and 0x4000000. If the width parameter is higher than 8, PROMJet start address will be divided by the bus width (in a 16 bit system a memory start address of 20 will correspond to a PROMJet start address of 10 since the data is split in high and low bytes).

WBR Address / SPI ID Data

This parameter is used to set the address of the Access Word for the Write-By-Reading option that allows the target processor to communicate with the host by only reading PROMJet (see the user's manual for more information). Please note that the WBR address will be set relative to the beginning of PROMJet and not to the beginning of the processor's memory (e.g. if PROMJet is located at address 0xE0000 in the target system, specifying 300 will locate the Access-Word in address 0xE0300 relative to the beginning of the target's memory). The first two bits of the address will be set to 0 for long-word alignment. If the width parameter is higher than 8 bit, the software will divide the Access-Word address by bus width. In SPI mode, this field is used to program the 24-bit RDID chip code (see SPI commands below).

Trigger Address

This parameter is used to set the address of the Trigger circuit inside PROMJet. Please note that the trigger address will be set relative to the beginning of PROMJet and not to the beginning of the processor's memory. If the width parameter is higher than 8 bit, the software will divide the trigger address by bus width. Adding an R or W after the address will configure PROMJet to trigger on reads or writes only. An S at the end of the address will activate the single shoot mode where the INT line will stay active until the user presses the WBR/Trig bottom to reactivate the circuit.

USB, LPT or TCP/IP Port

The port used to communicate with PROMJet. If using a USB device, the user should write USB in this field. The parallel port value can be 1 (for parallel port at IO address 0x3BC), 2 (0x378) or 3 (0x278) depending on the parallel port used. The user can also input the IO address of the parallel port by simply typing the hexadecimal value of the port address without using an 'X' before the value (e.g. B400). By specifying 0 as a port address, the software will search all available standard parallel ports and USB ports in the following order: 0x3BC then 0x378 then 0x278 and finally all USB ports. If the user has multiple PROMJets connected to multiple ports in a single PC and is using 0 as a parallel port address, the software will always use the PROMJet(s) in the first available port. If Ethernet communication is chosen, the value of this edit box gives the TCP/IP port used to communicate with PROMJet. The Ethernet adapter uses port 1000 for PROMJet communication.

Fill Value

This is the byte value used to fill unused memory locations when downloading a data file to PROMJet or viewing data with the binary editor. This field accepts a value between 0 and 0xFF. This field also specifies the fill value used when executing the fill command by pressing its button.

Start Position

This field is used to specify a PROMJet when multiple devices are connected to the PC. If a USB connection is used, it specifies which unit to operate if multiple USB PROMJets are connected to the same PC (1 to 16). In the case of parallel port operation, it specifies the first PROMJet in a number of daisy-chained units connected to a parallel port or an Ethernet adapter (using older style PROMJet devices that support daisy-chaining). The first PROMJet in a chain (the one closest to the port) has a position value of one; the one next to it has a value of two etc. The user can choose PROMJet position by either pressing the up/down arrow or entering the desired value using the keyboard.

Memory Size

This edit box is used to specify the size of the emulated EPROM or FLASH device. It is also used to configure the hardware jumpers of PROMJet to adjust the emulated memory size. The user can choose any capacity between 8K Byte and 32M Byte. This is per byte size. So to emulate a 29F040, the size will be 512 KB and the Bus Width is 8. For 29F400 devices, the size will be 256 KB and the Bus Width is 16.

Bus Width

This option gives the desired bus size in bit and can accept the following values: 8, 16, 32, 64, 128, 1 for SPI and 4 for LPC. Current PROMJets support data width up to 16 bits. Values of 32 bit and higher are used when daisy-chaining multiple old-style devices. If the values of this edit box are higher than 16, the software will split the data between multiple PROMJets when downloading and will collect it when uploading a data file from multiple daisy-chained PROMJets. To support the SPI or LPC interface, the user must choose the SPI or LPC mode. See the SPI and the LPC sections for more info.

Network Host Address

This field is only active when the Network flag is checked. In this case the communication with PROMJet will go through the Ethernet instead of a parallel port. This field specifies the address of PROMJet Ethernet adapter. When network communication is chosen, the parallel port edit box is used to specify the address of the TCP/IP port on the Ethernet adapter used by PROMJet.

Verify Flag

By checking this flag, the software will verify the contents of PROMJet after downloading data to it. Please use this option sparingly since it slows the download process.

Fill Flag

This switch determines whether to fill the unused memory locations when downloading a data file to PROMJet. If this switch is not checked when downloading a data file, the unused memory locations will not be written to PROMJet. This switch has no effect when executing the fill command by pressing the Fill button.

Flash Flag

Checking this flag will specify a FLASH memory pinout at PROMJet's 32-pin DIP connector. In this case pin 31 of PROMJet DIP connector will serve as a write input and pin one will serve as address 18. The write-input signal in the feature connector on the top of the PROMJet will serve as address 19. **Please leave this check box unchecked if you are using the 50-pin high-density connector instead of the DIP connector.** Consult the PROMJet user's manual for more information about EPROM and FLASH pinout configuration for the Dip connector.

ICE Flag

Checking this flag will instruct the selected PROMJet to enable the ICE feature. PROMJet unit(s) used must have the ICE option installed to allow this feature to work. Enabling the ICE option allows the user to read and write emulation memory on the fly while the CPU is executing from the PROMJet memory. The does not require any arbitration circuit between the PROMJet and the target CPU.

Cable Flag

Checking this flag will instruct the selected PROMJet to enable the noise circuit inside PROMJet. Check this box only if you are using an extension cable between PROMJet and your target board or having any difficulty with the ICE option. PROMJet must have the ICE option installed to allow this feature to work.

WBR Flag

This switch determines whether to activate the WBR access word. If this flag is checked and the Mode button is pressed, the software will enable the access word inside the PROMJet pointed to by the value in the start position edit box.

Trigger Flag

If this flag is checked and the Mode button is pressed, the software will enable the trigger circuit inside PROMJet. The trigger circuit enables the user to watch for an access to a memory location inside PROMJet. Once the CPU accesses the memory location, the INT line will be activated.

Restart Flag

This flag causes the software to activate the reset output signals when accessing the emulation memory of PROMJet. If this flag is checked when downloading a data file, the software will also configure the hardware jumpers to adjust the memory size as specified in the Memory Size edit box. It will also set other configuration options like FLASH, ICE and Cable according to their check boxes. It will also set the Access word of the Write-By-Reading and trigger function. This spares the user the need to press the Download and then Mode button after downloading a new data file. When using a PROMJet without an ICE option, the user can access the PROMJet memory without resetting the target processor.

Network Flag

This flag, when checked, routes all the communication between the PC and PROMJet through the Ethernet instead of using USB or parallel port. In this mode the Network Host Name edit box will be enabled to enter the host name or address. Also the parallel port edit box will point to the TCP/IP port used by PROMJet in the Ethernet adapter and should be set to 1000. When using a DebugJet with multiple PROMJet ports, the unit plugged in port I is unit 1 and in port II is unit 2. If more units are cascaded to the unit in port I using old style Data-In and Data-Out connections, their number is 2, 3 and so on followed by the units connected to port II. So for example if the user has 2 units connected to Port I and 3 units connected to Port II, then first unit in port I will be 1 and the one cascaded to it will be 2 followed by the first unit connected to port II as number 3 and the ones cascaded to it as 4 and 5 consecutively.

Verify Flag

By checking this flag, the software will verify the contents of PROMJet after downloading data to it. Please use this option sparingly since it slows the download process.

Fill Flag

This switch determines whether to fill the unused memory locations when downloading a data file to PROMJet. If this switch is not checked when downloading a data file, the unused memory locations will not be written to PROMJet. This switch has no effect when executing the fill command by pressing the Fill button.

Partition Flag

This flag instructs the software to take a large file and cut it among multiple PROMJet sets (for example a 2Mbyte file can be written in 2 PROMJets each has 1 Mbyte; The first unit will store the first MByte (x0-x100000) and the second will store the last). This feature will also work if the word size is higher than 8-bit and it is only supported with daisy-chained devices using parallel or Ethernet connection.

Merge Flag

This flag, when checked, will instruct the software to read multiple files and write them to PROMJet. This is useful if the user has an image consisting of multiple files (e.g. code and data) each is located in a different memory area. Instead of executing multiple downloads, this feature will do a single download for all the files. The user can enter multiple file names by pushing the '+' bottom next to the file name field.

Swap Flag

This flag is used to switch PROMJet between little and big Endean modes. If it is unchecked, the first PROMJet in a set of units will have the MSByte and the last one will have the LSByte. By checking it, the first PROMJet will have the LSByte. If this flag is used with a width of 8-bit, the software will split the file in high and low bytes. The low bytes will be written first followed by the high bytes. This feature is useful when emulating 16-bit FLASH memory (e.g. 28F800) configured to work in 8-bit mode.

ByteWide Flag

This feature is used only with 16 bit PROMJets to configure it as an 8-bit unit. It is useful if a user needs to split a data file among multiple PROMJets each configured as an 8-bit device.

Record Flag

The user has to check this flag to start recording a script file. Once this flag is checked, the software will request the user to enter a script file name and will then start recording it. The record mode will end when the user clears this check box at the end of the recording. For more information about recording and executing script files, refer to “Using Scripting Language” in this help file.

Progress Bar

The progress bar in the middle of the dialog box displays the status of the current command. The software is idle when this bar is empty. Otherwise the length of the bar indicates the progress of the executing function.

Windows Software Operation

Downloading a Data File to PROMJet

One of the most used functions in this software is downloading a data file to PROMJet to start emulating a memory device. To do this, the user has to enter the name of the data file in the appropriate edit box and then set the file format, File offset, Data length, Memory address, PROMJet size, width and start position. Any optional parameters such as WBR address, Fill value and Fill switch must be set before downloading the data file. The user can then download the data file by simply pressing the Download button in the dialog box. The button will stay pressed as long as the software is downloading the data and will pop back when finishing the download process. If the Restart check box is checked, the software will also reset the target system, set the configuration of PROMJet emulation memory and also set the address of the WBR and trigger circuits.

If the user chooses multiple files in the file edit box and the software finds more than one set of units connected to the same port, it will execute a parallel download. Such a function will write the first file to the first set of units (depending on the width), the second file will be written to the second set and so on. This feature will also work in 16, 32 and 64 bit modes. The total number of PROMJets needed cannot exceed 16. It is only supported with daisy-chained devices using parallel or Ethernet connection.

Uploading a Data File from PROMJet

This command is used to upload the contents of PROMJet into a data file. As with the download command, the user has to enter the name of the data file, File format, File offset, Data length, Memory address, PROMJet size, width and start position. If the user specifies a file name that already exists, the software will ask the user to confirm to overwrite the old file or to choose a new file. When the Upload button is pressed, the software will upload the data from PROMJet and the button will stay pressed until the data is transferred. If the Restart check box is checked, the software will also reset the target system while uploading the data. If the File Offset parameter is not 0 and an Intel or Motorola file format is specified, the software will add this offset to the address of the data in the hex file.

Comparing a Data File with PROMJet

To compare the contents of a data file with the contents of PROMJet's memory, the user has to activate this command by pressing the compare button. As with the download and upload commands, the user has to enter the name of the data file, File format, File offset, Data length, Memory address, PROMJet size, width and start position. The software will start comparing the file with PROMJet and it will return a success message in case of a successful comparison or an error message with the first memory address that did not match with the data file in case of a compare error.

Filling PROMJet with a Value

This command is used to fill a block of PROMJet memory with a given value. In this case the user has to enter only the Data length, Memory start address, Fill value, PROMJet size, width and Start position. The software will start writing the fill value to PROMJet memory starting at the specified start address. The length of the data written will be determined by the size of PROMJet or the specified data length whichever is smaller. The data length will be divided by the width parameter (e.g. if the length parameter is 8 and the width parameter is 32 bit, the software will write only two words of 32 bit each). To fill multiple blocks in different, non-consecutive memory locations, the user has to run the fill command multiple times or use a script file to do so.

Editing PROMJet memory using binary editor

The windows software has a binary editor similar to the DOS editor. The user starts the editor by pressing the PJ watch bottom. Figure 3 displays the binary editor window.

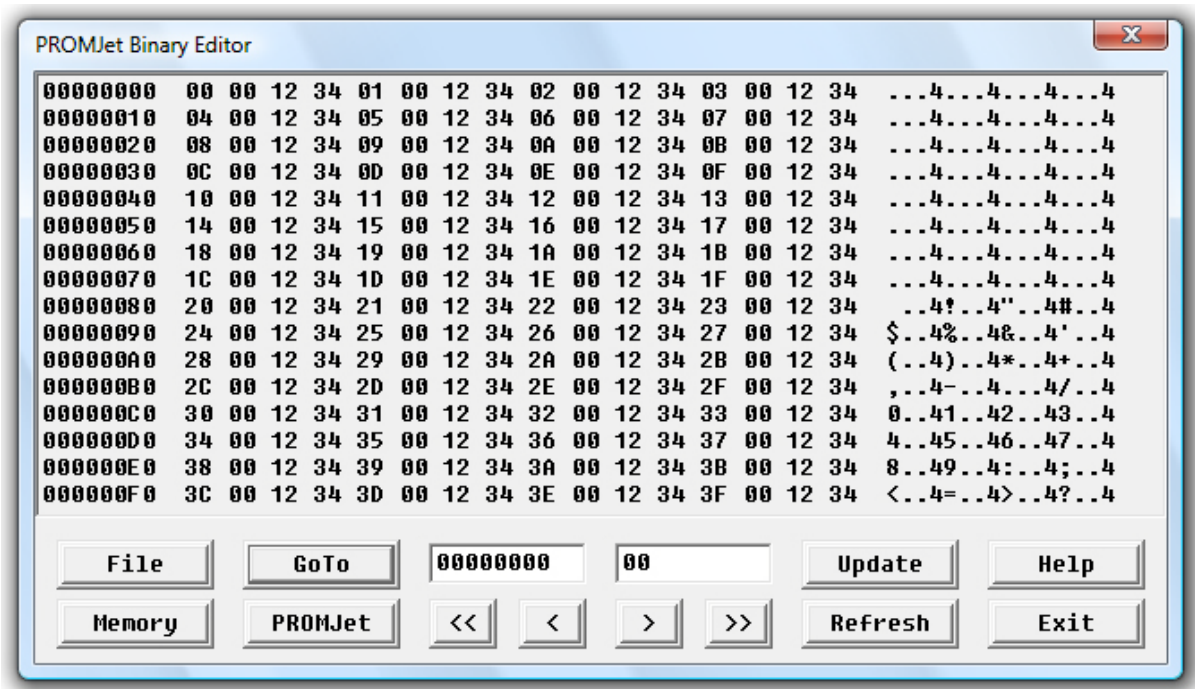


Figure 3: Windows binary editor.

At the top of the window, the data is displayed in both hex and ASCII formats. At the left of the main window, the address of the data is presented in hexadecimal format. The editor has also an active address and data edit boxes that are located in the middle of the screen below the main window. The user can select the active address by pointing the mouse to the byte location in the main window and clicking the left mouse bottom. Once the user selects an active byte, the address and data edit boxes will display the valid information of this memory location. The user can also go directly to the address edit box and type any valid address and press the GoTo bottom and the editor will display this address. To change the data in a memory location, the user can enter the address of this location (by using the

mouse or the keyboard) and then enter a series of hexadecimal characters (up to 256) in the data edit box and then press the Update bottom. The editor will write the data starting from the active address.

The page up(<<), line up(<), line down (>) and page down (>>) bottoms located under the active address and data edit boxes are used to move the editor window in the memory space. The Refresh bottom is used to update the data in the binary editor window with the current data in the PROMJet. This should be used with the processor in RESET if the user does not have an ICE unit or while the processor is running if the user has an ICE unit.

The File bottom manages data transfer between the editor's memory and data files. By pressing this bottom, the editor will open a new dialog box that allows the user to read, write, and compare data file with the current memory contents. Before executing these functions, the File format, Memory start address, File length and offset must be entered in the corresponding edit boxes.

Pressing the Memory bottom accesses memory functions in a new dialog box. The user then can search for an ASCII or hex data, replace it, fill or move a memory block. The user has to specify the "From Address" and the "Data Length" edit boxes. In the case of moving data, the "To Address" edit box has to be also specified.

The PROMJet functions can be found by pressing the PROMJet bottom. The user can use this dialog box to upload, download, and compare data with the PROMJet. The Reset check box is used to let the software reset the target system while executing these functions. The user can also reset the target system by just pressing the Reset bottom. While transferring data with PROMJet, the user must specify the "Start Address" and the "Data Length" edit boxes to determine the location and amount of data to be transferred.

Editing data files using binary editor

To edit the contents of a data file, enter the file name and parameters in the corresponding edit boxes. Start the binary editor by pressing the "Edit File" button. The binary editor will start and will display the contents of the data file selected. Use the binary editor as described above under editing PROMJet contents.

Setting PROMJet configuration

This function is used to set PROMJet memory size and mode. When executing this function by pressing the Mode button, the software will set the hardware jumpers inside PROMJet to emulate the specified memory device. This function will also set the FLASH, ICE, Cable, WBR and trigger parameters. The software will also activate the Reset output signals while executing this function.

Trigger logic Setup

Pressing the Trigger button in the dialog box activates this function. The software will set the trigger logic inside PROMJet according to the current parameters. The user has to connect the INT or /INT lines of the PROMJet to the appropriate logic to take advantage of this function.

Using PROMJet to Reset the Target Processor

If the user needs to restart the target system, he has to press the Reset button. The software will activate the reset outputs of PROMJet for 0.001 of a second and then deactivate it to let the target processor start again.

Recording a Script File

This feature is useful if the user has to execute more than one command multiple times. After storing the script file, the user can call it at any time to perform these functions. Here are the steps needed to record a script file:

1. Check the record check box. If there is a file name in the edit box and the file format is Script, the software will check if the file exists and if it does it will ask the user to confirm overwriting it. If the File format is not Script, the software will start the familiar file-save dialog box that enable the user to specify the script file name and path.

2. The user has to adjust all the parameters needed for the function he needs to execute as described in the above sections and then press only one of the following buttons: Download, Upload, Compare or Fill. If the user pushes more than one button, the system will store the last pressed button. If the user needs to download multiple files, he has to do this in different steps. The user can additionally press any combination of the following buttons Mode, WBR and Reset without starting a new step.

3. After pressing the desired function button(s), the user must press the script button to signal the software that the current step is completed. The system will then start recording a new step.

4. The user has to repeat the steps 2 and 3 until entering all the commands needed. At the end of the last command, the user must clear the record check box to signal the system that the record secession for this script is done. The system will then store the script file to the disk drive.

After storing a script file, the user can execute it any time he needs to run the command set stored in it. It is always helpful to call the script file with the name of the project it is used for to enable the user to recognize it. Next section explains how to execute the script file from Windows or DOS.

Executing a Script File form Windows or DOS

The user can execute a script file either from within windows or from a DOS window under Windows 95/NT. To execute a script file under windows, simply press the script button in PROMJet dialog box. If there is a file name in the edit box and the file format is Script, the software will attempt to read this file and execute it. If there is no file or the file format is not script, the software will open the open-file dialog box to allow the user to choose the file name from any available disk drive. After choosing the file name, the software will attempt to execute the script file and will display a message only in the case of an error.

To execute a script file from a DOS window or from a batch file running under DOS, the user has to type PROMJET <SCRIPT FILE NAME>. Where PROMJET is the name of PROMJet windows

software and SCRIPT FILE NAME is the name of the script file to be executed. If either of the files is not in the current directory, the user has to specify the whole path to the file. The script file name can be a long file name matching the windows 95/NT specification.

Updating PROMJet Firmware

PROMJet windows software has the latest PROMJet firmware. After the user downloads and installs the latest software from EmuTec website and start using it with a PROMJet via the USB port, the software will check if the firmware built into PROMJet is older than the one recently downloaded. If this is the case, the software will prompt the user to update PROMJet with the latest firmware by pressing the HELP button. Once the user does that, the software starts a firmware update process if PROMJet is not connected to a target system or if the target system is turned off. Otherwise the software will instruct the user to turn off the target system. The PROMJet light will turn RED while the upgrade is going on. Once the upgrade is done, the light will turn back to Green and the PROMJet windows software will exit. The user has to unplug the PROMJet USB cable and plug it back in. If the user connects PROMJet via the parallel port, this step is not necessary.

Operating Instructions for Linux

The PROMJet Linux software “Pjet” is used to operate PROMJet from a host computer running Linux OS via a USB or Ethernet connection. Printer port is not supported under Linux at this time. The software enables the user to download, upload, compare and setup PROMJet mode and size.

Before operating PROMJet, make sure that the unit is supplied with power either from the USB cable or from an external power supply and is connected to the PC via the provided USB or Ethernet cable via DebugJet. The supplied Pjet software should be first copied into the working directory or to any other directory included in the PATH statement of the host system. Please consult the Linux user’s manual for more information.

The Pjet software does support up to 16 PROMJets from a single USB or Ethernet port. It also supports targets with up to 128 bit wide data buses. The user can operate the PROMJets in any suitable configuration. For example if the user has 8 units he can either configure them as a single memory bank for a 64 bit target, or as two banks for a 32 bit target, or as four banks for a 16 bit target, or as eight banks for an 8 bit target. The user can also choose different data-width combinations for different processors in a single target system (e.g. one 32-bit bank, one 16 bit bank and two 8 bit banks).

The user starts the software with specific commands and parameters from the system prompt or from a batch file. The software executes the command and returns a completion or an error message in case of error. The software also returns an error level code to be used with a batch file to check the completion of the operation.

Linux Software operation

To start the Pjet program, type the below command on the Linux prompt:

```
pjet {Command} {Optional parameters} {Data File} ↵.
```

The use of commands and optional parameters is explained below. If no command is specified, the program will **download** the data file specified at the end of the command line to PROMJet. If no data file name is present, the program will execute any optional parameters specified in the command line such as resetting the target system.

To download a data file to PROMJet, the user does **not** have to specify a command but has to specify either the memory size or the data length via the I, S and/or the L parameter. Also the file format, if not binary, may be specified by the T option. The O parameter can be used to subtract an offset from the file start address. Use the A option to specify a memory address where the data will be downloaded if it is not 0. The F parameter is used to specify the value of the unused memory locations in an Intel or Motorola hex file. The user must specify a file name to download.

On-line help is available by simply typing Pjet followed by the ENTER key. The Pjet software can execute only one command per execution. A “Command” can be any one of the following (**NO** command is needed to download/write a data file to PROMJet):

- NO CMD** Download a data file. This is the default command if there is no command specified.
- B** This command is used to fill PROMJet memory with a given value. The user has to use the A parameter to specify PROMJet start address, the L and/or S parameter to specify the length of the memory block and the F parameter to provide the byte-value to be written to PROMJet (see below for how to use these parameters). The W parameter is used to specify the data width or the number of PROMJets to be written to. If the length parameter is used, it will be divided by the width parameter (e.g. if the length parameter is 8 byte and the width parameter is 32 bit, the software will write only two 32-bit words). Also the A parameter will be adjusted if the width parameter is higher than 8 bit (e.g. in a 16 bit system a memory start address of 20 will correspond to a byte start address of 10 since the data is split in high and low bytes). To fill multiple blocks in different, non-consecutive memory locations, you must run the Pjet software multiple times.
- C** To compare the contents of a data file with the contents of PROMJet memory, the user has to specify this command. The user may optionally use the A parameter to specify the PROMJet start address, the W parameter to specify the data width, the S and/or L parameter to specify the length of the data to be compared, the O parameter to subtract an offset from the file start address and the F parameter to specify the value of the unused memory locations in an Intel or Motorola hex file. The software will start comparing the file with PROMJet and it will return a success message in case of successful comparison or an error message with the first PROMJet address that did not match with the data file in case of a compare error.
- R** This command is used to upload the contents of PROMJet into a data file. The user has to specify either the memory size or the data length via the S and/or the L parameter. Also the file format, if not binary, may be specified by the T option. If the O option is used with an Intel or Motorola hex file, the software will **add** this offset to the address in the hex file. If the user specifies a file name that already exists, the software will **overwrite** it with the new file.
- U** This command is used to Upgrade PROMJet firmware. After receiving new PROMJet software updates, the user may use this command to upgrade PROMJet if a new firmware is available.

Optional parameters are used to change the program defaults. The command line can include all of the parameters that have to be changed. The program will use the default values (specified in ***BOLD-ITALIC***) for unused parameters.

- A=0** Sets PROMJet memory start address to a specified value between 0 and 0xFFFFFFFF. If the Pjet software is used with 16 bit or wider data buses, PROMJet start address will be divided by the width (e.g. in a 16 bit system a memory start address of 20 will correspond to a byte start address of 10 since the data is split in high and low bytes).
- D= S/Adr** This parameter is used to set the address of the Access-Word for the Write-By-Reading option that allows the target processor to communicate with the host only by reading PROMJet (see appendix for more information). This is also used to set the SPI 24-bit ID value. This option can be combined with any command (download, read etc.). The parameter begins with a switch (0 or 1) to enable or disable the Write-By-Reading mode. If the switch is set, the user has to provide the Access-Word address after the switch. Please note that the address is **relative to the beginning of PROMJet and not to the beginning of the processor's memory** (e.g. if PROMJet is located at address 0xE0000 in the target system, specifying D=1300 will locate the Access-Word in address 0xE0300 relative to the beginning of the target's memory). When using the W option to specify a data width higher than 8 bit, the software will divide the Access-Word address by the number of bytes to adjust the word address. Use the X parameter to execute this option for only one PROMJet. To set the CPI 24-bit Chip ID, simply add D=1{CHIP ID}.
- F=0xff** Fill switch to specify the value for unused memory locations. It accepts a value between 00 and 0xFF. This parameter, if used with the B command, will specify the fill value for the memory block.
- H=S/A{RWS}** This parameter is used to set the address of the trigger circuit. The parameter begins with a switch (0 or 1) to enable or disable the circuit. If the switch is set, the user has to provide the address after the switch. Please note that the address is **relative to the beginning of PROMJet and not to the beginning of the processor's memory** (e.g. if PROMJet is located at address 0xC0000 in the target system, specifying H=1300 will locate the Access-Word in address 0xC0300 relative to the beginning of the target's memory). When using the W option to specify a data width higher than 8 bit, the software will divide the address by the number of bytes to adjust the word address. While adding an R at the end of the address will activate the hardware trigger for read access only, adding a W will do it for Write access only. Adding an S at the end will do a single shot so the trigger circuit will be active after the first access until reactivated by the host software.
- I={FCNPL}0** The command configures the hardware jumpers to set the memory size of PROMJet. The value for this option (in Kbit/Mbit) can be 64 (I=64 for 27C64), 128, 256, 512, 10, 20, 40, 80, 160, 320, 640, 1280, 2560 and 5120 depending on the emulated memory size. Adding an F before the memory size will specify an FLASH memory pinout at PROMJet's DIP socket. In this case pin 31 of PROMJet socket will serve as a write input and pin one will serve as A18. The write-input signal in the feature connector on top of PROMJet will serve as address 19 (see appendix for more information). Be

aware that when using this option, the reset output signal of PROMJet will be activated to restart the target processor after changing the memory size. While a P will configure PROMJet to support the SPI interface, an L will activate the LPC/FWH interface instead of the parallel interface. Adding an N will disable the ICE feature and adding a C will switch on the cable mode.

- L=Max** The length of the data to be transferred between PROMJet and a data file. This is a hexadecimal value that can vary from 0 to 0xFFFFFFFF. If this option is not used, the software will assume the maximum length set by the S or I option.
- O=0** An unsigned long offset value to be subtracted from the start address of a data file while reading it or added to the start address while writing a hexadecimal data file. This offset can take an UNSIGNED hexadecimal value between 0 and 0xFFFFFFFF. Such an offset can be used to shift the start address of a file to be adjusted to the memory start address of PROMJet (e.g. if the target system has PROMJet in location 0xE0000 and an Intel hex file is used with a start address of 0xE0000, specifying O=E0000 will result in downloading the data to PROMJet address 0).
- P=USB/IP** The P option is used to specify which port to use for communication with PROMJet. The USB port value can be 1-16 depending on how many PROMJets are connected to the system and which PROMJet is used. If the P option is not specified, the software will use the first USB unit. If PROMJet is connected via an Ethernet adapter, an IP address can be used. The software can accept either a dotted format (P=192.168.1.4) or a server name (P=debugjet23).
- S=Min** This command is used to specify the length of data (in Kbit/Mbit) to be uploaded, downloaded or viewed on the screen. The value for this option can be 64 (S=64 for 27C64), 128, 256, 512, 10, 20, 40, 80, 160, 320, 640, 1280, 2560 and 5120 depending on the memory size. This option can be used instead of the L option if the data size is a standard size (e.g. 64 Kbit for s=64).
- T=B** Type of data file to be used. The Pjet program accepts and generates Binary, Intel Extended Hex, and Motorola S-Record data files. Available options for this parameter are B, M, and I.
- V** This option is used to verify the contents of PROMJet memory after downloading. Please use this option sparingly since it slows the download process.
- W={S} 8** This option gives the used word size in bit and can accept the following values: 8, 16, 32, 64 and 128 bit. If this option is used with values higher than 8 bit, the value of the A option will be divided by the number of bytes to generate the start address since the data is split in multiple bytes. Adding an S before the width value will cause the program to switch between little and big Endean modes.
- X={B} 1** This option is used to select a smaller group of PROMJets from a number of units cascaded in a chain. While this option specifies the first PROMJet to be selected, the W option specifies how many units will be selected thereafter. The first PROMJet in a chain (the one closest to the host) is always number one (X=1), the one next to it is

number 2 etc. Using B after the equal sign will instruct the software to use the PROMJet(s) in 8-bit mode only. This is useful when using 16-bit PROMJets to work in 8-bit mode. This can be also used to select which USB port to use if multiple PROMJets are connected to a single host unit (e.g. x=1 for first unit, x=2 for next one, etc...).

After specifying a command and the options, the user might indicate the name of a data file. If the data file is not in the current directory, the path to the data file must be included with the file name.

When using a PROMJet **without** the ICE option, **the user must reset the target processor when accessing PROMJet**. This can be done by specifying the “T” option while executing any command.

Examples

Before using your PROMJet under Linux via a USB connection, make sure you have permission to read and write the USB devices under the directory `/dev/bus/usb/*`. The USB endpoint files under this directory should have the `0x666` file attribute to allow all users to use them. Contact your system admin for more info. Also make sure `libusb.so` is installed on your PC.

In this section, we will present some examples to demonstrate the functionality of the Pjet software. If you have the ICE option and would like to leave the target running while executing any of the commands, do not enter the `I` option in the command line (except in cases when you have to change the memory size or when you need to restart the system).

Download a file in Motorola S format to a 1MBit FLASH (29F010) using Ethernet:

pjet P=192.168.1.4 I=F10 T=M project.hex ↵

Download a file in Intel Hex format to a 1MBit EPROM (27010) using USB port:

pjet I=10 T=I project.hex ↵

Download 0x100 bytes to address 0x150 from of an Intel hex file at offset 0x300 to a FLSAH setup (29F010) and set WBR to 5800:

pjet I=F10 T=I L=100 O=300 A=150 D=15800 project.hex ↵

Download a binary file and configuring the unit for an SPI flash memory (25F040):

pjet I=P40 project.hex ↵

Download an Intel hex file and configuring the unit for an LPC flash memory (49F080):

pjet I=L80 T=I project.hex ↵

Compares 0x1000 bytes starting at memory address 0x2000 after taking a 0x300 offset from the file:

pjet C T=I A=2000 O=300 L=1000 project.hex ↵

Using PROMJet with FTP Software

To use PROMJet from an FTP session, the user has to use a PROMJet Ethernet adapter (DebugJet). Check with your LAN administrator to assign an IP address for DebugJet. Connect DebugJet's CFG port to a dumb terminal using the supplied 6-wire RJ-11 cable and the 9 pin female modular connector. Set the dumb terminal parameters to 96-8-N-1. After switching DebugJet on, it should respond with a message displaying firmware version and serial number. Type Yes to start setting-up the Ethernet parameters and follow the setup menu. At the end of the menu, type FLASH to store the new parameters in the FLSAH memory. Recycle DebugJet power to allow it to boot using the new parameters.

After setting Ethernet parameters, connect DebugJet to an Ethernet hub using an eight-wire RJ45 cable. Using the supplied download cable, connect PROMJet to the Ethernet adapter. Connect PROMJet to your target system as described earlier in the hardware installation section of this manual.

Start an FTP session in your host system by typing FTP from the command line. The FTP software will respond with an FTP prompt. We will assume an IP address of 124.170.92.200 as a DebugJet IP address. Open the communication with PROMJet by typing:

ftp> open 124.170.92.200 and PROMJet will respond with the following message:

```
Connected to 192.168.1.200.  
220 DebugJet FTP Server Version 2.0. (IPaddress: 192.168.1.200,Port: 21)  
User (192.168.1.200:(none)):  
230 Logged in  
ftp>
```

Now PROMJet is ready to accept and execute commands from the host system. There are currently three commands available to use with PROMJet. The commands are **Put** and **Get** and are described below.

PUT *data-file name* */optional parameter1/optional parameter2/...*

This command sends a data file to PROMJet. The optional parameters are separated by forward slashes (without using any spaces) and are used to control the download process. The user must use at least one parameter such as I, S or L to indicate PROMJet memory size. After finishing the download, PROMJet will respond with a message indicating the success of the command. To set PROMJet size, mode, WBR and trigger access word use an L=0 data length.

GET */optional parameter1/optional parameter2/...* *data-file name*

Receives a data file from PROMJet. As with the previous command, the optional parameters are used to control the upload process and at least one parameter must be used to indicate the memory size. After finishing the upload, PROMJet will respond with a message indicating the success of the operation.

The optional parameters are used to change PROMJet defaults. The command line can include all of the parameters that must be changed. PROMJet will use the default values (specified in ***BOLD-ITALIC***) for unused parameters. The usage of these parameters is the same as with the command line DOS software described earlier in this manual.

- A=0** Sets the start address of PROMJet to a specified value between 0 and 0x3FFFFFFF.
- D=S/Adr** This parameter is used to set the address of the Access-Word for the Write-By-Reading option that allows the target processor to communicate with the host only by reading PROMJet. The parameter begins with a switch (0 or 1) to enable or disable the Write-By-Reading mode. If the switch is set, the user has to provide the Access-Word address after the switch. Please note that the address is relative to the beginning of PROMJet and not to the beginning of the processor's memory. The first two bits of the address will be set to 0 to align the long word.). This is also used to set the SPI 24-bit ID value by adding D=1{CHIP ID} to the command line.
- F=0xff** Fill switch to specify the value for unused memory locations. It accepts a value between 00 and 0xFF.
- H=S/A{RWS}** This parameter is used to set the address of the trigger circuit. The parameter begins with a switch (0 or 1) to enable or disable the circuit. If the switch is set, the user has to provide the address after the switch. Please note that the address is **relative to the beginning of PROMJet and not to the beginning of the processor's memory** (e.g. if PROMJet is located at address 0xC0000 in the target system, specifying H=1300 will locate the Access-Word in address 0xC0300 relative to the beginning of the target's memory). When using the W option to specify a data width higher than 8 bit, the software will divide the address by the number of bytes to adjust the word address. While adding an R at the end of the address will activate the hardware trigger for read access only, adding a W will do it for Write access only. Adding an S at the end will do a single shot so the trigger circuit will be active after the first access until reactivated by the host software.
- I={NCFPL}0** The command configures the hardware jumpers to set the memory size of PROMJet. The value for this option can be 64 (I=64 for 27C64), 128, 256, 512, 10 (for 27010), 20, 40, 80, 160, 320, 640, 1280, 2560 or 5120. Adding an F before the memory size will specify a FLASH memory pinout at PROMJet's DIP socket (29F040). Adding an N or C after the equal sign will disable the ICE feature or switch to cable mode. While a P will configure PROMJet to support the SPI interface, an L will activate the LPC/FWH interface instead of the parallel interface.
- L=Max** The length of the data to be transferred to or from PROMJet or from a data file. This is a hexadecimal value that can vary from 0 to 0xFFFFFFFF. If this option is not used, the software will assume the maximum length.
- O=0** An unsigned offset value to be subtracted from the start address of a data file while reading it.

- S=Min** This command is used to specify the length of data to be uploaded or downloaded. The value for this option can be 64 (for 27C64), 128, 256, 512, 10, 20, 40, 80, 160, 320, 640, 1280, 2560 or 5120. This option can be used instead of the L option if the data size is a standard EPROM size (e.g. 64 K bit for s=64).
- V** By using this flag, PROMJet will verify memory contents after writing. Please use this option sparingly since it slows the download process.
- W={S}8** This option gives the used word size in bit and can accept the following values: 8, 16, 32, 64 and 128 bit. Adding an S before the width value will cause the program to switch between little and big Endean modes.
- X={B}I** This option is used to select a smaller group of PROMJets from a number of units cascaded in a chain (using old-style PROMJet versions that support cascading). While this option specifies the first PROMJet to be selected, the W option specifies how many units will be selected thereafter. The first PROMJet in a chain (the one closest to the Ethernet adapter) is always number one (X=1), the one next to it is number 2 etc. If 'X=0' or 'X' is specified, all PROMJets in the chain will be selected. Using a B will instruct the software to use the PROMJet(s) in 8-bit mode only. This is useful to configure a 16-bit PROMJet to work in 8-bit mode to provide a higher memory capacity.

To demonstrate the functionality of the previous commands and parameters, some examples are presented below.

To download a data file to a PROMJet configured in 16 bit mode starting at memory address 0x5000, set PROMJet size and mode to 1Mbit flash (29F010) and set the WBR word to 0x8000, use this command:

```
ftp> put project.dat I=f10/d=18000/a=5000/w=16 ↵
```

Use this command to download a data file to PROMJet memory starting at address 0, configuring PROMJet as an 8MBit SPI device with a chip ID of "012014" and restarting the target system.

```
ftp> put project.dat I=p40/D=1012014 ↵
```

After any of these commands PROMJet will send a message to the host system indicating the success or failure of the operation.

PROMJet Access and Communication Library

This appendix explains the use of PROMJet access and communication library. This library is used to integrate any software application with PROMJet enabling the software to read, write and configure PROMJet. The Windows libraries (PJUSBLPT.dll and PJetENet.dll) come in DLL format (Dynamic Link Library) to be linked with the user software at runtime. While PJUSBLPT supports operations via the USB and LPT ports, PJetENet.dll supports Ethernet connections. If an application needs to support all connections, it must load both libraries.

PROMJet library (pjetlib) is also provided for the Linux operating system. Only the USB and Ethernet connections are supported. The function calls are the same for both operating systems as stated below.

PROMJet library has two different types of functions: access and communication functions. Access functions are the ones used to write, read and configure PROMJet without the help of the target processor. They are very similar to the functions provided by the PROMJet software. These functions can be used with any PROMJet (with or without the ICE option) since it does not assume that the target processor is running. If the access functions are used with a PROMJet that has an ICE option, most of the functions (except reset target and configure PROMJet) can be executed while the target is running. The user can use these functions to read or write a block of data to and from PROMJet.

Communication functions, on the other hand, are used to establish a communication channel between the target and host computer. This channel can be used to transfer data between the two computers (e.g. run-time data for process control). These functions assume that the target is running and it requires that at least one PROMJet has an ICE option (to allow target and host to access the memory at the same time). While the access library software is executed only on the host computer, the communication software must run on the host and the target at the same time (since both computers have to transfer data between each other). The user can use both the access and the communication functions on the host computer at the same time (e.g. the user can use access functions to modify the contents of a lookup-table and communication functions to send and receive process-control data to and from the target processor).

There are four communication functions on the host computer to open a communication channel, read from a channel, write to a channel and close a channel. The host communication library can open and manage up to 16 active channels at the same time. The communication library on the target side is provided in plan 'C' language and has to be customized for the used target processor. It includes functions such as initialize a communication channel at a given PROMJet address, read from a specific communication channel or writes to a channel. As with the host software, the target functions can handle multiple active communication channels at the same time. We will explain now PROMJet library functions starting with the access functions followed by the communication functions on the host computer then on the target system.

Host Access Functions (USB/LPT)

The functions presented here are the access functions used to read, write, and configure PROMJet. They all start with the four letters ACCS followed by the function name. We will explain in this chapter how to use each of these functions.

unsigned long ACCSPJcheck (unsigned short ioadr)

Checks if one or more PROMJets are connected to a given USB or parallel port. This function takes the address of the PROMJet port, a value between 0 and 0xFFFF, as a parameter. To specify a **USB** connection, the ioadr parameter should be set to **0xFFFF**. All other values (0x0 – 0xFFFE) will specify a parallel port address. The default values for LPTBASE1, LPTBASE2 and LPTBASE3 define the PC parallel ports built on the main CPU board. If using a PCI adapter card, you may have to get the memory address from the control panel. If one or more PROMJets are found, the function returns the number of PROMJets daisy-chained on this parallel port in the lower 16 bit of the long word. Otherwise it returns an error message such as PROMJet not found or PROMJet is powered off depending on PROMJet status. The upper 16 bits of the long word are used as a 16-bit victor to show the type of PROMJets connected to this parallel port (bit 16 corresponds to the 1st PROMJet and bit 31 to the 16th PROMJet in the chain). If the bit is set, this means the corresponding PROMJet can work in 8 or 16 bit mode. The user should use this function before using any other library functions to check how many PROMJets, if any, are connected to given parallel port.

Void ACCSPJCloseUSB ()

Use this function to close all PROMJet USB connections or to recheck all USB ports after attaching a new PROMJet. On Windows, this function will be called when the program exits. On Linux, the user must call this function to close all open connections to PROMJets.

unsigned short ACCSPJread (unsigned short ioadr, unsigned short startpj, unsigned long address, unsigned short width, unsigned long length, void* buffer)

The user has to call this function to read PROMJet contents. The first parameter is the parallel port address as used with the previous function. The second parameter, startpj, determines the location of the first PROMJet in a chain of PROMJets to read from. If this parameter is 0, all PROMJets connected to this port will be activated and the software will start reading from the first PROMJet. If the BWIDEFLAG bit is set in the startpj parameter, the software will use each PROMJet in 8-bit mode only. The user can use this parameter with the width parameter to determine how many units will be accessed while executing this function. The third parameter is PROMJet memory address where the software starts reading. This address will be adjusted properly according to the width parameter. This means that if the width parameter is higher than 1, the address will be divided by the width to calculate PROMJet actual address in a multi-byte target. The width parameter is a value between 1 and 16 that gives the word size. If the SWAPFLAG bit is set in this parameter (width), the software will switch from little to big Endean mode (this means in a 16 bit mode, the even bytes will be written to the first PROMJet in a chain and the odd bytes to the next unit). The length parameter is used to specify the number of data bytes to be read to the data buffer pointed to by buffer parameter. A result is returned to the caller to indicate the success of the function. It is either no error or an error message with the appropriate error code (see the PJETLIB.H file for more information).

unsigned short ACCSPJwrite (unsigned short ioadr, unsigned short startpj, unsigned long address, unsigned short width, unsigned long length, void* buffer)

This function is used to write a block of data to PROMJet. The parameters and return values are the same as the ACCSPJread with the exception that this function writes the contents pointed by buffer to PROMJet starting at the memory address specified by the address parameter.

unsigned long ACCSPJcomp (unsigned short ioadr, unsigned short startpj, unsigned long address, unsigned short width, unsigned long length, void* buffer)

This function compares the contents of PROMJet with the contents of the data pointed to by the buffer parameter. It takes the same parameters as the two previous functions but has a different return value. If the comparison result is positive, the function returns no error. Otherwise it returns a compare error (bit 31 set) and uses the other 31 bits (bits 30...0) to present PROMJet memory address where PROMJet contents did not match with the data pointed to by buffer. If any other error has occurred, it returns an error message with the error bit set and an appropriate error code. To check the return value of this function, the user has first to check if the compare-error bit is set. If not, the user has to check the error bit (bit 15) to see if there were any other errors. This function, if called after a write function using the same parameters, can be used to verify the success of the write function.

unsigned short ACCSPJreset (unsigned short ioadr, unsigned short startpj, unsigned char function, unsigned short width)

The user has to call this function to reset the target processor (assuming that the reset signal of the target processor is connected to the reset signal of PROMJet). The function takes, besides the address of the parallel port, the location of the first PROMJet in a chain of PROMJets to be activated, the number of PROMJets to use and the reset function to be executed. Available functions are: ON to activate the reset signal, OFF to deactivate the reset signal and PULSE to activate the Reset signal for 1ms and deactivate it before returning. If the startpj parameter is set to 0, the software will activate the reset signal of ALL PROMJets connected to the parallel port. The width parameter can be used to determine how many units will be used. This is important if the cascaded units are used in two different target systems and the user has to reset one system without the other. Return values are either no error or an error code such as invalid function or invalid PROMJet location.

unsigned short ACCSPJinter (unsigned short ioadr, unsigned short startpj, unsigned char function, unsigned short width)

This function is used to activate the interrupt signal of PROMJet. The parameters and return values for this function are the same as by the reset function except in the case of the PULSE function where the duration of the pulse is one microsecond only.

unsigned short ACCSPJsetmod (unsigned short ioadr, unsigned short startpj, unsigned char mode, unsigned short size, unsigned short width)

This function is used to set the size and mode (EPROM / FLASH) of PROMJet. The mode parameter can be either EPROM or FLASH. By adding ICEOFF, the ICE-option will be switched off. The NOISEON flag will switch the cable feature on. SPION and LPCON will activate the PROMJet SPI or LPC interface. The size parameter is the memory size in Kbits or Mbits. Acceptable values are 64 (Kbit), 128, 256, 512, 10, 20, 40, 80, 160, 320, 640, 1280, 2560 and 5120. This is per byte size. So to emulate a 29F040 device, the size will be 40 and the Width is 8. For 29F400 devices, the size will be 20 and the Bus Width is 16. The other three parameters, ioadr, startpj and width, are the same as with previous functions. The startpj parameter determines the first PROMJet to activate and the width parameter specifies how many units thereafter will be configured. If the startpj parameter is set to 0, all the units in the chain will be configured. To use a 16-bit device in 8-bit mode, set the BWIDEFLAG in the startpj parameter. Return value is either no error or an error code such as invalid size or mode.

unsigned short ACCSPJsetwbr (unsigned short ioadr, unsigned short startpj, unsigned char startwbr, unsigned long address, unsigned short width)

This function is used to set the address for the Write-By-Reading access word inside PROMJet. As with the previous functions, the parameters ioadr, startpj and width are used to specify the parallel port, the location and number of PROMJets to be used. The address parameter is used to specify the address of the access word. If the last two bits are not 0, the software will ignore them to adjust the address to a word (4 bytes) boundary. The width parameter will not be used to adjust the address parameter as with other previous functions or as with the jet software. This means that if the user has a width of 2 and an address of 0x300, the access word will be set to 0x300 in both PROMJets. As with the last four functions, the width parameter, if not used, will default to 1.

unsigned short ACCSPJsettrig (unsigned short ioadr, unsigned short startpj, unsigned char tmode, unsigned long address, unsigned short width)

This function is used to set the address for the trigger logic inside PROMJet. As with the previous functions, the parameters ioadr, startpj is used to specify the parallel port, the location of PROMJets to be used. The address parameter is used to specify the address of the access word. The tmode parameter is used to set the trigger circuit mode. SINGSHOT is used to set it in Single Shot mode. ON is used for repetitive mode (with every CPU access PROMJet will activate the INT line). RDSSHOT, RDREPEAT, WRSSHOT and WRREPEAT will do similar functions for read and write accesses only.

Host Access Functions (Ethernet)

The functions presented here are the access functions used to read, write, and configure PROMJet via Ethernet. They all start with the letter N followed by the function name to distinguish it from the USP/LPT functions. The Ethernet functions have the same arguments and return same values as the ones described above. We will explain the additional functions as well as the differences between USB and Ethernet functions.

unsigned long PJConnect (unsigned char* PJipadr, unsigned short PJipport, unsigned long* PJsock)

This function is called to connect to a DebugJet device. Since this library can manage multiple simultaneous PROMJet connections controlled by one or multiple DebugJets, the software has to call this function as many times as needed to connect to all DebugJet units.

The first parameter, PJipadr, is a pointer to the Ethernet address of the DebugJet device to connect to. The function can accept either a dotted format (192.245.83.200) or a server name (debugjet1). In both cases the string has to be NULL terminated. The second parameter is used to identify the PROMJet port, which is 1000, by default (see DebugJet user's guide).

The function will either return a NO_ERROR and a socket handler in PJsock or an error code. If it returns a socket error, it should be reported to the user.

unsigned long PJDisconnect (unsigned long sock)

This function is used to close a socket connection previously opened by the PJConnect function.

unsigned long NPJcheck (unsigned long sock)

This function is used to check how many PROMJets are connected to the DebugJet port(s). It takes a valid opened socket as an input and returns same value as **ACCSPJcheck**.

unsigned short NPJread (unsigned long sock, unsigned short startpj, unsigned long address, unsigned short width, unsigned long length, void *buf)

This function is used to write PROMJet memory. It takes a valid opened socket instead of the port address as an input. It returns same values as **ACCSPJread**.

unsigned short NPJwrite (unsigned long sock, unsigned short startpj, unsigned long address, unsigned short width, unsigned long length, void *buf)

This function is used to write PROMJet memory. It takes a valid opened socket instead of the port address as an input. It returns same values as **ACCSPJwrite**.

unsigned short NPJwritever (unsigned long sock, unsigned short startpj, unsigned long address, unsigned short width, unsigned long length, void *buf)

This function is used to write PROMJet memory. It takes a valid opened socket instead of the port address as an input. It returns same values as **ACCSPJwrite**. Additionally it verifies the data after writing it to PROMJet.

unsigned long **NPJcompare (unsigned long sock, unsigned short startpj, unsigned long address, unsigned short width, unsigned long length, void *buf)**

This function is used to verify PROMJet memory. It takes a valid opened socket instead of the port address as an input. It returns same values as **ACCSPJcompare**.

unsigned short **NPJfill (unsigned long sock, unsigned short startpj, unsigned long address, unsigned short width, unsigned long length, unsigned short fillval)**

This function is used to fill PROMJet memory. It takes a valid opened socket instead of the port address as an input. Additionally it takes a fill value 0-ff to write to PROMJet.

unsigned short **NPJsetmod (unsigned long sock, unsigned short startpj, unsigned char mode, unsigned short size, unsigned short width)**

This function is used to set PROMJet mode. It takes a valid opened socket instead of the port address as an input. It returns same values as **ACCSPJsetmode**.

unsigned short **NPJsetwbr (unsigned long sock, unsigned short startpj, unsigned char startwbr, unsigned long address, unsigned short width)**

This function is used to set PROMJet mode. It takes a valid opened socket instead of the port address as an input. It returns same values as **ACCSPJsetwbr**.

unsigned short **NPJsettrig (unsigned long sock, unsigned short startpj, unsigned char tmode, unsigned long address, unsigned short width)**

This function is used to set PROMJet mode. It takes a valid opened socket instead of the port address as an input. It returns same values as **ACCSPJsettrig**.

Host Communication Functions (USB/LPT)

The functions presented here are the communication functions running on the host computer and used to send and receive data to and from the target system. Before presenting these functions, we would like to explain first the functionality of PROMJet communication channel. The channel takes advantage of the in-circuit emulation option that allows the host and the target to access PROMJet memory simultaneously. Each channel takes a block of bytes inside PROMJet emulation memory and uses it as a communication media. The first 16 bytes of the buffer are used to store a case-sensitive text string to mark the buffer start address. The default value of this string is: “PromJet-ICEBuf” that can be altered by the host and target software. If the user is using multiple channels, it is recommended to have different strings or an indexed string to distinguish between the different channels. Following the first 16 bytes are the flags, counters and data buffers that are used by the channel and will be described in the next section. We will now explain the use of the host library function.

unsigned short COMMPJopen (unsigned short ioadr, void __far *buffer, unsigned short startpj = 1, unsigned char intrent = 0, unsigned char intscnt = 1, unsigned long stradd = 0, unsigned long endadd = MAXPJADR)

This function is used to search for a communication channel in the specified PROMJet and opens it. The ioadr parameter determines the used parallel port as with the access functions. The second parameter, buffer, is a pointer to a text string used by the open function, as mentioned above, to search for the channel. The user can use the default string “PromJet-ICEBuf” or any text string that **matches** the same string in the target communication software (see below). The third parameter is used to specify the location of PROMJet, if multiple PROMJets are daisy chained. Fourth and fifth parameters determine if the target communication is interrupt driven or not. If the forth parameter is one, the receive function will activate the interrupt output of PROMJet after receiving a data package. The fifth parameter is used to determine whether to activate the interrupt output of PROMJet when sending data to the target (1 for interrupt, 0 for polling). The communication on the host side works only in polled mode. The last two parameters start address (stradd) and end address (endadd) specifies PROMJet memory space to search for the communication channel. If the user does not use the last two parameters, the software will search the whole memory space of PROMJet (from 0 to MAXPJADR). If the function finds the channel buffer, it returns a handle value used to read, write and close the channel. The user has to store this handle value to be able to use this communication channel. In case of error, the function returns an error message such as no channel found or all channels used.

unsigned short COMMPJwrite (unsigned short handle, unsigned short length, void __far *buffer)

This function uses the handle value returned by the COMMPJopen function to send data to the target via the communication channel. It takes the handle value as the first parameter followed by the length parameter that determines the length of the data block. The last parameter is a pointer to a buffer containing the data. This function returns the number of bytes sent to the target in this call (0 indicates no data was sent). If the intscnt parameter in the open function was 1 as the channel was opened, the send function will activate the interrupt output after sending the data to the target (the interrupt will stay active for ca. 1 micro second). If an error occurs, it returns an error code such as invalid handle.

unsigned short COMMPJread (unsigned short handle, unsigned short length, void __far *buffer)

This function is used to receive data from the target via the communication channel specified by the handle value. The second parameter, length, determines the maximum number of bytes to receive from the target. The data is stored into a data buffer pointed to by the forth parameter (buffer). The function returns the number of data bytes that has been received from the target (equal or less to the value of length). If the intrcnt parameter in the open function was 1 as the channel was opened, the receive function will activate the interrupt output after receiving the data from the target (the interrupt will stay active for ca. 1 micro second). In the case of error, an error code will be returned to the caller to specify the cause of error.

unsigned short COMMPJclose (unsigned short handle)

To close an open communication channel, the software has to execute this function. It takes the channel handle as a parameter and returns either no error or, in the case of error, an error message such as invalid handle.

Host Communication Functions (Ethernet)

If using PROMJet via an Ethernet adapter (such as DebugJet), a socket server is provided to translate the socket traffic to work with the Target communication function. The communication parameters can be setup using the CFG serial port of DebugJet. It provides a way to enter the PROMJet address, position, and interrupt control. The host can simply transfer the data to the target by opening the socket and sending and receiving the data via the TCP/IP socket. The Ethernet adapter will handle all the communication functions needed to send and receive data with the target CPU.

Target Communication Functions

The functions presented here are the communication functions running on the target CPU and used to send and receive data to and from the host computer. Before explaining the use of these functions, we would like to present first the layout and functionality of the communication channel on the target side. The channel takes advantage of the in-circuit emulation option to allow the host and the target to access PROMJet memory simultaneously. Each channel takes a block of bytes (from 64 to 480 Bytes) inside PROMJet emulation memory and uses it as a communication media. The memory buffer of the communication channel must reside inside a single PROMJet equipped with an ICE option 'OPT-ICE'. Also it is very important to note that the start address of the channel memory must be 16 byte aligned inside the used PROMJet.

The first 16 bytes of the channel memory are used to store a text string to identify the channel when the host attempts to open it. The 16 bytes thereafter are used to store the counters and flags used by the channel. Following the first 32 bytes is the send buffer used to send data to the host. This buffer length varies from a minimum of 16 bytes to a maximum of 224 bytes. The receive buffer, used to receive data from the host, follows thereafter and has the same byte count as the send buffer. We will explain now the organization of the communication channel memory as shown below in figure 4.

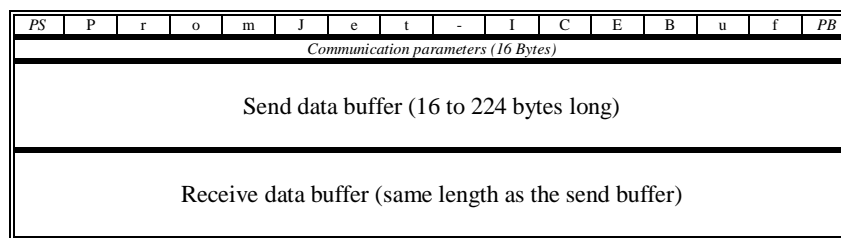


Figure 4: Memory layout of a communication channel.

The first byte of the channel memory (PS) is used as a scratch byte for the Write-By-Reading function and will be explained below. The 14 bytes thereafter are used to store a unique string. The open function in the host software will search for this string when it attempts to open the communication channel. This text string is case sensitive and must match the string pointed to by the buffer parameter in the COMMPJopen function. The default value of this text string is "PromJet-ICEBuf" that can be altered by the user on the host and the target side. Following the text string are some bytes used as counters and flags (byte 15 to byte 31) for communications.

After these bytes and at an offset of 32 the send buffer starts (target to host). It goes for the length specified by the channel length and is followed by the receive buffer which has the same length.

The target system can either use the write signal of the processor to modify the counters, flags and send data buffer or use the Write-By-Reading function of PROMJet to do so. If the target uses the write signal to modify channels memory contents, the location of the communication channel is irrelevant as long as it starts at a 16-byte boundary. If the target uses the WBR function, the first 32 bytes of the channel memory and the whole send buffer must be within 256 bytes boundary so that the target processor can modify them. The receive buffer can be outside this 256 bytes boundary since the target only reads it. This also explains why the send buffer has a maximum length of 224 bytes (16 bytes header + 16 bytes counter and flags + 224 bytes send buffer = 256 bytes which is the maximum length of a WBR block).

The first byte of the channel memory is used as a scratch byte and cannot be used to store any data. This byte could be randomly modified in case of an interrupt while executing the WBR code. The location of this byte can be modified to any other memory location within the 256-byte block by assigning the PJ_SCRT with a new offset. The software example that comes with the library locates the access word of the WBR function at the beginning of the channel memory. This location can be also changed by modifying the values of PJ_WBR0 in the define statements. The address of the access word for the Write-By-Reading logic must be set via the host software to allow the target to write to PROMJet memory.

After presenting the layout of the communication channel, we will now explain the target communication functions presented in the file "PJETCOMT.CPP". This file contains the following functions: PDVCSetup, PDVCSendReady, PDVCSendData, PDVCRecvReady, PDVCRecvData and PDVCWrByte that will be explained below.

unsigned long PDVCSetup (unsigned char *ChanBuf, unsigned long BufSize, unsigned short BusWidth, unsigned short TxInt, unsigned short RxInt, unsigned long GFlag, unsigned short TDelay)

This function is used to initialize the communication channel. It takes a pointer to the memory location of the channel buffer as its first parameter and the size of the send and receive buffer (between 16 and 244) as a second parameter. The third parameter is the BusWidth of the system (1 for 8, 2 for 16 and 4 for 32Bit). The Rest of the parameters are added for compatibility with DebugJet VCom Channels and have no meaning in this function. The function will write the data string used to identify the channel at the beginning of the channel memory and sets all communication flags to its initial values. If any one of the parameters is incorrect or the function cannot write to the memory location specified, it will return a zero. Otherwise it will return a channel handle to be used to send and receive data to the host.

unsigned long PDVCSendReady (unsigned long ChanHandle)

This function checks the available space in the send buffer. It takes the channel handle that was returned from PDVCSetup to specify the communication channel used. It returns the number of byte-space available in the buffer.

unsigned long PDVCSendData (unsigned long ChanHandle, unsigned short length, unsigned char *datbuf)

To send data to the host, the target processor has to execute this function that takes three parameters. The first parameter is a channel handle to specify the communication channel. The second and third parameters are the length and location of the data to be sent. As a return value, the function returns the number of data bytes written to the buffer.

unsigned long PDVCRecvReady (unsigned long ChanHandle)

This function checks the available bytes in the receive buffer. It takes the channel handle as a parameter. It returns the number of bytes available in the buffer.

unsigned long PDVCRecvData (unsigned long ChanHandle, unsigned short length, unsigned char *datbuf)

This function is used to read the received data from the host. It takes the same parameters as the send function and also returns a value indicating the number of bytes received from the host. As with the send function, the number of bytes received by this function can be equal or smaller than the length parameter.

unsigned short PDVCWrByte (unsigned char *datbuf, unsigned short buswid, unsigned short offset, unsigned char wdata)

This function is used to use a CPU WR strobe or the WBR option, if available, to write data bytes to the channel memory. This function is called from within the above functions and should not be called directly from the user code.

APPENDIX

Trouble Shooting Tips

If you are using Linux OS and the software cannot recognize PROMJet when connected to a USB port, make sure you have permission to read and write the USB devices under the directory /dev/bus/usb/*. The USB endpoint files under this directory should have the 0x666 file attribute to allow all users to use them. In new Linux/Fedora installations, you can add this to a rules file under /etc/udev/rules.d/. Contact EmuTec or your system admin for more info. Also make sure libusb.so is installed.

After downloading a new program to PROMJet, make sure that the size of PROMJet is configured properly (e.g. by using the Restart check box or the I-option). Also you have to restart the target by either resetting it (the RES and /RES output of PROMJet can be used for this purpose) or recycling the power.

If you are downloading a program in INTEL or Motorola format into an ROM space that does not start at address 0, make sure that either the hex file starts at address 0 or use an offset parameter to neutralize the address offset to 0.

When using the /WR signal to write to PROMJet, the target system must be reset while downloading a new program to hold the target processor. Failure to do so could result in corrupting the data by the target processor. This is not necessary, if the user is only modifying some data bytes while using a PROMJet with an ICE option.

If you are having difficulties using PROMJet with a parallel port that is built-into the PC motherboard, try to use a parallel port expansion card that plugs into the PCI slot. Some of the built-in parallel ports do not provide enough switching voltage under load to communicate with PROMJet. PROMJet requires a high level input between 4 and 5V where these ports guarantee only 2.5V for a high level, which is below the switching level. The new PROMJet version supports parallel ports operating as low as 1.8 volt.

Write-By-Reading Memory Access

The Write-By-Reading memory access is used to modify the contents of a 256 byte memory block without using the /WR signal. The host software (see below) selects the start address of the 256-byte block. The target can write to this 256-byte memory block by reading from a host-selected 4-byte word inside this block (**Access Word**). Please note that this function can be only used when PROMJet has both the ICE and the WBR option.

To use this feature, the user has to select a 256-byte memory block starting at any 256-byte boundary (e.g. from hex address 0xYYYY00 to address 0xYYYYFF) inside PROMJet. The user has also to choose a 4-byte word inside this memory block to be the access word that the target processor will use to modify the memory block. By using the host software, the user has to select **only** the access-word address inside PROMJet (e.g. by using the D option of the DOS software). The start address of the 256-byte memory block will be automatically determined by resetting the least significant 8 bit of the access-word address. Figure 6 shows an example for such a configuration with the access word starting at an offset of 0x84 relative to the beginning of the memory block.

00															0F
10															1F
20															2F
30															3F
40															4F
50															5F
60															6F
70															7F
80				84	85	86	87								8F
90															9F
A0															AF
B0															BF
C0															CF
D0															DF
E0															EF
F0															FF

Figure 5: Example for a WBR memory block.

After configuring PROMJet by the host software, the target can modify the 256 byte memory block by reading the access-word in a specific sequence. While doing so the target will use the least significant address lines A0 to A3 to supply PROMJet with the new data one nibble at a time. Here are the read cycles and address locations that the target must execute to write to the memory block.

1. Read the first byte of the access-word (byte 84) two times to reset the WBR logic.
2. Read the second byte of the access word (byte 85 in the above example).
3. Supply the lower four bits of the data byte by reading the byte in the 16 byte block of the access word that has an offset equal to the lower nibble. The start address of the 16 byte block is set by resetting the least significant 4 bits of the access-word address (e.g. in the example above, the 16 byte block starts at offset 0x80 and ends at 0x8F). This means if the lower nibble is 0xE, the processor has to read offset 0x8E in the above example.
4. Read the third byte of the long word (byte 86 in the above example).

5. Supply the higher four bits of the data byte by reading the byte in the 16 byte block of the access word that has an offset equal to this nibble. This means if the higher nibble is 0x4, the processor has to read offset 0x84 of the buffer in the above example.
6. Read the fourth byte of the access word **three consecutive** times (byte 87 in the example).
7. Read the byte in the 256-byte block that software has to modify. PROMJet will convert this read cycle to a write cycle and will write the data supplied in steps 3 and 5 into this byte.
8. Read this data byte again to see if the Write-By-Reading function is executed correctly and PROMJet has written the new data to the byte. This step is necessary to ensure that there was no hardware interrupts between the steps that might have accessed the 256 byte block and reset the WBR logic. If PROMJet did not modify the data byte, the software has to repeat step 1 to step 8 again.

Please note that **any** read or write access to the 256 byte memory block while executing the above sequence will result in resetting the Write-By-Reading logic and subsequently in failure to write any data to PROMJet. Also it is very important to make sure that there will be no reads to the 256-byte block between the sixth and seventh steps in the above procedure (e.g. from a hardware interrupt handler). If the user cannot guarantee this, he must use a “scratch byte” inside the 256-byte block. The user has to read this byte when starting any interrupt handler to make sure that if the interrupt has occurred after the sixth step and before the seventh, the data will be written in the “scratch byte” and not in the byte that the handler will read first from the 256 byte block. The user can also use the first byte in the 4-byte access word for this purpose.

SPI Interface Signals and functions

The following appendix explains the functionality of the SPI interface. The hardware signals used for the interface are presented first. PROMJet SPI interface supports mode 0 (0,0) and 3 (1,1). It uses 6 signals that are described in the table below:

SPI SIGNAL	FUNCTION	32-PIN DIP CONNECTOR	50-PIN CONNECTOR
/CE (IN)	Chip Enable. Activates the device internal circuits when it goes low	24 (/OE)	6 (/OE)
SCK (IN)	Serial Data Clock signal	22 (/CS)	4 (/CS)
SDI (IN)	Serial Data In	12 (A0)	3 (A0)
SDO (OUT)	Serial Data Out	21 (D15)	23 (D15)
/HOLD (IN)	Pauses serial communication without resetting the serial sequence	31 (A18)	41 (A18)
GND	Ground	16 (GND)	5 & 24 (GND)
VCC (IN)	Sense Voltage	32 (VCC)	15 (VCC)

Figure 6: SPI Signals.

Please **NOTE** that the pin numbering in the 32-PIN DIP connector follows chip pin numbering where pin 1,16,17 and 32 are the corner pins. In SPI mode, these signals will be used to emulate the SPI interface. Selecting “SPI: 1Bit” from the width pull-down menu in the PROMJet windows software activates the SPI mode. PROMJet supports 6 SPI commands as stated in the table below. The functionality of the 3 commands is explained below.

COMMAND	DESCRIPTION	COMMAND CODE	ADDRESS BYTES	DUMMY BYTE	DATA BYTES
Read	Read Data Bytes	0x03	3	0	1 - ∞
Fast Read	Read Data Bytes (dummy Byte)	0x0B	3	1	1 - ∞
Read ID	Read 24 bit CHIP ID	0x9F	0	0	1 - ∞
Read PID	Read PROMJet Size and ID	0xAB	0	0	1 - ∞
Read SR	Read Status Register	0x05	0	0	1 - ∞
Program	Write PROMJet memory	0x02	3	0	1 - ∞

Figure 7: Supported SPI commands.

Data Read Command:

This command starts by asserting the CE signal and sending the READ command code (0x03 HEX) to PROMJet. Following the command, the host sends 3 bytes representing the address information (MSByte first). PROMJet will respond by outputting the data of the first byte on the data out line followed by the next byte and so on until the end of the device address space and then it will wrap around the beginning of the device. So to read 3 bytes starting from memory location 0x123456, the host has to send this sequence: 0x03, 0x12, 0x34, 0x56, 0xFF, 0xFF, 0xFF where 0xFF is a dummy byte. PROMJet will output the data with the first dummy byte.

Data Read Fast Command:

This command starts by asserting the CE signal and sending the READ command code (0x0B HEX) to PROMJet. Following the command, the host sends 3 bytes representing the address information (MSByte first). PROMJet will stay inactive for 8 clock cycles (one byte) and then responds by outputting the data of the first byte on the data out line followed by the next byte and so on until the end of the device address space and then it will wrap around the beginning of the device. So to read 3 bytes starting from memory location 0x123456, the host has to send this sequence: 0x0B, 0x12, 0x34, 0x56, 0xXX, 0xXX, 0xXX where 0xXX is a dummy byte. PROMJet will output the data with the second dummy byte.

Read Status Register Command:

This command starts by asserting the CE signal and sending the RDSR command code (0x05 HEX) to PROMJet. Following the command, the PROMJet will output the Status register value on the output data line.

Data Program Command:

This command starts by asserting the CE signal and sending the WRITE command code (0x02 HEX) followed by 3 bytes representing the address information (MSByte first). Following the address information, the master will send the data byte for the current memory location followed by the data for the next memory location up to the end of the device space. The address then will wrap around to 0. So to write memory address 0x789abc and 0x789abd with 0x01 and 0x02, the host has to send this sequence of bytes to PROMJet: 0x02, 0x78, 0x9A, 0xBC, 0x01, 0x02.

Read ID Command:

This command starts by asserting the CE signal and sending the RDID command code (0x9F HEX) to PROMJet. Following the command, the host sends 3 dummy bytes. With the forth byte, PROMJet will output the programmed CHIP ID. This is the same ID that the user can program in the WBR.SPI ID edit box in the windows software.

Read PROMJet ID Command:

This command starts by asserting the CE signal and sending the RDID command code (0xab HEX) to PROMJet. Following the command, the host sends 3 dummy bytes. With the forth byte, PROMJet will output the memory configuration code and will repeat the dummy byte sequence back to the host (starting from the fifth byte). So if the master device sends the following byte sequence: 0xAB, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA PROMJet will respond with: 0xZZ, 0xZZ, 0xZZ, 0xZZ, **0x81**, 0x44, 0x55, 0x66, **0x81**, 0x88, 0x99, 0xAA where 0xZZ is high impedance and 0x81 states that PROMJet is configured as a 64Kbit device. It is also important to note that the configuration size presents the current PROMJet setup and NOT the PROMJet total size. So if a user has a 128Mbit PROMJet that was configured as 8K x 8, it will still generate the code id 0x81. The table below presents the PROMJet configuration data.

CODE	SIZE	CODE	SIZE	CODE	SIZE
0x81	64KBit	0x8F	2MBit	0xA3	64MBit
0x82	128KBit	0x93	4MBit	0xA7	128MBit
0x83	256KBit	0x97	8MBit	0xAB	256MBit
0x87	512KBit	0x9B	16MBit	0xAF	512MBit
0x8B	1MBit	0x9F	32MBit	0xB3	1GBit

Figure 8: PROMJet ID for memory size.

LPC/FWH Interface Signals and functions

The following appendix explains the functionality of the LPC/FWH (Low Pin Count / Firmware Hub) interface. The hardware signals used for the interface are presented in the table below. PROMJet LPC/FWH interface supports the Firmware hub as well as the LPC read/write cycles and will switch automatically to either mode depending on the target system.

LPC SIGNAL	FUNCTION	32-PIN DIP CONNECTOR	50-PIN CONNECTOR
FWH0/LAD0 (IN/OUT)	Data/Address/Control signal 0	13 (D8)	8 (D8)
FWH1/LAD1 (IN/OUT)	Data/Address/Control signal 1	14 (D9)	10 (D9)
FWH2/LAD2 (IN/OUT)	Data/Address/Control signal 2	15 (D10)	12 (D10)
FWH3/LAD3 (IN/OUT)	Data/Address/Control signal 3	17 (D11)	14 (D11)
FWH4/LFRAME (IN)	Memory access frame start	23 (A10)	31 (A10)
CLK (IN)	Clock signal	22 (/CE)	4 (/CE)
/INIT (IN)	Init signal (same as /RST)	24 (/OE)	6 (/OE)
/RST (IN)	Reset Signal (same as /INIT)	2 (A16)	26 (A16)
GPI0 (IN)	General Purpose Input signal 0	6 (A6)	46 (A6)
GPI1 (IN)	General Purpose Input signal 1	5 (A7)	43 (A7)
GPI2 (IN)	General Purpose Input signal 2	4 (A12)	29 (A12)
GPI3 (IN)	General Purpose Input signal 3	3 (A15)	28 (A15)
GPI4 (IN)	General Purpose Input signal 4	30 (A17)	44 (A17)
GND	Ground	16 (GND)	5 & 24 (GND)
VCC (IN)	Sense Voltage	32 (VCC)	15 (VCC)

Figure 9: LPC/FWH Signals.

Please **NOTE** that the pin numbering in the 32-PIN DIP connector follows chip pin numbering convention where pin 1,16,17 and 32 are the corner pins. Selecting “LPC: 4Bit” from the width pull-down menu in the PROMJet windows software activates the LPC/FWH mode. In this mode, these signals will be used to emulate the LPC interface. Also the ICE option cannot be activated while PROMJet is in the LPC mode (only in parallel mode). PROMJet recognizes 2 types of read and 2 types of write cycles for LPC and FWH interfaces that are described later in this section. PROMJet also supports 4 LPC/FWH functions: Read memory (READ), Write memory (WRITE), Read general-purpose input signals GPI0-4 (RDGPI) and Get manufacturer and device ID (RDID). The functionality of the 4 commands is explained below.

Memory Read Command:

This command uses a simple read cycle to access the memory contents of PROMJet. Either cycle type (LPC/FWH) can be used to do so.

Memory Write Command:

This command uses 2 write cycles to write a single byte. The first cycle is a write cycle to ANY location in the PROMJet address space with a data value of 0x10, 0x40 or 0x0A. The second cycle is to write the desired data to the specified address in the memory space. PROMJet memory contents can be read between the 2 write cycles. Depending on the Firmware version you are using, you can disable the Write function by checking the Cable box or adding I=Cxx to the command line.

Read manufacturer and device ID Command:

This command starts by writing a 0x90 to any memory location in PROMJet address space. The manufacturer ID can be read from memory address 0x00000 or 0xFBC0000 and it should return 0xA5. The device ID is read from memory address 0x00001 or 0xFBC0001 and it provides PROMJet memory configuration code as shown in the table below. It is important to note that the configuration size presents the current PROMJet setup and NOT the PROMJet total size. So a 128Mbit PROMJet that was configured as 1M x 8 will still generate the code id 0x97.

CODE	SIZE	CODE	SIZE	CODE	SIZE
0x81	64KBit	0x8F	2MBit	0xA3	64MBit
0x82	128KBit	0x93	4MBit	0xA7	128MBit
0x83	256KBit	0x97	8MBit	0xAB	256MBit
0x87	512KBit	0x9B	16MBit	0xAF	512MBit
0x8B	1MBit	0x9F	32MBit	0xB3	1GBit

Figure 10: PROMJet ID for memory size.

Read general-purpose Inputs Command:

This command uses a simple read cycle to address 0xFFBC0100. The value of GPIO-4 will be returned in bits 0-4. The GPI lines should be stable during the entire read command to have a valid data.

LPC Read Cycle:

The table below describes the read cycle of PROMJet for LPC memory.

CYCLE	NAME	LAD0-3	DIRECTION	COMMENTS
1	Start	0000	IN	/LFRAME must be active (low) for the part to respond.
2	Read Cycle	010X	IN	Start Read command.
3-10	Address Info	AAAA	IN	32-bit address info starting with the MS nibble first.
11	TAR0	1111	IN-FLOAT	Host will drive the bus to all 1s and then floats it.
12	TAR1	1111	FLOAT-OUT	PROMJet takes control of the bus.
13	SYNC	0000	OUT	PROMJet outputs 0000 indicating that data will be available during the next clock cycle.
14	DATA	DDDD	OUT	Least-significant nibble of the data byte.
15	DATA	DDDD	OUT	Most-significant nibble of the data byte.
16	TAR0	1111	OUT-FLOAT	PROMJet will drive the bus to 1s and then floats it.
17	TAR1	1111	FLOAT-IN	Host takes control of the bus.

Figure 11: LPC Read cycle.

LPC Write Cycle:

The table below describes the read cycle of PROMJet for LPC memory.

CYCLE	NAME	LAD0-3	DIRECTION	COMMENTS
1	Start Cycle	0000	IN	/LFRAME must be active (low) for the part to respond.
2	Write Cycle	011X	IN	Start Write command.
3-10	Address Info	AAAA	IN	32-bit address info starting with the MS nibble first.
11	DATA	DDDD	IN	Least-significant nibble of the data byte.
12	DATA	DDDD	IN	Most-significant nibble of the data byte.
13	TAR0	1111	IN-FLOAT	Host will drive the bus to all 1s and then floats it.
14	TAR1	1111	FLOAT-OUT	PROMJet takes control of the bus.
15	SYNC	0000	OUT	PROMJet outputs 0000 indicating that data was received.
16	TAR0	1111	OUT-FLOAT	PROMJet will drive the bus to 1s and then floats it.
17	TAR1	1111	FLOAT-IN	Host takes control of the bus.

Figure 12: LPC Write cycle.

FWH Read Cycle:

The table below describes PROMJet read cycle for LPC memory.

CYCLE	NAME	LAD0-3	DIRECTION	COMMENTS
1	Start Read	1101	IN	FWH4 must be active (low) for the part to respond.
2	IDSEL	0000	IN	ID select cycle. PROMJet will ignore this field.
3-9	Address Info	AAAA	IN	28-bit address info starting with the MS nibble first.
10	SIZE	0000	IN	This field must be 0 for single byte.
11	TAR0	1111	IN-FLOAT	Host will drive the bus to all 1s and then floats it.
12	TAR1	1111	FLOAT-OUT	PROMJet takes control of the bus.
13-14	WSYNC	0101	OUT	PROMJet outputs wait-sync for data not ready.
15	RSYNC	0000	OUT	PROMJet outputs 0000 indicating that data will be available during the next clock cycle.
16	DATA	DDDD	OUT	Least-significant nibble of the data byte.
17	DATA	DDDD	OUT	Most-significant nibble of the data byte.
18	TAR0	1111	OUT-FLOAT	PROMJet will drive the bus to 1s and then floats it.
19	TAR1	1111	FLOAT-IN	Host takes control of the bus.

Figure 13: FWH Read cycle.

FWH Write Cycle:

The table below describes PROMJet write cycle for FWH memory.

CYCLE	NAME	LAD0-3	DIRECTION	COMMENTS
0	Start Cycle	1110	IN	FWH4 must be active (low) for the part to respond.
2	IDSEL	0000	IN	ID select cycle. PROMJet will ignore this field.
3-9	Address Info	AAAA	IN	28-bit address info starting with the MS nibble first.
10	SIZE	0000	IN	This field must be 0 for single byte.
11	DATA	DDDD	IN	Least-significant nibble of the data byte.
12	DATA	DDDD	IN	Most-significant nibble of the data byte.
13	TAR0	1111	IN-FLOAT	Host will drive the bus to all 1s and then floats it.
14	TAR1	1111	FLOAT-OUT	PROMJet takes control of the bus.
15	RSYNC	0000	OUT	PROMJet outputs 0000 indicating that data was received.
16	TAR0	1111	OUT-FLOAT	PROMJet will drive the bus to 1s and then floats it.
17	TAR1	1111	FLOAT-IN	Host takes control of the bus.

Figure 14: FWH Write cycle.

Interface Signals and Pinout

The following is a listing of the signals on the 32-pin DIP connector of PROMJet (top view). Please note that the numbers between the parentheses in figure below are for an EPROM using a 28-pin adapter.

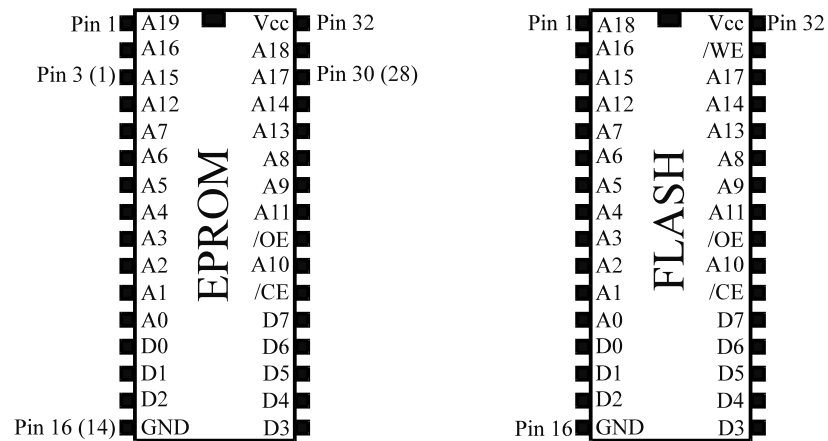


Figure 15: EPROM/FLASH pinout.

A0 to A19 present the address-input lines for PROMJet. D0 to D7 are the bi-directional data lines. CE (Chip Enable) and OE (Output Enable) are the chip select signals for PROMJet. WE signal in the FLASH configuration is the write enable input to write to the device. Please note also that the ground pin (GND) is **always** connected to the ground pin of the target socket in any configuration (32 or 28 pin). All other pins will be adjusted accordingly to the ground pin. In the case of 28 pin EPROMs, pin 32 must be connected to pin 30 to supply PROMJet with power if no external power supply is used.

The 32-pin adapter is configurable to support either an EPROM or FLASH depending on the software configuration. When PROMJet is configured to support FLASH memory, pin 31 is used as a WE input signal and pin 1 is used as address 18. Otherwise pin 31 and pin 1 are configured as address 18 and 19.

Binary and Hex File Formats

The following is a brief description of the file formats supported by the JET program.

Binary Format

Binary format is the actual data without any other formatting information such as address or checksum error.

Intel Hex Format

The Intel Hex format is a line-by-line specification of the data. The format of each line is:

: LLAAAATTDD...DDCC

The line always begins with a colon.

LL: Two digit hex field specifying the number of data bytes.

AAAA: Four digit hex field that specifies the first memory address.

TT: Two digit hex field specifying the type of record. Valid record types are:

- 00**: Normal data record
- 01**: End of file record
- 02**: Extended Segment address record
- 04**: Extended Linear Address Record

DD..DD: Data field in hex digits, two per byte.

CC: Two-digit hex checksum character.

Motorola S Record Format

The Motorola S format consists of ASCII data records with the following format:

STLLAA...AADD...DDCC

The line always begins with an S.

T: Type of record. Defined types are:

- 0**: header record
- 1**: normal data record (16-bit address)
- 2**: normal data record (24-bit address)
- 3**: normal data record (32-bit address)
- 7**: termination record for a block of S3 records (32-bit address)
- 9**: end of file record

LL: Two digit hex field specifying the number of bytes in the record.

AA...AA: 4, 6 or 8 digit hex field (depending on T) specifying the first memory address.

DD...DD: Data field in hex digits, two per byte.

CC: Two-digit hex checksum character.

Identifying your PROMJet

You will find a label on the bottom of PROMJet that identifies the capacity, speed, model and features of PROMJet. We will explain here how to use this label to identify your PROMJet. This label also specifies the serial number of PROMJet, which is very important for warranty repairs. Tampering, removing, replacing or modifying this label will void the warranty of PROMJet. The label has the following information:

PJ {Size} M {Speed} NS- {Model}{V}{W/P/L/M/N}
 {Serial Number of PROMJet}

The differences between PROMJets are in the Size, Memory Speed and Model. The size parameter presents PROMJet size in Mbit and can be 1, 4, 8, 16, 32, 64, 128 or 256 Mbit. The speed parameter can vary from 25, 45 to 85ns which is indicated by (2, 4, 8). PROMJet has 3 models to choose from: Standard, In-Circuit-Emulator (ICE) and In-Circuit-Emulator with the Write-By-Reading (WBR) option. These features are identified via the characters: **S**, **I** and **R**. The table below explains the feature of each type.

Features supported	Standard (S)	ICE (I)	WBR (R)
On-The-Fly memory access	N	Y	Y
Modifying PROMJet by reading it	N	N	Y

Figure 16: PROMJet features.

A V in the model code means that this unit can emulate any memory device with a supply voltage between 1.8V and 5V. While a W at the end of the model code indicates that this PROMJet can emulate 16-bit as well as 8-bit memory devices, a P indicates the support for the SPI memory interface in addition to the 8-bit support. An M is used to denote a PROMJet that can support 16-bit, 8-bit and SPI memory interface. An L or an N indicates LPC/FWH memory support where the N version will also support 16-bit memory accesses as explained in the table below.

Features supported	Blank	W	P	L	M	N
Data bus widths	8 Bit	8 & 16 Bit	8 Bit & SPI	8 Bit & LPC	8 & 16 bit and SPI	8 & 16 bit and LPC

Figure 17: PROMJet data modes.

To use the unit in 16-bit mode, the 50-pin high-density connector on the bottom of the unit must be used. The 32-pin DIP original connector only gives access to the high 8 bit that will be used in 8-bit memory emulation. Either connector will support the LPC or SPI functions. Contact EmuTec for a pinout of the high-density connector.

Electrical and Physical Specifications

Memory size:	4/8/16/32/64/128/256Mbit (configurable x8 or x16).
Access Time:	85nS standard (45 and 25NS optional).
Download speed:	8 Mb/S depending on host speed.
Supply Voltage:	5.0 V \pm 10% (external supply).
Operating Current:	300mA depending on capacity, speed and options.
I/O Specifications:	TTL levels input and output. 1.8-3V with variable voltage option
Physical dimensions:	2.3" x 1.7" x 0.7".