



PROGPPCNEXUS User Manual





Purchase Agreement

This software and accompanying documentation are protected by United States Copyright law and also by International Treaty provisions. Any use of this software in violation of copyright law or the terms of this agreement will be prosecuted.

All the software in this package is copyrighted by P&E Microcomputer Systems, Inc. Copyright notices have been included in the software.

P&E Microcomputer Systems authorizes you to make archival copies of this software for the sole purpose of back-up and protecting your investment from loss. Under no circumstances may you copy this software or documentation for the purpose of distribution to others. Under no conditions may you remove the copyright notices from this software or documentation.

This software may be used by one person on up to two different computers, provided that the software is never used on the two computers at the same time. P&E expects that group programming projects making use of this software will purchase a copy of the software and documentation for each user in the group. Contact P&E for volume discounts and site licensing agreements.

With respect to the physical media provided within, P&E Microcomputer Systems warrants the same to be free of defects in materials and workmanship for a period of 30 days from the date of receipt. If you notify us within the warranty period, P&E Microcomputer Systems will update the defective media at no cost.

P&E Microcomputer Systems does not assume any liability for the use of this product beyond the original purchase price. In no event will P&E Microcomputer Systems be liable for additional damages, including any lost profits, lost savings or other incidental or consequential damages arising out of the use or inability to use these programs, even if P&E Microcomputer Systems has been advised of the possibility of such damage.

By installing or using this software, you agree to the terms of this agreement. If you do not agree with these terms, you should not install this software.

©2016, 2018 P&E Microcomputer Systems, Inc.

Windows is a registered trademarks of Microsoft Corporation.

NXP is a registered trademark of NXP Semiconductor, Inc. ColdFire, Kinetis, and Qorivva are registered trademarks of NXP Semiconductor, Inc.

All other product or service names are the property of their respective owners.

P&E Microcomputer Systems, Inc.
98 Galen St.
Watertown, MA 02472
617-923-0053
<http://www.pemicro.com>

Manual version: 1.01
October 2019



1	OVERVIEW.....	1
1.1	Programming Algorithms (.PCP Files)	2
1.2	Start-Up Configuration.....	2
1.3	Manual Programming	2
1.4	Scripted Programming.....	3
1.5	Hardware Interfaces	3
2	PROGRAMMING ALGORITHMS	4
2.1	Algorithm File Contents	4
2.2	Modifying A Programming Algorithm.....	6
2.3	Creating A Programming Algorithm.....	6
3	PROGRAMMING COMMANDS.....	7
3.1	BM - Blank Check Module.....	8
3.2	CM - Choose Module .PCP	8
3.3	EM - Erase Module.....	8
3.4	EN - Erase If Not Blank	8
3.5	HE - Help.....	8
3.6	PM - Program Module	9
3.7	PS - Program Serial Number.....	9
3.8	PR - Program Module Range	9
3.9	QU - Quit	9
3.10	RE - Reset chip	9
3.11	SM - Show Module	9
3.12	SS - Specify S-Record	9
3.13	UM - Upload Module	10
3.14	UR - Upload Range	10
3.15	VM - Verify Module.....	10
3.16	VR - Verify Range	10
3.17	VC - Verify CRC Of Object File To Module	10
3.18	SC - Show Module CRC.....	11
3.19	VV - Verify Module CRC to Value	11
3.20	SA - Show Algorithm Source.....	11



4	START-UP CONFIGURATION	12
5	CONNECTION MANAGER.....	16
5.1	Advanced Settings	17
5.2	Additional Settings.....	19
5.3	Connect and Choose Algorithm	21
6	MANUAL PROGRAMMING	23
6.1	Manual Programming Procedure	23
7	SCRIPTED PROGRAMMING (CPROGPPCNEXUS)	25
8	HARDWARE INTERFACES	27
8.1	Multilink and Multilink FX.....	27
9	PROGRAMMING UTILITIES	31
9.1	Serialize.....	31
9.2	Unsecure_12.....	31
	APPENDIX A -SETUP COMMANDS.....	32
	APPENDIX B -TABLE ENTRY	34

1 OVERVIEW

PROGPPCNEXUS is PEmicro's programming software for Flash/EEPROM modules that are attached to an NXP MPC55xx-57xx or STMicroelectronics SPC5 processor. PROGPPCNEXUS talks to the processor's background debug module using one of PEmicro's compatible hardware interfaces. These interfaces connect a PC running Windows XP/Vista/7/8/10 to a debug connector on the target system. This connector provides access to the debug signals of the processor chip mounted on your target system hardware board.

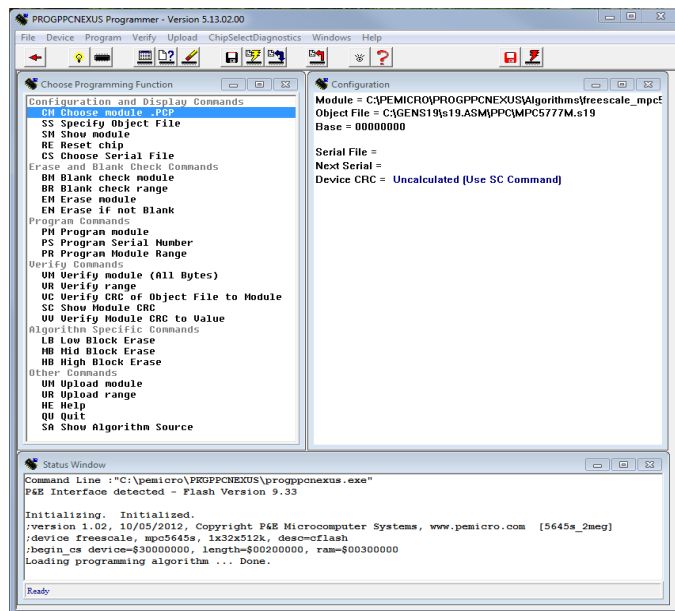


Figure 1-1: PROGPPCNEXUS User Interface

As part of the programming procedure, the user will need to select a programming algorithm that will enable the PROGPPCNEXUS software to properly manage their specific target device during programming. The user may also choose to set certain programming parameters before beginning to program. This chapter presents a brief overview of the programming procedure.

1.1 Programming Algorithms (.PCP Files)

PROGPPCNEXUS runs on the PC and provides a set of general interface functions and processor-specific user functions that are used to control the erasing, verifying, programming and viewing of modules to be programmed. These general functions are implemented for a particular target configuration and chip set by using specific Programming Algorithm (.PCP) files that the user can modify to reflect the setup of their particular target interface. PROGPPCNEXUS includes a library of these programming algorithms. For the most recent version of this library of algorithms, please visit our website, www.pemicro.com.

Programming algorithm files can also be modified by the user according to specific conventions. In addition, PE micro can create programming algorithms upon request if you are working with a device whose corresponding algorithm is not included in the current library. Some additional information about the contents and modification of programming algorithms is included in **CHAPTER 2 – PROGRAMMING ALGORITHMS**.

1.2 Start-Up Configuration

Certain programming parameters can be adjusted when launching the PROGPPCNEXUS software by using the executable command-line to input the appropriate parameters. These may include settings related to the type of hardware interface you are using, S-record verification, and more, depending on your target device. A list of specific parameters with examples of their usage is included in **CHAPTER 4 – START-UP CONFIGURATION**.

1.3 Manual Programming

PROGPPCNEXUS lists commands that are available to execute. Any of the programmer's enabled features can be selected by using the mouse, the up and down arrow keys, or by typing the selection letters to the left of the selection display. Pressing ENTER or double clicking the mouse will execute the highlighted entry if it is enabled. The user will be prompted for any additional information that is required to execute the selected function. Before you can program a module from an S record file, you must select such a file. If you try to do a program module function and you have not selected an S-record file, you will be asked to select one. A list of programming commands and their functions may be found in **CHAPTER 6 – MANUAL PROGRAMMING**.

1.4 Scripted Programming

Programming commands, in addition to being executed manually, may also be collected into script files which can be used to automate the programming process. These scripts are executed by a command-line programming application called CPROGPPCNEXUS, which is included with the PROGPPCNEXUS software. More information about scripted programming is located in the accompanying CPROGPPCNEXUS user guide.

1.5 Hardware Interfaces

In addition to PROGPPCNEXUS programming procedures, this manual discusses hardware interfaces that may be used in conjunction with the PROGPPCNEXUS. For supported NXP processors, PEmicro typically offers both value-oriented development solutions and more robust and versatile production solutions. You can learn about these interfaces in **CHAPTER 8 – HARDWARE INTERFACES**.

2 PROGRAMMING ALGORITHMS

PEmicro's .PCP programming algorithm files define the functions necessary for PROGPPCNEXUS to program an NXP MPC55xx-57xx or STMicroelectronics SPC5 processor's internal flash or connected external Flash/EEPROM. After you choose the appropriate algorithm, it will appear in the Configuration Window.

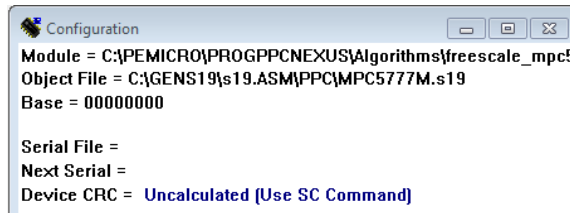


Figure 2-1: Configuration Window

2.1 Algorithm File Contents

You may view and, if necessary, modify the contents of an algorithm by opening it in any text editor. A .PCP programming algorithm file consists of four parts:

1. Comments
2. User-specified functions
3. Setup commands
4. S-records

2.1.1 Comments

Comments are usually placed in the file to identify the target system for which the .PCP file was written and what module on the target system it programs, as well as other useful information. If a specific .PCP file is selected in PROGPPCNEXUS, these comments are shown in the window at the bottom of the PC screen. Within the algorithm file a semicolon is used to designate the beginning of a comment.

2.1.2 User Specified Functions

There can be up to six user-specified functions included in a .PCP file. Each user statement in the .PCP file must have a corresponding address in same order as the table part of the S-records and an appropriate set of code. A line which defines a user specified programming function has a total of 57 characters in the form:

PROGPPCNEXUS to find them. The files are in ASCII and are thus readable using most text editors. The S records for a .PCP file can be generated using most assemblers.

2.2 Creating A Programming Algorithm

In certain situations, a user may wish to either create their own programming algorithm or make significant modifications to an existing file.

A .PCP file is a structured file which contains a table of essential system constants and routine addresses. This table is followed by the definitions of the routines. Register and memory usage conventions must be followed when you insert your own set of routines. Any routine which can not or need not be provided is given a zero (0) address in the table. The table, routines, stack and buffer reside in the CPU on chip RAM during the execution of PROGPPCNEXUS. Routines return to PROGPPCNEXUS by executing a breakpoint (BKPT) instruction.

The table contains several long word (32-bit) entries listed in an exactly specified order. In addition, the table is assembled at the starting address at which the on chip RAM will be configured during execution of PROGPPCNEXUS. Furthermore, the table must be the first thing assembled to insure that it is the first S record in the .PCP file.

Examples of the assembler files (.ASM files) used to produce the .PCP files for external flash are available upon request. The first part of each file is the table that generates S records. The origin of the table tells the PC program where the on chip RAM should be configured during the programming process. The choice is made in a manner that does not conflict with other things in the target system, such as the module to be programmed.

The programming routines for a particular module are loaded into the NXP MPC55xx-57xx or STMicroelectronics SPC5 processor's on-chip RAM for execution during erasure, programming, verification and showing of the module. The routines and associated comments are in the form of Motorola S-records stored in the .PCP programming algorithm.

3 PROGRAMMING COMMANDS

When the user performs manual programming, commands are executed by selecting them from the Choose Programming Function Window pick list. The user may either use the up/down arrow keys or type the two-letter abbreviation for the command (listed below) on the command line to select a command. Pressing ENTER causes the selected command to execute. Commands can also be executed from the Menus or from the Button Bar. If there is any additional information needed in order to execute the command, the user will be prompted for this information in a new window. Errors caused by a command or any other responses will be presented in the Status Window.

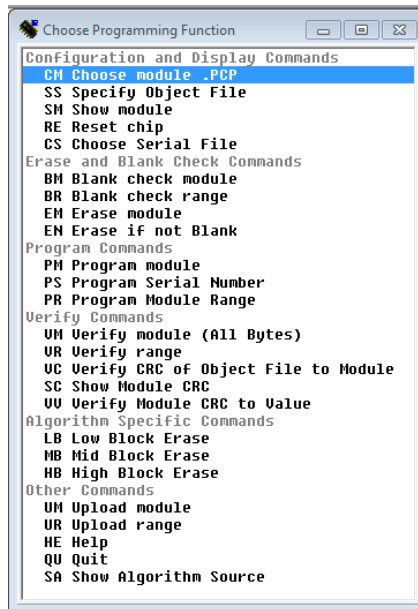


Figure 3-1: Choose Programming Function Window

At any given time, or for a particular module, some of the commands may not be active. Inactive commands are indicated as such in the Choose Programming Functions Window and will not execute.

Below is a description of each of the PROGPPCNEXUS commands used in manual programming. These same commands are also used in scripted programming. For

more information about scripted programming, see the **CPROGPPCNEXUS User Guide**.

3.1 BM - Blank Check Module

This command checks the entire module to see if it has been erased. If not, the address of the first non-blank location is given along with its contents.

3.2 CM - Choose Module .PCP

The user is presented with a list of available .PCP files. Each .PCP file contains information on how to program a particular module. Usually, the name of the file indicates what kind of module it relates to. For example, the file Freescale_MPC5744P_1x32x616k_CFlash.PCP specifies how to program the 32x616k CFlash block on a MPC5744P processor. Setup information and further descriptions of the module are provided in ASCII text within the module file. This information is presented in the status window when a .PCP file is selected. The user can also look at this information inside of the module itself by using any standard text editor to view the module contents.

A particular .PCP file is selected by using the arrow keys to highlight the file name and then pressing ENTER. The currently selected .PCP file is shown in the .PCP file selected window. After a .PCP file is selected, the user is prompted for the base address of the module. This address is used as the beginning address for the module during programming and verification. Certain .PCP files, such as those for external flash algorithms, will prompt the user for the base address of the module.

3.3 EM - Erase Module

This command erases the entire module. If the entire Module is not erased, an error message will be returned.

3.4 EN - Erase If Not Blank

A blank check is performed to determine whether the flash is already erased. If not, an erase command is executed.

3.5 HE - Help

Opens this PROGPPCNEXUS user manual.

3.6 PM - Program Module

For this command to work, the user must have previously selected an S-record file. The S-records are then checked to see if they all reside in the module to be programmed. If not, the user is asked if they want to continue. If the answer is yes, only those S-record addresses which lie in the module are programmed. If a location cannot be programmed, an error message will be returned.

3.7 PS - Program Serial Number

Program the .SER file selected using the CS command.

3.8 PR - Program Module Range

Program the object file within specified starting and ending addresses

3.9 QU - Quit

Terminates PROGPPCNEXUS and returns to Windows.

3.10 RE - Reset chip

This causes a hardware reset to the microcontroller. This command can be used to recover from errors which cause the programmer not to be able to talk to the processor through the background debug mode.

3.11 SM - Show Module

The user is prompted for a starting address. If this address is not in the module and error is given. A window is opened which shows the contents of memory as hex bytes and ASCII characters if printable. Non-printing characters are shown as periods ("."). This window stays on the screen until the user presses ESCAPE.

3.12 SS - Specify S-Record

If the file is not found, an error message is given. The currently selected file is shown in the S19 file selected window. The programmer accepts S1, S2, and S3 records. All other file records are treated as comments. If the user does not specify a file name extension, a default of .S19 is used.

3.13 UM - Upload Module

The user is asked for a filename into which to upload S-records. The default filename extension is set to .S19 if none is specified by the user. S-records for the entire module are then written to the specified file.

3.14 UR - Upload Range

The user is prompted for a starting address, which must be in the module. Next, the user is asked for an ending address, which must also be in the module. The user is then asked for a filename into which to upload S-records. The default filename extension is set to .S19 if none is specified by the user. S-records are then written to the specified file.

3.15 VM - Verify Module

For this command to work, the user must have previously selected an S-record file. The S-records are then checked to see if they all reside in the module to be programmed. If not, the user is asked if they want to continue. If the answer is yes, only those S-record addresses which lie in the module are verified. If a location cannot be verified, an error message will be returned which indicates the address, the contents of that address, and the contents specified in the S-record file.

3.16 VR - Verify Range

For this command to work, the user must have previously selected an S-record file. The user is prompted for a starting address, which must be in the module. Next, the user is asked for an ending address, which must also be in the module. S-record addresses which lie in the module are verified. If a location cannot be verified, an error message will be returned which indicates the address, the contents of that address, and the contents specified in the S-record file.

In addition, there is one function that is allowed to be unique to the module being programmed. The selection menu name and the length of up to one hexadecimal parameter may be specified in a supporting .PCP file.

3.17 VC - Verify CRC Of Object File To Module

Verify the flash against the object file using CRC calculations.



3.18 SC - Show Module CRC

Calculate and display the Checksum of the whole flash. Calculation also includes the blank addresses. Trim values are ignored.

3.19 VV - Verify Module CRC to Value

Verify against a specified CRC value. Used with the SC command to ensure each chip is programmed with the same data.

3.20 SA - Show Algorithm Source

Show the algorithm's source

4 START-UP CONFIGURATION

The PROGPPCNEXUS software may be started in a way that enables certain optional parameters, which can assist the programming process. To set these command-line parameters, highlight the Windows Icon for the PROGPPCNEXUS executable, right-click, and select "Properties" from the pop-up File Menu. The "General" Properties tab should open by default. There are several parameters that you may then include on the command line. A description of each is listed below, followed by specific examples of how these parameters are used.

Syntax:

```
PROGPPCNEXUS [io_delay_cnt n] [v] [interface=x]
              [port=y]
```

Where:

Optional parameters are in brackets []. The parameters are described as follows:

- [v]** If the optional parameter v is specified as either V or v, then the range of S-records is not verified during the programming or verification process. This can help speed up these functions.
- [interface=x]** where x is one of the following: (See examples section)
- USBMULTILINK (supports Multilink Universal, Multilink Universal FX, and OSJtag)
- [port=y]** Where the value of y is one of the following (see the showports command-line parameter for a list of connected hardware; always specify the "interface" type as well):
- USBx Where x = 1,2,3, or 4. Represents an enumeration number for each piece of hardware starting at 1. Useful if trying to connect to a Cyclone or Multilink product. If only one piece of hardware is connected, it will always enumerate as USB1.

An example to select the first Multilink found is:

```
INTERFACE=USBMULTILINK  
PORT=USB1
```

Ethernet IP address ###. Each # symbol represents a decimal number between 0 and 255. Valid for Cyclone and Tracelink interfaces.

Connection is via Ethernet.

```
INTERFACE=CYCLONE PORT=10.0.1.223
```

NAME Some products, such as the Cyclone and Tracelink, support assigning a name to the unit, such as "Joe's Max". The Cyclone may be referred to by it's assigned name. If there are any spaces in the name, the whole parameter should be enclosed in double quotes (this is a Windows requirement, not a PEmicro requirement).

Examples:

```
INTERFACE=CYCLONE  
PORT=MyCyclone99
```

```
INTERFACE=CYCLONE "PORT=Joe's Max"
```

UNIQUEID USB Multilink products all have a unique serial number assigned to them, such as PE5650030. The Multilink may be referred to this number. This is useful in the case where multiple units are connected to the same PC.

Examples:

```
INTERFACE=USBMULTILINK  
PORT=PE5650030
```

COMx Where x = 1,2,3, or 4. Represents a COM port number. Valid for Cyclone interfaces.

To connect to a Cyclone on COM1 :
INTERFACE=CYCLONE PORT=COM1

x Where x = 1,2,3, or 4. Represents a parallel port number

To select a parallel interface on Parallel Port #1 :
INTERFACE=PARALLEL PORT=1

PCIx Where x = 1,2,3, or 4. Represents a BDM Lightning card number. (Note: this is a legacy product)

To select a parallel cable on BDM Lightning #1 :
INTERFACE=PARALLEL PORT=PCI1

Example 1

```
C:\PROGPPCNEXUS C:\ENGINE.CFG Interface=USBMULTILINK Port=USB1
```

Opens C:\PROGPPCNEXUS with the following options:

- Run the C:\ENGINE.CFG script
- Interface is USB Multilink Universal, first cable detected.

Example 2

CPROGPPCNEXUS C:\ENGINE.CFG Interface=CYCLONEMAX
Port=209.61.110.251

Opens CPROGPPCNEXUS with the following options:

- Run the C:\ENGINE.CFG script
- Interface is Cyclone MAX via the Ethernet Port with an IP address of 209.61.110.251

5 CONNECTION MANAGER

Before programming your device, you will need to connect to your target using a hardware interface. Interface options for PROGPPCNEXUS are discussed in **Section 8 - HARDWARE INTERFACES**.

Once you have physically connected your PC to your target using the hardware interface, and the appropriate drivers are installed, the following Connection Manager dialog will appear:

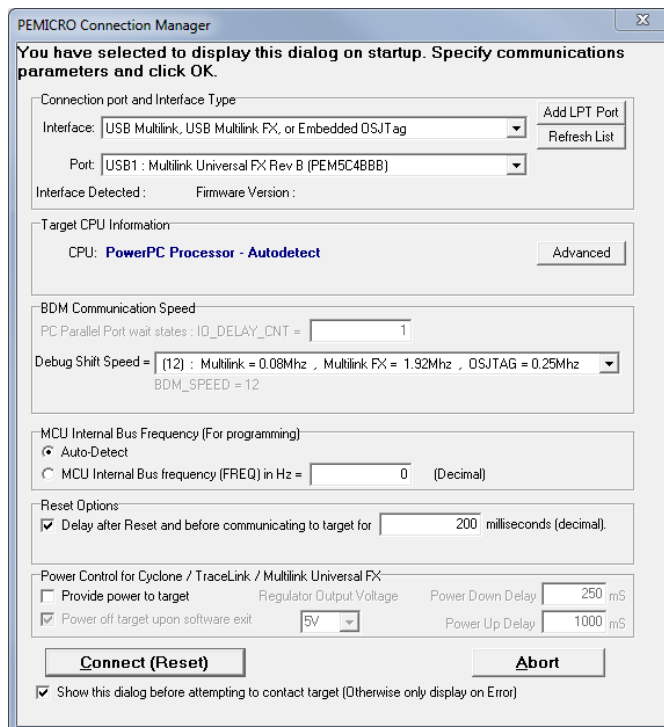


Figure 5-1: Connection Manager Dialog

The Connection Manager allows you to choose the interface that you wish to use and configure the connection.

Use the Interface drop-down menu to choose the type of interface that you plan to use.

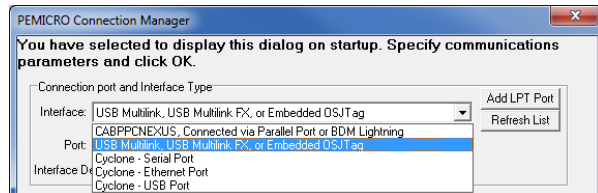


Figure 5-2: Connection Manager - Select Interface

Then select the interface from those available, which are listed in the Port drop-down list. The Refresh List button to the right may be used to update the list of available interfaces.

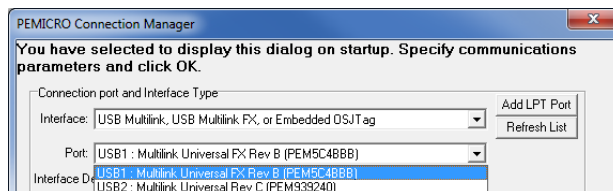


Figure 5-3: Connection Manager - Select Port

5.1 Advanced Settings

In the Target CPU Information section, the “Advanced” button allows access to some additional settings.

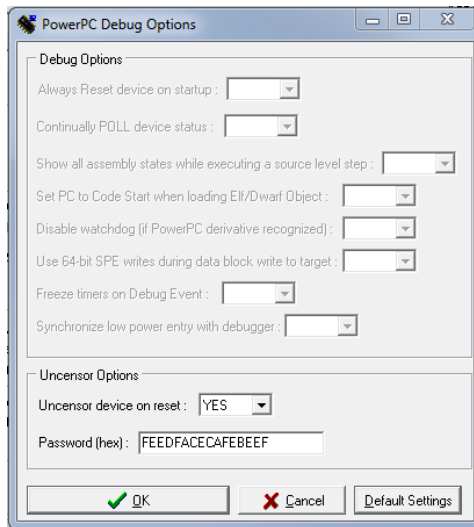


Figure 5-4: Advanced Dialog

Always reset device on startup

On startup, the device will reset which allows for the reset scripts to initialize the device.

Continually POLL device status

Continuous check if the device has reset, currently running, or still in halt.

Show all assembly states while executing a source level step

When stepping through High-level C source code, the code window will display the disassembly source between each line of C source code.

Set PC to Code Start when loading Elf/Dwarf Object

After loading the Elf/Dwarf file, the PC will be set to the entry point specified in the elf header.

Disable watchdog (if PowerPC Derivative recognized)

Legacy setting, no affect

Use 64-bit PSE writes during data block write to target

Legacy setting, no affect

Freeze timers on Debug Event

Timers will halt when the debug session is halted. Otherwise, timers are free-running.

Synchronize low power entry with debugger

Allows the debug session to start and stay active instead of losing communication and exiting.

Uncensor device on reset

If the chip is censored, this setting allows the debugger to enter a password to uncensor the device and allows debugging and flash programming.

Note: If the password was left blank (by erasing the shadow block or not specifying a password) then the chip may be permanently secured. The chip may no longer be accessed using external debuggers. Please refer to the chip's reference manual for more details.

Password (Hex)

The password can be either 64-bit (16 hex characters) or 256-bit (64 hex characters) depending on the device. The user should read their chip's reference manual for more details.

5.2 Additional Settings

The remainder of the PEMICRO Connection Manager allows the user to make settings related to BDM Communications Speed, MCU Internal Bus Frequency, and Power Control (for interfaces that can provide power to the target device)..

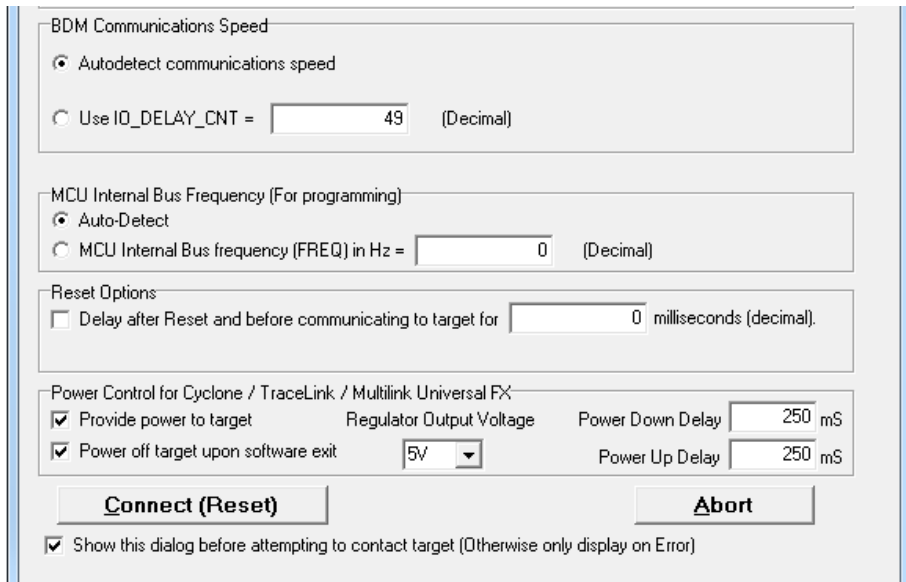


Figure 5-5: Connection Manager - BDM Shift Freq

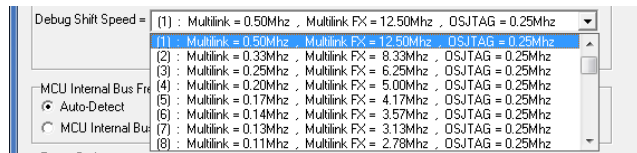


Figure 5-6: Debug Shift Speed

5.2.1 BDM Communications Speed

This software can automatically detect the proper communication speed to establish a connection with the target. For HC12/HCS12 devices, the BDM communication speed may be set (instead of being auto-detected) by setting the value of the IO_DELAY_CNT variable. If the software takes a long time to detect the proper value for IO_DELAY_CNT, it can be set manually.

5.2.2 MCU Internal Bus Frequency (For programming)

This option can be set to auto-detect in most situations.

5.2.3 Reset Options

If your board has any active components connected to your RESET signal such as a supply voltage supervisory circuit or a reset monitor, the RESET signal may have a longer rise time. This option can set a delay before beginning communication to give time for RESET to stabilize. A typical value is 300 milliseconds.

5.2.4 Power Control for Cyclone / TraceLink / Multilink Universal FX

This option controls how power is provided to the target board (only on supported debug interfaces).

5.3 Connect and Choose Algorithm

Once you have made all your selections in the PEMICRO Connection Manager, Click the Connect (Reset) button to connect to the target. If you are successful, you will be prompted to choose a programming algorithm for your target using the following browse window:

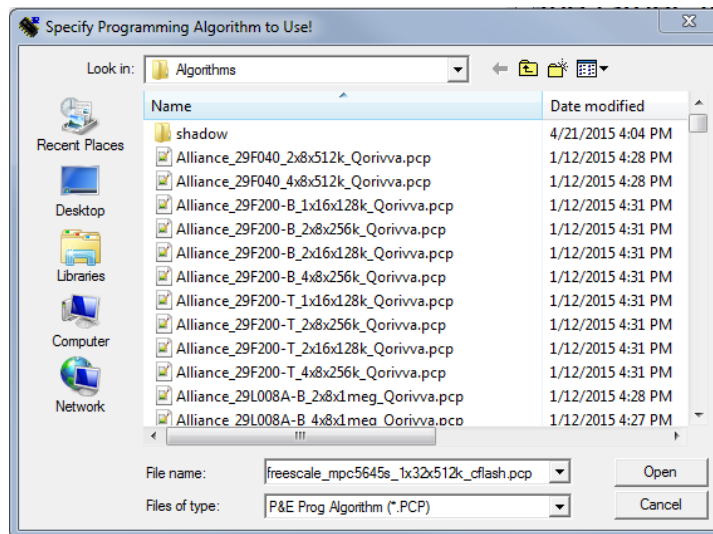


Figure 5-7: Select Algorithm

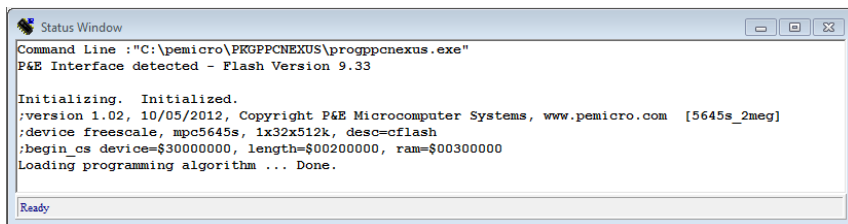
Once you have selected the appropriate algorithm, you are ready to begin



programming.

6 MANUAL PROGRAMMING

The Choose Programming Function Window (see **Figure 3-1**) lists commands that are available to execute. Any of the programmer's enabled features can be selected using the mouse, the up and down arrow keys, or by typing the two-letter command abbreviations that appear to the left of the list of programming functions into the Status Window. The Status Window also displays any error messages that might result from the commands that you perform.



```

Status Window
Command Line : "C:\pemicro\PROGPPCNEXUS\progppcnexus.exe"
P&E Interface detected - Flash Version 9.33

Initializing.  Initialized.
;version 1.02, 10/05/2012, Copyright P&E Microcomputer Systems, www.pemicro.com [5645s_2meg]
;device freescale, mpc5645s, 1x32x512k, desc=cflash
;begin_cs device=$30000000, length=$00200000, ram=$00300000
Loading programming algorithm ... Done.

Ready
  
```

Figure 6-1: Status Window

Pressing ENTER or double clicking the mouse in the Choose Programming Function Window will execute the highlighted entry if it is enabled. The user will be prompted for any additional information that is required to execute the selected function. Before you can program a module from an S record file, you must select such a file. If you try to execute a program module function and you have not selected a file, you will be asked to select one.

6.1 Manual Programming Procedure

Here is the procedure for performing manual programming:

1. Before turning on your power supply, check that the target power supply is on and the interface cable is connected to your target board. Be sure to apply proper target voltage before programming the flash. If you lose contact with your target board at any time during the procedure, you may double-click the "RE" command (Reset) to begin again.
2. Using the PROGPPCNEXUS software, choose the programming algorithm by selecting the appropriate .PCP file. Double clicking the "CM" (Choose Module) command will allow you to select the algorithm you wish to use.
3. After you select the .PCP file, you may be asked for the base address.

This is the address at which you would like to program the code. Enter the appropriate base address.

4. a) Use the "EM" (Erase Module) command to erase the module at that location. The process of erasing the module will vary according to the size of the flash, but should take no longer than 30 seconds. If this procedure seems to be taking much longer than 30 seconds, then the computer is probably not getting a proper response from the board. If this is the case:
 - b) Check the jumper setting on your target board, as well as the programming voltage.
5. Some programming algorithms have a special command, such as "BE," for block erase. If you are unable to double-click the "EM" (Erase Module) command, try using the "BE" (Block Erase) command. Some commands are hidden and you may need to use the scroll bar to scroll down to these commands.
6. You may check to see whether or not the module has been erased by double-clicking the "BM" command (Blank Check Module). If the flash is not properly erased then this command will give you an error message. You may also check the contents of the memory locations by double-clicking the "SM" (Show Module) command. If the flash has been erased properly then all the memory locations will display "FF".
7. Now use the "SS" command (Specify S Record) to load the object file (.S19), which you should have generated previously by using a compiler or an assembler. This command will ask for the name of the .S19 file.
8. Now you ready to program the flash. Double click the "PM" command (Program Module) to begin the programming process.
9. In order to check the results, use the "SM" command (Show Module) with the appropriate base address to view the contents of the flash. You should see that the flash has been correctly programmed. You may also double-click the "VM" command (Verify Module) to verify that all the bytes of the flash are correctly programmed.

7 SCRIPTED PROGRAMMING (CPROGPPCNEXUS)

Programming commands, in addition to be executed manually, may also be collected into script files which can be used to automate the programming process. These scripts are executed by a command-line programming application called CPROGPPCNEXUS, which is included with the PROGPPCNEXUS software. When you run the CPROGPPCNEXUS.EXE application, it will look for the *prog.cfg* script file and automatically execute the commands in that file.

For complete instructions on how to configure and execute the CPROGPPCNEXUS scripted programmer, please see the **CPROGPPCNEXUS User Guide**.



8 HARDWARE INTERFACES

PEmicro's Multilink debug probes and Cyclone in-system programmers are compatible hardware interfaces for use with PROGPPCZ. The Multilink and Multilink FX are development tools that communicate via USB and will enable you to debug your code and program it onto your target. The Cyclone LC and Cyclone FX are more versatile and robust development tools that communicate via Ethernet, USB, or Serial Port, and include advanced features and production programming capabilities, as well as Ethernet support.

Below is a review of their features and intended usage.

8.1 Multilink and Multilink FX

PEmicro's Multilink and Multilink FX debug probes offer an affordable and compact solution for your development needs, and allows debugging and programming to be accomplished simply and efficiently. Those doing rapid development will find the Multilink and Multilink FX easy to use and fully capable of fast-paced debugging and programming.



Figure 8-2: Multilink debug probe

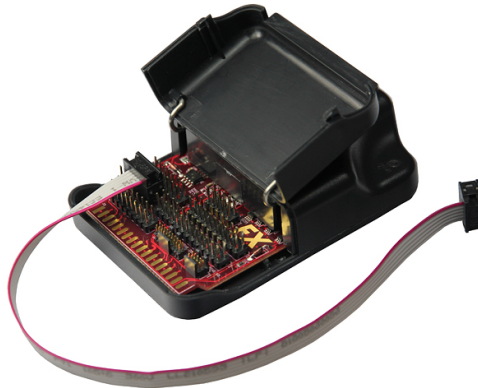


Figure 8-3: Multilink FX debug probe (open for access to headers)

8.1.1 Key Features

- Programming and debugging capabilities
- Compact and lightweight
- Communication via high-speed USB 2.0
- Supported by PEmicro software, NXP's MCUXpresso IDE, Kinetis® Design Studio, S32 Design Studio for ARM, S32 Design Studio for Vision, S32 Design Studio for Power, and other third-party software

8.1.2 Product Features & Implementation

PEmicro's Multilink debug probes allow a Windows XP/Vista/7/8/10 PC access to the debug mode on the target device via JTAG/SWD protocols (where applicable). Multilink ACP supports ARM Cortex-M devices from several manufacturers.. The Multilink Universal and Multilink FX debug probes also support ARM Cortex-M devices from several manufacturers, and in addition they support NXP's Kinetis®, LPC, i.MX, ColdFire® V1/ColdFire+ V1, ColdFire V2-4, MPC55xx-57xx, DSC, HC(S)12(X), HCS08 and RS08 microcontrollers, and STMicroelectronics' SPC5. The Multilink FX also supports a few legacy NXP architectures.

By using a Multilink debug probe, the user can take advantage of debug mode to halt normal processor execution and use a PC to control the processor. The user can then directly control the target's execution, read/write registers and memory values, debug code on the processor, and program internal or external FLASH memory devices. The

Multilink Universal enables you to debug, program, and test your code on your board.

8.1.3 Software

Multilink debug probes work with NXP's MCUXpresso, Kinetis and S32 Design Studios, Codewarrior, as well as PEmicro's flash programmer, PROGPPCZ.

8.2 Cyclone LC and Cyclone FX

PEmicro's Cyclone LC and Cyclone FX programmers are extremely flexible tools designed for in-circuit flash programming, debugging, and testing of many **8-/16-/32-bit microcontrollers from NXP & STMicroelectronics**, as well as **ARM® Cortex® devices from a variety of manufacturers**, including NXP, STMicroelectronics, Texas Instruments, Atmel, Infineon, Cypress, Silicon Labs, OnBright, and more. These Cyclones include a 4.3" touchscreen LCD and an access panel which provides easy access to all debug headers. Cyclones programmers offer multiple communications interfaces (including USB, Ethernet, and Serial), stand-alone programming functionality, high speed data transfer, a status LCD, and many other advanced capabilities.

Cyclones include, or can add, these additional advanced features:

- ProCryption Security (RSA/AES image encryption and programming limits)
- Advanced Automation and Control Features (e.g., gang programming)
- SDHC Port Activation (external storage via SD memory cards)

These items are all standard with the more advanced Cyclone FX model, which also features significantly larger internal storage, security, and speed enhancements, and the ability to launch programming via barcode scanner. This helps make the Cyclone FX PEmicro's premier production programming solution.

8.2.1 Supported Devices

ARM Cortex devices (all Cyclone part numbers):

NXP:	Kinetis®, LPC, i.MX, Automotive, Sensors, Vybrid
Atmel:	SAMxxx
Cypress:	CCG®, EZ-BLE, FM3, PRoC-BLE, PSoC® 4, PSoC 5, PSoC 6
Infineon:	XMC
Silergy (Maxim):	AM0, AM1x, MAX716xx
Nordic Semi:	nRF51, nRF52

OnBright:	OB90Rxx
Silicon Labs:	EFM32, EFR32, SiM3
STMicroelectronics:	STM32
Texas Instruments:	LM3S, LM4, TM4C12x, SimpleLink
Toshiba:	TX00, TX03, & TX04

NXP 8-/16-/32-bit devices (“Universal” part number only):

S32, MPC55xx-57xx, ColdFire +/V1, ColdFire V2/3/4, DSC, HC(S)12(X), S12Z, HCS08, RS08, HC08 (MON08), MAC7xxx, Power MPC5xx/8xx. Also: STMicroelectronics’ SPC5 and STM8.



Figure 8-4: Cyclone LC Models



Figure 8-5: Cyclone FX Models

8.2.2 Key Features

- Many Supported Architectures
- Multiple Communications Interfaces
- USB 2.0 (Universal: Full-Speed; FX: High-Speed), Ethernet, and Serial interfaces
- On-Board Storage -- Cyclone LC: 16MB internal memory, Cyclone FX: 1GB internal memory. The Cyclone may be pre-programmed with non-volatile programming images and controlled via the touchscreen LCD, start button, or remotely from a PC (serial, USB, ethernet). Stand-alone programming operation does not require a PC.
- High-Speed Target Communications (FX Only): Cyclone FX improves on the already fast target communications speed of the Cyclone LC. The Cyclone FX is capable of download rates up to 75Mb/s.
- Power Switching: Allows switching of the target's power supply via Cyclone "power-in" and "power-out" jacks. On-board electromechanical relays handle the power switching. Power can also be provided to the target via the debug connection.

- Multiple Image Support: Multiple programming images may be stored in Cyclone memory. Cyclone LC: 8 images max; Cyclone FX: no practical limit.
- Touchscreen LCD Display: The 4.3" touchscreen display, in conjunction with the status LEDs and Start button, allows stand-alone control and configuration of the Cyclone.
- Serial Number Programming: The Cyclone can program dynamic data, such as serial numbers.
- Expansion Ports (FX Only): The Cyclone Universal FX includes these expansion ports:
 - - Programming control header (trigger programming/read status via pins)
 - - USB host/device port (for external USB peripherals such as barcode scanners)
- Cyclone FX advanced features that can be added to the Cyclone LC:
 - ProCryption Security - RSA/AES encryption of programming images and the ability to set limits on programming operations help keep valuable IP safe
 - External Storage: The SDHC port supports SDHC memory cards, for added storage capacity and flexibility.
 - Advanced Control/Automation - adds gang programming capability and more

8.2.3 Product Implementation

By connecting to a debug header on the target, the Cyclone can program, test, or debug internal memory on a supported processor or external flash connected to the processor's address/data bus, in-circuit. The processor or memory device can be mounted on the final printed circuit board before programming.

The Cyclone LC and Cyclone FX may be operated interactively via Windows-based programming applications, as well as under batch or dll commands from a PC. Once loaded with data by a PC a Cyclone can be disconnected and operated manually in a completely stand-alone mode via the touchscreen LCD menu and start button. The Cyclone's internal non-volatile memory allows the on-board storage of multiple programming images. The Cyclone FX also includes support for expandable memory via SDHC memory cards. When connected to a PC for programming or loading the

Cyclone can communicate via Ethernet, USB, or serial interfaces.

8.2.4 Software

The Cyclone LC and Cyclone FX come with intuitive configuration software and interactive programming software, as well as easy to use automated control software call. These Cyclones also function as full-featured debug interfaces, and are supported by software from PEmicro and third-party vendors.

9 PROGRAMMING UTILITIES

The following no-cost programming utilities are available on PE micro's website. www.pemicro.com, by navigating to Support -> Documentation & Downloads -> Utilities.

9.1 Serialize

The Serialize utility allows the generation of a .SER serial number description file. This graphical utility sets up a serial number which will count according to the bounds set by the user. The .SER file can be called by the PROG flash programmer to program a serial number into the target.

More information on how to use the Serialize utility can be found on PE micro's website at: www.pemicro.com/newsletter/experts_corner/2005_08_17/serialize.cfm.

9.2 Unsecure_12

The Unsecure12 utility unsecures HC(S)12(X) devices via certain PE micro's hardware interfaces: Multilinks and legacy Cyclones. It does not support the latest generation of Cyclone programmers. Unsecure12 support includes XExxx and Pxxx devices, and will work on 64-bit operating systems.

APPENDIX A - SETUP COMMANDS

Setup Commands are commands that each appear on separate lines of a .PCP programming algorithm file, starting in column one. They are used to initialize the target CPU when it is not possible to do so using the enable function, which must first be loaded into target ram before execution. All setup commands must appear before the first S record in the .PCP file or they will be ignored.

The setup commands are:

BLANK_MODULE_ONLY

This command has 17 characters. It indicates to the programmer that if a blank byte address or blank word address is provided they can only be used to enable a blank module command.

SHORT_TABLE

This command has 11 characters. It indicates to the programmer that the algorithm table has 16 bit entries as opposed to the normal 32 bit entries. It is used to save space when only a small RAM is available.

BLOCKING_MASK=mmmmmmmm/

This command has 23 characters. First it tells the programmer that only full blocks of data can be programmed into the device and that blocks must occur on a block boundary. The mask *mmmmmmmm* is used to select those address lines which occur within a block. For example, blocks of 8 bytes would have a mask of 00000007. The buffer provided in the target must in size be an integral multiple of the blocking size in bytes.

SET_TIMING=nn/

This command has 14 characters and tells the programmer that at the end of executing an enable, it should calculate *nn* timing parameters. Enable passes back an address in ix which points to the timing parameters in target RAM. The number in each timing word (stored by enable) is multiplied by 10 microsecond timing constant and stored back in the same location.

ADDRESS_PAGING=mmmmmmmm/oooooo/

This command has 33 characters and tell the programmer that some form of address paging is being used. Under these circumstances, the function **BEFORE_READ** must set up the paging configuration address. The mask *mmmmmmmm* is used to determine which bits of the address represent the page address so that page changes can be detected. The actual address read is calculated by the PROG software as (address and not(*mmmmmmmm*)) +

00000000.

NO_BASE_ADDRESS

or

NO_BASE_ADDRESS=bbbbbbbb/

This 15 character command version tells the prog software to use a base address of 0 and not to ask the user to enter one. The 25 character version is the same except it sets the base address to *bbbbbbbb*.

NO_ON_CHIP_RAM

This command has 14 characters and tells the programmer not to turn on the on chip ram. You must provide ram to run the calibration routines and load your .12P file S records. If not deactivated by this command, the on chip RAM is turned on after all other setup commands are executed.

NO_TIMING_TEST

This command has 14 characters and tells the programmer not to evaluate the target processor speed the initialization process. Instead, both timing constants are set to 1. This option is only used when programming timing functions are not needed.

WRITE_LONG=IIIIIII/aaaaaaaa/

This command has 29 characters. It writes the hex long *IIIIIII* to the hex address *aaaaaaaa* in the current space.

WRITE_WORD=www/aaaaaaaa/

This command has 25 characters. It writes the hex word *www* to the hex address *aaaaaaaa* in the current space.

WRITE_BYTE=bb/aaaaaaaa/

This command has 23 characters. It writes the hex byte *bb* to the hex address *aaaaaaaa* in the current space.

APPENDIX B - TABLE ENTRY

Users who wish to make significant modifications to a programming algorithm may need to modify the table entries in their assembly (.ASM) file. Table entries provide information to the PROG software, including what functions are in the algorithm and where they are located. Each table entry consists of 32 bits and must be in the following order:

Stack Address

Address of the stack during routine execution. The stack is initialized each time one of the user-supplied routines is called.

Buffer Address

Address of the buffer used to transfer data from the PC to the target. This is data to be placed into the module.

Buffer Length

Length of available buffer space in bytes. The buffer should be at least 4,096 bytes long in order to accommodate the largest possible S record.

Module Address

The physical address of the beginning of the module to be programmed or erased.

Module Length

Length of the module to be programmed in bytes.

Blank Bytes Address

The address of a routine to check a block of bytes to see if they are erased. R1 contains the starting address and R2 contains the number of bytes to check. Checking is done on a byte by byte basis. If R2<>0 on return then an error occurred at word address R1-1. R2 = 0.

Blank Words Address

The address of a routine to check a block of words to see if they are erased. R1 contains the starting address and R2 contains the number of bytes to check. Checking is done on a word by word basis. If R2<>0 on return then an error occurred at word address R1-2.R2 = 0.

Erase Bytes Address

The address of a routine to erase a block of bytes. R1 contains the starting address and R2 contains the number of bytes to erase. Erasing is done on a byte

by byte basis. R2 = 0.

Erase Long Address

The address of a routine to erase a block of longs. R1 contains the starting address and R2 contains the number of bytes to erase. Erasing is done on a word by word basis. If R2<>0 on return an erase error occurred. R2 = 0.

Erase Module Address

The address of a routine which erases the entire module. R1 contains the starting address to be erased, R2 contains the length in bytes. Returning to PROGARM with R2 non zero indicates an error.

Program Bytes Address

The address of a routine which programs a block of bytes residing in the buffer. R2 contains the length of the block in bytes. R1 contains the starting address at which they are to be programmed. R3 contains the address of the buffer. Returning with R2 non zero indicates an error.

Program Words Address

The address of a routine which programs a block of bytes residing in the buffer. R2 contains the length of the block in bytes. R1 contains the starting address at which they are to be programmed. R3 contains the address of the buffer. Returning with R2 non zero indicates an error.

On Volts Address

Routine to turn on the programming voltage to the module to effect an erase or program function. If the programming voltage is left on all the time, then this routine should be implemented only as a BGND instruction. If no routine is provided, then the user is asked to turn on the voltage.

Off Volts Address

Routine to turn off the programming voltage to the module. If the programming voltage is left on all the time, then this routine should be implemented only as a BGND instruction. If no routine is provided, then the user is asked to turn off the voltage.

Enable Address

Routine to set up the programming process. This routine is called once after a .12P module is selected and each time a command is executed. It is used to do things such as set up chip selects, turn devices on, etc. The chip can be set up using commands (such as WRITE_WORD) in the .12P file, however, these

commands are only done when a .12P file is selected.

Disable Address

Routine to provide a graceful shutdown of any target resources. Usually not required.

Before Read Address

Routine which is called before reading data from a module. Not required in most cases.

After Read Address

Routine which is called after reading data from a module. Not required in most cases.

User Function Address

Optional routine (and table entry) to provide an arbitrary user function if one is specified in the .xxP file. (See section on USER function).