



# PROG16Z

USER MANUAL





## PROG Software License 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. The software being installed 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 without the express written permission of P&E Microcomputer Systems. Under no conditions may you remove the copyright notices from this software or documentation.

This software requires the use of a license code to operate.

If you have purchased a PROG software license from P&E Microcomputer Systems and been issued a hardware-based license code (a license code that begins with V2), you may (1) install the provided hardware-based PROG license code into a single Cyclone or Multilink unit and (2) install this software on any computer with which the specific Multilink or Cyclone will be used. This gives you the ability to run this software on multiple computers, used by multiple users, with the Multilink or Cyclone hardware which has the hardware license code installed.

If you have purchased a PROG software license from P&E Microcomputer Systems and been issued a legacy computer based license code (a license code that begins with V1), this software is licensed as a single user license which means: (1) This software may be used by one individual user on up to two different computers, provided that the software is never used on the two computers at the same time, (2) P&E Microcomputer Systems 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 Microcomputer Systems for volume discounts and site licensing agreements.

P&E Microcomputer Systems does not assume any liability for the use of this software beyond the original purchase price of the software. 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.

©2016, 2018, 2019, 2020 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: 1.00  
June 2021



1	OVERVIEW.....	1
1.1	Programming Algorithms (.16P Files) .....	2
1.2	Start-Up Configuration.....	2
1.3	Manual Programming .....	2
1.4	Scripted Programming.....	3
1.5	Hardware Interfaces .....	3
1.6	Programming Utilities .....	3
2	PROGRAMMING ALGORITHMS .....	4
2.1	Algorithm File Contents .....	4
2.2	Algorithm Timing Considerations .....	6
3	PROGRAMMING COMMANDS.....	7
3.1	BM - Blank Check Module.....	8
3.2	CM - Choose Module .16P .....	8
3.3	CS - Choose Serial File.....	8
3.4	EM - Erase Module.....	8
3.5	EN - Erase If Not Blank .....	9
3.6	HE - Help.....	9
3.7	PB - Program Bytes.....	9
3.8	PM - Program Module .....	9
3.9	PR - Program Module Range .....	9
3.10	PS - Program Serial Number.....	9
3.11	PT - Program Trim Value .....	9
3.12	QU - Quit .....	10
3.13	RE - Reset chip .....	10
3.14	RELAYSOFF - Turn off the relays that provide power to the target.....	10
3.15	RELAYSON - Turn on the relays to provide power to the target.....	10
3.16	SA - Show Algorithm Source.....	10
3.17	SM - Show Module.....	11
3.18	SS - Specify S-Record .....	11
3.19	UM - Upload Module .....	11
3.20	UR - Upload Range .....	11



---

---

3.21	VC - Verify CRC Of Object File To Module .....	11
3.22	VM - Verify Module.....	11
3.23	VR - Verify Range .....	12
4	START-UP CONFIGURATION.....	13
5	CONNECTION MANAGER.....	15
5.1	Additional Settings.....	16
5.2	Connect and Choose Algorithm .....	18
6	MANUAL PROGRAMMING .....	19
6.1	Manual Programming Procedure .....	19
7	SCRIPTED PROGRAMMING (CPROG16Z).....	21
8	HARDWARE INTERFACES .....	22
8.1	Multilink FX.....	22
9	PROGRAMMING UTILITIES .....	24
9.1	Serialize.....	24
APPENDIX A -SETUP COMMANDS.....		25
APPENDIX B -TABLE ENTRY .....		27

## 1 OVERVIEW

PROG16Z is PE micro's programming software for Flash/EEPROM modules that are attached to a 68HC16 processor. PROG16Z talks to the processor's debug module using one of PE micro's compatible hardware interfaces. These interfaces connect a PC running Windows 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.

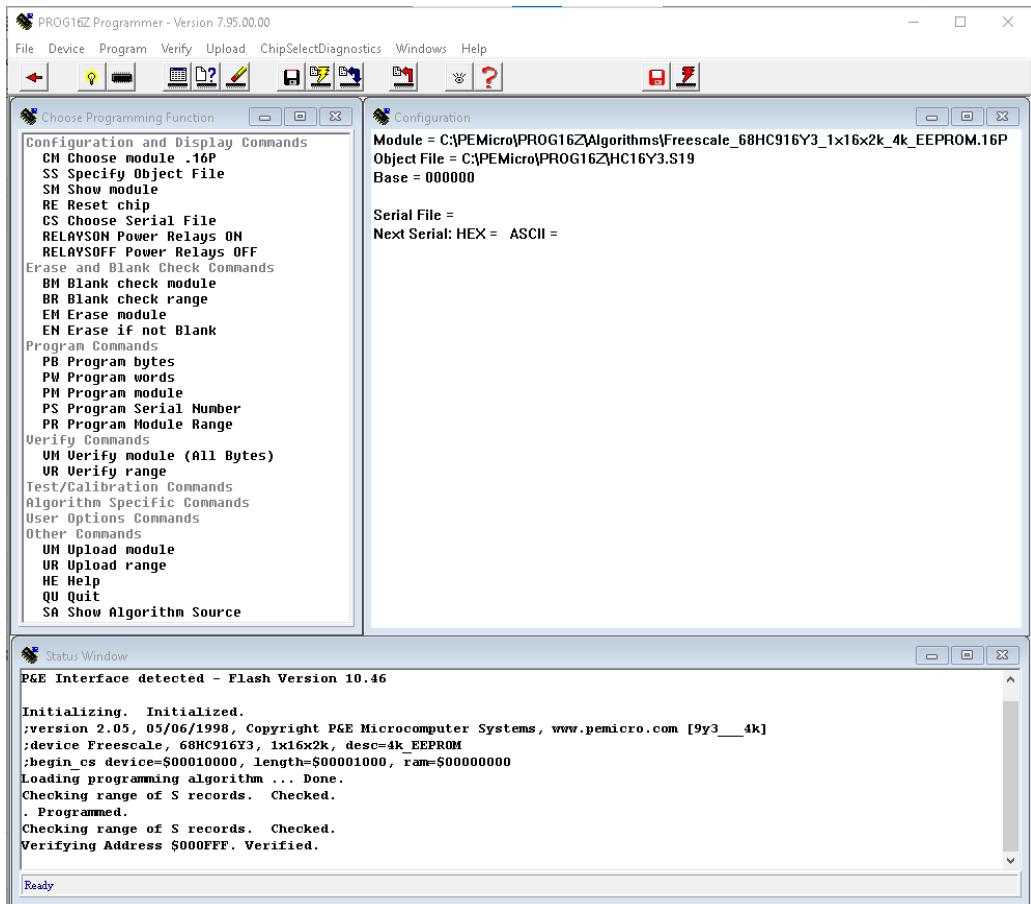


Figure 1-1: PROG16Z User Interface

As part of the programming procedure, the user will need to select a programming algorithm that will enable the PROG16Z 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 (.16P Files)

PROG16Z 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 (.16P) files that the user can modify to reflect the setup of their particular target interface. PROG16Z includes a library of these programming algorithms. For the most recent version of this library of algorithms, please visit our website, [www.pemicro.com](http://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 PROG16Z 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

PROG16Z 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 CPROG16Z, which is included with the PROG16Z software. More information about scripted programming is located in the accompanying CPROG16Z User Guide.

#### **1.5 Hardware Interfaces**

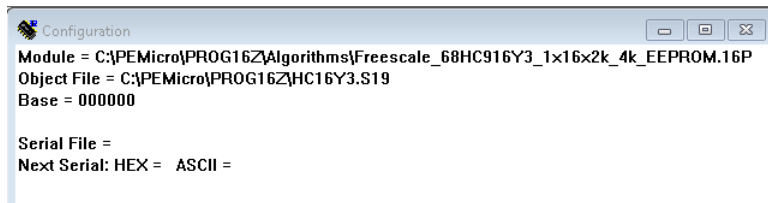
PROG16Z currently requires use of the Multilink FX debug probe. More information is available in **CHAPTER 8 – HARDWARE INTERFACES**.

#### **1.6 Programming Utilities**

PEmicro also offers some no-cost programming utilities to help the user perform certain tasks. More information is available in **CHAPTER 9 – PROGRAMMING UTILITIES**.

## 2 PROGRAMMING ALGORITHMS

PEmicro's .16P programming algorithm files define the functions necessary for PROG16Z to program a 68HC16 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 .16P 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 .16P file was written and what module on the target system it programs, as well as other useful information. If a specific .16P file is selected in PROG16Z, 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 .16P file. Each user statement in the .16P 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:





editors. The S records for a .16P file can be generated using most assemblers.

## 2.2 Algorithm Timing Considerations

Most current flash devices have an on-chip programming monitor. The processor passes a command to the flash device, such as Program Word, and the flash device executes this command. On all processors with On-Chip flash, and on some external flash devices, the timing is provided by the processor. In order to program the flash device according to specification, the programming software on the PC has to know how fast the target processor is running. By default, the PROG16Z software tries to determine automatically how fast the target is running by loading a delay routine in the processor and timing how long it takes to execute. Under a multitasking environment, such as Windows 7/8/10, although they are usually very accurate, these timing measurements are not always correct.

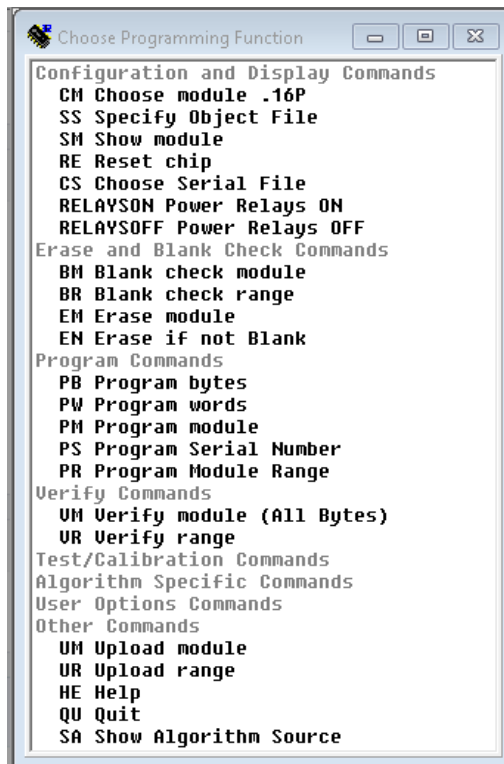
PEmicro addresses this by providing a command-line mechanism that allows the user to inform the PROG software how fast the target processor is running. This ensures that the timing in the algorithms is always correct. To do this, the user would include the **FREQ** identifier on the executable command-line, followed by the **INTERNAL** clock frequency in Hertz. For instance, if your processor is an NXP MC68HC16Y3 with a bus frequency of 4.194 MHz, your command-line parameters should look like this:

```
PROG16Z freq 4194000
```

See **CHAPTER 4 – START-UP CONFIGURATION** for more information about how to use command-line parameters.

### 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**

**Note:** 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 PROG16Z commands used in manual programming. These same commands are also used in scripted programming. For more information about scripted programming, see the **CPROG16Z 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 .16P**

The user is presented with a list of available .16P files. Each .16P 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\_68HC916Y3\_1x16x16k\_32k-1\_EEPROM.16P specifies how to program the 32KB block on a MC68HC16Y3 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 .16P 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 .16P file is selected by using the arrow keys to highlight the file name and then pressing ENTER. The currently selected .16P file is shown in the .16P file selected window. After a .16P 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 .16P files, such as those for external flash algorithms, will prompt the user for the base address of the module.

### **3.3 CS - Choose Serial File**

Used to select a Serial File generated by PEmicro's Serialize Utility.

### **3.4 EM - Erase Module**

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

### **3.5 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.

### **3.6 HE - Help**

Opens this PROG16Z user manual.

### **3.7 PB - Program Bytes**

The user is prompted for a starting address, which must be in the module. The user is then shown an address and a byte. Pressing ENTER shows the next location. The user can also enter in hex a byte to be programmed into the current location. In addition, the symbols +, -, or = may be appended to the value being written. They correspond respectively to increase the address (default), decrease the address, and hold the address constant. Failure to program a location, entering an invalid hex value or exceeding the address range of the module will exit the program bytes window. If a location fails to program, an error message will be returned.

### **3.8 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.

### **3.9 PR - Program Module Range**

Program the object file within specified starting and ending addresses.

### **3.10 PS - Program Serial Number**

Program a serial number according to the .SER file selected using the CS command.

### **3.11 PT - Program Trim Value**

Programs the non-volatile trim register or user-provided flash location. Devices that support trimming and have a dedicated non-volatile trim register will automatically program to that location. For devices that support trimming and do not have a

dedicated non-volatile trim register, the user will need to provide a flash location where the trim values will be stored. The format and exact number of bytes programmed is device dependent.

Click on the "change" button found within the Configuration window to change the desired trim reference frequency between the default value or a custom value. For more information, please read the chip's reference manual about the clock generation modules.

### **3.12 QU - Quit**

Terminates PROG16Z and returns to Windows.

### **3.13 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 debug mode.

### **3.14 RELAYSOFF - Turn off the relays that provide power to the target**

( only)

Includes a power down delay if specified. Especially useful for users who want to power cycle their board before running tests, allow their bootloader to run, or have the application code run after programming.

### **3.15 RELAYSON - Turn on the relays to provide power to the target**

( only)

Includes a power up delay if specified. The voltage supplied will be based on the last voltage setting specified. For Cyclone users, the CHANGEV command can change the voltage value. Especially useful for users who want to power cycle their board before running tests, allow their bootloader to run, or have the application code run after programming.

### **3.16 SA - Show Algorithm Source**

Show the algorithm's source

### **3.17 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.

### **3.18 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.

### **3.19 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.

### **3.20 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.

### **3.21 VC - Verify CRC Of Object File To Module**

Verify the flash against the object file using CRC calculations.

### **3.22 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.

### 3.23 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 .16P file.



---

---

## 4 START-UP CONFIGURATION

The PROG16Z 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 PROG16Z 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:

```
PROG16Z [freq n] [v] [reset_delay][interface=x]
        [port=y]
```

### Where:

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

- |                      |  |
|----------------------|--|
| <b>[reset_delay]</b> | Wait n milliseconds after a reset before entering debug mode.  |
| <b>[freq n]</b>      | This allows the user to specify the exact speed of the target. If this is not specified, the programmer tries to calculate the target's speed. The frequency specified is the INTERNAL clock frequency of the target. See <b>Section 2.2 - Algorithm Timing Considerations</b> for more information. |
| <b>[v]</b>           | 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.   |
| <b>[interface=x]</b> | where x is one of the following: (See examples section)<br>USBMULTILINK (supports Multilink Universal, Multilink Universal FX, and OSJtag)   |
| <b>[port=y]</b>      | 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
```

**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.

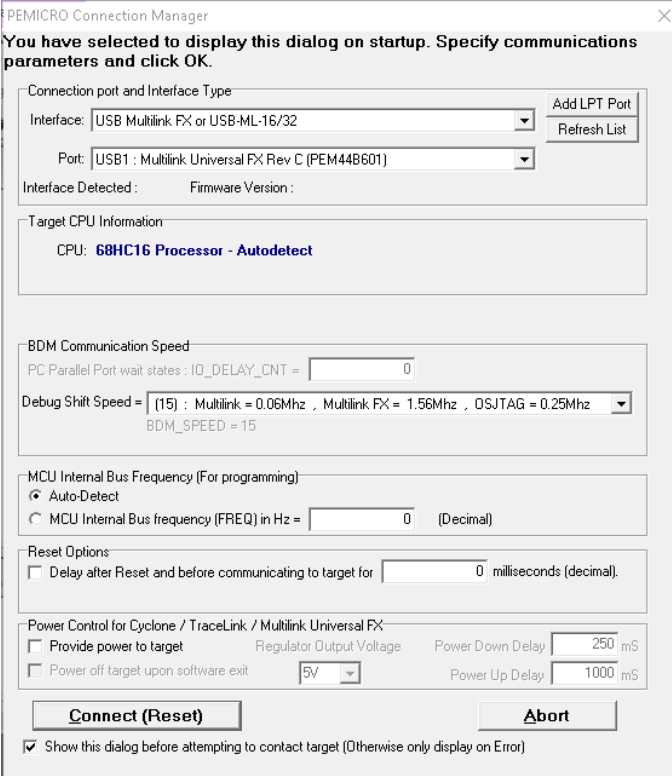
Examples:

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

## 5 CONNECTION MANAGER

Before programming your device, you will need to connect to your target using a compatible PE micro hardware interface. Interface options for PROG16Z 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:



PEMICRO Connection Manager

You have selected to display this dialog on startup. Specify communications parameters and click OK.

Connection port and Interface Type

Interface: USB Multilink FX or USB-ML-16/32

Port: USB1 : Multilink Universal FX Rev C (PEM44B601)

Interface Detected: Firmware Version:

Target CPU Information

CPU: 68HC16 Processor - Autodetect

BDM Communication Speed

PC Parallel Port wait states: IO\_DELAY\_CNT = 0

Debug Shift Speed = (15) : Multilink = 0.06Mhz , Multilink FX = 1.56Mhz , OSJTAG = 0.25Mhz

BDM\_SPEED = 15

MCU Internal Bus Frequency (For programming)

Auto-Detect

MCU Internal Bus frequency (FREQ) in Hz = 0 (Decimal)

Reset Options

Delay after Reset and before communicating to target for 0 milliseconds (decimal).

Power Control for Cyclone / TraceLink / Multilink Universal FX

Provide power to target Regulator Output Voltage Power Down Delay 250 mS

Power off target upon software exit 5V Power Up Delay 1000 mS

**Connect (Reset)** **Abort**

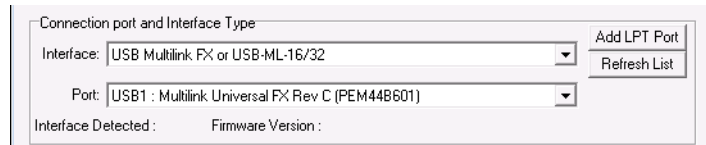
Show this dialog before attempting to contact target (Otherwise only display on Error)

**Figure 5-1: Connection Manager Dialog**

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

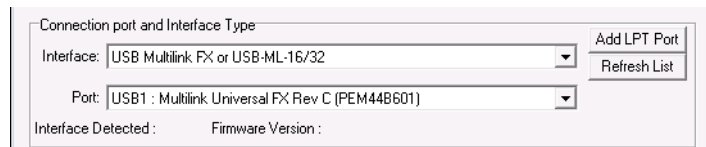
Use the Interface drop-down menu to choose the type of interface that you plan to

use.



**Figure 5-2: Connection Manager - Select Interface**

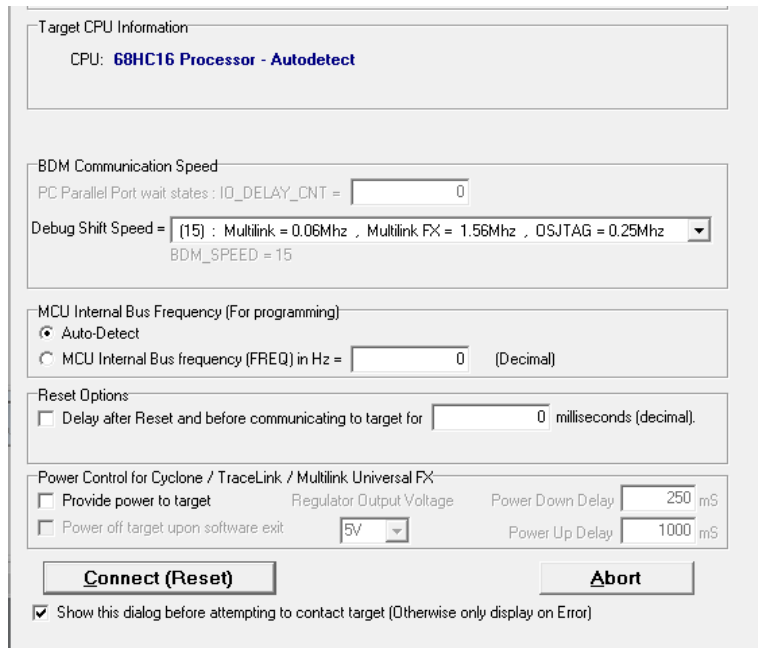
Then select the interface from those available, which are listed in the Port drop-down list. The Refresh List button to the right may be used to update the list of available interfaces:



**Figure 5-3: Connection Manager - Select Port**

## 5.1 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 5-4: Connection Manager - Additional Settings**

### 5.1.1 BDM Communications Speed

This software can automatically detect the proper communication speed to establish a connection with the target, but debug shift speed can also be set manually using the drop-down box.

### 5.1.2 Reset Options

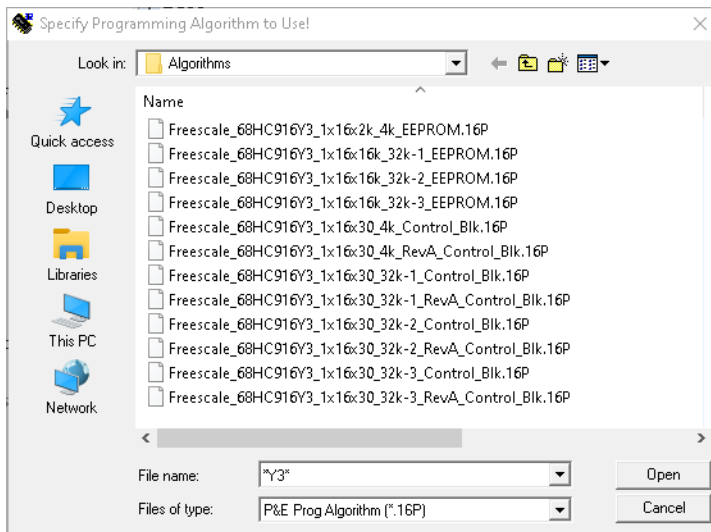
If your board has any active components connected to your RESET signal such as a supply voltage supervisory circuit or a reset monitor, the RESET signal may have a longer rise time. This option can set a delay before beginning communication to give time for RESET to stabilize. A typical value is 300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

### 5.1.3 Power Control for Cyclone / TraceLink / Multilink Universal FX

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

## 5.2 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:

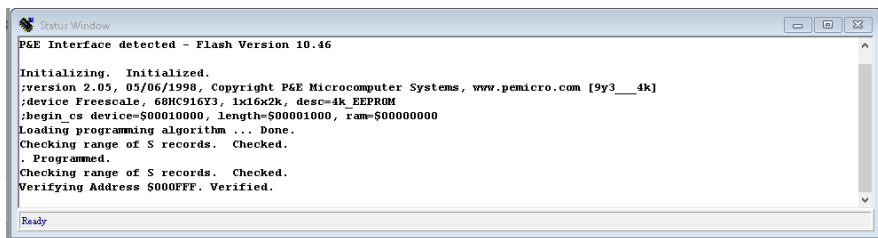


**Figure 5-5: Select Algorithm**

With the appropriate algorithm selected, 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.



```

P&E Interface detected - Flash Version 10.46

Initializing.  Initialized.
;version 2.05, 05/06/1998, Copyright P&E Microcomputer Systems, www.pemicro.com [9y3__4k]
;device Freescale, 68HC916Y3, 1x16x2k, desc=4k_EEPROM
;begin_cs device=$00010000, length=$00001000, ram=$00000000
Loading programming algorithm ... Done.
Checking range of S records.  Checked.
.  Programmed.
Checking range of S records.  Checked.
Verifying Address $000FFP.  Verified.

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 PROG16Z software, choose the programming algorithm by selecting the appropriate .16P file. Double clicking the "CM" (Choose Module) command will allow you to select the algorithm you wish to use.
3. After you select the .16P 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 (CPROG16Z)

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 CPROG16Z, which is included with the PROG16Z software. When you run the CPROG16Z.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 CPROG16Z scripted programmer, please see the **CPROG16Z User Guide**.

## 8 HARDWARE INTERFACES

PEmicro's Multilink Universal FX are compatible hardware interfaces for use with PROG16Z. The Multilink FX is a development tool that communicates via USB and will enable you to debug your code and program it onto your target. The not implemented are more versatile and robust production tools that communicates via Ethernet, USB, or Serial Port, and include advanced security, automation, and storage features as well as faster programming speeds.

Below is a review of their features and intended usage.

### 8.1 Multilink FX

PEmicro's Multilink FX debug probe offers 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 FX easy to use and fully capable of fast-paced debugging and programming.

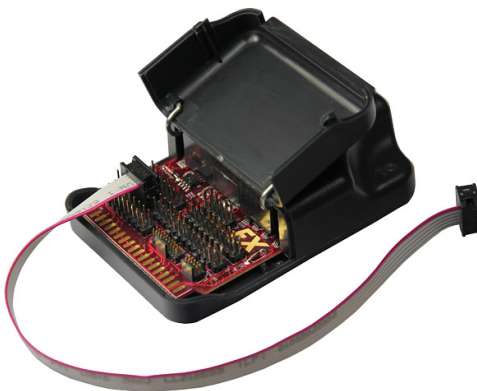


Figure 8-2: Multilink FX debug probe (open for access to headers)

---

### 8.1.1 Key Features

- Programming and debugging capabilities
- Can provide power to target device
- 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 FX debug probe allows a Windows 7/8/10 PC access to the debug mode on the 683xx target device via JTAG/SWD protocols (where applicable). It also supports ARM Cortex-M devices from several manufacturers, in addition to support for 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.

By using a Multilink FX 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 FX debug probes work with NXP's MCUXpresso, Kinetis and S32 Design Studios, Codewarrior, as well as PEmicro's flash programmer, PROG16Z.

## **9 PROGRAMMING UTILITIES**

The following no-cost programming utilities are available on PE micro's website. [www.pemicro.com](http://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](http://www.pemicro.com/newsletter/experts_corner/2005_08_17/serialize.cfm).

---

---

## APPENDIX A - SETUP COMMANDS

Setup Commands are commands that each appear on separate lines of a .16P 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 .16P 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 1KB 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).