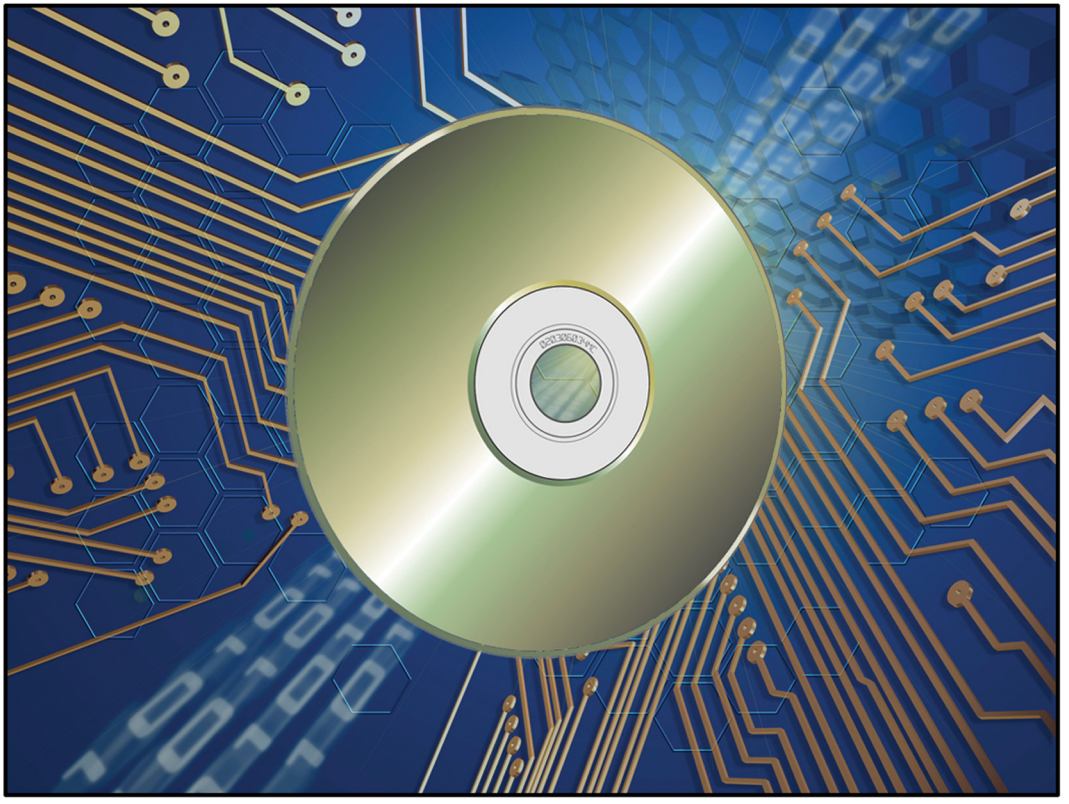
PROGACMP

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. ARM is a registered trademark and Cortex is a trademark of ARM Limited. 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: 1.10b
April 2021

1 OVERVIEW

PROGACMP is PEmicro's programming software for Flash/EEPROM modules that are attached toARM Cortex-M processors from NXP and many other manufacturers. PROGACMP talks to the processor's debug module using one of PEmicro'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.

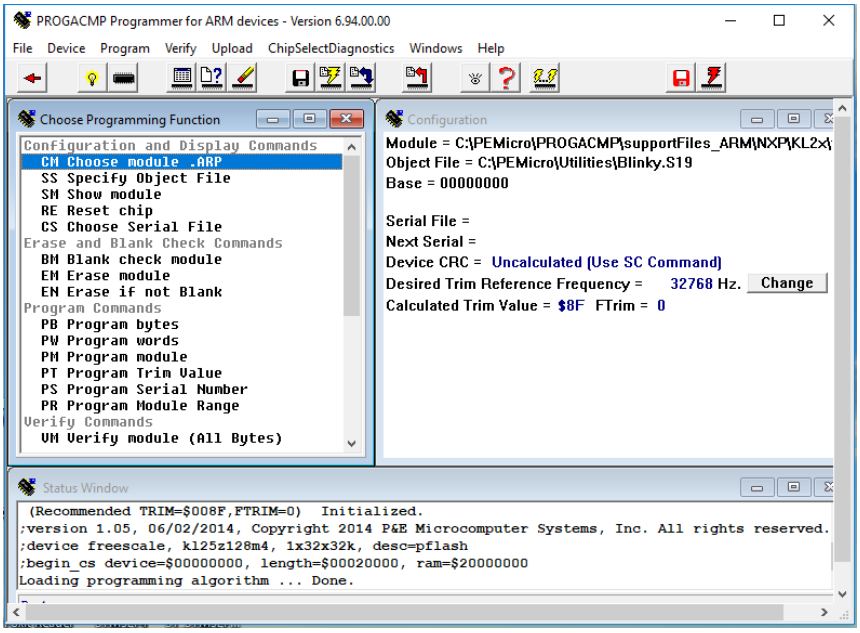


Figure 1-1: PROGACMP User Interface

As part of the programming procedure, the user will need to select a programming algorithm that will enable the PROGACMP 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 Supported Devices

PROGACMP supports ARM Cortex-M devices from NXP, STMicroelectronics, and many other manufacturers:

Atmel:	SAMxxx
Cypress:	CCG2, CCG3PA, EZ-BLEPSoC-PRoC, EZ-BLE-PSoC6, FM3 PRoC-BLE, PSoC4®, PSoC5®, PSoC6®, Traveo II
GigaDevice:	GD32
Infineon:	XMC
Maxim Integrated:	DARWIN
Nordic Semi:	nRF51, nRF52
NXP:	Kinetis®, LPC, i.MX, Automotive, Sensors, Vybrid
ON Semiconductor:	RSL10
OnBright:	OB90Rxx
Redpine Signals:	WiSeMCU
Renesas:	Synergy (S5/S7)
Silergy:	AM0x, AM1x, MAX716xx
Silicon Labs:	EFM32, EFR32, SiM3
STMicroelectronics:	Bluetooth, STM32
Texas Instruments:	LM3S, LM4, SimpleLink, TM4C12x
Toshiba:	TX00, TX03, & TX04
WIZnet:	W7500x

Please reference pemicro.com/ARM to determine if your specific device is su pported. Users may request support for an unsupported device, free of charge.

1.2 Programming Algorithms (.ARP Files)

PROGACMP 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 (.ARP) files that the user can modify to reflect the setup of their particular target interface. PROGACMP 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, PEmicro 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 1 – PROGRAMMING ALGORITHMS**.

1.3 Start-Up Configuration

Certain programming parameters can be adjusted when launching the PROGACMP 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 1 – START-UP CONFIGURATION**.

1.4 Manual Programming

PROGACMP 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 1 – MANUAL PROGRAMMING**.

1.5 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 CPROGACMP, which is included with the PROGACMP software. More information about scripted programming is located in the accompanying CPROGACMP User Guide..

1.6 Hardware Interfaces

More information is available in **CHAPTER 8 – HARDWARE INTERFACES**.In

addition to PROGACMP programming procedures, this manual discusses hardware interfaces that may be used in conjunction with the PROGACMP. For supported processors, PEmicro typically offers both value-oriented development solutions and more robust and versatile production solutions. You can learn about these interfaces in **CHAPTER 3 – HARDWARE INTERFACES**.

1.7 Programming Utilities

PEmicro also offers some no-cost programming utilities to help the user perform certain tasks. More information is available in **CHAPTER 1 – PROGRAMMING UTILITIES**.

2 PROGRAMMING ALGORITHMS

PEmicro's .ARP programming algorithm files define the functions necessary for PROGACMP to program an ARM Cortex-M processor's internal flash or connected external Flash/EEPROM. After you choose the appropriate algorithm, it will appear in the Configuration Window.

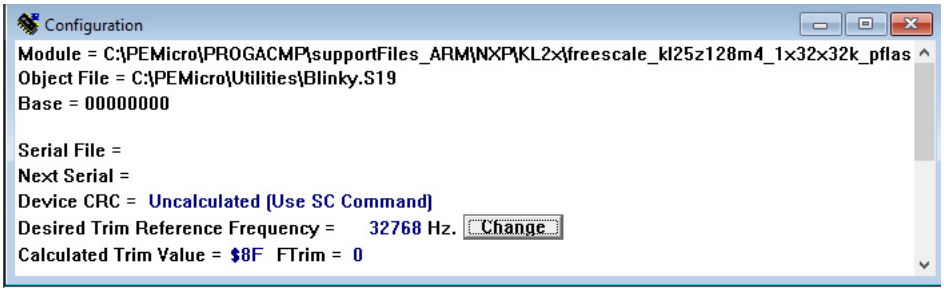


Figure 2-1: Configuration Window

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 .ARP programing 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 .ARP file was written and what module on the target system it programs, as well as other useful information. If a specific .ARP file is selected in PROGACMP, 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 .ARP file. Each user statement in the .ARP 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:

```
USER=uuuuuuuuuuuuuuuuuuuuuuuuNpppppppppp/llllllll/
uuuuuuuu/
```

Where:

USER= is the keyword to identify the line

uuuuuuuuuuuuuuuuuuuuuu is the 22 character string placed in the selection menu window on the PC screen. The first few characters define the menu select function and should be unique.

N is a single numeric digit between 0 and 4. If it is zero, the program will not ask for a user parameter. If it is non-zero, the user will be asked for a parameter with N hexadecimal digits. This parameter will be passed to the corresponding user routine in RAM.

pppppppppp is the 10 character prompt used to solicit the user input parameter.

/ is required for error checking.

llllllll is an 8 hex character lower bound on the user parameter.

uuuuuuuu is an 8 hex character upper bound on the user parameter.

2.1.3 Setup Commands

Setup Commands 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. Setup commands appear alone on a separate line of the .ARP file starting in column one. All setup commands must appear before the first S record in the .ARP file or they will be ignored. For a list of valid Setup Commands please see **APPENDIX A - SETUP COMMANDS**.

2.1.4 S-Records

Any line in the programming algorithm file starting with an "S" in column 1 is considered an S-record. S1, S2 and S3 records are allowed. S7, S8 and S9 termination records are ignored. PROGACMP uses the address field on the first S record detected in the file as the starting address of target RAM. Each .ARP file in the library contains a programming algorithm for a particular device.

The S records are loaded into on chip RAM on the ARM Cortex-M microcontroller and provide the functions necessary to carry out the functions specified below. All other records are written to the screen when the .ARP file is selected for programming. PROGACMP programming algorithm files must have the DOS filename extension

".ARP" in order for PROGACMP to find them. The files are in ASCII and are thus readable using most text editors. The S records for a .ARP 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 PROGACMP 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.

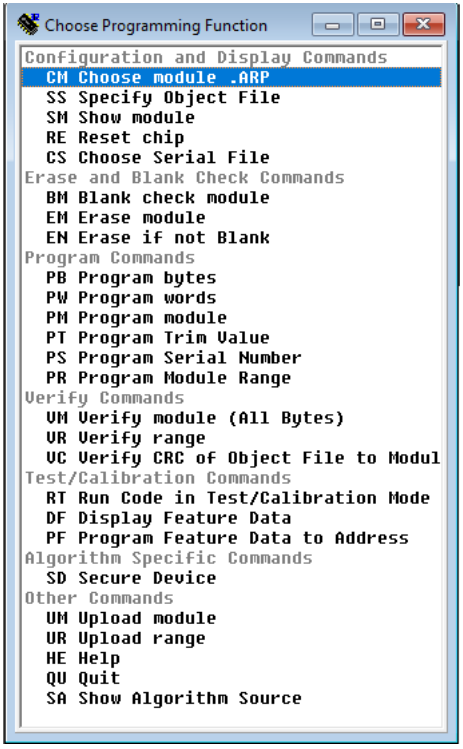
PEmicro addresses this by providing a command-line mechanism that allows the user to inform the PROG software how fast the target processor is running. The ensures that the timing in the algorithms is always correct. To do this, the user would include the FREQ identifier on the executable command-line, followed by the INTERNAL clock frequency in Hertz. For instance, if your processor is an NXP MK40X256 with a bus frequency of 20MHz, your command-line parameters should look like this:

```
PROGACMP freq 20000000
```

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.



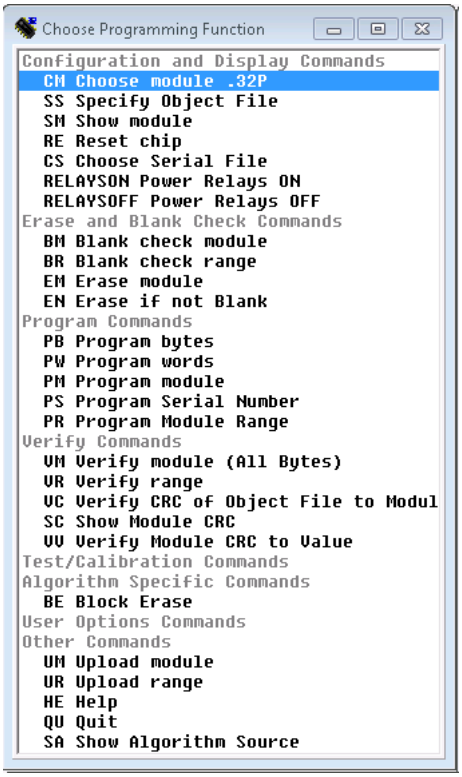


Figure 3-1: Choose Programming Function Window

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 PROGACMP commands used in manual programming. These same commands are also used in scripted programming. For more information about scripted programming, see the **CPROGACMP User Guide**.

3.1 BE - Block Erase

Erases an individual block of the flash memory. The user must specify a valid block number to erase.

3.2 BE - Block Erase

Some devices allow the BE command. This command erases individual blocks or sectors.

3.3 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.

3.4 BR - Blank Check Range

This command checks to see if a specified range of locations has been erased. The user is prompted for the starting and ending addresses. These addresses must lie within the addressing range of the module or an error will be returned. If the range is not erased, the first non-blank location is given along with its contents.

3.5 CA - Clear All Locks

Unlocks flash memory so it can be erased or programmed, only required for certain flash devices.

3.6 CHANGEV - Change the voltage provided to the target

(Cyclone only)

The user will be prompted to enter a value between 0.00 and 5.00, inclusive, that specifies the new voltage. When CHANGEV is executed the Cyclone will immediately change to that voltage. If the Cyclone relays are off prior to calling this command, then the relays will turn on and set the new voltage value when this command is executed. Note that too low of a voltage value may put the device into low-power mode which can lose debug communication altogether. The user should make sure the Cyclone's jumper settings are set correctly to send the power to the right ports.

3.7 CM - Choose Module .ARP

The user is presented with a list of available .ARP files. Each .ARP 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_MK40X256_1x32x64k_PFlash.ARP specifies how to program the 256K PFlash block on a MK40X256 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 .ARP 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 .ARP file is selected by using the arrow keys to highlight the file name and then pressing ENTER. The currently selected .ARP file is shown in the .ARP file selected window. After a .ARP 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 .ARP files, such as those for external flash algorithms, will prompt the user for the base address of the module.

3.8 CS - Choose Serial File

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

3.9 CU - Create/Modify User Options File

Launches a window that allows the user to either a) create a new user options (.OPT) file from scratch, or b) modify an existing .OPT file. See **Section 5.4 - User Options**.

3.10 EB - Erase Byte Range

This command erases bytes in a specified range of locations. The user is prompted for the starting and ending addresses. These addresses must lie within the addressing range of the module or an error will be returned. If the range is not erased, the first non-blank location is given along with its contents.

3.11 EM - Erase Module

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

3.12 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.

3.13 EW - Erase Word Range

This command erases words in a specified range of locations. The user is prompted

for the starting and ending addresses. These addresses must lie within the addressing range of the module or an error will be returned. If the range is not erased, the first non-blank location is given along with its contents.

3.14 HE - Help

Opens this PROGACMP user manual.

3.15 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.

3.16 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.

3.17 PR - Program Module Range

Program the object file within specified starting and ending addresses.

3.18 PS - Program Serial Number

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

3.19 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.

3.20 PU - Program User Options

Writes the values specified by the selected user option (.OPT) file. See **Section 5.4 - User Options**.

3.21 PW - Program Words

This command may be active, depending on the device/algorithm you are using. The user is prompted for a starting address, which must be in the module. The user is then shown an address and a word. Pressing ENTER shows the next location. The user can also enter in hex a word 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 words window. If a location fails to program, an error message will be returned.

3.22 QU - Quit

Terminates PROGACMP and returns to Windows.

3.23 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.

3.24 RELAYSOFF - Turn off the relays that provide power to the target

(Multilink FX & Cyclone 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.

3.25 RELAYSON - Turn on the relays to provide power to the target

(Multilink FX & Cyclone 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.

3.26 SA - Show Algorithm Source

Show the algorithm's source

3.27 SC - Show Module CRC

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

3.28 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.

3.29 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.

3.30 SU - Specify User Options File

Allows the user to specify an existing user options (.OPT) file for programming. See **Section 5.4 - User Options**.

3.31 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.

3.32 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.

3.33 VC - Verify CRC Of Object File To Module

Verify the flash against the object file using CRC calculations.

3.34 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.

3.35 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 .ARP file.

3.36 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.

3.37 SD - Secure Device

Secures the chip after a power-on-reset. Please read the chip's reference manual for more information. Depending on the chip's architecture, the method of unsecuring the chip will differ. Unsecuring the chip will erase the flash on the next attempt to enter debug mode, in order to prevent flash data from being viewed.

3.38 EP - Erase Page

Erase a specific page of memory. Please read the chip's reference manual to determine the size of each flash page.

4 START-UP CONFIGURATION

The PROGACMP 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 PROGACMP 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:

```
PROGACMP [bdm_speed n] [freq n] [v] [interface=x]
         [port=y]
```

Where:

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

[bdm_speed n] This option allows the user to set the BDM shift clock speed of PEmicro's BDM interfaces. This integer value may be used to determine the speed of communications according to the following equations:

Cyclone : (50000000/(2*N+5)) Hz

Multilink : (1000000/(N+1)) Hz

Multilink FX : (25000000/(N+1)) Hz

Tracelink : (50000000/(2*N+5)) Hz

[freq n] 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 **Section 2.2 - Algorithm Timing Considerations** for more information.

[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)

CYCLONE

TRACELINK

OpenSDA

[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, Tracelink, 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 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

Example

CPROGACMP C:\ENGINE.CFG Interface=USBMULTILINK Port=USB1

Opens CPROGACMP with the following options:

- Run the C:\ENGINE.CFG script

- Interface is Multilink, first cable detected.

Example

CPROGACMP C:\ENGINE.CFG Interface=CYCLONE Port=209.61.110.251

Opens CPROGACMP with the following options:

- Run the C:\ENGINE.CFG script

- Interface is Cyclone LC & Cyclone FX via the Ethernet Port with an IP address of 209.61.110.251

Example 3

CPROGACMP C:\ENGINE.CFG Interface=USBMULTILINK Port=USB1 bdm_speed 0

Opens CPROGACMP with the following options:

- Run the C:\ENGINE.CFG script

- Interface is Multilink, first cable detected.

- BDM shift clock speed set to 1,000,000 Hz. [bdm_speed n] = USB Multilink Universal: (1,000,000/(N+1)) Hz. For n = 0, BDM shift clock speed for Multilink = (1,000,000/(0+1)) Hz = 1,000,000 Hz

5 CONNECTION MANAGER

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

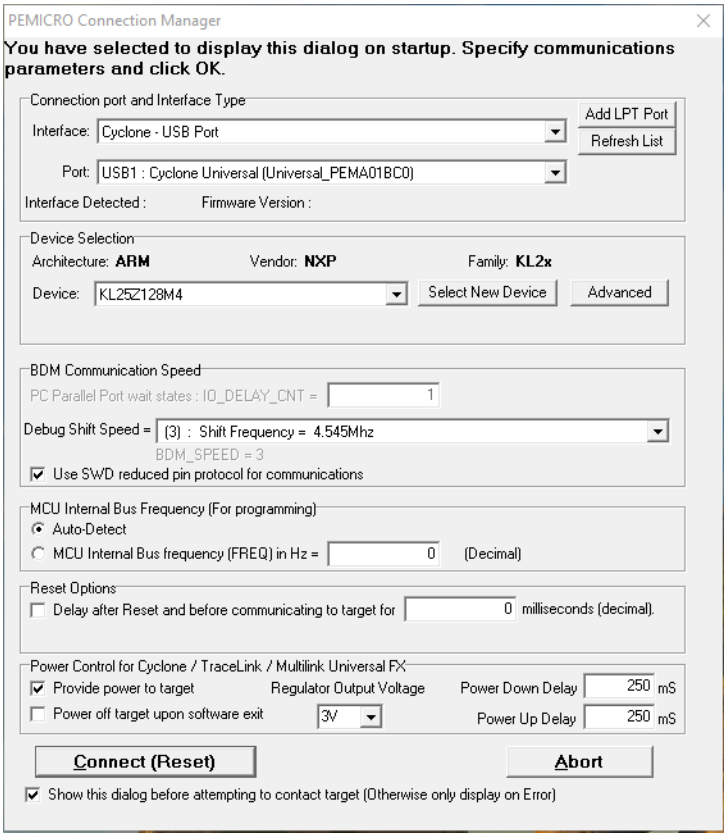


Figure 5-1: Connection Manager Dialog

The Connection 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.

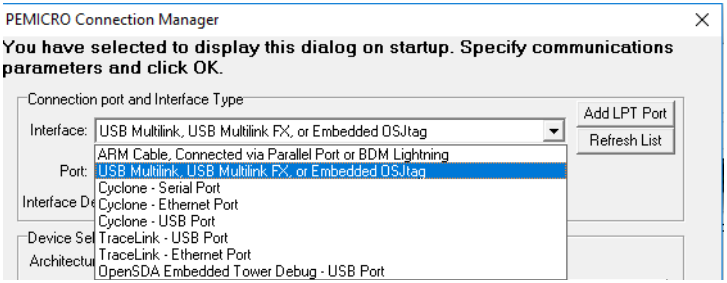


Figure 5-2: Connection Manager - Select Interface

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:

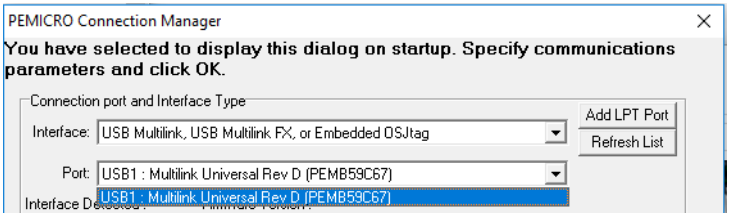
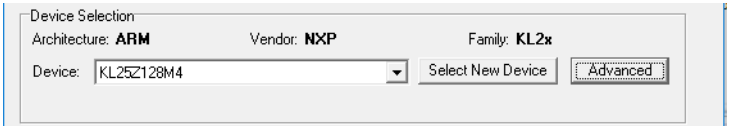


Figure 5-3: Connection Manager - Select Port

The next section of the Connection Manager is Device Selection.

Figure 5-4: Connection Manager - Device Selection



Click on Select New Device to open the Device Selection Dialog. Drill down the device tree to select your device. Optionally, you may right-click to copy the device string text for your device (e.g., when creating a CPROG script).

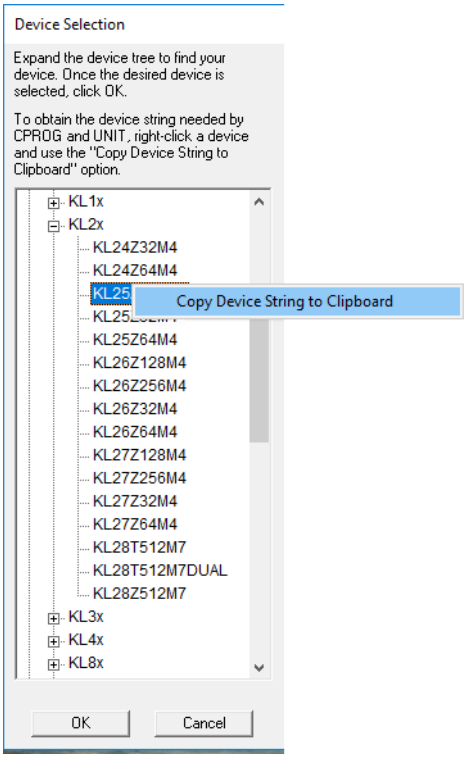


Figure 5-5: Device Selection Dialog

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).

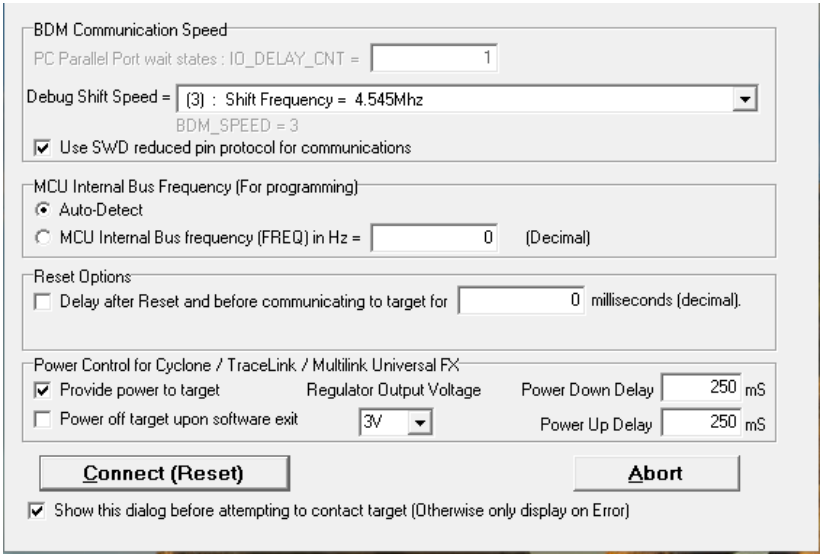


Figure 5-6: Connection Manager - Additional Settings

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 MCU Internal Bus Frequency (For programming)

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

5.1.3 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.

5.1.4 Power Control for Cyclone / TraceLink / Multilink Universal FX

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

5.2 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:

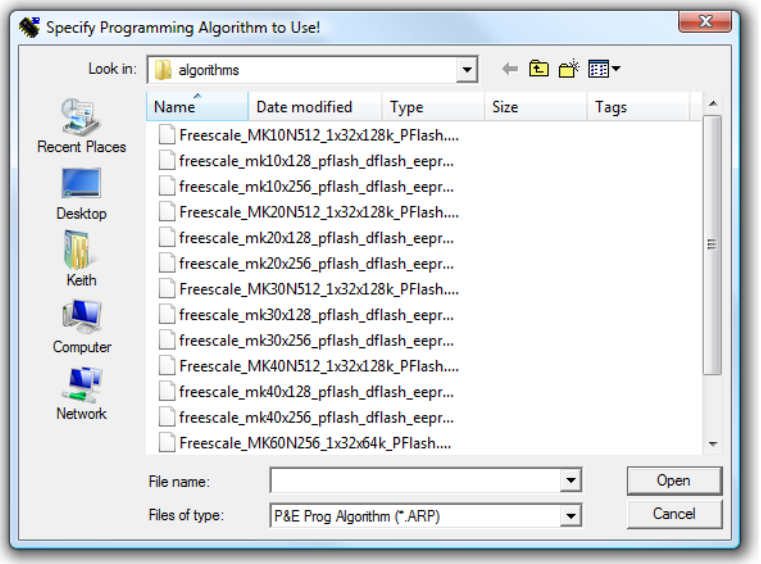


Figure 5-7: Select Algorithm

With the appropriate algorithm selected, you are ready to begin programming.

5.3 JTAG Daisy Chain

PROGACMP supports JTAG daisy chain configurations. This type of configuration is desirable if the user wants to share a single debug connector across multiple JTAG devices.

To start, it is required that the user selects the JTAG communications mode by clearing the following checkbox:

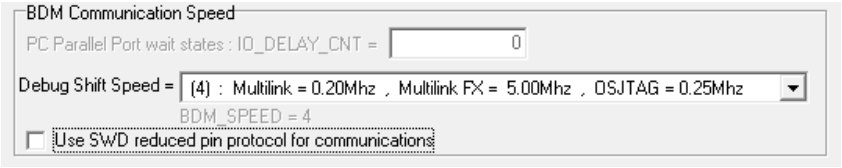


Figure 5-8: Clear Checkbox To Use JTAG Communications Mode

Afterwards, the daisy chain settings are configured by clicking on the "Advanced" button in the Connection Manager. A new dialog will appear, allowing the user to specify their exact daisy chain setup.

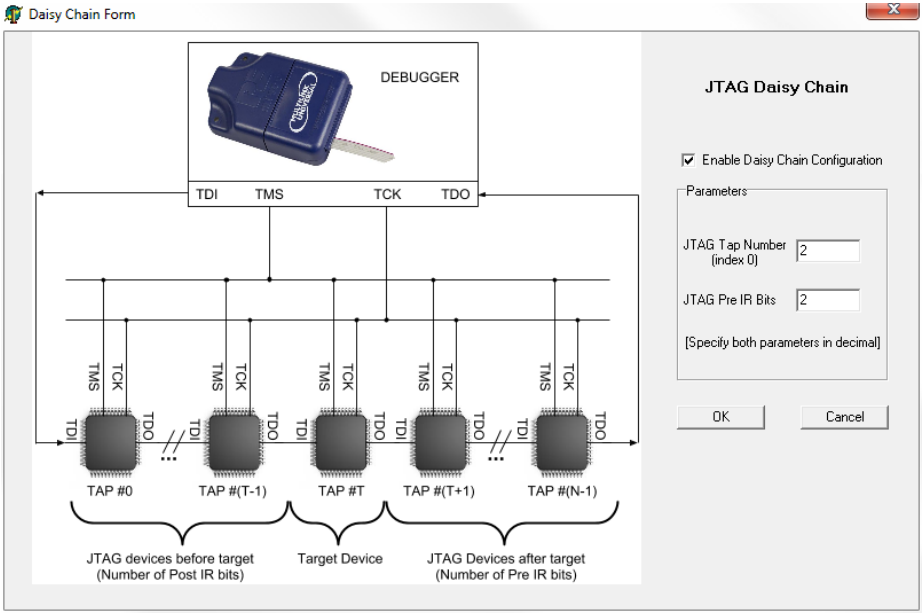


Figure 5-9: JTAG Daisy Chain Setup Dialog

As an example, consider a daisy chain containing two devices (TAP #0 and TAP #1) and both devices have a 4-bit JTAG IR. The following settings would access the first device in the chain (TAP #0):

JTAG Tap Number 0

JTAG Pre IR Bits 4

To access the second, and last device in the chain (TAP #1), use the following settings:

JTAG Tap Number 1

JTAG Pre IR Bits 0

For more information about how to daisy chain MCUs with PEmicro's Multilink and Cyclone interfaces, see:

http://www.pemicro.com/blog/index.cfm?post_id=136

5.4 User Options

Some ARM devices have areas of flash memory dedicated to programming user configuration data. As some writes to such areas can be sensitive or permanent, it is important that the developer is able to write these options correctly the first time and avoid mis-programming adjacent options that they wish to leave untouched.

Previously an object file was used to support the programming of user options. As of version 7.78 of PROGACMP, a set of three new commands has been introduced:

- Create/Modify User Options File (CU)
- Specify User Options File (SU)
- Program User Options (PU)

These commands allow the developer to individually program user options through the use of an IDE. A list of supported devices is available at:

http://www.pemicro.com/blog/index.cfm?post_id=177

5.4.1 Create/Modify User Options File (CU)

When the Create/Modify User Options File (CU) command is selected in interactive PROGACMP it allows the developer to either: a) create a new user options (.OPT) file from scratch, or b) modify an existing .OPT file. This JSON-type file represents the user options that the developer wishes to write, and will be used by the Specify User Options File (SU) and Program User Options (PU) commands later.

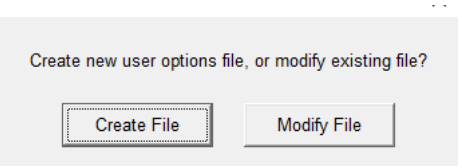


Figure 5-10: CU Command Dialog

a. If "Create File" is selected, a new window will open, in which the names of existing user area(s) and included options are displayed. In interactive PROGACMP only, the current value of each option will also be displayed. For each user option, the developer will have the ability to either "Write [a new] Value" or "Leave [the option] Unmodified." New values must be written in hexadecimal format.

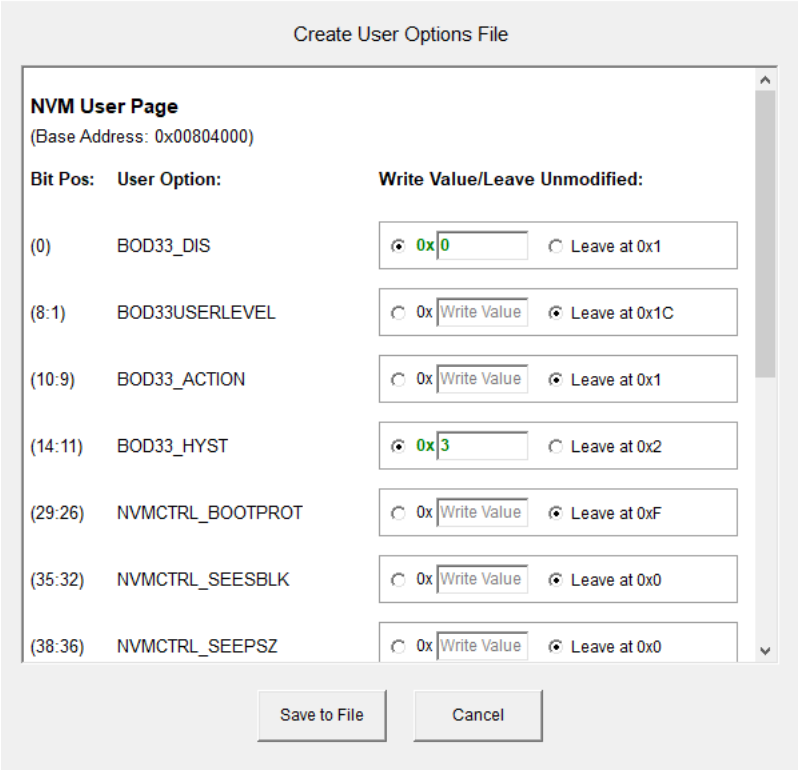


Figure 5-11: Create User Options File Window

Once all new values have been written in their appropriate fields, the developer can save them to a new user options (.OPT) file.

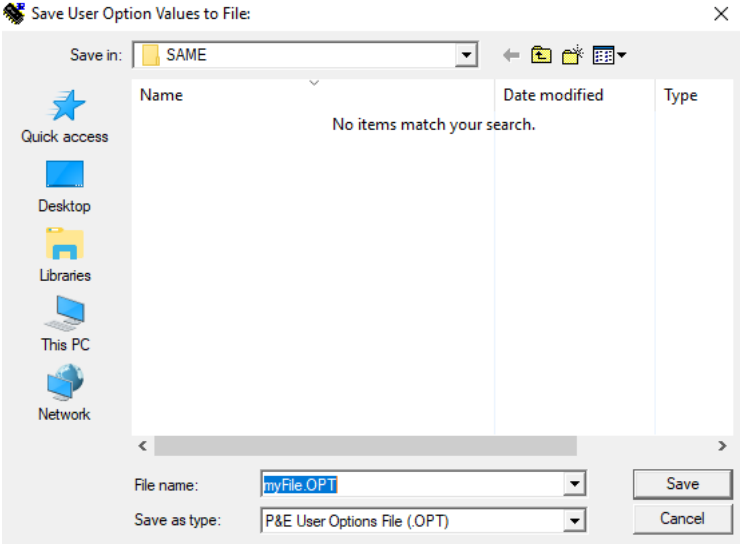


Figure 5-12: Save User Options File Dialog

b. If "Modify File" is selected, the user can choose an existing .OPT file to modify.

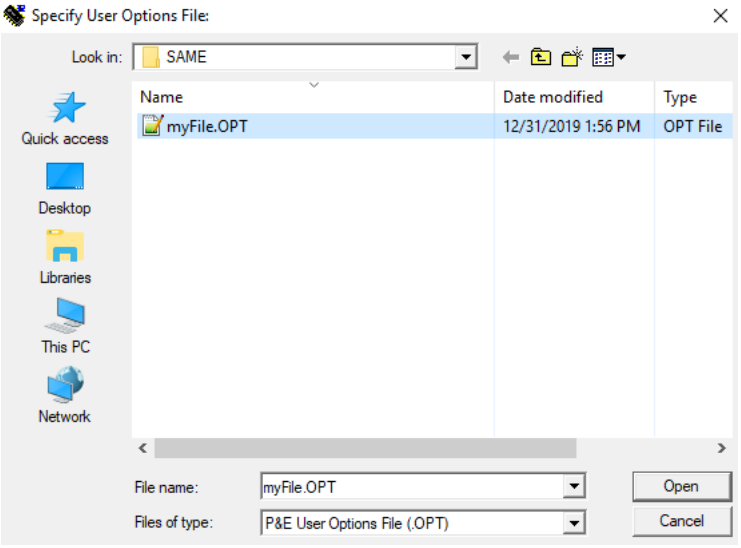


Figure 5-13: Open User Options File Dialog

The values specified in that file will be displayed in their appropriate fields.

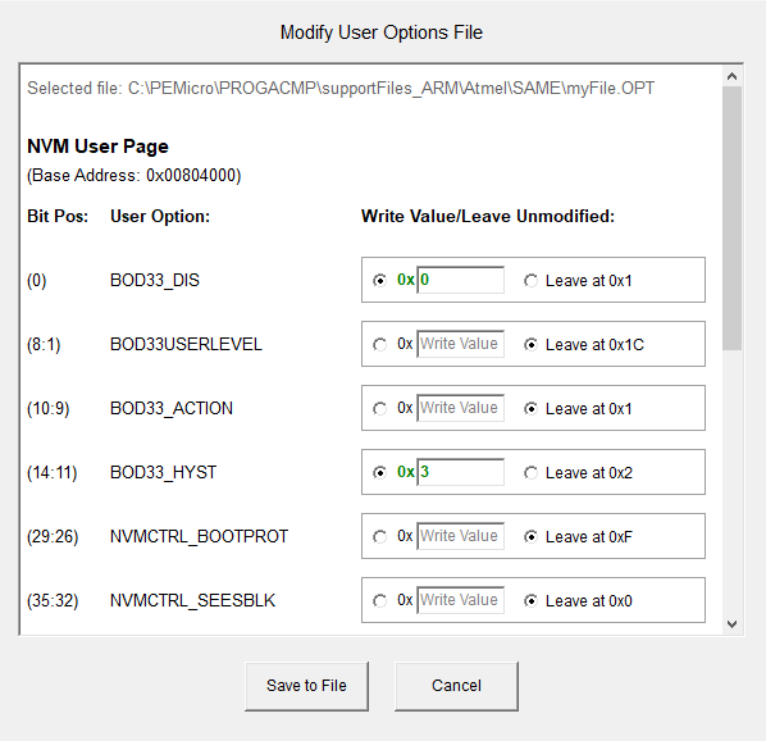


Figure 5-14: Modify User Options File Window

At this point, the developer can choose to modify any contents of the file. The developer can either overwrite the selected file, or save the modified contents to a new filename, leaving the selected file as it was before.

Once a user options (.OPT) file has been created/saved, it can be used to program the device's user options by using the Specify User Options File (SU) and Program User Options (PU) commands.

5.4.2 Specify User Options File (SU)

With the Specify User Options File (SU) command, the developer can select an existing user options (.OPT) file. In interactive PROGACMP, the "User Options File" path will show up in the Configuration window.

5.4.3 Program User Options (PU)

Once a file has been specified, the Program User Options (PU) command can be used to write the values specified by the file. For most devices, new option values won't take effect until the device is reset.

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.

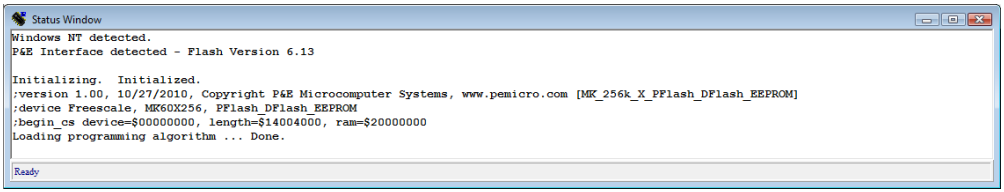


Figure 6-1: Status Window

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

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 CPROGACMP, which is included with the PROGACMP software. When you run the CPROG32Z.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 CPROGACMP scripted programmer, please see the **CPROGACMP User Guide**.

8 HARDWARE INTERFACES

PEmicro's Multilink & Multilink FX debug probes and Cyclone LC & Cyclone FX production programmers are compatible hardware interfaces for use with PROGACMP. 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 & 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 8-1: Multilink debug probe

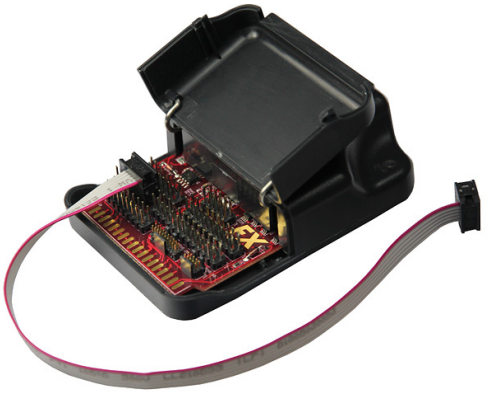


Figure 8-2: Multilink FX debug probe (open for access to headers)

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 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, PROGACMP.

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:	CCG2, CCG3PA, EZ-BLE-PSoC-PRoC, EZ-BLE-PSoC6, FM3, PRoC-BLE, PSoC® 4, PSoC 5, PSoC 6, Traveo II
GigaDevice:	GD32
Infineon:	XMC

Maxim:	DARWIN
Nordic Semi:	nRF51, nRF52
ON Semiconductor:	RSL10
OnBright:	OB90Rxx
Redpine:	WiSeMCU
Silergy (Maxim):	AM0, AM1x, MAX716xx
Silicon Labs:	EFM32, EFR32, SiM3
STMicroelectronics:	Bluetooth, STM32
Texas Instruments:	LM3S, LM4, SimpleLink, TM4C12x
Toshiba:	TX00, TX03, & TX04
WIZnet:	W7500x

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 (with STM8 adapter).

The Cyclone FX Universal also supports Renesas' R8C, H8, H8S/Tiny, M16C, M16C80, M32C, RL78, RX, RX63T, RH850 device families (with Renesas adapter).

Figure 8-3: Cyclone LC Models

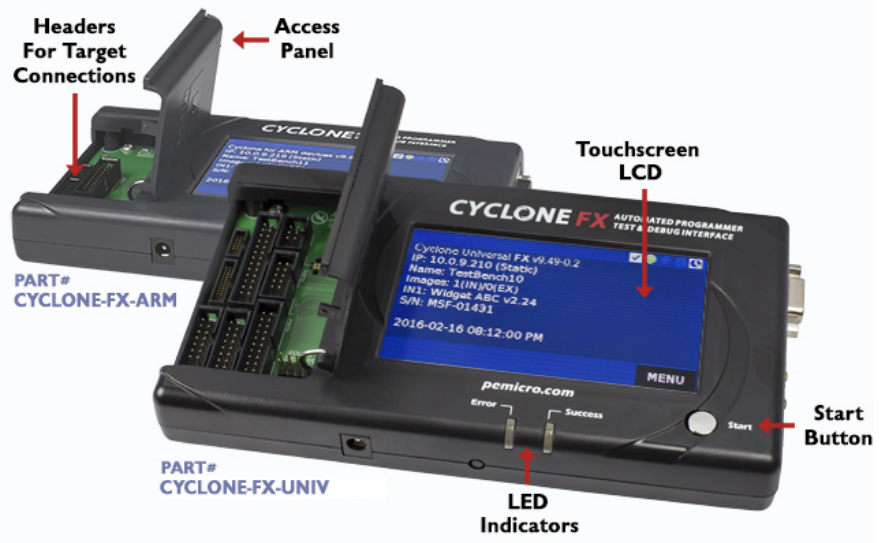


Figure 8-4: Cyclone FX Models

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.

8.3

9 PROGRAMMING UTILITIES

The following no-cost programming utilities are available on PEmicro'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 PEmicro's website at: www.pemicro.com/newsletter/experts_corner/2005_08_17/serialize.cfm.

APPENDIX A - ALGORITHM SETUP COMMANDS

Setup Commands are commands that each appear on separate lines of a .ARP 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 .ARP file or they will be ignored.

The setup commands are:

REQUIRES_PROG_VERSION=x.xx/

Sometimes algorithms will require features to be built into the PEmicro flash programmer itself. If the algorithm requires a minimum version number of the programmer, use this command. The interactive programmer will give the user a warning if the programmer version is not greater than or equal to the version referenced in this command. The commandline programmer will halt with error 14.

NO_ON_CHIP_RAM

This command has 14 characters and tells the programmer not to perform any action to turn on the on chip ram. You must provide RAM to run the calibration routines and load your .ARP file S records. If not deactivated by this command, the on chip RAM is turned on after all other setup commands are executed. On chip RAM is automatically enabled in most processor in order to load the programming algorithm. If your processor has on chip ram and it is turned on automatically, use this command without any writes to chip select. If your processor has no on_chip RAM, use this command and follow it with either WRITE_BYTE, WRITE_WORD or WRITE_LONG in order to turn on chip selects to enable external RAM. The RAM should be turned on at the location where the S records in the .CFP file start.

NO_TIMING_TEST

This command has 14 characters and tells the programmer not to evaluate the target processor speed during 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=llllllll/aaaaaaaa/

This command has 29 characters. It writes the hex long llllllll to the hex address aaaaaaaa in the current space. This command is most often used to enable the chip

selects to allow the CPU to see the flash at address 0. By default the debugger assumes the flash is on CSBOOT. You may also do all sorts of system configuration with these command.

WRITE_WORD=wwww/aaaaaaaa/

This command has 25 characters. It writes the hex word wwww to the hex address aaaaaaaa in the current space. This command is most often used to enable the chip selects to allow the CPU to see the flash at address 0. By default the debugger assumes the flash is on CSBOOT. You may also do all sorts of system configuration with these command.

WRITE_BYTE=bb/aaaaaaaa/

This command has 23 characters. It writes the hex byte bb to the hex address aaaaaaaa in the current space. This command is most often used to enable the chip selects to allow the CPU to see the flash at address 0. By default the debugger assumes the flash is on CSBOOT. You may also do all sorts of system configuration with these command.

BOUNDARY_MASK=mmmmmmmm/

This command has 23 characters. It indicates to the programmer that when buffering data down to the target, the data may not cross certain boundaries. If a value of $FFFFFF80 was used, this would indicate to the programmer that only 128 byte sections may be programming at once (aligned on 128-byte boundaries). This does not mean that the whole 128 bytes need to be programmed, only that the flash programmer will split the data up to be programmed in chunks which never cross a certain boundary. This is very useful for paged memory, or to adhere to block programming requirements of certain motorola flash.

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.

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.

NO_BASE_ADDRESS

or

NO_BASE_ADDRESS=bbbbbbbb/

The 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.

ADDR_RANGE=aaaaaaaa/bbbbbbbb/

Normally the valid flash range is set by the module_length constant in the algorithm which the programmer then uses to decide how to display memory in the code window. If not all memory between module_address and module_address+module_length is valid, this command can be used to override the default functionality and describe to the programmer what is valid memory which should be displayed and changed. Note that these addresses are relative to the base address of the flash. aaaaaaaa is the start address relative to the base address and bbbbbbbb is the end address relative to the base address.

APPENDIX B - ALGORITHM 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.

Erade 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 PROGACMP 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

The address of a routine which turns on the voltages necessary to program/erase the module. This address must be 0 form PROGACMP.

Off Volts Address

The address of a routine which turns off the voltages necessary to program/erase the module. This address must be 0 form PROGACMP.

Enable Address

The address of a routine which sets up and enable the module at startup and after each command is executed. Returning with R2 non zero indicates an error.

Disable Address

The address of a routine which shuts down the module. This address must be 0 form PROGACMP.

Before Read Address

The address of a routine which sets up the module to do a read. R1 contains the

address to be read.

After Read Address

The address of a routine which takes the module out of read mode.

User Function Address

This is an optional user function. It is created with a USER = statement in the .ARP file and a corresponding address as an extra address in the table. On entry, R2 is the module length, R1 is the module address, R4 is the user parameter if any, and R3 is the buffer address. If on return R2<>0 an error occurred.