EMBEDDETECH

QUICKFLASH FOR Microsemi

QF-ACT-400
USER MANUAL

Disclaimer:

The information contained in this User Manual is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that the device operation meets with your specifications. The device and all documentation are provided to you "AS IS" without warranty or support of any kind. Embeddetech makes no representations or commitments of any kind, whether express or implied, written or oral, statutory or otherwise, related to the device operation or information in this User Manual, including but not limited to its condition, quality, performance, merchantability, or fitness for purpose. Embeddetech disclaims all liability arising from this information and its use. The buyer agrees to defend, indemnify and hold harmless Embeddetech from any and all damages, claims, suits, or expenses resulting from such use. Use of any Embeddetech software is subject to Embeddetech's End User License Agreement. Buyer must read, understand, and abide by the End User License Agreement before making use of any Embeddetech device. The latest EULA may be downloaded at www.embeddetech.com/eula/ems_eula.pdf.

No part of this document may be reproduced without the prior written consent of Embeddetech, Inc. While the information contained herein is assumed to be accurate, Embeddetech, Inc. assumes no responsibility for any errors or omissions. In no event shall Embeddetech, Inc., its employees or authors of this document be liable for special, direct, indirect, or consequential damage, losses, costs, charges, claims, demands, claims for lost profits, fees, or expenses of any nature or kind.

Limit of liability: In no event will Embeddetech, Inc. be liable to you for any loss of use, interruption of business, or any direct, indirect, special incidental or consequential damages of any kind (including lost profits) regardless of the form of action whether in contract, tort (including negligence), strict product liability or otherwise, even if Embeddetech, Inc. has been advised of the possibility of such damages.

Embeddetech, Inc.
6415 S 110th E Ave
Tulsa, OK 74133
USA

Web site: www.embeddetech.com
E-mail: info@embeddetech.com
Phone: (918) 813-4941

Table of Contents

QuickFlash For Microsemi Programming Basics

Device Overview

The Embeddetech QuickFlash Programmer is an easy-to-use, stand-alone programming tool for Microsemi FPGAs. QuickFlash is compatible with the original 26-pin and 10-pin FlashPro connectors and supports in-system JTAG-based programming for all of the ProASICPLUS (APA), ProASIC-3 (A3P), IGLOO (AGL), Fusion (AFS), and SmartFusion (A2F) device families.

The QuickFlash programmer is designed to be a stand alone programmer that is easily configured to perform a user-defined set of programming operations with the press of a button. The target images are loaded into the programmer during configuration, so once they are configured they do not require a supporting computer to perform programming and/or verification. The compact, portable programmer can then easily be sent to the field for in-field system updating, with minimal training needed for field applications support. Field service technicians can simply connect the programmer to the target system's JTAG chain, power the programmer, and push "PROGRAM", and all programming operations are executed and validated.

Since the programmer requires no PC to operate, Embeddetech's QuickFlash programmers are ideal low-cost solutions for both field updates and production programming. All production programming processes are then executed with the press of a button, simplifying and accelerating production programming time.

USB Ports

QuickFlash Programmers support both USB Host and USB device interfaces for loading images onto the programmer and configuring the programming process. The DC power port, the USB Host port, and the USB Device port are all on the left side of the programmer, as shown below.

User Interface

The top of the programmer has four indicator LEDs and the "PROGRAM" button, which initiates the user configured programming operations. The indicator LEDs are as follows:

POWER – Indicates that the programmer is powered. If this indicator is not lit, then either the device is not powered properly, or an incorrect electrical wiring has caused the internal voltage regulation to shut down.

STATUS – Indicates the status of the programmer. This is a three-color RGB LED. Blue indicates that the programmer is currently busy executing an operation. Red indicates that the previously executed operation failed. Green indicates that the previously executed operation was successful.

USB DEVICE – Indicates that connection to a PC has been detected and the programmer is operating in USB device mode.

USB HOST – Indicates that a thumb drive has been detected and the programmer is operating in USB host mode.

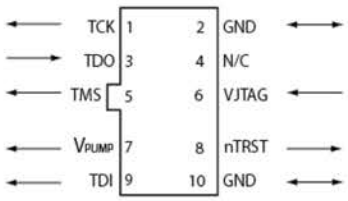
The PROGRAM button and indicating LEDs on the top of the programmer are shown below.

A3P Programming Port

The standard A3P programming port for connecting to target boards is on the right side of the programmer. The connector interface matches the standard 10-pin connector JTAG interface used in the FlashPro3 programmer. This is called the "A3P" programming port, and does not support ProASICPLUS device programming. This programming port is available for both the QuickFlash A3P Programmer and the QuickFlash A3PPLUS Programmer. The information for this connector interface can be found in the FlashPro User's Guide, starting on page 218, below.
http://www.actel.com/documents/flashpro_ug.pdf

The A3P programming port output is supplied via a connector to which a detachable 10-pin cable is fitted. The connector on the A3P programming port is a 2x5, right angle male header connector, which is manufactured by AMP and has a manufacturer's part number of 103310-1. This is a standard 2x5, 0.1 pitch connector which is keyed. The target board should use AMP P/N 103310-1 (DigiKey P/N A26285-ND) for the 10pin right-angle header, or AMP P/N 103308-1 (DigiKey P/N A26267-ND) for the straight version.

The signals on the pins of the QuickFlash A3P programming connector are shown in the figure below (extracted from the FlashPro3 product specification):



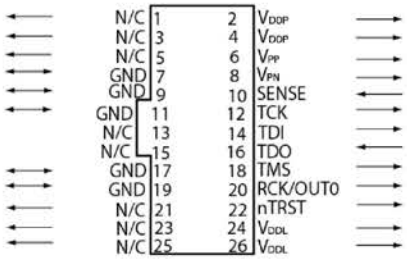
Note: All ground pins must be connected. The rectangular shape shows connections on the programmer itself. Arrows show current flow towards or from the A3P port of the programmer. The list below provides a description of the signals.

VPUMP:	3.3V Programming Voltage
GND:	Signal reference
TCK:	JTAG clock
TDI:	JTAG data input to device
TDO:	JTAG data output from device
TMS:	JTAG mode select
nTRST:	Programmable output pin may be set to off, toggle, low, or high level
VJTAG:	Reference voltage from the target board
N/C:	Programmer does not connect to this pin

Some designers of high-integrity boards (military and avionic) may arrange their boards so that TRST is tied to ground via a weak pull-down resistor. The purpose of this is to hold the JTAG state machine in a reset state by default, so that even with TCK oscillating, some sudden ion bombardment or other electrical event will not suddenly throw the JTAG state machine into an unknown state. If your design also uses a weak pull-down resistor on TRST on your board, then enabling the "Drive TRST" flag will be required to force the JTAG state machine out of reset to permit programming to take place. With most boards, there is no need to select this flag.

APA Programming Port

The APA programming port for connecting to ProASICPLUS target boards is the 26-pin port. This port is only available in the QuickFlash Plus The connector interface matches the 26-pin connector JTAG interface used in the FlashPro Lite programmer. The 26-pin connector is shown in the figure below, as shown on page 219 of the FlashPro User's Guide.



Note: All ground pins must be connected. The rectangular shape shows connections on the programmer itself. Arrows show current flow towards or from the rectangular programmer. The appropriate connector target cable for this is Samtec FFSD-13-D-12.00-01-N. This cable is not included with the QuickFlash A3PPLUS Programmer, but can be purchased separately.

A description of the connector signals is given below.

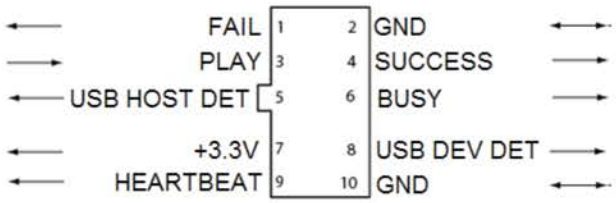
VDDP:	VDD supply logic for I/O pads (Driven by target by default. Use VDDP command to instruct the programmer to drive this rail with the required 2.5V or 3.3V voltage)
VDDL:	VDD supply for core (Defaults to off. Use VDDL ON command to set to 2.5V)
VPP:	Positive programming supply (+16.5V)
VPN:	Negative programming supply (-13.8V)
GND:	Signal reference
SENSE:	Input from target board to programmer to indicate connection to ground
TCK:	JTAG clock
TDI:	JTAG data input to device
TDO:	JTAG data output from device
TMS:	JTAG mode select
nTRST:	Programmable output pin may be set to off, toggle, low, or high level
RCK/OUT0:	Programmable output pin may be set to off, toggle, low, or high level
N/C:	Programmer does not connect to this pin

Some designers of high-integrity boards (military and avionic) may arrange their boards so that TRST is tied to ground via a weak pull-down resistor. The purpose of this is to hold the JTAG state machine in a reset state by default, so that even with TCK oscillating, some sudden ion bombardment or other electrical event will not suddenly throw the JTAG state machine into an unknown state. If your design also uses a weak pull-down resistor on TRST on your board, then enabling the "Drive TRST" flag will be required to force the JTAG state machine out of reset to permit programming to take place. With most boards, there is no need to select this flag.

Control I/O Port

The control I/O port allows remote access to the 7 status LEDs and PROGRAM button user interface. For production programming applications, the QuickFlash Programmer can be mounted in one location near the target board, and external status LEDs and/or an external PROGRAM button can be wired into the control I/O port to control programming and monitor the results. Or the programming control can be controlled by an automated system that uses the control I/O to initiate the programming operation and monitor the BUSY, PASS, and FAIL status. The active-high voltage level for the status outputs is +3.3V. The PROGRAM pin is pulled high internally in the programmer, and can be shorted to ground using a normally open momentary switch to initiate the programming operation remotely. Pin 7 provides a +3.3V power rail as may be needed.

The signals on the pins of the Control I/O connector are shown in the figure below.



The right side of the QuickFlash Programmer is shown below, with the ports labeled. The ProASICPLUS port is only available with the QuickFlash A3PPLUS Programmer.

Programmer Configuration

QuickFlash Configuration Wizard

The QuickFlash programmer must be configured to program the required devices with the required design files. Once configured, all programming operations are run sequentially with the press of the "PROGRAM" button. The programmer is configured by processing command line scripts, however the QuickFlash Configuration Wizard is recommended to be used for configuring the programmer. Using the configuration wizard, no knowledge of the command line commands are required.

The configuration wizard can be downloaded from the QuickFlash Programmer's product web page, located at [insert link]. After the wizard is downloaded and installed, connect the QuickFlash Programmer to your PC using the USB cable. Note: Do not power the programmer with the DC power supply when using the programmer as a USB device. The power supply should only be used when the programmer is being used in stand-alone mode or when configuring the programmer with a thumb drive. After connecting the programmer to the PC, use the driver located in the wizard's installation directory under the "Drivers" folder to install the programmer if it has not been installed yet.

Run the wizard by clicking All Programs -> Embeddetech -> QuickFlash Configuration Wizard from your start panel. Click on the "Create New Configuration" text and respond to the sequence of questions prompted by the wizard. When the configuration has been set up, review the settings to verify that everything is correct, then click the "Configure Programmer" button to configure the programmer. You can also create a thumb drive password key (if the programmer's configuration locks the programmer), or create a thumb drive configuration image to configure the programmer using a thumb drive. Save the configuration file for later use.

The wizard also has a built-in command line console that can be used to enter commands manually and troubleshoot configuration problems, such as not properly powering the JTAG bus before executing a programming operation. Details of the command line interface to the programmer are provided below.

Command Line Overview

The QuickFlash Programmer may be used as either a USB device or USB host. Both USB connectors are the same physical USB bus, and so both modes can not be used simultaneously. When the Programmer is connected to a PC and used as a USB device, the Programmer is powered by the USB bus, and the DC power supply should not be used. When the Programmer is used as a USB host for use with a thumb drive, or when the Programmer is used in stand-alone mode, the Programmer must be powered with the external power supply. The Programmer detects whether the external power supply is connected to determine which mode its USB bus needs to operate.

When the Programmer is connected to a PC as a USB device, it enumerates as a serial port. This serial port can be opened with any terminal application at any baud rate. The Programmer uses the serial port to present a command line interface. Individual commands are entered into the command line, and the result is printed to the terminal for the user to view. All configuration of the Programmer is accomplished with this command line interface.

When the Programmer is used as a USB host and a thumb drive is installed, the Programmer looks for a "script.txt" file in the root directory which contains all of the commands to be run. The Programmer reads the script file one line at a time and sends it to the command line parser, waits for the command line processing to be finished, and then reads the next line, until the entire file's script has been run. All commands sent to the command line and all results printed by the command line indicating the result of the operation are logged in a "report.txt" text file in the root directory of the thumb drive. When a thumb drive is inserted and the script file automatically runs and is completed, the thumb drive can be removed and the report file can be reviewed to evaluate the results of the operations.

Either the USB device mode with a terminal program or the USB host mode with a thumb drive may be used to configure the Programmer. Copying files over from the PC to the Programmer in USB device mode is done using the XMODEM protocol, and is a little more procedurally complex and slower than simply putting files on the thumb drive and instructing the Programmer to copy the files into local flash. So the thumb drive is generally used for copying the target image files over and setting up the script to be run when the PROGRAM button is pressed. However once the files are copied over to the Programmer, if the programming operations need to be debugged, connecting the Programmer to a PC and manually entering one command at a time can be easier.

In any case, regardless of whether the USB host (thumb drive) or USB device (serial port) method are used, they are essentially the same method of configuring the Programmer, in that they both essentially work with the same console interface.

USB Device Driver

When connecting the Programmer to a PC as a USB device, the PC requires an INF file to instruct the Hardware Installation Wizard on which driver to use. This INF file can be found on the website at http://embeddetech.com/download/quickflash-usb-device-driver/

Internal Flash Memory

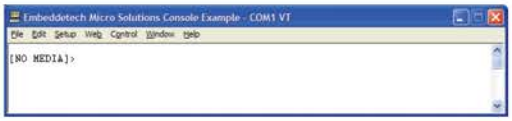
The Programmer has 64MBytes of internal Flash memory available for storing target image files and the stored script file that is played back when the RUN button is pressed. 512KB of this memory is reserved for storing the recorded script file and other Programmer settings, and the rest of the memory is managed as a user-defined number of memory partitions, each partition being able to store one file. The user only needs to specify the number of partitions the memory is to be divided into, and the partitions are resized correspondingly.

Thus if the user specifies 5 partitions as being needed, the Programmer divides the flash memory into 5 partitions, each one containing 12.7MBytes. Files up to 12.7MBytes in size could then be copied into the partitions.

When ProASICPLUS devices are being programmed with the QuickFlash A3PPLUS Programmer, the target image file that is copied into the Programmer is a STAPL file. These files store a significant amount of data in a compressed format, and the Programmer must uncompress the data to program the devices. For STAPL files, the partition size should be three times as large as the STAPL file. The extra memory is used to store the decompressed data the first time the programming operation is run on that STAPL file. If a STAPL file is 128KB, the partition size should be at least 384KB. This pre-decompression of the STAPL file significantly reduces the programming operation time when programming APA devices. This is a significant value when the Programmer is used in a production process. The first ProASICPLUS device programming operation will be noticeably slower the first time it is run, however after the first time when the Programmer has decompressed the compressed data into Flash memory, the programming operation runs much faster. This is a key advantage to using the QuickFlash A3PPLUS Programmer for production applications.

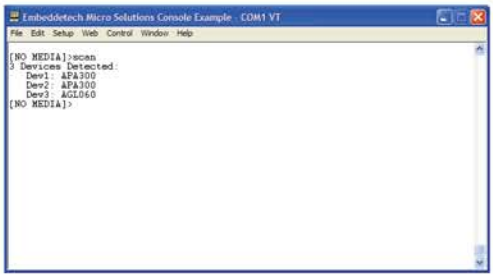
Console

When the Programmer is connected to a PC's terminal program, such as TeraTerm (as shown below) or HyperTerminal, pressing Enter reprints the command prompt. In the example below, the prompt is "[NO MEDIA]>", because no media will be connected when in serial port mode.



If a thumb drive is connected, the name of the thumb drive will show up in the prompt upon insertion, and the Programmer will look for the "script.txt" file and print a warning if one is not found, as shown below. This command line information would be printed to the "Report.txt" file on the thumb drive.

Commands are entered into the console prompt and executed when ENTER is pressed (<CR><LF> is received). In the example below, the "scan" command is entered, and as it is entered the characters are echoed back. After the command is entered the JTAG chain is scanned and the connected devices are printed out, then the command prompt reappears.



Previously entered commands can be scrolled through by pressing up (previous) or down (next). The documentation of the supported commands will be in the format below, with the text in blue indicating the command entered by the user, and the black text indicating console prompt and result text. For the above example, the format to show the example command is as follows:

```
[NO MEDIA]>scan
3 Devices Detected:
        Dev1: APA300
        Dev2: APA300
        Dev3: AGL060
[NO MEDIA]>
```

Command Syntax Format

In the command descriptions that follow, the following command syntax format is used:

- Parameters with "these | bars | between | them" indicate a list of available parameters.
- Parameters or lists of parameters in [brackets] are optional and can be left blank.
- ALL CAPS indicates a literal parameter.
- A 'D' in parentheses (D) after a parameter in a parameter list specifies the default parameter if none is specified.
- <image> indicates either a USB drive filename or a Flash partition index. For a memory configuration of N partitions, the valid partition indices range from 0 to N-1.
- <device#> indicates the device number to be used, if more than 1 device is connected to the JTAG chain. For a JTAG chain with N devices, the device number ranges from 1 to N.
- Filenames must be in an 8.3 format. The maximum filename size is XXXXXXXX.XXX.
- Processing of all command lines is case-insensitive.

Since the programmer requires no PC to operate, Embeddetech's QuickFlash programmers are ideal low-cost solutions for production programming. All production programming processes are then executed with the press of a button, simplifying and accelerating production programming time.

DirectC Commands

SCAN

Command Syntax:

SCAN

This command causes the Programmer to scan the JTAG chain and report the devices detected.

Example:

```
[NO MEDIA]>scan
3 Devices Detected:
        Dev1: APA300
        Dev2: APA300
        Dev3: AGL060
[NO MEDIA]>
```

DEVICE

Command Syntax:

DEVICE <device#>

This command specifies which device on the chain to use for subsequent operations. If only one device is connected to the JTAG chain, "device 1" is assumed and the device command is not needed. If there are multiple devices on the chain, the DEVICE command must be used prior to any operations that are performed on a specific device.

This example sets the Programmer to Device 1 for subsequent operations:

Example:

```
[NO MEDIA]>device 1
Device Selection Successful!
[NO MEDIA]>
```

IRLEN

Command Syntax:

IRLEN <device#> <LEN>

This command specifies the length of the JTAG instruction register for a particular device. For JTAG busses with all Actel-only devices, the instruction length detection is automatic, and this instruction is not needed.

This example sets the JTAG instruction register length of device #2 to 8 bits:

Example:

```
[NO MEDIA]>irlen 2 8
IRLEN Update Successful
[NO MEDIA]>
```

READ DEVICE INFO

Command Syntax:

READ DEVICE INFO [USING <image>]

This command reads the FPGA device information either using the specified image, or the previously-specified image, if one was previously specified. If no image is specified in the command and no image was previously specified since the Programmer powered up, an error message will be generated. The image can be resident either in an attached USB thumb drive, or in an internal Flash memory partition.

NOTES:

- This command requires a device to be previously specified if multiple devices are on the chain. If only one device is detected on the chain, that device will be used.
- The device type (APA or non-APA) must match the file type (.stp or .dat, respectively) specified.

This example uses the previously specified image:

```
[NO MEDIA]>device 2
Device Selection Successful!
[NO MEDIA]>read device info
ActID = CF ExpID = CF

Initializing Target Device...
Setting BSR Configurations...
Reading Security...
SILSIG: 00 00 00 00
ACTEL_SLOG_UROW = 00 00 00 40 87 FD A5 6F B0 19 0A E6 20 02 F1 F5
User information:
CYCLE COUNT: 8
CHECKSUM = F1 F5
Design Name = 87 FD A5 6F B0 19 0A E6 20
================================================

FlashROM Information:
00 FF FF 00 FF 00 00 FF FF 00 00 FF 00 FF FF 00
FF 00 00 FF 00 FF FF 00 00 FF FF 00 FF 00 00 FF
FF 00 00 FF 00 FF FF 00 00 FF FF 00 FF 00 00 FF
00 FF FF 00 FF 00 00 FF FF 00 00 FF 00 FF FF 00
FF 00 00 FF 00 FF FF 00 00 FF FF 00 FF 00 00 FF
00 FF FF 00 FF 00 00 FF FF 00 00 FF 00 FF FF 00
00 FF FF 00 FF 00 00 FF FF 00 00 FF 00 FF FF 00
FF 00 00 FF 00 FF FF 00 00 FF FF 00 FF 00 00 FF
================================================
FSN: 0A 88 10 0D 14 ED 00 EA 63 F7 00 00 54 BB 02 FF
Security Setting :
================================================
[NO MEDIA]>
```

This example uses device #2 and Flash partition #0:

```
[NO MEDIA]>device 2
Device Selection Successful!
[NO MEDIA]>read device info using 0
Now using partition 0
ActID = CF ExpID = CF

Initializing Target Device...
Setting BSR Configurations...
Reading Security...
SILSIG: 00 00 00 00
ACTEL_SLOG_UROW = 00 00 00 40 87 FD A5 6F B0 19 0A E6 20 02 F1 F5
User information:
CYCLE COUNT: 8
CHECKSUM = F1 F5
Design Name = 87 FD A5 6F B0 19 0A E6 20
================================================

FlashROM Information:
00 FF FF 00 FF 00 00 FF FF 00 00 FF 00 FF FF 00
FF 00 00 FF 00 FF FF 00 00 FF FF 00 FF 00 00 FF
FF 00 00 FF 00 FF FF 00 00 FF FF 00 FF 00 00 FF
00 FF FF 00 FF 00 00 FF FF 00 00 FF 00 FF FF 00
FF 00 00 FF 00 FF FF 00 00 FF FF 00 FF 00 00 FF
00 FF FF 00 FF 00 00 FF FF 00 00 FF 00 FF FF 00
00 FF FF 00 FF 00 00 FF FF 00 00 FF 00 FF FF 00
FF 00 00 FF 00 FF FF 00 00 FF FF 00 FF 00 00 FF
================================================
FSN: 0A 88 10 0D 14 ED 00 EA 63 F7 00 00 54 BB 02 FF
Security Setting :
================================================
[NO MEDIA]>
```

This example uses the file image2.dat in the attached USB thumb drive:

```
A3P250      :\>read device info using a3p250.dat
Now using a3p250.dat

ActID = 0x32A141CF ExpID = 0x03A141CF

Initializing Target Device...
Setting BSR Configurations...
Reading Security...
SILSIG: 00 00 00 00
ACTEL_SLOG_UROW = FF FF FF FF 00 00 00 00 00 00 00 00 C0 06 00 00
User information:
CYCLE COUNT: 27
CHECKSUM = 00 00
Design Name = 00 00 00 00 00 00 00 00 C0
================================================

FlashROM Information:
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
================================================
FSN: 95 68 60 28 00 DF 80 38 A6 FF 00 30 4F DB 02 FF
Security Setting :
================================================
A3P250      :\>
```

READ DEVICE ID

Command Syntax:

```
READ DEVICE ID [ USING <image> ]
```

This command reads the FPGA device ID either using the specified image, or the previously-specified image, if one was previously specified. If no image is specified and no image was previously specified, an error message will be generated.

NOTES:

- This command requires a device to be previously specified if multiple devices are on the chain. If only one device is detected on the chain, that device will be used.
- The device type (APA or non-APA) must match the file type (.stp or .dat, respectively) specified.

Example Usages:

This example uses the previously specified image:

```
[NO MEDIA]>using 0
Now using partition 0
[NO MEDIA]>read device id

ActID = 0x32A141CF ExpID = 0x03A141CF
Initializing Target Device...
Setting BSR Configurations...
Reading Security...
IDCODE: 0x32A141CF
[NO MEDIA]>
```

This example uses the file a3p250.dat in the USB thumb drive:

```
A3P250      :\>read device id using a3p250.dat
Now using a3p250.dat

ActID = 0x32A141CF ExpID = 0x03A141CF
Initializing Target Device...
Setting BSR Configurations...
Reading Security...
IDCODE: 0x32A141CF
A3P250      :\>
```

This example uses Flash partition #0:

```
[NO MEDIA]>read device id using 0
Now using partition 0

ActID = 0x32A141CF ExpID = 0x03A141CF
Initializing Target Device...
Setting BSR Configurations...
Reading Security...
IDCODE: 0x32A141CF
[NO MEDIA]>
```

ERASE

Command Syntax:

ERASE [USING <image>]

This command erases the specified FPGA device, either using the specified image, or the previously-specified image, if one was previously specified. If no image is specified and no image was previously specified, an error message will be generated.

NOTES:

- This command requires a device to be previously specified if multiple devices are on the chain. If only one device is detected on the chain, that device will be used.
- The device type (APA or non-APA) must match the file type (.stp or .dat, respectively) specified.

Example Usages:

This example uses the previously specified image:

```
[NO MEDIA]>using 0
Now using partition 0
[NO MEDIA]>erase

ActID = 0x32A141CF ExpID = 0x03A141CF
Initializing Target Device...
Setting BSR Configurations...
Reading Security...
Erase...
ACTEL_SLOG_UROW = FF FF FF FF FF FF FF FF FF FF FF FF FF 08 FF FF
Programming UROW...
Erase Operation Successful!
[NO MEDIA]>
```

This example erases the FPGA using Flash partition #2:

```
[NO MEDIA]>device 1
Device Selection Successful!
[NO MEDIA]>erase using 2
Now using partition 2
Checking CRC value...
  Actual CRC = 0x00002BB8
alg version in file = 23
DirectC version = 22
Reading STAPL file...
SERIAL# = 00065294322300

Erase Operation Successful!
[NO MEDIA]>
```

This example uses the file apa_51.stp in the USB thumb drive to erase the FPGA:

```
A3P250      :\>device 1
Device Selection Successful!
A3P250      :\>erase using apa_51.stp
Now using apa_51.stp
Checking CRC value...
  Actual CRC = 0x00002BB8
alg version in file = 23
DirectC version = 22
Reading STAPL file...
SERIAL # = 00065294322300

Erase Operation Successful!
A3P250      :\>
```

PROGRAM

Command Syntax:

PROGRAM [USING <image>]

This command programs the FPGA device, either using the image specified in the command, or the previously-specified image, if one was previously specified. If no image is specified and no image was previously specified, an error message will be generated.

NOTES:

- This command requires a device to be previously specified if multiple devices are on the chain. If only one device is detected on the chain, that device will be used.
- The device type (APA or non-APA) must match the file type (.stp or .dat, respectively) specified.

Example Usages:

This example uses the previously specified image:

```
[NO MEDIA]>using 0
Now using partition 0
[NO MEDIA]>program

ActID = 0x32A141CF ExpID = 0x03A141CF
Initializing Target Device...
Setting BSR Configurations...
Reading Security...
Erase...
ACTEL_SLOG_UROW = FF FF FF FF FF FF FF FF FF FF FF FF FF FF 08 FF FF
Programming UROW...
Programming FPGA Array.........................
Verifying FPGA Array.........................
Programming RLOCK...
Programming SILSIG...
Programming Operation Successful!
[NO MEDIA]>
```

This example programs the FPGA using Flash partition #2:

```
[NO MEDIA]>device 1
Device Selection Successful!
[NO MEDIA]>program using 2
Now using partition 2
Checking CRC value...
  Actual CRC = 0x00002BB8
alg version in file = 23
DirectC version = 22
Reading STAPL file...
SERIAL # = 00065294322300
PROGRAMMING ARRAY
Programming 100% Complete
VERIFYING PROGRAMMED BITS...
Verification 100% Complete
VERIFYING ERASED BITS...
Verification 100% Complete
Verifying UROW...

Programming Operation Successful!
 [NO MEDIA]>
```

This example programs the FPGA using the file image1.dat located in the thumb drive:

```
A3P250      :\>device 2
Device Selection Successful!
A3P250      :\>program using image1.dat
Now using image1.dat

ActID = 0x1B9121CF ExpID = 0x039121CF

Initializing Target Device...
Setting BSR Configurations...
Erase...
ACTEL_SLOG_UROW = FF FF FF FF 87 FD A5 6F B0 19 0A E6 60 02 F1 F5
Programming UROW...
Programming FPGA Array.........................
Verifying FPGA Array.........................
Programming RLOCK...
Programming SILSIG...
Programming Operation Successful!
A3P250      :\>
```

VERIFY

Command Syntax:

VERIFY [USING <image>]

This command verifies the specified portions of the FPGA device, either using the specified image, or the previously-specified image, if one was previously specified. If no image is specified and no image was previously specified, an error message will be generated.

NOTES:

- This command requires a device to be previously specified if multiple devices are on the chain. If only one device is detected on the chain, that device will be used.
- The device type (APA or non-APA) must match the file type (.stp or .dat, respectively) specified.

Example Usages:

This example uses the previously specified image to verify the FPGA:

```
[NO MEDIA]>using 0
Now using partition 0
[NO MEDIA]>verify

ActID = 0x32A141C ExpID = 0x03A141CF
Initializing Target Device...
Setting BSR Configurations...
Reading Security...
Verifying FPGA Array.........................
Verify Successful!
[NO MEDIA]>
```

This example verifies the FPGA using Flash partition #2:

```
A3P060      :\>device 1
Device Selection Successful!
A3P060      :\>verify using 2
Now using partition 2
Checking CRC value...
  Actual CRC = 0x00002BB8
alg version in file = 23
DirectC version = 22
Reading STAPL file...
SERIAL # = 00065294322300
VERIFYING PROGRAMMED BITS...
Verification 100% Complete
Verifying UROW...
Read inhibit: 0
Write inhibit: 0
Permanent Lock: 0

Verify Successful!
A3P060      :\>
```

This example verifies the FPGA using the file image1.dat in the USB thumb drive:

```
A3P060      :\>device 2
Device Selection Successful!
A3P060      :\>verify using image1.dat
Now using image1.dat

ActID = 0x1B9121CF ExpID = 0x039121CF

Initializing Target Device...
Setting BSR Configurations...
Reading Security...
Verifying FPGA Array.........................
Verify Successful!
A3P060      :\>
```

Programmer Commands

LOCK / UNLOCK / REFORMAT

Command Syntax:

LOCK <password>
UNLOCK <password>
REFORMAT

The LOCK command locks the programmer with the specified password, so that the PROGRAM button will only operate after the corresponding "UNLOCK <password>" command has been entered after power-up. This helps ensure the security of the stored images when the Programmer is being shipped. The lock command is optional, and if no lock password is specified, then the PROGRAM button will operate and the Programmer will function normally.

If the LOCK command has been specified, then each time the Programmer powers up, it must receive the corresponding UNLOCK command. In this case either the user of the Programmer must enter the UNLOCK command from the serial port terminal program after connecting it as a device, or it must attach a thumb drive with the "UNLOCK <password>" command stored in the script.txt file. The thumb drive then acts as a key that must be attached and run each time the Programmer powers up in order for the Programmer to operate.

Example Usages:

```
[NO MEDIA]>LOCK OPEN_SESAME
The Programmer is now locked.
[NO MEDIA]>repartition 32
The Programmer must be unlocked before instructions may be executed.
Either enter the unlock password or enter RESTORE to delete all
Stored information and unlock the Programmer.
[NO MEDIA]>UNLOCK WILD_GUESS
Invalid Password.
The Programmer must be unlocked before instructions may be executed.
Either enter the unlock password or enter RESTORE to delete all
Stored information and unlock the Programmer.
[NO MEDIA]>UNLOCK OPEN_SESAME
The Programmer is unlocked.
[NO MEDIA]>REFORMAT
Programmer memory cleared.
The Programmer is unlocked.
```

RECORD

Command Syntax:

```
RECORD START
RECORD STOP
```

The RECORD START command instructs the programmer to delete the previously recorded script and begin recording all subsequent command line entries. These subsequent command line entries will be played back when the PROGRAM button is pressed.

The RECORD STOP command instructs the programmer to stop recording command line entries into the recorded script file and validate the script file for playback when the PROGRAM button is pressed. If no valid script is recorded then the PROGRAM button will not do anything.

Example Usages:

This example script configures the programmer to program two devices on a chain. It begins by repartitioning the Flash memory, then copying two files from the thumb drive into the Flash memory. It then begins script recording, executes the program operations for both images stored natively in Programmer Flash memory, ends the script recording, and locks the Programmer.

```
A3P250      :\>repartition 2
Repartitioning Flash... Flash Operation Successful!
A3P250      :\>copy image1.dat 0
Loading file image1.dat into partition #0... Flash Operation Successful!
A3P250      :\>copy image2.stp 1
Loading file image2.stp into partition #1... Flash Operation Successful!
A3P250      :\>record start
Programmer Script Recording Started.
A3P250      :\>device 1
Device Selection Successful!
A3P250      :\>program using 0
[...Program output]
Programming Operation Successful!
A3P250      :\>device 2
Device Selection Successful!
A3P250      :\>program using 1
[...Program output]
Programming Operation Successful!
A3P250      :\>record stop
Programmer Script Recording Stopped.
A3P250      :\>lock OPEN_SESAME
The Programmer is now locked.
```

Once this script file is run, the Programmer is now configured to program both devices on the JTAG chain with a press of the PROGRAM button.

NOTE: The RECORD command itself can not be recorded to the Programmer's recorded script file.

PLAY

Command Syntax:

PLAY

The PLAY command initiates playback of the recorded script file. Executing this command in the console is equivalent to pressing the physical PROGRAM button. The PLAY command can not be recorded to the flash script file, and so it can not be executed when recording commands to the script file. The recorded script file playback is aborted when any command fails.

NOTE: The PLAY command can not be executed when a script is being recorded, because a PLAY command in the Programmer's script would cause the script to be played recursively when the PROGRAM button is pressed.

Example Usage:

```
[NO MEDIA]>PLAY
[Programmer plays back recorded commands to the console]
[NO MEDIA]>
```

RUN

Command Syntax:

RUN <filename>

The RUN command runs the specified thumb drive filename as a script, executing each command in the file sequentially until the end of the file is reached or an error has occurred.

Example Usage:

```
[NO MEDIA]>RUN myscript.txt
[Programmer plays back the commands recorded in myscript.txt to the
console]
[NO MEDIA]>
```

VDDL

Command Syntax:

VDDL <ON | OFF>

The VDDL command controls whether or not the Programmer applies 2.5V to pin 24 and 26 of the APA programming connector. VDDL is the core voltage for APA device programming. VDDL defaults to OFF when the Programmer powers up. If the target board requires the Programmer to provide +2.5V to VDDL, the user must use the VDDL command before executing program operations.

View pages 219-221 of the Flash Pro User's Guide for more information regarding VDDL.

http://www.actel.com/documents/flashpro_ug.pdf

Example Usage:

```
[NO MEDIA]>VDDL ON
VDDL set to +2.5V
[NO MEDIA]>VDDL OFF
VDDL internal drive disabled.
[NO MEDIA]>
```

VPUMP

Command Syntax:

VPUMP <ON | OFF>

The VPUMP command controls whether or not the Programmer applies +3.3V to PUMP (pin 7 of the A3P connector). By default, the Programmer monitors VPUMP before a programming operation and turns it on if no voltage is detected. If a VPUMP command is used, the VPUMP voltage stays at the specified level, whether it is on or off, during Programming operations.

View pages 217-219 of the Flash Pro User's Guide for more information regarding VPUMP.

http://www.actel.com/documents/flashpro_ug.pdf

Example Usage:

```
[NO MEDIA]>VPUMP ON
VPUMP set to +3.3V
[NO MEDIA]>VPUMP OFF
VPUMP internal drive disabled.
[NO MEDIA]>
```

VJTAG / VDDP

Command Syntax:

<VJTAG | VDDP > <OFF | 2.5V | 3.3V>

The VJTAG and VDDP commands do the same thing, as VJTAG (pin 6 of the A3P connector) and VDDP (pins 2 and 4 of the APA connector) are both the programming voltages used to determine the voltage level for the JTAG bus. They are electrically the same net in the Programmer. By default, VJTAG is off when the Programmer powers up, and the Programmer relies on the target providing a programming voltage of +2.5V or +3.3V. The user may use the VJTAG command to instruct the Programmer to drive the VJTAG bus with the specified voltage if necessary. It is recommended that the target board provides the programming voltage to be used, and thus this command is not used.

View the FlashPro User's Guide for more information regarding VJTAG / VDDP.

http://www.actel.com/documents/flashpro_ug.pdf

Example Usage:

```
[NO MEDIA]>VJTAG 2.5V
VJTAG set to +2.5V
[NO MEDIA]>VJTAG 3.3V
VJTAG set to +3.3V
[NO MEDIA]>VJTAG OFF
VJTAG internal drive disabled.
[NO MEDIA]>VDDP 3.3V
VDDP set to +3.3V
[NO MEDIA]>VDDP 2.5V
VDDP set to +2.5V
[NO MEDIA]>VDDP OFF
VDDP internal drive disabled.
[NO MEDIA]>
```

USB File System & Generic Commands

VERSION

Command Syntax:

VERSION

This command prints the Programmer firmware version to the console.

Example Usages:

```
[NO MEDIA]>version
Version 2.0.2
[NO MEDIA]>
```

USING

Command Syntax:

USING <image>

This command specifies which image, either in a Flash partition or a USB thumb drive file, to use for subsequent DirectC operations.

Example Usages:

This example instructs the Programmer to use partition 0 for subsequent DirectC operations:

```
A3P060      :\>using 0
Now using partition 0
A3P060      :\>
```

This example instructs the Programmer to use the file "a3p250.dat" on the thumb drive for subsequent DirectC operations:

```
A3P250      :\>using a3p250.dat
Now using a3p250.dat
A3P250      :\>
```

CD (Change Directory)

Command Syntax:

CD <name>

This command changes the console working directory.

IMPORTANT NOTE: It is recommended that all FPGA operations and file access be done at the root thumb drive directory.

Example Usages:

This example changes the directory to the subfolder "sub":

```
A3P250      :\>cd sub
A3P250      :\SUB>
```

This example changes the directory one level higher:

```
A3P250      :\SUB>cd ..
A3P250      :\>
```

COPY

Command Syntax:

COPY <image1> <image2>

This command copies image1 and stores it to image2, whether image1 or image2 is on the USB thumb drive or in a partition, or vice versa.

Example Usages:

This example copies the USB thumb drive file "a3p250.dat" into Flash partition 0:

```
A3P250      :\>copy a3p250.dat 0
Loading file a3p250.dat into partition #0... Flash Operation Successful!
A3P250      :\>
```

This example copies Flash partition 0 to Flash partition 1:

```
A3P250      :\>copy 0 1
Copying from partition 0 to partition 1... Flash Operation Successful!
A3P250      :\>
```

This example copies Flash partition 0 to the file "copytest.dat" on the thumb drive:

```
A3P250      :\>copy 0 copytest.dat
Copying file...
File successfully copied!
A3P250      :\>
```

DEL (Delete)

Command Syntax:

DEL <image>

This command deletes the specified image from either the USB thumb drive or Flash memory.

Example Usages:

This example deletes the USB thumb drive file "copytest.dat":

```
A3P250      :\>del copytest.dat
File successfully deleted.
A3P250      :\>
```

This example deletes partition 1:

```
A3P250      :\>del 1
Deleting partition #1... Flash Operation Successful!
A3P250      :\>
```

DIR (Directory)

Command Syntax:

DIR

This command prints the entire working directory of both the USB thumb drive (if connected) and Flash memory. The DIR function also prints out the size of total Flash memory.

Example Usages:

In this DIR printout, no USB thumb drive is connected, so instead of a thumb drive directory, "No media present" is shown. In this example, the Flash was formatted into two partitions, and only partition 1 had a file (a3p250.dat) loaded into the partition.

```
[NO MEDIA]>dir
USB disk directory:
No media present.

External Flash Status:
    Total Memory Detected:          67108864  Bytes
    Useable Memory:                 66584576  Bytes
    # Of Partitions: 2

Flash Directory:
 #     Date       Time          Size       Filename
--- ---------- -------- --------------- ------------
 00 2009-07-17 19:02:00          239522 a3p250.dat
 01 XXXX-XX-XX XX:XX:XX                 <Empty>

[NO MEDIA]>
```

The DIR printout example below shows the response when a thumb drive is attached. This thumb drive had the files "report.txt", "a3p250.dat", and the folder "SUB".

```
A3P250      :\>dir
USB disk directory:
   Date       Time          Size       Filename
---------- -------- --------------- ------------
1980-00-00 00:00:00           18335 REPORT.TXT
1980-00-00 00:00:00 <DIR>           SUB
2009-07-17 19:02:00          239522 A3P250.DAT

External Flash Status:
    Total Memory Detected:          67108864  Bytes
    Useable Memory:                 66584576  Bytes
    # Of Partitions: 2

Flash Directory:
 #     Date       Time          Size       Filename
--- ---------- -------- --------------- ------------
 00 2009-07-17 19:02:00          239522 a3p250.dat
 01 XXXX-XX-XX XX:XX:XX                 <Empty>

A3P250      :\>
```

MD (Make Directory)

Command Syntax:

MD <name>

This command creates a new folder in the current USB thumb drive working directory.

Example Usages:

This example creates a new subfolder "sub" under the root directory:

```
A3P250      :\>md sub
 - Directory created
A3P250      :\>
```

RD (Remove Directory)

RD <name>

This command deletes the specified directory folder in the current working directory of the USB thumb drive.

Example Usages:

This example deletes the subfolder "sub" under the root directory:

```
A3P250      :\>rd sub
Directory successfully deleted.
A3P250      :\>
```

REN (Rename)

Command Syntax:

REN <image> <NewName>

This command renames the specified image to "NewName".

Example Usages:

This example renames the file "a3p250.dat" in the root directory to "test.dat":

```
A3P250      :\>ren a3p250.dat test.dat
File successfully renamed.
A3P250      :\>
```

This example renames the file stored in partition 0 from "a3p250.dat" to "test.dat":

```
A3P250      :\>ren 0 test.dat
Renaming Successful!
A3P250      :\>
```

XMODEM

Command Syntax:

XMODEM <partition#> <filename>

This command is used to transfer files to the Programmer when in USB device mode. It initiates an XMODEM file transfer from the serial port to the specified <partition#>, and names the file that is downloaded <filename>.

Example Usages:

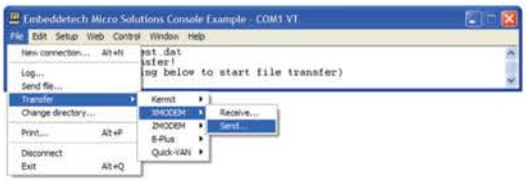
This example downloads a file to partition 0 from the computer and names it "test.dat":

```
[NO MEDIA]>xmodem 0 test.dat
Start XMODEM File Transfer!
(C's will start printing below to start transfer)
CCCC
```

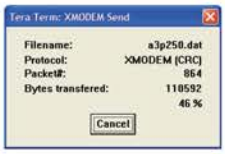
Once the command is entered, the Programmer begins to print the character 'C' to initiate the XMODEM transfer. Once the characters begin to be printed to the screen, the XMODEM transfer should be initiated on the PC end.

The snapshot below shows the console after the command is entered and before the transfer is initiated:

Once the Programmer is printing 'C' characters, the transfer is initiated. Using TeraTerm, this is initiated by using the menu commands File->Transfer->XMODEM->Send…:



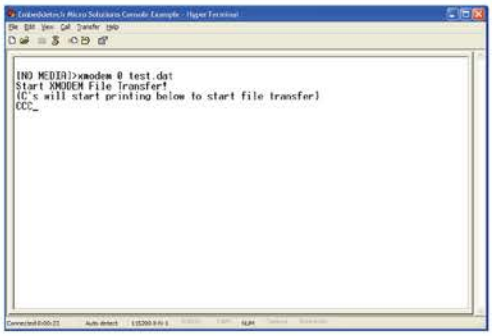
Once the desired file is selected in the dialog box, the transfer will begin, with the packets being quickly sent:



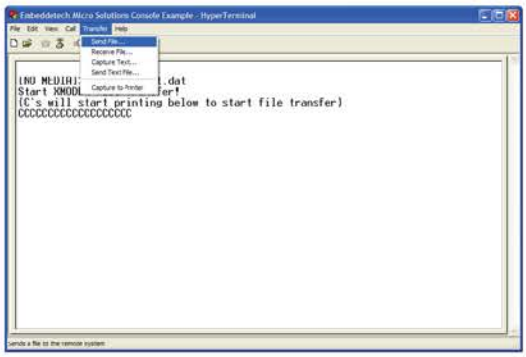
Once the file transfer is complete and the image information is updated in Flash, the command is successfully completed:

```
[NO MEDIA]>xmodem 0 test.dat
Start XMODEM File Transfer!
(C's will start printing below to start transfer)
CCCC
XMODEM File Transfer Successful!
[NO MEDIA]>
```

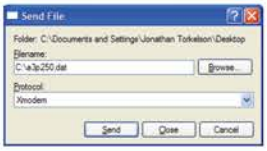
Xmodem file downloading can also be done with HyperTerminal by first entering the command:



… then selecting the menu commands Transfer->Send File…:



Select the desired file and select the XMODEM protocol:



… and then the downloading will begin:

HELP

Command Syntax:

HELP

or

?

This command prints all of the commands supported by the Programmer to the console. Due to the length of the list of commands, the commands are printed in 5 blocks. After each block is printed, the user can continue the printout by pressing enter or cancel the printout by pressing escape. The examples below show the printout after all commands have been printed.

Example Usages:

```
[NO MEDIA]>help
Command Syntax Format:
 - [] indicates list of optional parameters
 - <> indicates list of required parameters
 - All caps indicates a literal parameter
 - (D) specifies the default parameter if no parameter is given
 - "image" indicates USB drive filename or Flash partition index number
 - All filenames must have 8.3 format. Max filename size is XXXXXXXX.XXX
 - Address Argument Format For File I/O: "0xHHHHHHHH", 0xFFFFFFFF = EOF


Program Operation Commands:
    SCAN
    DEVICE           < DEVICE # >
    IRLEN            < DEVICE # >< LEN >
    READ DEVICE INFO [ USING <image> ]
    READ DEVICE ID   [ USING <image> ]
    ERASE            [ USING <image> ]
    PROGRAM          [ USING <image> ]
    VERIFY           [ USING <image> ]
    LOCK             <password>
    UNLOCK           <password>
    REFORMAT
    RECORD START
    RECORD STOP


USB File System & Generic Commands:
    VERSION                        - Prints the Programmer version
    USING  <image>                 - specifies image for subsequent commands
    CD     <name>                  - change directory
    COPY   <image1> <image2>       - copy [image1] to [image2]
    DEL    <image>                 - delete image, current directory only
    DIR                            - display directory
    PRINT  <image>                 - print image contents to console
    HELP or ?                      - display help
    MD     <name>                  - make directory
    RD     <name>                  - remove directory
    REN    <image> <NewName>       - rename <image> to <NewName>
    XMODEM <partition#> <filename> - Initializes XMODEM file download
                                     <partition#>: The partition to load into
                                     <filename>: The name to give the new file
```

```
File Access Commands:
    BINARY                - Data format for reads / writes will be binary
    HEX                   - Data format for reads / writes will be ASCII HEX
    FILE    <image>       - Selects file (partition or filename) to access
    NEW     <image>       - Creates new file or clears existing file
    ADDRESS <address>     - Specifies the current address for file access
                            (0xFFFFFFFF = End Of File)
    READ    <bytes>       - Reads <bytes> (0xHH format) bytes from file
                            Max <bytes> is 0x80
    WRITE <bytes> <data> - Writes <bytes> bytes to file
                            Max <bytes> is 0x80
                            ASCII Data Format: [CHAR0][CHAR1][CHAR2]...
                            BINARY Data Format: [BYTE0][BYTE1][BYTE2]...
                            HEX Data Format: [HH][HH][HH]... (no spaces)
    CLOSE                 - Closes the currently opened file
    REPARTITION <n>       - Repartition the Flash into 'n' images
                            NOTE: THIS COMMAND WILL ERASE ALL FLASH MEMORY!
```

DELAY

Command Syntax:

DELAY <milliseconds>

This command simply delays the specified milliseconds, before returning the prompt. This command is useful during automatic playback to insert delays between commands. The most common example of when this is needed is when a specific target hardware requires a delay after a programming voltage or JTAG voltage rail is turned on, before a JTAG scan or other operation can be successfully performed.

File Access Commands

BINARY

Command Syntax:

BINARY

The console interface allows the ability to read and write data to thumb drive files and partition images directly. The format that the data is written or read can be either a hexadecimal format or a binary format. In the hexadecimal format, the data will be presented as a string of ASCII hexadecimal numbers, such as the four-byte sequence "3F9B183C". In this example, "3F" is the first byte read or written to the lowest memory address, then "9B" is read or written next, then "18", then "3C". Binary mode is the default mode of operation upon every power-up of the Programmer.

Example Usages:

```
[NO MEDIA]>binary
Data Format: Binary
[NO MEDIA]>
```

HEX

Command Syntax:

HEX

The HEX command puts data format in hexadecimal format for direct file or partition access. Thus instead of having the format of "3F9B183C" in the example above, the exact binary bytes would be transmitted or received.

Example Usages:

```
[NO MEDIA]>hex
Data Format: Hexadecimal
[NO MEDIA]>
```

FILE

Command Syntax:

FILE <image>

The FILE command specifies an image to access for direct reading or writing. The file that is specified can either be an integer specifying a partition number to access, or a filename, specifying a file to access on the thumb drive. Once a file is opened with this command, subsequent read/write operations are set at address 0x00000000. Any previously-opened file is closed before this command is executed. This command prints either a value of 0x01 or 0x00 depending on whether or not the file could be opened, respectively. Files must be closed with the "CLOSE" command once file access is completed.

Example Usages:

```
[NO MEDIA]>file 1
0x01
[NO MEDIA]>

A3P250      :\>file image1.dat
0x01
A3P250      :\>
```

NEW

Command Syntax:

```
NEW <image>
```

The NEW command is used to create and open a new file. When used to create a new file on the thumb drive, if the file already exists, all of the data in the file will be deleted. The "NEW" command works with partitions as well, however it only functions to open the specified partition for direct access. It does not delete any of the data in the partition. This command prints either a value of 0x01 or 0x00 depending on whether or not the file could be created, respectively. Files must be closed with the "CLOSE" command once file access is completed.

Example Usages:

```
A3P250      :\>new 1
0x01
A3P250      :\>

A3P250      :\>new test.txt
0x01
A3P250      :\>
```

ADDRESS

Command Syntax:

```
ADDRESS <address>
```

This command specifies the address to use for subsequent file read or write operations. The <address> field must be specified in the 4-byte hexadecimal format "0xHHHHHHHH". An address of 0xFFFFFFFF indicates that the address should be set to the end of the file. If this is used for a partition, then the address will be set to the last byte in the partition, not the last byte that was previously written. However, the EOF address is useful for appending data to a thumb drive file. This command prints either a value of 0x01 or 0x00 depending on whether or not the address could be set, respectively. If the address is out of range of either the partition or the file, a value of 0x00 will be printed.

```
A3P250      :\>address 0x00000000
0x01
A3P250      :\>
```

READ

Command Syntax:

READ <bytes>

This command allows up to 64 bytes of data to be read from either a Flash partition or a thumb drive file. The data is printed in either hexadecimal or binary format, depending on the data format that was previously specified. Before this command is given, a file must first be opened and the address must be specified if not the starting address 0x00000000. Regardless of whether the data format is hexadecimal or binary, the <bytes> argument must always be given as a one-byte hexadecimal number with the format of "0xHH". After each successful read or write operation, the address is incremented the number of bytes that were read or written. This allows large blocks of data to be read with successive READ commands.

This command returns the number of bytes that were successfully read, in a one-byte hexadecimal "0xHH" format, regardless of the current data format.

Example Usages:

This example shows reading the first 64 bytes of the file "temp.txt" in both binary and hexadecimal data format.

```
A3P250      :\>file test.txt
0x01
A3P250      :\>binary
Data Format: Binary
A3P250      :\>read 0x40
0x40 abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ123456789012
A3P250      :\>hex
Data Format: Hexadecimal
A3P250      :\>read 0x40
0x40  6162636465666768696A6B6C6D6E6F707172737475767778797A41424344454647484 94A4B
4C4D4E4F505152535455565758595A313233343536373839303132
A3P250      :\>
```

WRITE

Command Syntax:

WRITE <bytes> <data>

This command allows up to 64 bytes of data to be written to either a Flash partition or a thumb drive file. The data must be provided in either hexadecimal or binary format, depending on the data format that was previously specified. Before this command is given, a file must first be opened and the address must be specified if not the starting address 0x00000000. Regardless of whether the data format is hexadecimal or binary, the <bytes> argument is always given as a one-byte hexadecimal number with the format of "0xHH". After each successful read or write operation, the address is incremented the number of bytes that were read or written. This allows large blocks of data to be written with successive WRITE commands.

IMPORTANT NOTE:
Because this command is processed by a console command interpreter, certain data sequences are not allowed when entering data in binary mode. When writing data to a file, neither the <CR> value 0x0D nor the <LF> value 0x0A may be included in the data sequence of a write command. Instead, these values should be replaced with the two byte characters "\r" and "\n", respectively. If either of these characters are received in the data stream, then the console handler will interpret these as the end of the command, and the command will result in a byte count error.

To allow "\" to be used as a break sequence, if the backslash ASCII value 0x5C is included in the datastream, then it must also be appended with an extra backslash symbol. Thus if the intended data sequence is "a\o<CR><LF>", then the actual sequence in the command line data argument would be "a\\o\r\n".

Any time a backslash character is received and not followed by either another backslash to indicate a literal backslash data byte, or the 'r' or 'n' signifying <CR> or <LF>, then the backslash will be dropped from the data stream.

In addition to scanning the command line for <CR> and <LF> symbols, the console interpreter also looks for the following sequences of characters, to indicate either an up arrow press, a down arrow press, or a backspace press:

<0x1B><0x41> (Up Arrow)
<0x5B><0x41> (Up Arrow)
<0x1B><0x42> (Down Arrow)
<0x5B><0x42> (Down Arrow)
<0x1B><0x08> (Backspace)
<0x5B><0x08> (Backspace)

As a result of these reserved character sequences, if the intended data sequence has any of these two-byte character sequences, then a 'junk' backslash should be inserted between the first and second character. This will ensure that the console interpreter does not interpret the data stream as an attempt to either delete a character or reset the command line to a previously-entered command line.

When the data format is in hexadecimal format, all of the data characters will be <0…9> or <a…f> or <A….F>. Therefore, there is no risk of an unintended special command being interpreted by the console interpreter, so no special precautions are needed .

This command returns the number of bytes that were successfully written, in a one-byte hexadecimal "0xHH" format, regardless of the current data format.

Example Usages:

This example shows writing the string "Hello World!<CR><LF>" to the first 14 bytes of the file "temp.txt". Since this is in binary format, the "\r" and "\n" break sequences were used for <CR> and <LF>.

```
A3P250      :\>file test.txt
0x01
A3P250      :\>binary
Data Format: Binary
A3P250      :\>write 0x0E Hello World!\r\n
0x01
A3P250      :\>address 0x00000000
0x01
A3P250      :\>read 0x4E
0x0E Hello World!

A3P250      :\>
```

CLOSE

Command Syntax:

CLOSE

This command closes the file or partition that is currently opened for direct access. This command returns whether (0x01) or not (0x00) the file could be successfully closed.

Example Usages:

This example demonstrates creating a new file, "test.txt", appending the string "Hello World!<CR><LF>" in binary data format, and closing the file.

```
A3P250      :\>new test.txt
0x01
A3P250      :\>address 0xFFFFFFFF
0x01
A3P250      :\>write 0x0E Hello World!\r\n
0x01
A3P250      :\>close
0x01
A3P250      :\>
```

PRINT

Command Syntax:

PRINT <image>

This command prints the contents of the specified image to the console in both binary and ASCII format in blocks of 128 bytes. After each 128 byte block is printed the console user can either print the next 128 byte block by pressing enter, or abort the printout by pressing <ESC>.

Example Usages:

This example prints the first 128 bytes of the file "a3p250.dat" stored in partition 0:

```
A3P250      :\>print 0
Priting partition 0.
Press ENTER for next block or ESC to exit.
  0x00000000  44 65 73 69 67 6E 65 72   Designer
  0x00000008  20 38 2E 35 2E 32 2E 34    8.5.2.4
  0X00000010  20 20 20 20 20 20 20 20
  0X00000018  38 A2 A7 03 00 00 00 CF   8.......
  0X00000020  41 A1 03 01 00 01 00 01   A.......
[File Printout Continues...]
```

This example prints the first 128 bytes of the file "a3p250.dat" stored on the USB thumb drive:

```
A3P250      :\>print a3p250.dat
Priting file.
Press ENTER for next block or ESC to exit.
  0x00000000  44 65 73 69 67 6E 65 72   Designer
  0x00000008  20 38 2E 35 2E 32 2E 34    8.5.2.4
  0X00000010  20 20 20 20 20 20 20 20
  0X00000018  38 A2 A7 03 00 00 00 CF   8.......
  0X00000020  41 A1 03 01 00 01 00 01   A.......
[File Printout Continues...]
```

INCREMENT

Command Syntax:

INCREMENT <filename>

This command increments the ASCII number stored in the specified thumb drive file. The file must contain only an ASCII integer with no carriage return or line feed. When this command is executed, the file is opened, the number stored in the file is parsed, the number is incremented, and then stored back into the file, in ASCII representation. This number may be useful for keeping track of the number of times a programmer has run its programming script.

Example Usages:

In this example, the file "incr.txt" contains the text "123". After the command is executed the file contains the text "124".

```
A3P250      :\>increment incr.txt
Incremented "incr.txt" to 124
A3P250      :\>
```

REPARTITION

Command Syntax:

REPARTITION <n>

This command repartitions the Flash memory into the specified number of equal-sized memory partitions, up to 32. **WARNING: THIS OPERATION PERMANENTLY DELETES ALL FLASH PARTITION DATA. USE WITH CAUTION!!!**

Example Usages:

```
[NO MEDIA]>repartition 1
Repartitioning Flash... Flash Operation Successful!
[NO MEDIA]>dir
USB disk directory:
No media present.
```

```
External Flash Status:
    Total Memory Detected:          67108864  Bytes
    Useable Memory:                 66584576  Bytes
    # Of Partitions: 1

Flash Directory:
 #     Date      Time         Size      Filename
--- ---------- -------- -------------- ------------
 00 XXXX-XX-XX XX:XX:XX                <Empty>

[NO MEDIA]>


[NO MEDIA]>repartition 32
Repartitioning Flash... Flash Operation Successful!
[NO MEDIA]>dir
USB disk directory:
No media present.

External Flash Status:
    Total Memory Detected:          67108864  Bytes
    Useable Memory:                 66584576  Bytes
    # Of Partitions: 32

Flash Directory:
 #     Date      Time         Size      Filename
--- ---------- -------- -------------- ------------
00 XXXX-XX-XX XX:XX:XX                <Empty>
01 XXXX-XX-XX XX:XX:XX                <Empty>
02 XXXX-XX-XX XX:XX:XX                <Empty>
...
26 XXXX-XX-XX XX:XX:XX                <Empty>
27 XXXX-XX-XX XX:XX:XX                <Empty>
28 XXXX-XX-XX XX:XX:XX                <Empty>
29 XXXX-XX-XX XX:XX:XX                <Empty>
30 XXXX-XX-XX XX:XX:XX                <Empty>
31 XXXX-XX-XX XX:XX:XX                <Empty>

[NO MEDIA]>
```

Command Quick List

The following is a quick reference list of the commands available for configuring the QuickFlash Programmer.

Command Syntax	Purpose
SCAN	Scans the JTAG chain and prints the available devices detected on the chain.
DEVICE <device#>	Specifies which device on the chain to use in subsequent operations. When N devices are attached to the JTAG chain, the valid device #'s are from 1 to N.
IRLEN <device#> <LEN>	Specifies the JTAG instruction register length for the specified device.
READ DEVICE INFO [USING <image>]	Reads the device info of the previously specified device, using the explicitly or previously specified image.
READ DEVICE ID [USING <image>]	Reads the device ID of the previously specified device, using the explicitly or previously specified image.
ERASE [USING<image>]	Erases the previously specified device, using the explicitly or previously specified image.
PROGRAM [USING<image>]	Programs the previously specified device, using the explicitly or previously specified image.
VERIFY [USING<image>]	Verifies the previously specified device, using the explicitly or previously specified image.
LOCK <password>	Locks the Programmer using the specified password as the unlock key.
UNLOCK <password>	Attempts to unlock the Programmer for programming operations using the specified password.
REFORMAT	Deletes all data in the Programmer and clears any lock password so that a password is not required.
RECORD <START | STOP>	Starts or stops script recording. When recording starts, the previous script is deleted. When a script is stored, the recorded script is played back when the PROGRAM button is pressed.
PLAY	Plays back the recorded command script. Equivalent to pressing the PROGRAM button.
VDDL <ON | OFF>	Turns the 2.5V supply to VDDL (pin 24 and 26 of the APA programming connector) on or off. VDDL defaults to OFF when the Programmer powers up.
VPUMP <ON | OFF>	Turns the 3.3V supply to VPUMP (pin 7 of the A3P programming connector) on or off. By default the Programmer monitors VPUMP before a programming operation and turns it on if no voltage is detected. If a VPUMP command is used, the VPUMP voltage stays at that level.
<VJTAG|VDDP> <OFF | 2.5V | 3.3V>	Turns VJTAG/VDDP on at 2.5V or 3.3V, or turns VJTAG/VDDP off. By default VJTAG/VDDP is off at power up and expects the target device to provide the programming voltage level used for the JTAG pins.
VERSION	Prints the Programmer firmware version to the console.
USING <image>	Specifies which file, either a local partition or a file on the thumb drive, is used for subsequent operations.
CD (Change Directory)	Changes the current operating directory in the attached thumb drive.
COPY <image1> <image2>	Copies a file from the source (image1) to the destination (image2). The source or destination files can be either Flash partitions in the Programmer or files on the thumb drive.
DEL <image>	Deletes the specified image from either the USB thumb drive or a Flash partition.
DIR	Prints the entire working directory of both the USB thumb drive and Flash memory.
MD <directory name>	Creates a new folder in the current USB thumb drive working directory.
RD <directory name>	Deletes the specified directory folder in the current working directory of the USB thumb drive.
REN <image> <NewName>	Renames the specified image of either a Flash partition file or thumb drive file to "NewName".
XMODEM <partition#> <filename>	Begins an XMODEM file transfer over the USB serial port to the specified partition with the specified filename.
<HELP | ?>	Prints the available commands to the console.
DELAY <milliseconds>	Delays a specified integer number of milliseconds.
BINARY	Changes the file data read/write format to binary.
HEX	Changes the file data read/write format to hexadecimal.
FILE <image>	Specifies an image in a Flash partition or on the thumb drive for direct reading or writing.
NEW <image>	Creates and opens a new file. Deletes all data in the file if it already exists.
ADDRESS <address>	Specifies the address to use for subsequent file read or write operations. The <address> field must be specified as a hex number with format 0xHHHHHHHH.
READ <bytes>	Reads up to 64 bytes of data from the specified file, in the specified data format (binary or hex).
WRITE <bytes> <data>	Writes up to 64 bytes of data from the specified file, in the specified data format (binary or hex).
CLOSE	Closes the file or partition that is currently opened for direct access.
PRINT <image>	Prints the contents of the specified image to the console in both binary and ASCII format, in blocks of 128 bytes at a time.
INCREMENT <filename>	Reads a thumb drive file, parses an integer, increments the integer, and stores it back in the file.
REPARTITION	Repartitions the Flash memory into the specified number of equal-sized memory partitions up to 32.

PROGRAM Button Script Playback

The PROGRAM button on the top of the Programmer provides a one-button-press user operation to execute the entire script that the user recorded previously. If a thumb drive is connected to the Programmer, all instructions and results are written to report.txt in the root menu as they are executed. If the Programmer is connected to a PC as a USB device, the results of the script playback can be observed as they are executed in any serial port terminal program.

As the script is played back, the BUSY LED will be lit when an individual command is being executed, and when complete either the SUCCESS or FAIL LED will flash momentarily, depending on the result of the command, before the next command is immediately executed. When all commands have been successfully executed and the script is complete, the BUSY light will turn off and stay off, and the SUCCESS LED will remain on until the next command line entry is made.

If the PROGRAM button is pressed and the Programmer has been locked and requires the user to unlock the Programmer before running the script, the FAIL LED will blink three times.

If the PROGRAM button is pressed and the Programmer has been unlocked or was never locked, but no valid script file has been loaded into the Programmer, the FAIL LED will blink twice.

Specifications

Electrical Specifications

External Power Supply:

Input Voltage Range:	100 Vac - 132 Vac
Input Frequency Range:	47 Hz – 63 Hz
Output Voltage:	9Vdc
Output Current (max):	0.66 A
Output Power (max):	6 W
Part Number:	CUI Inc. EPS090066-P1P

Control Outputs:	**Min**	**Max**	**Conditions**
Output Low Voltage (VOL):	0V	0.4V	IOL = 5mA
Output High Voltage (VOH):	2.4V	-	IOH = -5mA

Control Input (PROGRAM):	**Min**	**Max**
Input Low Voltage (VIL):	0V	0.66V
Input High Voltage (VIH):	2.64V	3.3V

+3.3V Power Output	**Min**	**Max**
Current Draw	-	200mA

Environmental Specifications

Operating Temperature Range:	0°C to 40°C
Storage Temperature Range:	-10°C to 70°C
Operating Humidity:	20% to 80%
Storage Humidity:	10% to 90°C

Mechanical Specifications

Enclosure Width:	3.800"
Enclosure Length:	5.004"
Enclosure Height:	1.650"
Weight (Carrying Case w/ Programmer):	1.45 lbs

6415 S 110th E Ave
Tulsa, OK 74133, USA

www.embeddetech.com