Rate This Document

FlashPro User's Guide

for Software v9.1

Actel

POWER MATTERS

Actel Corporation, Mountain View, CA 94043

Printed in the United States of America

Part Number: 5-02-9138-17

Release: November 2010

Trademarks

Table of Contents

About FlashPro

FlashPro is Actel's programming software tool for IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, and ProASIC devices. You will be able to navigate easily through the FlashPro software because of its similarities with other Actel software tools. The FlashPro software includes the following features:

- Supports modification of I/O states during programming
- Supports automatic construction of chain from scan chain operation
- Supports importing non-Actel BSDL files for automatic chain construction
- Supports direct multiple Actel device chain programming and serialization
- Supports single device STAPL files generation
- Supports single device SVF files generation
- Supports single device IEEE 1532 files generation
- Supports Chain STAPL file generation
- Supports Chain SVF file generation
- Supports a single GUI to drive multiple FlashPro4/3/3X programmers for parallel programming
- Supports 1.2V programming for IGLOO devices

Note: Note: Parallel programming via FlashPro (USB/LPT1) or FlashPro Lite programmers is not supported.

- Supports device serialization for parallel programming
- A redesigned GUI, which features a project manager to manage the programming files and data
- Enhanced In-System Programming (ISP) Support

An optional In-House Programming (IHP) service is available if you are purchasing Actel devices in volume. Contact Actel for more information.

For step-by-step instructions on how to use these features, see the FlashPro Tutorial.

If you arrived here by pressing the F1 key in FlashPro, use the **Search** tool in help for more information on specific content, or click the **Help** button embedded in any dialog box or GUI for context-specific help.

Programming Tool Model Overview

The FlashPro software is designed for use in the operation, user design, and production programming flows.

Design Debug

The figure below illustrates the programming design flow when an engineer is in debug mode. In the programming design flow, the new Programming files (STAPL/ PDB) are generated for a design change and are sent to the FlashPro software for testing and debugging the design.

Note: FlashPoint is integrated into Designer; therefore, the STAPL file is generated from Designer. FlashPro v6.2 and greater can be used to export STAPL files from PDB files created by FlashPoint (Designer).

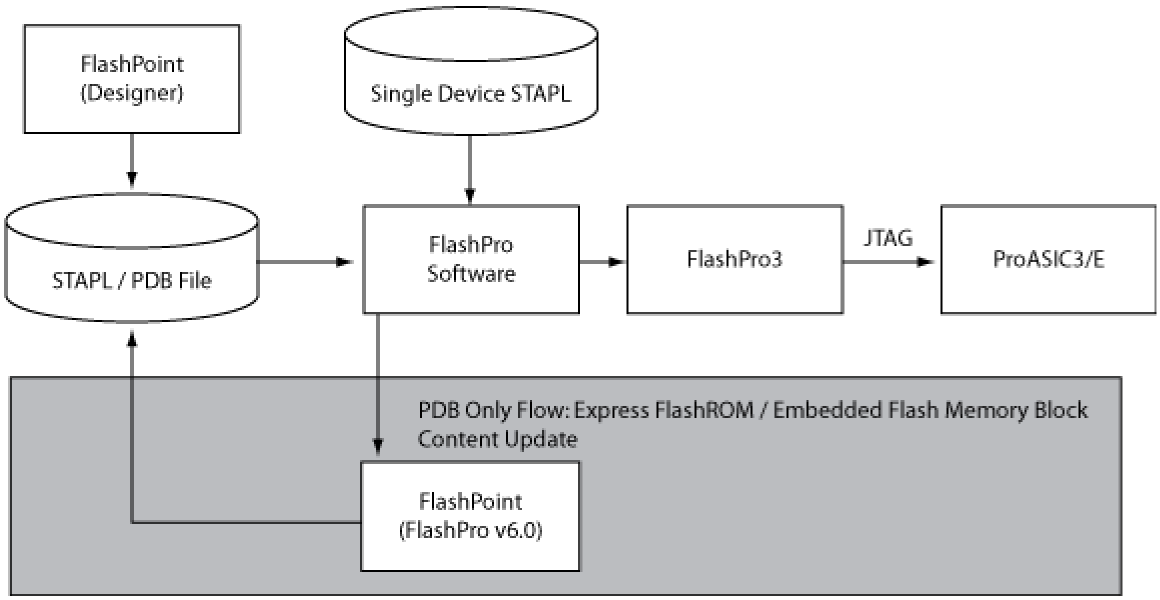


Figure 1 · Programming Design Flow

Operation/Production Planning

The figure below shows an illustration of the operation flow. In this illustration, the production coordinator generates the programming files (STAPL/PDB) with or without serialization and/or security settings (see Programming application note for further information). The production coordinator loads the programming file in the FlashPro software to set up the configurations for production programming, such as Serialization options, Action selections, and Procedure selections, etc.

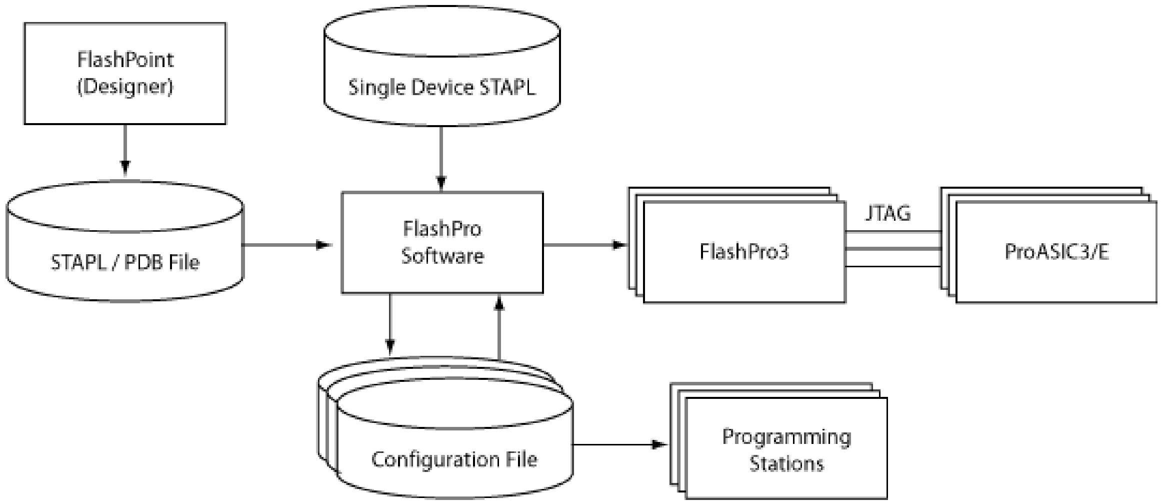


Figure 2 · Operation Flow

The production coordinator may want to generate different configuration files for each programming station (depending on the logistics and serialization options). For example, if the Programming file contains 10,000 serial data and the production coordinator decides to split the serial data designation to one thousand for each programming station, then ten configuration files will be generated (one for each of the ten programming stations). However, if you are not using serialization, you only need one configuration file.

The production coordinator can test the configuration files with one or more FlashPro4/3/3X programmers before sending it to the production programming floor. If PDB files are used in the production flow, warning icons may appear on the FlashPro/FlashPoint GUI because the automatic audit cannot find the source file on the production environment; the PDB file contains the valid programming data. If STAPL files are used, loaded STAPL files will be audited on execution of an action to determine if the original STAPL file has been modified. If it has not been modified, the action will continue to run. If it has been modified you will be prompted to reload the modified STAPL file, or to continue running the current action. If you select to reload the modified STAPL file, all previous programming settings will be refreshed and will need to be performed again.

Operation/Production Programming

The figure below shows an illustration of the production programming flow. The operator imports the configuration file and begins programming the devices by clicking the **Run** button. The operator's interaction with FlashPro should be limited.

At the end of a programming session, the serialization log file (if applicable) and the programming log file are sent back to the production coordinator for record keeping.

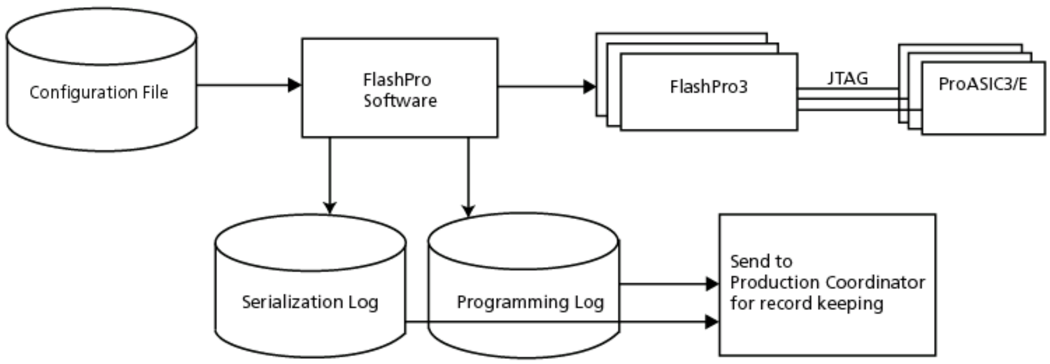


Figure 3 · Production Programming Flow

Express Configuration Programming (IGLOO, ProASIC3 and Fusion devices only)

The figure below illustrates the Express Configuration Programming Flow. In this flow, you can program the security setting into the IGLOO, ProASIC3 and Fusion family device directly from the FlashPro software.

Note: FlashPoint is integrated into the FlashPro v6.0 and later software.

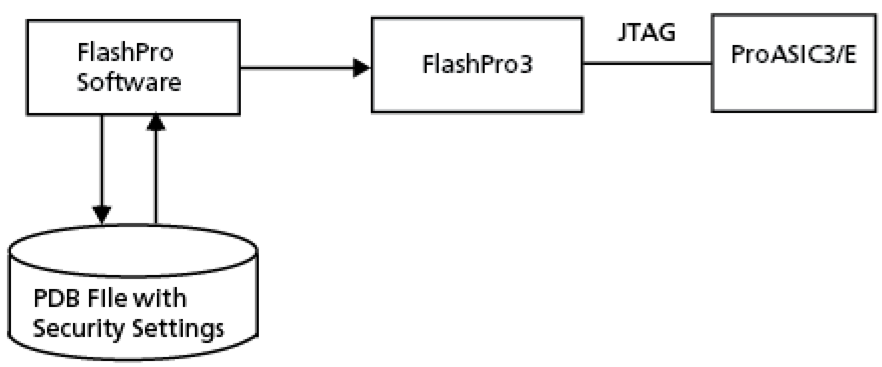


Figure 4 · Express Configuration Programming Flow

Programming Tool User Model Overview - SmartFusion Only

The FlashPro software is designed for use in the operation, user design, and production programming flows.

Design Debug

The figure below illustrates the programming design flow when an engineer is in debug mode. In the programming design flow, the new files (FDB, UFC, EFC) are generated for a design change and are sent to the FlashPro software for testing and debugging the design.

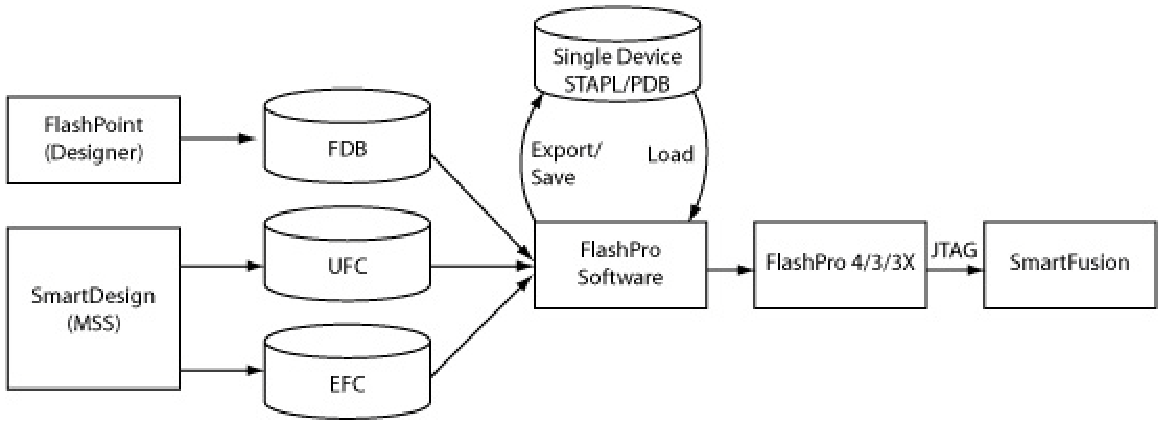


Figure 5 · SmartFusion Programming Design Debug Flow

Operation/Production Planning

The figure below shows an illustration of the operation flow. In this illustration, the production coordinator generates the programming files (STAPL/PDB) with or without serialization and/or security settings (see Programming application note for further information). The production coordinator loads the programming file in the FlashPro software to set up the configurations for production programming, such as Serialization options, Action selections, and Procedure selections, etc.

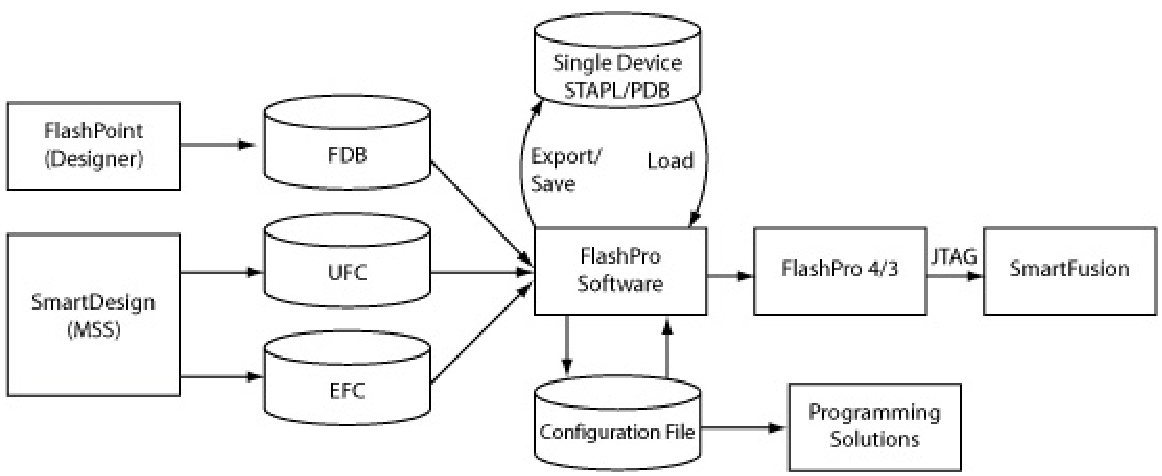


Figure 6 · SmartFusion Operation Flow

The production coordinator may want to generate different configuration files for each programming station (depending on the logistics and serialization options). For example, if the Programming file contains 10,000 serial data and the production coordinator decides to split the serial data designation to one thousand for each programming station, then ten configuration files will be generated (one for each of the ten programming stations). However, if you are not using serialization, you only need one configuration file.

The production coordinator can test the configuration files with one or more FlashPro4/3/3X programmers before sending it to the production programming floor. If PDB files are used in the production flow, warning icons may appear on the FlashPro/FlashPoint GUI because the automatic audit cannot find the source file on the production environment; the PDB file contains the valid programming data. If STAPL files are used, loaded STAPL files will be audited on execution of an action to determine if the original STAPL file has been modified. If it has not been modified, the action will continue to run. If it has been modified you will be prompted to reload the modified STAPL file, or to continue running the current action. If you select to reload the modified STAPL file, all previous programming settings will be refreshed and will need to be performed again.

Operation/Production Programming

The figure below shows an illustration of the production programming flow. The operator imports the configuration file and begins programming the devices by clicking the **Run** button. The operator's interaction with FlashPro should be limited.

At the end of a programming session, the serialization log file (if applicable) and the programming log file are sent back to the production coordinator for record keeping.

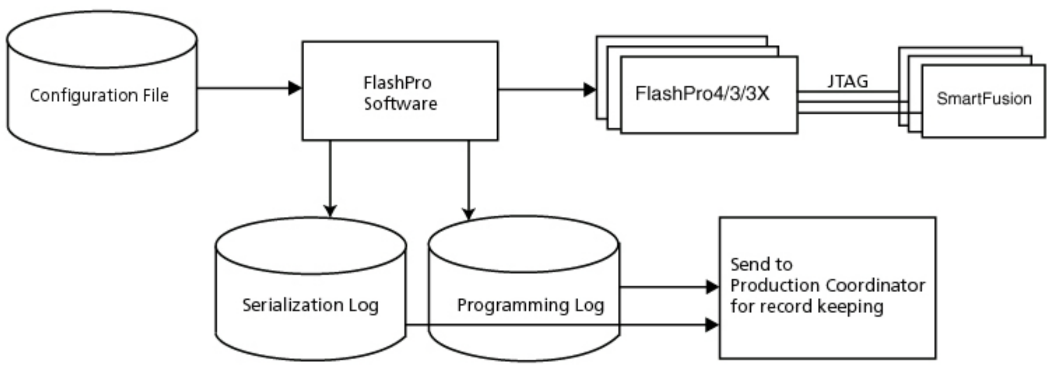


Figure 7 · SmartFusion Production Programming Flow

Creating a New PDB for SmartFusion

The figure below illustrates the new SmartFusion programming flow. In this flow you can program the security, FPGA Array, FlashROM and Embedded Flash Memory (NVM) for SmartFusion.

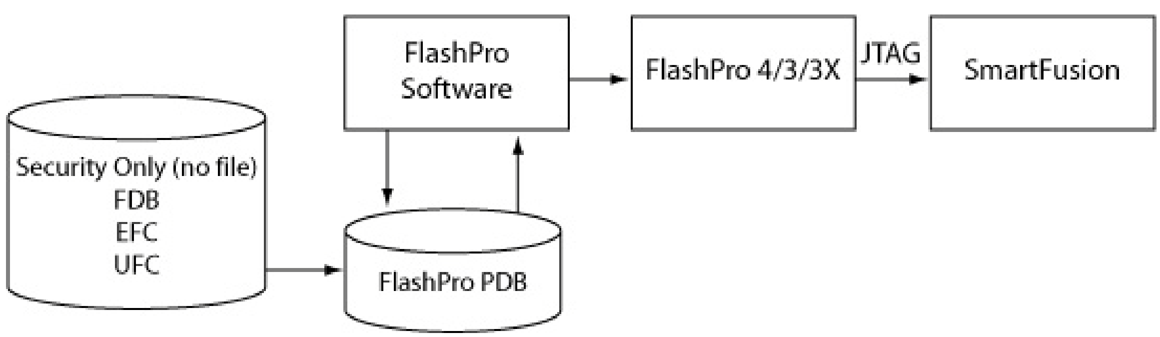


Figure 8 · Creating a New PDB for SmartFusion

Supported Families

Actel's Libero® Integrated Design Environment (IDE) and Designer software support the following families of devices:

- IGLOO®
- ProASIC3
- SmartFusion
- Fusion
- ProASICPLUS
- ProASIC
- Axcelerator
- eX
- SX-A
- MX
- RTAX-S/SL
- RTSX-SU

When we specify a family name, we refer to the device family and all its derivatives, unless otherwise specified. See the table below for a list of supported device families and their derivatives:

Table 1 · Actel's Product Families and Derivatives

Device Family	Family Derivatives	Description
IGLOO	IGLOO	The ultra-low-power, programmable solution
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest power, smallest size solution
	IGLOO PLUS	The low-power FPGA with enhanced I/O capabilities
ProASIC3	ProASIC3	The low-power, low-cost, FPGA solution
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards

Device Family	Family Derivatives	Description
	ProASIC3 nano	Lowest cost solution with enhanced I/O capabilities
	ProASIC3L	The FPGA that balances low power, performance, and low cost
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
SmartFusion	SmartFusion	SmartFusion intelligent mixed-signal FPGAs are the only devices that integrate an FPGA, ARM Cortex-M3, and programmable analog, offering full customization and IP protection.
Fusion	Fusion	Mixed-signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device.
ProASIC PLUS	ProASIC PLUS	Second-generation, high-density programmable flash devices with ASIC capabilities in a single-chip (75 k to 1 million gates)
ProASIC	ProASIC	This family has been discontinued and it is not recommended for new designs
Axcelerator	Axcelerator	Nonvolatile, high-speed antifuse FPGAs with FuseLock™ design security and embedded FIFO controller (125 k to 2 million gates)
eX	eX	Third-generation, low power, low density antifuse devices based on the SX-A architecture with greater than 350 MHz performance (3 k to 12 k gates)

Device Family	Family Derivatives	Description
SX-A	SX-A	Antifuse devices with 270 MHz system performance and sea-of-modules architecture enabled by Actel's patented metal-to-metal antifuse interconnect elements (12 k to 108 k gates)
MX	MX	Antifuse devices with 250 MHz system performance and MultiPlex I/O, an architectural feature that supports mixed-voltage systems and delivers high-performance operation at 5.0 V (3 k to 54 k gates)
RTAX-S/SL	RTAX-S/SL	New generation of high-reliable, radiation-tolerant, antifuse-based FPGAs, designed for space applications with greater than 350 MHz system performance (250 k to 4 million system gates)
RTSX-SU	RTSX-SU	High-reliable, radiation-tolerant antifuse-based FPGAs with 250 MHz system performance (48 k to 108 k system gates)

Installing FlashPro Software and Hardware

See the FlashPro Installation Instructions on the Actel website for information on how to install FlashPro software/hardware and relevant system requirements.

View the detailed Install Instructions and System Requirements at the Actel software System Requirements page:

http://www.actel.com/products/software/libero/sysreqs.aspx

Starting FlashPro

You can start the FlashPro software from **Programs > Actel FlashPro vx.x > FlashPro**. If you installed the program in a folder other than FlashPro, choose that folder from the **Programs** menu.

The figure below shows the FlashPro GUI. From this GUI, you can create a new project by clicking the **New Project** button or open an existing project by clicking the **Open Project** button.

You can also access the above features from the menu bar. You can access all the other features after you open or create a new project.

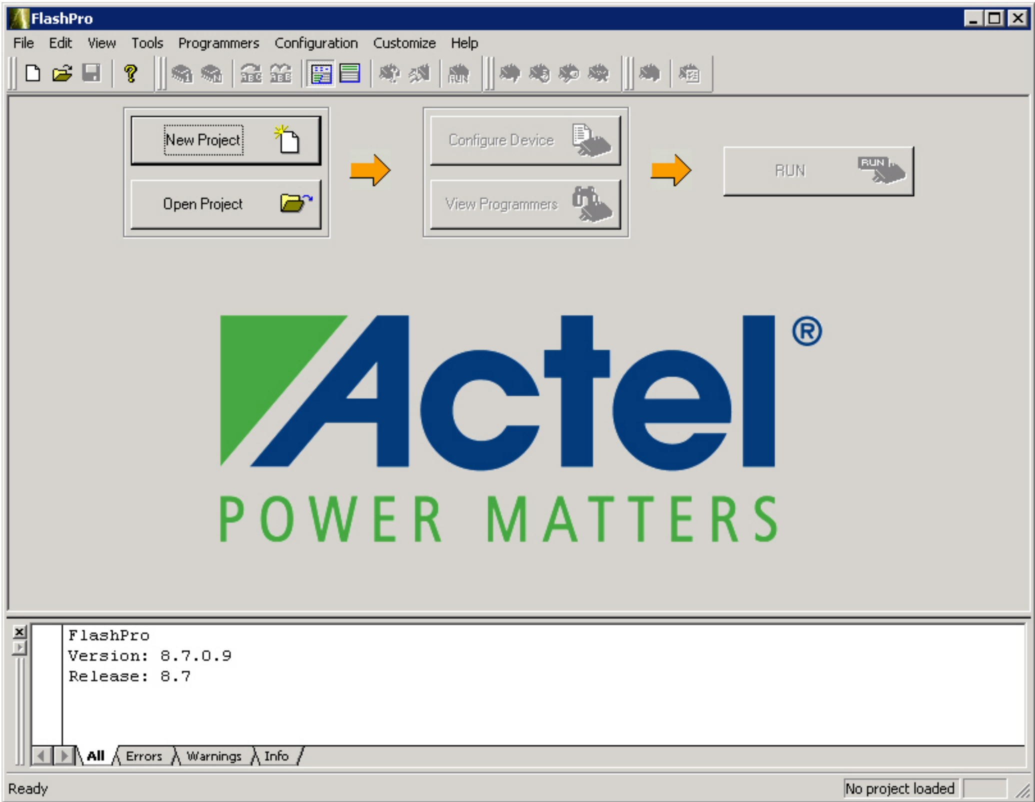


Figure 9 · FlashPro Startup Window

FlashPro Interface

The main FlashPro interface consists of two views, one for Single Device Programming and the other for Chain Programming (see figure below). The GUI consists of a Flow window, Device Configuration Window (for single or chain programming), Log window and a Status bar. The Log window displays programming information, error messages, and warning messages. The Status bar displays your programming mode (chain programming or single device programming) and file status.

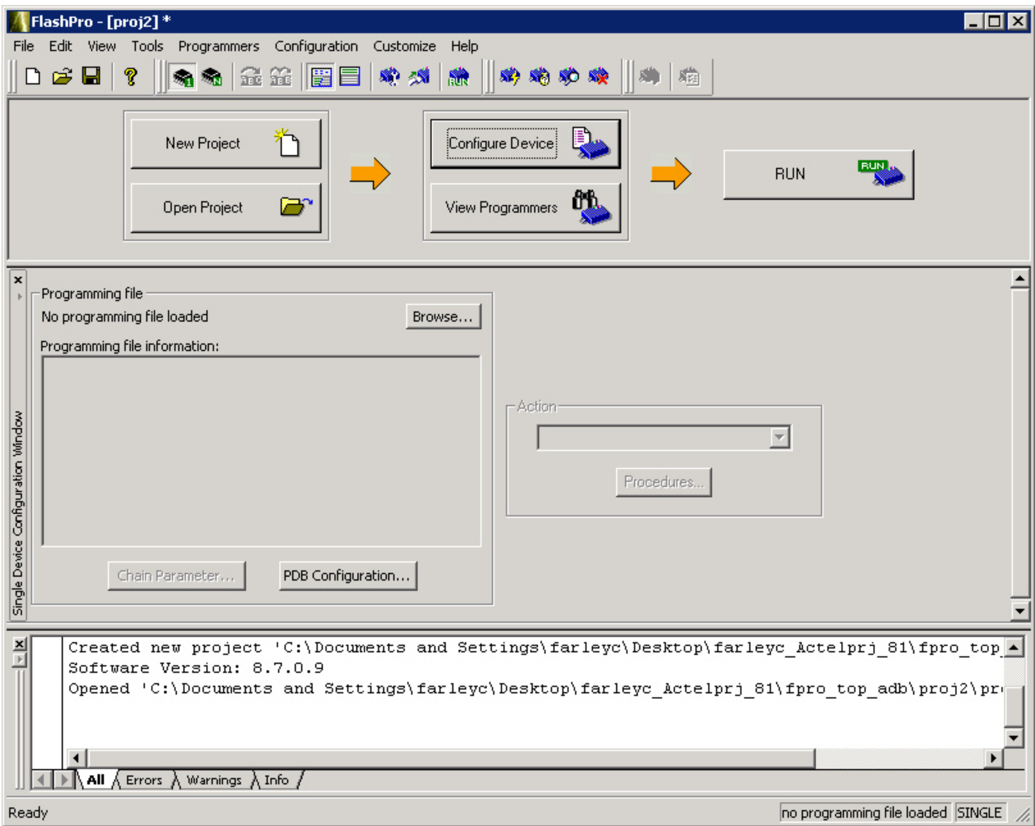


Figure 10 · FlashPro for Single Device File Programming

Note the different options in the Flow window for the Chain Programming GUI and the Single Device Programming GUI. In addition to the different Flow window options, the Chain Programming GUI view consists of the Chain Configuration window which displays the devices in your chain.

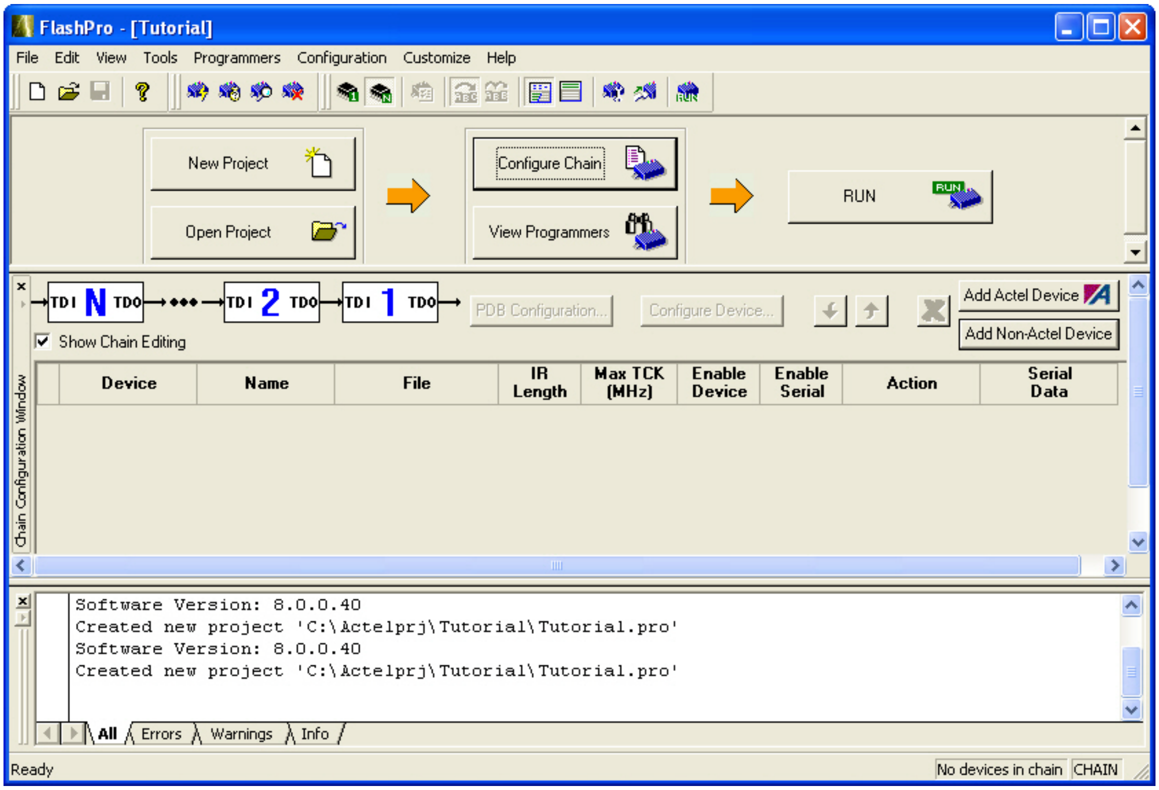


Figure 11 · FlashPro for Chain Programming

Creating a New Project

With the FlashPro software, you have the option of choosing either the **Single STAPL file** or **Chain programming** mode. You make this choice through the **New Project** dialog box (see figure below). By choosing the **Chain Programming** mode, you are enabling chain programming. The **Single STAPL file Programming** mode functions with the same programming capabilities as the FlashPro software v4.2.

To create a new project:

1. Click the **New Project** button or from the **File** menu choose **New Project**.
2. From the **New Project** dialog box, type in the name of your project in the **Project Name** field.

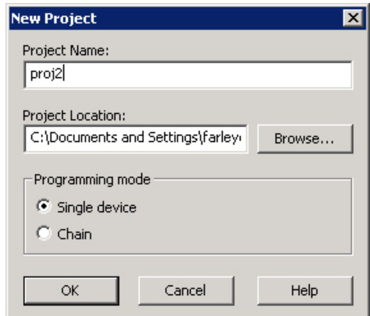


Figure 12 · New Project Dialog Box

3. If necessary, change the default location of your project in the **Project Location** field.
4. Choose your **Programming** mode (Single device or Chain).
5. Click **OK**. The FlashPro GUI displays (see figure below).

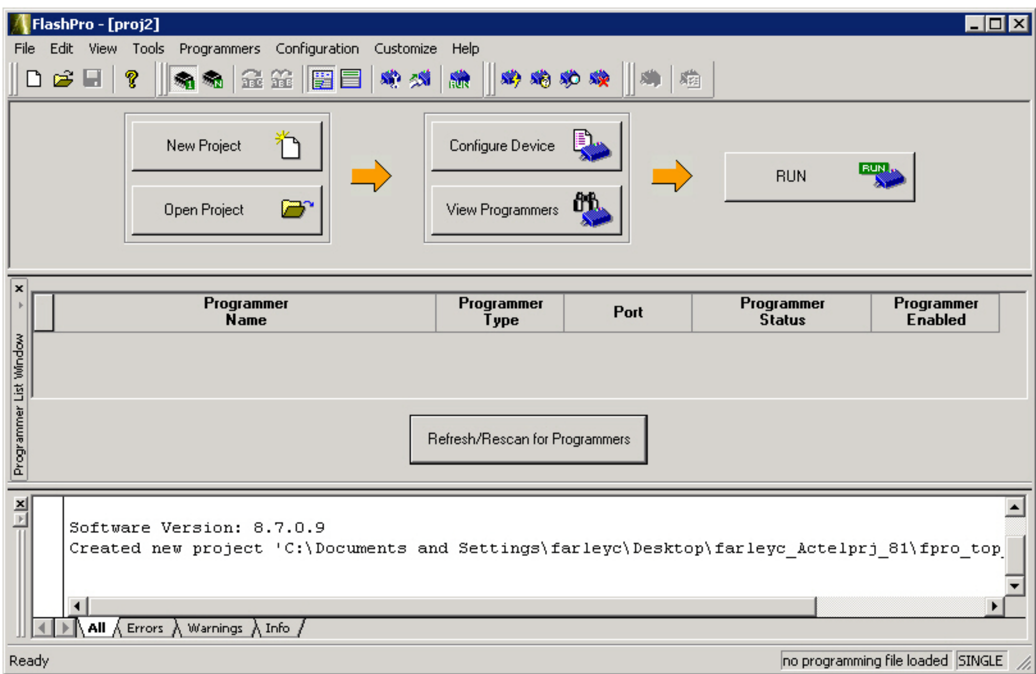


Figure 13 · FlashPro GUI

Note: You can switch between the two programming modes from Tools > Mode. From there, you can choose either Single Device Programming or Chain Programming.

Opening a Project

You can open a project from the **File** menu or by clicking on the **Open Project** button in the flow window.

To open a project:

1. From the **File** menu, choose **Open Project**. The **Open Project** dialog box appears.

2. Find your project file or type in your project file name in the **File name** field.
3. Click **Open**.

Saving a Project

Click the **Save** button on the toolbar, or from the **File** menu choose **Save Project** to save your project.

If you want to save your project under a different name/path, from the **File** menu choose **Save Project As** and save your project with the new name.

Parallel Programming with FlashPro4/3/3X

Parallel programming enables you to program multiple Actel devices in parallel with multiple programmers. In parallel programming, all targeted devices are programmed with the same programming file (STAPL). The targeted device or chain configuration that is connected to each programmer must be identical.

The FlashPro software together with the FlashPro4/3/3X programmers supports parallel programming via a USB port. You can connect up to sixteen FlashPro4/3/3X's to a PC via a USB v1.1 or a USB v2.0 port. FlashPro4/3/3X requires a self-powered hub.

Connecting FlashPro4/3/3X (a USB v2.0 enabled programmer) to USB v1.1 port increases device programming time due to a slow data transfer rate on the USB v1.1 port in comparison to a USB v2.0 port.

Note: FlashPro (USB/LPT1) or FlashPro Lite programmers do not support parallel programming.

The following figure illustrates how you can connect a FlashPro4/3/3X programmer for parallel programming.

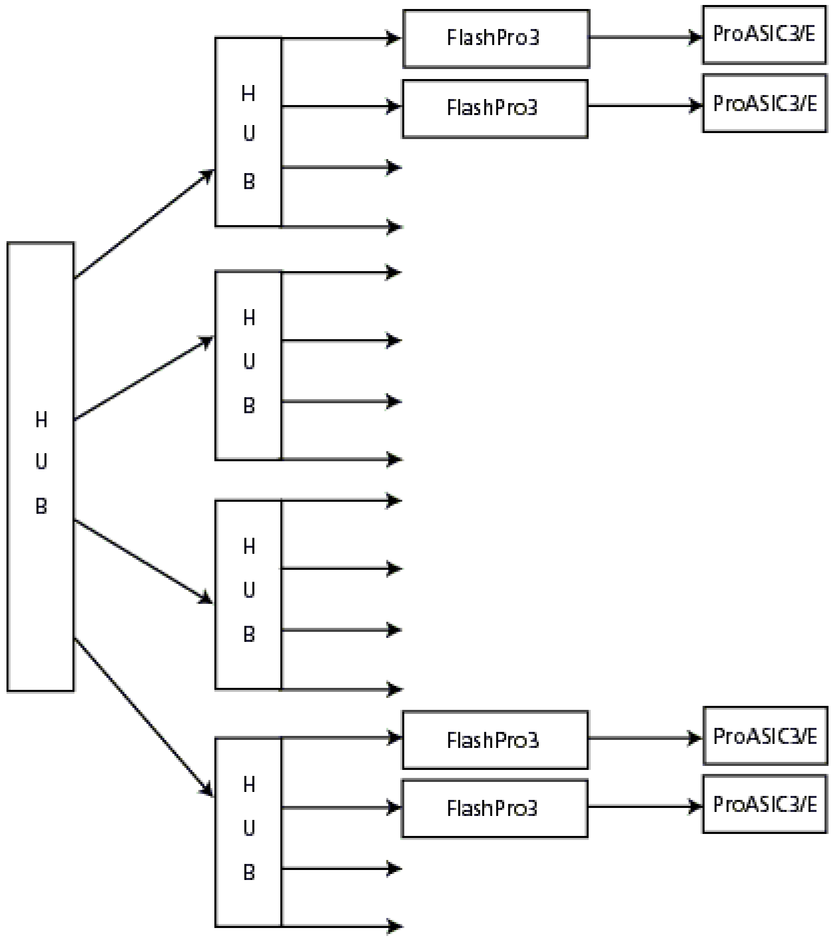


Figure 14 · Connecting a FlashPro4/3/3X Programmer

An independent thread processes the STAPL file during parallel programming. In an Actel test, parallel programming is approximately five times faster than programming 16 devices sequentially.

Note: Actel has tested Belkin, Adaptec, and D-Link PCI-USB cards and hubs. We have found that parallel programming works best with the vendor's latest driver installed and with the matching hubs.

Understanding Serialization

You can use the FlashROM in the ProASIC3 device for serialization. For each target ProASIC3 device, different FlashROM contents are generated.

Serial Programming enables you to program a sequence of ProASIC3 devices in serial with an identical FPGA program and with different serialization data. Serialization data can consist of different FlashROM content and/or AES key values. To learn how to activate the serialization feature, see Skip Serial Data or Reuse Serial Data.

There are two different STAPL formats that support serial programming, multiple actions to multiple serial data and single action to multiple FlashROM.

Multiple Actions to Multiple FlashROM Serial Data

This format supports a generic STAPL player because the STAPL player does not provide a mechanism for Serial Programming. One programming action is created to target different serial data. See examples below:

- PROGRAM_1 programs the FPGA Array and the first serial data.
- PROGRAM_2 programs the FPGA Array and the second serial data.

Single Action to Multiple FlashROM Serial Data

This format is created when the target programmer is FlashPro, Sculptor II, or BP auto programmer, where the newly innovated Actel Serial Programming mechanism is supported. One programming action will program multiple serial data in serial.

FlashPro and SVF

SVF (Serial Vector Format) is an industry standard file format that is used to describe JTAG operations. Like STAPL files, SVF files are used for describing the in-system programming algorithm for IGLOO, ProASIC3, SmartFusion and Fusion family devices. Unlike STAPL files, SVF files support only one ACTION or programming flow per file, due to language limitations. In addition, the SVF specification does not support message display and flow control, such as conditional statements or loops.

As a result, Actel tools (Designer and FlashPro software) generate a set of SVF files corresponding to the equivalent STAPL ACTIONS that are applicable to the silicon features selected.

For example, for a typical STAPL file that has the following ACTIONS: ERASE, ERASE_ALL, PROGRAM, PROGRAM_ARRAY, VERIFY, VERIFY_ARRAY, DEVICE_INFO, READ_IDCODE, and VERIFY_DEVICE_INFO, a set of corresponding SVF files are generated and named: ERASE.svf, ERASE_ALL.svf, PROGRAM.svf, PROGRAM_ARRAY.svf, etc. These files are generated in a folder, <Programming File Name>_svf, created during generation. The diagram below demonstrates the differences between the STAPL and SVF files that are created.

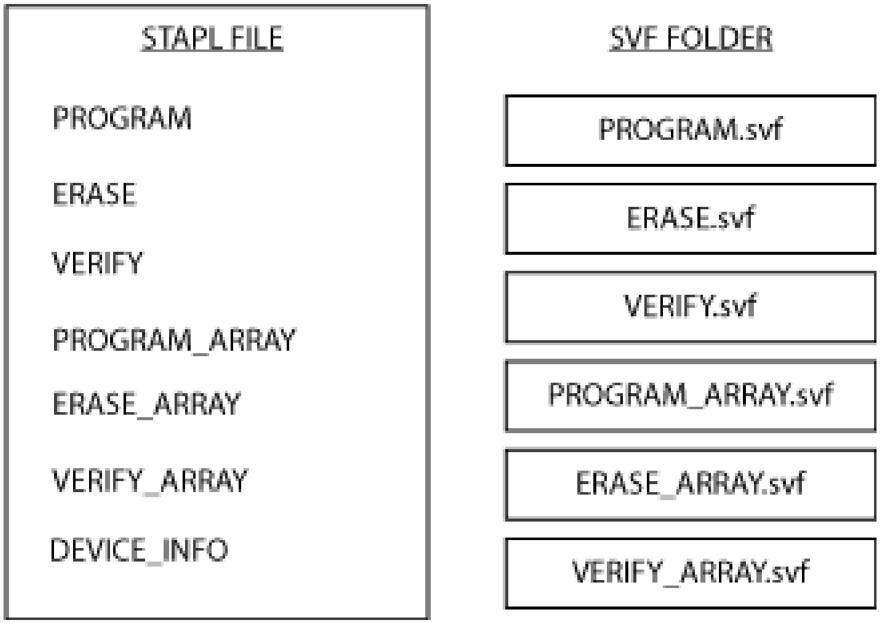


Figure 15 · STAPL vs SVF files

NOTE: DEVICE_INFO.svf file is not generated because SVF files do not support messsage display or flow control.

Table 2 · SVF Outline

SVF File	Array	FROM	NVM (Flash Memory System Builder)	Security Settings	Previously Programmed Device?
ERASE	X	X			YES or NO
ERASE_ALL	X	X		X	YES or NO
ERASE_ARRAY	X				YES or NO
ERASE_FROM		X			YES or NO
ERASE_SECURITY				X	YES or NO
PROGRAM	X	X			YES or NO
PROGRAM_ARRAY	X				YES or NO
PROGRAM_FROM		X			YES or NO
PROGRAM_NVM			X		YES or NO

SVF File	Array	FROM	NVM (Flash Memory System Builder)	Security Settings	Previously Programmed Device?
PROGRAM_SECURITY				X	YES or NO
VERIFY	X	X	X		YES or NO
VERIFY_ARRAY	X				YES or NO
VERIFY_FROM		X			YES or NO
VERIFY_NVM			X		NO
ENC_DATA_AUTHENTICATION	X				YES

STAPL Actions not Available with SVF

The following STAPL actions are not available with SVF: DEVICE_INFO, VERIFY_DEVICE_INFO, READ_IDCODE

FlashPro and the 1532 File Format

1532 is an IEEE industry standard file format that is used to describe JTAG operations. Like STAPL files, 1532 files are used for describing the in-system programming algorithm for IGLOO, ProASIC3, SmartFusion and Fusion family devices. 1532 programming file generation will generate two files (*.isc, *.bsd) within a folder.

The folder will be created with the following name <Programming File Name>_1532. The *.bsd file contains the IEEE 1532 programming algorithm. The *.isc file contains the programming data to be programmed into the device.

IEEE 1532 programming files will only be exported in FlashPro for SmartFusion devices when an FDB has been properly imported.

STAPL to 1532 Action Mapping

The IEEE 1532 standard requires using default ACTION names in order to function with 1532 compliant players. The table below describes the STAPL to 1532 ACTION name mappings.

NOTE: 1532 ACTIONs can have a data member parameter to allow reuse of the same ACTION name for different features.

Table 3 · STAPL to 1532 Action Name Mapping

STAPL Action	1532 Action
ERASE_FROM	ERASE(FROM)
PROGRAM_FROM	PROGRAM(FROM)
VERIFY_FROM	VERIFY(FROM)
PROGRAM	PROGRAM
PROGRAM_ARRAY	PROGRAM(ARRAY)
ERASE_ARRAY	ERASE(ARRAY)
ERASE	ERASE
ERASE_ALL	ERASE(ALLDATA)
VERIFY	VERIFY
VERIFY_ARRAY	VERIFY(ARRAY)
READ_IDCODE	READ(IDCODE)
ENC_DATA_AUTHENTICATION	VERIFY(ENCDATA)
PROGRAM_SECURITY	PROGRAM(SECURITY)
DEVICE_INFO	READ
VERIFY_NVM	VERIFY_NVM
VERIFY_SECURITY	VERIFY(SECURITY)
PROGRAM_NVM	PROGRAM_NVM

STAPL Actions not Available with 1532

The following STAPL action is not available with 1532: VERIFY_DEVICE_INFO

Introductory Programming Tutorials

Single STAPL/PDB File Basic Tutorial

This section provides step-by-step instructions to familiarize you with the basic features of the FlashPro software, specifically how to program a device. For more detailed step-by-step instructions and help with advanced features of the software, please see specific topics in the online help.

Note: This tutorial assumes that you have already installed the latest version of FlashPro software and have started the program.

First, create a new project and name it Tutorial. If FlashPro is launched through the Libero IDE, a new project will be created automatically and a PDB or FDB file loaded, if available.

To Create a Project:

1. Click the **New Project** button in FlashPro.
2. In the **New Project** dialog box, type Tutorial in the **Project Name** field.

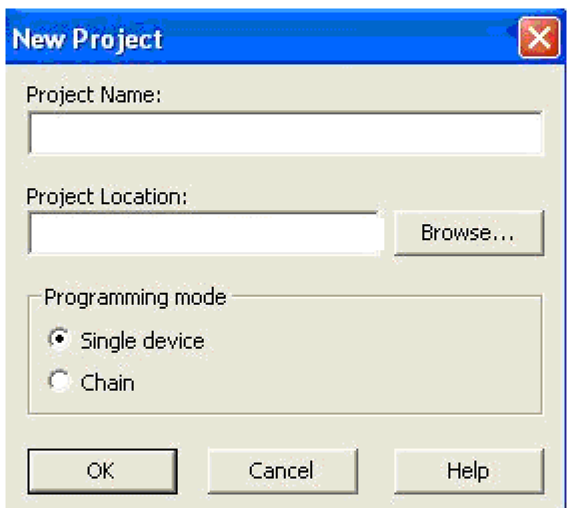


Figure 16 · New Project Dialog Box

3. If necessary, change the default location of your project in the **Project Location** field.
4. Select the Single device **Programming** mode
5. Click **OK**. The FlashPro GUI displays (see figure below). The Programmer List Window updates with your programmer information.

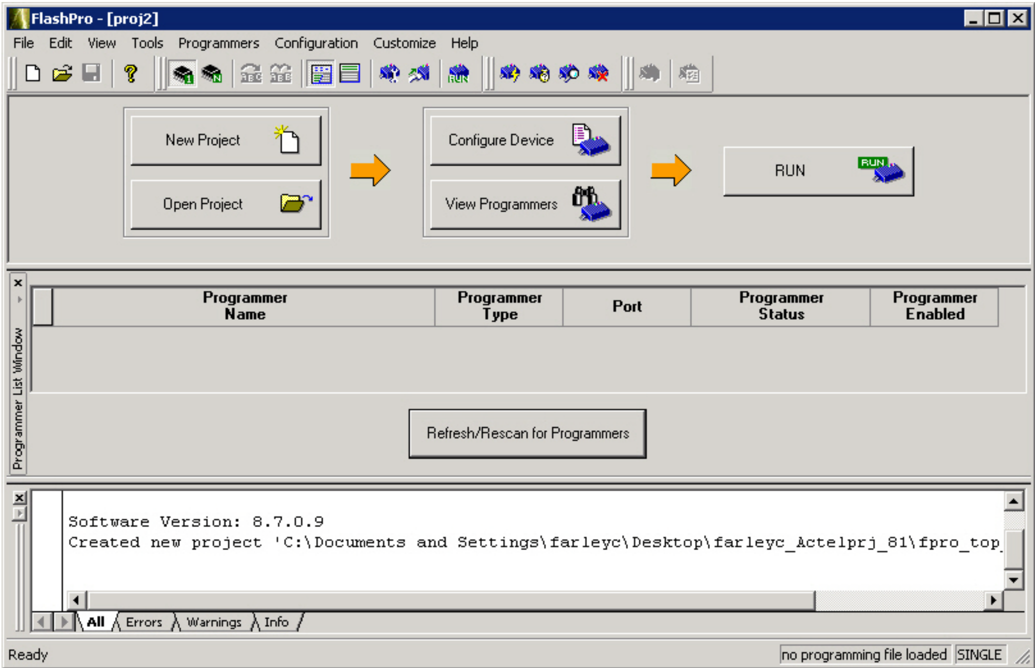


Figure 17 · FlashPro Main Window

Loading and Configuring a Programming File

Once you have created your project and connected your programmer, you are ready to load your PDB or STAPL file.

To load a Programming file:

1. Click the **Configure device** button. The **Single Device Configuration** window displays in FlashPro .
2. Click the **Browse** button to find your Programming file.
3. From the **Load Programming File** dialog box, select your Programming file and click **Open**.

The Single Device Configuration Window updates to list your Programming file information and the actions available with your Programming file in the Action list box (see figure below). Program is the default action displayed in the Action list box.

Note: Actel recommends using the default settings.

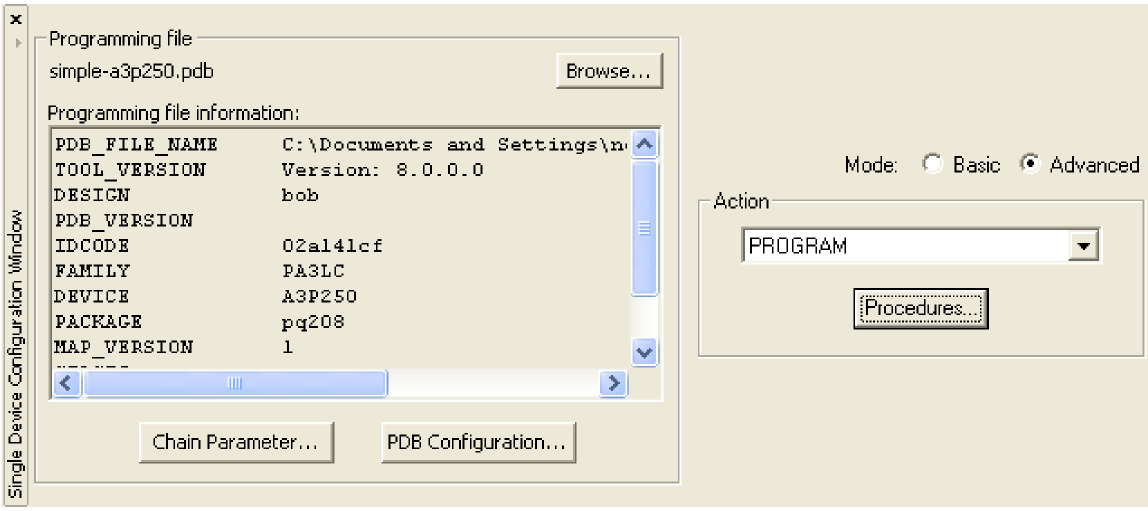


Figure 18 · Single Device Configuration Window

This tutorial gives instructions on how to program a device. For an explanation on the other actions available, see Programming File Actions.

Programming a Device

Now that you have loaded your PDB file, programming a device is the next step.

To program a device:

1. From the **Action** list, select **Program** (see figure below).

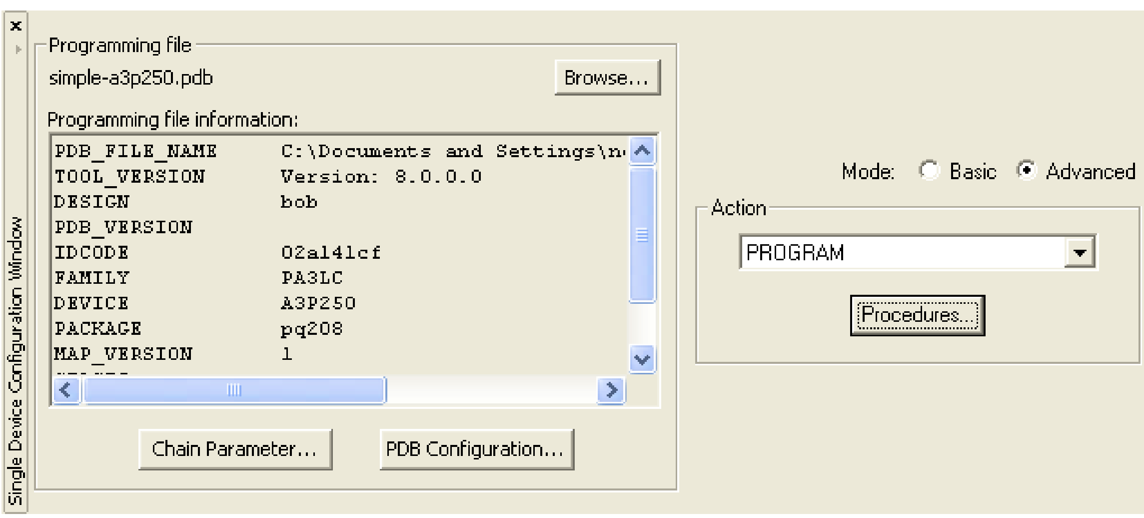


Figure 19 · Selecting Program from the Action List box

2. Click the **Procedures** button (see figure below).

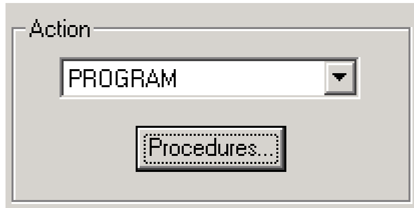


Figure 20 · Procedures Button

The **Select Action And Procedures** dialog box appears, showing the procedures for the Programming action (see figure below). Actel recommends using the default settings.

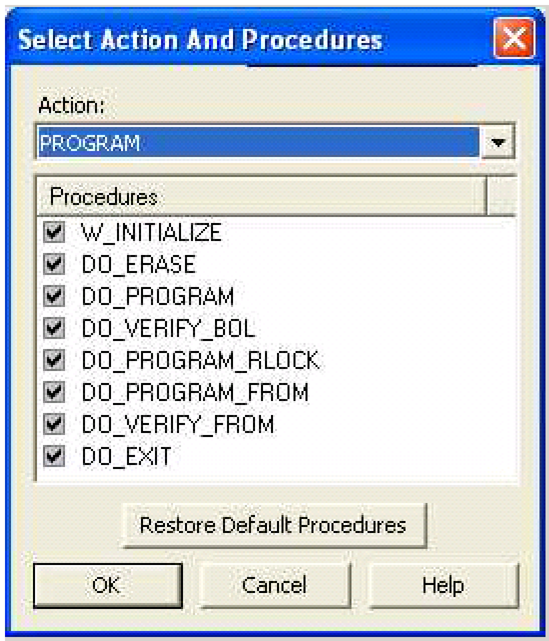


Figure 21 · Select Action and Procedures dialog box

3. Click the **Restore Default Procedures** button.
4. In FlashPro click the **Program** button to program your device.

The **Programmer List Window** updates the **Programmer Status** column with Run Passed indicating that you have successfully programmed the device (see figure below).

Note: The status indicator updates during programming to show the programming progress, then it will change to a pass or fail result when the operation is complete.

	Programmer Name	Programmer Type	Port	Programmer Status	Programmer Enabled
1	03375	FlashPro3	usb03375 (USB 1.1)	RUN PASSED	☑

Figure 22 · Successfully Programmed Device

5. View the **Log** window and take note of the details about your programmed device.

Single Actel Device with Serialization Tutorial

This tutorial provides step-by-step instructions on how to program a single Actel Device with Serialization. Before you begin this tutorial, make sure you have already installed the FlashPro software and that you are familiar with the basic features of using the FlashPro software.

First, create the file generator using FROM for device serialization. You must have access to the Libero IDE v8.0 or later software to complete this step.

To Configure the FROM data for serialization:

1. Generate FROM via the Catalog.
2. From the **Properties** section in the **FlashROM Settings** dialog box, select **Auto Inc** or **Read From File** region. For the **Auto Inc** region, specify the step value. You will not be able to modify this value in the FlashPoint software.
3. Complete the normal design flow and finish place and route.
4. Select **Program FlashROM** .
5. Click **Browse** to find the UFC file.
6. Check the FPGA Array box and click **Next**. The **FlashROM Settings** window appears (as shown in the figure below).

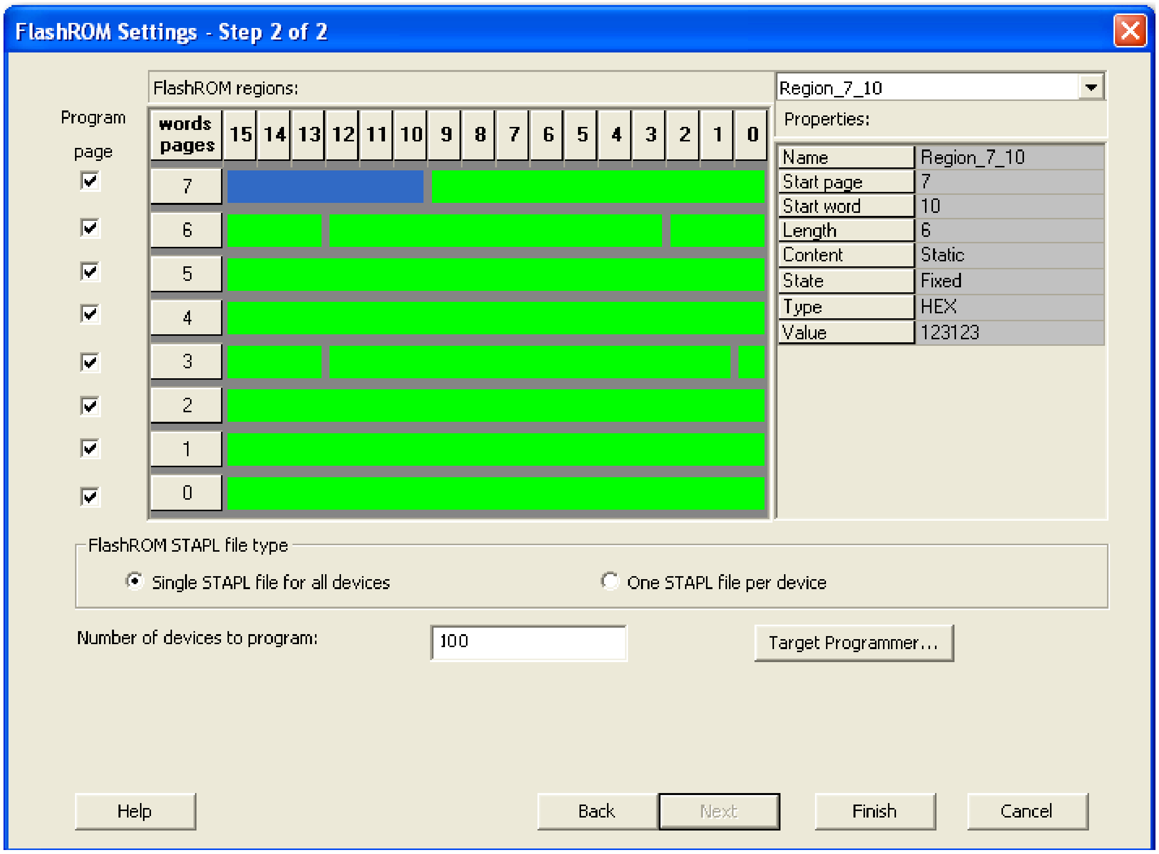


Figure 23 · FlashROM Settings- Step 2 of 2

7. Select the FROM page you want to program and data value for the configured regions.

Note: Note: The generated STAPL file contains only the data that targets the selected FROM page.

8. Modify properties for the serialization by specifying the **Start** and **Max** values. For the **Auto Inc** region, specify the **Start** and **Max** values. For the **Read From File** region, select the file name of the custom serialization file.
9. Select the FlashROM programming file type you want to generate from the two options below:
   - Choose single STAPL file for all devices: generates one programming file with all FROM values.
   - Choose one STAPL file per device: generates a separate programming file for each FROM value.
10. Enter the number of devices you want to program and generate the required programming file.
11. Click the **Finish** button.

You have completed the steps to enable device serialization. Now you are ready to program a device using Device Serialization in FlashPro.

To program a device using device serialization:

1. Click the **New Project** button in the FlashPro.
2. In the **New Project** dialog box, type Tutorial in the **Project Name** field.
3. Check the **Single STAPL** file option from the **Programming Mode** area.
4. If necessary, change the default location of your project in the **Project Location** field.
5. Click **OK**. The FlashPro GUI appears (see figure below).

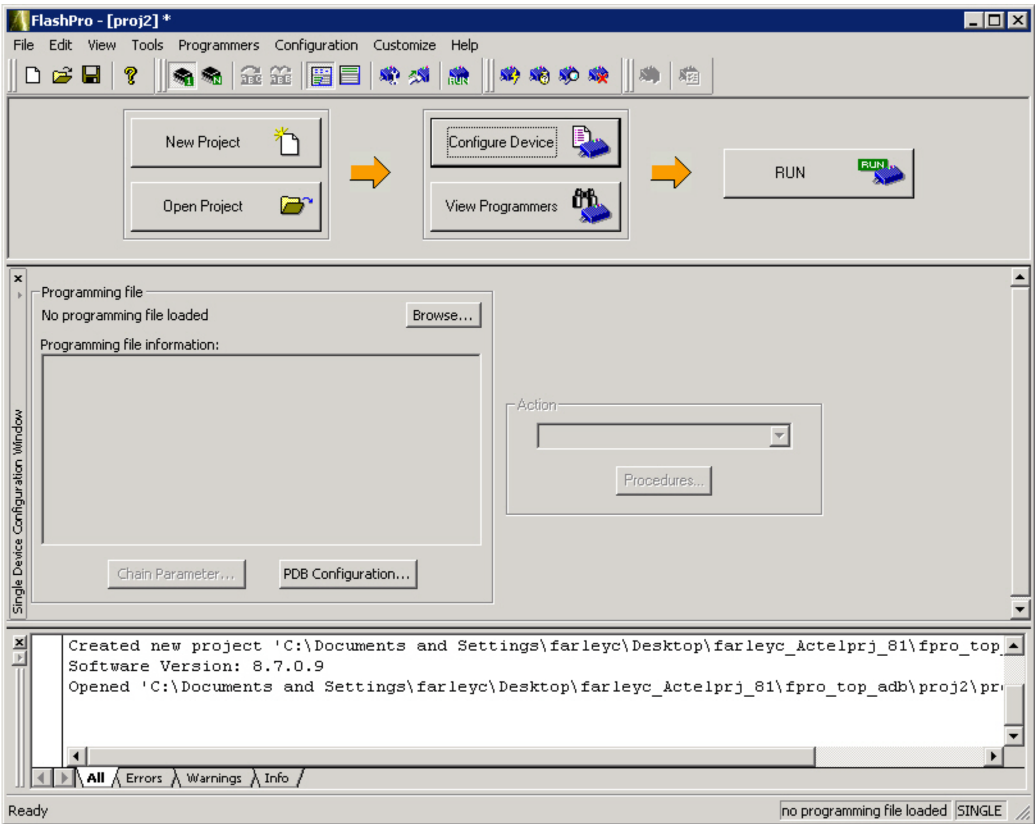


Figure 24 · FlashPro Main GUI

The **Programmer List Window** updates with your programmer information.

6. Click the **Configure STAPL File** button to load the STAPL file. The **Single STAPL Configuration Window** appears in the FlashPro GUI.
7. Click the **Browse** button to find your STAPL file.
8. From the **Load STAPL File** dialog box, find your STAPL file and click **Open**. The **Single STAPL Configuration Window** updates to list your STAPL file information and the actions available with your STAPL file in the **Action** list box (see figure below).

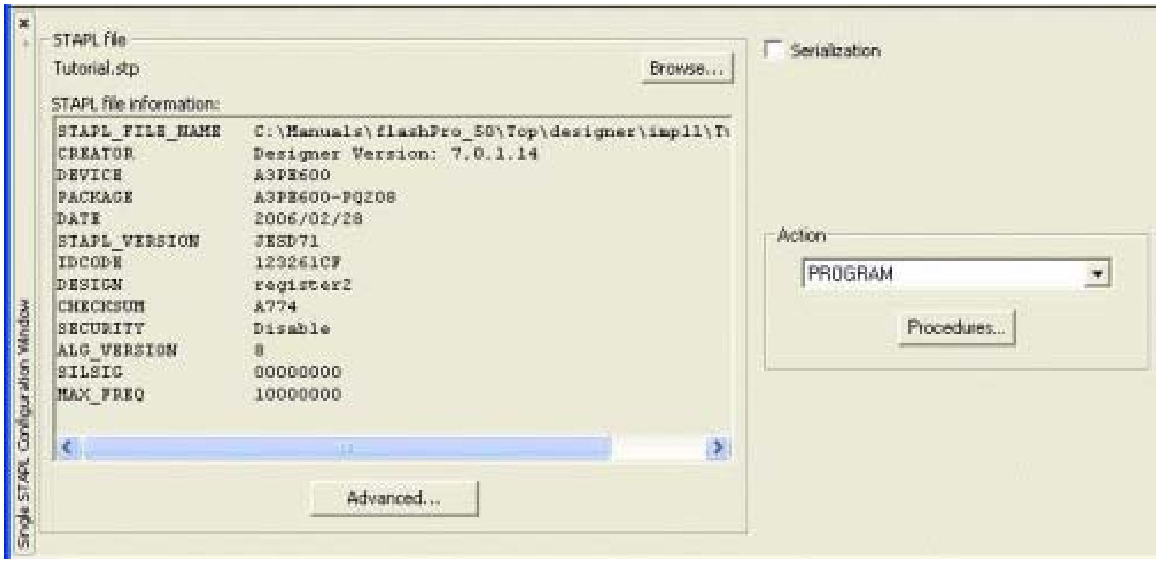


Figure 25 · Single STAPL Configuration Window with STAPL File Uploaded

9. From the **Single Device Configuration Window** in FlashPro, check the **Serialization** box and click the **Select Serialization Indexes** button.

The **Serial Settings** dialog box appears (see figure below).

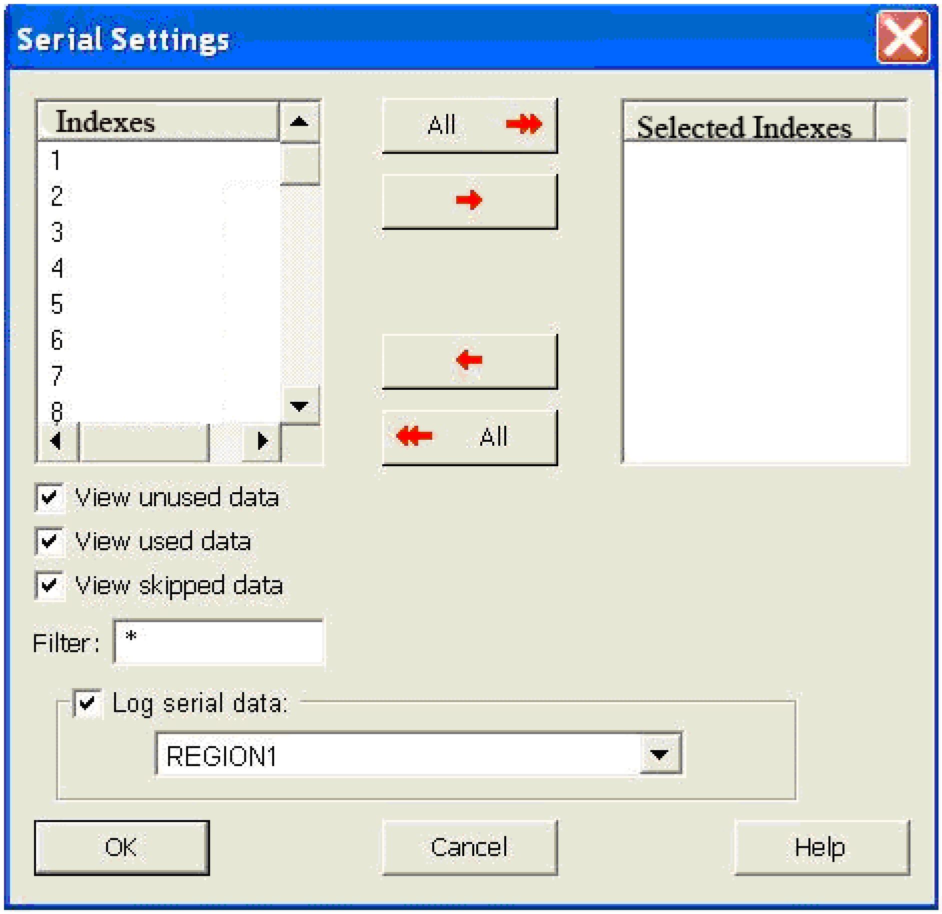


Figure 26 · Serial Settings Dialog Box

10. From the **Serial Settings** dialog box, click **All** to select all the serial data.

11. Click **OK**. The **Serialization Indexes** text box updates (see figure below).

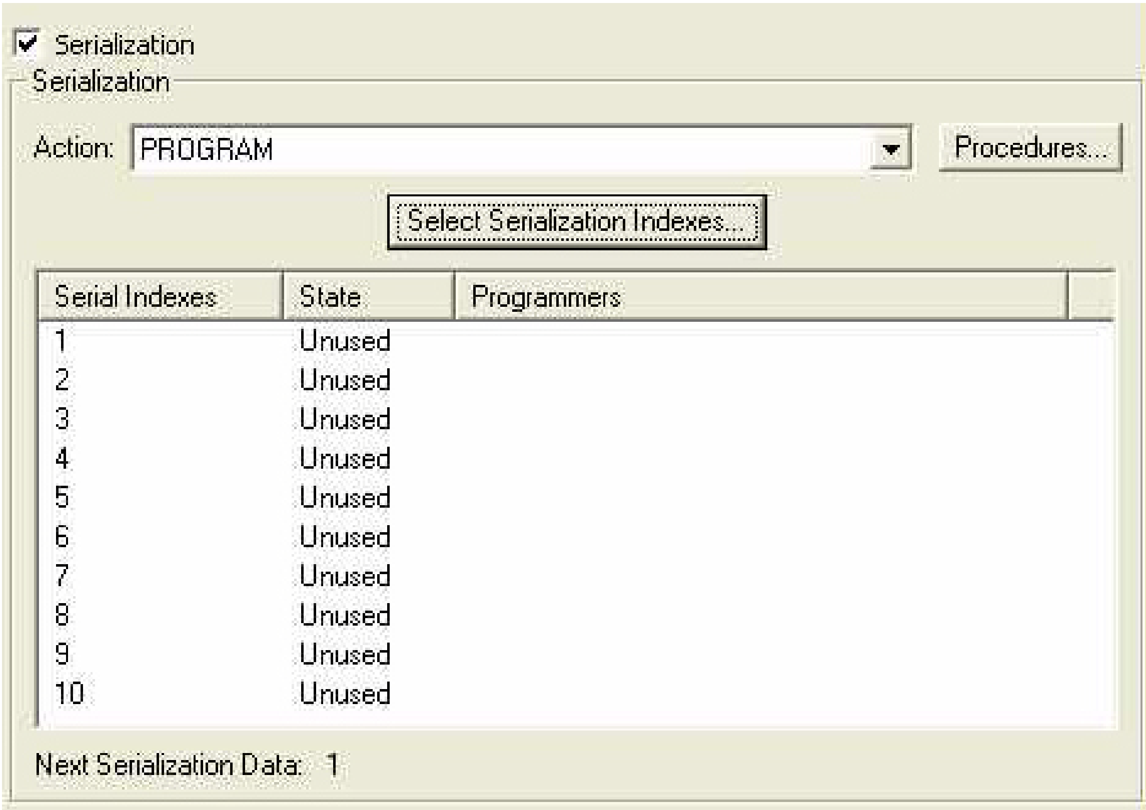


Figure 27 · Single STAPL File Configuration Window- Serialization Indexes Update

12. Click the **Program** button to program your device using serialization.

Chain Programming Tutorial

This tutorial demonstrates how to directly program an APA300 device that is part of a heterogeneous JTAG chain. The example in this tutorial uses one APA300 device and three non-Actel devices configured as shown in the figure below.

Note: This tutorial is performed in Advanced Mode. You can change your display mode to Advanced Mode from the Preferences dialog box.

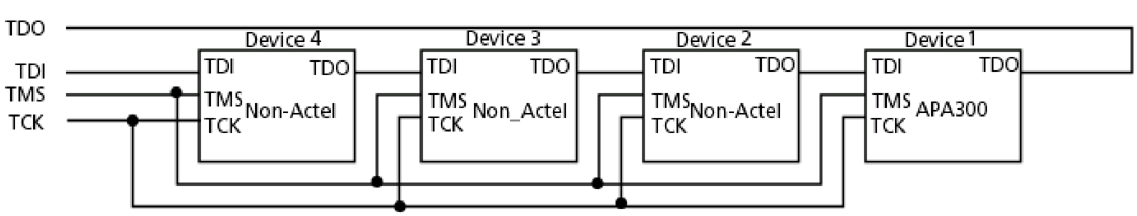


Figure 28 · APA Device Tutorial Example

First, create a new project.

To create a new project:

1. Click the **New Project** button in FlashPro.
2. In the **New Project** dialog box, type Tutorial in the Project Name field.
3. Select the **Chain** option in the **Programming Mode**.

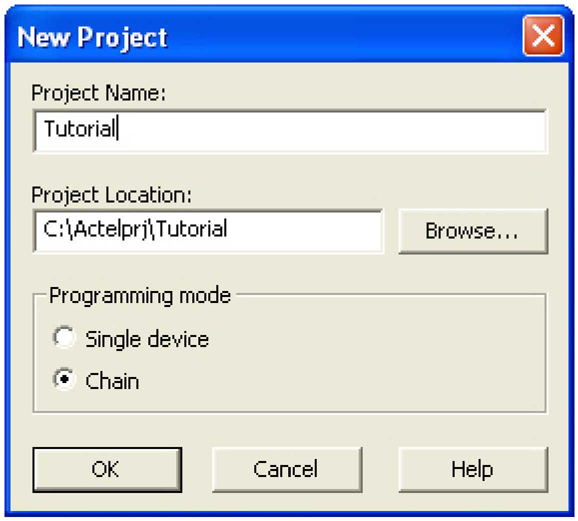


Figure 29 · New Project Dialog Box

4. If necessary, change the default location of your project in the **Project Location** field.
5. Click **OK**. The FlashPro GUI appears (see figure below).

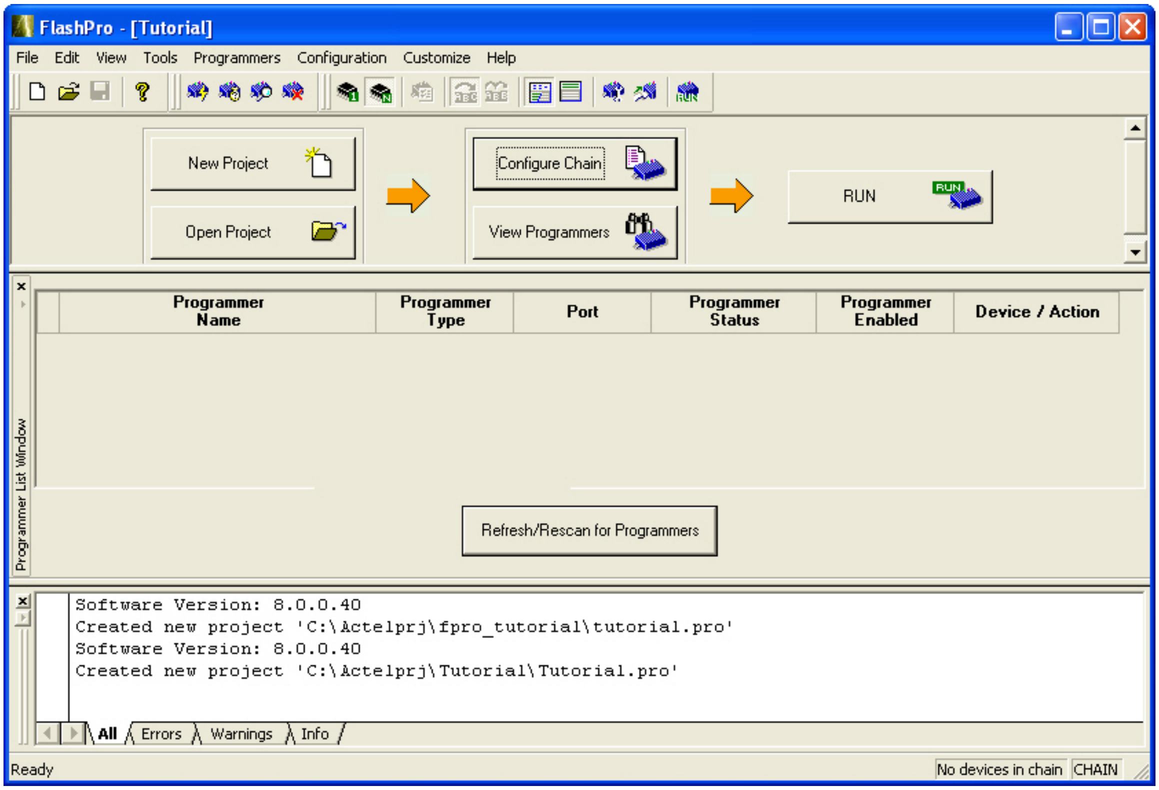


Figure 30 · FlashPro Main GUI

Note: Note: The Programmer List Window updates with your programmer information.

6. From the **Menu** bar, click **Programmers** > **Scan Chain** (or select the programmer in the **Programmer List Window**, right-click and choose **Scan Chain**).

Scan Chain shows how the devices are ordered in the chain in the **Log** window (see figure below). In this example, APA300 is the first device and will be programmed first in the chain since it is connected directly to TDO.

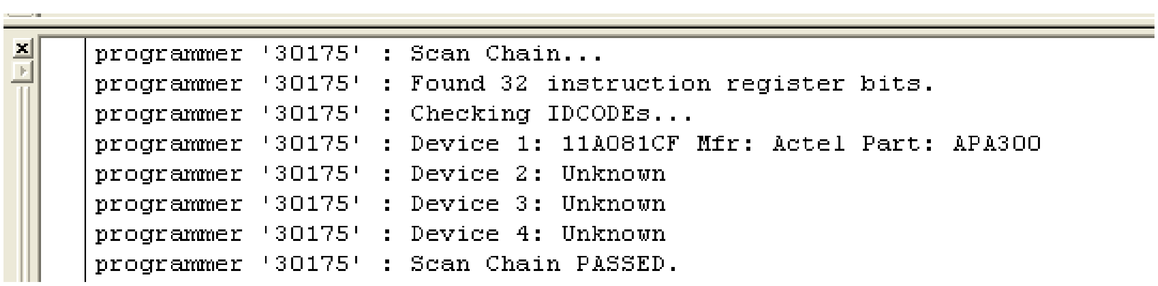


Figure 31 · Log Window Scan Chain Order

7. From the **Chain Configuration** window, click either ;Add Actel Device or Add Non-Actel Device buttons to add devices to the chain. In this example, click the **Add Actel Device** button because the APA300 is the first device in the chain.

The **Add Actel Device** dialog box displays (see figure below).

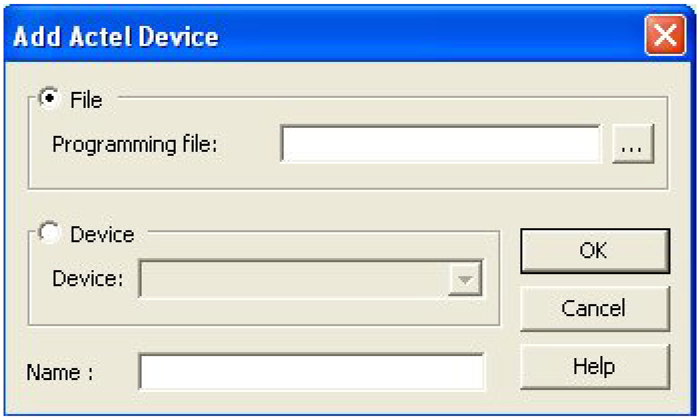


Figure 32 · Add Actel Device Dialog Box

8. Select the **File** radio button and click the **Browse** button to find your programming file.
9. Select the **Device** radio button, then choose the APA300 device from the **Device** drop-down.
10. In the **STAPL File** field, load the APA300.stp file by using the **Browse** button to locate the file.
11. In the **Name** field, keep APA300 as the default name.
12. The APA300 device is added to the **Chain Configuration** Window (see figure below).

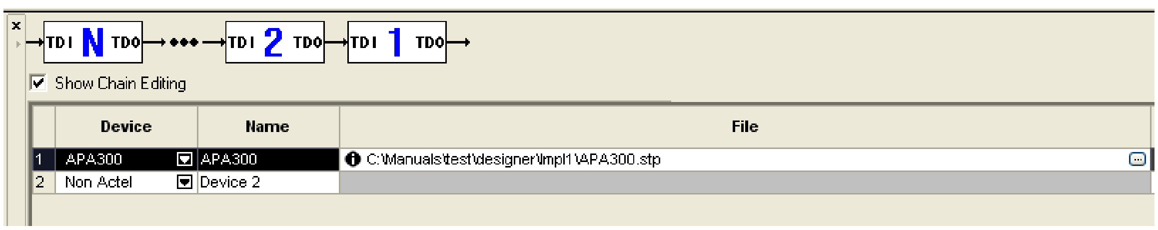


Figure 33 · Chain Configuration Window: Device One

12. Click the **Add Non-Actel Device** button to add the non-Actel device. The Add Non-Actel Device dialog box appears (see figure below). You can load the BSDL file or enter the IR length and Max TCK Frequency of the device. In this tutorial, you will enter the IR length and Max TCK frequency for this device.

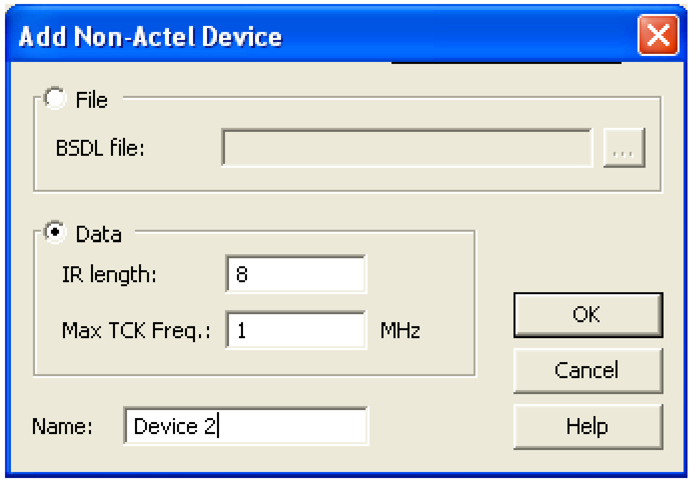


Figure 34 · Add Non-Actel Device Window

14. For this device, enter 8 in the **IR length** field and keep the **Max TCK freq** default to 1MHz.

15. Name the device, "Device 2" and click **OK**. The second device now appears in the Chain Configuration Window (as shown in the figure below).

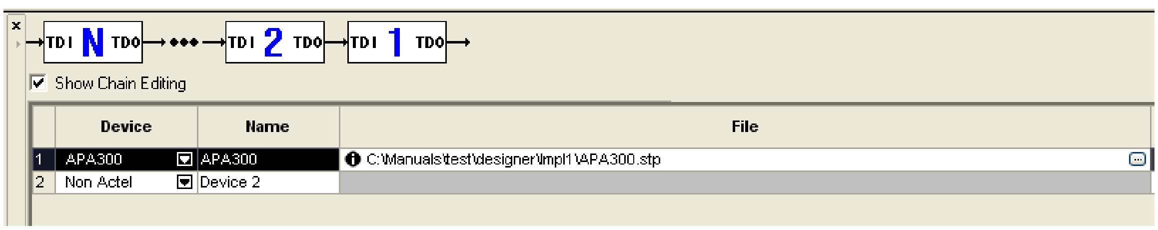


Figure 35 · Chain Configuration Window: Device Two

16. Repeat step 15 for Device 3 and Device 4.

17. Check the **Enable Device** box for the APA300 device. After you add all the devices in the chain, the **Chain Configuration Window** should look like the figure below.

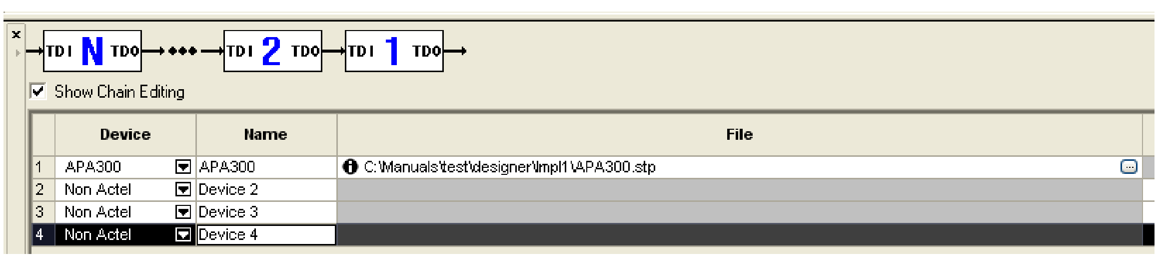


Figure 36 · Chain Configuration Window: All Devices in the Chain

18. After you have added all of the devices to the chain in the correct order, click the **Run** button to program the chain.

19. When programming is complete, the results are listed in the Log window (see figure below).

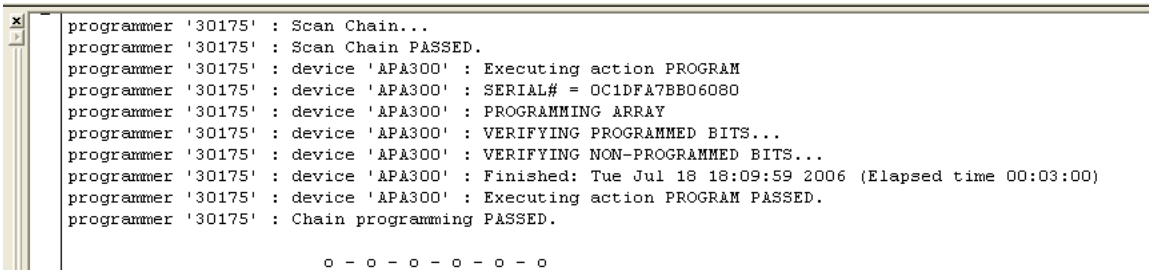


Figure 37 · Programmer List Window: Programming Complete

SmartFusion Programming Tutorial

You can program your SmartFusion device without using the Libero IDE by using an EFC or UFC file from standalone SmartDesign, or using an FDB file from standalone Designer.

To program a SmartFusion device without using the Libero IDE:

1. Start FlashPro and click **New Project** to create a new project. Specify your **Project Name**, **Project Location** and **Programming Mode**.
2. Click **Configure Device**.
3. **Single Mode**: Click the **Create** button to create your new PDB programming file. The create PDB dialog box appears (as shown in the figure below).

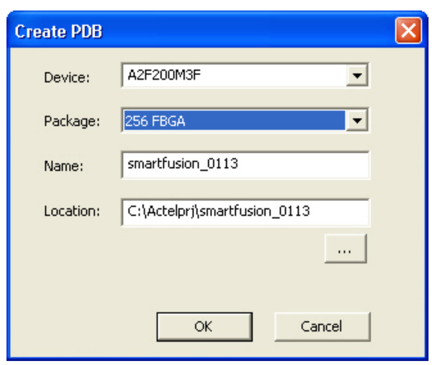


Figure 38 · Create PDB Dialog Box

Chain Mode: Click **Add Actel Device** and choose a SmartFusion device from the drop-down menu. Click the **Create PDB** button in the Chain Configuration Window. The Create PDB dialog box appears.

4. Specify your PDB parameters. Click **OK** to continue. The FlashPoint SmartFusion Programming File dialog box appears.
5. Specify your security settings and select which silicon features you want to program. Click the **Import** button for your FPGA Array, FlashROM and Embedded Flash Memory files to add them to your PDB file.

   You must have a FDB file to program your FPGA Array, a UFC file to program your FlashROM, and a EFC file to program your Embedded Flash Memory.

Click the **Modify** buttons if you wish to modify your FlashROM or Embedded Flash Memory files before you save your PDB file.

6. (Optional) Specify your I/O States During Programming.
7. Click **Save PDB** to save your new PDB file.

If you make changes to your Security, I/O States During Programming, EFC, UFC or FDB file, click **Modify** in FlashPro to open and re-save your PDB with the updated files and settings.

See Reprogramming a Secured Device for information on programming a secured SmartFusion device.

Modifying Memory Contents and Programming a Device Tutorial

This tutorial provides step-by-step instructions on how to load a Program Database (PDB) file, modify the memory contents, and program the device.

Before you begin this tutorial, you should have a design with an EFMB client in it with a generated programming file for this design. You will first create a new project and title it "tutorial." If FlashPro is launched through Libero IDE, a new project will automatically be created and a PDB file will be loaded, if available.

Creating a new project

If you are familiar with this feature, follow the basic procedures for creating a new project. However, if you would like step-by-step instructions, see the creating a new project section in the Single STAPL/PDB File Basic Tutorial.

Loading and Configuring a PDB File

Once you have created your project and connected your programmer, you are ready to load your PDB file.

To load a PDB file:

1. Click the **Configure Device** button. The **Single PDB Configuration** window appears in FlashPro.
2. Click the **Browse** button to find your PDB file.
3. From the **Load PDB File** dialog box, find your PDB file and click **Open**.

Modify Embedded Flash Memory Block Content

Now, you are ready to modify the Embedded Flash Memory Block content.

To modify Embedded Flash Memory Block content:

1. Click the PDB Configuration button to open FlashPoint.

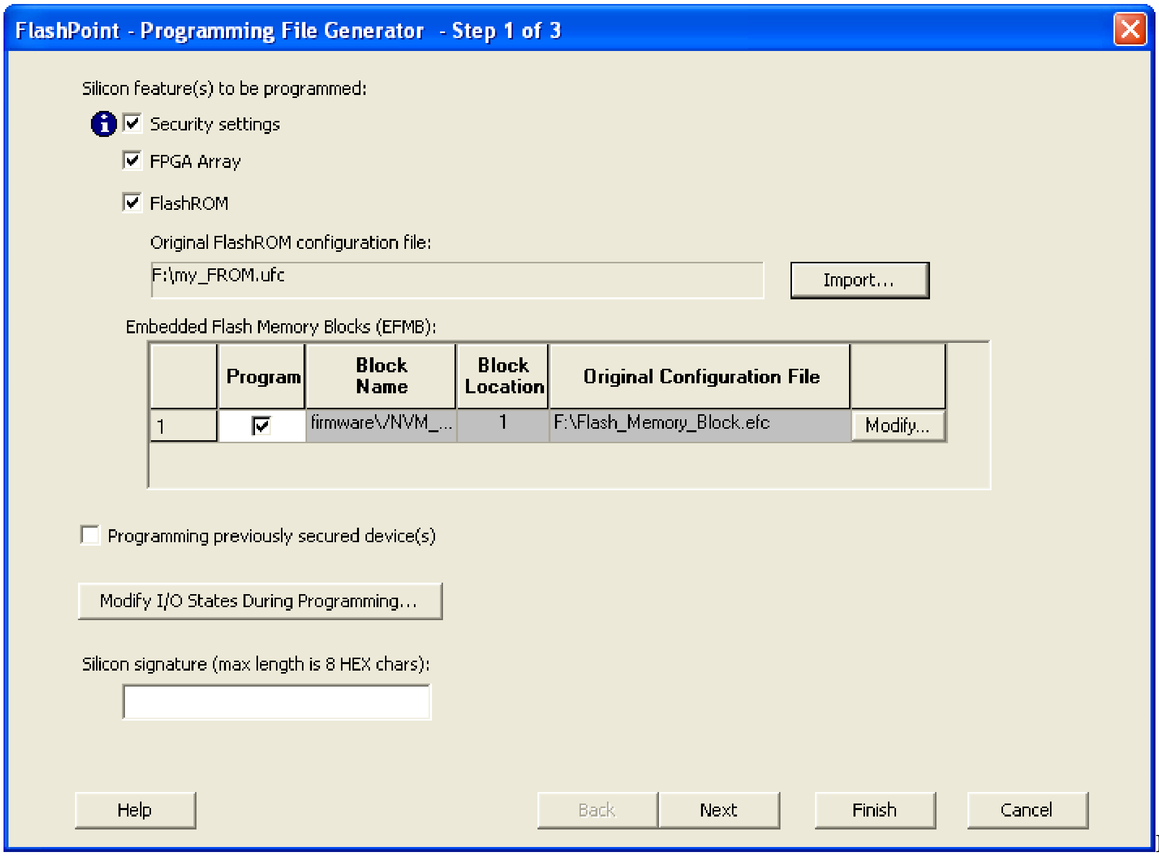


Figure 39 · Program File Generator

2. Check the **Program** box.

3. Click the **Modify** button to import Embedded Flash Memory Block configuration and memory content file. The **Modify Embedded Flash Memory Block** dialog box appears

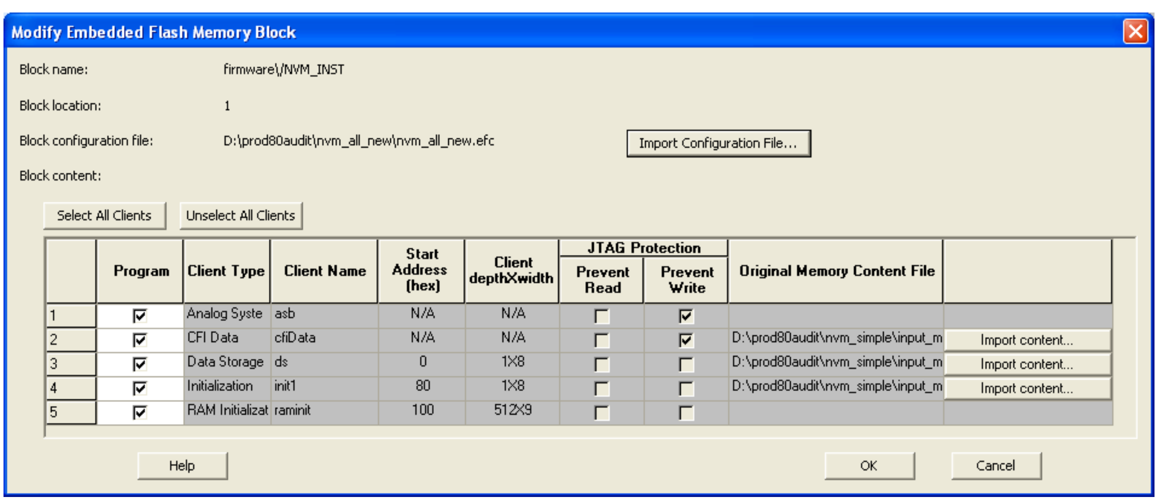


Figure 40 · Modify Embedded Flash Memory Block Content Dialog Box

4. Click the **Import Configuration File** button to import the Embedded Flash Memory Block configuration and memory content from the EFC file. This will populate the client table below. All clients that belong to this block will be selected by default.
5. Click the **Import content** button if you want to change the client memory content.
6. Click **OK**.
7. Click **Finish**.

Note: FlashPoint audits original configuration and memory content files and warns you if the files cannot be located or if they have been updated. These files are not required as the last updated configuration and memory content is stored in the PDB.

Figure 41 · Audit Warning

Proceed to program the device. For steps on how to program a device, see the **Programming a device** section of the **Single STAPL/PDB file basic tutorial**.

Modifying FlashROM Contents and Programming a Device Tutorial

This tutorial provides step-by-step instructions on how to load a Program Database (PDB) file, modify the memory contents, and program the device.

Before you begin this tutorial, you should have a design with an EFMB client in it with a generated programming file for this design. You will first create a new project and title it "tutorial." If FlashPro is launched through the Libero IDE Project Manager, a new project will automatically be created and a PDB file will be loaded, if available.

Creating a new project

If you are familiar with this feature, follow the basic procedures for creating a new project. However, if you would like step-by-step instructions, see the creating a new project section in the Single STAPL/PDB File Basic Tutorial.

Loading and Configuring a PDB File

Once you have created your project and connected your programmer, you are ready to load your PDB file.

To load a PDB file:

1. Click the **Configure Device** button. The **Single Device Configuration** Window displays in FlashPro (see figure below).

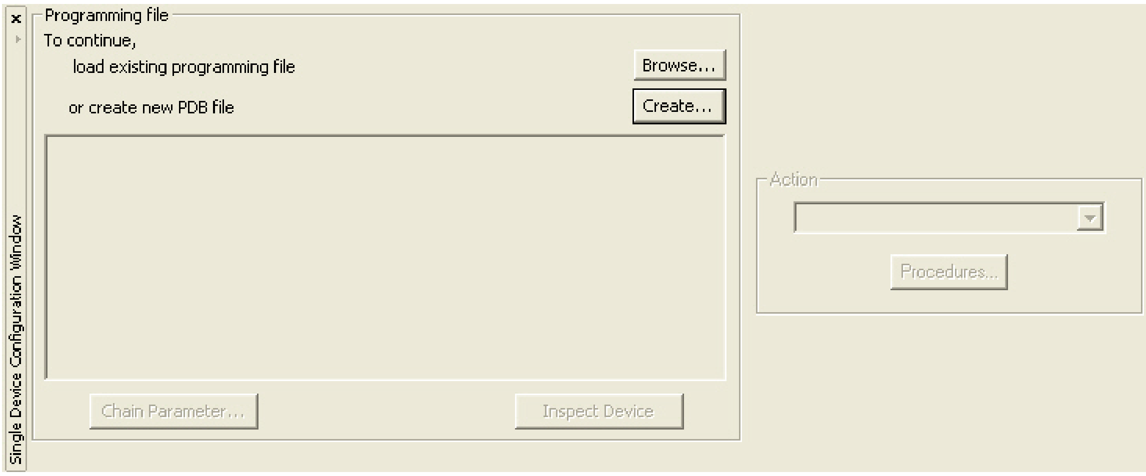


Figure 42 · Single Device Configuration Window

2. Click the **Browse** button to find your PDB file. From the **Load Programming File dialog box**, find your PDB file and click **Open.** .

Modify FlashROM Content

Now you are ready to modify the FlashROM content.

1. Click the **PDB Configuration** button. This opens FlashPoint.
2. Select **FlashROM** under Silicon feature(s) to be programmed (see figure below).

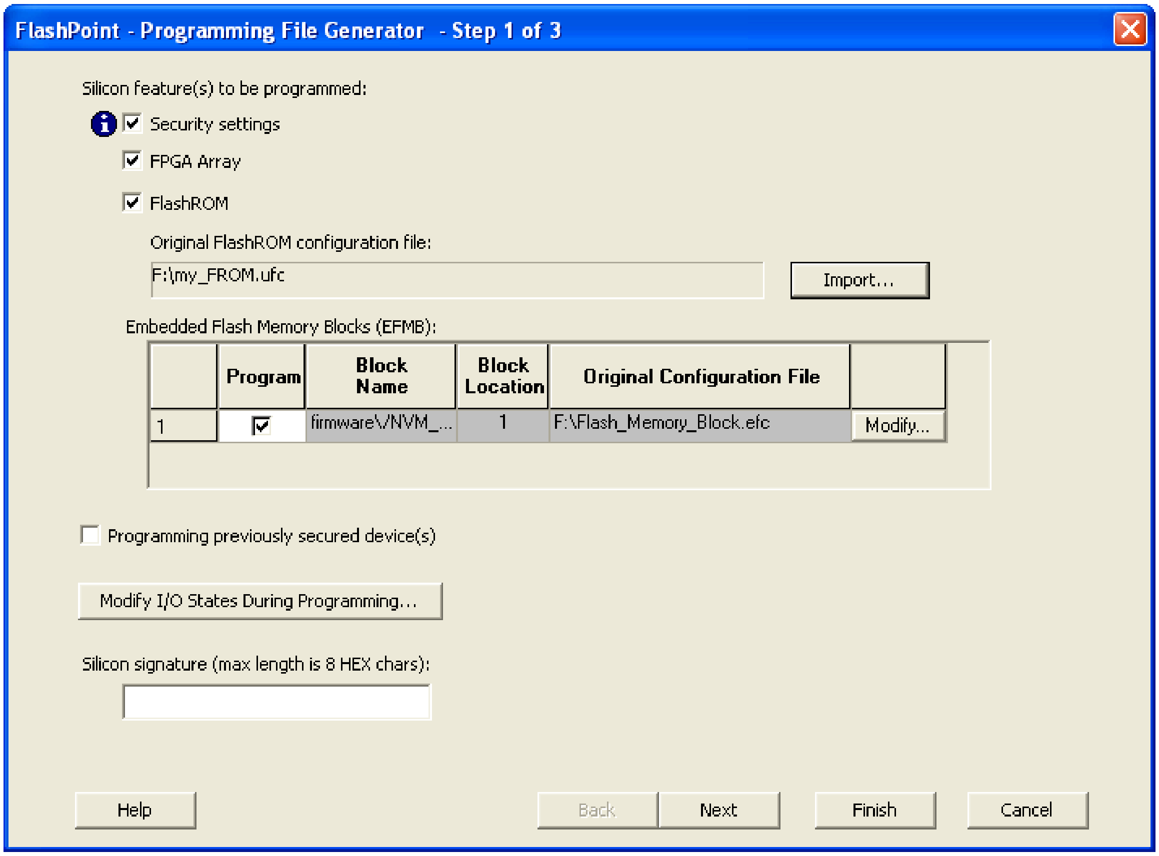


Figure 43 · FlashPoint Programming File Generator

3. Click the **Browse** button to select the *.ufc FlashROM configuration file by and navigating to the configuration file. This file is normally present in the SmartGen subfolder of the Libero IDE project, in a folder with the FlashROM IP block's name.
4. Click **Next**.
5. Select the FlashROM pages you want to program (see figure below).

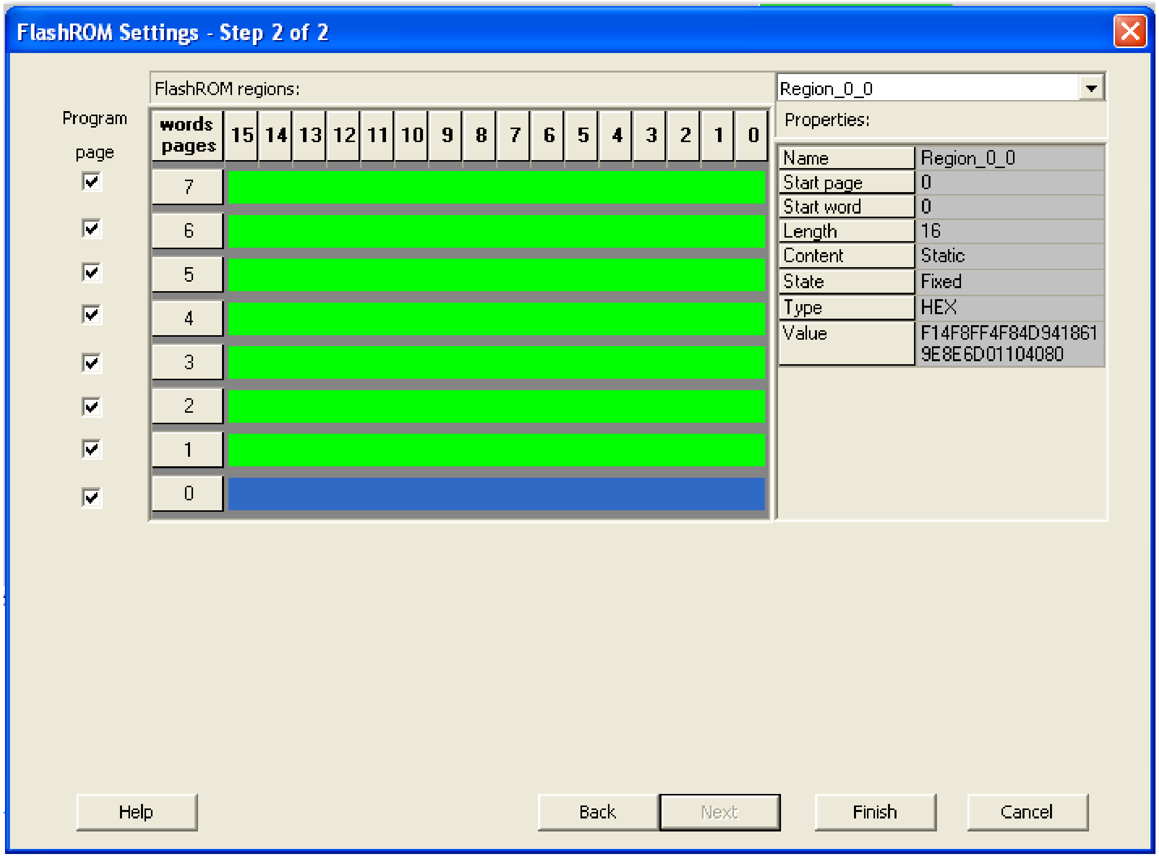


Figure 44 · FlashROM Settings Dialog Box

6. Click **Finish**.

Proceed to program the device. For steps on how to program a device, see the **Programming a device** section of the **Single STAPL/PDB file basic tutorial**.

Programming Only Security Settings Tutorial

This tutorial provides step-by-step instructions on how to program only the security settings into a device.

No design or PDB file is needed to follow this tutorial.

First create a new project and name it tutorial. If FlashPro is launched from the Project Manager, a new project will automatically be created and a PDB file will be loaded, if available. For this tutorial you always need to create a new project.

Creating a New Project

If you are familiar with this feature, follow the basic procedures for creating a new project. However, if you would like step-by-step instructions, see the creating a new project section in the Single STAPL/PDB file basic tutorial.

Configuring the Security Settings

Once you have created your project and connected your programmer, you are ready to load your PDB file.

To configure the security settings:

1. Click the **Configure Device** button. The **Single Device Configuration window** appears in FlashPro (see figure below).

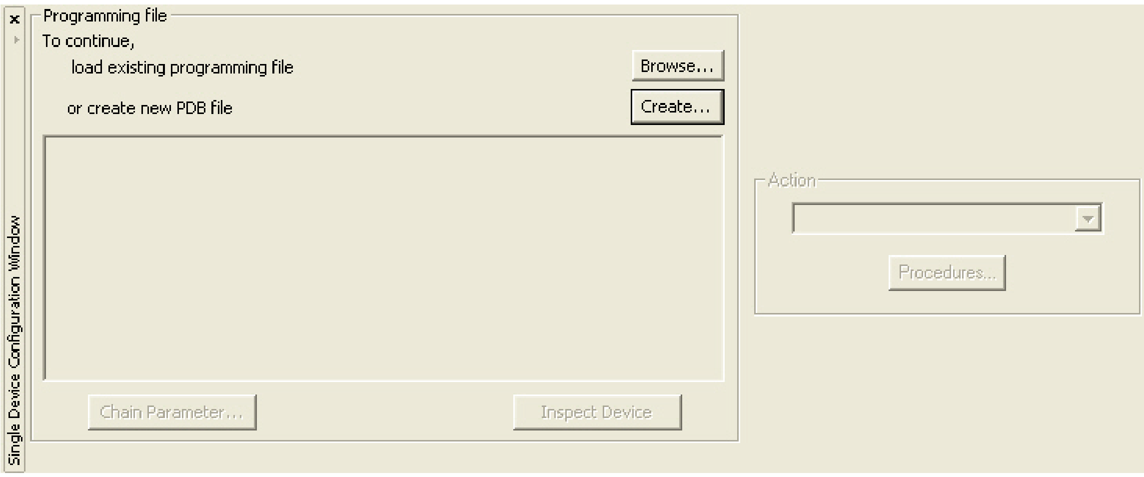


Figure 45 · Single Device Configuration Window

2. Click **Create**. This opens the Create PDB dialog box, as shown in the figure below.

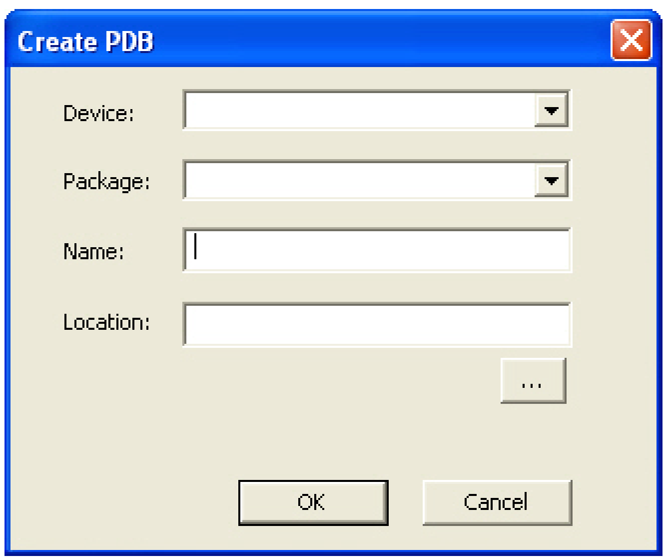


Figure 46 · Create PDB Dialog Box

3. Select the desired device and package (if available) from the drop down list, and specify the filename and location. Click **OK**. FlashPoint opens. IGLOO, ProASIC3, SmartFusion and Fusion family devices support securing the device with a pass key as well as encrypting programming files using an AES key. Flash devices can also be permanently locked, preventing reprogramming.

4. Check the **Security Settings** checkbox to secure the unsecured device.

   Warning: Make a note of the security keys that you are using. Once a device is secured, it cannot be reprogrammed without those keys.

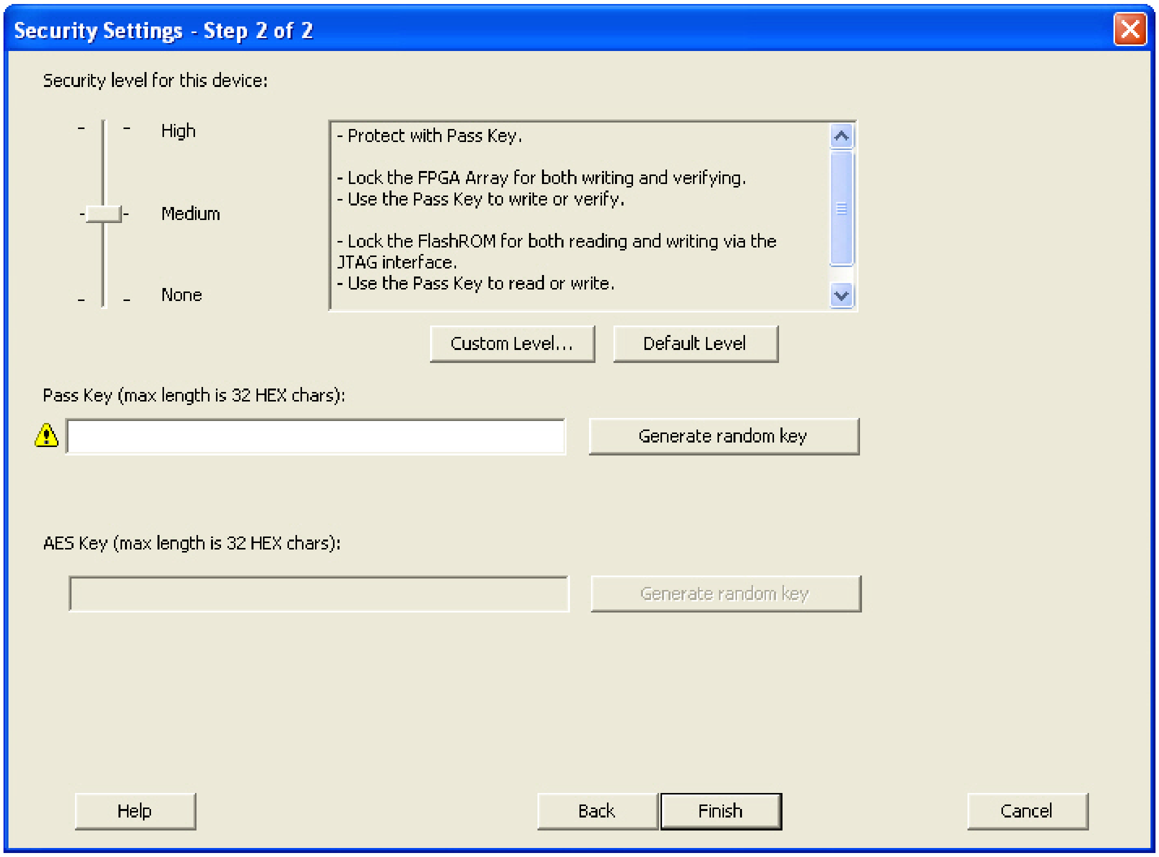


Figure 47 · Security Settings Dialog Box

5. Click **Finish**.

Proceed to program the device. For steps on how to program a device, see the Programming a device section of the Single STAPL/PDB file basic tutorial.

Automatic Chain Construction Tutorial

This tutorial demonstrates how to automatically scan a chain of devices and construct the chain within FlashPro. Automatic chain construction saves the effort of manually adding each device to your chain.

The software also scans the chain before constructing it, which reduces the possibilities of having errors in the chain. This feature is fully automated if your chain is composed of only Actel devices. If you have non-Actel devices in your chain, you can still use the Auto Chain Construction feature. However, you will be required to either manually add the BSDL file or enter the IR length and max TCK for each non-Actel device. This tutorial goes through the flow for an Actel-only chain first, followed by instructions on adding Non-Actel devices to the database.

Note: This tutorial requires that your chain is connected to the computer you are using, via an Actel programmer, and that you have suitable programming files to program the devices in your chain.

To automatically scan a chain of devices and construct the chain:

1. Start a new project in FlashPro. Select **Chain** as the Programming Mode.
2. Click the **Configure Chain** button in FlashPro.
3. From the **Configuration** menu, choose **Construct Chain Automatically**; or click the **Construct the chain from a Scan Chain operation** link in the Chain Configuration Window, see below.

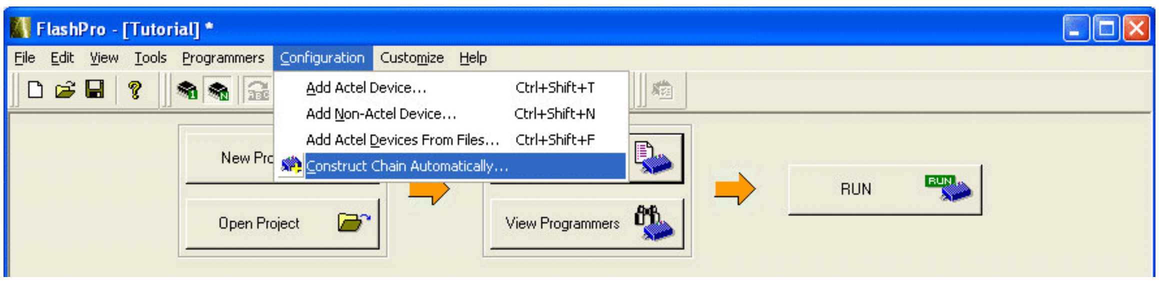


Figure 48 · Construct Chain Automatically

4. A popup appears asking you to select the programmer you would like to use from the ones attached to your computer. Choose the appropriate programmer (as shown in the figure below) and click **OK**.

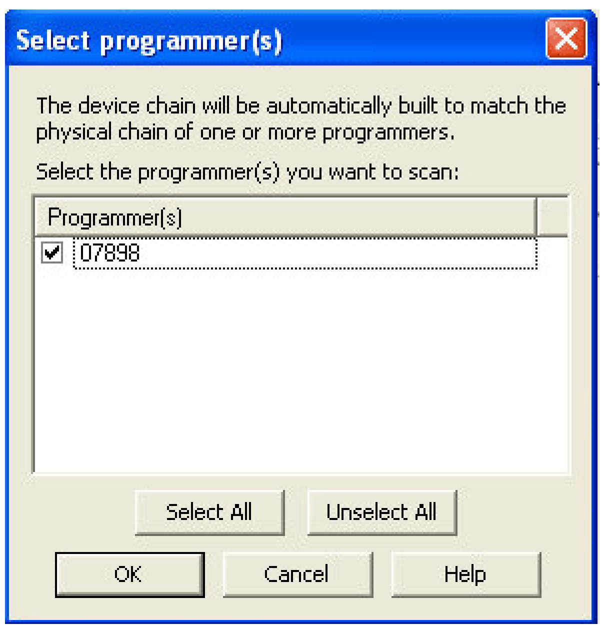


Figure 49 · Select Programmer Popup

Automatic chain construction starts. The Log window documents the detection and verification of all devices in your chain. The devices are added to the chain in the Chain Configuration Window; see figure below for an example.

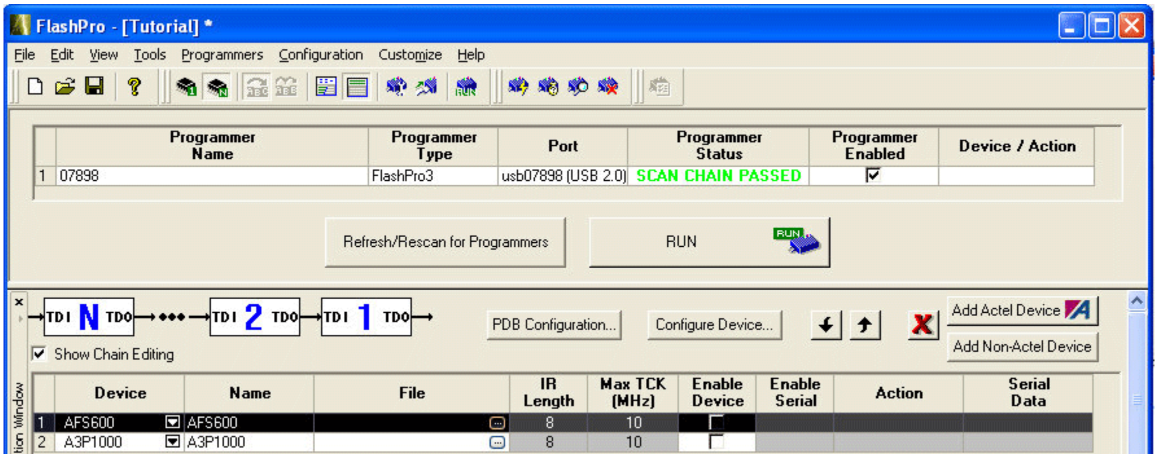


Figure 50 · Scan Chain Configuration Passed

In some cases, FlashPro is not able to uniquely identify the device due to shared IDCODEs, and lists all possible devices (ex: AGL030V2/AGL030V5). Once a programming file is loaded for that device, the device field only shows one device, since the programming file will only be targeted to one device.

Adding Non-Actel Devices to the Chain

FlashPro recognizes non-Actel devices in the chain, but it does not contain any device information, such as IR length or Max TCK. The figure below lists Actel devices and non-Actel devices in the chain.

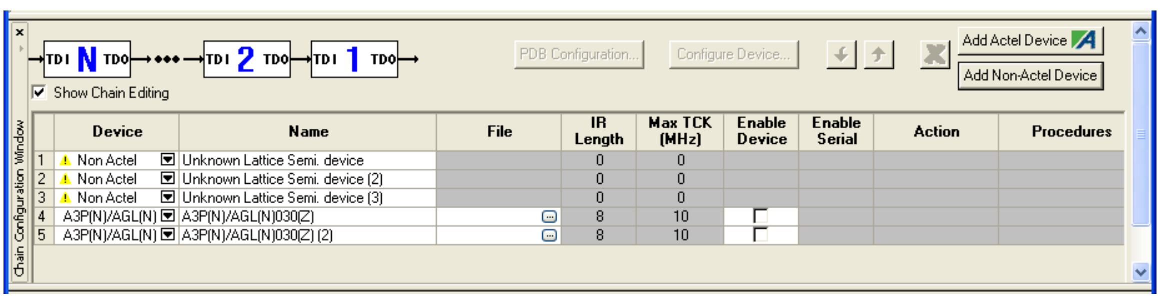


Figure 51 · Non-Actel Devices in a Chain

You must import the BSDL file into Actel's non-Actel device database for FlashPro to recognize your non-Actel device.

To import the device BSDL into the FlashPro non-Actel device database and run the scan chain:

1. From the **Tools** menu, choose **Import Settings for Non-Actel Devices**. This opens the Import Settings for Non-Actel Devices dialog box. This dialog box enables you to import and remove BSDL files from the database and lists all the device information contained in the BSDL file.

2. Click the **Import BSDL Files** button and navigate to the folder that contains your BSDL files. Select the file and click **OK.** Once the BSDL is imported into the database, the original BSDL file is no longer audited by FlashPro. If changes are made to the original source BSDL file, it will not affect the BSDL file that has been imported into the non-Actel device database.

Remove BSDL files from the database by selecting the file and clicking the **Remove** button.

3. Once you have the appropriate BSDL files loaded to the database, you can construct the chain. To do so, from the **Configuration** menu, choose **Construct Chain Automatically** and select the appropriate programmer from the dialog box. FlashPro runs a scan chain, detects the devices in the chain, and associates them with the BSDL files in the database, as shown in the figure below.

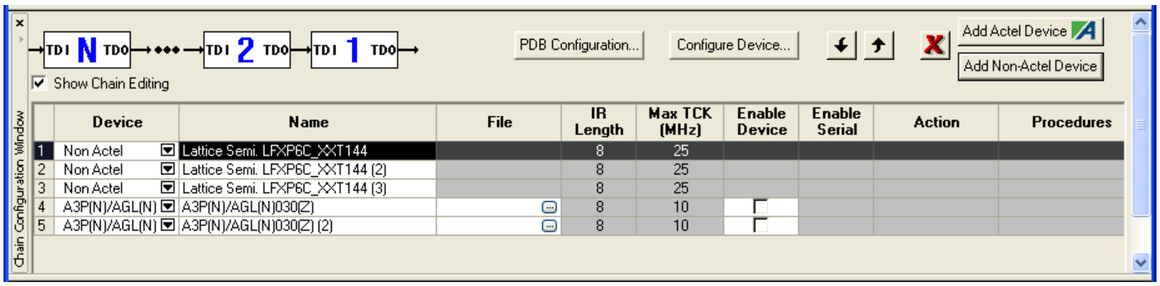


Figure 52 · Non-Actel Devices in the Chain with Associated BSDL Files

It is possible to add multiple BSDL files to your Non-Actel device database that have the same IDCODE. If the BSDL files list the same IR length but different TCK values, FlashPro automatically chooses the file with the lowest TCK value by default and no action is required. If the IR lengths are different you receive an error message asking you to resolve the conflict.

To resolve the issue, click the drop-down arrow adjacent to the device name. This opens the Non-Actel Device Configuration dialog box (as shown in the figure below). From here you can choose the device that you wish to use. Select the device from the dropdown menu and enter a new name or use the default.

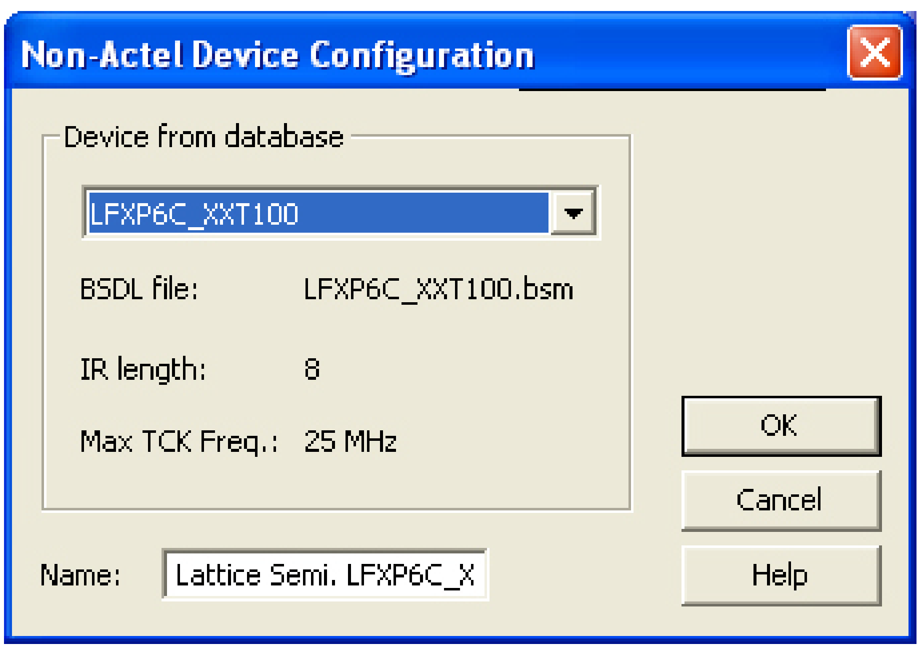


Figure 53 · Non-Actel Device Configuration Dialog Box

For a tutorial on manually adding both Actel and non-Actel devices to your chain as well as programming the chain, refer to the Chain Programming Tutorial.

See Also

Understanding the Chain Configuration Window

Import Settings for Non-Actel Devices

Fusion Calibration Backup and Recovery Tutorial

This tutorial provides step-by-step instructions on how to backup and recover default calibration data on a Fusion device. It assumes that you have created a new project, connected your programmer, and loaded a Fusion PDB/STAPL file created in Designer v8.4 or above.

If you would like step-by-step instructions on how to create a new project, see the Creating a New Project section in the FlashPro Single STAPL Basic Tutorial..

If you would like step-by-step instructions on loading a programming file, see the Loading and Configuring a Programming File section in the FlashPro Single STAPL Basic Tutorial.

Note: This feature is only supported in STAPL and PDB programming files.

Backing Up Default Fusion Calibration Data

A backup copy of the Fusion calibration data is created once after ANY programming ACTION, except READ_IDCODE, is executed. The copy will be stored in the spare pages of eNVM. The FlashPro Log window shows that a backup copy of the calibration data has been created (as shown in the figure below).

```
programmer '07898' : Scan Chain...
programmer '07898' : Scan Chain PASSED.
programmer '07898' : Executing action PROGRAM
programmer '07898' : Checking for Backup Calibration Data...
programmer '07898' : Reading Master Calibration Data...
programmer '07898' : Writing Calibration Backup Copy
programmer '07898' : Erase ...
programmer '07898' : Completed erase
programmer '07898' : Programming FPGA Array
```

Figure 54 · FlashPro Log Window

Recovering Default Fusion Calibration Data

1. Load the PDB/STAPL file created in Designer v8.4 or above.
2. In the FlashPro Configuration window, click **Advanced** and select **RECOVER_CALIB** (as shown in the figure below).

Figure 55 · RECOVER_CALIB

3. Click **Run** to restore the original Fusion calibration data. The Log window shows the data is restored (as shown in the figure below).

```
programmer '07898' : Scan Chain...
programmer '07898' : Scan Chain PASSED.
programmer '07898' : Executing action RECOVER_CALIB
programmer '07898' : Checking for Backup Calibration Data...
programmer '07898' : Reading Master Calibration Data...
programmer '07898' : Writing Calibration Backup Copy
programmer '07898' : Checking for Backup Calibration Data...
programmer '07898' : Restoring Master Calibration Data.
```

Figure 56 · Restoring Original Calibration Data

Note: The Calibration data can only be restored after a backup has been made.

Specify I/O States During Programming Tutorial

This tutorial explains how to modify the I/O states during programming within FlashPro for used and unused I/Os. It also explains how to modify the Boundary Scan Registers (BSRs) for each I/O to allow for more detailed customization of the I/O states during programming. Finally, it shows how to save and load these settings with a file.

Note: This tutorial requires a design with a valid *.pdb file associated with it. If you launch FlashPro from a Libero IDE project, a FlashPro project is created automatically and the PDB file loaded. Otherwise, you can start a new FlashPro project and load a PDB file; refer to Single STAPL/PDB file basic tutorial for more information.

You can also modify the individual Boundary Scan Registers; see the Modify Boundary Scan Registers section for more information.

To modify the state of an I/O during programming:

1. Once your PDB is successfully loaded, from the **Configuration** menu, choose **PDB Configuration**. This brings up FlashPoint . FlashPoint is the tool that allows you to modify the PDB programming file from within FlashPro.

2. In FlashPoint, click the **Specify I/O States During Programming** button. The Specify I/O States During Programming dialog box appears (as shown below). This dialog box enables you to modify the I/O states during programming for all used and unused I/Os.

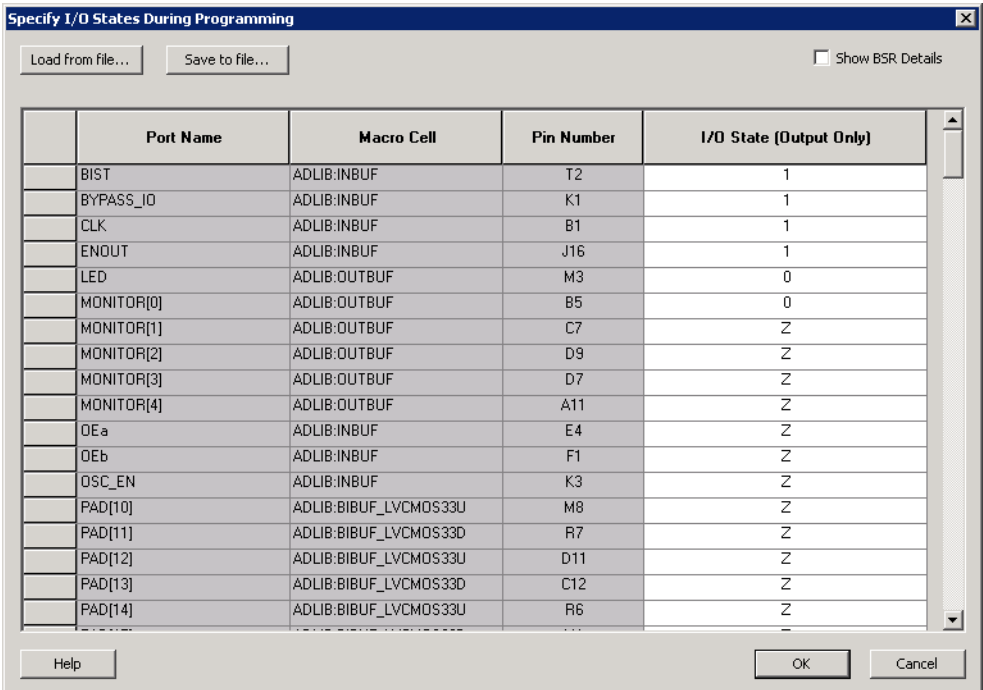


Figure 57 · I/O States During Programming Window

The default view displays a grid with 4 columns: Port Name, Macro Cell, Pin Number, and I/O State (Output Only). Port Name lists the port associated with each of theses pins, if the pin is not used in the design, the Port Name for this pin reads Unused. The Pin Number column contains a list of all the pins for the package associated with the design open in FlashPro.The Macro Cell column contains the Actel macro associated with each pin, as with Port Names, if the pin is not used, the Macro Cell for this pin reads Unused. The I/O State column is the only column editable in FlashPro.

3. Select an I/O State from the drop-down menu for each I/O you want to modify.

Please refer to Specifying I/O States During Programming for information on sorting and selecting multiple entries in the grid.

4. Click **Save** in the Specify I/O States During Programming window, then **Finish** in FlashPoint to return to FlashPro. The PDB is updated with your new settings.

Congratulations, you have successfully modified the I/O states that will be held during programming.

Modifying Boundary Scan Registers

Each I/O in your device is comprised of an Input, Output and Output Enable Boundary Scan Register (BSR) cell.

The BSR cells enable you to define I/O states during programming and control the individual states for each Input, Output, and Output Enable register.

To modify the individual Boundary Scan Registers of an I/O in your device:

1. Select the **Show BSR Details** checkbox in the **Specify I/O States During Programming** window. This replaces the I/O State (Output Only) column with a Boundary Scan Registers column that is split into Input, Output Enable and Output.

2. Modify each of the registers for any I/O to set your custom options. See the Specifying I/O States During Programming - I/O States and BSR Details help topic for an explanation of the individual BSR settings.
3. (Optional) Uncheck the Show BSR Details checkbox to return to the default view.

Note: Note: Updated I/Os with non-default settings are displayed as *User-Defined BSR* in the default view.

Click OK and complete programming to save your updated settings to the ADB and programming files.

Saving and Loading I/O State Settings

Click Save to File to save your changes. This enables you to save your custom I/O settings in an IOS file.

Click Load from File to load a previously saved *.ios file.

You must click OK and complete programming to save your updated settings to the ADB and programming files.

See Also

Specifying I/O States During Programming

Specifying I/O States During Programming - I/O States and BSR Details

Advanced Tutorials

Multiple Device Chain Programming

This tutorial provides step-by-step instructions on how to program multiple Actel devices in a chain. You should already be familiar with the basic features of the FlashPro software.

Note: This tutorial does not provide software installation instructions. Please have FlashPro already installed before you begin.

In the figure below, there are three devices in a chain (two A3P250 and one A3PE600). In this section, we will program these three devices in the chain.

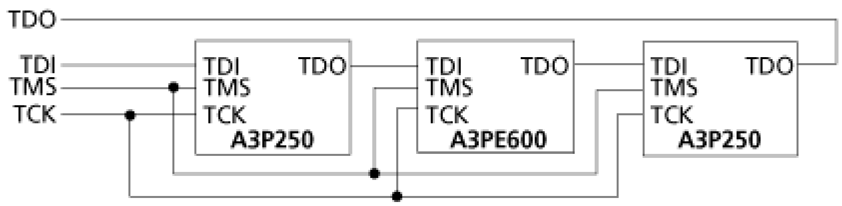


Figure 58 · APA Device Tutorial Example

First you need to create a project.

To create a new project:

1. Click the **New Project** button in FlashPro.
2. From the **New Project** dialog box, type "Tutorial" in the Project Name field.
3. Check the **Chain** box (see figure below).

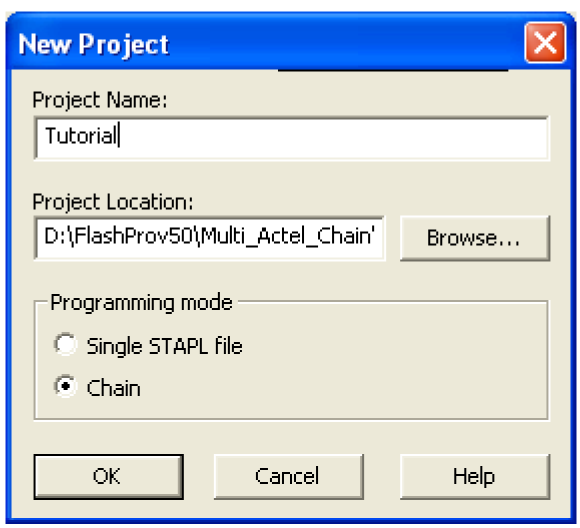


Figure 59 · New Project Dialog Box

4. If necessary, change the default location of your project in the **Project Location** field.

5. Click **OK**. The FlashPro main window appears.

Note: Note: The Programmer List window updates with your programmers information.

6. From the **Programmers** menu, choose **Scan Chain** (or select the programmer in the **Programmer List** window, right-click, then choose **Scan Chain)** (see figure below).

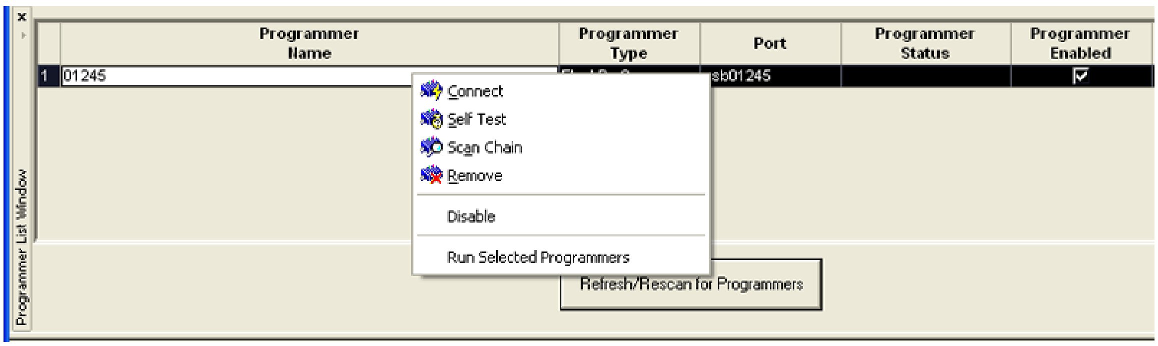


Figure 60 · Select Programmer Window

Scan Chain shows how the devices are ordered in the chain in the log window (see figure below). In this case, A3P250 is the first device that will be programmed in the chain since it is connected directly to TDO.

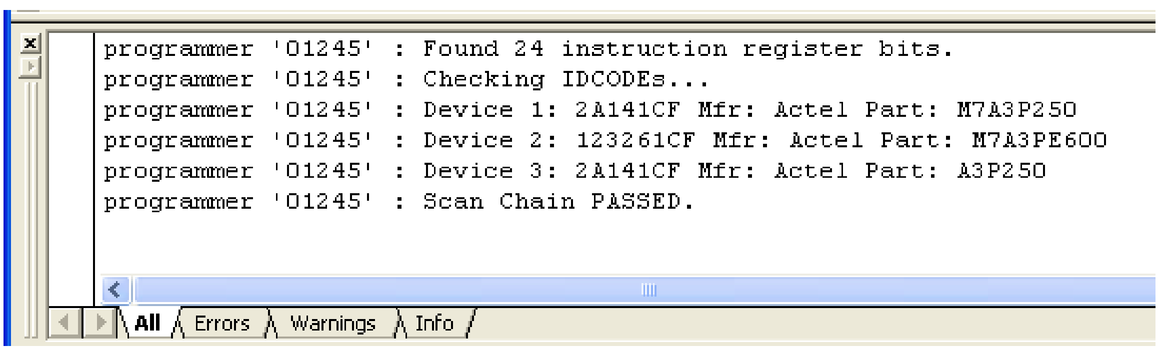


Figure 61 · Scan Chain Order in the Log Window

7. Click the **Configure Chain** button. The **Chain Configuration** window displays (see figure below).

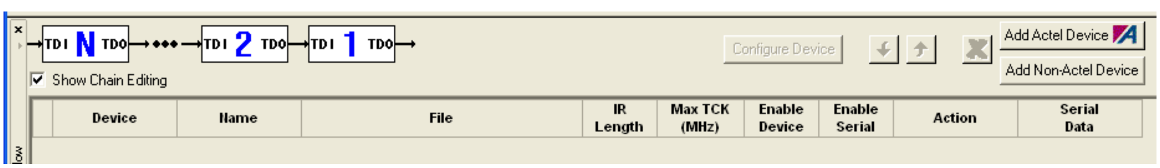


Figure 62 · Chain Configuration Window

8. In the **Chain Configuration** window, click the **Add Device** button to add devices to the chain. The **Add Actel Device** dialog box appears (as shown in the figure below).

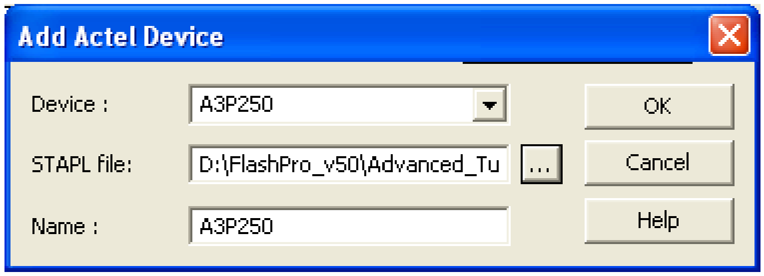


Figure 63 · Add Actel Device Dialog Box

9. Choose the "A3P250" device from the **Device** drop-down.
10. In the **STAPL file** field, use the **Browse** button to locate the A3P250.stp file.
11. In the **Name** field, leave A3P250 as default. The A3P250 device is added into the **Chain Configuration** window (see figure below).

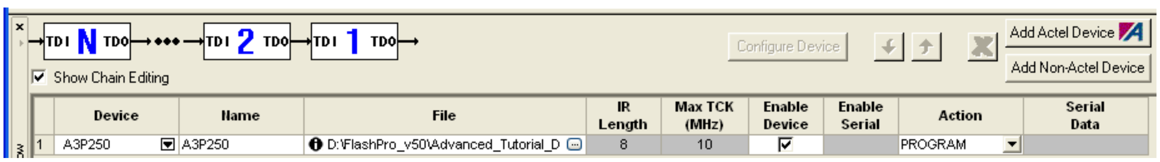


Figure 64 · Device One Chain Configuration Window

12. Repeat the same process for A3PE600 and the other A3P250 respectively.
13. After you have finished adding all of the devices in the chain, the **Chain Configuration** window updates.

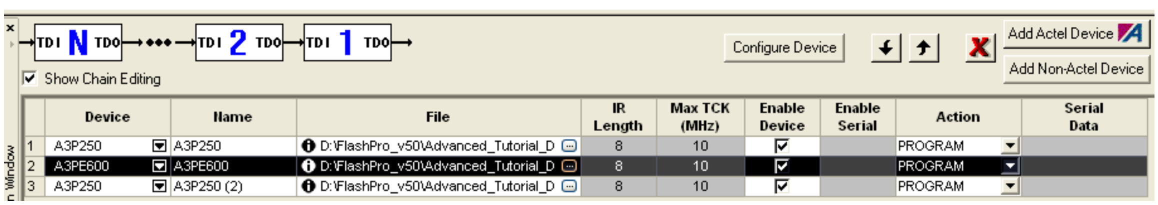


Figure 65 · Chain Configuration Window: All Devices in the Chain

14. Once all the devices have been added to the chain in the correct order, click the **Run** button to program the chain.
15. When Programming is complete, the **Programmer List** window displays. See figure below.

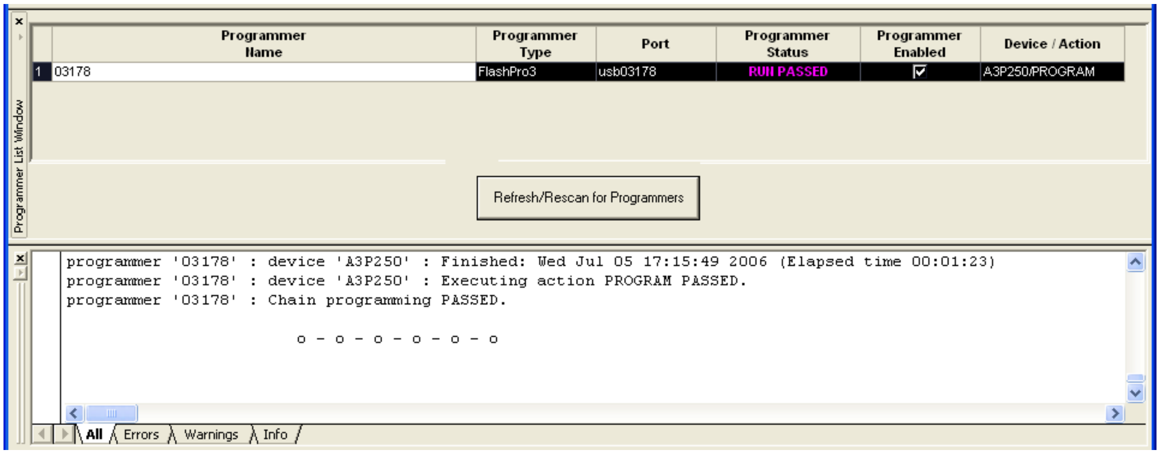


Figure 66 · Programmer List Window Done

Congratulations! You have just completed the FlashPro Multiple Actel Device Chain Programming tutorial.

Multiple Device Serialization Chain Programming

This tutorial provides step-by-step instructions on how to program multiple Actel devices with serialization. Before you begin this tutorial, you should already be familiar with the basic features of the FlashPro software.

Note: This tutorial does not provide software installation instructions. Please have FlashPro already installed before you begin.

In this tutorial you will program two devices in a chain (one device is A3P250 and the other is A3PE600). The STAPL file for the first A3P250 device contains 10 serialization data. See figure below.

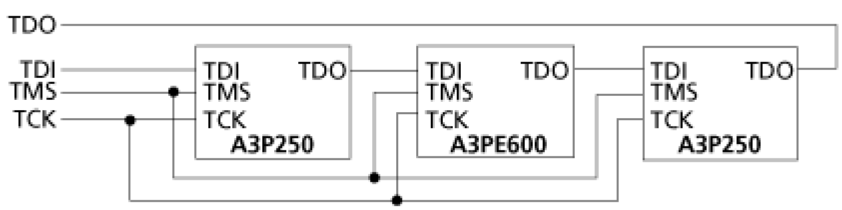


Figure 67 · APA Device Tutorial Example

First you need to create a project.

To create a new project:

1. Click the **New Project** button in FlashPro.
2. From the **New Project** dialog box, type "Tutorial" in the Project Name field.
3. Check the **Chain** box (as shown in the figure below).

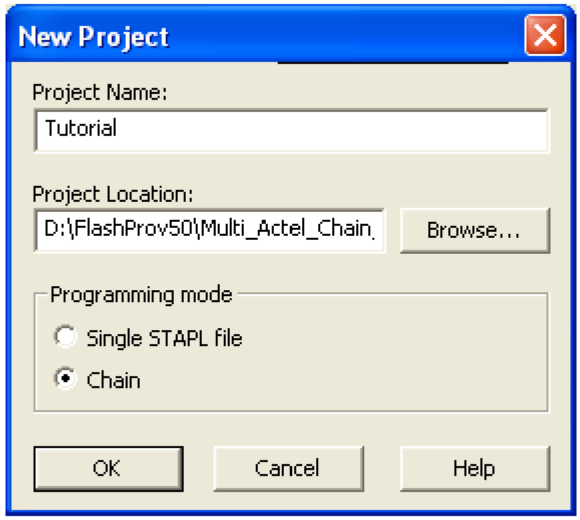


Figure 68 · New Project Dialog Box

4. If necessary, change the default location of your project in the **Project Location** field.
5. Click **OK**. The FlashPro main window appears and updates the Programmer List info with your programmer information.
6. From the **Programmers** menu, choose **Scan Chain** (or select the programmer in the Programmer List window, right-click, then choose **Scan Chain)** (as shown in the figure below).

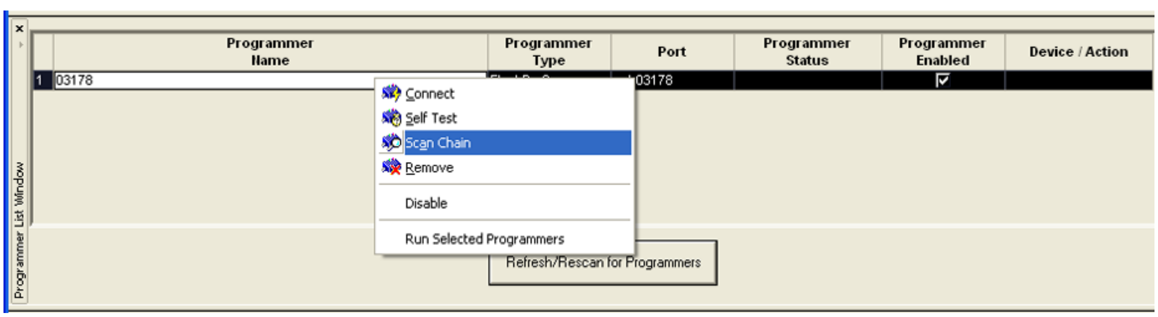


Figure 69 · Scan Chain Selection

Scan Chain shows how the devices are ordered in the chain in the log window (see figure below). In this case, A3P250 is the first device will be programmed in the chain since it is connected directly to TDO.

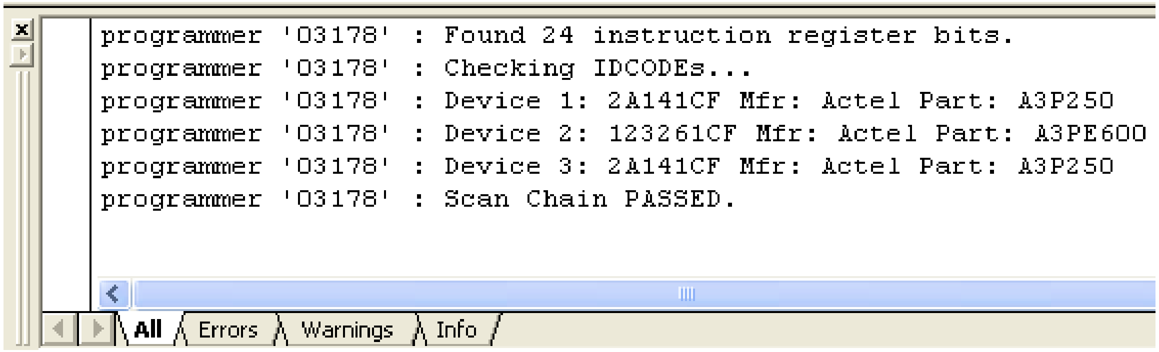


Figure 70 · Scan Chain Order in the Log Window

7. Click the **Configure Chain** button . The **Chain Configuration** window appears (see figure below).

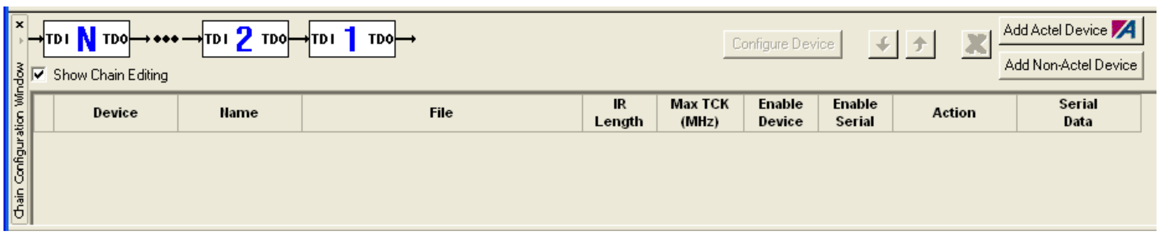


Figure 71 · Chain Configuration Window

8. In the **Chain Configuration** window, click the **Add Device** button to add devices to the chain. The **Add Actel Device** dialog box appears.

9. Choose **A3P250** device from the **Device** drop-down menu (as shown in the figure below).

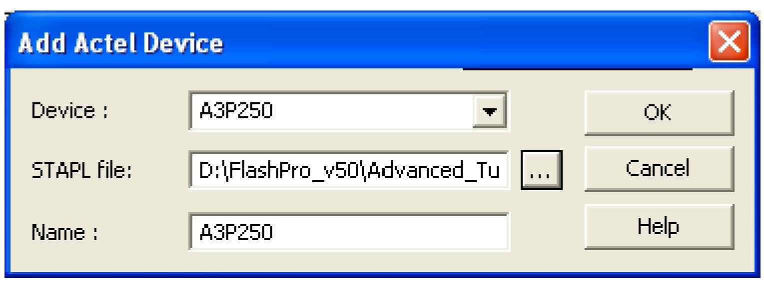


Figure 72 · Add Actel Device Dialog Box

10. In the **STAPL file** field, use the **Browse** button to locate the A3P250.stp file.

11. In the **Name** field, leave A3P250 as default.

12. Click **OK**. The A3P250 device is added into the **Chain Configuration** window.

13. Repeat steps 8 to 11 for A3PE600 and A3P250 respectively. After you are finished adding all devices in the chain, the **Chain Configuration** window updates (as shown in the figure below).

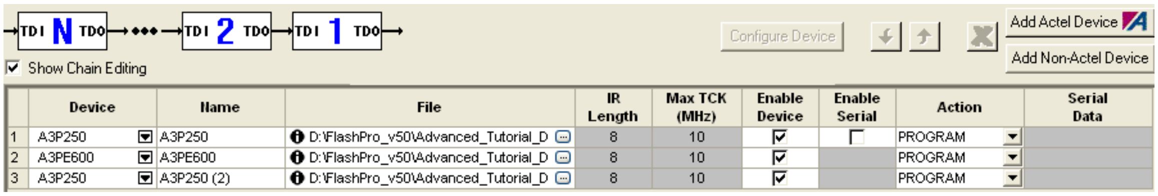


Figure 73 · Chain Configuration Window for all Devices

14. From the **Chain Configuration** Window, check the **Enable Serial** box. This enables the Serial Data option in the Chain Configuration window.
15. Click **Select** in the **Serial Data** column, the **Serial Settings** dialog box displays as shown in the figure below.

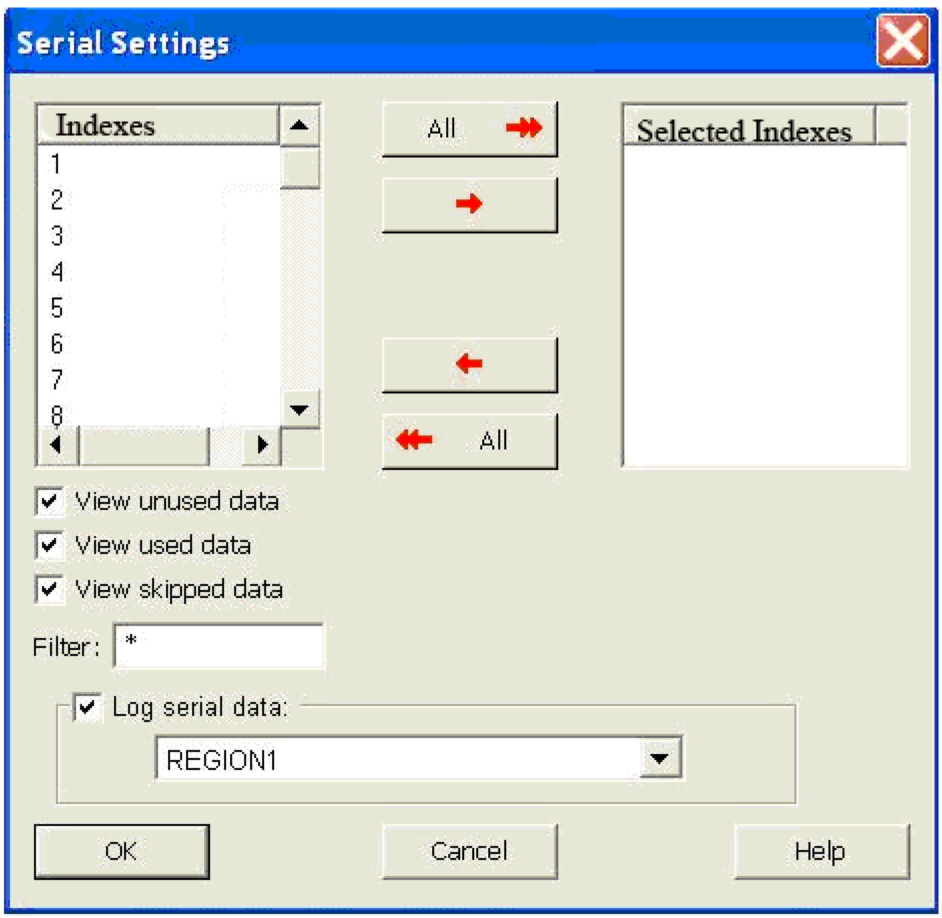


Figure 74 · Serial Settings Dialog Box

16. From the **Serial Settings** dialog box, click the **All** button to select all the serial data.
17. Click **OK**.
18. Once all the devices have been added to the chain in the correct order and serialization has been selected, click the **Run** button to program the chain.

19. When programming is complete, the **Programmer List** window appears and indicates that the devices are ready for programming (as shown in the figure below).

Figure 75 · Programmer List Window Done

Congratulations! You have just completed the FlashPro Multiple Device Serialization Chain Programming tutorial.

Multiple Programmer Multiple Device Chain Programming

This tutorial demonstrates step-by-step instructions on how to parallel program two chains using two programmers, each with two Actel Devices (A3P250 and A3PE600). See the figure below for an illustration.

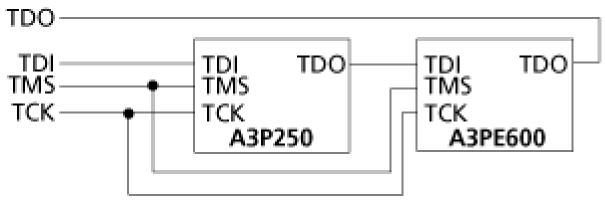


Figure 76 · APA Device Tutorial Example

You should already be familiar with the basic features of the FlashPro software before you begin this tutorial.

Note: This tutorial does not provide software installation instructions. Please have FlashPro already installed before you begin.

First you need to create a project.

1. Click the **New Project** button in FlashPro.
2. From the **New Project** dialog box, type "Tutorial" in the **Project Name** field.
3. Check the **Chain** box (see figure below).

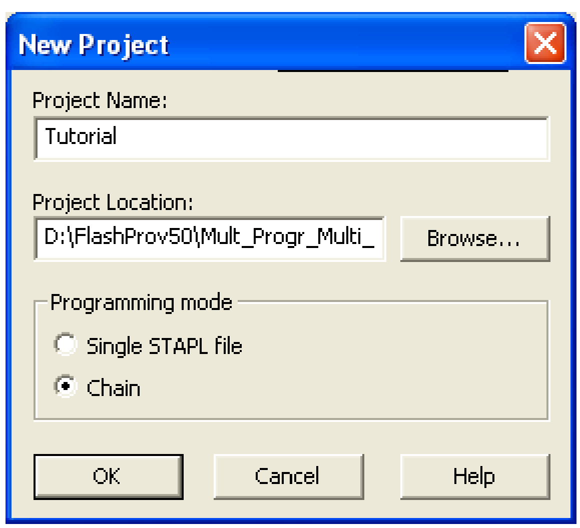


Figure 77 · New Project Dialog Box

4. If necessary, change the default location of your project in the **Project Location** field.
5. Click **OK**. The FlashPro main window appears and displays your updated programmer information (as shown in the figure below).

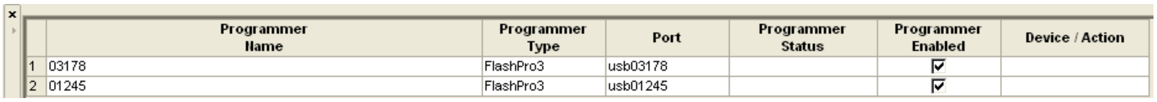


Figure 78 · FlashPro User Interface

6. From the **Programmers** menu, choose **Scan Chain** (or select the programmer in the **Programmer List** window, right-click, then choose **Scan Chain)**. The **Select Programmer(s)** dialog box displays (see figure below).

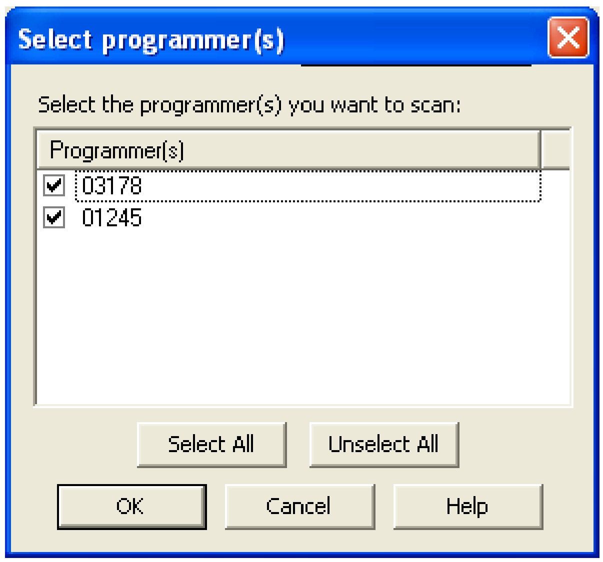


Figure 79 · Select Programmer Window

The **Programmer List** window shows the Scan Chain Test was passed and how the devices are ordered in the chain (see figure below).

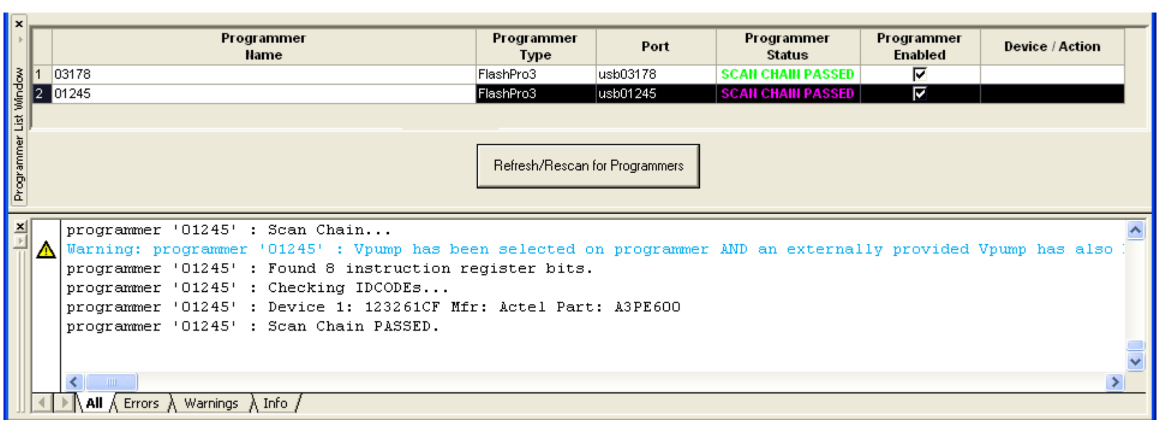


Figure 80 · Scan Chain Order in the Log Window

7. Click the **Configure Chain** button . The **Chain Configuration** window appears (as shown in the figure below).

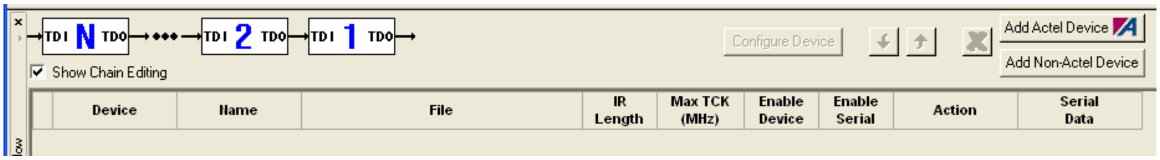


Figure 81 · Chain Configuration Window

8. In the **Chain Configuration** window, click the **Add Device** button to add devices to the chain. The **Add Actel Device** dialog box appears.

9. Choose **A3PE600** device from the **Device** drop-down menu (as shown in the figure below).

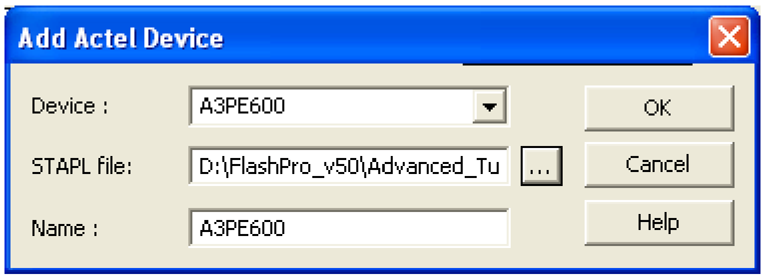


Figure 82 · Add Actel Device Dialog Box

10. In the **STAPL file** field, use the **Browse** button to locate the A3PE600.stp file.

11. In the **Name** field, leave A3PE600 as default.

12. The A3PE600 device is added into the **Chain Configuration** window (as shown in the figure below).

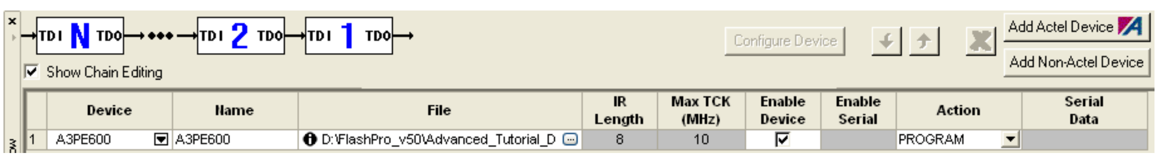


Figure 83 · Device One Chain Configuration Window

13. Repeat steps 8 to 11 for A3P250. After you are finished adding all devices in the chain, the **Chain Configuration** window updates (as shown in the figure below).

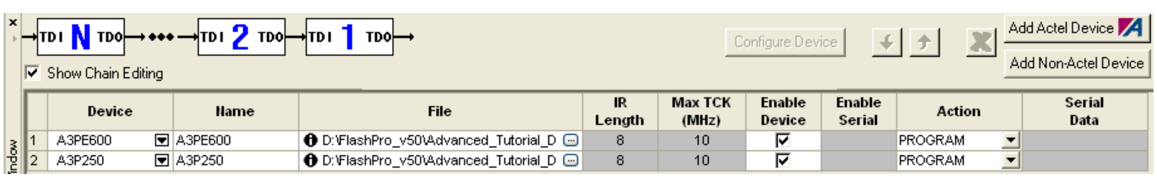


Figure 84 · Chain Configuration Window for all Devices

14. Once all the devices have been added to the chain in the correct order and serialization has been selected, click the **Run** button to program the chain.

15. When programming is complete, the **Programmer List** Window appears (as shown in the figure below).

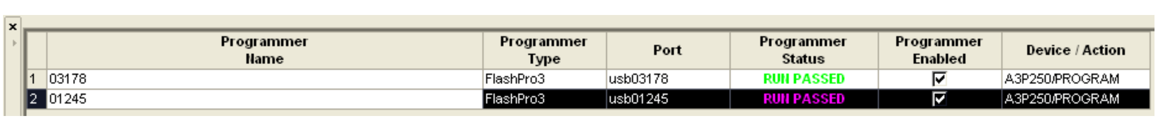


Figure 85 · Programmer List Window Done

Congratulations! You have just completed the FlashPro Multiple Device Serialization Chain Programming tutorial.

Multiple Programmer and Multiple Device Serialization Chain Programming

This tutorial demonstrates step-by-step instructions on how to parallel program two chains using two programmers, each with two Actel Devices (A3P250 with Serialization and A3PE600). See the figure below for an illustration.

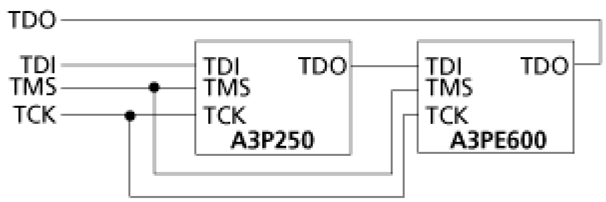


Figure 86 · APA Device Tutorial Example

You should already be familiar with the basic features of the FlashPro software before you begin this tutorial. The STAPL file for the A3P250 device contains 10 serialization data.

Note: This tutorial does not provide software installation instructions. Please have FlashPro already installed before you begin.

First you need to create a project.

1. Click the **New Project** button in FlashPro.
2. From the **New Project** dialog box, type "Tutorial" in the **Project Name** field.
3. Check the **Chain** box (see figure below).

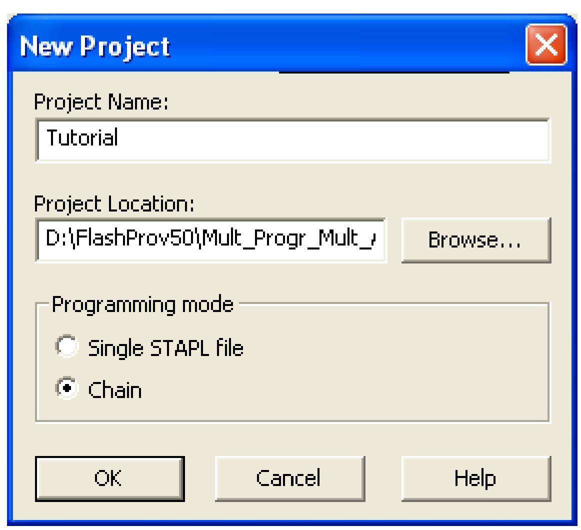


Figure 87 · New Project Dialog Box

4. If necessary, change the default location of your project in the **Project Location** field.

5. Click **OK**. The FlashPro main window and the Programmer List window displays your updated programmer information.

6. From the **Programmers** menu, choose **Scan Chain** (or select the programmer in the **Programmer List** window, right-click, then choose **Scan Chain)**. The **Select Programmer(s)** dialog box appears (as shown in the figure below).

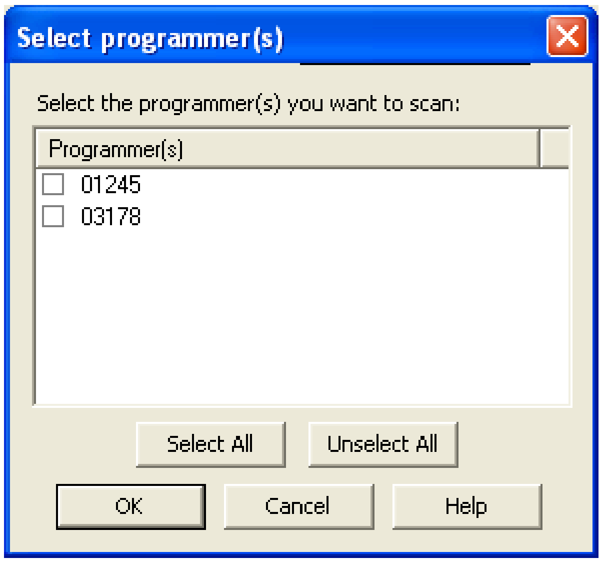
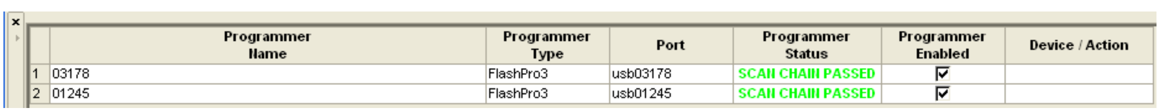


Figure 88 · Select Programmer Window

The **Programmer List** window shows the Scan Chain Test was passed and how the devices are ordered in the chain (see figure below). In this example, A3PE600 will be programmed first in the chain since it is connected directly to TDO.



	Programmer Name	Programmer Type	Port	Programmer Status	Programmer Enabled	Device / Action
1	03178	FlashPro3	usb03178	SCAN CHAIN PASSED	☑	
2	01245	FlashPro3	usb01245	SCAN CHAIN PASSED	☑	

Figure 89 · Scan Chain Order in the Log Window

7. Click the **Configure Chain** button . The **Chain Configuration** window appears (as shown in the figure below).

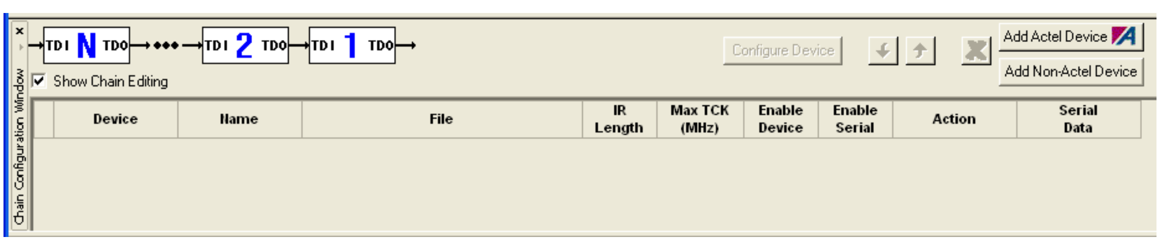


Figure 90 · Chain Configuration Window

8. In the **Chain Configuration** window, click the **Add Actel Device** button to add devices to the chain. The **Add Actel Device** dialog box appears.

9. Choose **A3PE600** device from the **Device** drop-down menu (as shown in the figure below).

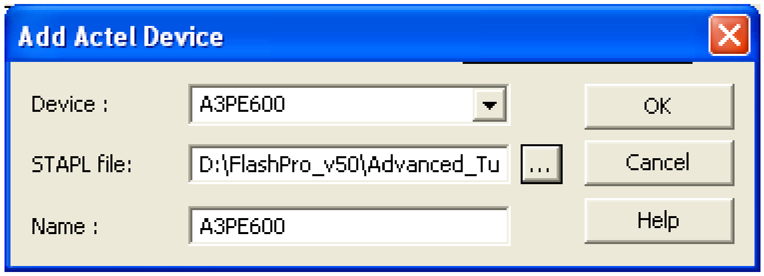


Figure 91 · Add Actel Device Dialog Box

10. In the **STAPL file** field, use the **Browse** button to locate the A3PE600.stp file.
11. In the **Name** field, leave A3PE600 as default. The A3PE600 device is added into the **Chain Configuration** window.
12. Repeat the steps above to add the A3P250. After finished adding all devices in the chain, the **Chain Configuration** window updates (see figure below).

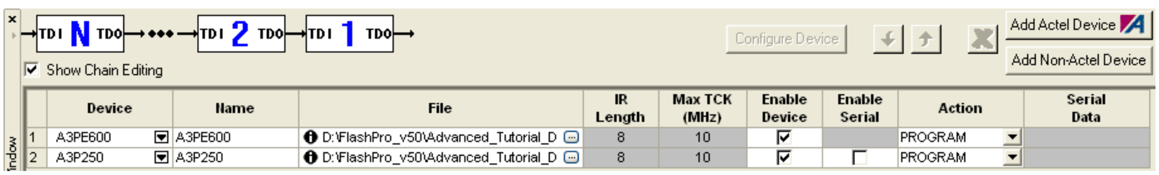


Figure 92 · Chain Configuration Window for all Devices

14. In the **Chain Configuration** Window, check the **Enable Serial** box.
15. Click **Select** in the **Serial Data** column. The **Serial Settings** dialog box appears (as shown in the figure below).

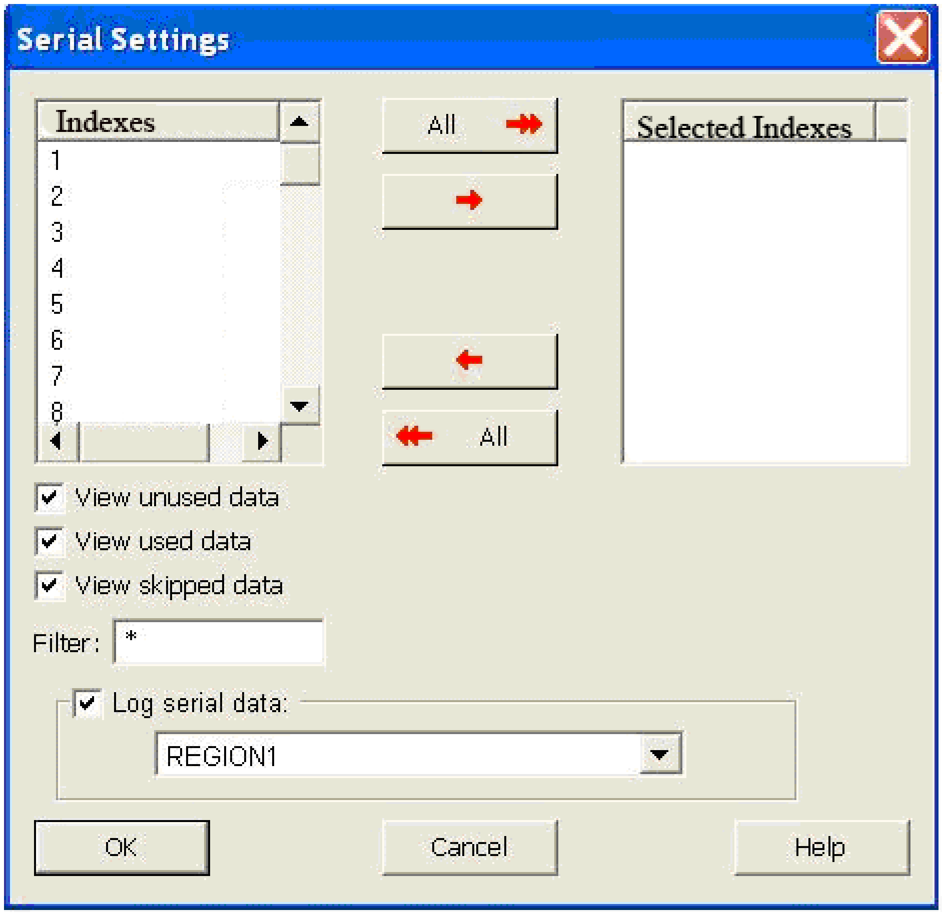
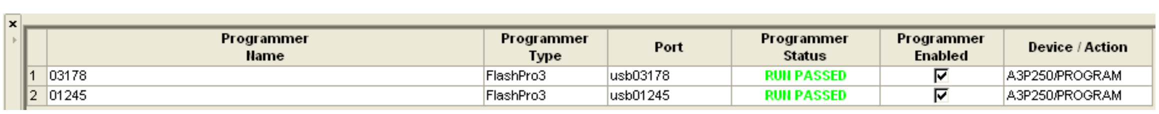


Figure 93 · Serial Settings Dialog Box

16. From the **Serial Settings** dialog box, click the **All** button to select all the serial data.
17. Click **OK**.
18. Once all the devices have been added to the chain in the correct order and serialization has been selected, click the **Run**button to program the chain.
19. When programming is complete, the **Programmer List** window updates (as shown in the figure below).



	Programmer Name	Programmer Type	Port	Programmer Status	Programmer Enabled	Device / Action
1	03178	FlashPro3	usb03178	RUN PASSED	☑	A3P250/PROGRAM
2	01245	FlashPro3	usb01245	RUN PASSED	☑	A3P250/PROGRAM

Figure 94 · Programmer List Window Done

Congratulations! You have just completed the FlashPro Multiple Programmer and Multiple Device Serialization Chain Programming tutorial.

Setting Disabled Actel Devices to HIGH-Z

This tutorial explains how to set disabled Actel IGLOO, ProASIC3, SmartFusion, Fusion and ProASIC^{PLUS} devices in a chain to HIGH-Z during chain programming.

Note: This tutorial requires a design with valid *.pdb/*.stp files for a chain of devices. If you launch FlashPro from a Libero IDE project, a FlashPro project is created automatically and the PDB/STP file loaded. Otherwise, you can start a new FlashPro project and load a PDB/STP file; refer to the [Chain Programming Tutorial] for more information.

1. Once all your devices have been added to the chain, from the Chain Configuration window, choose which devices you would like disabled during programming by de-selecting the appropriate checkbox from the Enable Device column.

Now that you have disabled devices a checkbox appears in the HIGH-Z column of the Chain Configuration window. If the HIGH-Z column is not shown in the Chain Configuration Grid, right-click any column header and choose **HIGH-Z**.

2. Select the HIGH-Z checkbox to ensure your disabled devices enter HIGH-Z mode and remains in that mode until chain programming is complete.

HIGHZ is not supported if enabled Actel devices are executing one of the following ACTIONS:

- PROGRAM_NVM_ACTIVE_ARRAY
- VERIFY_NVM_ACTIVE_ARRAY
- READ_IDCODE

Disabled devices I/Os will not go to HIGHZ and the they will not tri-state. Any other ACTION will work as expected

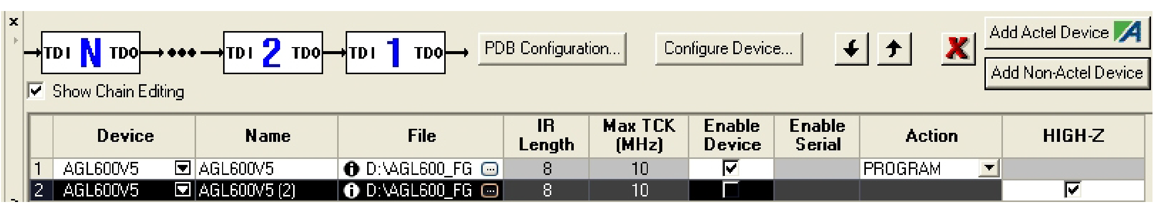


Figure 95 · HIGH-Z Option in FlashPro

Programming Settings and Operations

Introduction

The FlashPro software enables you to connect multiple programmers to your computer. With each programmer you select, you can connect the programmer, perform a self-test, customize, add, and remove and analyze the JTAG chain.

Programmer Settings

The Programmer Settings dialog box includes setting options for FlashPro4/3/3X, FlashPro Lite and FlashPro.

Note: You can set the TCK setting in the PDB/STAPL file by selecting the TCK frequency in the Programmer Settings dialog box.

Limitation of the TCK frequency for the selected programmer:

- FlashPro supports 1-4 MHz
- FlashPro Lite is limited to 1, 2, or 4 MHz only.
- FlashPro4/3/3X supports 1-4 MHz.

Limitation of the TCK frequency for the target device:

- IGLOO, ProASIC3, and Fusion – 10MHz to 20MHz
- ProASICPLUS and ProASIC – 10 MHz.

During execution, the frequency set by the FREQUENCY statement in the PDB/STAPL file will override the TCK frequency setting selected by you in the Programmer Settings dialog box unless the **Force TCK Frequency** checkbox is selected.

To set your programmer settings:

1. From the **Tools** menu, choose **Programmer Settings**.The **Programmer Settings** dialog box appears (as shown in the figure below).

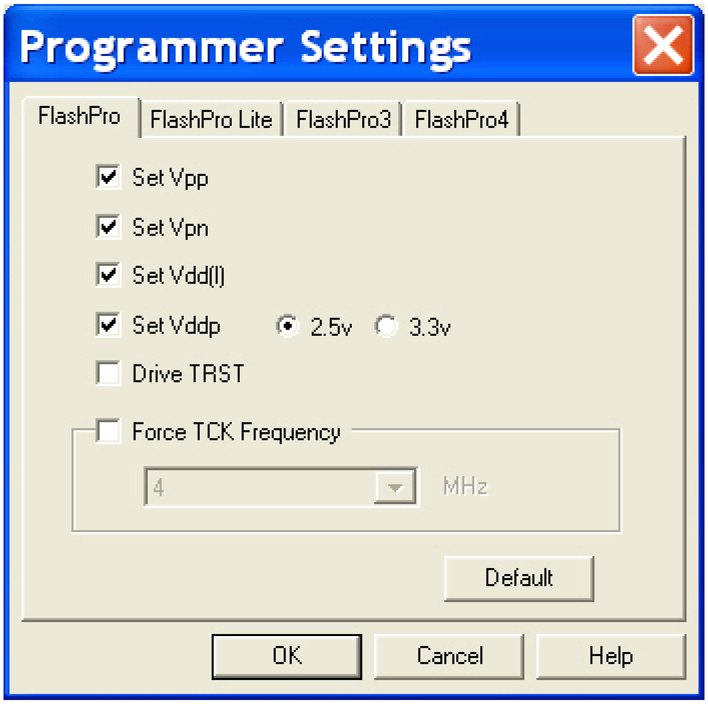


Figure 96 · Programmer Settings Dialog Box for FlashPro

2. Click a programmer tab and check the appropriate settings for your programmer.
3. Click **OK**.

FlashPro Programmer Settings

Choose your programmer settings for FlashPro (see above figure). If you choose to add the Force TCK Frequency, select the appropriate MHz frequency. After you have made your selection(s), click **OK**.

Default Settings

- The Vpp, Vpn, Vdd(l), and Vddp options are checked (Vddp is set to 2.5V) to instruct the FlashPro programmer(s) to supply Vpp, Vpn, Vdd(l) and Vddp.
- The Driver TRST option is unchecked to instruct the FlashPro programmer(s) NOT to drive the TRST pin.
- The Force TCK Frequency option is unchecked to instruct FlashPro to use the TCK frequency specified by the Frequency statement in the STAPL file(s).

FlashPro Lite Programmer Settings

If you choose to add the Force TCK Frequency, select the appropriate MHz frequency. After you have made your selection(s), click **OK**.

Default Settings

- The Vpp and Vpn options are checked to instruct the FlashPro Lite programmer(s) to supply Vpp and Vpn.

- The Driver TRST option is unchecked to instruct the FlashPro Lite programmer(s) NOT to drive the TRST pin.
- The Force TCK Frequency option is unchecked to instruct the FlashPro Lite to use the TCK frequency specified by the Frequency statement in the STAPL file(s).

FlashPro4/3/3X Programmer Settings

For FlashPro3, you have the option of choosing the Set Vpump setting or the Force TCK Frequency. If you choose the Force TCK Frequency, select the appropriate MHz frequency. For FlashPro4/3X settings, you have the option of switching the TCK mode between Free running clock and Discrete clocking. After you have made your selections(s), click **OK**.

Default Settings

- The Vpump option is checked to instruct the FlashPro3 programmer(s) to supply Vpump to the device.
- The Force TCK Frequency option is unchecked to instruct the FlashPro3 to use the TCK frequency specified by the Frequency statement in the PDB/STAPL file(s).
- FlashPro3x default TCK mode setting is Free running clock

FlashPro4 Programmer Settings

For FlashPro4, you have the option of choosing the Set Vpump setting or the Force TCK Frequency. If you choose the Force TCK Frequency, select the appropriate MHz frequency. You have the option of switching the TCK mode between Free running clock and Discrete clocking. After you have made your selections(s), click OK.

Default Settings

- The Vpump option is checked to instruct the FlashPro4 programmer(s) to supply Vpump to the device.
- The Force TCK Frequency option is unchecked to instruct the FlashPro4 to use the TCK frequency specified by the Frequency statement in the PDB/STAPL file(s).
- The default TCK mode setting is Free running clock

TCK Setting (ForceTCK Frequency)

If **Force TCK Frequency** is checked (in the **Programmer Setting**) then the selected TCK value is set for the programmer and the Frequency statement in the PDB/STAPL file is ignored.

Note: FlashPro Lite RevA supports only 4MHz on TCK.

Default TCK frequency

When the PDB/STAPL file or Chain does not exist, the default TCK frequency is set to 4MHz. In the **Single Device File Programming** mode, FlashPro will parse through the file and search for the "freq" keyword and the "MAX_FREQ" Note field, which are expected in all Actel Flash device files. The FlashPro software uses the lesser value of the two as the default TCK frequency.

In **Chain Programming** mode, when more than one Actel flash device is targeted in the chain, the FlashPro software passes through all of the files and searches for the "freq" keyword and the "MAX_FREQ" **Note** field. The FlashPro software uses the lesser value of all the TCK frequency settings and the "MAX_FREQ" **Note** field values.

Ping Programmers

To ping a programmer(s):

1. From the **Programmers** menu, choose **Ping**.
2. Select the programmers you want to connect from the **Select Programmer(s)** dialog box.
3. Click **OK**.

Note: You can click the Refresh/Rescan for Programmers button to quickly ping new programmers.

Performing a Self-Test

To perform a self-test:

1. From the **Programmers** menu, choose **Self Test.**
2. Select the programmer(s) you want to self-test from the **Select Programmer(s)** dialog box.
3. Click **OK**.

Note: You must connect the programmer to the self-test board that comes with your programmer before performing a self-test.

You can also perform self-test by right-clicking on a specific programmer from the **Programmer List Window** and selecting **Self-Test**.

Note: Self-test is not supported with FlashPro Lite programmers.

Scanning a Chain

The scan chain operation scans and analyzes the JTAG chain connected to programmer(s) you have selected.

To scan a chain:

1. From the **Programmers** menu, choose **Scan Chain**.
2. Select the programmers you want to scan from the **Select Programmer(s)** dialog box.
3. Click **OK**.

You can also perform Scan Chain by right-clicking on a specific programmer from the **Programmer List Window** and selecting **Scan Chain**.

To scan and check a chain:

1. From the **Tools** menu, choose **Modes** > **Chain Programming**.
2. From the Chain Configuration window, select **auto construct or add devices**.

3. From the **Programmers** menu, choose **Scan and Check Chain**.
4. Select the programmers that you want to scan and check chain from the **Select Programmer(s)** dialog box.
5. Click **OK**.

You can also perform Scan Chain and Scan and Check Chain by right-clicking a specific programmer from the **Programmer List Window** and selecting **Scan Chain** or **Scan and Check Chain**.

Enabling and Disabling Programmers

Once your programmer is enabled you can connect the programmer, perform a self-test, scan the chain, or remove it.

To enable a programmer:

1. From the **View** menu, choose **Programmer Details Window**.
2. Check the **Enable programmer** checkbox in the **Programmer Details** Window.

The **Programmer Details** window displays all the information about your programmer.

Note: Note: You can also enable your programmer from the Programmer List window by checking the checkbox in the Programmer Enabled column.

Disable your programmer by unchecking the **Enable programmer** checkbox from the **Programmer Details** Window or by unchecking the checkbox in the **Programmer Enabled** column in the **Programmer** window.

Renaming a Programmer

Enter the new programmer name in the **Programmer Details** window to rename the programmer. By default, the programmer name is the same as the programmer ID.

Removing a Programmer

To remove a programmer:

1. From the **Programmers** menu, choose **Remove**.
2. Select the programmers you want to remove from the **Select Programmer(s)** dialog box.
3. Click **OK**.

Selecting Programmers

The **Select Programmer(s)** dialog box gives you the option of selecting all and unselecting all of the programmers that you want to ping. See figure below.

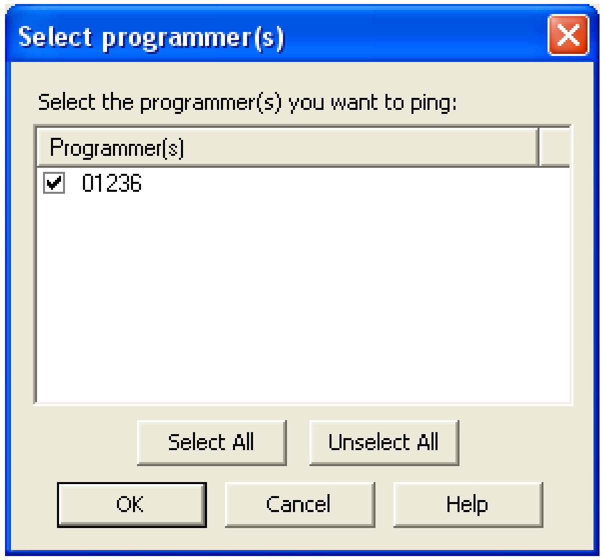


Figure 97 · Selecting Programmer(s) Dialog Box

Single Device Configuration

Single Device Programming

When devices are joined together in a JTAG chain, all of their Instruction Registers (IR) and Data Registers (DR) are put in a long shift register from TDI to TDO. The IR length defers from device to device and the DR length depends on the instruction that shifts into the instruction register.

When targeting Device 2 (see figure below), you need to know the IR length for Device 1 and Device 3. Given this information, you can bypass both devices by shifting an all one pattern into their instruction registers before and after the instruction targeted at Device 2. The number of bits you shift before Device 2's instruction is the pre IR length, and the number of bits you need to add after Device 2's instruction is the post IR length. In this case, the pre IR bits are shifted into

Device 1 and post IR bits are shifted into Device 3.

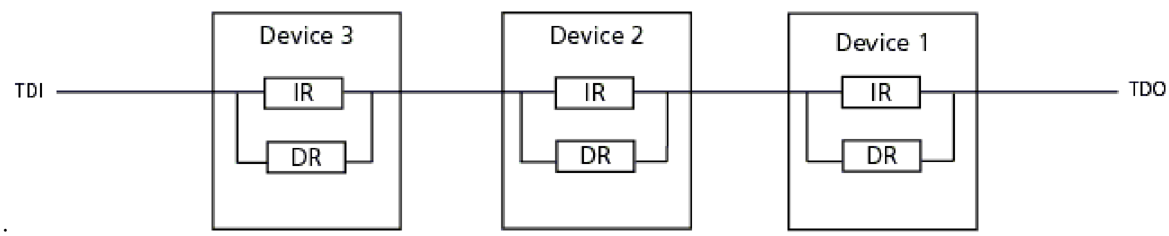


Figure 98 · Targeting Device 2

When the bypass instruction is shifted into Device 1 and Device 3, the TDI and TDO of the two devices are connected to the 1-bit bypass register at the Shift-DR state. To correctly shift the data in and data out of Device 2's register, you need to shift one bit of data before and after Device 2's data.

The number of bits you need to shift into the data register for Device 1 is the pre-DR length and the number of bits we need to shift into the data register for Device 3 is the post-DR length. With the IR and DR length information, you can shift instructions and data into Device 2 with the correct registration.

To create the JTAG chain (shown in the above figure):

1. Connect the TCK and TMS from the programmer to all of the devices.
2. Connect the programmer’s TDI pin to the TDI pin of device 3.
3. For all devices in the chain, connect the TDO output of one device to the TDI input of the next device.
4. Connect the TDO output of the last device to the programmer’s TDO input.

The order of devices in the chain is set by the connections of TDI to TDO.

The ChainBuilder software takes the order of the devices in a chain and their IR lengths and adds the pre-IR, post-IR, pre-DR, and post-DR padding bits in the device you want to program, which properly aligns the instructions and data within the IR and DR of the devices.

If you do not use the ChainBuilder software, the FlashPro software tries to find the pre-IR, post IR, pre-DR, and post-DR values during the Analyze Chain operation.

For more information, see ProASIC programming and ProASICPLUS and ProASIC programming introduction.

To find out how to set the IR length, see Chain Settings.

Loading a Programming File

You can either load a programming file from the **Configuration** menu or from the **Single Device Programming** Window. The section below describes how to load a programming file from the **Single Device Programming** Window.

To Load a programming file from the Single Device Programming window:

1. From the **View** menu, choose **Single Device Configuration** to activate the **Single Device Configuration** Window.
2. Click the **Browse** button in the **Single Device Configuration** Window (see figure below).

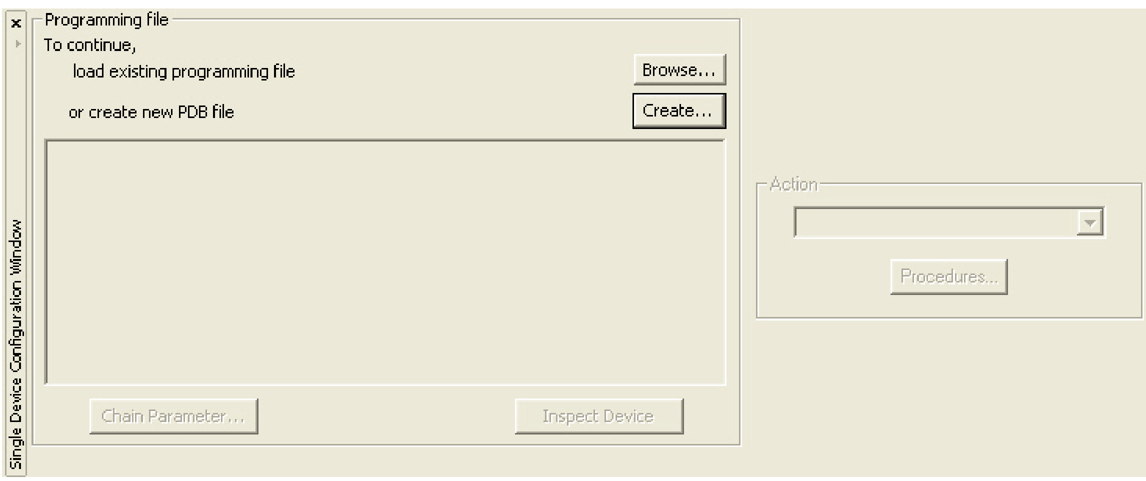


Figure 99 · Signal Device Configuration Window

The **Load Programming File** dialog box appears.

3. Navigate to your programming file, select it, and click **Open**. The programming file is loaded and the **Single Device Configuration** Window updates.

Select Target Device

The **Select Target Device** dialog box is located in the **Configuration** menu.

The **Select Target Device** dialog box enables you to select the target device you want to program. If you are only programming one device in a chain, there is no need for you to make a selection. The **Select Target Device** dialog box automatically displays your device (see figure below).

Figure 100 · Select Target Device (One Device in a Chain)

If you are programming more than one Flash device in a chain, you need to select the target device you want to program. If you attempt to program your device without selecting a target device, a warning message appears.

If the warning message appears, click **OK** and the **Select Target Device** dialog box appears. From the **Select Target Device** dialog box, select the device you want to program (see figure below).

Figure 101 · Select Target Device (Multiple Devices in a Chain)

Click the down arrow to display the list of devices in your chain. Then, make your selection and click **OK**.

Note: When the FlashPro software does not detect your chain configuration, you must specify the Pre/Post IR fields by entering these values in the Set Pre/Post IR Values dialog box (see figure below).

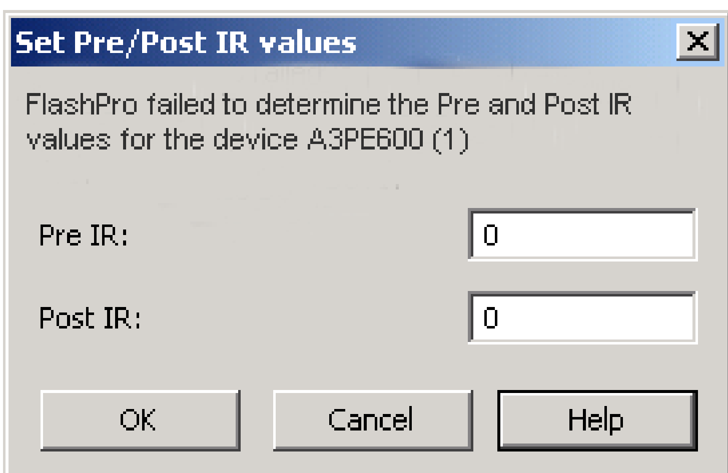


Figure 102 · Pre IR and Post IR Values for the Target Device

For more information, see Single STAPL file programming information.

Chain Settings

Click the **Chain Parameter** button in the **Single Device Configuration** window to set the chain settings (see the **Chain Settings** dialog box below). See Single Device Programming Information for more information about these STAPL settings.

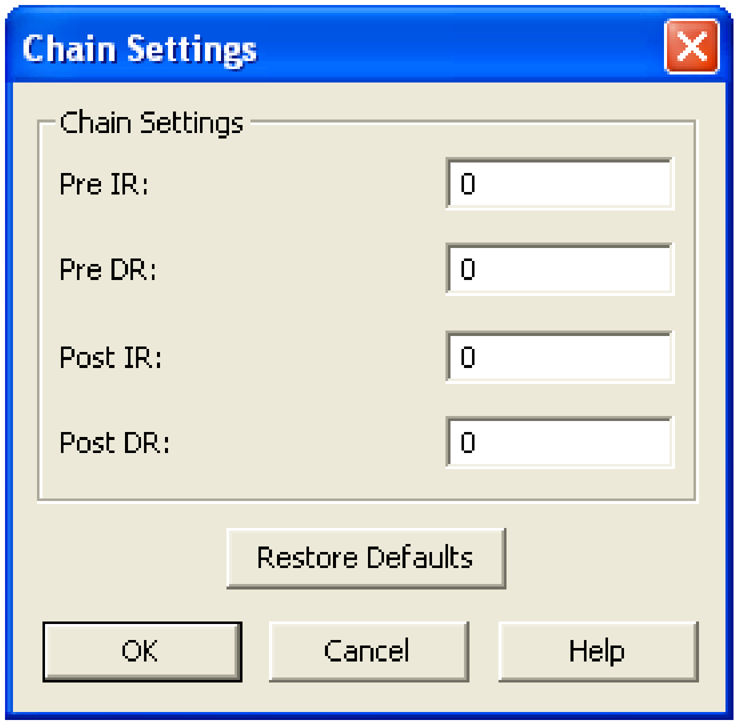


Figure 103 · Chain Settings Dialog Box

Serial Settings

Click the **Select Serialization Indexes/Select Serialization Actions** button from the **Single Device Configuration** Window. The Serial Settings dialog box appears (as shown in the figure below).

Note: Depending on the STAPL file format (Actel format or generic format) used, you will either see Indexes columns or Actions columns in the Serial Settings dialog box.

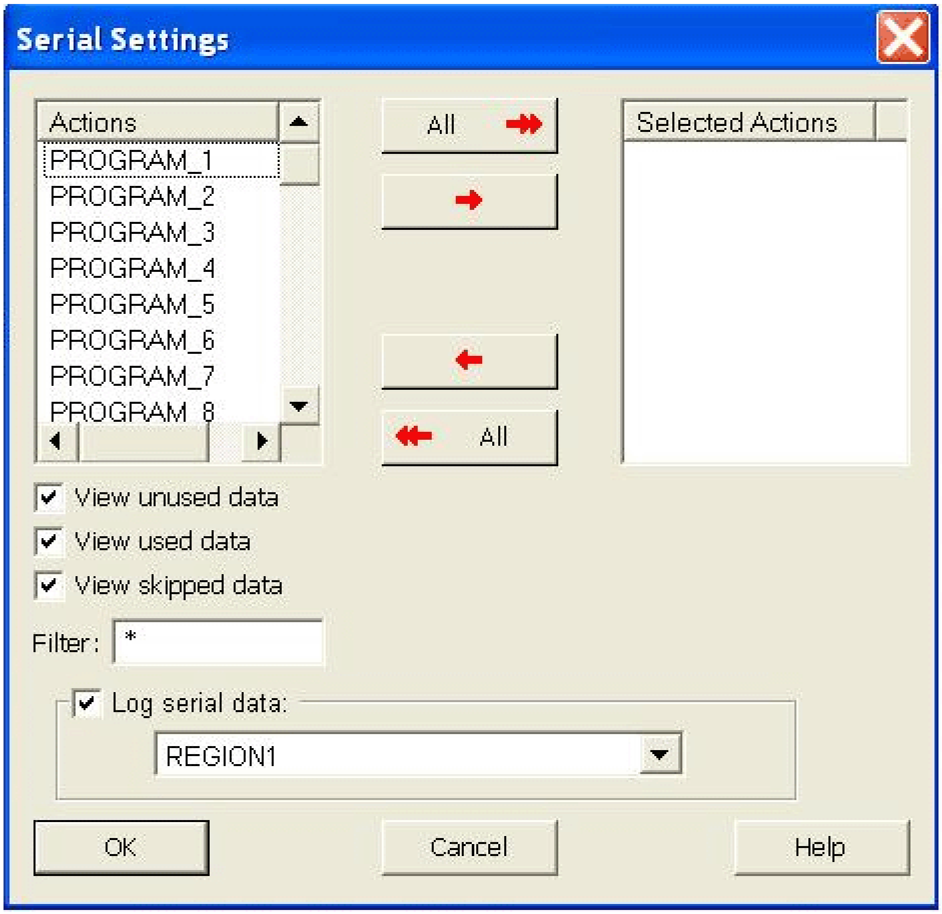


Figure 104 · Serial Settings Dialog Box

Click the red arrow buttons in the center of the dialog box to move from the **Actions** column to the **Selected Actions** column. The indexes/actions available for selection are located on the left and the indexes/actions you choose to select are located on the right column. Viewing options are available in the checkboxes under the **Actions** column. If you check Log serial data, you can select the FlashROM region name where the serial data will be stored.

Chain Programming

Chain Order

The chain order is located in the **Chain Configuration Window**. The devices you add to the chain must be in the correct order and must match the physical chain to be programmed. The TDO for the first device connects to the programmer, and the last device's TDI connects to the programmer. The devices in the chain go in order from a device's TDI into the next device's TDO, as shown in the figure below.

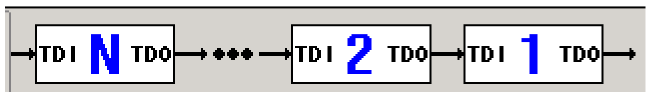


Figure 105 · Chain Order

Multiple Device Chain Programming

The FlashPro software enables direct chain programming without generating a chain STAPL file. Each device will be programmed in sequential order starting from device 1 to device N. See example below. For more information about chain order, see the Chain Order help topic.

TDI > Device N > Device N-1 >… > Device 2 > Device 1 > TDO

You have the advantage of using the Chain Builder GUI interface to construct the target physical chain. Therefore, you do not need to calculate the PRE/POST IR/DR value of the target device. Instead, you must provide either a valid BSDL file or the IR length and TCK Fmax values when you add a non-Actel device to the chain.

You also have the advantage of automatically generating the chain from a scan chain operation. If you connect the target chain to your Actel programmer, then you can automatically construct the chain. Refer to the Automatic Chain Construction Tutorial for more information.

Note: Even though the FlashPro software enables direct chain programming without generating a Chain STAPL file, this functionality is still available. For more information, see Export Chain STAPL file.

Device Programming Compatibility

The following is a list of flash devices that can be programmed together in a chain.

- IGLOO, ProASIC3, SmartFusion and Fusion, excluding ProASIC3L, families can be programmed in the same chain.
- ProASIC^{PLUS} can only be programmed with other ProASIC^{PLUS} devices.
- ProASIC can only be programmed with other ProASIC devices.

Programmer Support

FlashPro4/3/3X supports only IGLOO, ProASIC3, SmartFusion and Fusion family devices. The Vpump on FlashPro4/3/3X is designed to support the programming of only one device. Please make sure that Vpump, Vcc and Vjtag are provided on board for chain programming. Connect the Vpump to the header as the FlashPro software will attempt to check for all external supplies, including Vpump, to ensure successful programming. There is no limitation to the chain length; however, ensure that the JTAG signal integrity and the timing are preserved.

FlashPro and FlashPro Lite support both ProASIC^{PLUS} and ProASIC devices. However you cannot program both devices in the same chain.

Unless all supplies are provided on board, there is a limitation of programming eight ProASIC^{PLUS} or ProASIC devices in a chain.

Multiple Device Serialization in a Chain

When you program multiple IGLOO, ProASIC3, SmartFusion and Fusion family parts, you can use the serialization functionality for more than one device. You must generate STAPL files with the correct serialization data in them to use this functionality.

Each serialization enabled STAPL file may contain a different number of serialization data, but you may only select the same number of serialization data to program in a single Serialization/Programming session.

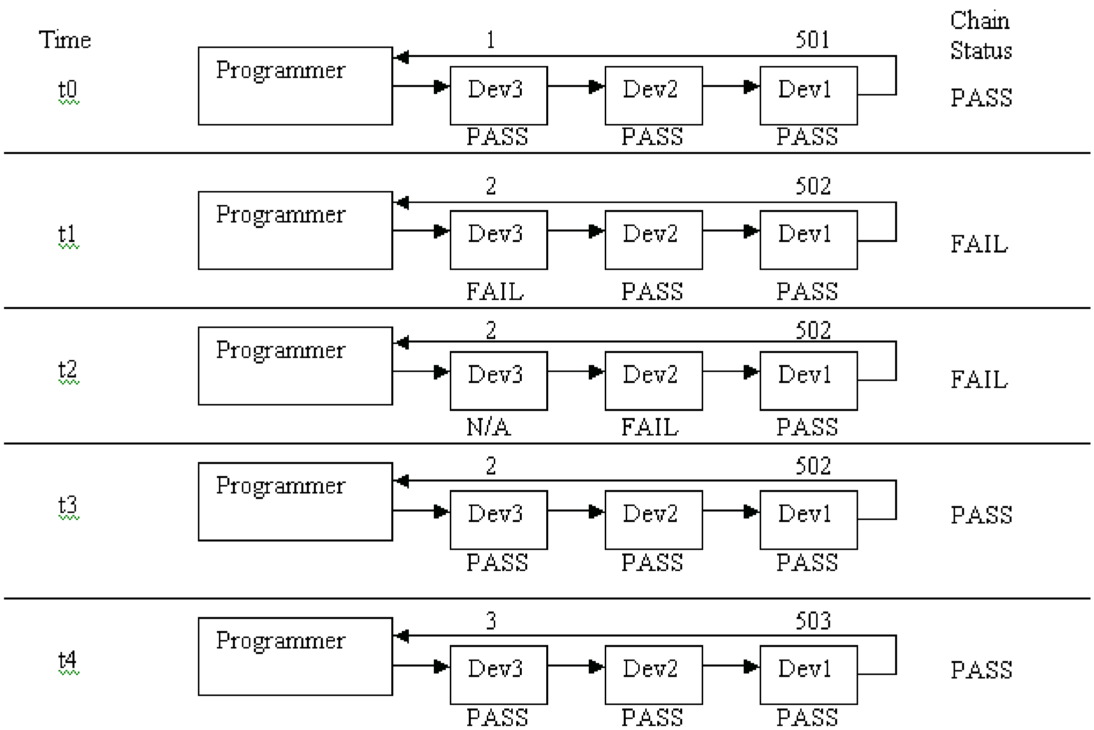
See the example below for further explanation:

In this example, you have a chain of two devices (one device is an A3P250 and the other device is an A3PE600). The STAPL file for the A3P250 contains 10,000 serialization data and the STAPL file for the A3PE600 has 5,000 serialization data. In a single Serialization/Programming session, you are allowed to select serialization data indexed from 1 to 1,000 for the A3PE600 device and serialization data indexed from 5,001 to 6000 for the A3P250 device. However, FlashPro errors out (at the beginning of a programming operation) when the amount of the Serialization data you select is different from the devices.

Reuse Serial Data That Failed Programming

If any of the devices in the chain fail programming, the entire chain fails. All of the devices with serialization enabled will fail as well. The serialization data will be reused or skipped based on your settings. See the example below for more information:

You have a chain with three devices. Device 1 and Device 3 are serialization enabled. You have selected Serialization Data 1 to 100 for Device 1 and 501 to 600 for Device 3, and you have set to reuse any unused Serialization Data. Device 2 is targeted for programming without serialization. See the figure below for an illustration.

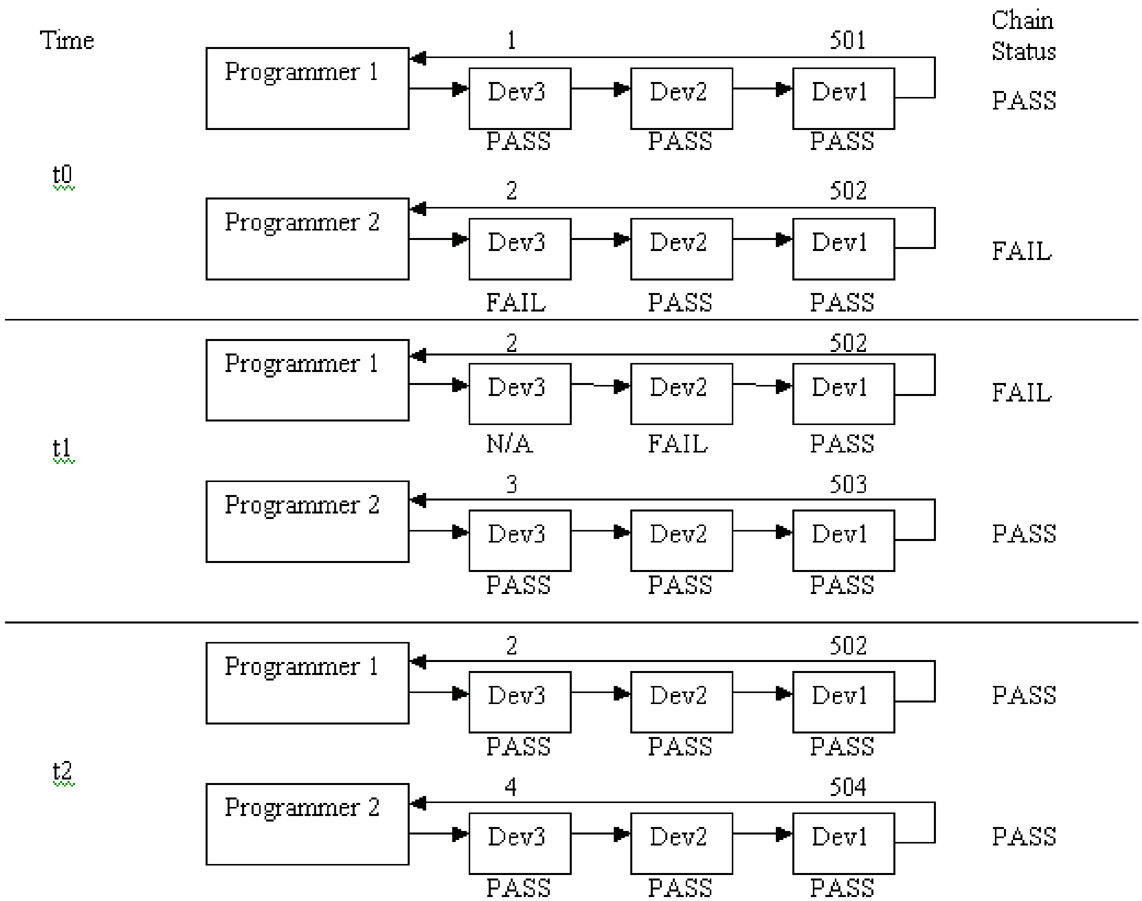


At time t0, all devices in the chain passed programming so the Serialization Data indexes are advance to 2 and 502 for Device 3 and Device 1 respectively. At time t1 and t2, one of the devices failed programming so the device indexes are reused for time t3.

Note that at time t2, when Device 2 failed to program; Device 3 will not be programmed.

Multiple Device Serialization and Parallel Programming

The FlashPro software enables parallel programming for ProASIC3, excluding ProASIC3L, family devices using multiple FlashPro4/3/3X programmers. The following figure illustrates how the indexes are reused in a parallel programming environment.



At time t0, the chain failed to program so index 2 and 502 are reassigned to Device 3 and Device 1 respectively at time t1. The failed indexes are not assigned to the programmer that previously failed. It will always be assigned to the devices in the first programmer in the Programmer List.

Chain Configuration Window

The Chain Configuration Window displays the chain order, the chain editing options, and the chain configuration grid (see figure below). The Chain Configuration Grid enables you to view and set options for each of your devices. Right-click a column heading in the grid and choose a menu option to show or hide a specific column.

The Chain Configuration Grid enables you to view, sort and/or set the following options:

- Device - Device name
- Name - Editable field for a user-specified device name. If you have two or more identical devices in your chain you can use this column to give them unique names.
- File - Path to programming file
- IR Length - Device instruction length.

- Max TCK (MHz) - Maximum clock frequency to program a specific device. FlashPro uses this information to ensure that the programmer operates at a frequency lower than the slowest device in the chain.
- Enable Device - Select to enable the device for programming
- Enable Serial- Select to enable serialization when you have loaded a serialization programming file
- Action - List of programming actions for your device.
- Procedures - Advanced option; enables you to customize the list of recommended and optional procedures for the selected Action.
- Serial Data - Opens the Serial Settings dialog box; enables you to set your serialization data.
- Serial Status - Displays serialization status; lists serialization index(es)/action(s) that have been used and shows the next serialization data that will be programmed.
- HIGH-Z - Sets disabled ActelIGLOO, ProASIC3, SmartFusion, Fusion and ProASIC^{PLUS} devices in the chain to HIGH-Z (tri-states all the I/Os) during chain programming of enabled Actel devices in the daisy chain.

The **Show Chain Editing** checkbox, when checked displays your chain editing options (Configure device, Add Actel Device, Add Non-Actel Device, and organization buttons to move your device within the grid).

Note: For information on how to Add Actel and Non-Actel devices, see Chain Editing.

Note: For information on how to use the Organize buttons (located next to the Add Actel and Add Non-Actel buttons) in the Chain Configuration grid, see Using the Organize buttons in the Chain Programming grid.

You can enable programming and serialization by checking the **Enable Device** checkbox and the **Enable Serial** checkbox in the **Chain Configuration** grid.

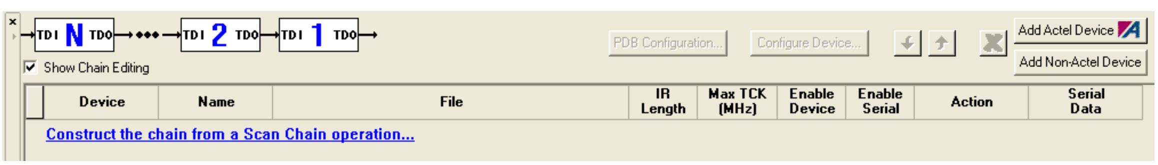


Figure 106 · Chain Configuration Window

Chain Editing Options

The FlashPro software enables you to automatically construct the chain by clicking the **Construct the chain from a Scan Chain operation** link, or by selecting **Construct Chain Automatically** from the **Configuration** menu.

FlashPro also enables you to manually edit your chain by adding Actel and Non-Actel devices. You can add devices by clicking the **Add Actel Device** button and the Add **Non-Actel Device** button, or you can select these options from the **Configuration** menu.

For more information about how to edit the chain, see Chain Editing.

Editing the Chain Configuration Grid

The Chain Configuration Grid enables you to select an Action for your device, Enable Serialization, and edit the grid using the right-click menu.

To select an Action from the Configuration Grid:

1. Choose the device you would like to program and check the **Enable Device** checkbox.
2. In the Action column, click the down arrow to expose the drop down menu (see figure below).
3. Select your desired action.

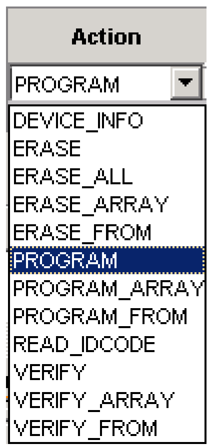


Figure 107 · Drop Down Menu for Select Action

Before you can enable serialization, you must check the **Enable Device** checkbox.

To enable Serialization:

1. Check the **Enable Serial** checkbox. By enabling serialization, the action options change.
2. In the **Action** column, click the down arrow to expose the drop down menu (see figure below).

Figure 108 · Drop Down Menu for Select Action

3. Select your desired action.

Serial Data Column

4. Click the **Select** button from the **Serial Data** column, which is next to the **Action** column (see above figure). The **Serial Settings** dialog box displays.

5. Choose your serial settings from the **Serial Settings** dialog box.

See [Serial Settings](#) for more information about this topic.

Note: Uncheck the Enable Serial checkbox to disable serialization.

To edit the Chain Configuration Grid:

1. Select the device you would like to edit and right click anywhere in the row of the selected device.

2. Select and click an option from the right-click menu.

Note: The Device Configuration menu (see figure below) includes options for configuring your device.

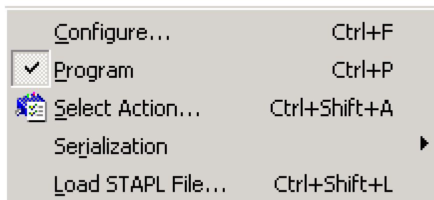


Figure 109 · Device Configuration Menu

Chain Editing

The chain order is located in the [Chain Configuration Window](#) (see figure below). The devices you add to the chain must be in the correct order and must match the physical chain to be programmed.

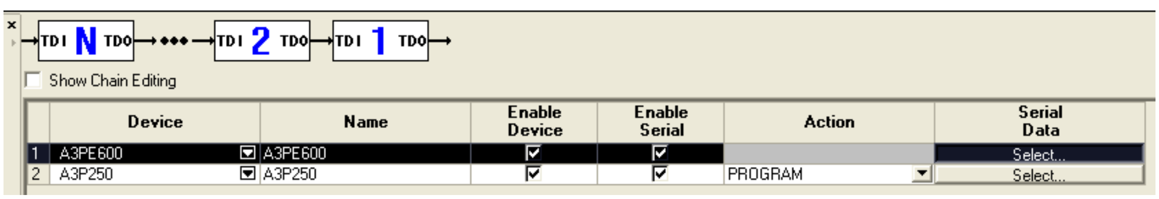


Figure 110 · Chain Configuration Window

Check the **Show Chain Editing** checkbox to display chain editing options (**Add Actel Device**, **Add Non-Actel Device**). See figure below.

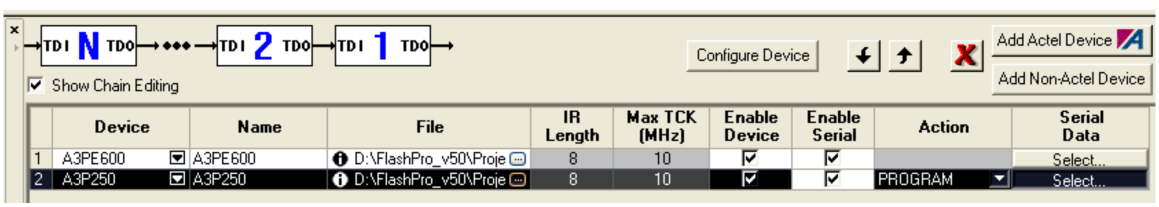


Figure 111 · Chain Configuration Window

You can edit the chain by adding Actel and Non-Actel Devices, for information refer to:

- [Adding Actel Devices](#)
- [Adding Non-Actel Devices](#)
- [Adding Actel Devices from a STAPL File](#)
- [Automatic Chain Construction Tutorial](#)

- Chain Programming Tutorial

Using the Organize Buttons in the Chain Programming Grid

The organize buttons enable you to select the order of the devices in your **Chain Programming** grid (see figure below).

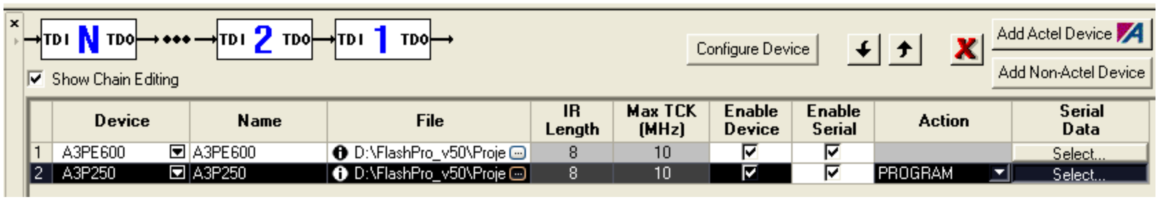


Figure 112 · Chain Configuration Window (Displaying Organize Buttons)

You can move devices up and down or delete devices within the grid. See the table below for a description of each button.

Table 4 · Organize Buttons

Button	Description
	Moves your device up in the Chain Programming grid.
	Moves your device down in the Chain Programming grid.
	Deletes your device from the Chain Programming grid.

Cutting, Copying and Pasting Devices from the Chain

If you want to make changes to your chain, you must make these changes from the spreadsheet in the Chain Programming grid.

To copy or cut a device from the chain programming grid:

1. Select the device you would like to edit and right click anywhere in the row of the selected device. The right-click menu appears.
2. Select **Copy** or **Cut** from the right-click menu to copy your device.

To paste a device from the chain programming grid:

1. Right-click the location where you would like to **Paste** the device.
2. Select **Paste** from the right-click menu .

Removing Devices from the Chain

If you want to make changes to your chain, you must make these changes from the spreadsheet in the Chain Programming grid.

To remove a device from the chain programming grid:

1. Select the device you would like to remove and right click anywhere in the row of the selected device. The right-click menu appears.
2. Choose**Remove** from the right-click menu to delete your device.

Moving Devices within the Chain

You can move devices within the chain by using the Organize buttons (located next to the Add Actel and Add Non-Actel Device buttons) in the Chain Programming grid (as shown in the figure below).

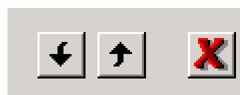


Figure 113 · Organize Buttons

To move or delete a device within the chain:

1. Click a device to select it.
2. Click one of the **Organize** button arrows to move your device up or down the spreadsheet. Click the delete button (red X) to remove a device.

For more information about the Organize buttons, see Using the Organize buttons in the Chain Programming grid.

Skip Serial Data

If you are unable to perform the programming action on your device, if your device fails to program, and you have selected the **Skip Serial Data** serialization setting, the software automatically uses the next serial data when you program the next device. By default, the software is set to **Skip Serial Data**.

You can change the serialization setting by selecting **Tools** > **Serialization** or you can click the **Skip serial data** icon or the **Reuse serial data** icon from the toolbar.

Reuse Serial Data

If your device fails to program, and you have selected the **Reuse Serial Data** serialization setting, the software automatically reuses the current serial data when you program the next device.

Note: The FlashPro default setting is Skip Serial Data.

You can change the serialization setting by selecting **Tools** > **Serialization** or you can click the **Skip serial data** icon or the **Reuse serial data** icon from the toolbar.

Serialization with Parallel Programming

When programming the multiple ProASIC3 devices in parallel, while performing serialization at the same time, each target device is assigned a Serial Index/Action for each programming run. Upon each successful completion of each programming run, a new index is assigned to the each target device for the next programming run. This process continues until the selected **Serial Indices/Actions** are exhausted.

Note: If programming failure is encountered, depending on the user setting, the failed serial data may be reused or skipped in the next programming run.

If, in the last programming run, the remaining number selected Serial Indices/Actions is less than the number of targeted ProASIC3 devices, the targeted devices without an assigned Serial Index/Action are skipped in the final serial programming run.

Chain Editing

Adding an Actel Device

To add an Actel device:

1. Click the **Add Actel Device** button from the **Chain Programming** grid. The **Add Actel Device** dialog box appears (see figure below).

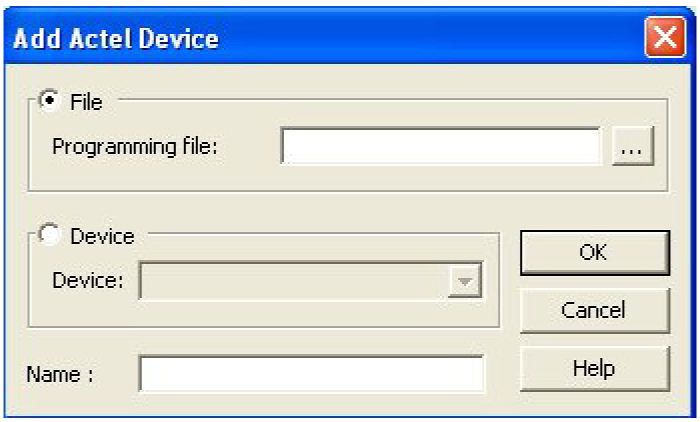


Figure 114 · Add Actel Device

2. Click the **Device** radio button and choose your device from the **Device** list drop-down menu.
3. Click the **File** radio button, then click the **Browse** button in the **Programming file** text box to find your PDB/STAPL file. The **Use File** dialog box appears.
4. Find your PDB/STAPL file and click **Open**. Your PDB/STAPL file name appears in the **Name** text box. You can change the name by clicking in the text box.
5. Click **OK**. Your device displays in the **Chain Programming** grid.

You can also add an Actel device from **Configuration** > **Add Actel Device**.

Adding an Actel Device from Files

To add an Actel device from a file:

1. From the **Configuration** menu, choose **Add Actel Devices From Files**. The **Add Actel Devices From Files** dialog box appears.
2. Locate your file and click **Open**. Your device displays in the **Chain Programming** grid.

Adding a Non-Actel Device

When adding a non-Actel device, you must choose either a BSDL file or customize the Instruction Register (IR) length and the Max TCK frequency of the device.

IR Length

The IR length specifies the number of IR bits in a specific device.

Max TCK Frequency

Maximum clock frequency to program a specific device. FlashPro uses this information to ensure that the programmer operates at a frequency lower than the slowest device in the chain.

BSDL File

Boundary Scan Description Language (BSDL) files describe the characteristics of a specific device. When using a BSDL file, FlashPro extracts the IR length and TCK frequency for the specific device and uses the information to build the FlashPro STAPL file. If you do not have a BSDL file for your specific device, you must manually enter the IR length and Max TCK for your device. This information should be found in the datasheet for the device.

To add a non Actel device using a BSDL file:

1. Click the **Add Non-Actel Device** button in the **Chain Programming** window. The **Add Non-Actel Device** dialog box appears (see figure below).

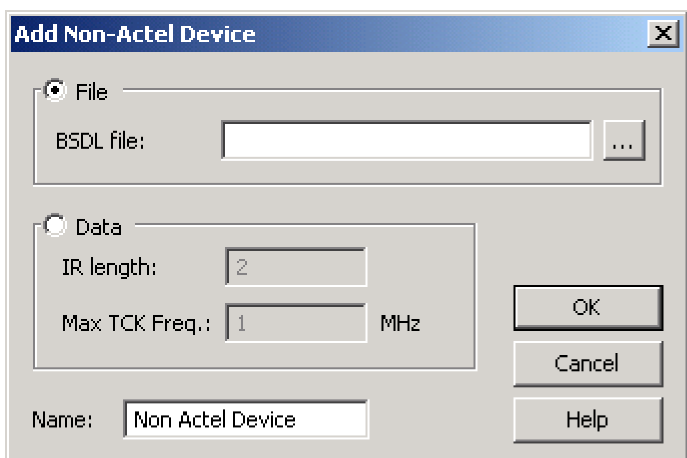


Figure 115 · Add Non-Actel Device Dialog Box

2. Type in the BSDL file or locate it by clicking the **Browse** button. If you click the **Browse** button to find your BSDL file, the **Use File** dialog box displays.
3. Select your BSDL file from the **Use File** dialog box, and click **Open**.
4. Click **OK**, and your device appears in the **Chain Programming** grid.

When closing the **Add Non-Actel Device** window, if you specified a BSDL file, it is parsed and its IR length and Max TCK frequency are retrieved.

Note: If you select a BSDL file, you cannot specify an IR length and Max TCK frequency.

To add a Non-Actel device using an IR length and a Max TCK frequency:

1. Click the **Add Non-Actel Device** button from the **Chain Programming** window.
2. Click the **Data** option from the **Add Non-Actel Device** dialog box.
3. Enter the IR length **AND** the Max TCK frequency in MHz.
4. Click **OK**.

If you decide to use custom data, you must specify both an IR length and Max TCK frequency.

Note: The IR length must be an integer greater than or equal to 2, and the Max TCK frequency must be a float greater than or equal to 1.

Non-Actel Device Configuration Dialog Box

It is possible to add multiple BSDL files to your Non-Actel device database that have the same IDCODE. If the BSDL files list the same IR length but different TCK values, FlashPro automatically chooses the file with the lowest TCK value by default and no action is required. If the IR lengths are different you receive an error message asking you to resolve the conflict.

To resolve the issue, click the drop-down arrow adjacent to the device name. This opens the Non-Actel Device Configuration dialog box (as shown in the figure below). From here you can choose the device that you wish to use. Select the device from the dropdown menu and enter a new name or use the default.

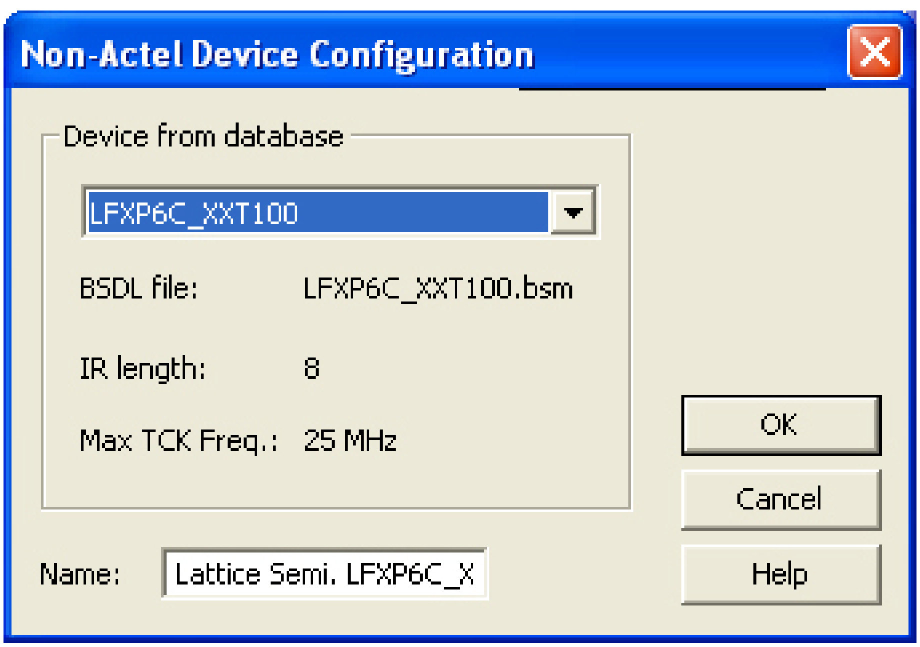


Figure 116 · Non-Actel Device Configuration Dialog Box

Configuring a Programmer

Selecting an Action

The actions you have available are based upon what type of STAPL file you have loaded into the software.

To configure a programmer:

1. From the **Configure** menu, choose **Select Action and Procedures**. The **Select Action and Procedures** dialog box appears (see figure below).

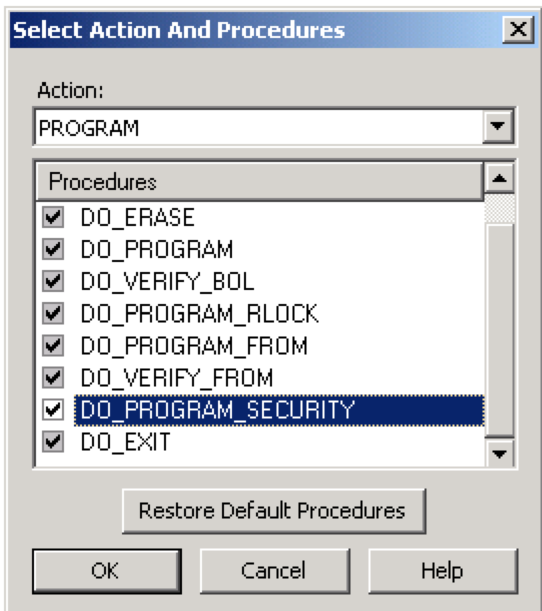


Figure 117 · Select Action of Procedures Dialog Box />

2. Click the checkboxes of the available procedures or click the **Restore Default Procedures** button in the **Select Action and Procedures** dialog box.
3. Click **OK**.

Click the **Restore Default Procedures** to return to default settings.

Using Serialization

To use serialization:

1. Enable serialization by checking the **Serialization** checkbox.
2. Select an action in the **Action** text box.
3. Click the **Select Serialization Indexes** button. The **Serial Settings** dialog box displays (see figure below).

Note: Note: Depending on the STAPL file format (Actel format or generic format) used, you will either see Indexes columns or Actions columns in the Serial Settings dialog box.

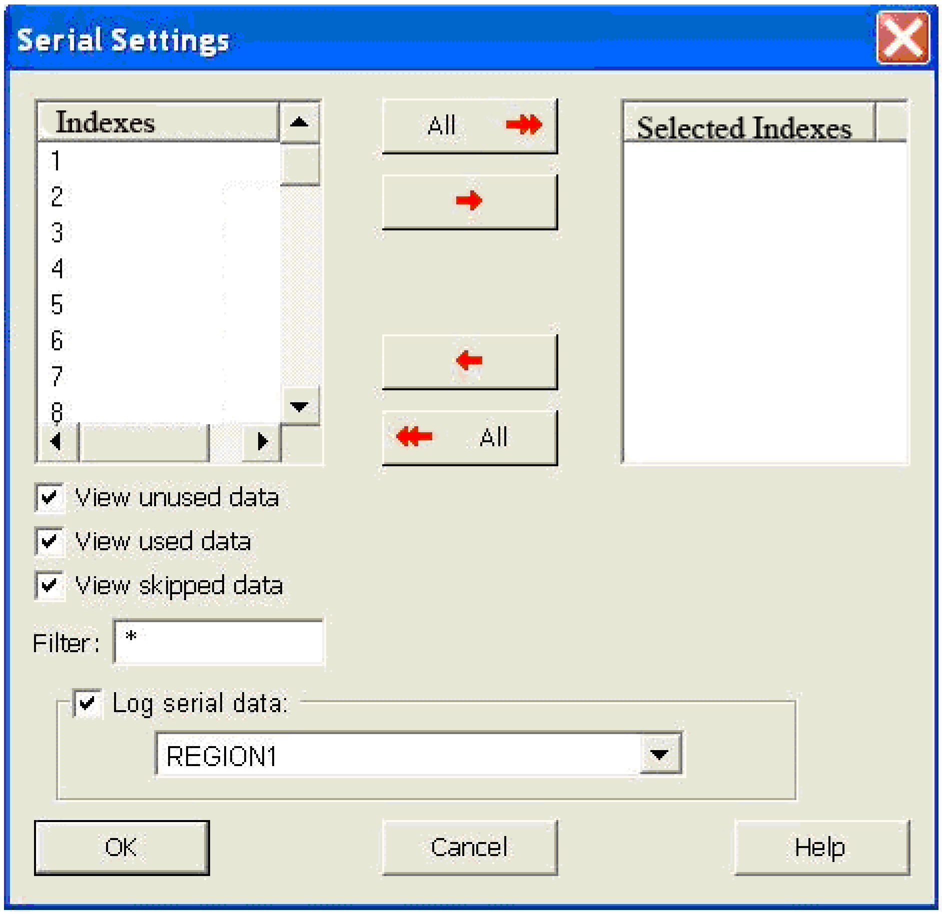


Figure 118 · Serial Settings Dialog Box

Note: Depending on your STAPL file, you would click the Select Serialization Action button (see figure below).

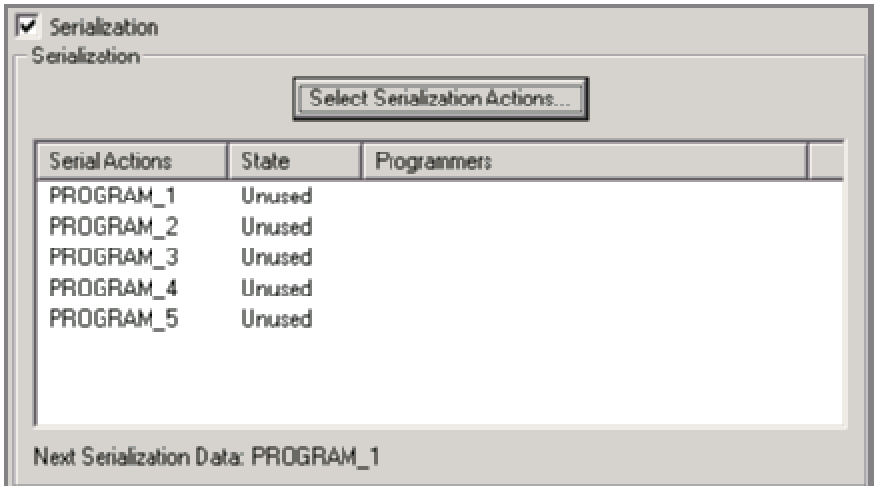


Figure 119 · Select Serialization Actions Button

Modifying Programming Settings in FlashPro with a PDB File

FlashPro enables you to modify programming settings within the software by using a PDB file. This feature is available only for PDB files generated from Designer v8.1 or greater. This feature allows modification of features being programmed, security settings, and memory content update for FlashROM and Embedded Flash Memory Blocks (Fusion only). Please refer to Modifying Memory Contents and Programming a Device Tutorial (EFMB) and Modifying FlashROM Contents and Programming a Device Tutorial for an example.

Note: You cannot add or remove the FPGA feature from your PDB. If you would like to add or remove this feature from your PDB, regenerate the PDB from Designer.

See Also

Configuring security, FlashROM and Embedded Flash Memory Block settings in FlashPro

Configuring Security

Configuring Security, FlashROM and Embedded Flash Memory Settings in FlashPro

1. From the **Configuration** menu, choose **Load Programming File (PDB).**
2. Select the PDB file and click **Open,** this loads the programming file.
3. From the **Configuration** menu, choose **PDB Configuration.** The **Programming File Generator** appears (see figure below).

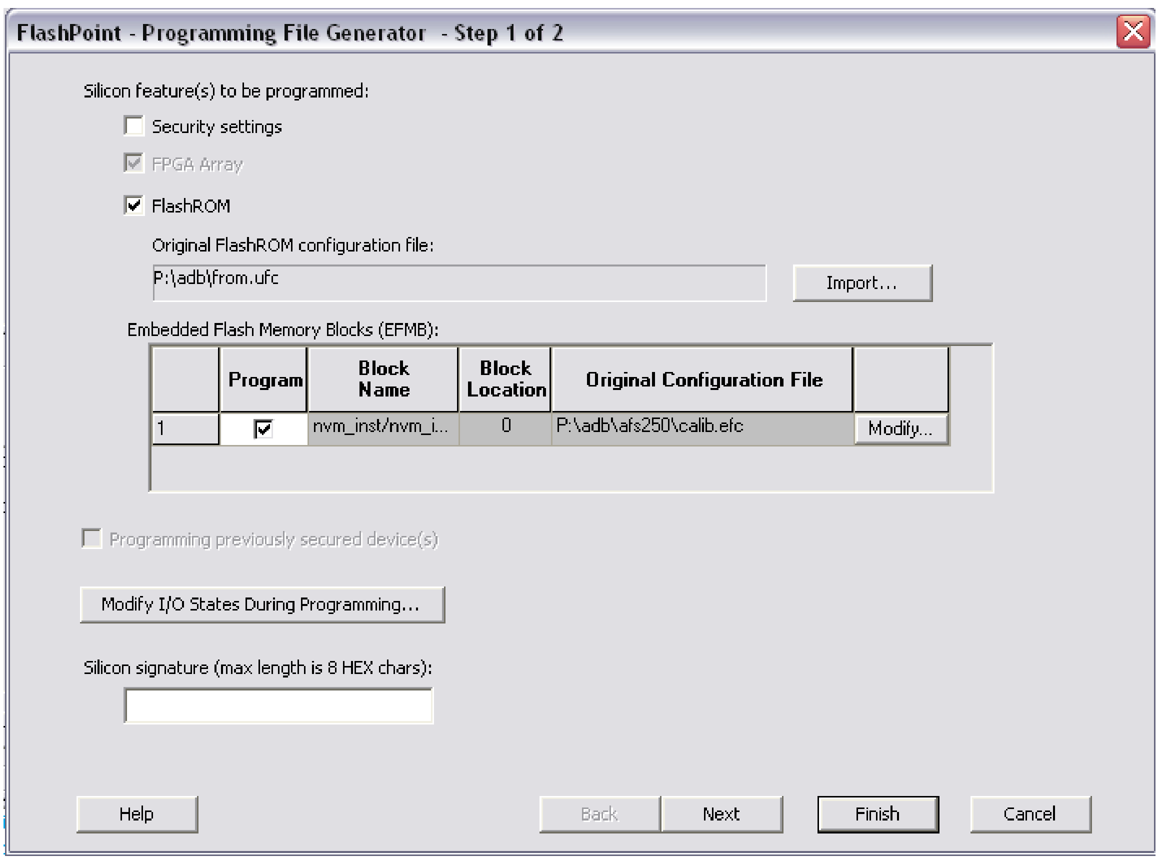


Figure 120 · Programming File Generator

4. Select the Silicon feature(s) you want to program:

- Security settings
- FlashROM
- Embedded Flash Memory Block

5. Check the **Programming previously secured device(s)** box if you are reprogramming a device that has been secured.

Because the IGLOO, ProASIC3, SmartFusion and Fusion families enable you to program the Security Settings separately from the FPGA Array and/or FlashROM, you must indicate if the Security Settings were previously programmed into the target device. This requirement also applies when you generate programming files for reprogramming.

6. Enter the **Silicon signature** (0-8 HEX characters).
7. Click **Next**.

Configuring Security Settings in FlashPro

To configure the security settings:

1. From the **Configuration** menu, choose **Load Programming File (PDB)**.
2. Select the PDB file and click **Open**, this loads the programming file.
3. From the **Configuration** menu, choose **PDB Configuration**. The **Programming File Generator** appears (as shown in the figure below).

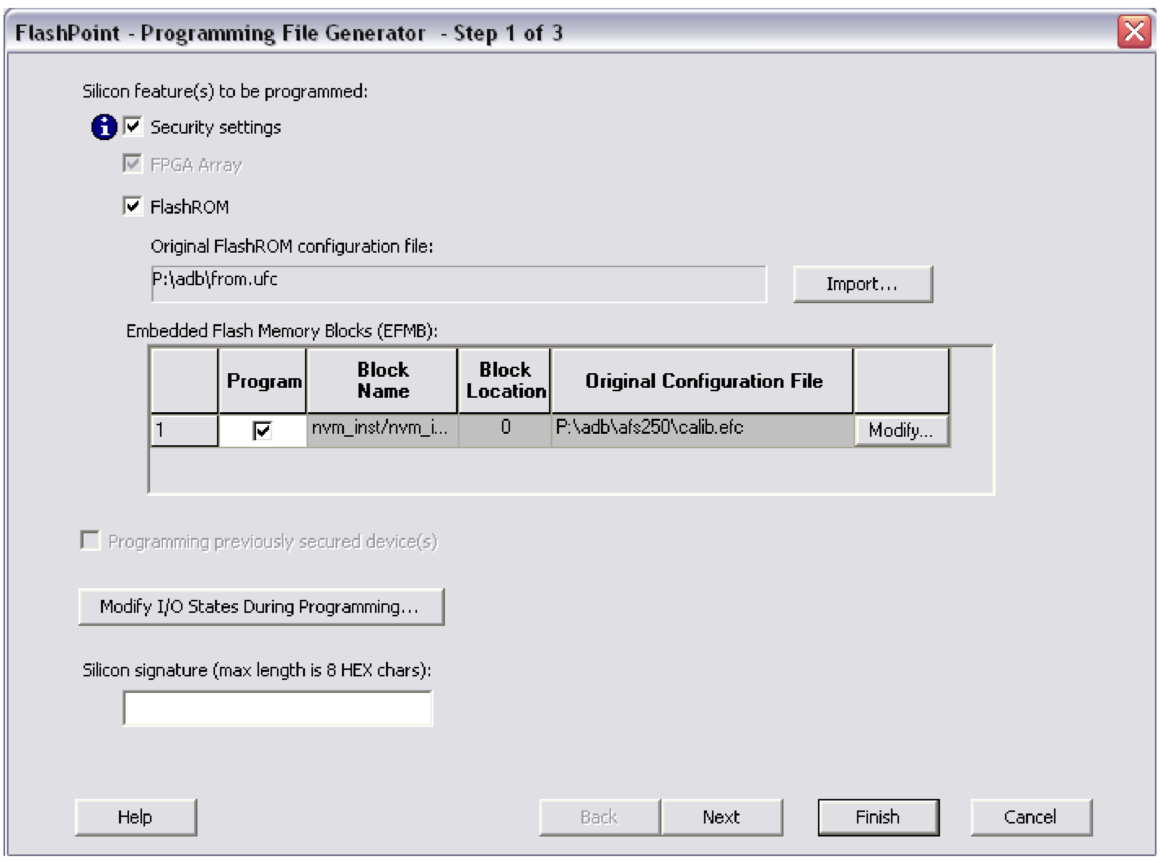


Figure 121 · Programming File Generator

4. Check the **Security Settings** checkbox and click **Next**. This brings up the **Security Settings** dialog box (shown below).

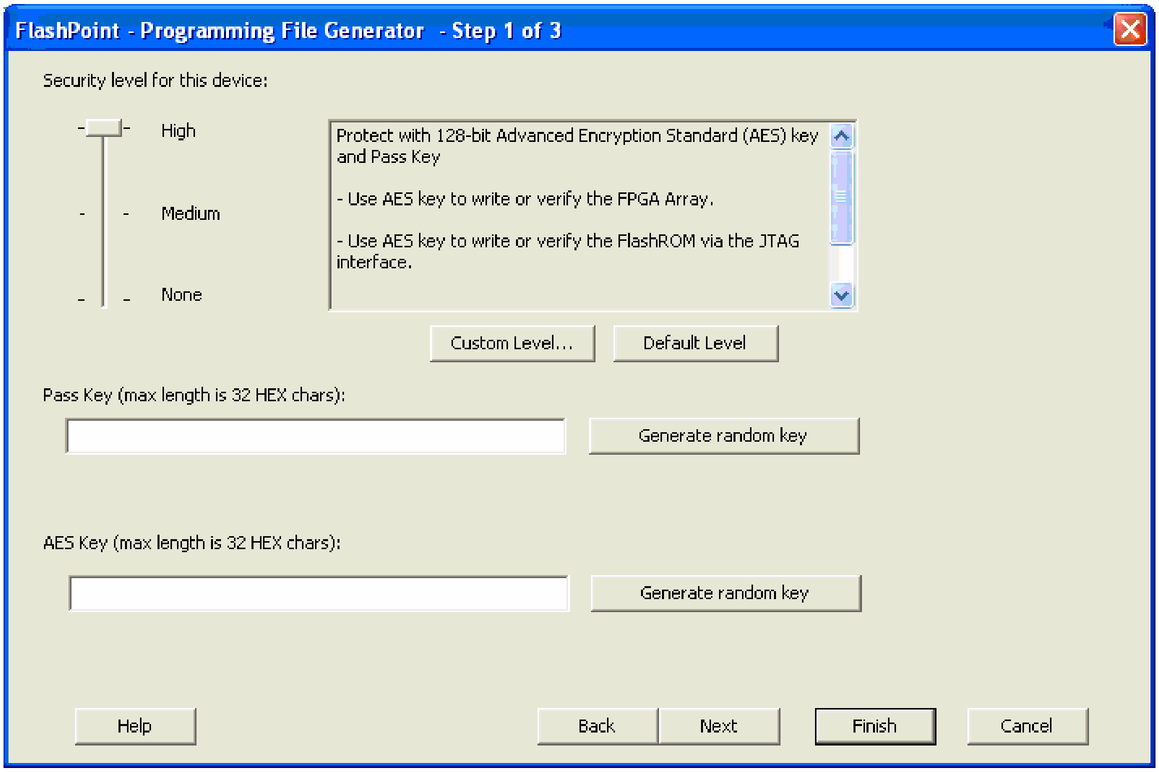


Figure 122 · Security Settings Dialog Box

5. Move the sliding bar to select the security level for FPGA, FlashROM, and EFMB (see table for a description of the security levels).

Table 5 · Security Levels Descriptions

Security Level	Security Option	Description
High	Protect with a 128-bit Advanced Encryption Standard (AES) key and a Pass Key	Access to the device is protected by an AES Key and the Pass Key. The Write and Verify operations of the FPGA Array use a 128-bit AES encrypted bitstream. From the JTAG interface, the Write operation of the FlashROM uses a 128-bit AES encrypted bitstream. Read back of the FlashROM content via the JTAG interface is protected by the Pass Key. Read back of the FlashROM content is allowed from the FPGA Array.
Medium	Protect with Pass Key	The Write and Verify operations of the FPGA Array require a Pass Key. From the JTAG interface, the Read and Write

Security Level	Security Option	Description
		operations on the FlashROM content require a Pass Key. You can Verify the FlashROM content via the JTAG interface without a Pass Key. Read back of the FlashROM content is allowed from the FPGA Array.
None	No Security	The Write and Verify operations of the FPGA Array do not require keys. The Read, Write, and Verify operations of the FlashROM content also do not require keys.

6. Enter the **Pass Key** and/ or the **AES Key** as appropriate. You can generate a random key by clicking the Generate random key button.
   The **Pass Key** protects all the Security Settings for the FPGA Array and/or FlashROM.
   The **AES Key** decrypts FPGA Array and/or FlashROM programming file content. Use the AES Key if you intend to program the device at an unsecured site or if you plan to update the design at a remote site in the future.
7. Click **Finish**.

You can also customize the security levels by clicking the Custom Level button.

Custom Security Settings

For advanced use, you can customize your security levels.

To set custom security levels:

1. Click the **Custom Level** button in the **Setup Security** page. The **Custom Security** dialog box appears (see figure below).

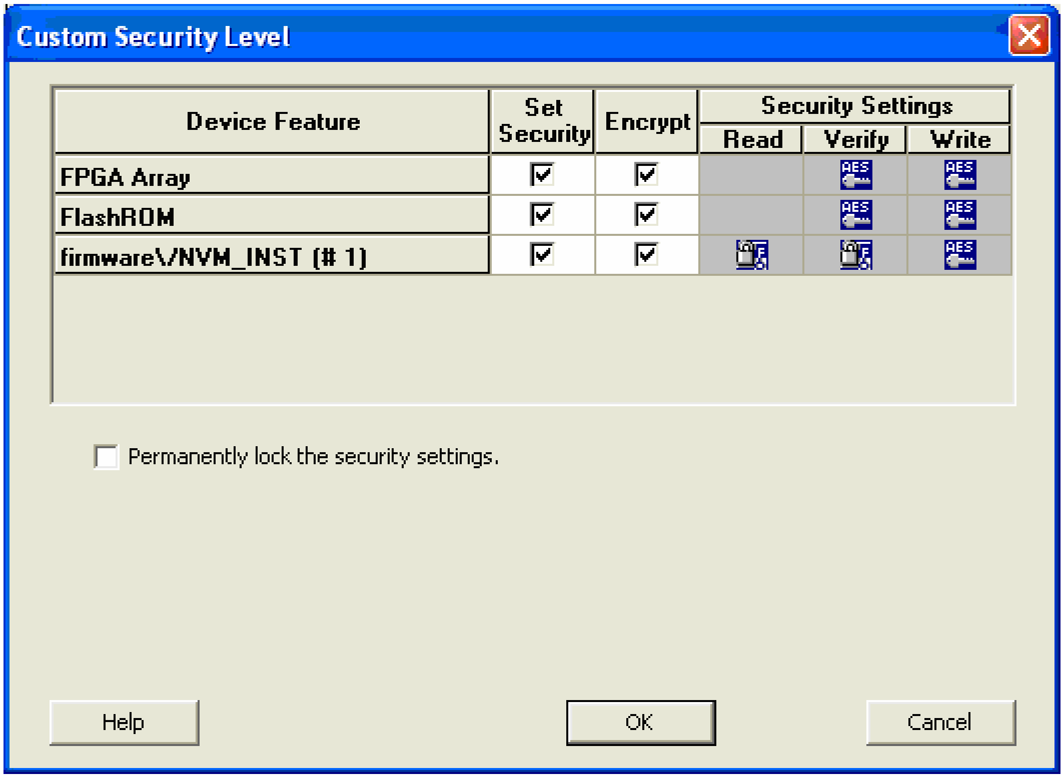


Figure 123 · Custom Security Level

2. Select the **FPGA Array Security,** the **FlashROM Security**, and Embedded Flash Memory block levels.

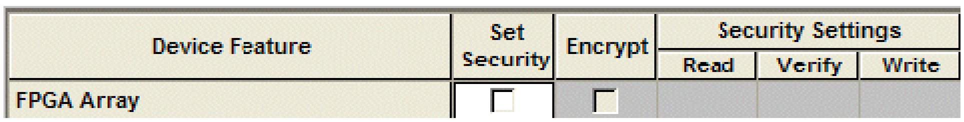
The silicon features can have different Security Settings. See the tables below for a description of the custom security option levels for FPGA Array, FlashROM, and Embedded Flash Memory block.

Table 6 · FPGA Array

Security Option	Description
Lock for both writing and verifying 	Allows writing/erasing and verification of the FPGA Array via the JTAG interface only with a valid Pass Key.
Lock for writing	Allows the

<table>
<tr><th>Security Option</th><th>Description</th></tr>
<tr><td>
</td><td>writing/erasing of the FPGA Array only with a valid Pass Key. Verification is allowed without a valid Pass Key.</td></tr>
<tr><td>Use the AES Key for both writing and verifying

</td><td>Allows the writing/erasing and verification of the FPGA Array only with a valid AES Key via the JTAG interface. This configures the device to accept an encrypted bitstream for reprogramming and verification of the FPGA Array. Use this option if you intend to complete final programming at an unsecured site or if you plan to update the design at a remote site in</td></tr>
</table>

Security Option	Description
	the future. Accessing the device security settings requires a valid Pass Key.
Allow write and verify 	Allows writing/erasing and verification of the FPGA Array with plain text bitstream and without requiring a Pass Key or an AES Key. Use this option when you develop your product in-house.

Note: The ProASIC3 family FPGA Array is always read protected regardless of the Pass Key or the AES Key protection.

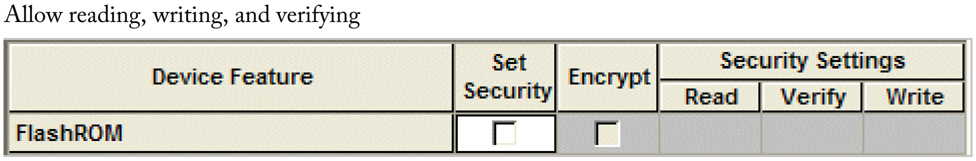
Table 7 · FlashROM

Security Option	Description
Lock for both reading and writing 	Allows the writing/erasing and reading of the FlashROM via the JTAG interface only with a valid Pass Key. Verification is allowed without a valid Pass Key.

<table>
<tr><th>Security Option</th><th>Description</th></tr>
<tr><td>Lock for writing

</td><td>Allows the writing/erasing of the FlashROM via the JTAG interface only with a valid Pass Key. Reading and verification is allowed without a valid Pass Key.</td></tr>
<tr><td>Use the AES Key for writing

</td><td>Allows the writing/erasing of the FlashROM via the JTAG interface only with a valid AES Key. This configures the device to accept an encrypted bitstream for reprogramming of the FlashROM. Use this option if you complete final programming at an unsecured site or if you plan to update the design at a remote site in the future.

Note: The bitstream that is read back from the FlashROM is always unencrypted (plain text).</td></tr>
<tr><td>Allow reading, writing, and verifying

</td><td>Allows writing/erasing, reading and verification of the FlashROM content with a plain text bitstream</td></tr>
</table>

Security Option	Description
	and without requiring a valid Pass Key or an AES Key.

Note: The FPGA Array can always read the FlashROM content regardless of these Security Settings.

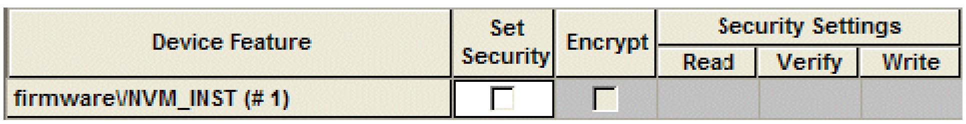
Table 8 · Embedded Flash Memory Block

<table>
<tr><th>Security Option</th><th>Description</th></tr>
<tr><td>Lock for reading, verifying, and writing

</td><td>Allows the writing and reading of the Embedded Flash Memory Block via the JTAG interface only with a valid Pass Key. Verification accomplished by reading back and compare.</td></tr>
<tr><td>Lock for writing

</td><td>Allows the writing of the Embedded Flash Memory Block via the JTAG interface only with a valid Pass Key. Reading and verification is allowed</td></tr>
</table>

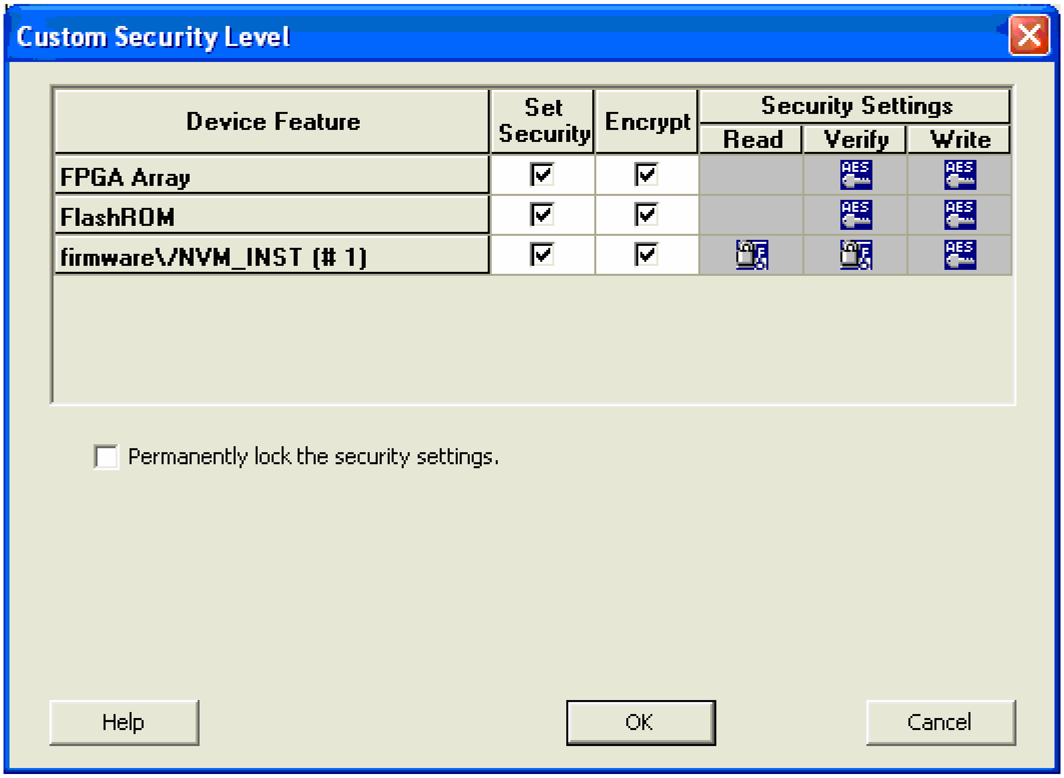
Security Option	Description
	without a valid Pass Key.
Use AES Key for writing 	Allows the writing of the Embedded Flash Memory Block via the JTAG interface only with a valid AES Key. This configures the device to accept an encrypted bitstream for reprogramming of the Embedded Flash Memory Block. Use this option if you complete final programming at an unsecured site or if you plan to update the design at a remote site in the future. The bitstream that is read back from the Embedded Flash Memory

Security Option	Description
	Block is always unencrypted (plain text), when a valid pass key is provided.
Allow reading, writing, and verifying 	Allows writing, reading and verification of the Embedded Flash Memory Block content with a plain text bitstream and without requiring a valid Pass Key or an AES Key.

3. To make the Security Settings permanent, select the **Permanently lock the security settings** check box. This option prevents any future modifications of the Security Setting of the device. A Pass Key is not required if you use this option.

Note: Note: When you make the Security Settings permanent, you can never reprogram the Silicon Signature. If you lock the write operation for the FPGA Array or the FlashROM, you can never reprogram the FPGA Array or the FlashROM, respectively. If you use an AES key, this key cannot be changed once you permanently lock the device.

To use the Permanent FlashLock™ feature, select **Lock for both writing and verifying** for FPGA Array and **Lock for both reading and writing** for FlashROM and select the **Permanently lock the security settings** checkbox as shown in the figure below. This will make your device one-time-programmable.



Custom Security Level- Permanent Lock

4. Click the **OK** button. The **Security Settings** page appears with the **Custom security setting** information.

Configuring FlashROM Settings in FlashPro

To configure the FlashROM settings:

1. From the **Configuration** menu, choose **Load Programming File (PDB).**
2. Select the PDB file and click **Open,** this loads the programming file.
3. From the **Configuration** menu, choose **PDB Configuration**. The **Programming File Generator** appears.
4. Check the **FlashROM** checkbox and click **Browse** to load a FlashROM configuration file. Click **Next**. This brings up the **FlashROM Settings** dialog box (see figure below).

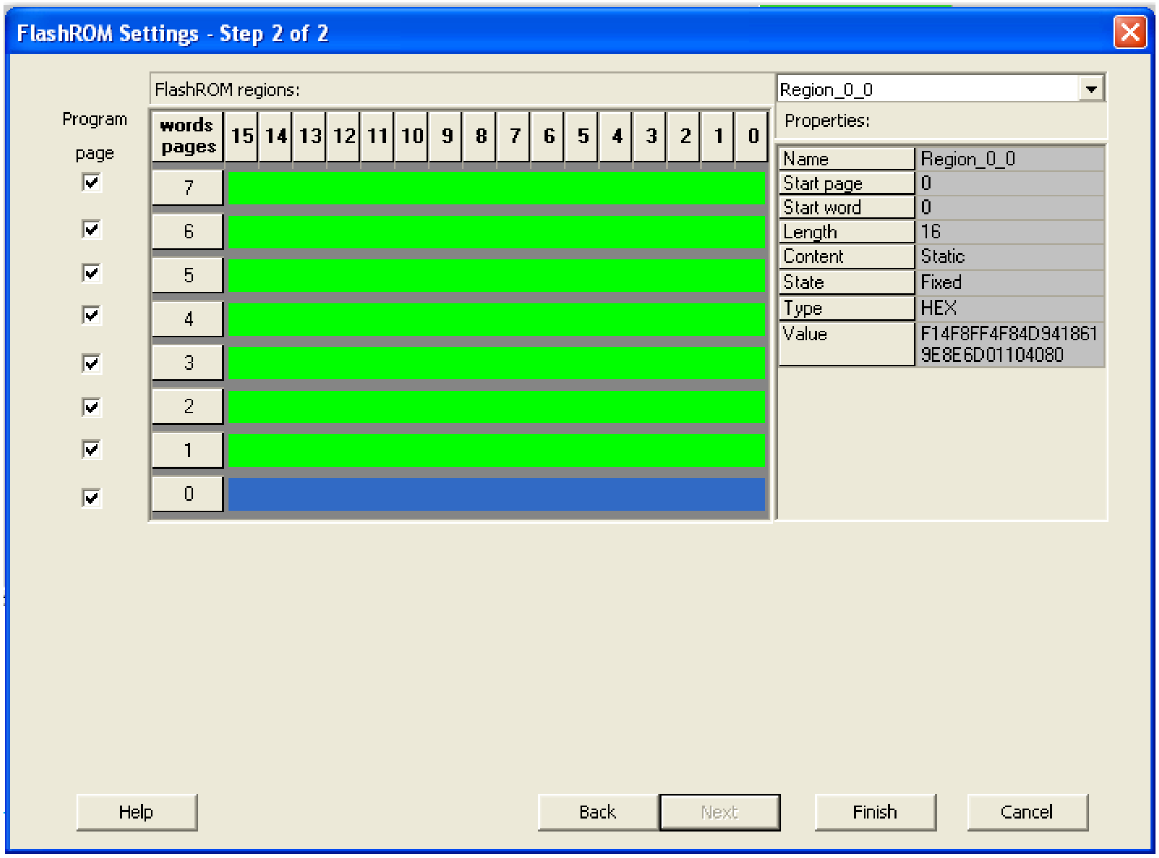


Figure 124 · FlashROM Settings Dialog Box

5. Select the FlashROM memory page that you want to program.
6. Enter the data value for the configured regions.
7. If you selected the region with a **Read From File**, specify the file location.
8. If you selected the **Auto Increment** region, specify the **Start** and **Max** values.
9. Enter the number of devices you want to program.
10. Click the **Target Programmer** button. The **Select Programmer Type** dialog box appears.
11. Click **Finish**. FlashPoint generates your programming file.

Note: You cannot change the FlashROM region configuration from FlashPoint. You can only change the configuration from the FlashROM core generator.

Express Configuration

The express configuration feature in FlashPro allows you to set the security settings as well as FlashROM content without a design. This allows the production flow to be executed in parallel to the design effort if needed.

For example, you can pre-program the security settings with the **High Security** setting and serialize the device using FlashROM without the FPGA design in a secured programming environment. The FPGA Array and EFMB design can be programmed in unsecured programming environment using encrypted programming file. Refer to Programming Only Security Settings Tutorial for more information.

Note: This feature is only available for IGLOO, ProASIC3, SmartFusion and Fusion devices.

IGLOO and ProASIC3 Programming

Programming File Actions for IGLOO and ProASIC3 Devices

IGLOO and ProASIC3 devices support the following features:

- Security settings
- FPGA Array
- FlashROM

You can program these features separately or together using different programming files or by using one programming file.

Note: When we specify a family name, we refer to the device family and all its derivatives, unless otherwise specified. See the Supported Families topic for a complete list of families and their derivatives.

The STAPL files for IGLOO and ProASIC3, excluding ProASIC3L, devices include actions targeted at one, two, or all three of the IGLOO and ProASIC3 features (FPGA Array and FlashROM and Security Settings). The combinations of the features you selected to target results in different actions that are available in the STAPL file. See the following table for an illustration.

Figure 125 · IGLOO and ProASIC3 Device Programming Actions

Features	Features Selected									
FPGA Array	X				X	X		X	X	
FlashROM		X			X		X	X		X
Security			X			X	X		X	X
	STAPL Actions Available (correspond with Features Selected above)									
PROGRAM	X	X			X	X	X	X	X	X
VERIFY	X	X			X	X	X	X	X	X
ERASE	X	X			X	X	X	X	X	X
ERASE_ALL	X	X	X	X	X	X	X	X	X	X
DEVICE_INFO	X	X	X	X	X	X	X	X	X	X
READ_IDCODE	X	X	X	X	X	X	X	X	X	X
ERASE_FROM		X			X		X	X		X
PROGRAM_FROM										

Features	Features Selected									
ERASE_ARRAY	X				X	X		X	X	
VERIFY_ARRAY	X				X	X		X	X	
ENC_DATA_AUTHENTICATION	X				X	X		X	X	
PROGRAM_SECURITY			X			X	X		X	X
ERASE_SECURITY			X			X	X		X	X

Note: The ENC_DATA_AUTHENTICATION Action is only available when you choose encrypted programming.

Programming Actions

See the table below for a list of all the actions for the programming file.

Table 9 · Programming File Actions

Action	Description
PROGRAM	Programs all selected family features: FPGA Array, targeted FlashROM pages, security setting and silicon signature (if provided).
VERIFY	Verifies all selected features: FPGA Array, targeted FlashROM pages, security setting and silicon signature (if provided).
ERASE	Erases all selected family features: FPGA Array, targeted FlashROM pages, security setting and silicon signature (if provided).
ERASE_ALL	Erases all features in the targeted device regardless of the features selected to generate the STAPL file.
DEVICE_INFO	Displays the IDCODE, Silicon Signature, the design name, the checksum, and device security settings and programming environment information programmed into the device.
READ_IDCODE	Reads the device ID code from the device.
ERASE_FROM	Erases only the targeted FlashROM pages, not the entire

Action	Description
	FlashROM.
PROGRAM_FROM	Programs only the targeted FlashROM pages.
PROGRAM_ARRAY	Programs the FPGA Array and Silicon Signature (if applicable) into the device.
VERIFY_ARRAY	Verifies the FPGA Array and Silicon Signature (if applicable) into the device.
ERASE_ARRAY	Erases the FPGA Array and Silicon Signature (if provided).
PROGRAM_SECURITY	Programs only the Security Settings.
ERASE_SECURITY	Erases only the Security Settings.

Note: FIX_INT_ARRAYS - This function is only applicable to STAPL file only. Depending on the STAPL player implementation, the indexing of an integer array may start from different direction. The STAPL standard did not clearly specify how it should be implemented. The FIX_INT_ARRAYS function detects the indexing implemented by the STAPL player and flips the content of the integer array if needed.

UNLOCK_UKEY: This function unlocks a secured device if it is locked by FlashLock.

Options available in Programming Actions

The table below shows the available actions in the programming file.

Table 10 · Programming File Actions

Action	Description
PROGRAM	When you target the Security Setting, you have the option of not erasing and programming the Security Setting by deselecting the following 2 procedures before executing the action. - SET_ERASE_SEC, - DO_PROGRAM_SECURITY. When you perform encrypted programming, you have the option of skipping the data authentication before programming by deselecting the DO_ENC_AUTHENTICATION procedure before executing the action.
ERASE	When you target the Security Setting, you have the option of not erasing the Security Setting by deselecting the SET_ERASE_SEC

Action	Description
	procedures before executing the action.
PROGRAM_ARRAY	When you perform encrypted programming, you have the option of skipping the data authentication before programming by deselecting the DO_ENC_AUTHENTICATION procedure before executing the action.

Note: The DO_ENC_AUTHENTICATION procedure prevents you from proceeding with encrypted programming with incorrect data due to corruption or an operator error. If incorrect data is detected during encrypted programming, the device will not be functional after programming.

SmartFusion and Fusion (AFS) Programming

Programming File Actions - SmartFusion and Fusion

FlashPro enables you to program security settings, FPGA Array, embedded flash memory blocks (EFMB), and FlashROM features for AFS device support. You can program these features separately using different programming files or you can combine them into one programming file.

The STAPL files for Fusion devices include actions targeted at one, two, or all four of the Fusion programming features: FPGA Array and FlashROM, Security Settings, and Embedded Flash Memory Block (EFMB). The combinations of the features you selected to target, results in different actions that are available in the STAPL file. See the following table for an illustration.

Figure 126 · SmartFusion and Fusion Programming File Actions

Programming Features	**Features Selected**												
FPGA Array	X	X			X			X	X		X	X	
FlashROM	X	X				X		X		X	X		X
Security	X	X	X				X		X	X		X	X
Embedded Flash Memory Block (EFMB)		X		X	X	X	X				X	X	X
	STAPL Actions Available (correspond with Features Selected above)												
PROGRAM	X	X			X	X		X	X	X	X	X	X
VERIFY	X	X			X	X		X	X	X	X	X	X
ERASE	X	X			X	X		X	X	X	X	X	X
ERASE_ALL	X	X	X	X	X	X	X	X	X	X	X	X	X
DEVICE_INFO	X	X	X	X	X	X	X	X	X	X	X	X	X
READ_IDCODE	X	X	X	X	X	X	X	X	X	X	X	X	X
ERASE_FROM		X				X		X		X	X		X

Programming Features	Features Selected												
PROGRAM_ARRAY	X				X			X	X		X	X	
VERIFY_ARRAY	X				X			X	X		X	X	
ENC_DATA_AUTHENTIFICATION	X				X			X	X		X	X	
PROGRAM_SECURITY			X				X		X	X		X	X
ERASE_SECURITY			X				X		X	X		X	X
VERIFY_SECURITY			X				X		X	X		X	X
PROGRAM_FP				X	X	X	X				X	X	X
VERIFY_FP				X	X	X	X				X	X	X
PROGRAM_NVM				X	X	X	X				X	X	X
VERIFY_NVM				X	X	X	X				X	X	X
PROGRAM_NVM_ACTIVE_ARRAY				X	X	X					X		
VERIFY_NVM_ACTIVE_ARRAY				X	X	X					X		
PROGRAM_NVM_ACTIVE_RSTM3				X	X	X					X		
RESET_CORTEXM3				X	X	X					X		

Note: The ENC_DATA_AUTHENTICATION Action is only available when you choose encrypted programming.

Note: PROGRAM_NVM_ACTIVE_ARRAY and VERIFY_NVM_ACTIVE_ARRAY actions are not available when the EFMB read/write/verify is locked with FlashLock.

STAPL Actions

See the table below for a list of all the actions for the STAPL file.

Figure 127 · STAPL File Actions

Action	Description
PROGRAM	Programs all selected family features: FPGA Array,

Action	Description
	targeted FlashROM pages, security setting and silicon signature (if provided).
VERIFY	Verifies all selected family features: FPGA Array, targeted FlashROM pages, security setting and silicon signature (if provided).
ERASE	Erases the selected family features.
ERASE_ALL	Erases all features in the targeted family device except Embedded Flash Memory Blocks, regardless of the features selected to generate the STAPL file.
DEVICE_INFO	Displays the IDCODE, Silicon Signature, the design name, the checksum, and device security settings and programming environment information programmed into the device.
READ_IDCODE	Reads the device ID code from the device.
ERASE_FROM	Erases only the targeted FlashROM pages, not the entire FlashROM.
PROGRAM_FROM	Programs only the targeted FlashROM pages.
PROGRAM_ARRAY	Programs the FPGA Array and Silicon Signature (if applicable) into the device.
VERIFY_ARRAY	Verifies the FPGA Array and Silicon Signature (if applicable) into the device.
ERASE_ARRAY	Erases the FPGA Array and Silicon Signature (if provided).
PROGRAM_SECURITY	Programs only the Security Settings.
ERASE_SECURITY	Erases only the Security Settings.
PROGRAM_NVM	Programs the targeted EFMBs.

Action	Description
VERIFY_NVM	Verifies the targeted EFMBs.
PROGRAM_NVM_ACTIVE_ARRAY	Programs the targeted EFMBs while the FPGA Array remains active.
VERIFY_NVM_ACTIVE_ARRAY	Verifies the targeted EFMBs while the FPGA Array remains active.
PROGRAM_NVM_ACTIVE_RSTM3	SmartFusion only; programs the eNVM while the core is active and then resets the CORTEX M3.
RESET_CORTEXM3	SmartFusion only; resets the CORTEX M3.

Options available in STAPL Actions

The table below shows the available actions in the STAPL file.

Table 11 · STAPL File Actions

Action	Description
PROGRAM	When you target the Security Setting, you have the option of not erasing and programming the Security Setting by deselecting the following 2 procedures before executing the action. - SET_ERASE_SEC, - DO_PROGRAM_SECURITY. When you perform encrypted programming, you have the option of skipping the data authentication before programming by deselecting the DO_ENC_AUTHENTICATION procedure before executing the action.
ERASE	When you target the Security Setting, you have the option of not erasing the Security Setting by deselecting the SET_ERASE_SEC procedures before executing the action.
PROGRAM_ARRAY	When you perform encrypted programming, you have the option of skipping the data authentication before programming by deselecting the DO_ENC_AUTHENTICATION procedure before executing the action.

Note: The DO_ENC_AUTHENTICATION procedure prevents you from proceeding with encrypted programming with incorrect data due to corruption or an operator error. If incorrect data is detected during encrypted programming, the device will not be functional after programming.

ProASIC^{PLUS} and ProASIC Families Programming Introduction

ProASIC^{PLUS} and ProASIC family devices support security settings and FPGA Array. You can program these features together using one programming file. The table below shows the STAPL actions for the FPGA Array only.

Table 12 · FPGA Array Only STAPL File

Action	Description
PROGRAM	Checks the security status of the device. If you are programming the security key, you may select to program only the FPGA Array by unselecting the PROGRAM_SECURITY procedure. If the device is programmed with the security key, then this command returns with Read inhibit:1 Write inhibit:1. If the security key is not present, the values are Read inhibit:0 Write inhibit:0.
ERASE_ARRAY	Erases the device.
READ_IDCODE	Reads the device ID code.
VERIFY	Verifies whether the device was programmed with the loaded STAPL file. Can be used to ensure that the bitstream programmed in the device is the same as the original STAPL file. If you load the wrong STAPL file, Exit 11 appears in the log window. A successful operation results in Exit 0.
PROGRAM	Programs the device.
DEVICE_INFO	Displays the serial number of the device, the Design Name that is programmed into the device, and the checksum that is programmed into the device.

Generating Programming Files

Generate a Programming File in FlashPoint

FlashPoint enables you to program security settings, FPGA Array, and FlashROM features for IGLOO, ProASIC3, SmartFusion, Fusion, and ProASIC family devices. You can program these features separately using different programming files or you can combine them into one programming file. Each feature is listed as a silicon feature in the GUI.

You can generate a programming file with one, two, or all of the silicon features from the Programming File Generator first page.

To generate a programming file:

1. Select the **Silicon feature(s)** you want to program.

- Security settings
- FPGA Array
- FlashROM

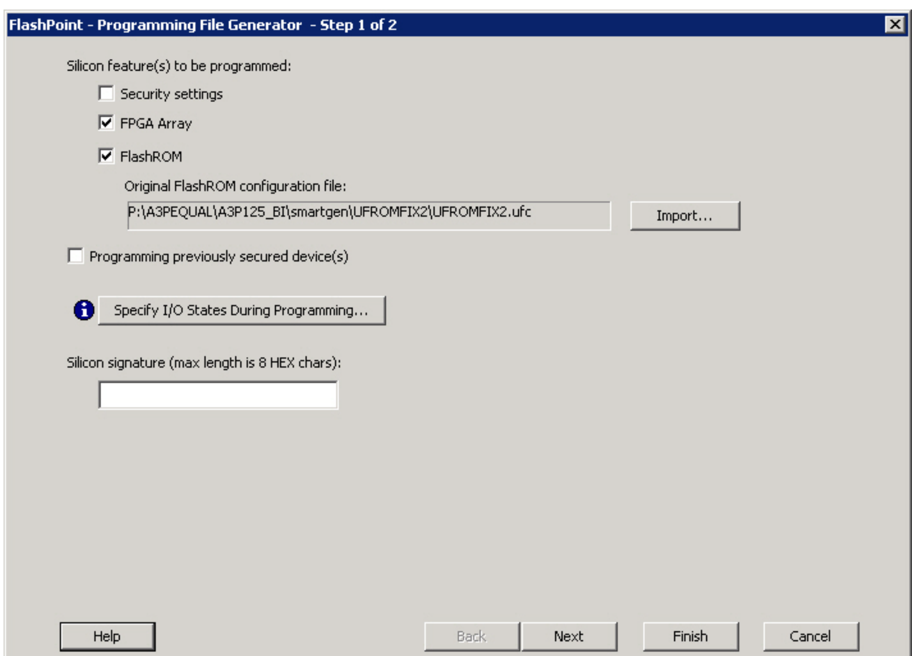


Figure 128 · Programming File Generator – Step 1 of 2

Note: Note: When FlashPoint is invoked for the first time, after netlist files are imported and the design is in post-layout state, the software retrieves the FlashROM and EFM blocks configuration files from the imported netlists and imports the configuration files. Otherwise, you need to import configuration files.

2. Click the **Programming previously secured device(s)** check box if you are reprogramming a device that has been secured.

Because the IGLOO, ProASIC3, SmartFusion, Fusion, and ProASIC families enable you to program the Security Settings separately from the FPGA Array and/or FlashROM, you must indicate if the Security Settings were previously programmed into the target device. This requirement also applies when you generate programming files for reprogramming.

3. Enter the silicon signature (0-8 HEX characters). See Silicon Signature for more information.
4. Depending upon the Silicon features you selected, click **Next** or **Finish**.

If you click **Next**, follow the instructions in the appropriate dialog box. If you click **Finish**, the **Generate Programming Files** dialog box appears (as shown in the figure below). Use this dialog box box to specify the programming file name, location, output format (STAPL file, SVF file, PDB file, DirectC DAT file, 1532 file), and, if necessary, limit the file size (as explained below).

Some testers may have memory size restrictions for a single SVF file. The SVF limit file option enables you to limit the size of each SVF file by either file size or vectors.

The generated SVF files append an index to the file name indicating the sequence of files. The format is:

```
<SVF_filename>_XXXXX.svf
```

where XXXXX is the index of the SVF file. The first SVF file begins with <SVF_filename>_00000.svf and increments by 1 until file generation is complete.

Maximum file size: Max file size limit for the SVF file; use this option to limit your SVF file size based on number of kB.

Maximum number of vectors: Max vector limit for the SVF file; use this option to limit the size of your SVF based on number of vectors.

For more information on DAT files, refer to the Data File Generator (DatGen) section of the *DirectC User's Guide*.

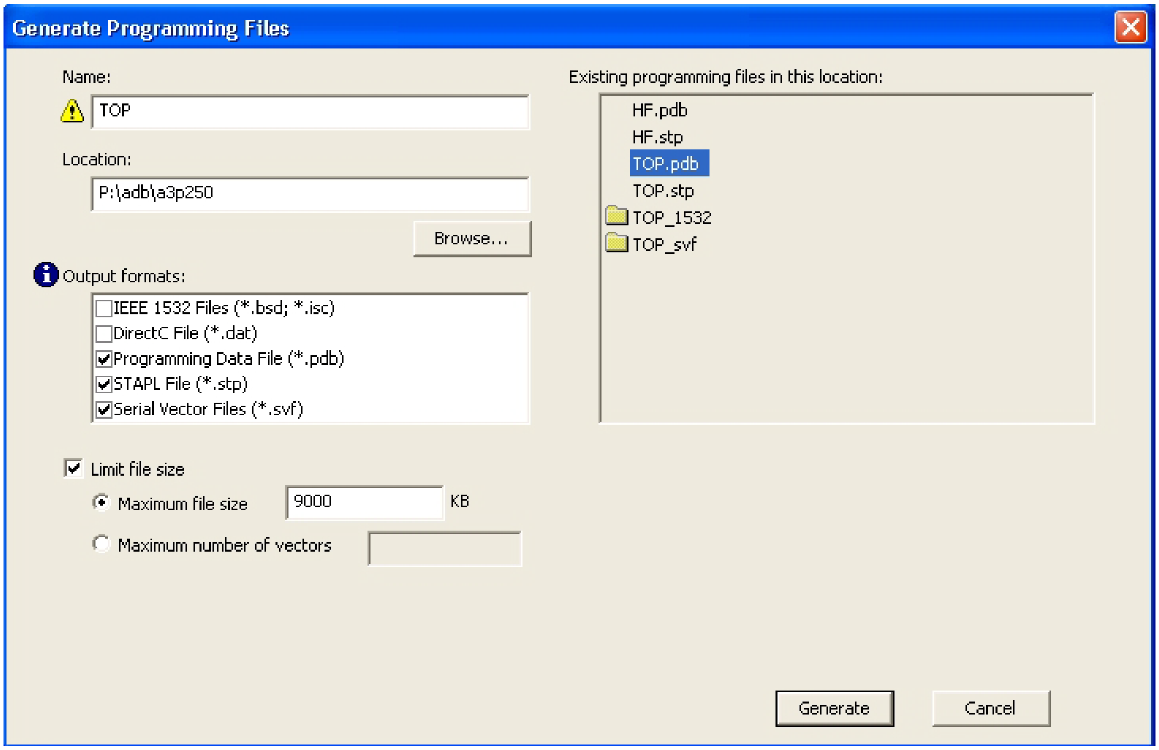


Figure 129 · Generate Programming Files Dialog Box (Flashpoint)

Generate a Programming File for SmartFusion

You can configure and generate a new PDB file from FlashPoint.

If you are using Single Mode, click **Create** to add a new PDB, or click **Modify** to make changes to a loaded PDB.

In Chain Mode, if you have not already done so, construct a chain and click **Create PDB** to create a new PDB for programming, or click **Modify PDB** to make changes to a loaded PDB.

FlashPoint enables you to specify your security settings and silicon features when you generate your programming file in SmartFusion. You can specify your FPGA Array, FlashROM, and Embedded Flash Memory by importing FDB, UFC and EFC files, respectively (as shown in the figure below). If you have imported a FlashROM and Embedded Flash Memory file you can click **Modify** to configure these feature before saving your PDB file.

Click Specify I/O States During Programming to set custom I/O states.

NOTE: You must import an FDB to populate Port Name and Macro Cell columns.

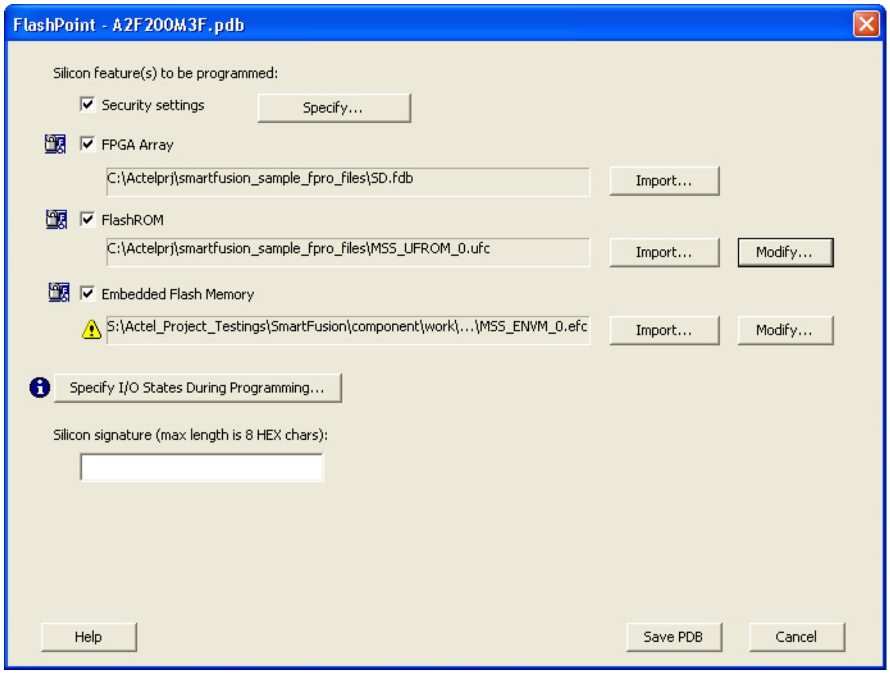


Figure 130 · FlashPoint Programming Settings for SmartFusion

Creating a Programming Database (PDB) File in Designer

The programming database (PDB) file supports IGLOO, ProASIC3, SmartFusion and Fusion devices only. This allows reconfiguration of the security settings, FlashROM, FPGA Array, and Embedded Flash Memory Blocks. You create the file in Designer using FlashPoint and you modify the file in FlashPro.

You must create programming files for SmartFusion in FlashPro; see the Generate a Programming File for SmartFusion topic for more information.

1. From the Designer main window, click the **Programming File** button. This brings up FlashPoint (see figure below).

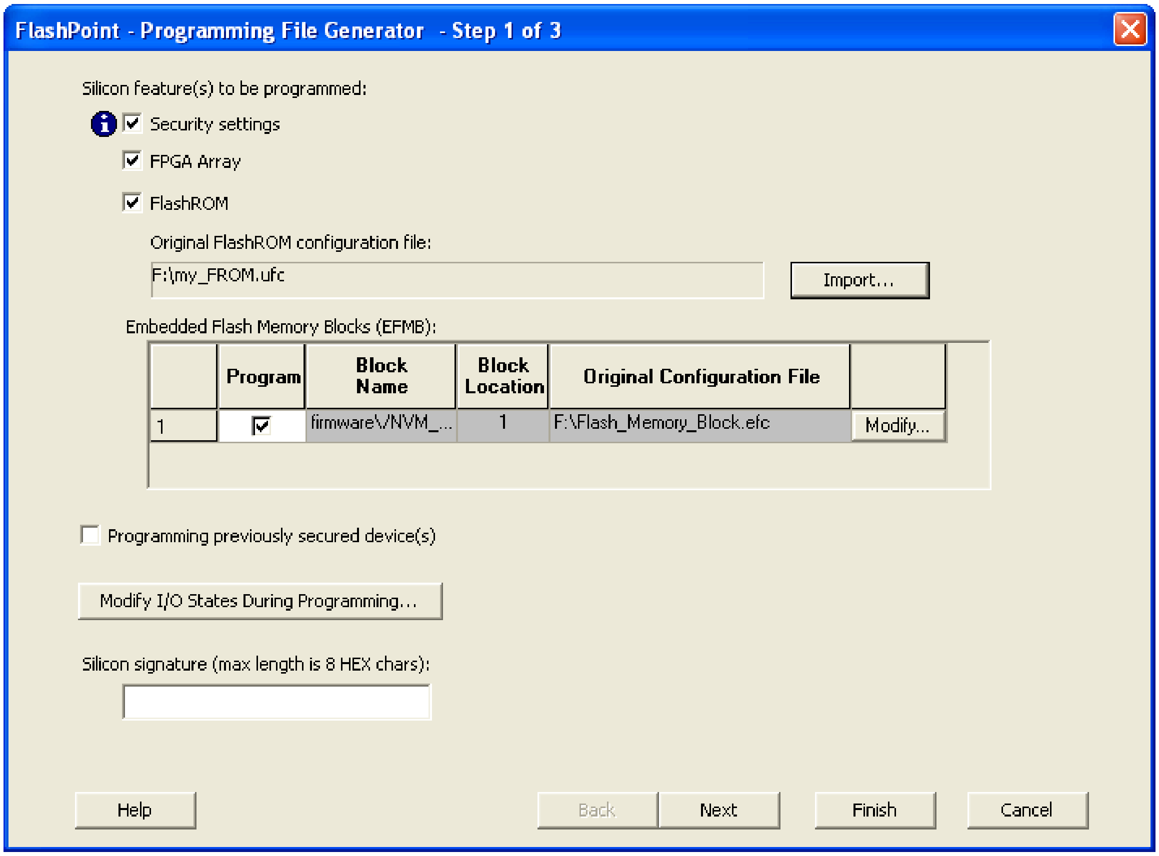


Figure 131 · FlashPoint Programming File Generator - PDB File

2. Select the silicon feature(s) to be programmed: Security Settings, FPGA array, FlashROM, and Embedded Flash Memory Block. If you are programming a previously secured device, check the Programming previously secured device(s) and enter the silicon signature.
3. Click **Finish** to create the PDB file.

See Also

Configuring security and FlashROM settings in FlashPro

Configuring security settings in FlashPro

Configuring FPGA array settings

Configuring FlashROM settings in FlashPro

Configuring Embedded Flash Memory Block settings in FlashPro

Programming Embedded Flash Memory Block

For more information about the Embedded Flash Memory Block, see the Flash Memory System Builder online help.

To program the Embedded Flash Memory Block:

1. Check the **Program** box to enable Embedded Flash Memory Block modification.
2. Click the **Modify** button to import Embedded Flash Memory Block configuration and memory content.

The **Modify Embedded Flash Memory Block** dialog box appears.

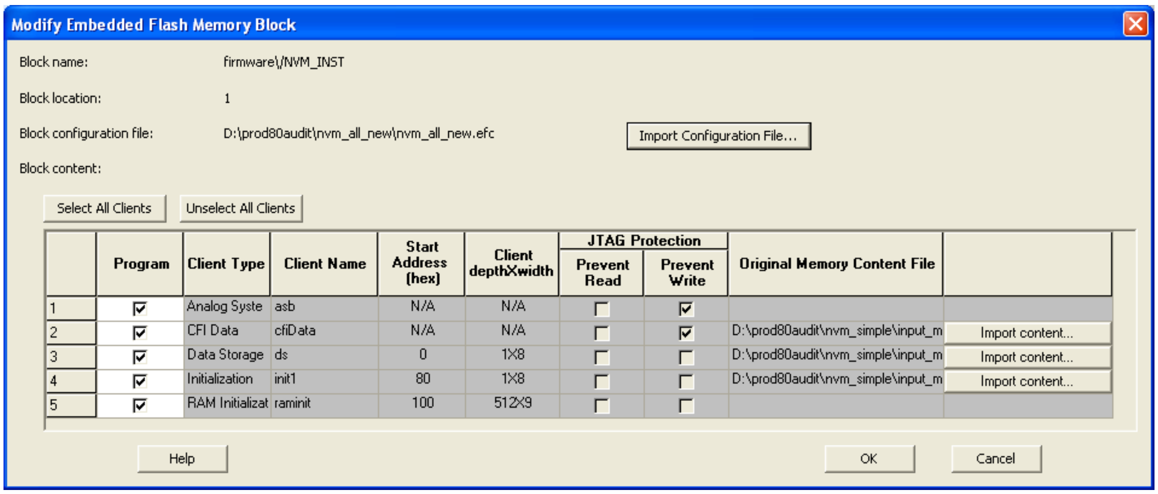


Figure 132 · Modify Embedded Flash Memory Block Content Dialog Box

3. Click the **Import Configuration File** button (if available) to import the Embedded Flash Memory Block configuration and memory content from the EFC file. This will populate the client table below. All clients that belong to this block will be selected by default.
4. Click the **Import content** button if you want to change the client memory content.
5. Click **OK**.

Note: Note: FlashPoint audits original configuration and memory content files and warns you if the files cannot be located or if they have been updated.

Programming the FPGA Array

The FPGA Array contains your design; in FlashPro for SmartFusion you must have an FDB file to program your FPGA Array.

You can program the FPGA Array by selecting the silicon feature **FPGA Array** in the Generate Programming File page and clicking **OK**.

In FlashPro, if you are using a PDB with an FPGA Array you cannot de-select it for programming unless you are using SmartFusion.

See Generate a programming file for more information.

Programming the FlashROM

You can program selected memory pages and specify the region values of the FlashROM.

To program FlashROM:

1. Select **FlashROM** from the **Generate Programming File** page.
2. Enter the location of the FlashROM configuration file. The **FlashROM Settings** page appears (see figure below).
3. Select the FlashROM memory page that you want to program.
4. Enter the data value for the configured regions.
5. If you selected the region with a **Read From File**, specify the file location.
6. If you selected the **Auto Increment** region, specify the **Start** and **Max** values.
7. Click **Finish**.
   FlashPoint generates your programming file.

Note: Note: You cannot change the FlashROM region configuration from FlashPoint. You can only change the configuration from the FlashROM core generator.

Silicon Signature

With Libero IDE tools, you can use the silicon signature to identify and track Actel designs and devices. When you generate a programming file, you can specify a unique silicon signature to program into the device. This signature is stored in the design database and in the programming file, and programmed into the device during programming.

The silicon signature is accessible through the USERCODE JTAG instruction.

Note: If you set the security level to high, medium, or custom, you must program the silicon signature along with the Security Setting. If you have already programmed the Security Setting into the target device, you cannot reprogram the silicon signature without reprogramming the Security Setting.

The previously programmed silicon signature will be erased if:

- You have already programmed the silicon signature and
- You are programming the security settings, but you do not have an entry in the silicon signature field

Programming Security Settings

FlashPoint allows you to set a security level of high, medium, or none (SmartFusion uses radio buttons and the option Clear Security instead of None).

To program Security Settings on the device:

1. If you choose to program Security Settings on the device from the **Generate Programming File** page, the wizard takes you to the **Security Settings** page.

Your Security Settings page depends on your family.

2. Set the security level for FPGA and FlashROM (see the table below for a description of the security levels).

Table 13 · FPGA and FlashROM Security Levels

Security Level	Security Option	Description
High	Protect with a 128-bit Advanced Encryption Standard (AES) key and a Pass Key	Access to the device is protected by an AES Key and the Pass Key. The Write and Verify operations of the FPGA Array use a 128-bit AES encrypted bitstream. From the JTAG interface, the Write and Verify operations of the FlashROM use a 128-bit AES encrypted bitstream. Read back of the FlashROM content via the JTAG interface is protected by the Pass Key. Read back of the FlashROM content is allowed from the FPGA Array.
Medium	Protect with Pass Key	The Write and Verify operations of the FPGA Array require a Pass Key. From the JTAG interface, the Read and Write operations on the FlashROM content require a Pass Key. You can Verify the FlashROM content via the JTAG interface without a Pass Key. Read back of the FlashROM content is allowed from the FPGA Array.
None	No security	The Write and Verify operations of the FPGA Array do not require keys. The Read, Write, and Verify operations of the FlashROM content also do not require keys. This option is available for SmartFusion; to choose it, de-select the Security Settings checkbox.

3. Enter the **Pass Key** and/ or the **AES Key** as appropriate. You can generate a random key by clicking the **Generate random key** button.

The **Pass Key** protects all the Security Settings for the FPGA Array and/or FlashROM.

The **AES Key** decrypts FPGA Array and/or FlashROM programming file content. Use the AES Key if you intend to program the device at an unsecured site or if you plan to update the design at a remote site in the future.

You can also customize the security levels by clicking the **Custom Level** button. For more information, see the Custom Security Levels section.

Custom Security Levels

For advanced use, you can customize your security levels.

To set custom security levels:

1. Click the **Custom Level** button in the **Security Settings** page. The **Custom Security Level** dialog box appears.
2. Select the **FPGA Array Security** and the **FlashROM Security** levels. ForSmartFusion and Fusion devices, you can also choose the Embedded Flash Memory Block level of security. The FPGA Array and the FlashROM can have different Security Settings. See the tables below for a description of the custom security option levels for FPGA Array and FlashROM.

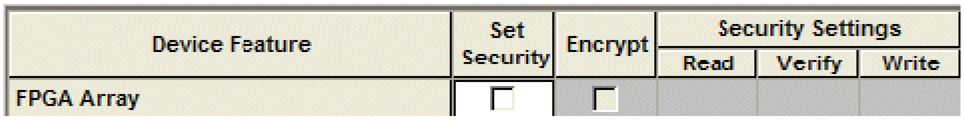
Table 14 · FPGA Array

<table>
<tr><th>Security Option</th><th>Description</th></tr>
<tr><td>Lock for both writing and verifying

</td><td>Allows writing/erasing and verification of the FPGA Array via the JTAG interface only with a valid Pass Key.</td></tr>
<tr><td>Lock for writing

</td><td>Allows the writing/erasing of the FPGA Array only with a valid Pass Key. Verification is allowed without a valid</td></tr>
</table>

Security Option	Description
	Pass Key.
Use the AES Key for both writing and verifying 	Allows the writing/erasing and verification of the FPGA Array only with a valid AES Key via the JTAG interface. This configures the device to accept an encrypted bitstream for reprogramming and verification of the FPGA Array. Use this option if you intend to complete final programming at an unsecured site or if you plan to update the design at a remote site in the future. Accessing the device security settings requires a valid Pass Key.
Allow write and verify	Allows

Security Option	Description
	writing/erasing and verification of the FPGA Array with plain text bitstream and without requiring a Pass Key or an AES Key. Use this option when you develop your product in-house.

Note: The ProASIC3 family FPGA Array is always read protected regardless of the Pass Key or the AES Key protection.

Table 15 · FlashROM

Security Option	Description
Lock for both reading and writing 	Allows the writing/erasing and reading of the FlashROM via the JTAG interface only with a valid Pass Key. Verification is allowed without a valid Pass Key.
Lock for writing	Allows the writing/erasing of the

Security Option	Description
	FlashROM via the JTAG interface only with a valid Pass Key. Reading and verification is allowed without a valid Pass Key.
Use the AES Key for both writing and verifying 	Allows the writing/erasing and verification of the FlashROM via the JTAG interface only with a valid AES Key. This configures the device to accept an encrypted bitstream for reprogramming and verification of the FlashROM. Use this option if you complete final programming at an unsecured site or if you plan to update the design at a remote site in

<table>
<tr><th>Security Option</th><th>Description</th></tr>
<tr><td></td><td>the future.
Note: The bitstream that is read back from the FlashROM is always unencrypted (plain text).</td></tr>
<tr><td>Allow reading, writing, and verifying
<table>
<tr><th rowspan="2">Device Feature</th><th rowspan="2">Set Security</th><th rowspan="2">Encrypt</th><th colspan="3">Security Settings</th></tr>
<tr><th>Read</th><th>Verify</th><th>Write</th></tr>
<tr><td>FlashROM</td><td>☐</td><td>☐</td><td></td><td></td><td></td></tr>
</table>
</td><td>Allows writing/erasing, reading and verification of the FlashROM content with a plain text bitstream and without requiring a valid Pass Key or an AES Key.</td></tr>
</table>

The FPGA Array can always read the FlashROM content regardless of these Security Settings.

Table 16 · Embedded Flash Memory Block

<table>
<tr><th>Security Option</th><th>Description</th></tr>
<tr><td>Lock for reading, verifying, and writing
<table>
<tr><th rowspan="2">Device Feature</th><th rowspan="2">Set Security</th><th rowspan="2">Encrypt</th><th colspan="3">Security Settings</th></tr>
<tr><th>Read</th><th>Verify</th><th>Write</th></tr>
<tr><td>firmware\NVM_INST (# 1)</td><td>☑</td><td>☐</td><td></td><td></td><td></td></tr>
</table>
</td><td>Allows the writing and reading of the Embedded Flash Memory Block via the JTAG interface only</td></tr>
</table>

<table>
<tr><th>Security Option</th><th>Description</th></tr>
<tr><td></td><td>with a valid Pass Key. Verification accomplished by reading back and compare.</td></tr>
<tr><td>Lock for writing
<table>
<tr><th rowspan="2">Device Feature</th><th rowspan="2">Set Security</th><th rowspan="2">Encrypt</th><th colspan="3">Security Settings</th></tr>
<tr><th>Read</th><th>Verify</th><th>Write</th></tr>
<tr><td>firmware\NVM_INST (# 1)</td><td>☑</td><td>☐</td><td></td><td></td><td></td></tr>
</table></td><td>Allows the writing of the Embedded Flash Memory Block via the JTAG interface only with a valid Pass Key. Reading and verification is allowed without a valid Pass Key.</td></tr>
<tr><td>Use AES Key for writing
<table>
<tr><th rowspan="2">Device Feature</th><th rowspan="2">Set Security</th><th rowspan="2">Encrypt</th><th colspan="3">Security Settings</th></tr>
<tr><th>Read</th><th>Verify</th><th>Write</th></tr>
<tr><td>firmware\NVM_INST (# 1)</td><td>☑</td><td>☑</td><td></td><td></td><td>AES</td></tr>
</table></td><td>Allows the writing of the Embedded Flash Memory Block via the JTAG interface only with a valid AES Key. This configures the device to accept an encrypted bitstream for</td></tr>
</table>

Security Option	Description
	reprogramming of the Embedded Flash Block. Use this option if you complete final programming at an unsecured site or if you plan to update the design at a remote site in the future. The bitstream that is read back from the Embedded Flash Memory Block is always unencrypted (plain text), when a valid pass key is provided.
Allow reading, writing, and verifying 	Allows writing, reading and verification of the Embedded Flash Memory Block content with a plain text bitstream and without requiring a valid Pass Key

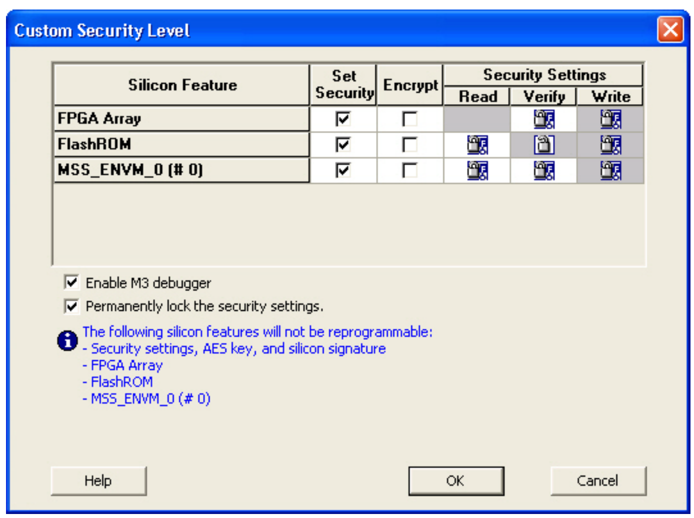
Security Option	Description
	or an AES Key.

3. To make the Security Settings permanent, select **Permanently lock the security settings** check box. This option prevents any future modifications of the Security Setting of the device. A Pass Key is not required if you use this option.

Note: Note: When you make the Security Settings permanent, you can never reprogram the Silicon Signature. If you Lock the write operation for the FPGA Array or the FlashROM, you can never reprogram the FPGA Array or the FlashROM, respectively. If you use an AES key, this key cannot be changed once you permanently lock the device.

4. (SmartFusion Only) Enable M3 Debugger option enables access to the M3 debugger even if security is enforced. Select the **Enable M3 debugger** checkbox if you want to access the M3 debugger after programming.

5. To use the Permanent FlashLock™ feature, select **Lock for both writing and verifying** for FPGA Array and **Lock for both reading and writing** for FlashROM and select the **Permanently lock the security settings** checkbox as shown in the figure below. This will make your device one-time-programmable.



Custom Security Level

6. Click the **OK** button. The **Security Settings** page appears with the **Custom security settings** information as shown in the figure below.

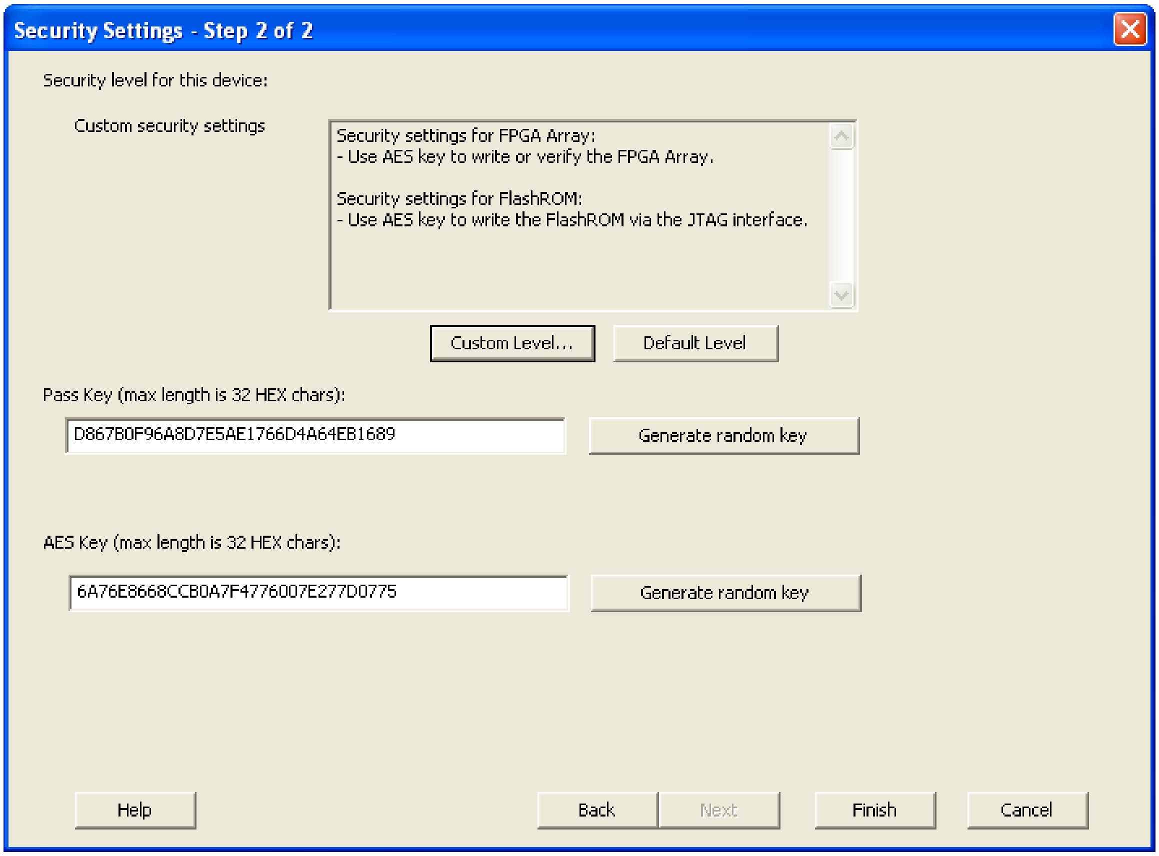


Figure 133 · Security Settings

Reprogramming a Secured Device

You must know the previous Security Settings of the device before you can reprogram a device with Security Settings.

Programming a Secured SmartFusion Device

After you create a PDB you may wish to export a programming file for a secured device. To do so:

1. Create a PDB file (as explained above) with security set to **High** or **Medium**. Save the PDB file.
2. From the **File** menu, choose **Export Single Programming File**. The Export Programming Files dialog box appears.
3. Click the **Export programming file(s) for currently secured device** checkbox. This exports programming files for devices that already have security settings programmed.
4. Choose your outputs and enter your output file **Name** and **Location**.
5. Click **Export** to create the file(s). Your updated secured programming files are in the directory you specified.

Custom Serialization Data for FlashROM Region

FlashPoint enables you to specify a custom serialization file as a source to provide content for programming into a Read from file FlashROM region. You can use this feature for serializing the target device with a custom serialization scheme.

To specify a FlashROM region:

1. From the Properties section in the FlashROM Settings page, select the file name of the custom serialization file (see figure below). For more information on custom serialization files, see Custom Serialization Data File Format.

Figure 134 · FlashROM Settings

Custom Serialization Data File Format

FlashPoint supports custom serialization data files that specify the data in binary, HEX, decimal, or ASCII text. The custom serialization data files may contain multiple data with the Line Feed (LF) character as the delimiter. You can create a file by entering serialization data into any type of text editor. Depending on the serialization data format (hex, ASCII, binary, decimal), input the serialization data according to the size of the region you specified in the FlashROM settings page.

Semantics

Each custom serialization file has only one type of data format (binary, decimal, Hex or ASCII text). For example, if a file contains two different data formats (i.e. binary and decimal) it is considered an invalid file.

The length of each data file must be shorter or equal to the selected region length. If the data is shorter then the selected region length, the most significant bits shall be padded with 0's. If the specified region length is longer then the selected region length, it is considered an invalid file.

The digit / character length is as follows:

```
-Binary digit: 1 bit
-Decimal digit: 4 bits
-Hex digit: 4 bits
-ASCII Character: 8 bits
```

Note the standard example below:

If you wanted to use, for example, device serialization for three devices with serialization data 123, 321, and 456, you would create file name from_read.txt. Each line in from_read.txt corresponds to the serialization data that will be programmed on each device. For example, the first line corresponds to the first device to be programmed, the second line corresponds to the second device to be programmed, and so on.

Hex serialization data file example

The following example is a Hex serialization data file for a 40-bit region. Enter the serialization data below into file created by any text editor:

```
123AEd210
AeB1
0001242E
```

Note: If you enter an invalid Hex digit such as 235SedF1, an error occurs. An error will also occur if you enter data that is out of range, i.e. 4300124EFE.

The following is an example of programming "AeB1" into Region_7_1 located on page 7, from Word 5 to Word 1 in the FlashROM settings page. See Custom serialization data for FlashROM region for more information.

	Table 15	...		Word 5	Word 4	Word 3	Word 2	Word 1	Word 0
Page 7	...	...	...	00	00	00	AE	B1	...

Binary serialization data file example

The following example is a binary serialization data file for a 16-bit region:

1100110011010001

```
100110011010011
11001100110101111 (This is an error: data out of range)
1001100110110111
1001100110110112 (This is an error: invalid binary digit)
```

Decimal serialization data file example

The following example is a decimal serialization data file for a 16-bit region:

```
65534
65535
65536 (This is an error: data out of range)
6553A (This is an error: invalid decimal digit)
```

Text serialization data file example

The following example is a text serialization data file for a 32-bit region:

```
AESB
A )e
ASE3 23 (This is an error: data out of range)
65A~
1234
AEbF
```

Syntax

Indentations in the syntax below indicate a wrapped line. If a line wraps and is not indented, then it should appear on one line; you may need to expand your help window to view the syntax correctly.

```
Custom serialization data file =
        <hex region data list> | <decimal region data list> |
        <binary region data list> | <ascii text data list>
Hex region data list = <hex data> <new line> { < hex data> <new line> }
Decimal region data list = <decimal data> <new line> {<decimal data><new line> }
Binary region data list = <binary data> <new line> { <binary data> <new line> }
ASCII text region data list = < ascii text data> <new line> { < ascii text data> <new
line> }
hex data = <hex digit> {<hex digit>}
decimal data = < decimal digit> {< decimal digit>}
binary data = < binary digit> {< binary digit>}
ASCII text data = <ascii character> {< ascii character >}
new line = LF
binary digit = ‘0’|‘1’
decimal digit = ‘0’|‘1’|‘2’|‘3’|‘4’|‘5’|‘6’|‘7’|‘8’| ‘9’
hex digit = ‘0’|‘1’|‘2’|‘3’|‘4’|‘5’|‘6’|‘7’|‘8’|‘9’|‘A’|‘B’|‘C’|‘D’ | ‘E’| ‘F’ |
        ‘a’| ‘b’ | ‘c’| ‘d’ | ‘e’| ‘f’
ascii character = characters from SP(0x20) to‘~’(0x7E).
```

Specifying I/O States During Programming

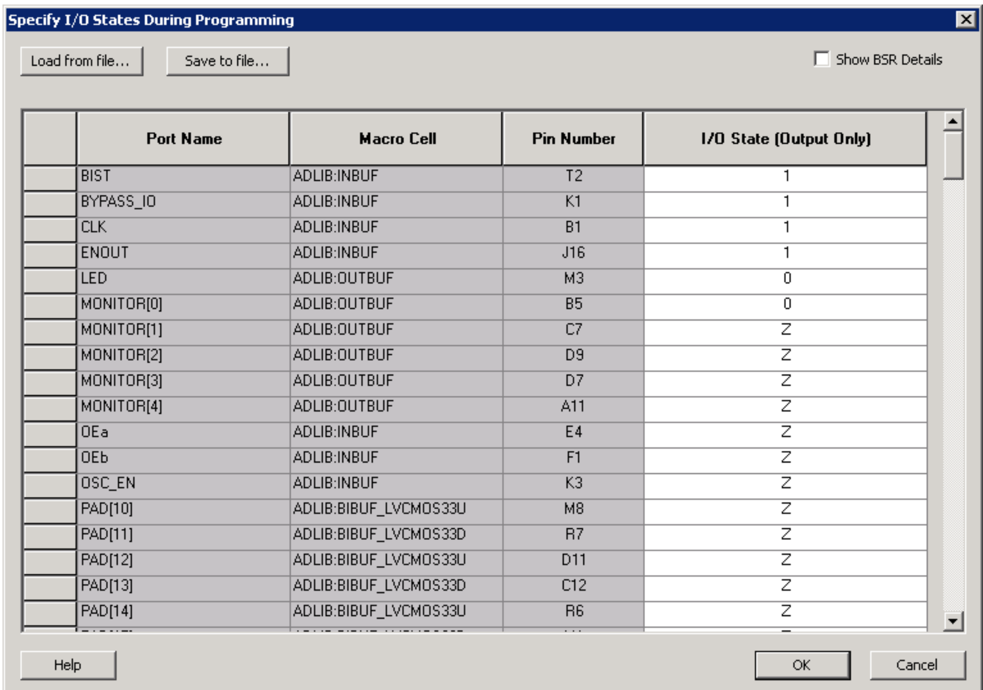
You can modify the I/O states during programming in FlashPro. In FlashPro, this feature is supported for PDB files generated from Designer v8.5 or greater.

Note: PDB files generated from Designer v8.1 to Designer v8.4 (including all service packs) have limited display of Pin Numbers only.

1. Load a PDB from the FlashPro GUI. You must have a PDB loaded to modify the I/O states during programming.
2. From the FlashPro GUI, click **PDB Configuration.** A FlashPoint – Programming File Generator window appears.
3. Click the **Specify I/O States During Programming** button to display the Specify I/O States During Programming dialog box.
4. Sort the pins as desired by clicking any of the column headers to sort the entries by that header. Select the I/Os you wish to modify (as shown in the figure below).
5. Set the I/O Output State. You can set Basic I/O settings if you want to use the default I/O settings for your pins, or use Custom I/O settings to customize the settings for each pin. See the Specifying I/O States During Programming - I/O States and BSR Details help topic for more information on setting your I/O state and the corresponding pin values. Basic I/O state settings are:

- 1 – I/O is set to drive out logic High

- 0 – I/O is set to drive out logic Low
- Last Known State: I/O is set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming
- Z - Tri-State: I/O is tristated



Port Name	Macro Cell	Pin Number	I/O State (Output Only)
BIST	ADLIB:INBUF	T2	1
BYPASS_IO	ADLIB:INBUF	K1	1
CLK	ADLIB:INBUF	B1	1
ENOUT	ADLIB:INBUF	J16	1
LED	ADLIB:OUTBUF	M3	0
MONITOR[0]	ADLIB:OUTBUF	B5	0
MONITOR[1]	ADLIB:OUTBUF	C7	Z
MONITOR[2]	ADLIB:OUTBUF	D9	Z
MONITOR[3]	ADLIB:OUTBUF	D7	Z
MONITOR[4]	ADLIB:OUTBUF	A11	Z
OEa	ADLIB:INBUF	E4	Z
OEb	ADLIB:INBUF	F1	Z
OSC_EN	ADLIB:INBUF	K3	Z
PAD[10]	ADLIB:BIBUF_LVCMOS33U	M8	Z
PAD[11]	ADLIB:BIBUF_LVCMOS33D	R7	Z
PAD[12]	ADLIB:BIBUF_LVCMOS33U	D11	Z
PAD[13]	ADLIB:BIBUF_LVCMOS33D	C12	Z
PAD[14]	ADLIB:BIBUF_LVCMOS33U	R6	Z

Figure 135 · I/O States During Programming Window

6. Click **OK** to return to the FlashPoint – Programming File Generator window.

Note: NOTE: I/O States During programming are saved to the ADB and resulting programming files after completing programming file generation.

Custom I/O Settings and Boundary Scan Registers

Each I/O in your device is comprised of an Input, Output and Output Enable Boundary Scan Register (BSR) cell..

The BSR cells enable you to define I/O states during programming and control the individual states for each Input, Output, and Output Enable register.

The Specify I/O States During Programming dialog box enables access to each of these BSR cells for control over the individual states. You can use the I/O State (Output Only) settings to set a specific output state and ignore the other values for the individual BSR elements, or you can click the Show BSR Details checkbox for control over the settings for each Input, Output Enable, and Output as you exit programming.

Specifying I/O States During Programming - I/O States and BSR Details

The I/O States During Programming dialog box enables you to set custom I/O states prior to programming.

I/O State (Output Only)

Sets your I/O states during programming to one of the values shown in the list below.

- 1 – I/Os are set to drive out logic High
- 0 – I/Os are set to drive out logic Low
- Last Known State: I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming
- Z - Tri-State: I/Os are tristated

When you set your I/O state, the Boundary Scan Register cells are set according to the table below. Use the Show BSR Details option to set custom states for each cell.

Table 17 · Default I/O Output Settings

Output State	Settings		
	Input	**Control (Output Enable)**	**Output**
Z (Tri-State)	1	0	0
0 (Low)	1	1	0
1 (High)	0	1	1
Last_Known_State	Last_Known_State	Last_Known_State	Last_Known_State

Table Key:

- 1 – High: I/Os are set to drive out logic High
- 0 – Low: I/Os are set to drive out logic Low
- Last_Known_State - I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming

Boundary Scan Registers - Enabled with Show BSR Details

Sets your I/O state to a specific output value during programming AND enables you to customize the values for the Boundary Scan Register (Input, Output Enable, and Output). You can change any Don't Care value in Boundary Scan Register States without changing the Output State of the pin (as shown in the table below).

For example, if you want to Tri-State a pin during programming, set Output Enable to 0; the Don't Care indicates that the other two values are immaterial.

If you want a pin to drive a logic High and have a logic 1 stored in the Input Boundary scan cell during programming, you may set all the values to 1.

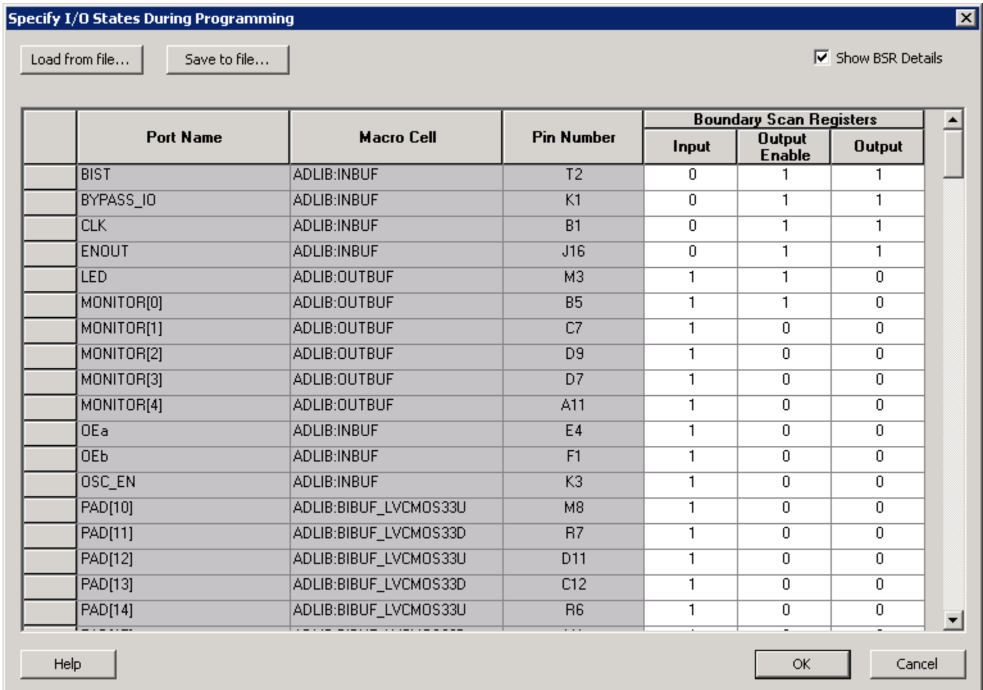
Table 18 · BSR Details I/O Output Settings

Output State	**Settings**		
	Input	**Output Enable**	**Output**
Z (Tri-State)	Don't Care	0	Don't Care
0 (Low)	Don't Care	1	0
1 (High)	Don't Care	1	1
Last Known State	Last State	Last State	Last State

Table Key:

- 1 – High: I/Os are set to drive out logic High
- 0 – Low: I/Os are set to drive out logic Low
- Don't Care – Don't Care values have no impact on the other settings.
- Last_Known_State – Sampled value: I/Os are set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming

The figure below shows an example of Boundary Scan Register settings.



	Port Name	Macro Cell	Pin Number	Boundary Scan Registers		
				Input	Output Enable	Output
	BIST	ADLIB:INBUF	T2	0	1	1
	BYPASS_IO	ADLIB:INBUF	K1	0	1	1
	CLK	ADLIB:INBUF	B1	0	1	1
	ENOUT	ADLIB:INBUF	J16	0	1	1
	LED	ADLIB:OUTBUF	M3	1	1	0
	MONITOR[0]	ADLIB:OUTBUF	B5	1	1	0
	MONITOR[1]	ADLIB:OUTBUF	C7	1	0	0
	MONITOR[2]	ADLIB:OUTBUF	D9	1	0	0
	MONITOR[3]	ADLIB:OUTBUF	D7	1	0	0
	MONITOR[4]	ADLIB:OUTBUF	A11	1	0	0
	OEa	ADLIB:INBUF	E4	1	0	0
	OEb	ADLIB:INBUF	F1	1	0	0
	OSC_EN	ADLIB:INBUF	K3	1	0	0
	PAD[10]	ADLIB:BIBUF_LVCMOS33U	M8	1	0	0
	PAD[11]	ADLIB:BIBUF_LVCMOS33D	R7	1	0	0
	PAD[12]	ADLIB:BIBUF_LVCMOS33U	D11	1	0	0
	PAD[13]	ADLIB:BIBUF_LVCMOS33D	C12	1	0	0
	PAD[14]	ADLIB:BIBUF_LVCMOS33U	R6	1	0	0

Figure 136 · Boundary Scan Registers

Specify I/O States During Programming Dialog Box

The I/O States During Programming dialog box enables you to specify custom settings for I/Os in your programming file. This is useful if you want to set an I/O to drive out specific logic, or if you want to use a custom I/O state to manage settings for each Input, Output Enable, and Output associated with an I/O.

Load from file

Load from file enables you to load an I/O Settings (*.ios) file. You can use the IOS file to import saved custom settings for all your I/Os. The exported IOS file have the following format:

- Used I/Os have an entry in the IOS file with the following format:

```
set_prog_io_state -portName {<design_port_name>} -input <value> -outputEnable <value> -output <value>
```

- Unused I/Os have an entry in the IOS file with the following format:

```
set_prog_io_state -pinNumber {<device_pinNumber>} -input <value> -outputEnable <value> -output <value>
```

Where <value> is:

- 1 – I/O is set to drive out logic High
- 0 – I/O is set to drive out logic Low

- Last_Known_State: I/O is set to the last value that was driven out prior to entering the programming mode, and then held at that value during programming
- Z - Tri-State: I/O is tristated

Save to file

Saves your I/O Settings File (*.ios) for future use. This is useful if you set custom states for your I/Os and want to use them again later in conjunction with a PDC file.

Port Name

Lists the names of all the ports in your design.

Macro Cell

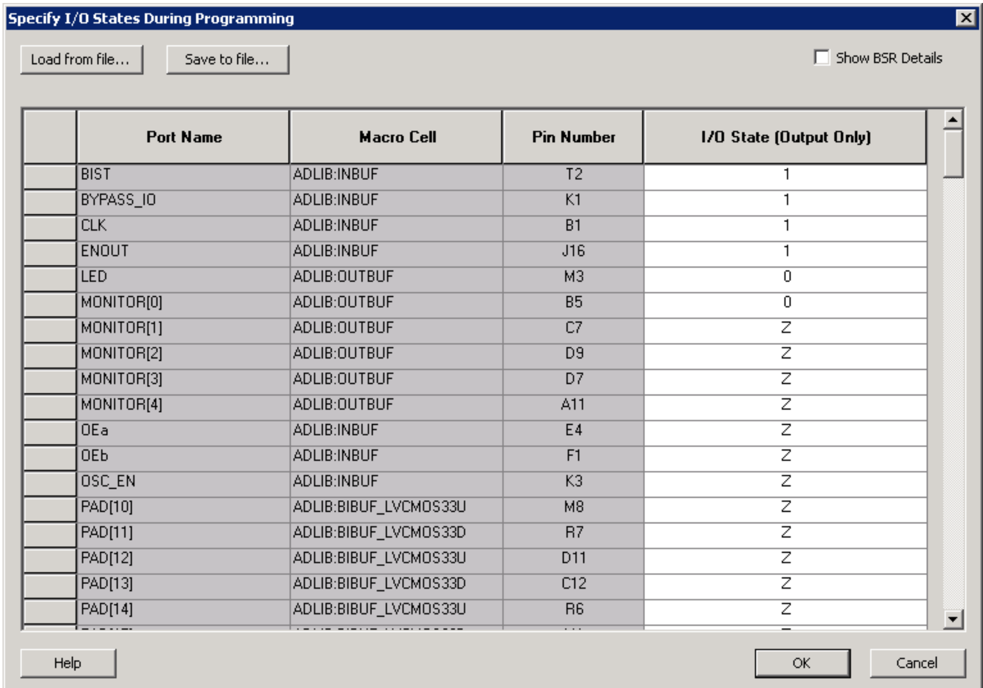
Lists the I/O type, such as INBUF, OUTBUF, PLLs, etc.

Pin Number

The package pin associate with the I/O.

I/O State (Output Only)

Your custom I/O State set during programming. This heading changes to Boundary Scan Register if you select the BSR Details checkbox; see the Specifying I/O States During Programming - I/O States and BSR Details help topic for more information on the BSR Details option.



Port Name	Macro Cell	Pin Number	I/O State (Output Only)
BIST	ADLIB:INBUF	T2	1
BYPASS_IO	ADLIB:INBUF	K1	1
CLK	ADLIB:INBUF	B1	1
ENOUT	ADLIB:INBUF	J16	1
LED	ADLIB:OUTBUF	M3	0
MONITOR[0]	ADLIB:OUTBUF	B5	0
MONITOR[1]	ADLIB:OUTBUF	C7	Z
MONITOR[2]	ADLIB:OUTBUF	D9	Z
MONITOR[3]	ADLIB:OUTBUF	D7	Z
MONITOR[4]	ADLIB:OUTBUF	A11	Z
OEa	ADLIB:INBUF	E4	Z
OEb	ADLIB:INBUF	F1	Z
OSC_EN	ADLIB:INBUF	K3	Z
PAD[10]	ADLIB:BIBUF_LVCMOS33U	M8	Z
PAD[11]	ADLIB:BIBUF_LVCMOS33D	R7	Z
PAD[12]	ADLIB:BIBUF_LVCMOS33U	D11	Z
PAD[13]	ADLIB:BIBUF_LVCMOS33D	C12	Z
PAD[14]	ADLIB:BIBUF_LVCMOS33U	R6	Z

Figure 137 · I/O States During Programming Dialog Box

Parallel Port Cable Information

The FlashPro software supports the generic Parallel Port Cable.

To connect to the Parallel Port Cable:

1. From the **Parallel Port Cable** text box, select the Parallel Port Buffer Cable (as shown in the figure below).
2. Select the parallel port that is connected to the cable from the **Parallel Port** text box.

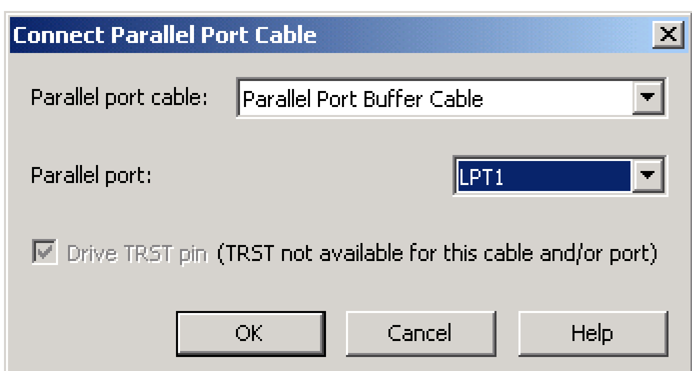


Figure 138 · Connect Parallel Port Cable

3. Click **OK.**

The **Para2Buff** programmer is added to the programmer list.

Importing and Exporting Files

Importing Configuration Files

To import a configuration file:

1. From the **File** menu, choose **Import Configuration File** . The **Import Configuration File** dialog box appears.
2. Navigate to your file and click **Open**.

Exporting Configuration Files

To export a configuration file:

1. From the **File** menu, choose **Export** and then choose **Export Configuration File**. The **Export Configuration File** dialog box appears.
2. Navigate to your file and click **Save**.

Export Programming Files (SmartFusion Only)

Export Programming Files enables you to export DirectC DAT, PDB and STP programming files. Exporting programming files is supported in both Chain and Single mode; to export programming files in Chain mode you must select one SmartFusion device in your chain.

To export a programming file:

1. From the FlashPro **File** menu choose **Export > Single Programming File**. The Export Programming Files dialog box appears.
2. Specify the **Output format**, **Name** and **Location** and click **Export** to create the files.

Export programming files for currently secured device enables you to generate PDB files for devices that have already been programmed with security settings. It generates encrypted data for encrypted features.

Target Programming Solution and STAPL file type options are available only if you have serialization.

Target Programming Solution

- Select Actel IHP (In House Programming) when generating STAPL or SVF files for Actel IHP.
- Select Silicon Sculptor II, BP Auto Programmer, or FlashPro4/3 when generating programming files for those programmers.
- Select Generic STAPL Player when generating STAPL files for generic STAPL players.

STAPL File Type Options

- **Single STAPL file for all devices**: Generates one programming file with all the generated increment values or with values in the custom serialization file.

- **One STAPL file per device**: Generates one programming file for each generated increment value or for each value in the custom serialization file.
- **Limit file size**: Some testers may have memory size restrictions for a single SVF file. The SVF limit file option enables you to limit the size of each SVF file by either file size or vectors.

The generated SVF files append an index to the file name indicating the sequence of files. The format is:

```
<SVF_filename>_XXXXX.svf
```

where XXXXX is the index of the SVF file. The first SVF file begins with <SVF_filename>_00000.svf and increments by 1 until file generation is complete.

Maximum file size: Max file size limit for the SVF file; use this option to limit your SVF file size based on number of kB.

Maximum number of vectors: Max vector limit for the SVF file; use this option to limit the size of your SVF based on number of vectors.

When serialization is available, choose the appropriate Target Programming Solution and STAPL File Type (if necessary) for your programming chain.

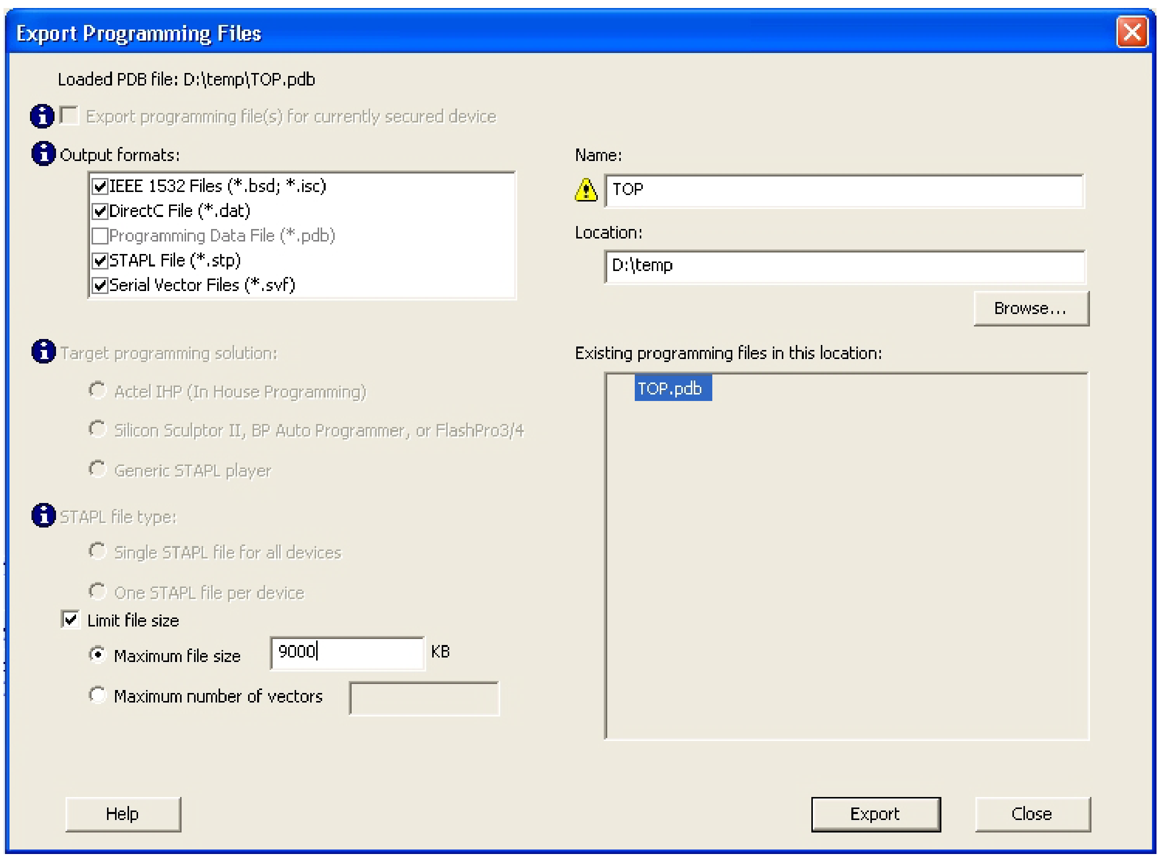


Figure 139 · Export Programming Files Dialog Box

Exporting a Chain STAPL File

To export a chain STAPL file:

1. From the **File** menu, select **Export** and then choose **Export Chain STAPL File**. The **Export Chain STAPL File** dialog box appears.
2. Name your file and click **Save**.

Note: Note: Chain STAPL file export is supported if all selected IGLOO, ProASIC3, SmartFusion and Fusion devices have STAPL or PDB files loaded.

Exporting a Chain SVF File

To export a chain SVF file:

1. From the **File** menu, choose **Export** and then choose **Export Chain SVF File**. The **Export Chain SVF File** dialog box appears.
2. Name your file and click **Save**.

Note: Note: Chain SVF file export is supported if all selected devices have STAPL or PDB files loaded.

Exporting Single Device STAPL Files

To export a single STAPL file in single mode:

This option is available only for single programming mode projects with a PDB file loaded (refer to Single STAPL file basic tutorial for more information).

1. From the **File** menu, choose **Export** > **Export Single Device STAPL File**. The **Export Single Device STAPL File** dialog box appears.
2. Name your file and click **Save**.

To export a single device STAPL file in chain mode:

This option is available only for chain programming mode projects with a PDB file loaded (refer to Chain programming tutorial for more information). Exporting a single device STAPL file is only supported for one device in the chain.

1. Select only one device from the chain, and from the **File** menu, select **Export** and then choose **Export Single Device STAPL File**. The **Export Single Device STAPL File** dialog box appears.
2. Name your file and click **Save**.

Or

1. Right-click a device in the Chain Configuration Window, and then choose **Export Single Device STAPL File**. The **Export Single Device STAPL File** dialog box appears.
2. Name your file and click **Save**.

Figure 140 ·

Exporting Single Device SVF Files

The following steps describe how to export SVF files.

To export single device SVF files in single mode:

This option is available only for single programming mode projects with a PDB file loaded (refer to Single STAPL file basic tutorial for more information).

1. From the **File** menu, select **Export** and then choose **Export Single Device SVF File**. The **Export Single Device SVF File** dialog box appears.
2. Name your file and click **Save**.

Note: Multiple SVF files will be generated from a single PDB. Each file corresponds to a PDB action, and will be saved in the *<SVF_filename>* folder as *<SVF_filename>_<action name>.SVF*.

To export single device SVF files in chain mode:

This option is available only for chain programming mode projects with a PDB file loaded (refer to Chain programming tutorial for more information).

1. Select only one device from the chain, and from the **File** menu, choose **Export** and then choose **Export Single Device SVF File**. The **Export Single Device SVF File** dialog box appears.
2. Name your file and click **Save**.

Or

1. Right-click a device in the Chain Configuration Window, and then choose **Export Single Device SVF File**. The **Export Single Device SVF File** dialog box appears.
2. Name your file and click **Save**.

Note: Multiple SVF files will be generated from a single PDB. Each file corresponds to a PDB action, and will be saved in the *<SVF_filename>* folder as *<SVF_filename>_<action name>.SVF*.

Exporting Single Device 1532 Files

IEEE 1532 programming files will only be exported in FlashPro for SmartFusion devices when an FDB has been properly imported.

To export single device 1532 files in single mode:

This option is available only for single programming mode projects with a PDB file loaded (refer to Single STAPL file basic tutorial for more information).

1. From the **File** menu, choose **Export Single Device 1532 File**. The **Export Single Device 1532 File** dialog box appears.
2. Name your file and click **Save**.

Note: Note: Two files will be generated from a single PDB and will be saved in the <1532_filename>_1532 folder as

Note: IEEE 1532 BSDL file - <1532_filename>.bsd

Note: IEEE 1532 Data file - <1532_filename>.isc

To export single device 1532 files in chain mode:

This option is available only for chain programming mode projects with a PDB file loaded (refer to Chain programming tutorial for more information). Exporting a single device STAPL file is only supported for one device in the chain.

1. Select only one device from the chain, and from the **File** menu, choose **Export** and then choose **Export Single Device 1532 File**. The **Export Single Device 1532 File** dialog box appears.
2. Name your file and click **Save**.

Or

1. Right-click a device in the Chain Configuration Window, and then choose **Export Single Device 1532 File**. The **Export Single Device 1532 File** dialog box appears.
2. Name your file and click **Save**.

Note: Note: Two files will be generated from a single PDB and will be saved in the <1532_filename>_1532 folder as

Note: IEEE 1532 BSDL file - <1532_filename>.bsd

Note: IEEE 1532 Data file - <1532_filename>.isc

Using Hot Keys

General Hot Keys

You can use hot keys for a lot of the features of the FlashPro software. See the table below for a list of general hot keys.

Table 19 · FlashPro Software General Hot Keys

Feature	Hot Key
New Project	Ctrl+N
Open Project	Ctrl+O
Save Project	Ctrl+S
Import Configuration File	Ctrl+I
Refresh Views	F5
Refresh/Rescan for Programmers	Ctrl+F5

See Also

[Single STAPL programming hot keys]

[Chain programming hot keys]

Single Device Programming Hot Keys

See the table below for the hot keys for single device programming.

Table 20 · Single Device Programming Hot Keys

Feature	Hot Key
Load a STAPL file	Ctrl + Shift + L
Select Action and Procedures	Ctrl + Shift + A
Enable Serialization	Ctrl + Shift + S
Select Serialization Data	Ctrl + Shift + R
View Serialization Status	Ctrl + Shift + U

Feature	Hot Key
View Chain Parameter (Pre/Post IR/DR)	Ctrl + Shift + H
Configure Target Device	Ctrl + Shift + D
Run	Ctrl + Return

Chain Programming Hot Keys

See the table below for the hot keys for chain programming.

Table 21 · Chain Programming Hot Keys

Feature	Hot Key
Add Actel Device	Ctrl + Shift + T
Add non Actel Device	Ctrl + Shift + N
Remove Device	Ctrl + R
Configure Device	Ctrl + F
Load STAPL File	Ctrl + Shift + L
Load BSDL File	Ctrl + Shift + B
Enable Device	Ctrl + E
Select Action and Procedures	Ctrl + Shift + A
Enable Serialization	Ctrl + Shift + S
Select Serialization Data	Ctrl + Shift + R
View Serialization Data	Ctrl + Shift + u
Copy Device	Ctrl + Shift + C
Cut Device	Ctrl + Shift + X
Paste Device	Ctrl + Shift + V

Feature	Hot Key
Move Device Down	Ctrl + D
Move Device Up	Ctrl + U
Run	Ctrl + Return

Batch Mode

Batch mode programming can be achieved by executing FlashPro TCL scripts from the command line.

The example below executes The FlashPro TCL script *batch.tcl* from the command line:

```
<location of Actel software>/bin/flashpro.exe script:batch.tcl
```

Batch.tcl contains the following script:

```
new_project -name {newproject} -location {./newproject} -mode {single}
 set_programming_file -file {./design.stp}
set_programming_action -action {PROGRAM}
run_selected_actions
close_project
```

About TCL Commands - FlashPro Tcl Command Reference

A Tcl (Tool Command Language) file contains scripts for simple or complex tasks. You can run scripts from the Windows command line or store and run a series of Tcl commands in a *.tcl batch file. The Tcl commands supported by FlashPro are listed in the table below.

Note: Tcl commands are case sensitive. However, their arguments are not.

Command	Action
add_actel_device	Adds an Actel device to the chain
add_non_actel_device	Adds a non-Actel device in the chain
check_flash_memory	Performs diagnostics of the page status and data information
close_project	Closes the FlashPro project
compare_analog_config	Compares the content of the analog block configurations in your design against the actual values in the device
compare_flashrom_client	Compares the content of the FlashROM configurations in your

Command	Action
	design against the actual values in the selected device
compare_memory_client	Compares the memory client in a specific device and block
configure_flashpro_prg	Changes FlashPro programmer settings
configure_flashpro3_prg	Changes FlashPro3 programmer settings
configure_flashproLite_prg	Changes FlashPro Lite programmer settings
connect_cable	Connects a parallel cable to a port
construct_chain_automatically	Automatically starts chain construction for the specified programmer
copy_device	Copies a device in the chain to the clipboard
cut_device	Removes one or more devices from the chain
dump_tcl_support	Unloads the list of supported FlashPro Tcl commands
enable_device	Enables or disables a device in the chain
enable_prg	Enables or disables one or more programmers
enable_prg_type	Enables or disables all programmers of a specified programmer type
enable_procedure	Enables/disables an optional procedure for an action
enable_serialization	Enables/disables serialization for a device
export_config	Exports a configuration file
export_script	Exports the history in a Tcl script
export_secured_pdb	Exports a single device secured PDB from the loaded PDB
export_single_1532	Exports a single device 1532 file
export_single_dat	Exports a single device DirectC data file

Command	Action
export_single_stapl	Exports a single device STAPL file
export_single_svf	Exports a single device SVF file
export_stapl	Exports the ChainBuilder STAPL file in chain programming mode
import_config	Imports a configuration file
new_project	Creates a new FlashPro project or convert an old ChainBuilder project into a new FlashPro project
open_project	Opens a FlashPro project
paste_device	Pastes the devices that are on the clipboard in the chain
ping_prg	Pings one or more programmers
read_analog_block_config	Reads analog block configuration information
read_device_status	Compares the memory client in a specific device and block
read_flash_memory	Reads information from the eNVM modules
read_flashrom	Reads the content of the FlashROM
read_id_code	Reads IDCode from the device without masking any IDCode fields
recover_flash_memory	Removes ECC2 errors due to memory corruption by reprogramming specified flash memory (NVM) pages
refresh_prg_list	Refreshes the programmer list
remove_device	Removes the device from the chain
remove_prg	Removes the programmer from the programmer list
run_selected_actions	Runs the selected action on the specified programmer and returns the exit code from the action

Command	Action
save_log	Saves the log file
save_project	Saves the FlashPro project
save_project_as	Saves the FlashPro project under a new project name
scan_chain_prg	Runs scan chain on a programmer
select_serial_range	Selects the serialization data
self_test_prg	Runs Self-Test on a programmer
set_bsdl_file	Sets a BSDL file to a non-Actel device in the chain
set_chain_param	Sets the chain parameters in single programming mode
set_debug_device	Identifies the device you intend to debug
set_debug_programmer	Identifies the programmer you want to use for debugging (if you have more than one)
set_device_ir	Sets the IR length of a non-Actel device in the chain
set_device_name	Changes the user name of a device in the chain
set_device_order	Sets the order of the devices in the chain to the order specified
set_device_tck	Sets the maximum TCK frequency of a non-Actel device in the chain
set_device_type	Changes the family of an Actel device in the chain
set_main_log_file	Sets the FlashPro log file
set_prg_name	Changes the user name of a programmer
set_programming_action	Selects the action for a device
set_programming_file	Sets the programming file for a device
set_programming_mode	Sets the programming mode

Command	Action
set_serialization_log_file	Sets the FlashPro log file to be used for serialization
set_serialization_mode	Sets the serialization mode
signature	Optional for AFM, DIO, and FUS file types
update_programming_file	Updates the programming file with the selected parameters

Running Tcl Scripts from within FlashPro

Instead of running scripts from the command line, you can use FlashPro's Run Script dialog box to run a script.

To execute a Tcl script file within FlashPro:

1. From the **File** menu, choose **Run Script** to display the **Execute Script** dialog box.

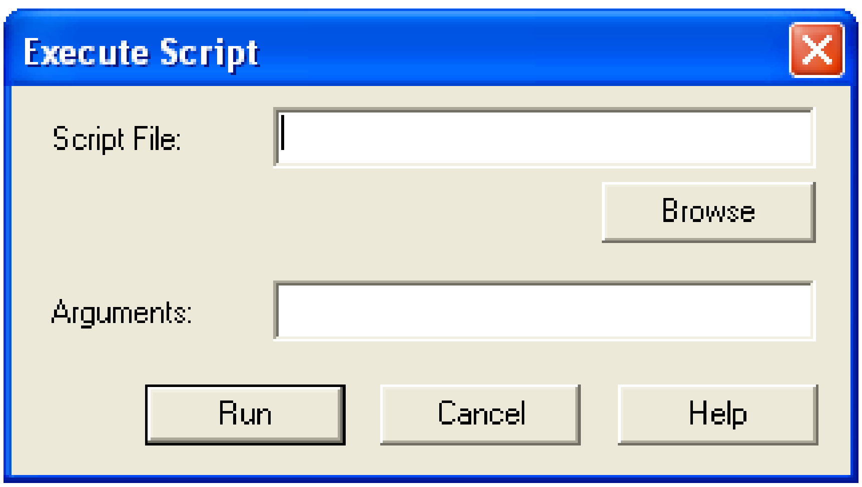


Figure 141 · Execute Script Dialog Box

2. Click **Browse** to display the **Open** dialog box, in which you can navigate to the folder containing the script file to open. When you click **Open**, FlashPro enters the full path and script filename into the Execute Script dialog box for you.
3. In the Arguments box, enter the arguments to pass to your Tcl script. Separate each argument by a space character. For information about accessing arguments passed to a Tcl script, see
4. Click **Run**.

Running Tcl Scripts from the Command Line

You can run Tcl scripts from your Windows command line.

To execute a Tcl script file in the FlashPro software from a shell command line:

1. At the prompt, type the path to the Actel software followed by the word "SCRIPT" and a colon, and then the name of the script file as follows:

```
<location of Actel software>/bin/flashpro.exe SCRIPT:<filename>
```

The example below executes in batch mode the script *foo.tcl*:

```
<location of Actel software>/bin/flashpro.exe script:foo.tcl
```

The example below executes in batch mode the script foo.tcl and exports the log in the file *foo.txt*:

```
<location of Actel software>/bin/flashpro.exe script:foo.tcl logfile:foo.txt
```

The example below executes in batch mode the script *foo.tcl*, creates a console where the log is displayed briefly, and exports the log in the file *foo.txt*:

```
<location of Actel software>/bin/flashpro.exe script:foo.tcl console_mode:brief
logfile:foo.txt
```

If you leave console_mode unspecified or set it to 'hide' FlashPro executes without a console window. If you want to leave the console window open you can run the script with the console_mode parameter set to 'show', as in the following example:

```
<location of Actel software>/bin/flashpro.exe script:foo.tcl console_mode:show
logfile:foo.txt
```

2. If you want to pass arguments to the Tcl script from the command line, then use the "SCRIPT_ARGS" variable as follows:

```
<location of Actel software>/bin/flashpro.exe SCRIPT:<filename> SCRIPT_ARGS:"param1
param2 param3"
```

Arguments passed to a Tcl script can be accessed through the Tcl variables *argc* and *argv*. The example below demonstrates how a Tcl script accesses these arguments:

```
puts "Script name: $argv0"
puts "Number of arguments: $argc"
set i 0
foreach arg $argv {
   puts "Arg $i : $arg"
   incr i
}
```

Note: Note: Script names can contain spaces if the script name is protected with double quotes:

```
flashpro.exe script:"flashpro tcl/foo 1.tcl"
```

Exporting Tcl Scripts from within FlashPro

To export a set of Tcl commands from the FlashPro history:

1. From the **File** menu, choose **Export** > **Export Script**.
2. Enter the filename and click **Save**. The Script Export Options dialog is appears (see image below).

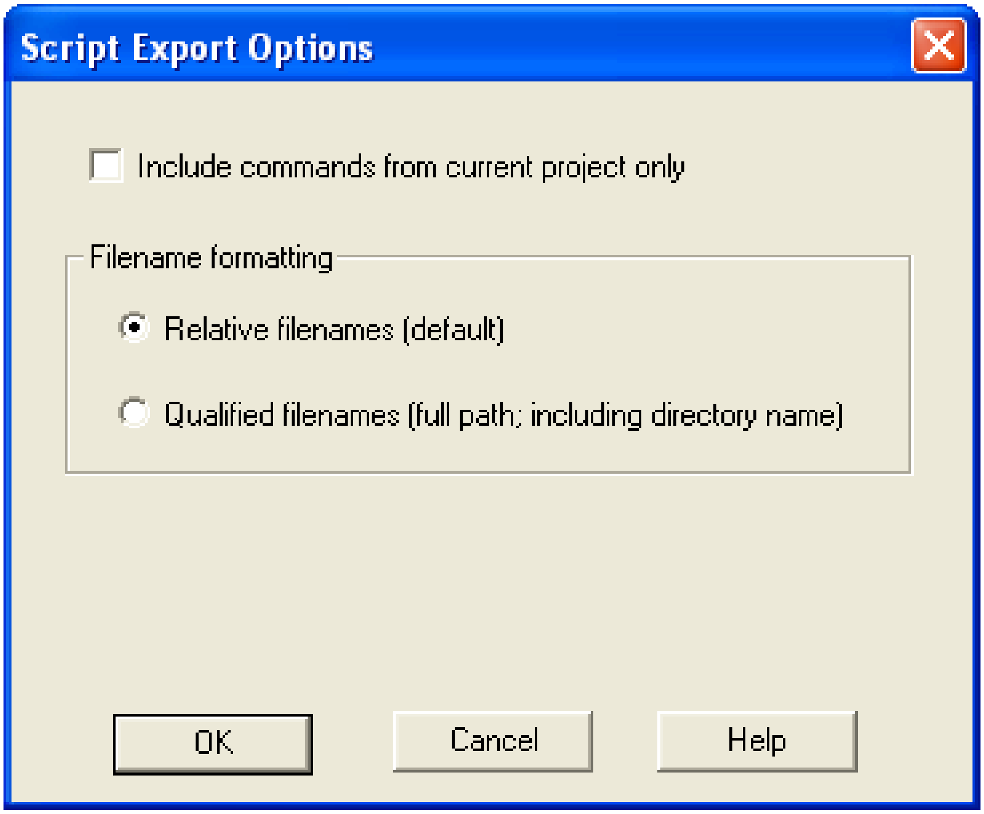


Figure 142 · Script Export Options Dialog Box

Check the **Include commands from current project only** to export commands of the current project only. You can specify the filename formatting by selecting **Relative filenames** (relative to the current directory) or **Qualified filenames**(absolute path, including the directory name).

4. Click **OK**.

add_actel_device

Adds an Actel device to the chain. Either the *file* or *device* parameter must be specified. Chain programming mode must have been set.

```
add_actel_device [-file {filename}] [-device {device}] -name {name} [-ukey {ukey_value}]
```

Arguments

Where:

`-file{filename}`

Specifies a programming filename.

`-device{device}`

Specifies the Actel device family(such as AFS600).

`-name{name}`

Specifies the device user name.

-ukey{*ukey_value*}

Optional (SmartFusion only) - Specifies the ukey value.

Supported Families

All

Exceptions

None

Example

```
add_actel_device –file {e:/design/stp/TOP.stp} –name {MyDevice1}
add_actel_device –device {A3P250} –name {MyDevice2}
```

add_non_actel_device

Adds a non-Actel device in the chain. Either the file, or (-tck And -ir) parameters must be specified. The Chain programming mode must have been set.

```
add_non_actel_device [-file {file}] [-ir {ir}] [-tck {tck}] [-name {name}]
```

Arguments

-file {*filename*}

Specifies a BSDL file.

-ir {*ir*}

Specifies the IR length.

-tck {*tck*}

Specifies the maximum TCK frequency (in MHz).

-name {*name*}

Specifies the device user name.

Supported Families

All

Exceptions

None

Examples

```
add_non_actel_device –file {e:/design/bsdl/DeviceX.bsdl } –name {MyDevice3}
add_non_actel_device –ir 8 – tck 5 –name {MyDevice4}
```

check_flash_memory

The command performs diagnostics of the page status and data information as follows:

- Page Status – includes ECC2 check of the page status information, write count
- Page Data - ECC2 check

```
check_flash_memory
[-name {device_name}]
[-block {integer_value}]
[-client {client_name}]
[-startpage {integer_value}]
[-endpage {integer_value}]
[-access {all | status | data}]
[-show {summary | pages}]
[-file {filename}]
```

At a minimum you must specify `-client <name>` OR

`-startpage <page_number> -endpage <page_number> -block <number>`

Arguments

`-name {device_name}`

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.

`-block {integer_value}`

(Optional argument; you must set -client or –startpage, –endpage and –block before use.) Specifies location of block for memory check.

`-client {client_name}`

Name of client for memory check.

`-startpage {integer_value}`

Startpage for page range; value must be an integer. You must specify a –endpage and –block along with this argument.

`-endpage {integer_value}`

Endpage for page range; value must be an integer. You must specify a –startpage and -block along with this argument.

`-access {all | status | data}`

(Optional argument; you must set -client or –startpage, –endpage and –block before use.) Specifies what NVM information to check: page status, data or both.

Value	Description
all	Shows the number of pages with corruption status, data corruption and out-of-range write count (default)
status	Shows the number of pages with corruption status and the number of pages with out-of-range write count
data	Shows only the number of pages with data corruption

-show {*summary | pages*}

(Optional argument; you must set -client or –startpage, –endpage and –block before use.) Specifies output level, as explained in the table below.

Value	Description
summary	Displays the summary for all checked pages (default)
pages	Displays the check results for each checked page

-file {*filename*}

(Optional argument; you must set -client or –startpage, –endpage and –block before use.) Name of output file for memory check.

Supported Families

SmartFusion, Fusion

Exceptions

None

Example

The following command checks the page status for block 0 from starpage 0 to endpage 2:

```
check_flash_memory -startpage 0 -endpage 2 -block 0
```

The following command checks the memory status for the client 'DS8bit' and saves it to the file 'checkFlashMemory.log':

```
check_flash_memory -client {DS8bit} -file {checkFlashMemory.log}
```

close_project

Closes the FlashPro project.

```
close_project
```

Arguments

None

Supported Families

All

Exceptions

None

Example

```
close_project
```

compare_analog_config

Compares the content of the analog block configurations in your design against the actual values in the device. In a typical IDE project, this directory is located at <project_root>/smartgen/<analog_block_core_name>.

```
compare_analog_config
[-name "device_name"] -mem_file_dir "mem_file_directory"
[-file "filename"]
```

Arguments

`-name {device_name}`

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.

`-mem_file_dir {mem_file_directory}`

Location of memory file.

`-file {filename}`

Output filename.

Supported Families

Fusion

Exceptions

None

Example

The following command reads the analog block configuration in the directory F:/tmp/Analog_Block and saves the data in the logfile compare_analogReport.log:

```
compare_analog_config -mem_file_dir {F:/tmp/Analog_Block} -file
{compare_analogReport.log}
```

The following command reads the analog block configuration information in the device 'AFS600' in the directory F:/tmp/Analog_Block and saves the data in the log file compare_analogReport.log:

```
compare_analog_config –name {AFS600} -mem_file_dir {F:/tmp/Analog_Block} -file
{compare_analogReport.log}
```

compare_flashrom_client

Compares the content of the FlashROM configurations in your design against the actual values in the selected device.

```
compare_flashrom_client [-name {device_name}] [-file "filename"]
```

Arguments

`-name {device_name}`

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.

`-file {filename}`

Identifies the name of the file you wish to compare to your device.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

None

Example

The following command saves the FlashROM data to the file 'FlashRomCompReport.log':

```
compare_flashrom_client -file {FlashRomCompReport.log}
```

The following command compares the data in the device 'A3P250' and saves the data in the logfile 'FlashRomCompReport.log':

```
compare_flashrom_client –name {A3P250} -file {FlashRomCompReport.log}
```

compare_memory_client

Compares the memory client in a specific device and block.

```
compare_memory_client [-name {device_name}] [-block integer_value] -client {client_name}
[-file {filename}]
```

Arguments

`-name { device_name}`

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.

`-block {integer_value}`

(Optional argument; you must set -client.) Specifies location of block for memory compare.

`-client {client_name}`

Name of client for memory compare.

`-file {filename}`

Optional file name.

Supported Families

SmartFusion and Fusion

Exceptions

None

Example

The following command compares the memory in the client 'DS32' on the device 'AFS600'.

```
compare_memory_client -client DS32 -name AFS600
```

The following command compares the data at block '0' to the client 'DS8bit':

```
compare_memory_client -block 0 -client {DS8bit}
```

The following command compares the memory in the device 'AFS600' at block '0' to the memory client 'DS8bit':

```
compare_memory_client –name {AFS600} -block 0 -client {DS8bit}
```

The following command compares the memory at block '1' to the memory client 'DS8bit' and saves the information in a log file to F:/tmp/NVMCompReport.log:

```
compare_memory_client -block 1 -client {DS8bit} -file {F:/tmp/NVMCompReport.log}
```

configure_flashpro_prg

Changes FlashPro programmer settings.

```
configure_flashpro_prg [-vpp {ON|OFF}] [-vpn {ON|OFF}] [-vddl {ON|OFF}] [-force_vddp
{ON|OFF}] [-vddp {2.5|3.3}] [-drive_trst {ON|OFF}] [-force_freq {ON|OFF}] [-freq {freq}]
```

Arguments

`-vpp {ON|OFF}`

Enables FlashPro programmer to drive VPP. Set to ON to drive VPP.

`-vpn {ON|OFF}`

Enables FlashPro programmer to drive VPN; set to ON to drive VPN.

`-vddl {ON|OFF}`

Enables FlashPro programmer to drive VDDL; set to ON to drive VDDL.

`-force_vddp {ON|OFF}`

Enables FlashPro programmer to drive VDDP; set to ON to drive VDDP.

`-vddp {2.5|3.3}`

Sets VDDP to 2.5 or 3.3 volts.

`-drive_trst {ON|OFF}`

Enables FlashPro programmer to drive TRST; set to ON to drive TRST.

`-force_freq {ON|OFF}`

Forces the FlashPro software to use the TCK frequency specified by the software rather than the TCK frequency specified in the programmer file.

`-freq {freq}`

Specifies the TCK frequency in MHz.

Supported Families

ProASIC^{PLUS}, ProASIC

Exceptions

None

Example

The following example enables the FlashPro programmer to drive the VPP, VPN, VDDL, VDDP, sets the drive voltage to 3.3v, disables the driver for TRST, and does not force the programmer to use the TCK frequency specified in the software.

```
configure_flashpro_prg –vpp {ON} –vpn {ON} –vddl {ON} –force_vddp {ON} –vddp {3.3} –
drive_trst {OFF} –force_freq {OFF}
```

configure_flashpro3_prg

Changes Actel FlashPro3 programmer settings.

```
configure_flashpro3_prg [-vpump {ON|OFF}] [-force_freq {ON|OFF}] [-freq {freq}]
```

Arguments

`-vpump {ON|OFF}`

Enables FlashPro programmer to drive VPUMP. Set to ON to drive VPUMP.

`-force_freq {ON|OFF}`

Forces the FlashPro software to use the TCK frequency specified by the software rather than the TCK frequency specified in the programmer file.

`-freq {freq}`

Specifies the TCK frequency in MHZ.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

None

Example

The following example sets the VPUMP option to OFF and uses the TCK frequency specified in the programmer file (force_freq is set to OFF).

```
configure_flashpro3_prg –vpump {OFF} –force_freq {OFF}
```

configure_flashproLite_prg

Changes Actel FlashPro Lite programmer settings.

```
configure_flashproLite_prg [-vpp {ON|OFF}] [-vpn {ON|OFF}] [-drive_trst {ON|OFF}] [-
force_freq {ON|OFF}] [-freq {freq}]
```

Arguments

`-vpp {ON|OFF}`

Enables FlashPro programmer to drive VPP. Set to ON to drive VPP.

`-vpn {ON|OFF}`

Enables FlashPro programmer to drive VPN; set to ON to drive VPN.

`-drive_trst {ON|OFF}`

Enables FlashPro programmer to drive TRST; set to ON to drive TRST.

`-force_freq {ON|OFF}`

Forces the FlashPro software to use the TCK frequency specified by the software rather than the TCK frequency specified in the programmer file.

`-freq {freq}`

Specifies the TCK frequency in MHz.

Supported Families

ProASICPLUS

Exceptions

None

Example

The following example sets the programmer to drive the VPP, drive VPN, drive the TRST and uses the frequency set by the programmer file (sets force_freq to OFF):

```
configure_flashprolite_prg –vpp {ON} –vpn {ON} –drive_trst {ON} –force_freq {OFF}
```

connect_cable

Connects a parallel cable to a port.

```
connect_cable -cable_name {cable_name} -port_name {port_name} [-drive_trst {ON|OFF}]
```

Arguments

`-cable_name {cable_name}`

Identifies the name of the parallel port cable you wish to connect.

`-port_name {port_name}`

Specifies the parallel port where the parallel programmer is connected.

`-drive_trst {ON|OFF}`

Enables the parallel port cable to drive TRST.

Supported Families

All

Exceptions

None

Example

The following example connects the cable named Parallel_Port_Buffer_Cable to the port LPT1 and enables drive TRST:

```
connect_cable -cable_name {Parallel_Port_Buffer_Cable} -port_name {Lpt1} -drive_trst
{ON}
```

construct_chain_automatically

Automatically starts chain construction for the specified programmer.

```
construct_chain_automatically[(-name {name})+]
```

Arguments

-name {*name*}

Specifies the programmer(s) name(s).

Supported Families

All

Exceptions

N/A

Example

Example for one programmer:

```
construct_chain_automatically -name {21428}
```

Example for two programmers:

```
construct_chain_automatically -name {21428} –name {00579}
```

copy_device

Copies a device in the chain to the clipboard. Chain programming mode must be set. See the paste_device command for more information.

```
copy_device (-name {name})*
```

Arguments

-name {name}

Specifies the device name. Repeat this argument to copy multiple devices.

Supported Families

All

Exceptions

None

Example

The example copies the device 'mydevice1' to the same location with a new name 'mydevice2'.

```
copy_device -name {MyDevice1} -name {MyDevice2}
```

cut_device

Removes one or more devices from the chain. It places the removed device in the clipboard. Chain programming mode must be set to use this command. See the `paste_device` command for more information.

```
cut_device (-name {name})*
```

Arguments

`-name {name}`
Specifies the device name. You can repeat this argument for multiple devices.

Supported Families

All

Exceptions

None

Example

The following example removes the devices 'mydevice1' and 'mydevice2' from the chain.

```
cut_device -name {MyDevice1} -name {MyDevice2}
```

dump_tcl_support

Unloads the list of supported FlashPro Tcl commands.

```
dump_tcl_support -file {file}
```

Arguments

`-file {file}`

Supported Families

All

Exceptions

None

Example

The following example dumps your Tcl commands into the file 'tcldump.tcl'

```
dump_tcl_support -file {tcldump.tcl}
```

enable_device

Enables or disables a device in the chain (if the device is disabled, it is bypassed). Chain programming mode must be set. The device must be an Actel device.

```
enable_device -name {name} -enable {TRUE|FALSE}
```

Arguments

`-name {name}`

Specifies your device name

`-enable {TRUE|FALSE}`

Specifies whether the device is to be enabled or disabled. If you specify multiple devices, this argument applies to all specified devices. (TRUE = enable. FALSE = disable)

Supported Families

All

Exceptions

None

Example

The following example disables the device 'mydevice1' in the chain.

```
enable_device –name {MyDevice1} –enable {FALSE}
```

enable_prg

Enables or disables one or more programmers.

```
enable_prg (-name {name})* -enable {TRUE|FALSE}
```

Arguments

`-name {name}*`

Specifies the programmer name. You can repeat this argument for multiple programmers.

`-enable {TRUE|FALSE}`

Specifies whether the programmer is to be enabled or disabled. If you specify multiple programmers, this argument applies to all of them (TRUE = enable. FALSE = disable).

Supported Families

All

Exceptions

None

Example

The following example enables the programmers 'FP300085' and 'FP300086'.

```
enable_prg -name {FP300085} -name {FP300086} -enable {TRUE}
```

enable_prg_type

Enables or disables all programmers of a specified programmer type.

```
enable_prg_type -prg_type {prg_type} -enable { TRUE | FALSE }
```

Arguments

`-prgType { FP | FPLite | FP3 | PP }`

Specifies the programmer type to be enabled/disabled (FP–FlashPro type programmers, FPLite–FlashPro Lite type programers, FP3–FlashPro3 type programmers, PP–Parallel port cable type programmers).

`-enable {TRUE|FALSE}`

Specifies whether the programmers are to be enabled or disabled (TRUE–enable, FALSE–disable).

Supported Families

All

Exceptions

None

Example

The following example enables the FlashPro3 programmer.

```
enable_prog_type -prg_type{FP3} -enable{TRUE}
```

enable_procedure

To enable/disable an optional procedure of an action. The device name parameter must be specified only in chain programming mode. A programming file must have been loaded.

```
enable_procedure [-name {name}] -action {action} -procedure {procedure} -enable
{TRUE|FALSE}
```

Arguments

```
-name {name}
-action {action}
-procedure {procedure}
-enable {TRUE|FALSE}
```

Supported Families

All

Exceptions

None

Example

In single programming mode:

```
enable_procedure –action {PROGRAM} –procedure {DO_ERASE} –enable {TRUE}
```

In chain programming mode:

```
enable_serialization –name {MyDevice2} –action {PROGRAM} –procedure {DO_ERASE} –enable
{FALSE}
```

enable_serialization

Enables/disables serialization for a device. The device name parameter must be specified only in chain programming mode. A programming file allowing serialization must be loaded.

```
enable_serialization [-name {name}] -enable {TRUE|FALSE}
```

Arguments

`-name {name}`

Specifies device name

`-enable {TRUE|FALSE}`

Enables (TRUE) or disables (FALSE) serialization.

Supported Families

All

Exceptions

None

Example

Enabling serialization in single programming mode:

```
enable_serialization -enable {TRUE}
```

Disabling serialization on the device 'mydevice2' in chain programming mode:

```
enable_serialization -name {MyDevice2} -enable {FALSE}
```

export_chain_stapl

Exports the ChainBuilder STAPL file in chain programming mode.

```
export_chain_stapl -file {file}
```

Arguments

`-file {file}`

Specifies the file to be exported.

Supported Families

All

Exceptions

None

Example

The following example exports the STAPL file 'tcl_testing_chain.stp':

```
-export_chain_stapl -file {./tcl_testing_chain.stp}
```

export_config

Exports a configuration file.

```
export_config -file {file}
```

Arguments

`-file {file}`

Specifies the file to export.

Supported Families

All

Exceptions

None

Example

The following example exports the configuration file 'myconfig1'

```
export_config -file {myconfig1}
```

export_secured_pdb

Exports a single device secured PDB from the loaded PDB.

```
export_secured_pdb -file {file} [-name {name}]
```

Arguments

`-file {file}`

Specifies the file to export.

`-name {name}`

Specifies the name of the device in chain mode to export a single device currently secured PDB.

Supported Families

SmartFusion, Fusion

Exceptions

'-secured' is only supported for SmartFusion devices.

Example

In single mode, the following command exports the secured PDB 'my_design.pdb':

```
export_secured_pdb -file {D:/TOP/my_design.pdb}
```

In chain mode, the following example exports the secured PDB 'my_design.pdb' from the A2F200M3F device in the chain:

```
export_secured_pdb -name {A2F200M3F} -file {./my_design.pdb}
```

export_script

Exports the history in a Tcl script.

```
export_script -file {file} -relative_path {TRUE|FALSE}
```

Arguments

`-file {file}`

Specifies the file to export.

`-relative_path {TRUE|FALSE}`

Specifies whether the file path must be exported as a relative path or an absolute path.

Supported Families

All

Exceptions

None

Example

The following example exports your Tcl history to the file 'history.tcl' with absolute pathnames.

```
export_script -file {./history.tcl} -relative_path {FALSE}
```

export_single_1532

Exports a single device IEEE 1532 file.

```
export_single_1532 -file {file} [-name {name}] [-pdb {pdb_file}] [-secured]
```

Arguments

`-file {file}`

Specifies the file to export.

`-name {name}`

Specifies the name of the device in chain mode to export single device IEEE 1532 programming file.

`-pdb {pdb_file}`

Specifies the PDB to use for exporting a IEEE 1532 programming file. By default, the loaded PDB is used for exporting a IEEE 1532 programming file.

`-secured`

Exports a IEEE 1532 programming file for a secured device.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

'-secured' is only supported for SmartFusion devices.

Example

Single Mode, exports the secured IEEE 1532 files 'my_design.isc' and 'my_design.bsd' into folder 'D:/TOP/my_design_1532' using the PDB 'my_design.pdb':

```
export_single_1532 -file {D:/TOP/my_design_1532) -pdb {D:/TOP/my_design.pdb} -secured
```

Chain Mode example exports secured IEEE 1532 files 'my_design.isc' and 'my_design.bsd' into folder 'D:/TOP/my_design_1532' from a device in the chain named 'A2F200M3F' using the PDB 'my_design.pdb':

```
export_single_1532 -name {A2F200M3F} -file {./my_design_1532} -pdb {./my_design.pdb} -
secured
```

export_single_dat

Exports a single device DirectC data file.

```
export_single_dat -file {file} [-name {name}] [-pdb {pdb_file}] [-secured]
```

Arguments

-file {*file*}

Specifies the name of the file you are exporting.

-name {*name*}

Specifies the name of the device in chain mode to export a single device DirectC data file.

-pdb {*pdb_file*}

Specifies the PDB to use for exporting a DirectC data file. By default, the loaded PDB will be used for exporting a DirectC data file.

-secured

Use this argument to export a secured DirectC data file.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

'-secured' is only supported for SmartFusion devices.

Example

Single Mode, exports a secured DirectC DAT file 'my_design.dat' using the PDB file 'my_design.pdb':

```
export_single_dat -file {D:/TOP/my_design.dat} -pdb {D:/TOP/my_design.pdb} -secured
```

Chain Mode, exports a secured DirectC DAT file 'my_design.dat' from a device in the chain named 'A2F200M3F', using the PDB 'my_design.pdb':

```
export single_dat -name {A2F200M3F} -file {./my_design.dat} -pdb {./my_design.pdb} -
secured
```

export_single_stapl

Exports a single device STAPL file.

```
export_single_stapl -file {file} [–name {name}] [-pdb {pdb_file}] [-secured]
```

Arguments

-file {*file*}

Specifies the file to export.

-name {*name*}

Specifies the name of the device in chain mode to export a single device STAPL file.

`-pdb {pdb_file}`

Specifies the PDB to use for exporting a STAPL file. By default, the loaded PDB is used for exporting a STAPL file.

`-secured`

Exports a secured STAPL file.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

None

Example

Single Mode example exports the secured file 'my_design.stp' using the PDB 'my_design.pdb':

```
export_single_stapl -file {D:/TOP/my_design.stp} -pdb {D:/TOP/my_design.pdb} -secured
```

Chain Mode example exports secured STAPL file 'my_design.stp' from device 'A2F200M3F' using the PDB 'my_design.pdb':

```
export single_stapl -name {A2F200M3F} -file {./my_design.stp} -pdb {./my_design.pdb} -
secured
```

export_single_svf

Exports a single device SVF programming file.

```
export_single_svf -file {file} [–name {name}] [-pdb {pdb_file}] [-secured]
```

Arguments

`-file {file}`

Specifies the file to export.

`-name {name}`

Specifies the name of the device in chain mode to export a single device SVF programming file.

`-pdb {pdb_file}`

Specifies the PDB to use for exporting a SVF programming file. By default, the loaded PDB is used for exporting a SVF programming file.

`-secured`

Exports a SVF programming file for a secured device.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

'-secured' is only supported for SmartFusion devices.

Example

Single Mode, exports the secured SVF files for each programming ACTION with the following format 'my_design_ACTION.svf' into folder 'D:/TOP/my_design_SVF' using the PDB 'my_design.pdb':

```
export_single_SVF -file {D:/TOP/my_design_SVF) -pdb {D:/TOP/my_design.pdb} -secured
```

Chain Mode example exports secured SVF files for each programming ACTION with the following format 'my_design_ACTION.svf' into folder 'D:/TOP/my_design_SVF' from a device in the chain named 'A2F200M3F' using the PDB 'my_design.pdb':

```
export_single_SVF -name {A2F200M3F} -file {./my_design_SVF} -pdb {./my_design.pdb} -
secured
```

import_config

Imports a configuration file.

```
import_config -file {file}
```

Arguments

`-file {file}`

Specifies the file to import.

Supported Families

All

Exceptions

None

Example

The following example imports the configuration file 'my_config1.ufc':

```
import_config -file {my_config1.ufc'}
```

new_project

Creates a new FlashPro project or convert an old ChainBuilder project into a new FlashPro project (the mode parameter must be 'chain' in this case).

```
new_project -name {name} -location {location} -mode {single|chain} [-convert_chb
{convert_chb}]
```

Arguments

`-name {name}`

Specifies the project name.

`-location {location}`

Specifies the project location.

`-mode {single|chain}`

Specifies programming mode; either single or chain.

`-convert_chb {convert_chb}`

An optional argument that specifies the ChainBuilder project to be converted.

Supported Families

All

Exceptions

None

Example

Create a new FlashPro single device project named 'FPPrj1' in a directory with the name 'FPProject1':

```
new_project -name {FPPrj1} -location {./FPProject1} -mode {single}
```

Create a new FlashPro project named 'FPPrjChb' in the directory 'ChbProject1'; converts the ChainBuilder project 'prj1.chb' project to FlashPro.

```
new_project -name {FPPrjChb} -location {./ChbProject1} -mode {chain} -convert_chb
{./chb_prj/prj1.chb}
```

open_project

Opens a FlashPro project.

```
open_project -project {project}
```

Arguments

`-project {project}`

Specifies the location and name of the project you wish to open.

Supported Families

All

Exceptions

None

Example

Opens the 'FPPrj1.pro' project from the FPProject1 directory

```
open_project -project {./FPProject1/FPPrj1.pro}
```

paste_device

Pastes the devices that are on the clipboard in the chain, immediately above the *position_name* device, if this parameter is specified. Otherwise it places the devices at the end of the chain. The chain programming mode must be enabled.

```
paste_device [-position_name {position_name}]
```

Arguments

`-position_name {position_name}`

Optional argument that specifies the name of a device in the chain.

Supported Families

All

Exceptions

None

Examples

The following example pastes the devices on the clipboard immediately above the device 'mydevice3' in the chain.

```
paste_device -position_name {MyDevice3}
```

ping_prg

Pings one or more programmers.

```
ping_prg (-name {name})*
```

Arguments

`-name {name}`
Specifies the programmer to be pinged. Repeat this argument for multiple programmers.

Supported Families

All

Exceptions

None

Example

The following example pings the programmers 'FP300085' and 'FP30086'.

```
ping_prg -name {FP300085} -name {FP300086}
```

read_analog_block_config

Reads each channel configuration on your analog system, enabling you to identify if/how each channel is configured.

```
read_analog_block_config [-name {device_name}] [-file {filename}]
```

Arguments

`-name {device_name}`
Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.
`-file {filename}`
(Optional) Identifies the name of the file to which read results will be saved.

Supported Families

Fusion

Exceptions

None

Example

The following command reads the analog block configuration information in the device 'AFS600':

```
read_analog_block_config -name {AFS600}
```

read_device_status

Displays the Device Information report; the Device Information report is a complete summary of your device state, analog block test values, user information, factory serial number and security information..

```
read_device_status [-name {device_name}] [-file {filename}]
```

Arguments

`-name device_name`

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.

`-file {filename}`

(Optional) Identifies the name of the file to which read results will be saved.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

None

Example

The following reads device info from the 'AFS600' device.

```
read_device_status -name AFS600
```

read_flash_memory

The command reads information from the NVM modules. There are two types of information that can be read:

- Page Status – includes ECC2 status, write count, access protection
- Page Data

```
read_flash_memory
[-name {device_name}]
[-block {integer_value}]
[-client {client_name}]
[-startpage {integer_value}]
[-endpage {integer_value}]
[-access {all | status | data}]
[-file {filename}]
```

At a minimum you must specify `-client` <*name*> OR

`-startpage <page_number> -endpage <page_number> -block <number>`

Arguments

`-name {device_name}`

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.

`-block {integer_value}`

(Optional argument; you must set -client or –startpage and –endpage before use.) Specifies location of block for memory read.

`-client {client_name}`

Name of client for memory read.

`-startpage {integer_value}`

Startpage for page range; value must be an integer. You must specify a –endpage and -block along with this argument.

`-endpage {integer_value}`

Endpage for page range; value must be an integer. You must specify a –startpage and -block along with this argument.

`-access {all | status | data}`

(Optional argument; you must set -client or –startpage, –endpage and –block before use.) Specifies what eNVM information to check: page status, data or both.

Value	Description
all	Shows the number of pages with corruption status, data corruption and out-of-range write count (default)
status	Shows the number of pages with corruption status and the number of pages with out-of-range write count
data	Shows only the number of pages with data corruption

`-file {filename}`

(Optional argument; you must set -client or –startpage, –endpage and –block before use.) Name of output file for memory read.

Supported Families

SmartFusion, Fusion

Exceptions

None

Example

The following command reads the flash memory for the client 'DS8bit' and reports the data in a logfile 'readFlashMemoryReport.log':

```
read_flash_memory -client {DS8bit} -file {readFlashMemoryReport.log}
read_flash_memory –startpage 0 –endpage 2 –block 0 –access {data}
```

read_flashrom

Reads the content of the FlashROM from the selected device.

```
read_flashrom [-name {device_name}] [-mapping {logical | physical}] [-file {filename}]
```

Arguments

-name *device_name*

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.

-mapping {*logical | physical*}

(Optional) Specifies how the data read from the UFROM is mapped. Values are explained in the table below.

Value	Description
logical	Logical mapping (default)
physical	Physical mapping

-file {*filename*}

(Optional) Identifies the name of the file to which read results will be saved.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

None

Example

The following reads the FROM content on the device 'AFS600' and sets to physical mapping:

```
read_flashrom -name {AFS600} -mapping {physical}
```

read_id_code

The command reads IDCode from the device without masking any IDCode fields. This is the raw IDcode from the silicon.

Note: Being able to read the IDCode is an indication that the JTAG interface is working correctly.

```
read_id_code [-name {device_name}]
```

Arguments

-name *device_name*

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

None

Example

The following command reads the IDCODE from the device 'AFS600':

```
read_id_code –name {AFS600}
```

recover_flash_memory

The command removes ECC2 errors due to memory corruption by reprogramming specified flash memory (NVM) pages and initializing all pages to zeros. The recovery affects data blocks and auxiliary blocks.

The write counters of the corrupted pages might not be accurate due to corruption. The recovery operation will not change state of the page write counters.

Use the check_flash_memory command to detect flash memory errors.

```
recover_flash_memory
[-name {device_name}]
[-block {integer_value}]
[-client {client_name}]
[-startpage {integer_value}]
[-endpage {integer_value}]
```

At a minimum you must specify `-client <name>` OR

```
-startpage <page_number> -endpage <page_number> -block <number>
```

Arguments

`-name  {device_name}`

Optional user-defined device name. The device name is not required if there is only one device in the current configuration, or a device has already been selected using the set_debug_device command.

`-block {integer_value}`

(Optional argument; you must set -client or –startpage and –endpage before use.) Specifies location of block for memory recovery.

`-client {client_name}`

Name of client for memory recovery.

`-startpage {integer_value}`

Startpage for page range; value must be an integer.You must specify a –endpage and -block along with this argument.

`-endpage {integer_value}`

Endpage for page range; value must be an integer. You must specify a –startpage and -block along with this argument.

Supported Families

SmartFusion, Fusion

Exceptions

None

Example

The following command recovers flash memory data in the client 'DS8bit':

```
recover_flash_memory -client {DS8bit}
```

The following command recovers flash memory from block 0, startpage 0, and endpage 3:

```
recover_flash_memory -block 0 -startpage 0 -endpage 3
```

refresh_prg_list

Refreshes the programmer list. This is useful to have FlashPro detect a programmer that you have just connected.

```
refresh_prg_list
```

Arguments

None

Supported Families

All

Exceptions

None

Example

```
refresh_prg_list
```

remove_device

Removes the device from the chain. Chain programming mode must be set.

```
remove_device (-name {name})*
```

Arguments

`-name {name}`

Specifies the device name. You can repeat this argument for multiple devices.

Supported Families

All

Exceptions

None

Example

Remove a device 'A3P250' from the chain:

```
remove_device (-name {A3P250})*
```

remove_prg

Removes the programmer from the programmer list.

```
remove_prg (-name {name})*
```

Arguments

`-name {name}*`

Specifies the programmer to be removed. You can repeat this argument for multiple programmers.

Supported Families

All

Exceptions

None

Example

The following example removes the programmer '03178' from the programmer list:

```
remove_prg (name {03178})*
```

run_selected_actions

Runs the selected action on the specified programmer and returns the exit code from the action. If no programmer name is specified, the action is run on all connected programmers. Only one exit code is returned, so return code cannot be used when action is run on more than one programmer. A programming file must be loaded.

```
run_selected_actions [(-name {name})*]
```

Arguments

`-name {name}`

Optional argument that specifies the programmer name. You can repeat this argument for multiple programmers.

Supported Families

All

Exceptions

None

Example

The following example runs the selected actionS on the programmers 'FP30085' and 'FP30086'.

```
run_selected_actions -name {FP300085} -name {FP300086}
```

Example using return code:

```
if {[catch {run_selected_actions} return_val]} {puts "Error running Action"} else {puts
"exit code $return_val"}
```

Example returning exit code to the command line (returns exit 99 on script failure, otherwise returns exit code from selected action):

```
if {[catch {run_selected_actions} return_val]}{exit 99} else {exit $return_val}
```

save_log

Saves the log file.

```
save_log -file {file}
```

Arguments

`-file {file}`

Specifies the log filename.

Supported Families

All

Exceptions

None

Example

The following example saves the log file with the name 'my_logfile1.log':

```
save_log -file {my_logfile1.log}
```

save_project

Saves the FlashPro project.

```
save_project
```

Arguments

None

Supported Families

All

Exceptions

None

Example

```
save_project
```

save_project_as

Saves the FlashPro project under a new project name.

```
save_project_as -name {name} -location {location}
```

Arguments

`-name {name}`

Specifies the project name.

`-location {location}`

Specifies the project location.

Supported Families

All

Exceptions

None

Example

The following example saves the FlashPro project 'FPPrj2' to the directory 'FPProject2':

```
save_project_as -name {FPPrj2} -location {./FPProject2}
```

scan_chain_prg

In single mode, this command runs scan chain on a programmer.

In chain mode, this command runs scan and check chain on a programmer if devices have been added in the grid.

```
scan_chain_prg [(-name {name})+]
```

Arguments

`-name {name}`
Specifies the programmer name.

Supported Families

All

Exceptions

None

Example

The following example runs scan chain on a single programmer (single mode) named '21428':

```
scan_chain_prg -name {21428}
```

select_serial_range

Selects the serialization data. Either the *data* or the *from_data* and *to_data* arguments must be specified.

A programming file allowing serialization must be loaded.

```
select_serial_range [-name {name}] [(-data {data})*] [-from_data {from_data} -to_data
{to_data}]
```

Arguments

`-name {name}`
Specifies the device name. This argument must be specified in chain programming mode.
`-data {data}`
Specifies the data. You can repeat this argument for multiple serialization data.
`-from_data {from_data}`

Specifies the start of a range of data. Must be used along with the to_data argument.

`-to_data {to_data}`

Specifies the end of a range of data. Must be used along with the from_data argument.

Supported Families

All

Exceptions

None

Examples

The following example selects serial range data on devices 1-4 in Single Programming mode:

```
select_serial_range -data {1} -data {2} -data {3} -data {4}
select_serial range -from data {1} -to_data {4}
```

The following example selects serial range data on devices 1-4 on the device 'MyDevice1' in Chain Programming mode. :

```
select_serial_range -name {MyDevice1} -data {1} -data {2} -data {3} -data {4}
select_serial range -name {MyDevice1} -from data {1} -to_data {4}
```

self_test_prg

Runs Self-Test on a programmer.

```
self_test_prg (-name {name})*
```

Arguments

`-name {name}`

Specifies the programmer name. You can repeat this argument for multiple programmers.

Supported Families

All

Exceptions

None

Example

The following examples runs the self test on the programmer '30175':

```
self_test_prg (-name {30175})*
```

set_bsdl_file

Sets a BSDL file to a non-Actel device in the chain. Chain programming mode must have been set. The device must be a non-Actel device.

```
set_bsdl_file -name {name} -file {file}
```

Arguments

`name {name}`

Specifies the device name.

`-file {file}`

Specifies the BSDL file.

Supported Families

Any non-Actel device supported by FlashPro.

Exceptions

None

Example

The following example sets the BSDL file /design/bsdl/NewBSDL2.bsdl to the device 'MyDevice3':

```
set_bsdl_file –name {MyDevice3} –file {e:/design/bsdl/NewBSDL2.bsdl}
```

set_chain_param

Sets the chain parameters in single programming mode. Single programming mode must be set .

```
set_chain_param [-pre_ir {pre_ir}] [-pre_dr {pre_dr}] [-post_ir {post_ir}] [-post_dr
{post_dr}]
```

Arguments

`-pre_ir {pre_ir}`

Specifies the pre IR length.

`-pre_dr {pre_dr}`

Specifies the pre DR length.

`-post_ir {post_ir}`

Specifies the post IR length.

`-post_dr {post_dr}`

Specifies post DR length.

Supported Families

All

Exceptions

None

Example

The following example sets the chain parameters for pre IR length to 2, pre DR length to 3, post IR length to 4, and post DR length to 5:

```
set_chain_param –pre_ir {2} –pre_dr {3} –post_ir {4} –post_dr {5}
```

set_debug_device

Identifies the device you intend to debug.

```
set_debug_device -name {device_name}
```

Arguments

`name {device_name}`
Device name. The device name is not required if there is only one device in the current configuration.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

None

Example

The following example identifies the device 'A3P250' for debugging:

```
set_debug_device –name {A3P250}
```

set_debug_programmer

Identifies the programmer you want to use for debugging (if you have more than one). The name of the programmer is the serial number on the bar code label on the FlashPro programmer.

```
set_debug_programmer -name {programmer_name}
```

Arguments

`-name {programmer_name}`
Programmer name is the serial number on the bar code label of the FlashPro programmer.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Exceptions

None

Example

The following example selects the programmer 10841

```
set_debug_programmer -name {10841}
```

set_device_ir

Sets the IR length of a non-Actel device in the chain. Chain programming mode must be set. The device must be a non-Actel device.

```
set_device_ir -name {name} -ir {ir}
```

Arguments

-name {*name*}

Specifies the device name.

-ir {*ir*}

Specifies the IR length.

Supported Families

Any non-Actel device supported by FlashPro.

Exceptions

None

Example

The following example sets the IR length to '2' for the non-Actel device 'MyDevice4':

```
set_device_ir –name {MyDevice4} –ir {2}
```

set_device_name

Changes the user name of a device in the chain. Chain programming mode must be set .

```
set_device_name -name {name} -new_name {new_name}
```

Arguments

-name {*name*}

Identifies the old device name.

-new_name {*new_name*}

Specifies the new device name.

Supported Families

All

Exceptions

None

Example

The following example changes the user name of the device from 'MyDevice4' to 'MyDevice5':

```
set_device_name -name {MyDevice4} -new_name {MyDevice5}
```

set_device_order

Sets the order of the devices in the chain to the order specified. Chain programming mode must have been set. Unspecified devices will be at the end of the chain.

```
set_device_order (-name {name})*
```

Arguments

`-name {name}`

Specifies the device name. To specify a new order you must repeat this argument and specify each device name in the order desired.

Supported Families

All

Exceptions

None

Example

The following example sets the device order for 'MyDevice1', 'MyDevice2', 'MyDevice3', and 'MyDevice4'. 'MyDevice2' is unspecified so it moves to the end of the chain.

```
set_device_order -name {MyDevice3} -name {MyDevice1} -name {MyDevice4}
```

the new order is:

```
MyDevice3 MyDevice1 MyDevice4 MyDevice2
```

set_device_tck

Sets the maximum TCK frequency of a non-Actel device in the chain. Chain programming mode must be set. The device must be a non-Actel device.

```
set_device_tck -name {name} -tck {tck}
```

Arguments

`-name {name}`
Specifies the device name.
`-tck {tck}`
Specifies the maximum TCK frequency (in MHz).

Supported Families

Any non-Actel device supported by FlashPro.

Exceptions

None

Example

The following example sets the maximum TCK frequency of the non-Actel device 'MyDevice4':

```
set_device_tck -name {MyDevice4} -tck {2.25}
```

set_device_type

Changes the family of an Actel device in the chain. The device must be an Actel device. The device parameter below is now optional.

```
set_device_type -name {name} -type {type}
```

Arguments

`-name {name}`
Identifies the name of the device you want to change.
`-type {type}`
Specifies the device family.

Supported Families

Any Actel device supported by FlashPro.

Exceptions

None

Example

The following example sets the device 'MyDevice2' to the type A3PE600.

```
set_device_type -name {MyDevice2} -type {A3PE600}
```

set_main_log_file

Sets the FlashPro log file.

```
set_main_log_file -file {file}
```

Arguments

`-file {file}`

Specifies the log file.

Supported Families

All

Exceptions

None

Example

The following example sets the FlashPro log file to 'log1000.txt'.

```
set_main_log_file –file {e:/log/log1000.txt}
```

set_prg_name

Changes the user name of a programmer.

```
set_prg_name -name {name} -new_name {new_name}
```

Arguments

`-name {name}`

Identifies the old programmer name.

`-new_name {new_name}`

Specifies the new programmer name.

Supported Families

All devices supported by FlashPro.

Exceptions

None

Example

The following example changes the name of the programmer 'FP300086' to 'FP3Prg2':

```
set_prg_name –name {FP300086} –new_name {FP3Prg2}
```

set_programming_action

Selects the action for a device. The device name parameter must be specified only in chain programming mode. A programming file must be loaded. The device must be an Actel device.

```
set_programming_action [-name {name}] -action {action}
```

Arguments

-name {*name*}

Specifies the device name.

-action {*action*}

Specifies the action.

Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^PLUS, and ProASIC

Exceptions

Must be an Actel device

Example

The following example sets the programming action in single programming mode:

```
set_programming_action –action {PROGRAM}
```

And in chain programming mode:

```
set_programming_action –name {MyDevice1} –action {ERASE}
```

set_programming_file

Sets the programming file for a device. Either the *file* or the *no_file* flag must be specified. A programming file must be loaded. The device must be an Actel device .

```
set_programming_file [-name {name}] [-file {file}] [-no_file { }]
```

Arguments

-name {*name*}

Specifies the device name. This argument must be specified only in chain programming mode.

-file {*file*}

Specifies the programming file.

-no_file

Specifies to unload the current programming file.

Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^PLUS, and ProASIC

Exceptions

Must be an Actel device.

Examples

in single programming mode:

```
set_programming_file -file {e:/design/pdb/TopA3P250.pdb}
```

in chain programming mode:

```
set_programming_file -name {MyDevice2} -file {e:/design/pdb/TopA3P250.pdb}
set_programming_file -name {MyDevice1} -no_file
```

set_programming_mode

Sets the programming mode.

```
set_programming_mode -mode {single|chain}
```

Arguments

`-mode {single|chain}`

Specifies the mode, either single programming or chain programming.

Supported Families

All devices supported by FlashPro.

Exceptions

None

Example

The following example sets the programming mode to 'single':

```
set_programming_mode -mode {single}
```

set_serialization_log_file

Sets the FlashPro log file to be used for serialization.

```
set_serialization_log_file -file {file}
```

Arguments

`-file {`*`file`*`}`

Specifies the log file name.

Supported Families

All devices supported by FlashPro.

Exceptions

None

Example

The following device sets the log file used for serialization to 'log1000_serial.txt':

```
set_serialization_log_file -file {e:/log/log1000_serial.txt}
```

set_serialization_mode

Sets the serialization mode.

```
set_serialization_mode -mode {skip|reuse}
```

Arguments

```
-mode {skip|reuse}
```

Supported Families

All devices supported by FlashPro.

Exceptions

None

Example

The following example sets the serialization mode to 'skip':

```
set_serialization_mode -mode {skip}
```

Signature

Optional for AFM, DIO, and FUS file types.

update_programming_file

Updates the programming file with the selected parameters.

```
update_programming_file
[(-name {name})*]
-feature {value}
-signature {value}
-from_content {name}
```

```
-from_config_file {name}
-number_of_devices {value}
-from_program_pages {value}
-custom_security {value}
-security_permanent {value}
-fpa_security_level {value}
-from_security_level {value}
-efm_block_security{location:X;security_level: value}
-pass_key {value} -aes_key {value}
-efm_content {location:X;source: value}
-efm_block {location:X;config_file: value}
-efm_client {location:X;client:value; mem_file: value}
-tie_off_arch {value}
-set_io_state {value}
-pdb_file {name}
-enable_m3debugger {value}
```

Arguments

`-name {name}`

Specifies the device name. This argument must be specified only in chain programming mode.

`-feature {value}`

Select the silicon feature(s) you want to program. Possible values for this option are listed in the table below, or the instance-specific program options available only for specific families (as shown in the table below). Actel recommends that you specify your program parameters for each Embedded Flash Memory Block (EFMB) instance, from 0-3. The instance specific program options replace [-feature {*value*}].

value	Family
{setup_security:on/off}	SmartFusion
{prog_fpga:on/off}	SmartFusion
{prog_from:on/off}	SmartFusion
{prog_nvm:on/off}	SmartFusion
{setup_security}	Fusion
{prog_from}	Fusion
{all}	IGLOO; ProASIC3

To program the Embedded Flash Memory Block, use the following EFM arguments: -efm_block, -efm_client, and -efm_block_security.

`-signature {value}`

Optional argument that identifies and tracks Actel designs and devices.

`-from_content {name}`

Optional argument that identifies the source file for the FlashROM content. The file type is UFC or PDB (default). This argument only applies when programming the FlashROM (prog_from option).

`-from_config_file {name}`

Optional argument that specifies the location of the FlashROM configuration file. This argument only applies when programming the FlashROM (prog_from option) and the from_content is set to UFC.

`-number_of_devices{value}`

Optional argument that specifies the number of devices to be programmed. This argument only applies when FlashROM has serialization regions. This argument only applies when programming the FlashROM (prog_from option).

`-from_program_pages {value}`

Optional argument that identifies the program pages in FlashPoint. This argument only applies when programming the FlashROM (prog_from option).

`-custom_security {value}`

Optional argument that specifies the security level. This argument only applies when programming the security settings (setup_security) or programming previously secured devices. The following table shows the acceptable values for this argument:

Value	Description
Yes	Custom security level
No	Standard security level

`-security_permanent {value}`

Optional argument that specifies whether the security settings for this file are permanent or not. This argument only applies when programming the security settings (setup_security) or programming previously secured devices. The following table shows the acceptable values for this argument:

Value	Description
Yes	Permanently disables future modification of security settings
No	Enables future modifications of security settings

`-fpga_security_level {value}`

Optional argument that specifies the security level for the FPGA Array. This argument only applies when programming the security settings (setup_security) or programming previously secured devices. Possible values:

Value	Description
write_verify_protect	The security level is medium (standard) and the FPGA Array cannot be written or verified without a Pass Key
write_protect	The security level is write protected. The FPGA Array cannot be written without a Pass Key, but it is open for verification (custom FPGA)

Value	Description
encrypted	The security level is high (standard) and uses a 128-bit AES encryption
none	The FPGA Array can be written and verified without a Pass Key

`-from_security_level {value}`

Optional argument that specifies the security level for the FlashROM. This argument only applies when programming the security settings (setup_security) or programming previously secured devices. Possible values:

Value	Description
write_verify_protect	The security level is medium (standard) and the FlashROM cannot be read, written or verified without a Pass Key
write_protect	The security level is write protected. The FlashROM cannot be written without a Pass Key, but it is open for reading and verification (custom FlashROM)
encrypted	The security level is high (standard) and uses a 128-bit AES encryption
none	The FlashROM can be written and verified without a Pass Key

`-efm_block_security{location:X;security_level: value}`

This option is available only for SmartFusion and Fusion; X identifies an Embedded Flash Memory Block instance from 0-3.

Possible values:

Value	Description
write_verify_protect	The security level is medium (standard) and the Embedded Flash Memory Block cannot be read, written or verified without a Pass Key
write_protect	The security level is write protected. The Embedded Flash Memory Block cannot be written without a Pass Key, but it is open for reading (custom FB)
encrypted	The security level is high (standard) and uses a 128-bit AES encryption
none	The Embedded Flash Memory Block can be written and read without a Pass Key

`-pass_key {value}`

Protects all the security settings for FPGA Array, FlashROM, and Embedded Flash Memory Block. The maximum length of this value is 32 characters. You must use hexadecimal characters for the pass key value.

`-aes_key {value}`

Decrypts FPGA Array and/or FlashROM and Embedded Flash Memory Block programming file content.

Max length is 32 HEX characters.

`-efm_content {location:X;source: value}`

This option is available only for SmartFusion and Fusion; X identifies an Embedded Flash Memory Block from 0-3. Option identifies the source file for the Embedded Flash Memory Block configuration content, either an EFC or PDB file. If you specify EFC as your source, you need to specify the -efm_block parameter. Possible values:

Value	Description
PDB	Embedded Flash Memory Block configuration and content is taken from your PDB
EFC	FlashPoint uses the Embedded Flash Memory Block configuration and content from the EFC file specified in -efm_block parameter

`-efm_block {location:X;config_file: value}`

This option is available only for SmartFusion and Fusion; X identifies an Embedded Flash Memory Block (EFMB) from 0-3.

Config_file specifies the location of the EFMB configuration file (must be an EFC file with full pathname).

`-efm_client {location:X;client:value; mem_file: value}`

This option is available only for SmartFusion and Fusion; X identifies an EFMB from 0-3.

You must specify the client name and its memory content file for each client of EFMB you wish to program.

Mem_file specifies the file with the memory content for the client. If you want to program a client with a PDB or EFC file memory content (as defined by the -efm_content argument), the mem_file path should be empty (see example 3); but if a mem_file path is specified, the memory content from this file will overwrite the client content in PDB or EFC (as defined by the -efm_content argument).

`-tie_off_arch {value}`

This optional argument is used only for IT6X6M2 and M7IT6X6M2 devices. Possible values:

Value	Description
pull-down	Pull-down resistor: reduced quiescent power consumption
pull-up	Pull-up resistor: compatible behavior for migrated ProASIC[PLUS] designs

`-set_io_state {value}`

Sets the I/O state during programming by port name or pin number.

To set the I/O by port name, use `-set_io_state {portName:<port name>; state:<state>}`.

To set the I/O by pin number, use `-set_io_state {pinNumber:<pin number>; state:<state>}`.

To set all I/Os to the specified state, use `-set_io_state {all; state:<state>}`.

Possible state values:

Value	Description

Value	Description
Tri-State	Sets the I/O state to tristate
Last Known State	Sets the I/O state to last known state
High	Sets the I/O state to high
Low	Sets the I/O state to low

`-pdb_file {name}`

Optional PDB filename; if not specified the default is 'expresspdbX'.

`-enable_m3debugger {yes | no}`

SmartFusion only; .enables the M3 debugger.

Supported Families

All devices supported by FlashPro.

Exceptions

None

Example

Fusion example 1

```
update_programming_file \
-feature {setup_security} \
 -feature {prog_from} \
 -from_content {ufc} \
-from_config_file {D:/from_ah.ufc} \
 -number_of_devices {1} \
 -from_program_pages {1 2 3 4 5 6 7 } \
 -custom_security {no} \
-security_permanent {no} \
 -fpga_security_level {write_verify_protect} \
 -from_security_level {write_verify_protect} \
-efm_block_security {location:1;security_level:write_verify_protect} \
 -efm_content {location:1;source:efc} \
 -efm_block {location:1;config_file:{D:\ nvm_all_new.efc}} \
-efm_client {location:1;client:asb;mem_file:{}} \
-efm_client {location:1;client:cfiData;mem_file:{D:\cfid.mem}} \
 -efm_client {location:1;client:ds;mem_file:{D:\ds.hex}} \
-efm_client {location:1;client:init1;mem_file:{D:\init1.hex}} \
 -efm_client {location:1;client:raminit;mem_file:{}}
```

Update loaded PDB file: use ufc file for FlashROM configuration and content; use efc file for block 1 configuration; efc memory content will be overwritten by memory content from specified mem files for each client.

Fusion example 2:

```
update_programming_file \
-feature {prog_from} \
-from_content {pdb} \
-from_program_pages {1} \
-efm_content {location:1;source:pdb} \
-efm_client {location:1;client:cfiData;mem_file:{D:\cfid.mem}}
```

Update loaded PDB file: use pdb data for FlashROM; program only page 1; use pdb data for block 1; program only client cfiData; overwrite memory content for this client with memory content from the specified file.

Fusion example 3:

```
update_programming_file \
-efm_content {location:1;source:pdb} \
 -efm_client {location:1;client:cfiData;mem_file:{D:\cfid.mem}} \
-efm_client {location:1;client:init1;mem_file:{}}
```

Update loaded PDB file: use pdb data for block 1; program client cfiData using memory content from the specified file; program client init1 using memory content from pdb (no mem_file path is specified)

Troubleshooting

Loopback Test

The console software supports all JTAG functions and a diagnostic loopback test. Note that loopback is not supported on all boards.

To perform the diagnostic test

1. Connect the loopback test board to the FlashPro.
2. Connect the FlashPro to the parallel/USB port of the PC.
3. Power-on the FlashPro.
4. From the **Start** menu, choose **Programs** > **Actel FlashPro** > **Diagnostics**. This opens the diagnostics console program.
5. Connect to the FlashPro by entering *openport lpt1* or *openport usb*. The parallel port number depends on the port used to connect the FlashPro.
6. Enter *test*. The unit runs into self-test mode. Do not interrupt the unit during self-test mode.

Note: Note: To see a complete list of all functions, enter *help*. To get a more detail description of each function, enter *help* <*command*>.

IGLOO, ProASIC3, SmartFusion and Fusion Device Exit Codes for Software v8.6 and Above

The table below lists exit codes for IGLOO, ProASIC3, SmartFusion and Fusion devices in software v8.6 and ABOVE only. See the Device Exit Codes for pre-v8.6 Software help topic for exit codes for older versions.

Note: Exit codes with positive integers are reserved for current and future standard EXIT codes of the STAPL standard. Exit codes with negative integers are reserved for vendor-specific EXIT codes.

Table 22 · Exit Codes for IGLOO, ProASIC3, SmartFusion and Fusion Family Devices in Software v8.6 and Above

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
	0	Passed (no error)		
	1	A physical chain does not match the expected set	Physical chain configuration has been altered. Something has become disconnected in the chain. The specific IR length of non-Actel	

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		up from the STAPL file. Also known as Checking Chain Error.	devices may be incorrect. The order of the specified chain may be incorrect.	
0x8052	5	Failed to enter programming mode.	Unstable VPUMP voltage level. Signal integrity issues on JTAG pins. Older software or programming file used.	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection. Generate STAPL file with the latest version of Designer/FlashPro. Use latest version of FlashPro software.
0x801D 0x8053	6	Failed to verify IDCODE	Incorrect programming file Incorrect device in chain Signal integrity issues on JTAG pins	Choose the correct programming file and select the correct device in chain. Measure JTAG pins and noise or reflection. If TRST is left floating then add pull-up to pin. Reduce the length of ground connection.
0x8005 0x8007 0x8009 0x800B	6	Failed to verify AES Sec.	Programming file generated with an older version of software	Generate STAPL file with the latest version of Designer/FlashPro. Use latest version of FlashPro software. Try again at a slower TCK. Contact Actel Technical Support.
0x8008	6	Failed to verify	File is not for M7, but target device is M7	Check that the target device is M7 enabled.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		IDCODE. Target is an M7 device	Signal integrity issues on JTAG pins.	Make sure that the programming file you generated is for an M7-enabled device. Measure JTAG pins, noise and reflection.
0x800A	6	Failed to verify IDCODE Target is an M1 device	Files not for M1, but target device is M1. Signal integrity issues on JTAG pins	Check that the target device is M1 enabled. Make sure the programming file generated is for an M1-enabled device. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x800C	6	Failed to verify IDCODE. Core enabled device detected	File is not for target device. Signal integrity issues on JTAG pins	Check the target device; make sure the programming file generated is matches the target device. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x800D	6	Failed to verify IDCODE. The target is not M7 device	File is for M7 but target device is not M7. Signal integrity issues on JTAG pins.	Check that the target device is not M7 enabled. Make sure that the programming file generated is for non-M7 enabled device. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x800E	6	Failed to verify IDCODE. Target is not an M1 device	File is for M1, but target device is not M1. Signal integrity issues on JTAG pins	Check that the target device is not M1 enabled. Make sure that the generated programming file is for non-M1 enabled device. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
0x8006	6	Failed to verify IDCODE. Target is not a P1 device	File is not for P1, but target device is a P1 device. Signal integrity issues on JTAG pins	Check that the target device is P1 enabled. Make sure programming file generated is for M1 enabled device. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x801E	6	A3PE600 Engineering Sample Device Detected. This device is supported with pre-v8.3 SP1 STAPL files only		Contact Actel Technical Support
0x8057	8	Failed Erase Operation.	Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins.	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x8058	10	Failed to program FPGA array at row <row number>.	Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only)	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
			Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins.	Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x805D 0x805E 0x807B	10	Failed to enable FPGA Array.	Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins.	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x8061 0x8062	10	Failed to program FlashROM.	Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins.	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x801B 0x801C 0x806C 0x806D 0x806E	10	Error programming Embedded Flash Memory Block (EFMB)	Unstable VCC_NVM/VCC_OSC voltage level (Fusion only) Unstable VCC_ENVM/VCC_RCOSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins NVM corruption is possible when	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
			writing from your design; check the NVM status for confirmation.	Reset signal is not properly tied off in your design. Inspect device using Device Debug.
0x807D 0x807E	10	Error programming system init and boot clients	Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection. Inspect device using Device Debug.
0x8069 0x806A 0x806B	10	Error programming Embedded Flash Memory Block (EMFB)	Programming file generated with an older version of software	Generate STAPL file with the latest version of Designer/FlashPro; use the latest version of FlashPro software Try again at a slower TCK Inspect device using Device Debug. Contact Actel Technical Support
0x807F 0x8080	10	Error programming system init and boot clients	Programming file generated with an older version of software	Generate STAPL file with the latest version of Designer/FlashPro; use the latest version of FlashPro software Try again at a slower TCK Inspect device using Device Debug. Contact Actel Technical Support
0x8059	11	Verify 0 failed	Unstable VPUMP voltage level.	Monitor related power supplies that cause the

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
0x805B		at row <row number> Verify 1 failed at row <row number>.	Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins.	issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x8060	11	Failed to verify FlashROM at row <FlashROM row number>.	Device is programmed with a different design. Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins.	Run VERIFY_DEVICE_INFO to verify the device is programmed with the correct data/design. Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x8075 0x8076 0x8077	11	Failed to verify Embedded Flash Memory Block (EFMB)	Device is programmed with a different design. Unstable VCC Unstable VCC_NVM/VCC_OSC (Fusion only) Unstable VCC_ENVM/VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins.	Verify the device is programmed with the correct data/design. Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Measure JTAG pins, and noise or reflection. Run DEVICE_INFO to confirm if the target

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
			The EFMB data was modified in your FPGA design after programming. This could have occurred during standalone verify. The target EFMB is locked with FlashLock when running ACTION PROGRAM_NVM_ACTIVE_ARRAY or VERIFY_NVM_ACTIVE_ARRAY.	EFMB block is locked with FlashLock (pass key). If the target EFMB block is locked, then you must unlock it by erasing the security and then reprogramming with the desired security settings. After unlocking the target EFMB block attempt to rerun the target ACTION. Inspect device using Device Debug.
0x8085 0x8086	11	Failed to verify system init and boot clients	Device is programmed with a different design. Unstable VCC Unstable VCC_NVM/VCC_OSC (Fusion only) Unstable VCC_ENVM/VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins. The EFMB data was modified in your FPGA design after programming. This could have occurred during standalone verify. The target EFMB is locked with FlashLock when running ACTION PROGRAM_NVM_ACTIVE_ARRAY or VERIFY_NVM_ACTIVE_ARRAY.	Verify the device is programmed with the correct data/design. Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Measure JTAG pins, and noise or reflection. Run DEVICE_INFO to confirm if the target EFMB block is locked with FlashLock (pass key). If the target EFMB block is locked, then you must unlock it by erasing the security and then reprogramming with the desired security settings. After unlocking the target EFMB block attempt to rerun the target ACTION. Inspect device using Device Debug.
0x8072 0x8073	11	Failed to verify	Programming file generated with an older version of software	Generate STAPL file with the latest version of Designer/FlashPro; use the latest version of

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
0x8074		Embedded Flash Memory Block (EFMB)		FlashPro software Try again at a slower TCK Inspect device using Device Debug. Contact Actel Technical Support
0x8083 0x8084	11	Failed to verify system init and boot clients	Programming file generated with an older version of software	Generate STAPL file with the latest version of Designer/FlashPro; use the latest version of FlashPro software Try again at a slower TCK Inspect device using Device Debug. Contact Actel Technical Support
0x8014 0x8015	11	Failed to verify calibration data	Unstable VCC Unstable VCC_NVM/VCC_OSC (Fusion only) Unstable VCC_ENVM/VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection. Try reprogramming. Workaround: Disable optional procedure CHECK_AND_BACKUP_CALIB
0x805A 0x805C	11	Verify 0 failed at row <row number> . Verify 1 failed	Device is programmed with a different design Unstable VPUMP voltage level.	Run VERIFY_DEVICE_INFO to verify the device is programmed with the correct data/design. Monitor related power supplies that cause the

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		at row <row number>	Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins	issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x8063	14	Failed to program Silicon Signature. Failed to program security lock settings.	Signal integrity issues on JTAG pins.	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x8068	-18	Failed to authenticate the encrypted data.	Incorrect AES key. Signal integrity issues on JTAG pins.	Generate a programming file with the correct AES key. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x805F	-20	Failed to verify FlashROM at row <FlashROM row number>.	Programming file generated with an older version of software Device is programmed with a different design. Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level	Generate STAPL file with the latest version of Designer/FlashPro; use the latest version of FlashPro software. Program with the correct data/design. Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Measure JTAG pins and noise or reflection.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
			(SmartFusion only) Signal integrity issues on JTAG pins.	
0x8065	-22	Failed to program pass key.	Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins.	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x8066	-23	Failed to program AES key.	Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins.	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Measure JTAG pins and noise or reflection.
0x8055 0x8056	-24	Failed to program UROW.	Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins.	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection. Make sure you mounted **0.01µF** and **0.33µF** caps on Vpump (close to the pin).

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
0x802A	-27	FlashROM Write/Erase is protected by the passkey. A valid passkey needs to be provided.	File contains no passkey and device is secured with a passkey. Passkey in the file does not match device.	Provide a programming file with a passkey that matches the passkey programmed into the device.
0x8025	-28	FPGA Array Write/Erase is protected by the passkey. A valid pass key needs to be provided.	File contains no passkey and device is secured with a passkey. Passkey in the file does not match device.	Provide a programming file with a passkey that matches the passkey programmed into the device.
0x802B 0x802D	-29	FlashROM Read is protected by passkey. A valid passkey needs to be provided.	File contains no passkey and device is secured with a passkey. Passkey in the file does not match device.	Provide a programming file with a pass key that matches the passkey programmed into the device
0x8024 0x8026	-30	FPGA Array verification is protected by a passkey. A valid passkey needs to be provided.	File contains no passkey and device is secured with a passkey. Passkey in the file does not match device.	Provide a programming file with a passkey that matches the passkey programmed into the device.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
0x804B 0x8001	-31	Failed to verify AES key.	AES key in the file does not match the device. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Unstable JTAG/VPUMP voltage level.	Provide a programming file with an AES key that matches the AES key programmed into the device. Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x8000	-31	Failed to verify AES key.	Programming file generated with an older version of software	Generate STAPL file with the latest version of Designer/FlashPro; use the latest version of FlashPro software. Try again at a slower TCK Contact Actel Technical Support
0x8020 0x8022 0x8028	-33	FPGA Array encryption is enforced. A programming file with encrypted FPGA array data needs to be provided.	File contains unencrypted array data, but device contains AES key.	Provide a programming file with an encrypted FPGA Array data.
0x802C 0x802F	-34	FlashROM encryption is enforced. A programming file with	File contains unencrypted FlashROM data, but the device contains an AES key.	Provide a programming file with an encrypted FlashROM data.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		encrypted FlashROM data needs to be provided.		
0x801F 0x804A	-35	Failed to match pass key.	Pass key in file does not match pass key in device.	Provide a programming file with a pass key that matches the pass key programmed into the device.
0x802E 0x8030	-36	FlashROM Encryption is not enforced. Cannot guarantee valid AES key present in target device. Unable to proceed with Encrypted FlashROM programming.	File contains encrypted FlashROM, but device encryption is not enforced for FlashROM	Regenerate security programming file with proper AES key. Program device security. Retry programming FlashROM with encrypted programming file.
0x8021 0x8023 0x8027 0x8029	-37	FPGA Array Encryption is not enforced. Cannot guarantee valid AES key present in target device. Unable to proceed with	File contains encrypted FPGA Array, but the device encryption is not enforced for FPGA Array.	Regenerate security programming file with proper AES key. Program device security. Retry programming FPGA Array with encrypted programming file.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		Encrypted FPGA Array verification.		
0x8067	-38	Failed to program pass key.	Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins. Bad device.	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Measure JTAG pins and noise or reflection.
0x806F 0x8070 0x8071 0x8081 0x8082 0x8089	-39	ERROR: 2 or more errors found on this page	Unstable VCC_NVM/VCC_OSC voltage (Fusion only) Unstable VCC_ENVM/VCC_ROSC (SmartFusion only) NVM reset signal is floating in user design 2 or more ECC errors found when reading the eNVM	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Bias NVM reset to a logic state in user design. Try reprogramming.
0x8010	-39	ERROR: 2 or more errors found on this page.	2 or more ECC errors found when reading the master calibration data	The master calibration data has been corrupted. Try restoring master calibration from backup, if it exists, by running RECOVER_CALIB. Workaround: Disable optional procedure CHECK_AND_BACKUP_CALIB
0x8013	-39	ERROR: 2 or more errors	2 or more ECC errors found when verifying the backup calibration	Rerun action to attempt to write backup calibration again.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		found on this page.		Workaround: Disable optional procedure CHECK_AND_BACKUP_CALIB
0x8078 0x8079 0x807A 0x8087 0x8088	-40	Embedded Flash Memory Block MAC Failure.	Data in the file is encrypted with a different AES key than the device.	Verify the programming file is generated from the latest version of Designer/FlashPro.
0x8002 0x8003	-42	Failed to verify security settings.	File security settings do not match device.	Provide a programming file with security setting that match the security settings programmed into the device.
0x8004	-43	Failed to verify design information.	File checksum and design name do not match the device.	Verify the device is programmed with the correct data and design.
0x8049	-44	Failed to verify AES key.	The AES key in the file does not match the AES key in the device. File does not contain an AES key and the device is secured with an AES key.	Provide a programming file with an AES key that matches the AES key programmed into the device.
0x8054	-45	Device package does not match the programming file.	Programming file was generated with an older version of software	Generate STAPL file with the latest version of Designer/FlashPro; use the latest version of FlashPro software.
0x8033 0x8038 0x803D 0x8042 0x8045 0x8046 0x8047 0x8048	-46	Embedded Flash Memory Block X Read is protected by pass key. A valid pass key needs to be	File contains no pass key or incorrect pass key but EFMB read is secured with a pass key.	Provide a programming file with the correct pass key.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		provided.		
0x8034 0x8039 0x803E 0x8043	-46	Embedded Flash Memory Block (EFMB) block X Read is not protected by pass key. EFMB content is not secure after encrypted programming. Unable to proceed with encrypted NVM programming.	File contains encrypted EFMB for block X but the device encryption is not enforced for EFMB block X.	Regenerate security programming file with the proper AES key. Program device security. Retry programming with EFMB block X with encrypted programming file.
0x8032 0x8037 0x803C 0x8041	-47	Embedded Flash Memory Block (EFMB) block X encryption is enforced. A programming file with encrypted EFMB data	The programming EFMB data is not encrypted, but the device contains an AES key with encryption enforced.	Provide a programming file with encrypted EFMB data.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		needs to be provided.		
0x8031 0x8036 0x803B 0x8040	-48	Embedded Flash Memory Block (EFMB) block X Write is protected by pass key. A valid pass key needs to be provided.	File contains no pass key or incorrect pass key, but device is secured with a pass key.	Provide a programming file with a passkey that matches the passkey programmed into the device.
0x8035 0x803A 0x803F 0x8044	-49	Embedded Flash Memory Block (EFMB) block X Encryption is not enforced. Cannot guarantee valid AES key present in target device. Unable to proceed with Encrypted EFMB	File contains encrypted EFMB for block X, but the device encryption is not enforced for EFMB block X.	Regenerate security programming file with proper AES key. Program device security. Retry programming EFMB block X with encrypted programming file.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		programming.		
0x801A	-50	No backup calibration data found or backup calibration data has been corrupted	No backup calibration copy has been made or the backup copy has been corrupted	If master copy is still intact, rerun Action to create backup calibration copy. Workaround: Disable optional procedure CHECK_AND_BACKUP_CALIB
8x804E	-51	Failed to access Embedded Flash Memory. (AFS600 only)	This version of the silicon does not support programming of the Embedded Flash Memory Block while the FPGA Array is active.	If programming the EFMB while the FPGA is active is not required, then use actions PROGRAM_NVM or VERIFY_NVM. Otherwise, use latest revision of silicon.
0x804F	-52	Failed to access Embedded Flash Memory. (AFS1500 only)	This version of the silicon does not support programming of the Embedded Flash Memory Block while the FPGA Array is active.	If programming the EFMB while the FPGA is active is not required, then use actions PROGRAM_NVM or VERIFY_NVM. Otherwise, use latest revision of silicon.
0x8050	-53	Failed to access Embedded Flash Memory. (AFS1500 only)	This version of the silicon does not support programming block 3 of the EFMBs while the FPGA Array is active.	If programming the EFMB while the FPGA is active is not required, then use actions PROGRAM_NVM or VERIFY_NVM. Otherwise, use EFMB blocks 0, 1, or 2, but do not use block 3.
0x8051	-54	Failed to access Embedded	FPGA Array is accessing the target EFMB block while attempting programming.	If programming the EFMB while the FPGA is active is not required, then use actions PROGRAM_NVM or VERIFY_NVM.

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		Flash Memory.	NVM reset signal is stuck in design. MSS Clock is disabled during programming. MSS Clock is not properly routed to the correct pin.	Otherwise, check the FPGA design or use a different EFMB block that is not being accessed. Check if target EFMB block logic is tied to reset. Verify that the NVM reset signal in the design is not stuck. Verify the MSS clock is enabled during programming. If the MSS clock is defined as an external I/O, then verify that it is properly routed to the correct pin.
0x808A	-55	Failed to read Embedded Flash Memory Block (EFMB)	Programming file generated with an older version of software	Generate STAPL file with the latest version of Designer/FlashPro; use the latest version of FlashPro software Try again at a slower TCK Inspect device using Device Debug Contact Actel Technical Support
0x808B	-55	Failed to read Embedded Flash Memory Block (EFMB)	Unstable VPUMP voltage level. Unstable VCC Unstable VCC_OSC (Fusion only) Unstable VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Monitor VJTAG during programming; measure JTAG signals for noise or reflection.
0x808C	-55	Failed to read Embedded	Internal errror	Contact Actel Technical Support

ERROR_CODE	Exit Code	Exit Message	Possible Cause	Possible Solution
		Flash Memory Block (EFMB)		
0x8011	-56	Failed to read calibration data		Try reprogramming. Workaround: Disable optional procedure CHECK_AND_BACKUP_CALIB
0x8012	-56	Failed to read calibration data	Unstable VCC Unstable VCC_NVM/VCC_OSC (Fusion only) Unstable VCC_ENVM/VCC_ROSC voltage level (SmartFusion only) Signal integrity issues on JTAG pins	Monitor related power supplies that cause the issue during programming; check for transients outside of Actel specifications. See your device datasheet for more information on transient specifications. Measure JTAG voltages, noise, and reflection. Try reprogramming. Workaround: Disable optional backup procedure CHECK_BACKUP_CALIB

IGLOO, ProASIC3, SmartFusion and Fusion Device Exit Codes for pre-v8.6 Software

The table below lists exit codes for IGLOO, ProASIC3, SmartFusion and Fusion devices in pre-v8.6 software only. This includes v8.5 SP2, v8.5 SP1, v8.5, etc. See the Device Exit Codes for Software v8.6 and Above help topic for exit codes for older versions.

Note: Exit codes with positive integers are reserved for current and future standard EXIT codes of the STAPL standard. Exit codes with negative integers are reserved for vendor-specific EXIT codes.

Table 23 · Exit Codes for IGLOO, ProASIC3, SmartFusion and Fusion Family Devices in pre-v8.6 Software

Exit Code	Exit Message	Possible Cause	Possible Solution
0	Passed (no error).		
1	A physical chain does not match the expected set up from the STAPL file. Also known as Checking Chain Error.	Physical chain configuration has been altered. Something has become disconnected in the chain. The specific IR length of non-Actel devices may be incorrect. The order of the specified chain may be incorrect.	
5	Failed to enter programming mode.	Unstable VPUMP voltage level. Signal integrity issues on JTAG pins. Older software or programming file used.	Monitor VPUMP voltage during programming Measure JTAG voltages, noise, and reflection. Generate STAPL file with the latest version of Designer/FlashPro. Use latest version of FlashPro software.
6	Failed to verify IDCODE.	Signal integrity issues on JTAG pins.	Measure JTAG pins, noise and reflection.
8	Failed Erase Operation.	Signal integrity issues on JTAG pins.	Monitor VPUMP voltage during programming. Measure JTAG voltages, noise, and reflection.
10	Failed to program FPGA array at row ", rowNumber,"."	Signal integrity issues on JTAG pins.	Monitor VPUMP voltage during programming. Measure JTAG voltages, noise, or reflection.

Exit Code	Exit Message	Possible Cause	Possible Solution
10	Failed to enable FPGA Array.	Signal integrity issues on JTAG pins.	Monitor VPUMP voltage during programming. Measure JTAG voltages, noise, or reflection.
10	Failed to program FlashROM.	Signal integrity issues on JTAG pins.	Monitor VPUMP voltage during programming. Measure JTAG voltages, noise, and reflection.
11	Verify 0 failed at row",rowNumber,"." Verify 1 failed at row",rowNumber,"." Failed to verify FlashROM at row",from rowNumber-1.	Device is programmed with a different design. Signal integrity issues on JTAG pins.	Run VERIFY_DEVICE_INFO to verify the device is programmed with the correct data/design. Monitor VPUMP voltage during programming. Measure JTAG voltages, noise and reflection .
14	Failed to program Silicon Signature. Failed to program security lock settings.	Signal integrity issues on JTAG pins.	Monitor VPUMP voltage during programming. Measure JTAG voltages, noise, and reflection.
-18	Failed to authenticate the encrypted data.	Incorrect AES key. Signal integrity issues on JTAG pins.	Generate a programming file with the correct AES key. Measure JTAG voltages, noise and reflection
-20	Failed to verify FlashROM at row ", FRomRowNumber-	Device is programmed with a different design. Signal integrity issues on JTAG pins.	Program with the correct data/design. Monitor VPUMP level

Exit Code	Exit Message	Possible Cause	Possible Solution
	1.		during programming. Measure JTAG pins and noise or reflection.
-22	Failed to program pass key.	Unstable VPUMP voltage level. Signal integrity issues on JTAG pins.	Monitor VPUMP voltage during programming. Measure JTAG voltages, noise, and reflection.
-23	Failed to program AES key.	Unstable VPUMP voltage level. Signal integrity issues on JTAG pins.	Monitor VPUMP voltage during programming. Measure JTAG pins and noise or reflection.
-24	Failed to program UROW.	Unstable VPUMP voltage level. Signal integrity issues on JTAG pins.	Monitor VPUMP voltage during programming. Measure JTAG voltages, noise, and reflection. Make sure you mounted **0.01iF** and **0.33iF** caps on Vpump (close to the pin).
-25	Failed to enter programming mode	Signal integrity issues on JTAG pins.	Measure JTAG voltages, noise, and reflection.
-26	Failed to enter programming mode	Signal integrity issues on JTAG pins.	Measure JTAG voltages, noise, and reflection.
-27	FlashROM Write/Erase is protected by the passkey. A valid passkey needs to be	File contains no passkey and device is secured with a passkey. Passkey in the file does not match device.	Provide a programming file with a passkey that matches the passkey programmed into the device.

Exit Code	Exit Message	Possible Cause	Possible Solution
	provided.		
-28	FPGA Array Write/Erase is protected by the passkey. A valid pass key needs to be provided.	File contains no passkey and device is secured with a passkey. Passkey in the file does not match device.	Provide a programming file with a passkey that matches the passkey programmed into the device.
-29	FlashROM Read is protected by passkey. A valid passkey needs to be provided.	File contains no passkey and device is secured with a passkey. Passkey in the file does not match device.	Provide a programming file with a pass key that matches the passkey programmed into the device
-30	FPGA Array verification is protected by a passkey. A valid passkey needs to be provided.	File contains no passkey and device is secured with a passkey. Passkey in the file does not match device.	Provide a programming file with a passkey that matches the passkey programmed into the device.
-31	Failed to verify AES key.	AES key in the file does not match the device. Unstable JTAG/VPUMP voltage level.	Provide a programming file with an AES key that matches the AES key programmed into the device. Monitor VPUMP/VJTAG voltage during programming. Measure JTAG voltages, noise, and reflection.

Exit Code	Exit Message	Possible Cause	Possible Solution
-32	Failed to verify IDCODE. Target is an M7 device	File is not for M7, but target device is an M7. Signal integrity issues on JTAG pins.	Check that the target device is M7 enabled. Make sure programming file generated is for M7 enabled device. Measure JTAG pins , noise, and reflection.
-32	Failed to verify IDCODE. Target is an M1 device	File is not for M1, but target device is an M1 device. Signal integrity issues on JTAG pins.	Check that the target device is M1 enabled. Make sure programming file generated is for M1 enabled device. Measure JTAG pins, noise, and reflection.
-32	Failed to verify IDCODE. Core enabled device detected	File is not for target device. Signal integrity issues on JTAG pins	Check the target device. Make sure programming file generated for target device. Measure JTAG voltages, noise, and reflection.
-32	Failed to verify IDCODE. The target is not an M7 device	File is for M7, but target device is not M7. Signal integrity issues on JTAG pins.	Check that the target device is not M7 enabled. Make sure programming file generated is for non M7 enabled device. Measure JTAG voltages, noise, and reflection.
-32	Failed to verify IDCODE.	File is for M1, but target device is not an M1 device.	Check that the target device is not M1 enabled.

Exit Code	Exit Message	Possible Cause	Possible Solution
	The target is not an M1 device	Signal integrity issues on JTAG pins.	Make sure programming file generated is for non M1 enabled device. Measure JTAG voltages, noise and reflection.
-33	FPGA Array encryption is enforced. A programming file with encrypted FPGA array data needs to be provided.	File contains unencrypted array data, but device contains AES key.	Provide a programming file with an encrypted FPGA Array data.
-34	FlashROM encryption is enforced. A programming file with encrypted FlashROM data needs to be provided.	File contains unencrypted FlashROM data, but the device contains an AES key.	Provide a programming file with an encrypted FlashROM data.
-35	Failed to match pass key.	Pass key in file does not match pass key in device.	Provide a programming file with a pass key that matches the pass key programmed into the device.
-36	FlashROM Encryption is not enforced. Cannot guarantee valid AES key	File contains encrypted FlashROM, but device encryption is not enforced for FlashROM	Regenerate security programming file with proper AES key. Program device security.

Exit Code	Exit Message	Possible Cause	Possible Solution
	present in target device. Unable to proceed with Encrypted FlashROM programming.		Retry programming FlashROM with encrypted programming file.
-37	FPGA Array Encryption is not enforced. Cannot guarantee valid AES key present in target device. Unable to proceed with Encrypted FPGA Array verification.	File contains encrypted FPGA Array, but the device encryption is not enforced for FPGA Array.	Regenerate security programming file with proper AES key. Program device security. Retry programming FPGA Array with encrypted programming file.
-38	Failed to program pass key.	Unstable VPUMP voltage level. Signal integrity issues on JTAG pins. Bad device.	Monitor VPUMP voltage during programming. Measure JTAG pins and noise or reflection.
-39	Failed to verify Embedded Flash Memory Block (EFMB).	Device is programmed with a different design. Signal integrity issues on JTAG pins. The EFMB data was modified through user FPGA design after programming; this could occur during standalone verify. The target EFMB block is locked with	Verify the device is programmed with the correct data/design. Monitor VPUMP voltage during programming. Measure JTAG pins and noise or reflection.

Exit Code	Exit Message	Possible Cause	Possible Solution
		FlashLock when running ACTION PROGRAM_NVM_ACTIVE_ARRAY or VERIFY_NVM_ACTIVE_ARRAY.	Run DEVICE_INFO to confirm if the target EFMB block is locked with FlashLock (pass key). If the target EFMB block is locked, then you must unlock it by erasing the security and then reprogramming with the desired security settings. After unlocking the target EFMB block attempt to rerun the target ACTION.
-40	Embedded Flash Memory Block MAC Failure.	Data in the file is encrypted with a different AES key than the device.	Verify the programming file is generated from the latest version of Designer/FlashPro.
-41	Error programming Embedded Flash Memory Block. (EFMB)	Signal integrity issues on JTAG pins.	Measure JTAG pins and noise or reflection.
-42	Failed to verify security settings.	File security settings do not match device.	Provide a programming file with security setting that match the security settings programmed into the device.
-43	Failed to verify design information.	File checksum and design name do not match the device.	Verify the device is programmed with the correct data and design.
-44	Failed to verify AES key.	The AES key in the file does not match the AES key in the device. File does not contain an AES key and	Provide a programming file with an AES key that matches the AES key

Exit Code	Exit Message	Possible Cause	Possible Solution
		the device is secured with an AES key.	programmed into the device.
-45	Device package does not match the programming file.		
-46	Embedded Flash Memory Block X Read is protected by pass key. A valid pass key needs to be provided.	File contains no pass key or incorrect pass key but EFMB read is secured with a pass key.	Provide a programming file with the correct pass key.
-47	Embedded Flash Memory Block, block X encryption is enforced. A programming file with encrypted EFMB data needs to be provided.	The programming EFMB data is not encrypted, but the device contains an AES key with encryption enforced.	Provide a programming file with encrypted EFMB data.
-48	Embedded Flash Memory Block (EFMB) block X Write is protected by pass key. A valid pass key needs to be provided.	File contains no pass key or incorrect pass key, but device is secured with a pass key.	Provide a programming file with a passkey that matches the passkey programmed into the device.
-49	Embedded Flash Memory Block (EFMB) block X Encryption is not	File contains encrypted EFMB for block X, but the device encryption is not enforced for EFMB block X.	Regenerate security programming file with proper AES key.

Exit Code	Exit Message	Possible Cause	Possible Solution
	enforced. Cannot guarantee valid AES key present in target device. Unable to proceed with Encrypted EFMB programming.		Program device security. Retry programming EFMB block X with encrypted programming file.
-51	Failed to access Embedded Flash Memory. (AFS600 only)	This version of the silicon does not support programming of the Embedded Flash Memory Block while the FPGA Array is active.	If programming the EFMB while the FPGA is active is not required, then use actions PROGRAM_NVM or VERIFY_NVM. Otherwise, use latest revision of silicon.
-52	Failed to access Embedded Flash Memory. (AFS1500 only)	This version of the silicon does not support programming of the Embedded Flash Memory Block while the FPGA Array is active.	If programming the EFMB while the FPGA is active is not required, then use actions PROGRAM_NVM or VERIFY_NVM. Otherwise, use latest revision of silicon.
-53	Failed to access Embedded Flash Memory. (AFS1500 only)	This version of the silicon does not support programming block 3 of the EFMBs while the FPGA Array is active.	If programming the EFMB while the FPGA is active is not required, then use actions PROGRAM_NVM or VERIFY_NVM. Otherwise, use EFMB

Exit Code	Exit Message	Possible Cause	Possible Solution
			blocks 0, 1, or 2, but do not use block 3.
-54	Failed to access Embedded Flash Memory.	FPGA Array is accessing the target EFMB block while attempting programming. NVM reset signal is stuck in design.	If programming the EFMB while the FPGA is active is not required, then use actions PROGRAM_NVM or VERIFY_NVM. Otherwise, check the FPGA design or use a different EFMB block that is not being accessed. Check if target EFMB block logic is tied to reset. Verify that the NVM reset signal in the design is not stuck.

ProASIC^{PLUS} and ProASIC Exit Codes

The table below lists the exit codes for ProASIC^{PLUS} and ProASIC family devices.

Table 24 · ProASIC^{PLUS} and ProASIC Family Devices Exit Codes

Exit Code	Exit Message	Possible Cause	Possible Solution
0	This message means passed. This does not indicate an error.		
1	A physical chain does not match the expected set up from the STAPL file. Also known as	Physical chain configuration has been altered. Something has become disconnected in the chain. The specific IR length of	

Exit Code	Exit Message	Possible Cause	Possible Solution
	Checking Chain Error.	non-Actel devices may be incorrect. The order of the specified chain may be incorrect.	
2	There is a reading device ID failure.	The device either does not have a valid device ID or the data cannot be read correctly.	Check the device ID.
3	This occurs when using ProASIC^{PLUS} devices.	Connect was set up for a ProASIC device and the device is actually ProASIC^{PLUS}.	Set up for a ProASIC^{PLUS} device.
5	Programming set up problem. Also known as Entering ISP Failure.	The A500K device senses the VDDL power supply as being on.	Power the VDDL down during programming. Check the device has the correct voltages on VDDP, VDDL, VPP, and VPN.
6	The IDCODE of the target device does not match the expected value in the STAPL file. This is a JEDEC standard message.	The device targeted in the STAPL file does not match the device being programmed. User selected wrong device. Device TRST pin is grounded. Noise or reflections on one or more of the JTAG pins caused by the IR Bits reading it back incorrectly.	Choose the correct STAPL file and select the correct device. Measure JTAG pins and noise or reflection. TRST should be floating or tied high. Cut down the extra length of ground connection.
7	Unknown	This occurs with current	Re-generate STAPL

Exit Code	Exit Message	Possible Cause	Possible Solution
	algorithm: alg=x, prev=x Invalid data read from device	STAPL files when the revision written into the factory row is not rev 1 for ProASICPLUS or rev 2 for ProASIC devices. The STAPL files from last year may "exit 7" with newer devices or the older revision may cause this failure if the STAPL file used is from latest version. It can occur if you are using Engineering Sample parts that are no longer supported, such as ProASIC Engineering Sample parts. This error can also occur if the programmer has trouble reading the factory row due to signal noise, crosstalk, or reflections on the JTAG signal and clock lines. It can occur if you program an -F ProASICPLUS device with an old STAPL file. This error occurs if you connected VPP and VPN the wrong way. It occurs if there are no bypass Caps on VPP VPN, which damaged the device. This error may occur if your power supply cannot source	file from Designer 6.1 SP1. Replace A500K ES parts with commercial parts. Double check VPP and VPN connections. Make sure VPP and VPN have correct bypass caps. Make sure that your power supply can deliver the correct current during programming.

Exit Code	Exit Message	Possible Cause	Possible Solution
		the correct current for programming.	
8	FPGA failed during the erase operation.	The device is secured, and the corresponding STAPL file is not loaded. The device has been permanently secured and cannot be unlocked.	Load the correct STAPL file.
11	FPGA failed verify	The device is secured and the corresponding STAPL file is not loaded. You used the Libero IDE software v2.3 or earlier or the Designer R1-2003 software or earlier to generate the STAPL file. VPN caps were soldered in the wrong polarity.	Load the correct STAPL file. Use later software versions —at least Libero v2.3 SP1 and Designer R1-2003 SP1. Double-check the VPN bypass caps polarity.
12	Security is enabled.	The device is secured and the wrong key/STAPL file was entered. The device is damaged. The verification was interrupted and therefore fails, causing the software to think the device is secure.	
14	Program security failure.		
15	This is a factory Calibration Data CRC error.	During program, erase, or verify, you must read back Calibration Data from the FPGA. The data contains a CRC. You use the CRC to	

Exit Code	Exit Message	Possible Cause	Possible Solution
		ensure the data is not corrupted/wrong. Device is damaged. Noise on the FTAL signals causes the programmer to read back wrong data.	
17	The device has been secured. Write-security is enabled.	The device is secured and the wrong key or STAPL file was entered. The device is damaged.	Load the correct STAPL file.
-80	Error code results from STAPL files for A500K devices.	An internal calibration (based on DDP and VPP) failed.	Check voltages on the device pins. Check voltages on the VDDP and VPP pins.
-90	Unexpected RCK detected.	Noise on the RCK signal. You connected a CLK source to the RCK signal. The polarized bypass capacitors on VPP or VPN are reversed-biased and are affecting the programmer's VPP or VPN output voltage. This causes programming to fail. Several FlashPros are programming at the same time and are too close to each other. Programmer not properly installed by Admin.	Disconnect the RCK and make sure TCK has a clean signal. Separate FlashPros away from each other while they are programming Internal ISP. Connect programmer as an Admin in FlashPro.
-91	Calibration data parity error.	Device is damaged.	Replace the device.

Exit Code	Exit Message	Possible Cause	Possible Solution
	Null	Several FlashPros are programming at the same time and are too close to each other. FlashPro connects to PC parallel port through a dongle key. Data length mismatch when performing DRSCAN on STAPL file.	
	Cable to target is not connected properly.	When the Analyze command is executed, the FlashPro looks for target devices. If the cable connection is wrong, FlashPro assumes that nothing is connected at all.	Confirm the connection between the header to the device. If the board supplies the power to the device, make sure the voltage level is correct.
	Chain integrity test failed: xx	The connection between the FlashPro programmer and the device is broken. The programmer cable might not be securely inserted into the header. The header is not connected to the JTAG pins of the FPGA correctly. The configuration setting (ProASIC/ProASICPLUS) does not match the target device. Noise or reflections on the JTAG pins has caused	Secure the connections. Check the JTAG pins for signal activity. Check for broken TDO, TMS, and TCK pins. After checking all type of connections if the failure exists, you may need to replace the first device (the devices closest to the TDO

Exit Code	Exit Message	Possible Cause	Possible Solution
		communication between the programmer and the device to fail. A dongle is plugged in between the PC parallel port and the FlashPro parallel port cable.	of the programming header) in the chain. Remove the dongle.
	Could not connect to programmer on port lp1 or parallel port device does not support IEEE-1284 negotiation protocol	The remote device does not respond to the negotiation protocol, for a variety of reasons.	Make sure the port is connected. Make sure the connected device is a FlashPro/Lite programmer. Turn the programmer on. Check parallel port setting in BIOS. Make sure that there are no dongles in between the parallel port and the FlashPro connection. Try another parallel cable, the parallel cable might be defective. Check to see if the programmer is damaged. Make sure the FlashPro Lite has power. The FlashPro Lite is powered from the target board through

Exit Code	Exit Message	Possible Cause	Possible Solution
			the Vdd pin of the programming header. Make sure the Vdd pin is connected and the target board is powered up. Secure the connection between the cable connector and the programming header. Before you program any devices, you should run the self-diagnostic test (see "Self-test" on page 16). The diagnostic software can be found on the Actel web site. If the test fails, please contact Actel Customer Technical Support at tech@actel.com for credit and replacement. Note: The Self-test is only available for FlashPro, not FlashPro Lite.
	External voltage detected on <Supply>	The voltage supply for the FPGA is driven by another source (board, external power-supply), but the user	Set appropriate options in the Connect menu.

Exit Code	Exit Message	Possible Cause	Possible Solution
		forgot to turn off the supply in the Connect menu.	
	VDPP Disconnected.	There is no Vddp voltage supply to the FPGA. You accidentally turned off the Vddp supply in the Connect menu. The Vddp supply on the board is not functioning.	Check the Vddp supply on the board for appropriate voltages and correct the Connect menu.
	More than one unidentified device.		
	If you want to perform an operation on the ProASIC device, the rest of the devices in the chain must be in bypass mode. To put devices in bypass mode, select Configuration > Chain Parameter (or click the Chain Parameter button in the Single STAPL Configuration window), then set the Pre IR, Pre DR, Post IR or Post DR.	STAPL settings of Pre IR, Pre DR, Post IR, and Post DR do not match the chain configuration. One or more of the devices in the chain is damaged and the ID CODE cannot be read back.	Make sure you have set Pre IR, Pre DR, Post IR, and Post DR to match the chain configuration. If you are still experiencing the failure, it is likely that the device's ID CODE cannot be read and you need to replace the device.
	Cannot find the programmer with ID xxx	The programmer is removed from the PC.	Delete programmer (or reconnect programmer) and select the **Refresh**

Exit Code	Exit Message	Possible Cause	Possible Solution
			Programmer button. See Connecting Programmers for more information.
	Fatal Error: Please check programmer set up.	Software cannot resolve the error encountered in the programmer.	Save the project file, restart the software, and power cycle the programmer.
	External voltage xxx mV is detected on xxx.	You have specified the programmer to drive the xxx but external xxx is detected.	**Deselect the xxx in the programmer setting.**
	Executing action xxx failed.	The STAPL runtime failed.	
	Executing action xxx with serial index/action xx failed.	The STAPL runtime failed.	
	No Vpump voltage source is detected.		Select the Vpump in the Programmer setting. Make sure the external Vpump is properly turned on.
	Vpump short detected.		Use a different programmer. If the problem persists, check the board layout.
	xxx Mhz TCK frequency in this STAPL file is not		Check FlashPro Lite version being used. Use FlashPro Lite

Exit Code	Exit Message	Possible Cause	Possible Solution
	supported by the FlashPro Lite detected. It supports only 4 MHz TCK frequency.		Rev C or modify the STAPL file to 4 MHz.
	xxx Mhz TCK frequency in this STAPL file is not supported by the FlashPro Lite RevC detected. It supports only 1, 2 or 4 Mhz TCK frequency.		Modify STAPL file to 1, 2, or 4 MHz.
	Cannot find the serial Index/Action xxx in STAPL file.	Mismatch between STAPL file and the Index/Action selection.	Make sure the STAPL file was not overwritten. Save the project with updated serial/action selection.
	Duplicated serial Index/Action xxx was removed.	Mismatch between STAPL file and the Index/Action selection.	Make sure the STAPL file was not overwritten. Save the project with updated serial/action selection.
	Using local backup copy xxx	Cannot find original copy.	Check for available space on the disk. Check that write permissions are enabled.
	FlashPro cannot rename the programmer/device	Name is already in use.	Create a new name.

Exit Code	Exit Message	Possible Cause	Possible Solution
	with an existing name.		
	FlashPro cannot rename the programmer/device with an invalid character.	Invalid character used in programmer/device name.	Do not use invalid characters.
	Automatic check for updates.		FlashPro can check the Actel website to find if an updated version of the software is available. If you would like to have FlashPro automatically check for software updates, choose Preferences from the File menu. From the Updates tab, you can choose your automatic software update settings. You can also select Software Updates from the Help menu for updates to the FlashPro software.
	FlashPro parse error.	FlashPro software failed to parse the file.	
	FlashPro does not support STAPL files for xxx.	STAPL file not allowed.	Use a STAPL file for your device that is supported by

Exit Code	Exit Message	Possible Cause	Possible Solution
			FlashPro.

Electronic Parameters

DC Characteristics for FlashPro4/3/3X

Note: The target board must provide the VCC, VCCI, VPUMP, and VJTAG during programming. However, if there is only one ProASIC3 device on the target board, the FlashPro4/3/3X can provide the VPUMP power supply via the USB port.

Table 25 · DC Characteristic for FlashPro4/3/3X

Description	Symbol	Min	Max	Unit
Input low voltage, TDO	VIL	-0.5	0.35*VJTAG	V
Input high voltage, TDO	VIH	0.65*VJTAG	3.6	V
Input current, TDO	IIL, IIH	-20	+20	mA
Input capacitance, TDO			40	pF
Output voltage, VPUMP, operating	VPP	+3.0	+3.6	V
Output current, VPUMP	IPP		250	mA
VJTAG = 1.5V				
Output low voltage, TCK, TMS, TDI, 100µA load	VOL	0.0	0.2	V
Output low voltage, TCK, TMS, TDI, 4mA load	VOL	0.0	0.30*VJTAG	V
Output high voltage, TCK, TMS, TDI, 100µA load	V	VJTAG-0.2	VJTAG	V
Output high voltage, TCK, TMS, TDI, 4mA load	VOH	0.70*VJTAG	VJTAG	V
Output current, TCK, TMS, TDI	IOL, IOH	-4	+4	mA

Description	Symbol	Min	Max	Unit
VJTAG = 2.5V				
Output low voltage, TCK, TMS, TDI, 100µA load	VOL	0.0	0.2	V
Output low voltage, TCK, TMS, TDI, 8mA load	VOL	0.0	0.6	V
Output high voltage, TCK, TMS, TDI, 100µA load	VOH	VJTAG-0.2	VJTAG	V
Output high voltage, TCK, TMS, TDI, 8mA load	VOH	1.8	VJTAG	V
Output current, TCK, TMS, TDI	IOL, OH	-8	+8	mA
VJTAG = 3.3V				
Output low voltage, TCK, TMS, TDI, 100µA load	VOL	0.0	0.2	V
Output low voltage, TCK, TMS, TDI, 8mA load	VOL	0.6	V	
Output high voltage, TCK, TMS, TDI, 100µA load	VOH	VJTAG-0.2	VJTAG	V
Output high voltage, TCK, TMS, TDI, 8mA load	VOH	2.4	VJTAG	V
Output current, TCK, TMS, TDI	IOL, IOH	-8	+8	mA

DC Characteristics for FlashPro Lite

Table 26 · DC Characteristic for FlashPro Lite

Description	Symbol	Min	Max	Unit
Input low voltage, TDO	VIL	-0.5	0.7	V
Input high voltage, TDO	VIH	1.7	5.0	V
Input current, TDO	IIL, IIH	-10	+10	uA
Input capacitance, TDO			40	pF
Input voltage, VDD, operating (see note)		+2.3	+3.5	V
Input voltage, VDD, power off		-1.0	+1.0	V
Input current, VDD	IVDD		500	mA
Output voltage, VPP, operating	VPP	+15.9	+16.5	V
Output voltage, VPN, operating	VPN	-13.8	-13.4	V
Output current, IPP	IPP	0	35	mA
Output current, IPN	IPN	0	-15	mA
Output low voltage, TCK, TMS, TDI, 100uA load	VOL	0.0	0.2	V
Output low voltage, TCK, TMS, TDI, 1mA load	VOL	0.0	0.5	V
Output low voltage, TCK, TMS, TDI, 2mA load	VOL	0.0	0.8	V
Output high voltage, TCK, TMS, TDI, 100uA load	VOH	2.1	2.5	V
Output high voltage, TCK, TMS, TDI, 1mA load	VOH	1.9	2.5	V
Output high voltage, TCK, TMS, TDI, 2mA load	VOH	1.6	2.5	V
Output current, TCK, TMS, TDI, nTRST	IOL, IOH	-2	+2	mA

Note: Up to 3.5 V can be supplied to the FlashPro Lite on the VDD pin. However, if the VDD supply for the FlashPro is also connected to the APA VDD supply, the voltage for the VDD pin cannot exceed 2.7 V.

DC Characteristics for FlashPro

Table 27 · DC Characteristic for FlashPro

Description	Symbol	Min	Max	Unit
Input low voltage, TDO	VIL	-0.5	0.30 * VDDP	V
Input high voltage, TDO	VIH	0.70 * VDDP	5.5	V
Input current, TDO	IIL, IIH	-10	+10	uA
Input voltage, VDDP, VDDL		0	5.25	V
Input voltage, VPP		0	21.0	V
Input voltage, VPN		-21.0		V
Input current, VDDP, VDDL, VPN, VP	IVCC		5.0	mA
Output voltage range, VDDP	VDDP	1.5	3.3	V
Output voltage range, VPP	VPP	15.0	18.0	V
Output voltage range, VPN	VPN	-16.0	-12.0	V
Output voltage resolution / Acccuracy			100 / ±50	mV
Output current, IDDP	IDDP	-135[1]	+135	mA
Output current, IDDL	IDDL	-135[1]	+135	mA
Output current, IPP	IPP	-2701[1]	+270	mA
Output current, IPN	IPN	-270	+270[1]	mA
Output low voltage, TCK, TMS, TDI, OUT0, nTRST	VOL	0.0	0.4	V
Output high voltage, TCK, TMS, TDI, OUT0, nTRST	VOH	0.85 * VDDP	+ 0.3 VDDP	V

Description	Symbol	Min	Max	Unit
Output current, TCK, TMS, TDI, OUT0, nTRST	IOL, IOH	-12	+12	mA

Note (1): When power supply mode is set to ABI_GROUND.

*** - If you want to power-up the device from the board power supply, clear the checkboxes for VDDL and VDDP. VPP and VPN are required during programming only and are supplied by the FlashPro programmer.**

Electronic Specifications

FlashPro4

The FlashPro4 output is supplied via a connector to which a detachable 10-pin cable is fitted. The connector on the FlashPro4 unit is a 2x5, RA 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. Use the 10 pin right-angle header, AMP P/N 103310-1 (DigiKey P/N A26285-ND) for FlashPro4 and use the 10 pin straight header, AMP P/N 103308-1 (DigiKey P/N A26267-ND) for the straight version..

The signals on the pins of the FlashPro4 10-pin connector are shown in the figure below (extracted from FlashPro4 product specification):

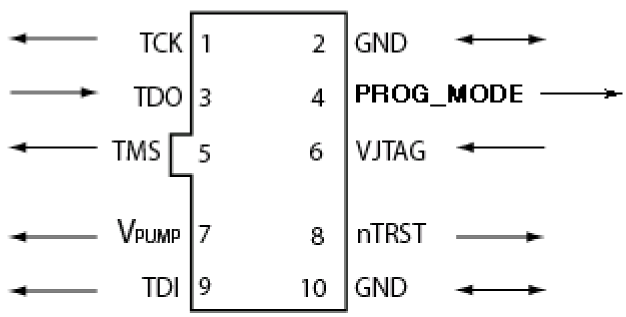


Figure 143 · FlashPro4 10-Pin Connector

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 table below shows a description of the signals.

Table 28 · FlashPro4 Signal Description

Signal	Description
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

Signal	Description
nTRST	Programmable output pin may be set to off, toggle, low, or high level
VJTAG	Reference voltage from the target board
PROG_MODE	IGLOO v2 family - Used for switching from VCC 1.2V to 1.5V during programming

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 even 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.

FlashPro3

The FlashPro3 output is supplied via a connector to which a detachable 10-pin cable is fitted. The connector on the FlashPro3 unit is a 2x5, RA 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. Use the 10 pin right-angle header, AMP P/N 103310-1 (DigiKey P/N A26285-ND) for FlashPro4/3/3X and use the 10 pin straight header, AMP P/N 103308-1 (DigiKey P/N A26267-ND) for the straight version.

The signals on the pins of the FlashPro3 10-pin connector are shown in the figure below (extracted from FlashPro3 product specification):

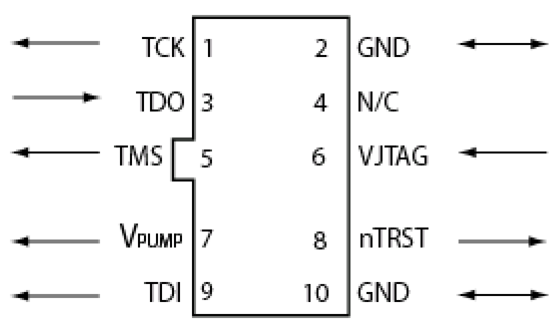


Figure 144 · FlashPro3 10-Pin Connector

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 table below shows a description of the signals.

Table 29 · FlashPro3 Signal Description

Signal	Description
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 even 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.

FlashPro Lite

For FlashPro Lite, the existing 26-pin connector is shown in the figure below.

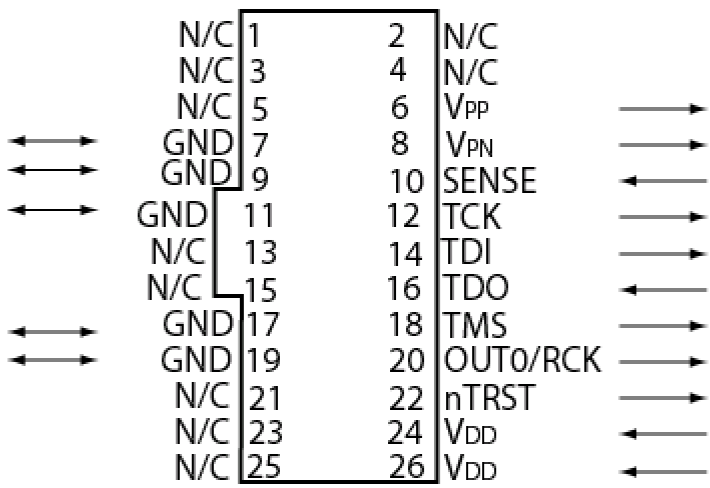


Figure 145 · 26-pin Connector for FlashPro Lite

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 SAMTEC micro connector target cable for this is:

Samtec FFSD-13-D-12.00-01-N.

The 12 inch cable is specified. This is likely to be more than enough to connect to the board and reducing the inductance will help compared with 18 inches, which is supplied by the default with FlashPro Lite.

See the table below for a description of the signals.

Table 30 · FlashPro Lite Signal Description

Signal	Description
VDDP	VDD supply for logic I/O pads
VDDL	VDD supply for core
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

Signal	Description
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 even 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.

FlashPro

For FlashPro, you can use the same 26-pin target cable you used for FlashPro Lite, but the connections are shown in the figure below.

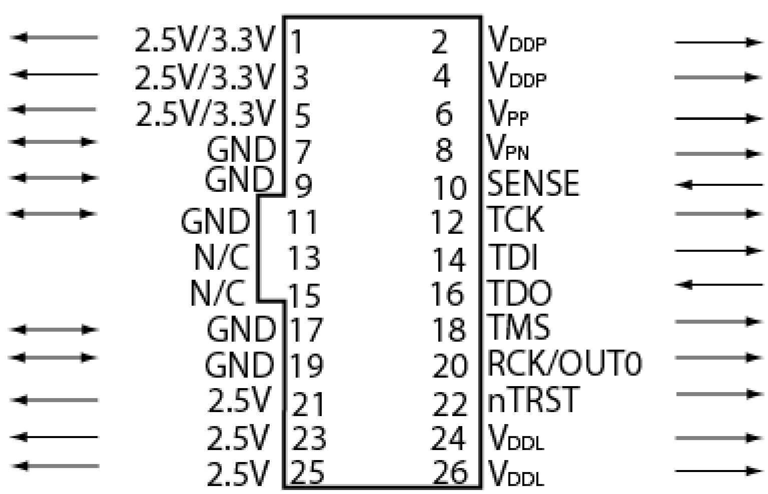


Figure 146 · 26-pin connections for FlashPro

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 table below shows the signal pin descriptions for FlashPro.

Table 31 · FlashPro Signal Description

Signal	Description
VDDP	VDD supply for logic I/O pads
VDDL	VDD supply for core
VPP	Positive programming supply (+16.5 V)
VPN	Negative programming supply (-13.8 V)
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
2.5V, 2.5V/3.3V, N/C	Programmer does not connect to these pins

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 even 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.

FlashPro4/3/3X Characteristics

Table 32 · JTAG Switching Characteristics for FlashPro4/3/3X

Description	Symbol	Min	Max	Unit
Output delay from TCK to TDI, TMS	TTCKTDI	-2	2	ns
TDO setup time before TCK rising, VJTAG=3.3	TTDOTCK	12		ns
TDO setup time before TCK rising, VJTAG=1.5	TTDOTCK	14.5		ns
TDO hold time after TCK rising	TTCKTDO	0		ns
TCK period	TTCK	41.7	10667	ns

FlashPro and FlashPro Lite Characteristics

The table below shows the JTAG switching characteristics for FlashPro and FlashPro Lite measured at the programmer end of the JTAG cable.

Table 33 · JTAG Switching Characteristics for FlashPro and FlashPro Lite

Description	Symbol	Min	Max	Unit
Output delay from TCK falling to TDI, TMS	TTCKTDI	-2	2	ns
TDO setup time before TCK rising	TTDOTCK	5.0		ns
TDO hold time after TCK rising	TTCKTDO	0		ns
TCK period	TTCK	40	10240	ns

Illustration of the JTAG Switching Characteristics

The figure below is an illustration of the JTAG switching characteristics.

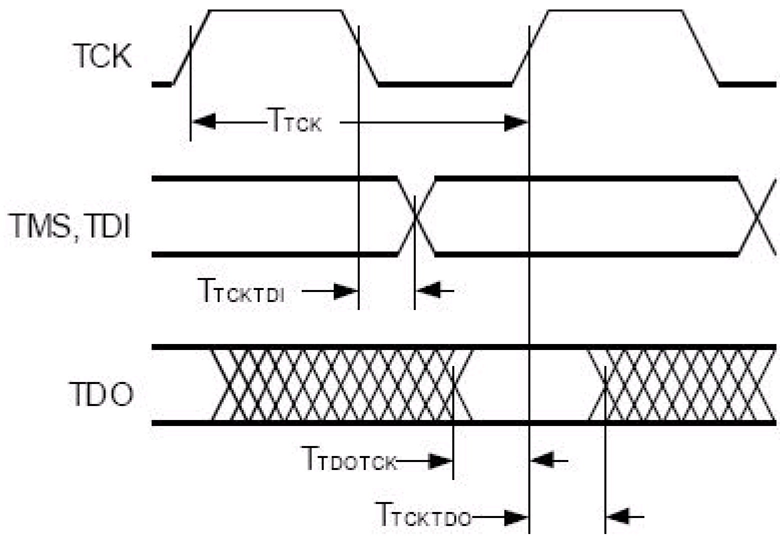


Figure 147 · JTAG Switching Characteristics

Device Debug

Device Debug enables you to use JTAG to interrogate and view embedded silicon features and device status (FlashROM, Security Settings, Embedded Flash Memory (NVM) and Analog System).

It provides tools to help troubleshoot some of the common issues related to the Embedded Flash Memory and Analog System.

Device Debug supports IGLOO, ProASIC3, SmartFusion and Fusion devices.

Note: The Device Debug for Analog Block is Fusion only.

The user support is separated into two sections:

- Using Device Debug to Find Solutions to Common Issues - Contains common issues and troubleshooting instructions that will enable you to solve your problems as quickly as possible.
- Frequently Asked Questions - Answers to the most frequently asked questions about the tools and silicon features related to your solution.

If you are unfamiliar with Device Debug, you may find it helpful to review the Getting Started with Device Debug topic. You can view descriptions of the Device Debug interface in the Reference section of the help.

Getting Started with Device Debug

This topic introduces the basic elements and features of Device Debug. If you are already familiar with the user interface you should proceed to Solutions to Common Problems or Frequently Asked Questions sections.

Device Debug enables you to use JTAG to interrogate and view embedded silicon features and device status (FlashROM, Security Settings, Embedded Flash Memory (NVM) and Analog System). Device Debug is available as a part of the FlashPro programming tool.

See the Using Device Debug topic for an overview of the use flow.

You can use the debugger to:

- Get device status and view diagnostics
- Use the FlashROM debug GUI to read out and compare content
- Use the Embedded Flash Memory Debug GUI to read out and compare your content with your original files
- Use the Analog System Debug to read out and compare your analog block configuration with your original file

Using Device Debug

The most common flow for Device Debug is:

1. Start FlashPro. If necessary, create a new project.
2. Set up your FlashPro Project with or without a PDB file. If you are in single-device mode you will need a PDB file.

   With PDB, you will get additional information such as FlashROM and Embedded Flash Memory partitions when debugging the silicon features. Actel recommends that you use a PDB with a valid-use design to start a debug session.
3. Select the target device from your chain and click **Inspect Device**.
4. Click **Device Status** to get device status and check for issues
5. Examine individual silicon features (FlashROM, Embedded Flash Memory Block and Analog System) on the device

Solutions to Common Issues Using Device Debug

Embedded Flash Memory (NVM) - Failure when Programming/Verifying

If the Embedded Flash Memory failed verification when executing the PROGRAM_NVM or PROGRAM_NVM_ACTIVE_ARRAY action, the failing page may be corrupted. To confirm and address this issue:

1. In the **Inspect Device** window click **View Flash Memory Content**.
2. Select the Flash Memory block and client (or page range) to retrieve from the device.
3. Click **Read from Device**; the retrieved data appears in the lower part of the window.
4. Click View Detailed Status to check the NVM Status.
5. If the NVM is corrupted you must reset the affected NVM pages.

   To reset the affected NVM pages, either re-program the pages with your original data or 'zero-out' the pages by using the Tcl command recover_flash_memory.

If the Embedded Flash Memory failed verification when executing a VERIFY_NVM or VERIFY_NVM_ACTIVE_ARRAY action, the failure may be due to the change of content in your design. To confirm this, repeat steps 1-3 above.

NOTE: NVM corruption is still possible when writing from user design. Check NVM status for confirmation.

Analog System Not Working as Expected

If the Analog System is not working correctly, it may be due the following:

1. System supply issue. To troubleshoot:

- Physically verify that all the supplies are properly connected to the device and they are at the proper level. Then confirm by running the Device Status.
- Physically verify that the relevant channels are correctly connected to the device.

2. Analog system is not properly configured. You can confirm this by examining the Analog System.

ADC Not Sampling the Correct Value

If the ADC is sampling all zero values then the wrong analog pin may be connected to the system, or the analog pin is disconnected. If that is not the case and the ADC is not sampling the correct value, it may be due to the following:

1. System supply issues - Run the device status to confirm.

2. Analog system is not configured at all - To confirm, read out the ACM configuration and verify if the ACM content is all zero.
3. Analog system is not configured correctly - To confirm, read out the ACM configuration and verify that the configuration is as expected .

   If you have access to your Analog System Builder settings project (<Libero IDE project>/Smartgen/AnalogBlock), you may use the compare function provided by the tool.

Frequently Asked Questions

How do I unlock the device security so I can debug?

You must provide the PDB file with a User Pass Key in order to unlock the device and continue debugging.

If you do not have a PDB with User Pass Key, you can create a PDB file in FlashPro (if you know the Pass Key value).

How do I export a report?

You can export three reports from the Device Debug GUI: Device Status, Client Detailed Status from the NVM, or the Compare Client Content report from the NVM. Each of those reports can be saved and printed.

If using a Tcl command, you can use the `-file <filename>` option for the following commands:

`read_flash_memory`
`check_flash_memory`
`compare_memory_client`
`read_device_status`
`read_flashrom`
`read_analog_block_config`
`compare_flashrom_client`
`compare_analog_config`

For example, you can use the following command to export the content of the client 'datastore1' in NVM block 0 to the report file datastore1_content.txt:

```
read_flash_memory -block 0 -client “datastore1” -file {C:\temp\datastore1_content.txt}
```

How do I generate diagnostic reports for my target device?

A set of diagnostic reports can be generated for your target device depending on which silicon feature you are debugging. A set of Tcl commands are available to export those reports. The following is a summary of those Tcl commands based on the silicon features.

When using the –file parameter, ensure that you use a different file name for each command so you do not overwrite the report content. If you do not specify the –file option in the Tcl, the output results will be directed to the FlashPro log window.

For the overall device:

`read_device_status`
`read_id_code`

For FlashROM:

`compare_flashrom_client`
`read_flashrom`

For Embedded Flash Memory (NVM):

compare_memory_client
check_flash_memory
read_flash_memory

For Analog Block:

read_analog_block_config
compare_analog_config

To execute the Tcl command, from the **File** menu choose **Run Script**.

Where can I find files to compare my contents/settings?

FlashROM

You can compare the FlashROM content in the device with the data in the PDB file. You can find the PDB in the <Libero IDE project>/Designer/Impl directory.

Embedded Flash Memory (NVM)

You can compare the Embedded Flash Memory content in the device with the data in the PDB file. You can find the PDB in the <Libero IDE project>/Designer/Impl directory.

Analog System

You can compare the Analog System configuration in the device with the data in the Analog System folder <Libero IDE project>/Smartgen/AnalogBlock. The tool automatically identifies the necessary files in the selected folder for comparison.

What is a UFC file? What is an EFC file?

UFC is the User FlashROM Configuration file, generated by the FlashROM configurator; it contains the partition information set by the user. It also contains the user-selected data for region types with static data.

However, for AUTO_INC and READ_FROM_FILE, regions the UFC file contains only:

- Start value, end value, and step size for AUTO_INC regions, and
- File directory for READ_FROM_FILE regions

EFC is the Embedded Flash Configuration file, generated by the Flash Memory Builder in the Project Manager Catalog; it contains the partition information and data set by the user.

Both UFC and EFC information is embedded in the PDB when you generate the PDB file.

Is my FPGA fabric enabled?

When your FPGA fabric is programmed, you will see the following statement under Device State in the Device Status report:

```
FPGA Array Status: Programmed and Enabled
```

If the FPGA fabric is not programmed, the Device State shows:

```
FPGA Array Status: Not Enabled
```

Embedded Flash Memory (NVM) Frequently Asked Questions

Is my Embedded Flash Memory (NVM) programmed?

To figure out if your NVM is programmed, read out and view the NVM content or perform verification with the PDB file.

To examine the NVM content, see the FlashROM Memory Content Dialog Box.

To verify the NVM with the PDB select the VERIFY or VERIFY_NVM action in FlashPro.

How do I display Embedded Flash Memory (NVM) content in the Client partition?

You must load your PDB into your FlashPro project in order to view the Embedded Flash Memory content in the Client partition. To view NVM content in the client partition:

1. Load your PDB into your FlashPro project.
2. Click **Inspect Device**.
3. Click **View Flash Memory Content**.
4. Choose a block from the drop-down menu.
5. Select a client.
6. Click **Read from Device**. The Embedded Flash Memory content from the device appears in the Flash Memory dialog box.

See the Flash Memory Dialog Box topic for more description on viewing the NVM content.

How do I know if I have Embedded Flash Memory (NVM) corruption?

When Embedded Flash Memory is corrupted, checking Embedded Flash Memory may return with any or all of the following page status:

- ECC1/ECC2 failure
- Page write count exceeds the 10-year retention threshold
- Page write count is invalid
- Page protection is set illegally (set when it should not be)

See the How do I interpret data in the Flash Memory (NVM) Status Report? topic for details.

If your Embedded Flash Memory is corrupted, you can recover by reprogramming with original design data. Alternatively, you can 'zero-out' the pages by using the Tcl command `recover_flash_memory`.

Why does Embedded Flash Memory (NVM) corruption happen?

Embedded Flash Memory corruption occurs when Embedded Flash Memory programming is interrupted due to:

- Supply brownout; monitor power supplies for brownout conditions. For SmartFusion monitor the VCC_ENVM/VCC_ROSC voltage levels; for Fusion, monitor VCC_NVM/VCC_OSC.
- Reset signal is not properly tied off in your design. Check the Embedded Memory reset signal.

How do I recover from Embedded Flash Memory corruption?

Reprogram with original design data or 'zero-out' the pages by using the Tcl command `recover_flash_memory`.

What is a JTAG IR-Capture value?

JTAG IR-Capture value contains private and public device status values. The public status value in the value read is ISC_DONE, which indicates if the FPGA Array is programmed and enabled.

The ISC_DONE signal is implemented as part of IEEE 1532 specification.

What does the ECC1/ECC2 error mean?

ECC is the Error Correction Code embedded in each Flash Memory page.

ECC1 – One bit error and correctable.

ECC2 – Two or more errors found, and not correctable.

How can I tell if my FlashROM is programmed?

To verify that your FlashROM is programmed, read out and view the FlashROM content or perform verification with the PDB file by selecting the VERIFY or VERIFY_FROM action in FlashPro.

Can I compare serialization data?

To compare the serialization data, you can read out the FlashROM content and visually check data in the serialization region. Note that a serialization region can be an AUTO_INC or READ_FROM_FILE region.

For serialization data in the AUTO_INC region, check to make sure that the data is within the specified range for that region.

For READ_FROM_FILE region, you can search for a match in the source data file.

Can I tell what security options are programmed in my device?

To determine the programmed security settings, run the Device Status option from the Inspect Device dialog and examine the Security Section in the report.

This section lists the security status of the FlashROM, FPGA Array and Flash Memory blocks.

Is my analog system configured?

To determine if the analog block is configured, run the Device Status option from the Inspect Device dialog and examine the Analog Block Section in the report. For example, the excerpt from the Device Status report below shows that the analog block status is operational:

```
Analog Block:
  OABTR Register (HEX): 0dbe37b
  3.3V (vdd33): PASS
  1.5V (vdd15): PASS
  Bandgap: PASS
  -3.3V (vddn33): PASS
  ADC Reference: PASS
  FPGA_Good: PASS
  Status: Analog Block is operational
```

If you read out an all zero value when examining the Analog System Configuration, then it is possible that the Analog System is not configured.

You need to compare your analog system configuration with the design configuration from the Analog System Builder

How do I interpret data in the Device Status report?

The Device Status Report generated from the FlashPro Device Debug Feature contains the following sections:

- IDCode (see below)
- User Information
- Device State
- Analog Block (SmartFusion and Fusion only)
- Factory Data
- Security Settings

IDCode

The IDCode section shows the raw IDCode read from the device. For example, in the Device Status report for an AFS600 device, you will find the following statement:

```
  IDCode (HEX): 233261cf
```

The IDCode is compliant to IEEE 1149.1. The following table lists the IDCode bit assignments:

Table 34 · IDCode Bit Assignments

Bit Field (little endian)	Example Bit Value for AFS600 (HEX)	Description
Bit [31-28] (4 bits)	2	Silicon Revision
Bit [27-12] (16 bits)	3326	Device ID
Bit [11-0] (12 bits)	1cf	IEEE 1149.1 Manufacturer ID for Actel

Device Status Report: User Info

The User Information section reports the information read from the User ROW (UROW) of IGLOO, ProASIC3, SmartFusion and Fusion devices. The User Row includes user design information as well as troubleshooting information, including:

- Design name (10 characters max)
- Design check sum (16-bit CRC)
- Last programming setup used to program/erase any of the silicon features.
- FPGA Array / Fabric programming cycle count

For example:

```
User Information:
UROW data (HEX): 603a04e0a1c2860e59384af926fe389f
Programming Method: STAPL
Programmer: FlashPro3
Programmer Software: FlashPro vX.X
Design Name: ABCBASICTO
Design Check Sum: 603A
Algorithm Version: 19
Array Prog. Cycle Count: 19
```

Table 35 · Device Status Report User Info Description

Category	Field	Description
User Row Data	(Example) UROW data (HEX): 603a04e0a1c2860e59384af926fe389f	Raw data from User Row (UROW)
Programming Troubleshooting Info	(Example) Programming Method: STAPL Programmer: FlashPro3	Known programming setup used. This includes: Programming method/file, programmer and

Category	Field	Description
	Programmer Software: FlashPro v8.6 Algorithm Version: 19	software. It also includes programming Algorithm version used.
Design Info	(Example) Design Name: ABCASICTO Design Check Sum: 603A	Design name (limited to 10 characters) and check sum. Design check sum is a 16-bit CRC calculated from the fabric (FPGA Array) datastream generated for programming. If encrypted datastream is generated selected, the encrypted datastream is used for calculating the check sum.

Device Status Report: Device State

The device state section contains:.

- IR-Capture register value, and
- The FPGA status

The IR-Capture is the value captured by the IEEE1149.1 instruction register when going through the IR-Capture state of the IEEE 1149.1 state machine. It contains information reflecting some of the states of the devices that is useful for troubleshooting.

One of the bits in the value captured is the ISC_DONE value, specified by IEEE 1532 standard. When the value is '1' it means that the FPGA array/fabric is programmed and enabled. This is available for IGLOO, ProASIC3, SmartFusion and Fusion devices.

For example:

```
Device State:
IRCapture Register (HEX): 55
FPGA Array Status: Programmed and enabled
```

For a blank device:

```
Device State:
IRCapture Register (HEX): 51
FPGA Array Status: Not enabled
```

Device Status Report: Analog Block

The Analog block of the SmartFusion and Fusion devices monitors some of the key power supplies needed by the device to function. These power supply status is captured in the OABTR test register in the Analog block.

For example, if you run Device Status when the Fabric and Analog configuration is programmed and powered up successfully the report indicates:

```
Analog Block:
OABTR Register (HEX): 0dbe3bb
3.3V (vdd33): PASS
1.5V (vdd15): PASS
Bandgap: PASS
-3.3V (vddn33): PASS
ADC Reference: PASS
FPGA_Good: PASS
Status: Analog Block is operational
```

Table 36 · Device Status Report - Analog Block Description

Analog Block Status	Description
OABTR Register	RAW data captured from the device
3.3V (vdd33)	Vcc33a supply status
1.5V (vdd15)	Vccnvm supply status
Bandgap	Internal bandgap supply status
-3.3V (vddn33)	ADC reference voltage status
FPGA Good	FPGA array or Fabric status

If the Fusion device is erased, the report indicates:

```
Analog Block:
OABTR Register (HEX): 188e3ba
3.3V (vdd33): PASS
1.5V (vdd15): PASS
Bandgap: PASS
-3.3V (vddn33): FAIL
ADC Reference: FAIL
FPGA_Good: FAIL
Status: Analog Block is non-operational
Analog Block is not programmed
```

Device Status Report: Factory Data

The Factory Data section lists the Factory Serial Number (FSN).

Each of the IGLOO, ProASIC3, SmartFusion and Fusion devices has a unique 48-bit FSN.

Device Status Report: Security

The security section shows the security options for the FPGA Array, FlashROM and Flash Memory (NVM) block that you programmed into the device.

For example, using a Fusion AFS600 device:

```
Security:
Security Register (HEX): 0000000088c01b
FlashROM
Write/Erase protection: Off
Read protection: Off
Encrypted programming: Off
FPGA Array
Write/Erase protection: Off
Verify protection: Off
Encrypted programming: Off
FlashMemory Block 0
Write protection: On
Read protection: On
Encrypted programming: Off
FlashMemory Block 1
Write protection: On
Read protection: On
Encrypted programming: Off
```

Table 37 · Device Status Report - Security Description

Security Status Info	Description
Security Register (HEX)	Raw data captured from the device's security status register
Write/Erase Protection	Write protection is applicable to FlashROM, FPGA Array (Fabric)and Flash Memory (NVM) blocks. When On, the Silicon feature is write/erase protected by user passkey.
Read Protection	Read protection is applicable to FlashROM and Flash Memory (NVM) blocks. When On, the Silicon feature is read protected by user passkey.
Verify	Verify Protection is only applicable to FPGA Array (Fabric) only. When On,

Security Status Info	Description
Protection	the FPGA Array require user passkey for verification. Reading back from the FPGA Array (Fabric) is not supported. Verification is accomplished by sending in the expected data for verification.
Encrypted Programming	Encrypted Programming is supported for FlashROM, FPGA Array (Fabric) and Flash Memory (NVM) blocks. When On, the silicon feature is enable for encrypted programmed. This allows field design update with encrypted datastream so the user design is protected.

Encrypted Programming

To allow encrypted programming of the features, the target feature cannot be Write/Erase protected by user passkey.

The security settings of each silicon feature when they are enabled for encrypted programming are listed below.

FPGA Array (Fabric)

```
Write/Erase protection: Off
Verify protection: Off
Encrypted programming: On
```

Set automatically by Designer or FlashPro when you select to enable encrypted programming of the FPGA Array (Fabric). This setting allows the FPGA Array (Fabric) to be programmed and verified with an encrypted datastream.

FlashROM

```
Write/Erase protection: Off
Read protection: On
Encrypted programming: On
```

Set automatically by Designer or FlashPro when you select to enable encrypted programming of the FlashROM. This setting allows the FlashROM to be programmed and verified with an encrypted datastream.

FlashROM always allows verification. If encrypted programming is set, verification has to be performed with encrypted datastream.

Designer and FlashPro automatically set the FlashROM to be read protected by user passkey when encrypted programming is enabled. This protects the content from being read out of the JTAG port after encrypted programming.

Flash Memory (NVM) Block

```
Write/Erase protection: Off
Read protection: On
```

```
Encrypted programming: On
```

The above setting is set automatically set by Designer or FlashPro when you select to enable encrypted programming of the Flash Memory (NVM) block. This setting allows the Flash Memory (NVM) block to be programmed with an encrypted datastream.

The Flash Memory (NVM) block does not support verification with encrypted datastream.

Designer and FlashPro automatically set the Flash Memory (NVM) block to be read protected by user passkey when encrypted programming is enabled. This protects the content from being read out of the JTAG port after encrypted programming.

How do I interpret data in the Flash Memory (NVM) Status Report?

The Embedded Flash Memory (NVM) Status Report generated from the FlashPro Device Debug Feature consists of the page status of each NVM page. For example:

```
Flash Memory Content [ Page 34 to 34 ]
FlashMemory Page #34:
Status Register(HEX): 00090000
Status ECC2 check: Pass
Data ECC2 Check: Pass
Write Count: Pass (2304 writes)
Total number of pages with status ECC2 errors: 0
Total number of pages with data ECC2 errors: 0
Total number of pages with write count out of range: 0
FlashMemory Check PASSED for [ Page 34 to 34 ]
The 'check_flash_memory' command succeeded.
The Execute Script command succeeded.
```

Table 38 · Embedded Flash Memory Status Report Description

Flash Memory Status Info	Description
Status Register (HEX)	Raw page status register captured from device
Status ECC2 Check	Check for ECC2 issue in the page status
Data ECC2 Check	Check for ECC2 issue in the page status
Write Count	Check if the page-write count is within the expected range. The expected write count is greater than or equal to: 6,384 - SmartFusion devices 2,288 - Fusion devices

Device Debug User Interface

Inspect Device Dialog Box

Inspect Device is available as a part of the FlashPro programming tool. Refer to Using Device Debug for information on how to configure the FlashPro to get access to this feature.

The Inspect Device dialog box enables you to access all the Device Debug features, such as the FlashROM, Embedded Flash Memory (NVM) and Analog Block. If you have multiple devices and programmers connected, choose your target device/programmer from the dropdown menu and use the ID code to verify that you are inspecting the correct device.

View Device Status - Displays the Device Status Report. The Device Status Report is a complete summary of your device state, analog block test values, user information, factory data and security information. Use this dialog box to save or print your information for future reference.

View Analog Block Configuration - Opens the Analog Block Configuration dialog box. Enables you to view the channel configuration for your analog block and compare the channel configuration with any other analog block file.

View Flash Memory Content - Opens the Flash Memory dialog box. This dialog box enables you to view the details for each flash memory block in your device.

View FlashROM Content - Opens the FlashROM data dialog box, enables you to view a list of the physical blocks in your FlashROM and the client partitions in FlashROM configuration files.

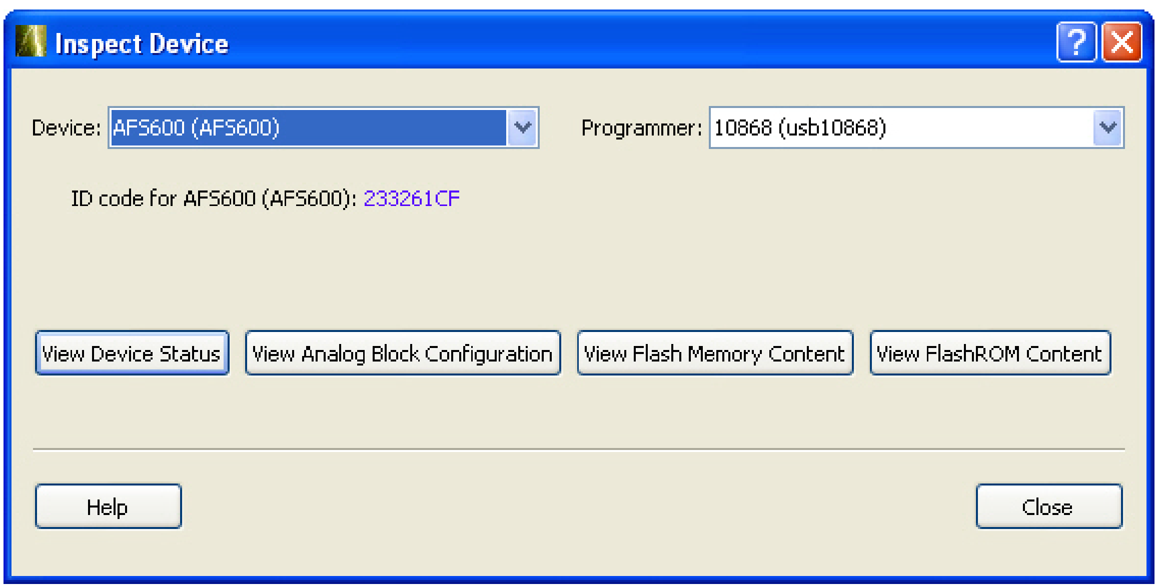


Figure 148 · Inspect Device Dialog Box

Device Status Report

This dialog box displays the Device Information report. The Device Information report is a complete summary of your device state, analog block test values, user information, factory serial number and security information. Use this dialog box to save or print your information for future reference. See the Interpreting the Device Status Report topic for information on the report contents.

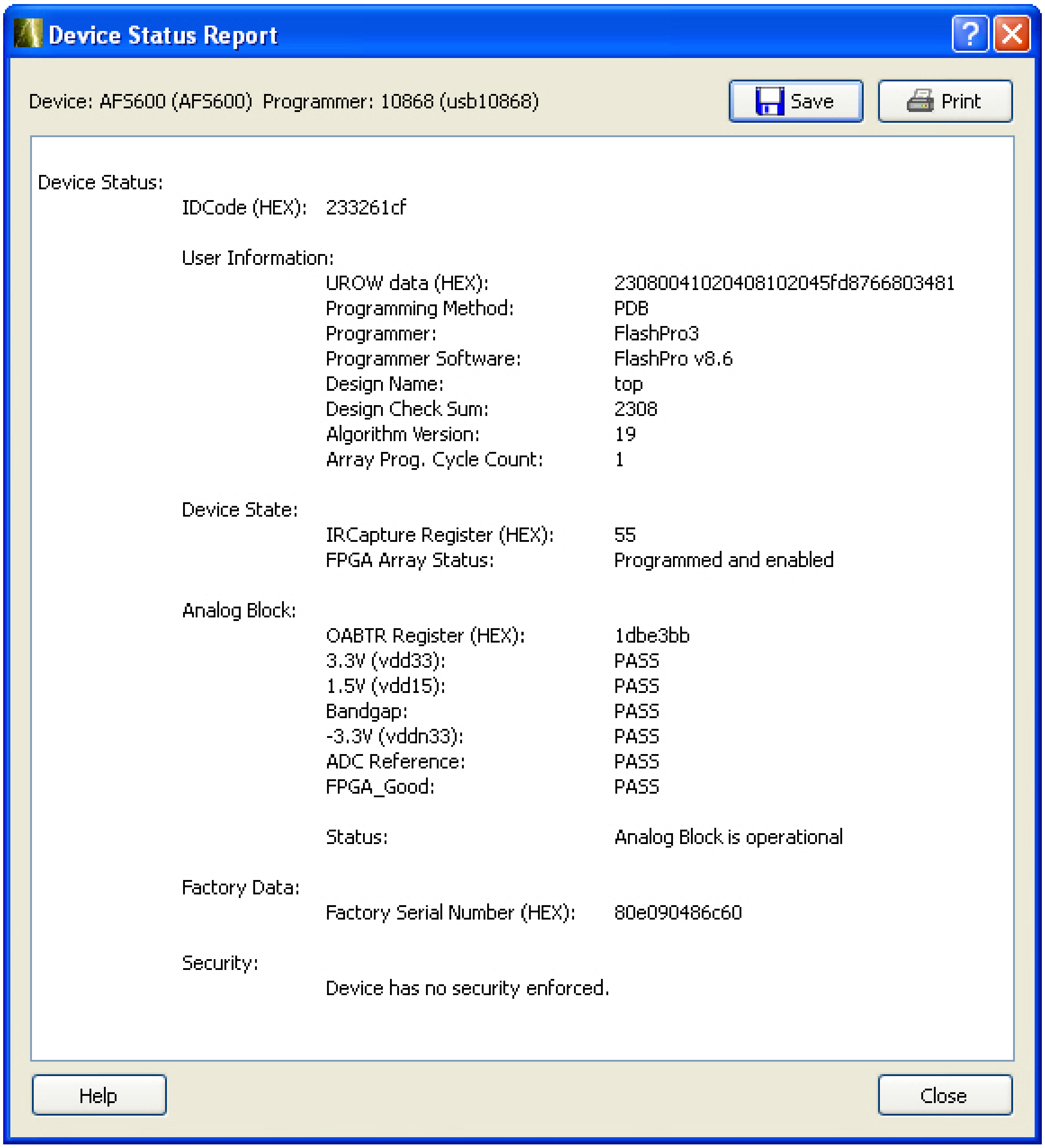


Figure 149 · Device Status Report

Analog Block Configuration Dialog Box (Fusion Only)

Enables you to:

- View the channel configuration on your analog system, identify if/how the channels are configured.
- Compare with the design configuration from the Analog System Builder

The values displayed for each channel vary depending on the channel you select; the dialog box may display the following values:

- Byte - 8-bit raw byte data read from the ACM of the respective analog. Individual, decoded bit fields of the byte are listed immediately below (as described in the Analog Quad ACM Byte Assignment table in the Fusion handbook).
- Analog MUX select
- Internal chip T monitor
- Scaling factor control
- Current monitor switch
- Current monitor drive control
- Direct analog input switch
- Pad polarity - G, T, V, C pad polarity, positive or negative
- Select low/high drive
- Prescaler op amp mode

To use the compare feature, select the **Compare with** checkbox, click **Browse** and navigate to the Analog System Builder directory. In a typical IDE project, this directory is located at <project_root>/smartgen/<analog_block_core_name>.

After specifying the compare directory the differences (if any) are indicated in red on a channel by channel basis, as shown in the figure below.

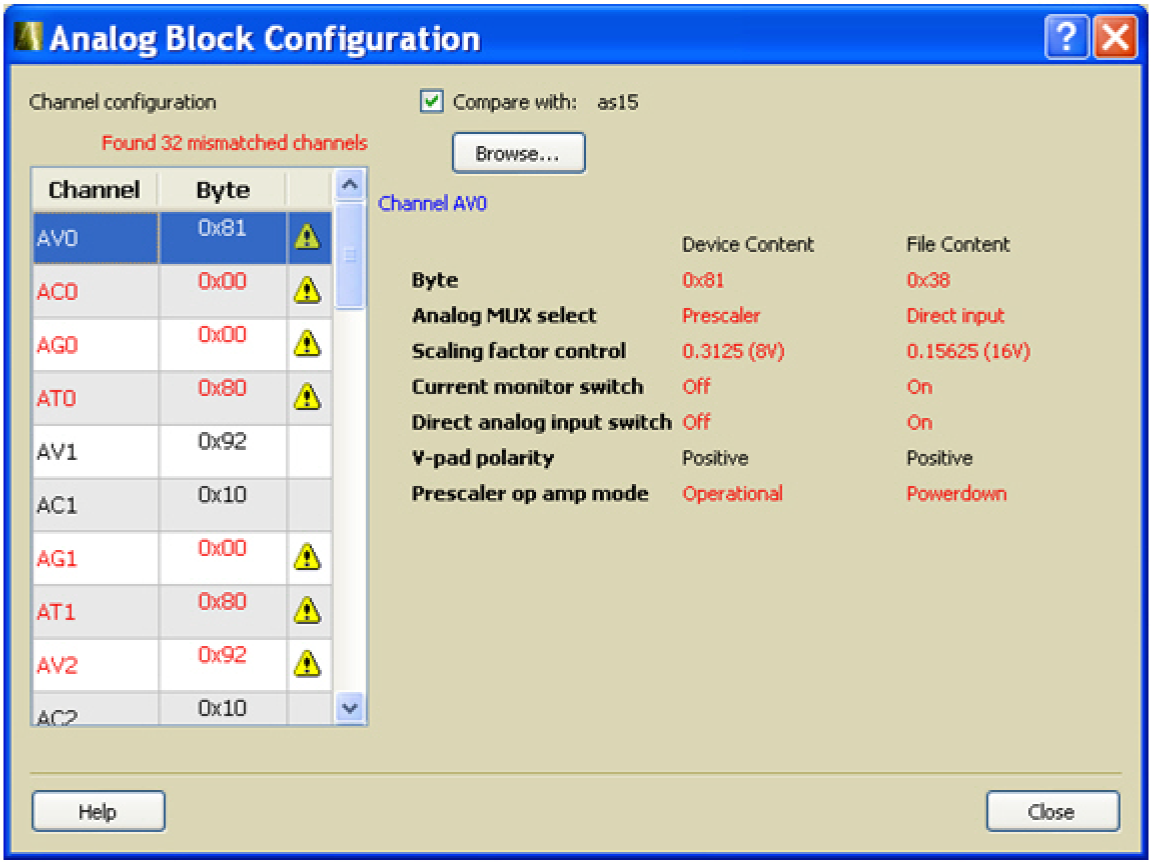


Figure 150 · Analog Block Configuration Dialog Box (Differences in Red)

Embedded Flash Memory (NVM) Content

The NVM content dialog is divided into two sections. The top section shows the data that is retrieved from the PDB that you specified. The bottom section shows the data that is retrieved from the actual device. The View Flash Memory Content diagnostic enables you to:

- View content of Flash Memory pages (as shown in the figure below)
- Compare device content with original design content (requires a PDB that contains your EFC data)
- Check page status and identify if a page is corrupted or if the write count limit has exceeded the 10-year retention threshold

Choose your block from the **From block** dropdown list This action populates the Select dropdown list with the names of the clients in the selected block that is configured in the Flash Memory System Builder.

Choose a client name from the Select dropdown list and click **Read from Device** to view the values. You can also view a specific page range in a block by selecting the <Page Range> option in the Select dropdown list and then specifying the start page and the end page.

You must click **Read from device** each time you specify a new page range to update the view.

If you do not have your original design programming database (PDB) file, then you can also examine and retrieve a range of pages. Specify a page range if you wish to examine a specific set of pages in the block.

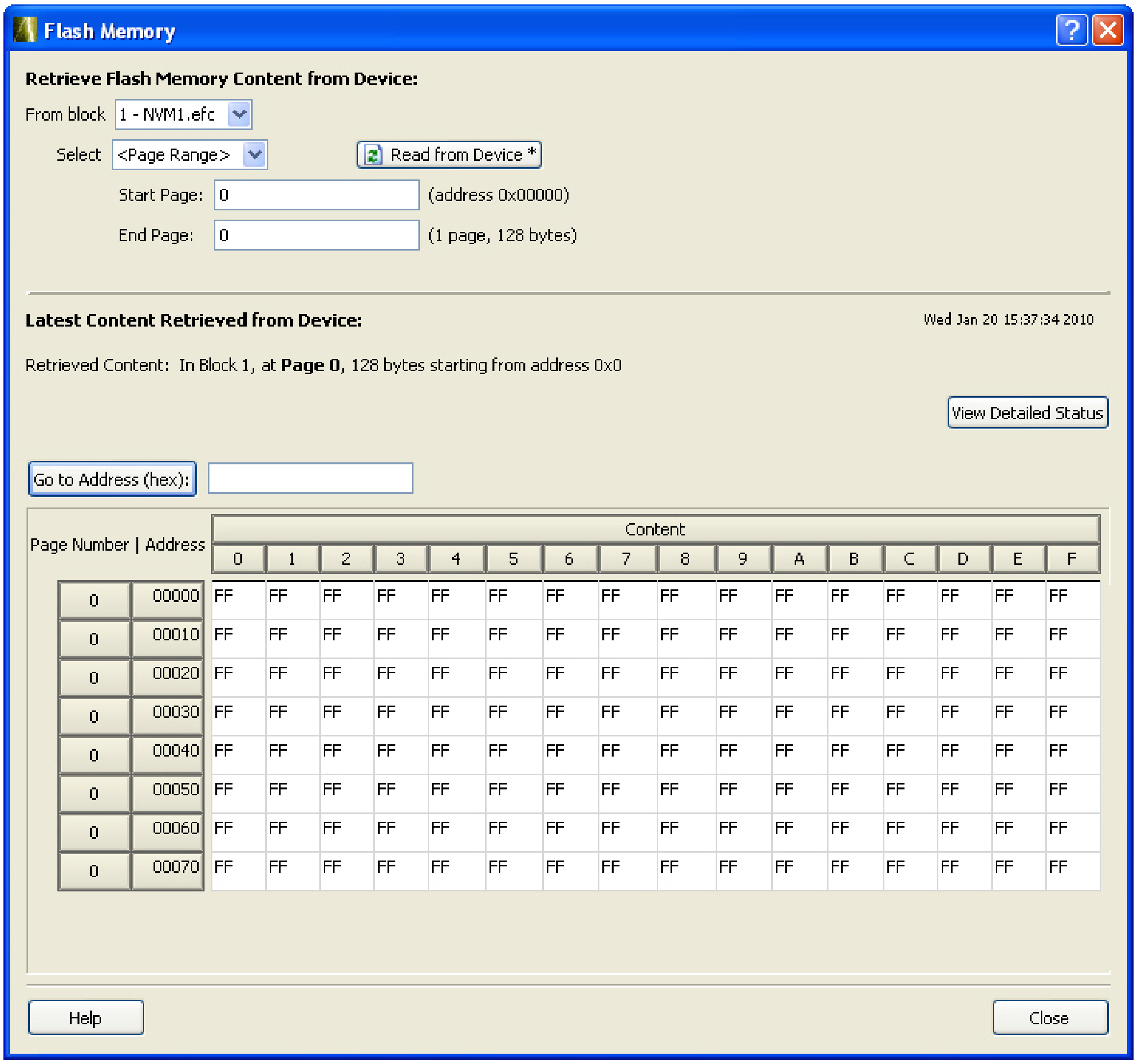


Figure 151 ·

Figure 152 · Flash Memory Dialog Box (Device Debug)

Embedded Flash Memory: Browse Retrieved Data

The retrieved data table displays the content of the selected client or the page range selection. Corrupted pages content is displayed in red. Read-only page content, corresponding to clients defined with the Prevent read option in Flash Memory System Builder, is displayed on gray background. If content cannot be read (e.g. pages are read-protected, but security has been erased), it is displayed as XX. The mouse tooltip summarizes abnormal content status (as shown in the figure below).

The corresponding page number and address (relative to the current block) are displayed in the left column. The client size specified in the Flash Memory System Builder is shown at the top of the content table.

In the Retrieved Data View you can enter an Address value (such as 0010) in the Go to Address field and click the corresponding button to go directly to that address.

Click **View Detailed Status** for a detailed report on the page range you have selected.

For example, if you want to view a report on pages 1-3, set the **Start Page** to 1, **End Page** to 3 and click **Read from Device**, then click **View Detailed Status**; the figure below is an example of the data for a specific page range.

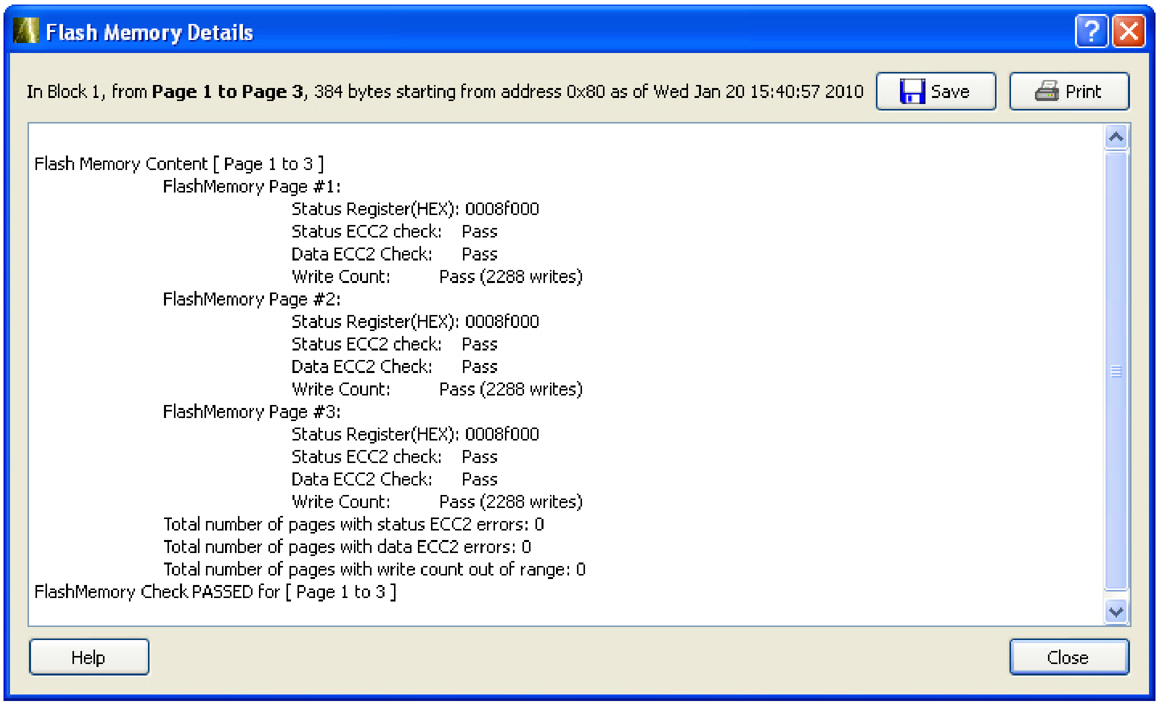


Figure 153 · Flash Memory Details Dialog Box (Device Debug)

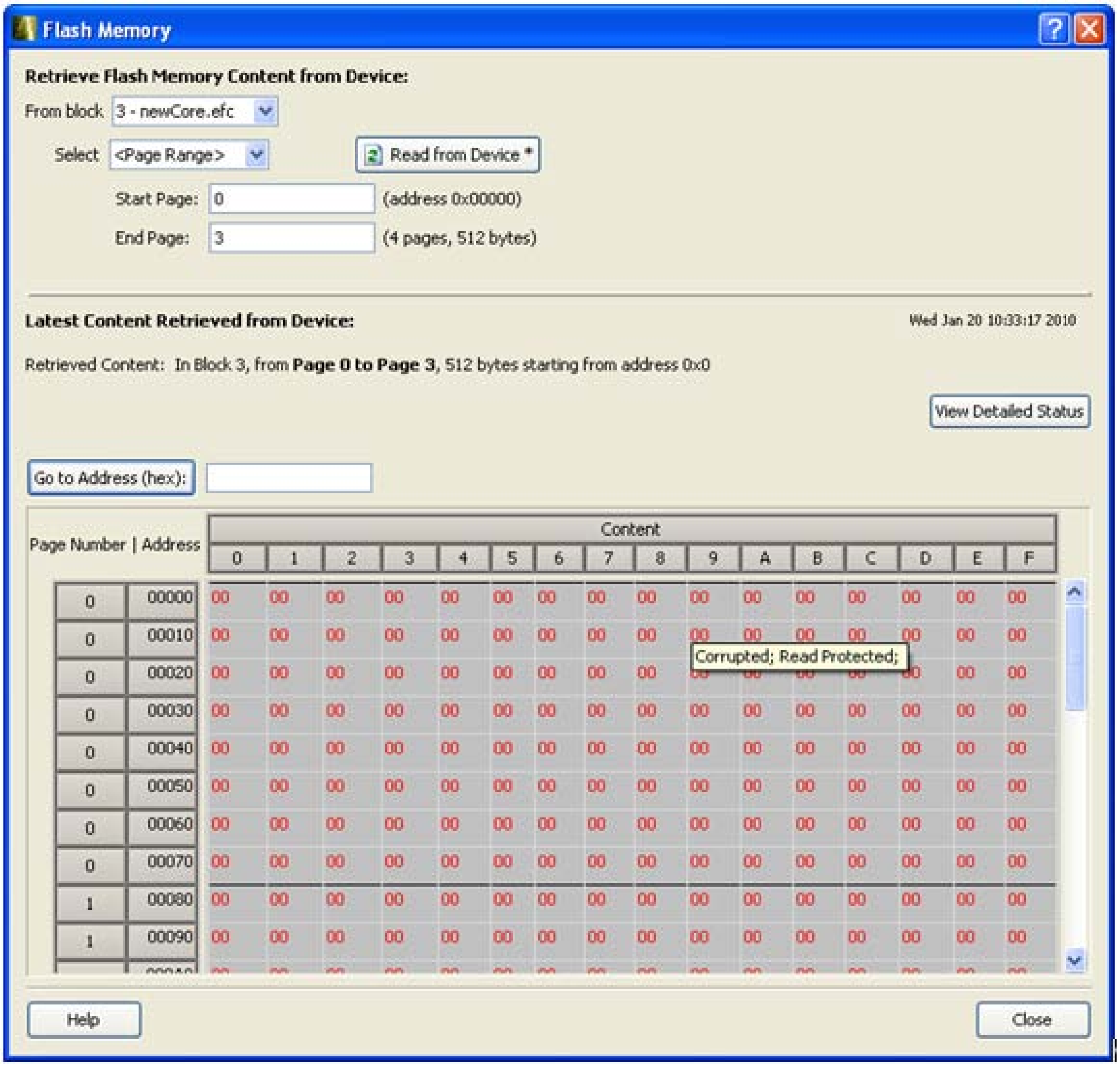


Figure 154 · Flash Memory Browse Retrieved Data

Embedded Flash Memory: Compare Memory Client

After you retrieve the data from the device, the Compare Client Content button enables you to compare the content of the selected client from the device with the original programming database (PDB) file. The differences are shown in the Compare Memory Client dialog (as shown in the figure below).

NOTE: This option is not available when you select to retrieve the data based on a page range.

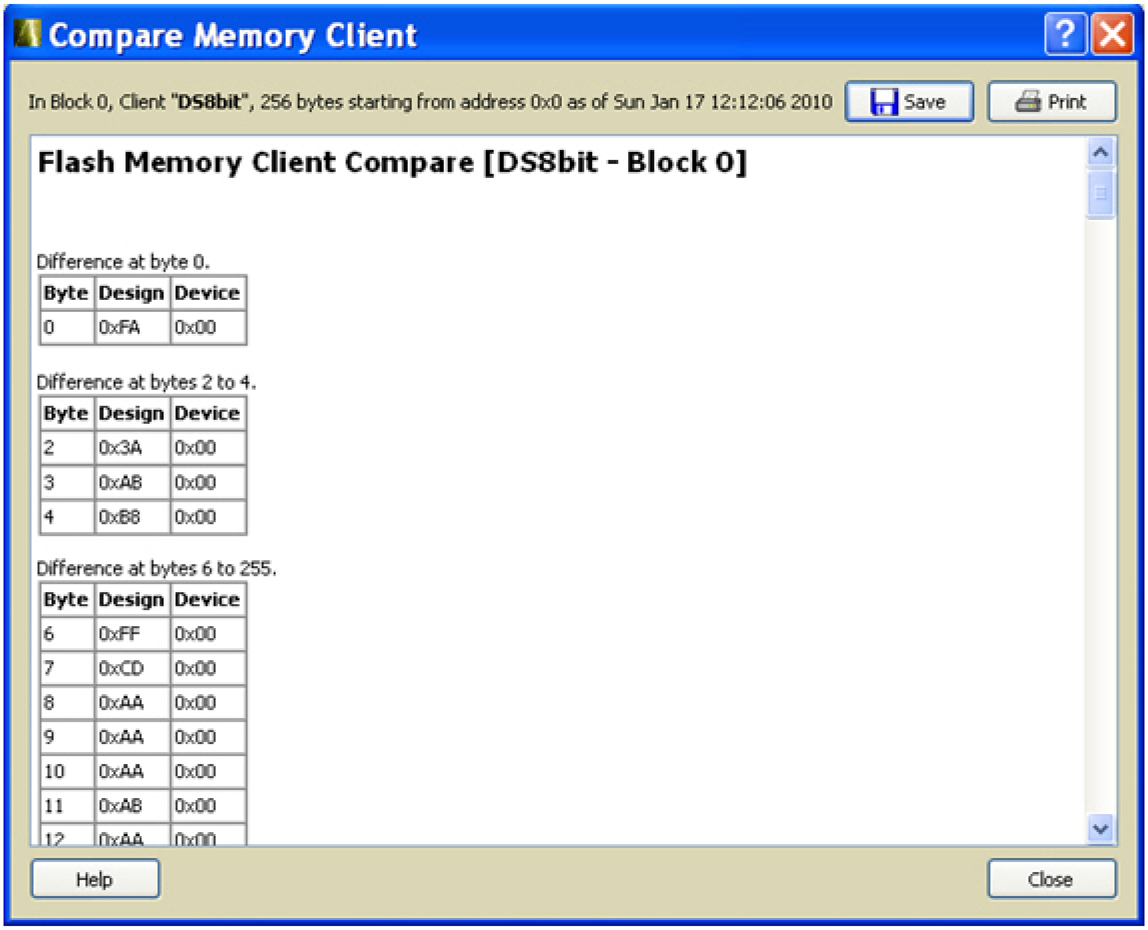


Figure 155 · Compare Memory Client Dialog Box

FlashROM Content Dialog Box

Enables you to view the physical blocks in your FlashROM and the client partitions specified in the original design content (requires a PDB that contains your UFC data). If the project's PDB does not contain UFC data, only the physical blocks are displayed.

Scroll through the table to view the Words and Pages for your physical blocks.

The Client Partitions section lists the names and configuration details of the clients set up in the FlashROM Builder. It automatically finds all mismatched client regions. To view the differences between a client and the device content, select a region row in the Client Partitions table. This action will highlight the corresponding device content in the Physical Blocks table. The mismatch details are displayed below the Client Partitions table.

To copy to clipboard the content of the Physical Blocks table, select one or more cells in the table and type Ctrl+C.

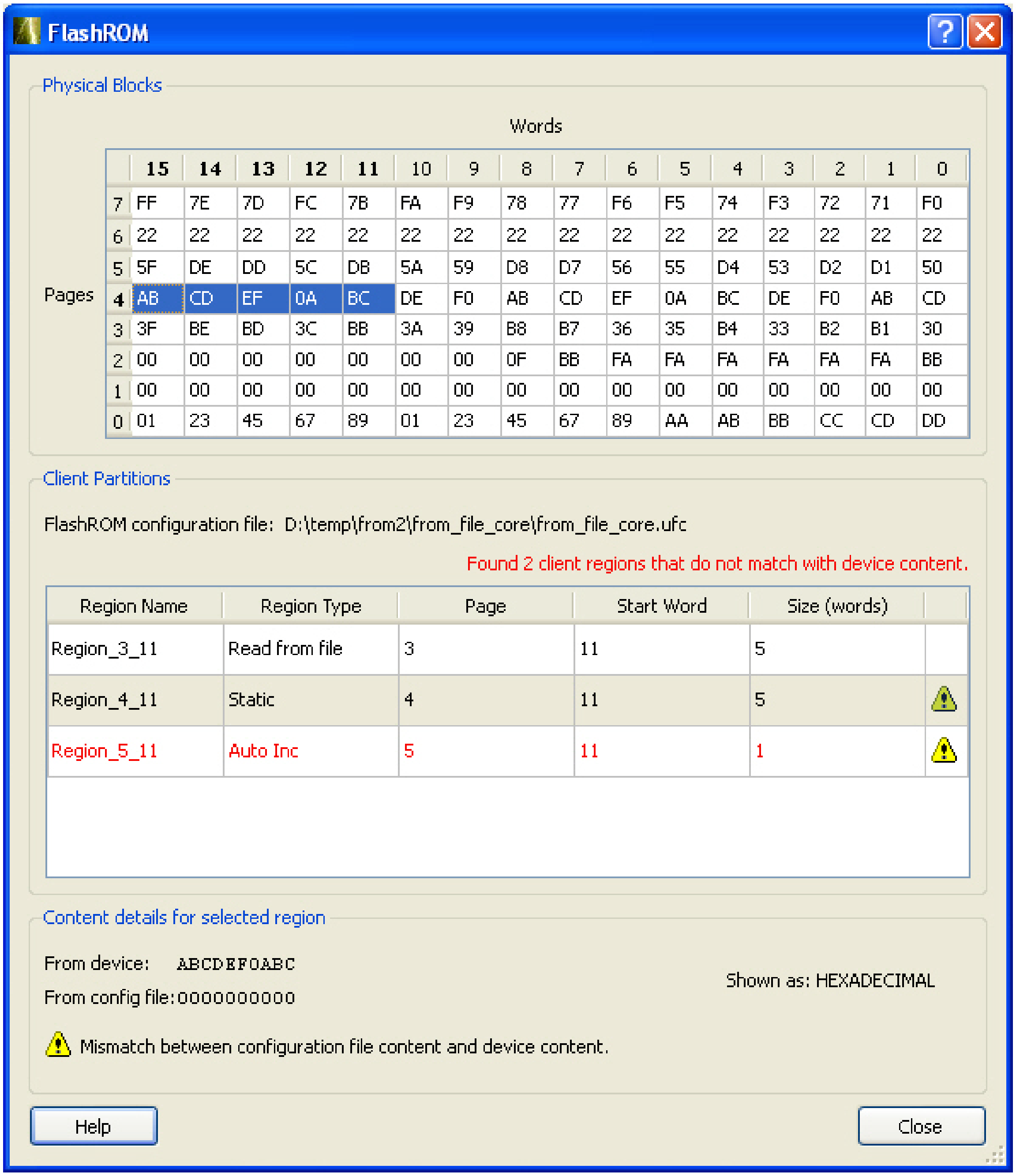


Figure 156 · FlashROM Content Dialog Box

Device Debug Tcl Commands

The following table lists the Tcl commands related to Device Debug. Click the command to view more information.

Table 39 · Device Debug Tcl Commands

Command	Action	Type
check_flash_memory	Performs diagnostics of the page status and data information	Embedded Flash Memory (NVM)
compare_analog_config	Compares the content of the analog block configurations in your design against the actual values in the device.	Analog Block
compare_flashrom_client	Compares the content of the FlashROM configurations in your design against the actual values in the selected device.	FlashROM
compare_memory_client	Compares the memory client in a specific device and block	Embedded Flash Memory (NVM)
read_analog_block_config	Reads each channel configuration on your analog system, enabling you to identify if/how each channel is configured.	Analog Block
read_device_status	Displays a summary of the selected device	
read_flashrom	Reads the content of the FlashROM from the selected device	FlashROM
read_flash_memory	Reads information from the NVM modules (page status and page data)	Embedded Flash Memory (NVM)
read_id_code	Reads IDCode from the device without masking any IDCode fields	
recover_flash_memory	Removes ECC2 errors due to memory corruption by reprogramming specified flash memory (NVM) pages and initializing all pages to zeros.	Embedded Flash Memory (NVM)
set_debug_device	Identifies the device you intend to debug.	

Command	Action	Type
set_debug_programmer	Identifies the programmer you want to use for debugging (if you have more than one).	

Customizing the Toolbar

Display the tools and commands you frequently use in the toolbar by customizing it.

To customize the toolbar:

1. From the **Customize** menu, choose **Toolbars**. The **Customize** dialog appears.
2. Click the **Toolbar** tab and check the tools you want to display by checking their respective boxes, (see figure below).

Note: Note: You can remove tools from your toolbar by deselecting tools from the Toolbar field.

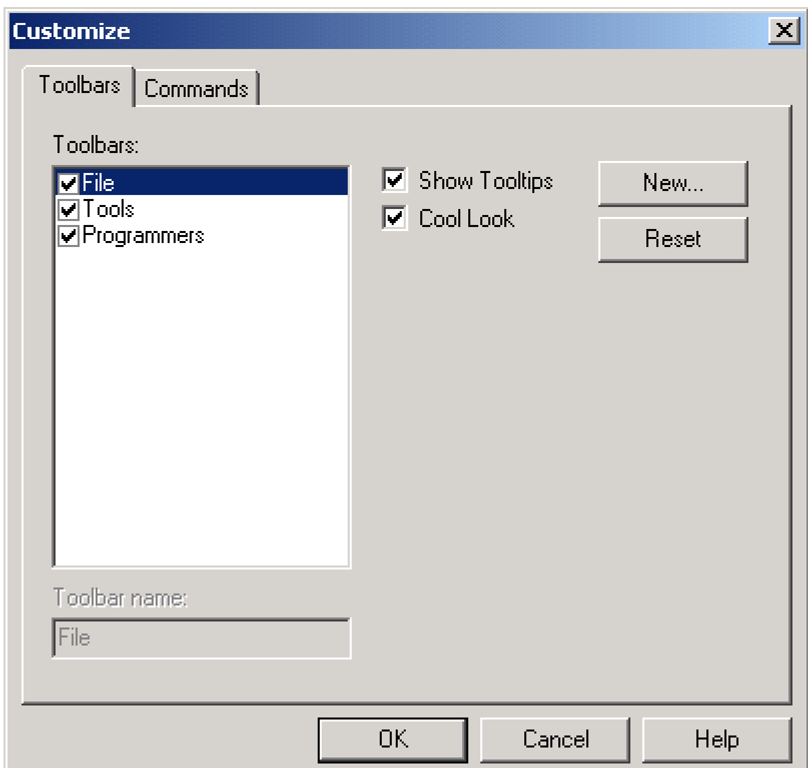


Figure 157 · Customize Dialog Box

3. Click inside the **Show Tooltips** checkbox for assistance in identifying icons on your toolbar when you scroll across them with your mouse.
4. Click inside the **Cool Look** checkbox to change the look of your toolbar.

5. Click **OK**.

You can create multiple toolbars and assign names to them. Click the **New** button and type in a name in the **New toolbar** dialog box to create a new toolbar. The name of your toolbar will display in the **Toolbar** field. Reset your toolbar to the default settings by clicking the **Reset** button.

To customize commands:

1. From the **Customize** menu, choose **Toolbars**. The **Customize** dialog appears.
2. Click the **Commands** tab.

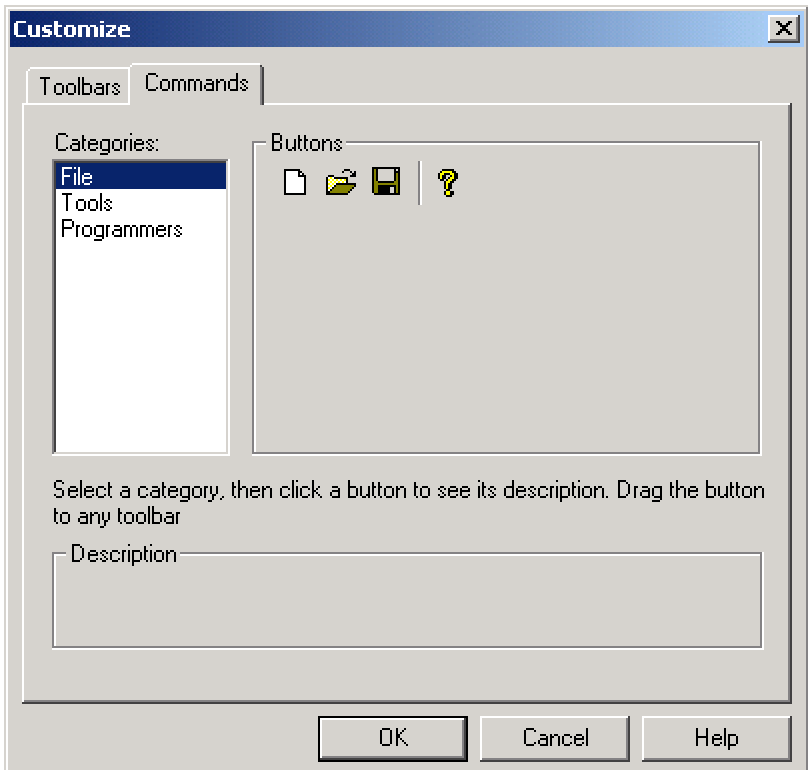


Figure 158 · Customize Dialog Box

3. Select a category by clicking one of three options (**File**, **Tools**, or **Programmers**). As you click an option, the buttons to the right of the category area change accordingly.
4. Click and drag a button to your toolbar.
5. Click **OK** after you have customized your toolbar.

You can also remove commands from your toolbar by reversing the click and drag method described in the steps above. Click and drag tools from your toolbar to the **Buttons** field in the **Customize** dialog box.

Customizing the Programming Window

The FlashPro software also enables you to customize the programmer window by right-clicking on the programmer window's header (see figure below).

Figure 159 · Programming Window Header

The following right-click menu displays (see figure below).

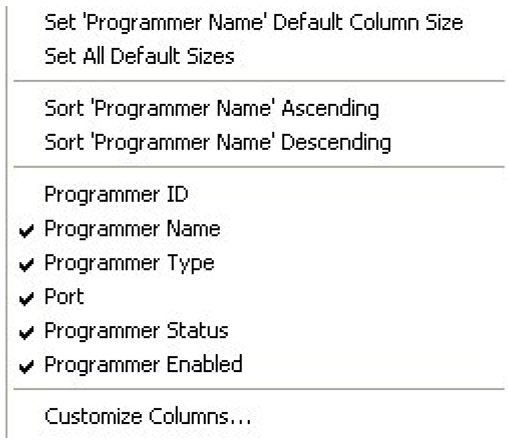


Figure 160 · Right-click Menu

The Customize right-click menu (as shown above) is divided into three sections. Click an item in the first section to set default sizes. Click an item in the second/middle section to add that item to the programmer window, and click the **Advanced** or last section to customize the columns in the **Programmer** window from the **Customize Columns** dialog box (see figure below).

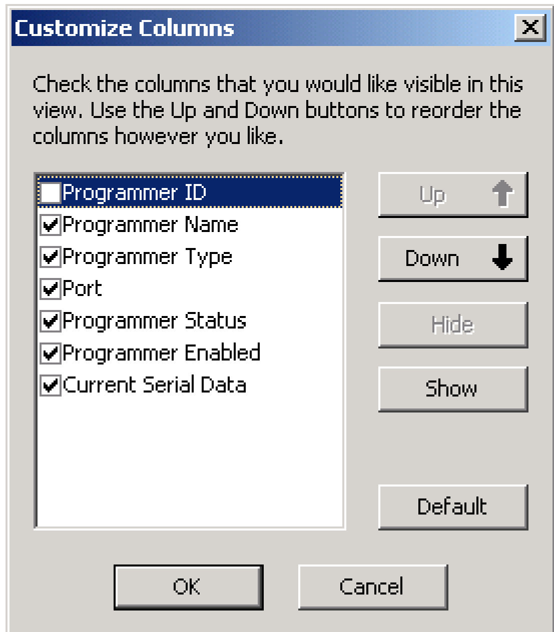


Figure 161 · Customize Columns Dialog Box

Note: Follow the instructions in the Customize Columns dialog box to customize the programming window. Use the Up and Down buttons to move through the list. Use the Show and Hide buttons to hide or show columns in the programmer window.

FlashPro Preferences

The Preferences dialog box includes three tabs: Log Window, Display Mode, and Updates (see figure below). You can access the Preferences dialog box by choosing **File** > **Preferences**.

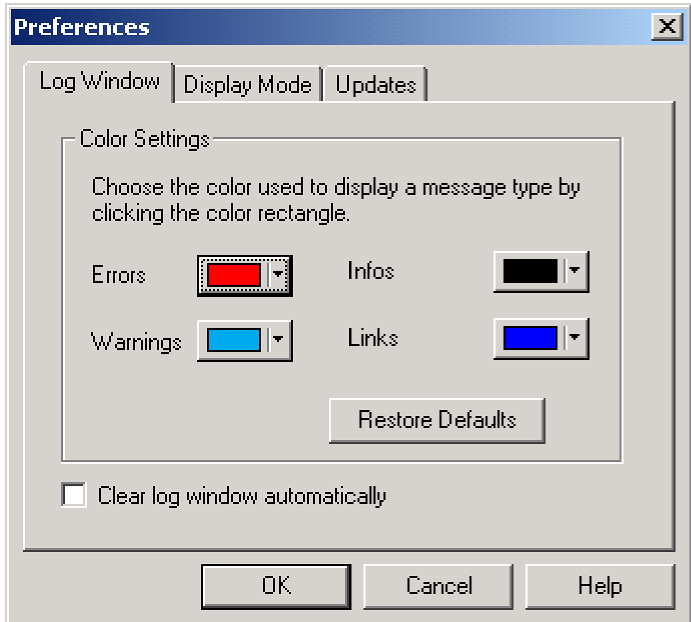


Figure 162 · Preferences Dialog Box

Log Window

The Log Window tab includes options for you to choose color settings for the various messages (Errors, Warnings, Information, Links) displayed in the Log window (see figure above).

Display Mode

The Display Mode tab describes the two display modes available in the FlashPro software (as shown in the figure below). Read each option carefully and choose the mode that will meet your programming needs. As the Preferences dialog box indicates, the Classic Mode is designed for multiple programming runs when it is not necessary for you to change your device settings. The Advanced Mode differs from the Classic Mode because displays both windows (Programmer List and Device Configuration) in the same GUI. Use this mode when you need to change device settings frequently.

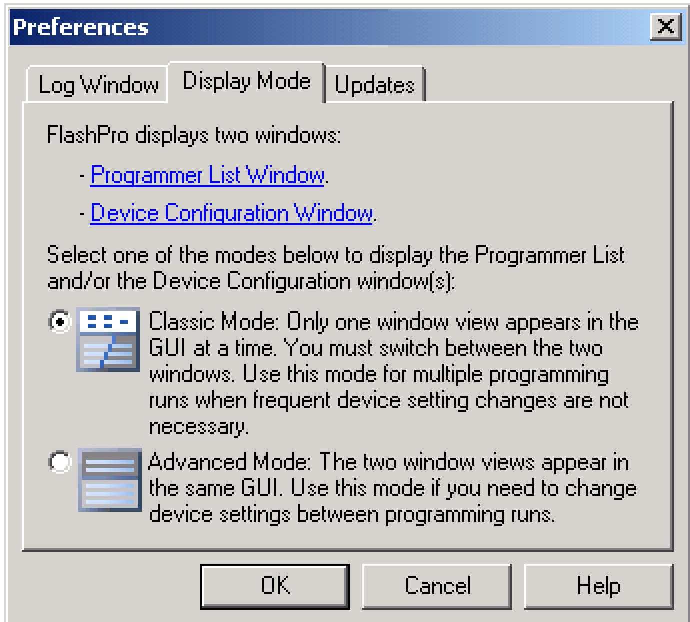


Figure 163 · Preferences Dialog Box- Display Mode

Software Updates

The Updates tab lists the FlashPro software setting options. You can choose to have the FlashPro software automatically check for updates at startup (from the Actel website) or remind you to check for updates at startup (requires you to go to the Actel website). If you want to decline both options, choose the last option: Do not check for updates or remind me at startup.

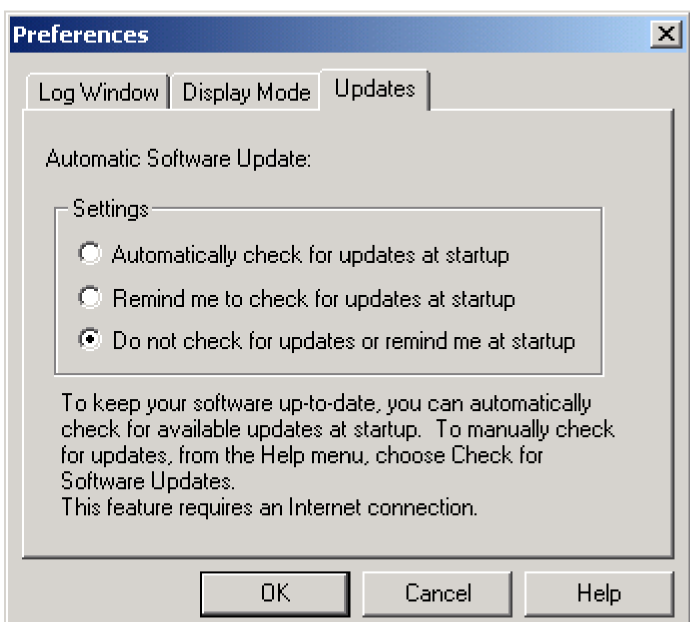


Figure 164 · Preferences Dialog Box- Updates

Software version is up to date

This informational message notifies you that there are no software updates available from Actel at this time. You can set your update preferences to automatically check for software updates.

FlashPro File Menu

In the **Chain Programming** mode, the **Edit** menu and the **Configuration** menu changes. The table notes these changes.

Command	Icon	Shortcut	Sub-menu	Function
New Project		Ctrl + N		Create a new project
Open Project		Ctrl + O		Opens the FlashPro Open Project dialog box
Restore Chainbuilder Project				Restores a ChainBuilder project in FlashPro
Close Project				Closes the current project
Save Project		Ctrl + S		Saves the current project
Save Project As				Opens the Save As dialog box; enables you to save your project in a different directory or with a different name
Import Configuration File		Ctrl + I		Opens the Import Configuration File dialog box; enables you to import configuration files for your device(s)
Set Project Log File >			Main Log File	Opens the Set Log File dialog box; sets the location of your main log file
			Serialization Log File	Opens the Set Serialization Log File dialog box; sets the location of your serialization log file
Run Script				Opens the Execute Script dialog box; enables you to run Tcl Scripts with arguments

Command	Icon	Shortcut	Sub-menu	Function
Export >			Configuration File	Opens the Export Configuration File dialog box; enables you to export configuration file(s)
			Script	Opens the Save As dialog box; enables you to export your actions as a Tcl script
			Chain STAPL File	Opens the Export Chain STAPL File dialog box; enables you to export and save your Chain STAPL file
			Chain SVF File	Opens the Export Chain SVF File dialog box; enables you to export and save your Chain SVF file
			Single Device STAPL File	Opens the Export Single Device STAPL File dialog box; enables you to export and save your single device STAPL file
			Single Device SVF File	Opens the Export Single Device SVF File dialog box; enables you to export and save your single device SVF file
			Single 1532 File	Opens the Export Single 1532 File dialog box; enables you to export and save your single 1532 file
Preferences				Opens the Preferences dialog box; enables you to set your Log window, Display Mode and Update preferences for FlashPro
Exit				Exits FlashPro

FlashPro Edit Menu

Command	Shortcut	Function
Cut Devices	Ctrl + Shift + X	Removes (cuts) devices from the project
Copy Devices	Ctrl + Shift + C	Copies the selected device(s) to your Clipboard
Paste Devices	Ctrl + Shift + V	Pastes the devices from your Clipboard into the project
Clear Log Window		Clears the Log window (deletes all Log window content)

FlashPro View Menu

The View menu shows or hides the FlashPro GUI elements.

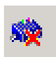
Command	Sub-menu	Function
Status Bar		Shows/hides the FlashPro Status Bar
Programmer List Window		Shows/hides the Programmer List Window
Programmer Details Window		Shows/hides the Programmer Details Window
Single Device Configuration Window		Shows/hides the Single Device Configuration Window
Log Window		Shows/hides the Log window
Single Device Configuration >	Basic View	Enables the Basic view for the Single Device Configuration window
	Advanced View	Enables the Advanced view for the Single Device Configuration window

FlashPro Tools Menu

Command	Icon	Shortcut	Sub-menu	Function
Mode >			Single-Device Programming	Sets FlashPro to Single-Device Programming mode
			Chain Programming	Sets FlashPro to Chain Programming mode
Serialization >			Skip Serial Data	Sets FlashPro to skip serial data during programming
			Reuse Serial Data	Sets FlashPro to reuse serial data during programming
Programmer Settings				Opens the Programmer Settings dialog box; enables you to set options for all FlashPro programmer types
Import Settings for Non-Actel Devices				Opens the Import Settings for Non-Actel Devices dialog box; enables you to import your settings for non-Actel devices you wish to program with FlashPro
Connect Parallel Port Cable				Opens the Connect Parallel Port Cable dialog box; enables you to connect your parallel port buffer cable
Run		Ctrl + Enter		Programs your device

FlashPro Programmers Menu

Command	Icon	Shortcut	Function
Ping			Pings a selected programmer(s)
Self Test			Runs a self-test on the selected programmer(s)

Command	Icon	Shortcut	Function
Scan Chain			Runs scan chain on the selected programmer(s)
Remove			Removes the selected programmer(s) from FlashPro
Refresh/Rescan		Ctrl + F5	Refreshes FlashPro and rescans for programmers

FlashPro Configuration Menu

Command	Icon	Shortcut	Sub-menu	Function
Select Action		Ctrl + Shift + A		
Serialization >		Ctrl + Shift + S	Use Serialization	Enables you to use Serialization in FlashPro
		Ctrl + Shift + R	Select Range	Enables you to set your Serialization range
		Ctrl + Shift + U	View Status	Enables you to view your Serialization status
Load Programming File		Ctrl + Shift + L		Opens the Load Programming File dialog box
Unload Programming File				Removes (unloads) your programming file from FlashPro
PDB Configuration		Ctrl + Shift + P		Opens the PDB Configuration dialog box; enables you to set your PDF configuration options
Select Target Device		Ctrl Shift + D		Opens the Select Target Device dialog box; enables you to set your target

Command	Icon	Shortcut	Sub-menu	Function
				device for programming
Chain Parameter		Ctrl + Shift + H		Opens the Chain Parameter dialog box; enables you to set parameters for your programming chain

FlashPro Customize Menu

Command	Function
Toolbars	Opens the Customize dialog box to the Toolbars tab; enables you to show/hide toolbars and tooltips
Commands	Opens the Customize dialog box to the Commands tab; enables you to add/remove individual commands to your toolbars

FlashPro Help Menu

Command	Icon	Sub-menu	Function
Help >		Help Topics	Opens the help
		Programmer View	Opens the help to the FlashPro Programmer List Window topic
		Details on Programmer View	Opens the help to the Programmer Details Window topic
		Single Device Programming	Opens the Single Device Programming help topic
		Chain Programming	Opens the Chain Configuration Window help topic
Actel Web Site			Opens www.actel.com in your default

Command	Icon	Sub-menu	Function
			browser
Check for Software Updates			Checks for software updates (works only if you are connected to the internet)
About FlashPro			Lists the FlashPro release information

FlashPro Flow Window

The **Flow** window (located between the toolbar and **Log** window in the FlashPro GUI)consists of the following buttons: New Project, **Open Project**, **Configure Device**, **View Programmers**, and **Run**. See the table below for a description of these features.

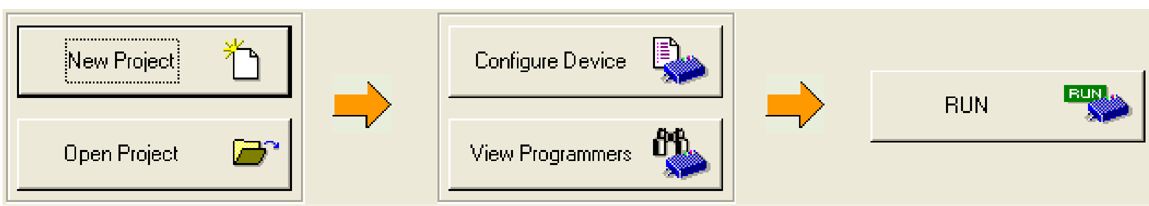


Figure 165 · Flow Window

Table 40 · Flow Window Button Description

Button	Description
New Project	Creates a new project. Opens the **New Project** dialog box.
Open Project	Opens a new project. Opens the **Open Project** dialog box.
Configure Device	Opens the Single Device Configuration window to configure your file.
View Programmers	Opens the Programmer List Window for you to view your programmers.
Refresh/Rescan for Programmers	Rescans for programmers.
Run	Executes programming.

FlashPro Log Window

The Log window displays errors, warnings, and basic information about your device. Click the tabs at bottom of the Log window to toggle between messages or click the **All** tab to display all of the messages.

You can access the Log window from the View menu.

Setting Log window preferences

From the **Preferences** dialog box, you can change the text color of the messages that appear in the **Log** window.

To set Log window preferences:

1. From the **File** menu, choose **Preferences**.
2. Follow the directions in the **Color Settings** area or click the **Restore Defaults** button.

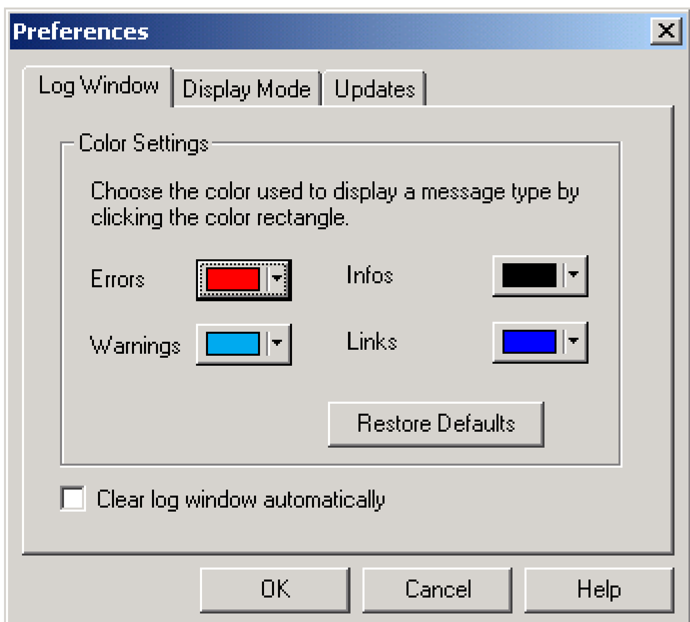


Figure 166 · Preferences

3. Click **OK** to apply settings.

Note: You can clear the Log window by choosing Clear Log window from the Edit menu or you can automatically erase the information by checking the Clear Log Window Automatically checkbox.

FlashPro Status Bar

The **Status** bar displays the program status, the programmer information (the file name for single device programming and number of devices for chain programming), and the programming mode (single or chain).

FlashPro Programmer List Window

To activate the **Programmer List** Window, select **View** > **Programmer List Window**. The FlashPro **Programmer List** Window consists of a spreadsheet with the programmer name, programmer type, port number, programmer status, programmer enable check box, and a **Refresh/Rescan for New Programmers** button as shown in the figure below.

Note: Double-clicking any of the spreadsheet columns opens the Programmer Details window.

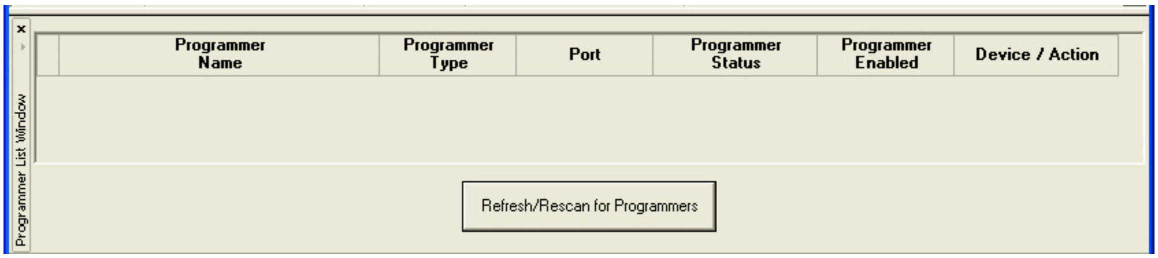


Figure 167 · Programmer List Window

Changing the Name of your Programmer

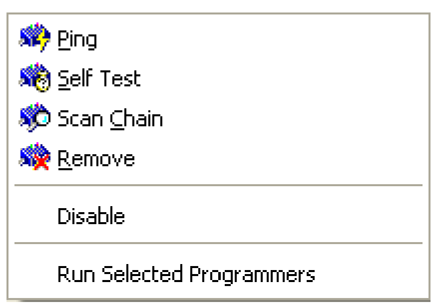
You can change the name of your programmer by double-clicking in the spreadsheet cell or you can choose **Edit Cell** from the right-click menu.

Connecting New Programmers

You can connect new programmers by clicking the **Refresh/Rescan for Programmers** button.

Accessing Right-Click Menus

If you have checked the **Programmer Enabled** checkbox, you can right-click on any of the spreadsheet fields to access the menu in the figure below.



Right-Click Menu

If you have not checked the **Programmer Enabled** checkbox, you can right-click in the on any of the spreadsheet fields, to access a menu to remove or enable the programmer (see figure below).

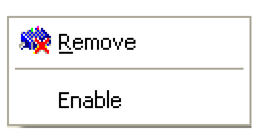


Figure 168 · Right-Click Menu

Programmer Details Window

The Programmer Details Window displays your programmer ID, port, type, name, and programming status (see figure below). Use this window to check the status and access common commands (ping, self-test, scan chain) and to enable/disable or remove the programmer from the chain.

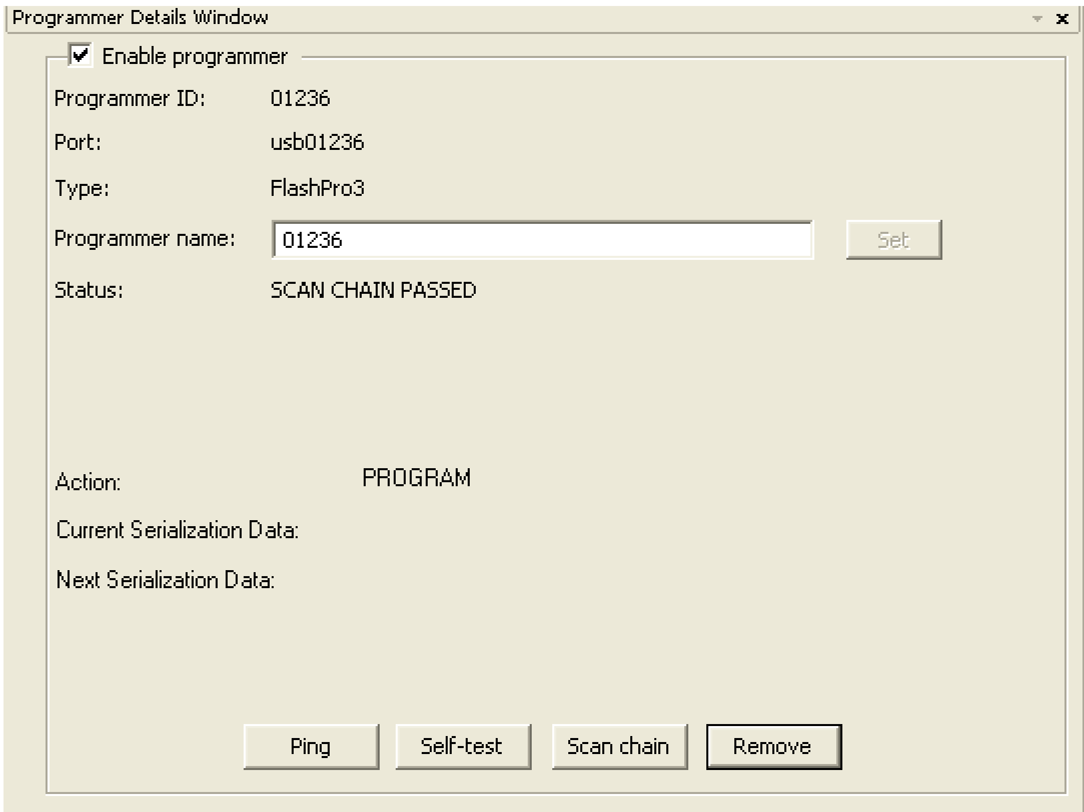


Figure 169 · Programmer Details Window

You can access this window from the **View** menu or you can double click any of the fields in the **Programmer List** Window (Programmer Name, Programmer Type, Port, Programmer Status, and Programmer Enabled).

Click the **Enable Programmer** checkbox to enable your programmer and activate the **Programmer Details** Window.

From the Programmer Details Window, you can ping a programmer, perform a self-test, scan a programmer, or remove a programmer.

FlashPro Single Device Configuration Window

To access the Single Device Configuration Window click the **Configure Device** button in the **Flow window**. The Single Device Configuration window displays PDB/STAPL file and serialization information (see figure below). You can also deactivate serialization by clicking the **Serialization** checkbox.

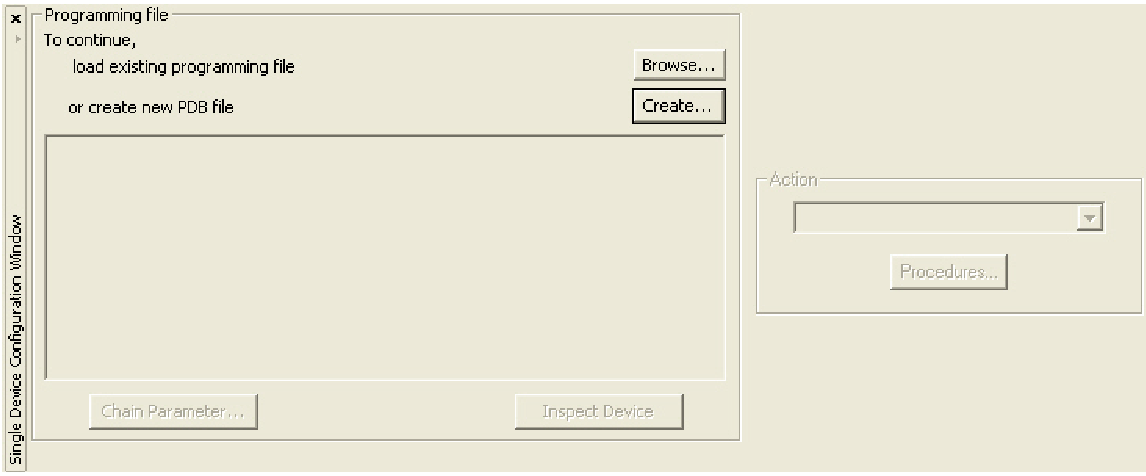


Figure 170 · Single Device Configuration Window

Loading the PDB/STAPL File

You can load your PDB/STAPL file from the **Configuration** menu by choosing **Load Programming File** or by clicking the **Browse** button in the **Single Device Configuration Window**.

You can set chain parameter settings by clicking the **Chain Parameter** button.

Selecting Serialization Indexes

To select the serialization indexes:

1. Check the **Serialization** checkbox to activate serialization.
2. Click the **Select Serialization Indexes** button. The **Serial Settings** dialog box appears (as shown in the figure below).

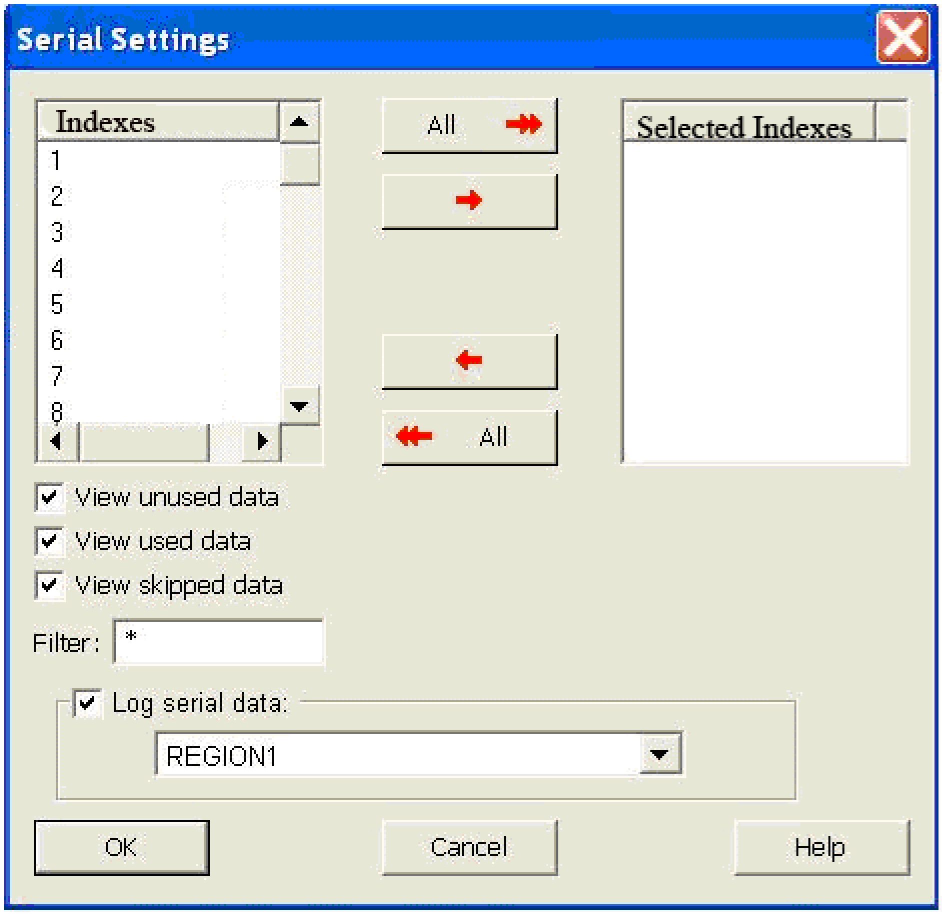


Figure 171 · Serial Settings Dialog Box

3. Click to select an index in the **Indexes** column.
4. Click the red arrow button that is pointing toward the **Selected Indexes** column to move an index to the **Selected Indexes** column.
5. If you want to move all the Indexes to the **Selected Indexes** column, click the **All** button.
6. Click **OK**. Information about your serial indexes displays.

You can move indexes from the **Selected Indexes** column to the **Indexes** column by clicking the red arrow buttons pointing toward the **Indexes** column.

You can set your indexes by choosing the following check boxes: View unused data, View used data, and View skipped data.

Selecting Action and Procedures

You can select actions from the Basic mode and the Advanced mode. The Basic mode is provided for users that only require the Program, Verify and Erase actions. In Basic mode, other actions are not visible, and the Procedures run by an Action cannot be modified.

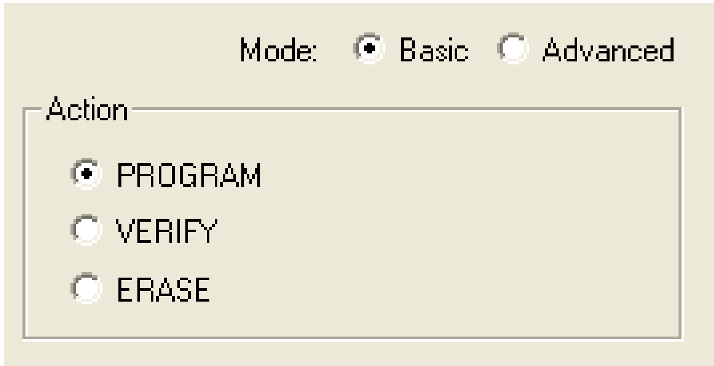


Figure 172 · Basic Mode

The Advanced mode enables you to select an action and modify the procedures for the selected action. In Advanced mode, actions are determined by the stapl standard used.

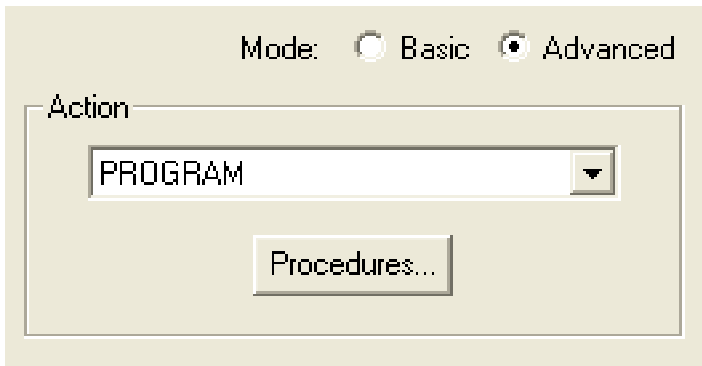


Figure 173 · Advanced Mode

To select an action (Advanced Mode):

1. Click the down arrow in the **Action** menu and select and action (see figure below).

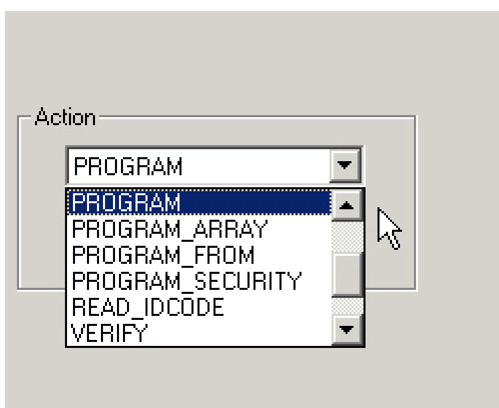


Figure 174 · Action Menu

2. Click the **Procedures** button. The **Select Action and Procedures** dialog box appears.

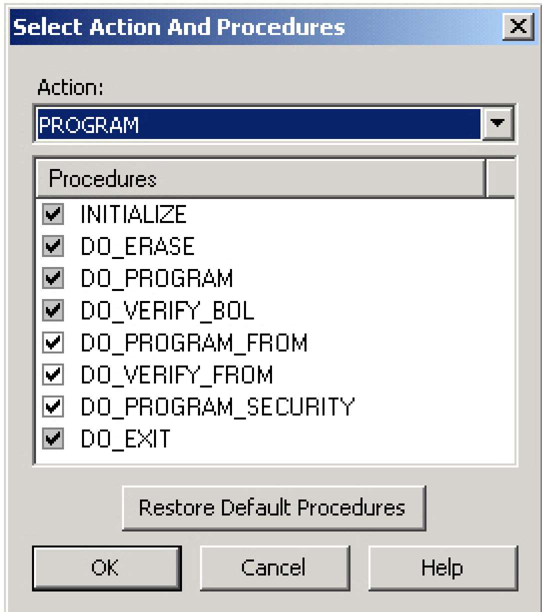


Figure 175 · Select Action And Procedures Dialog Box

3. Select a procedure or click the **Restore Default Procedures** button. Gray checkboxes indicate that the procedure is mandatory.

   Note: The procedures in the Select Action And Procedures dialog box are determined by the STAPL standard. The Recommended procedures are selected by default and the Optional procedures are unselected by default.

4. Click **OK**.

Note: You can also click the Select Action and Procedures dialog box from the toolbar.

Chain Configuration Window

The Chain Configuration Window displays the chain order, the chain editing options, and the chain configuration grid (see figure below).

The **Show Chain Editing** checkbox, when checked, displays your chain editing options (Configure device, Add Actel Device, Add Non-Actel Device, and organization buttons to move your device within the grid).

For information on how to add Actel and Non-Actel devices, see the Chain Editing help topic.

For information on how to use the Organize buttons, see Using the Organize Buttons in the Chain Programming Grid.

You can enable programming and serialization by checking the **Enable Device** checkbox and the **Enable Serial** checkbox in the **Chain Configuration** grid.

Figure 176 · Chain Configuration Window

Auto-Construction of Chain from Scan Chain

When in chain programming mode, the FlashPro software enables you to automatically construct the chain by clicking the **Construct the chain from a Scan Chain operation link**, or by selecting **Construct Chain Automatically** from the **Configuration** menu.

This enables you to scan a chain of devices and automatically construct the chain within FlashPro. If you are using non-Actel devices, you will need to import the device settings into the database by using the Import Settings for Non-Actel Devices dialog box. The software also scans the chain before constructing it, which reduces the possibilities of having errors in the chain. For more information on how to automatically construct a chain from scan chain, refer to the Automatic Chain Construction Tutorial.

Chain Editing Options

The FlashPro software enables you to edit your chain by adding Actel and Non-Actel devices. You can add devices by clicking the **Add Actel Device** button and the Add **Non-Actel Device** button or you can select these options from the **Configuration** menu.

Note: For more information about how to edit the chain, see Chain Editing.

Editing the Chain Configuration Grid

The **Chain Configuration Grid** enables you to select an **Action** for your device, **Enable Serialization**, and edit the grid using the right-click menu.

To select an Action from the Configuration Grid:

1. Choose the device you would like to program and check the **Enable Device** checkbox.
2. In the **Action** column, click the down arrow to expose the drop-down menu (see figure below).
3. Select your desired action.

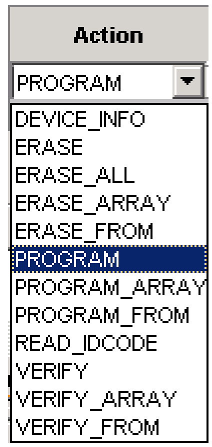


Figure 177 · Drop-Down Menu for Select Action

To enable Serialization:

1. Check the **Enable Serial** checkbox. By enabling serialization, the action options change.

   Note: Note: Before you can enable serialization, you must check the Enable Device checkbox.

2. In the **Action** column, click the down arrow to expose the drop-down menu (see figure below).

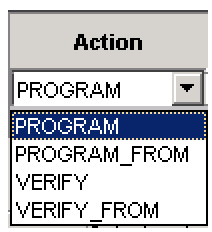


Figure 178 · Drop-Down Menu for Select Action

3. Select your desired action.

4. Choose **Select** button from the **Serial Data** column, which is next to the **Action** column (see figure below).

Figure 179 · Serial Data Column

The **Serial Settings** dialog box appears.

5. Choose your serial settings from the **Serial Settings** dialog box.

See Serial Settings for more information about this topic.

Note: Uncheck the Enable Serial checkbox to disable serialization.

To edit the Chain Configuration Grid:

1. Select the device you would like to edit and right click anywhere in the row of the selected device. The right-click menu below displays.

2. Select and click an option from the right-click menu.

Note: The Device Configuration menu includes options for configuring your device.

Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650. 318.8044**

Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Actel Technical Support

Visit the Actel Customer Support website (http://www.actel.com/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at http://www.actel.com/.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/company/contact/default.aspx.

Actel is the leader in low-power and mixed-signal FPGAs and offers the most comprehensive portfolio of system and power management solutions. Power Matters. Learn more at http://www.actel.com .

Actel Corporation *• 2061 Stierlin Court • Mountain View, CA 94043 • USA*
Phone 650.318.4200 • Fax 650.318.4600 • Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

Actel Europe Ltd*. • River Court, Meadows Business Park • Station Approach, Blackwater • Camberley Surrey GU17 9AB • United Kingdom*
Phone +44 (0) 1276 609 300 • Fax +44 (0) 1276 607 540

Actel Japan *• EXOS Ebisu Building 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan*
Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • http://jp.actel.com

Actel Hong Kong *• Room 2107, China Resources Building • 26 Harbour Road • Wanchai • Hong Kong*
Phone +852 2185 6460 • Fax +852 2185 6488 • www.actel.com.cn

5-02-9138-17/11.10