Using the HP 3070 Tester for In-System Programming

January 2003, ver. 1.2 Application Note 109

Introduction

In-system programming has become a mainstream feature in programmable logic devices (PLDs), offering system designers and test engineers significant cost benefits by integrating PLDs into board-level testing. These benefits include reduced inventory of pre-programmed devices, lower costs, fewer devices damaged by handling, and increased flexibility in engineering changes. Altera provides software and device support that integrates in-system programmability (ISP) into existing test flows for the HP 3070 system (ISP is occasionally referred to as “on-board programming”). This application note discusses how to use the HP 3070 test system to achieve faster programming times for Altera ISP-capable devices.

Device Support

In-system programming can be accomplished with either an adaptive or constant (i.e., fixed) algorithm. Altera offers devices tested with a constant algorithm, which should be used when programming with HP 3070 systems. “F” devices are designated by an “F” suffix in the ordering code and are marked with an “F” in the bottom right-hand corner of the device. Table 1 shows which devices are supported when programming with the HP 3070 test system.

Table 1. “F” Device Support for the HP 3070 Test System

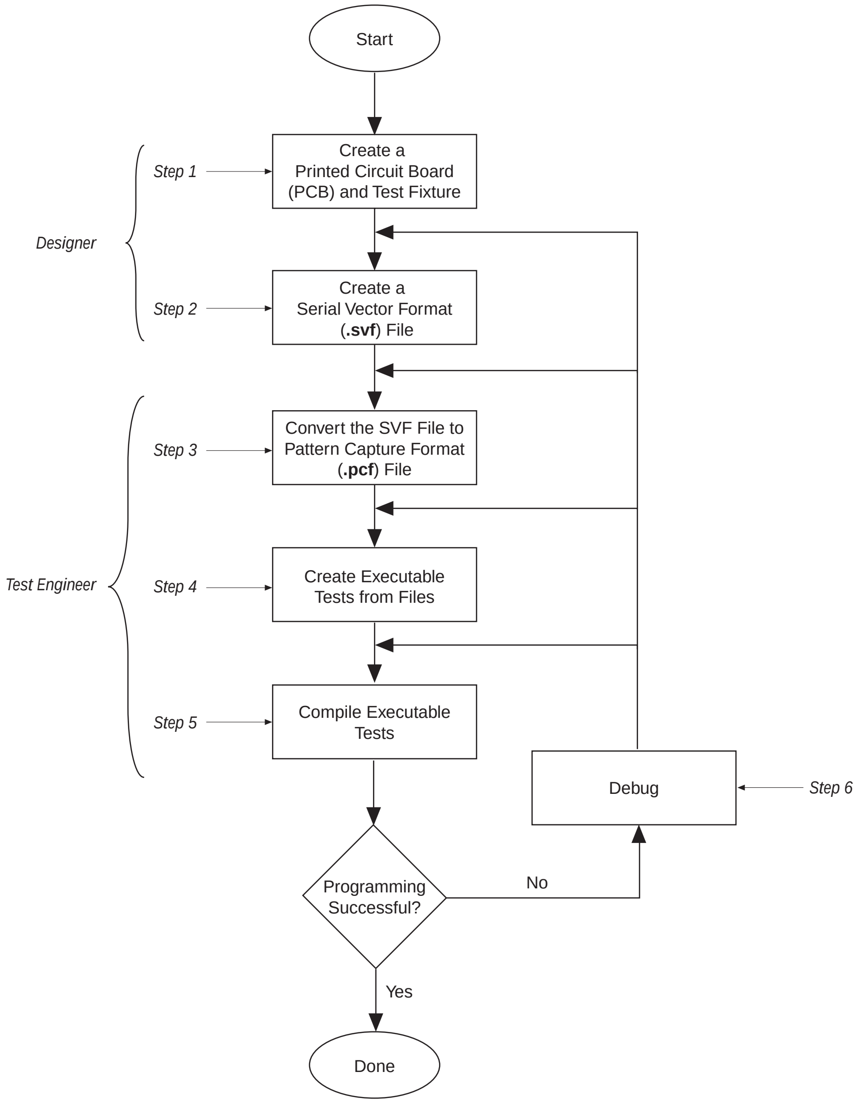
Family	Supported Devices
MAX 7000S	“F”
MAX 7000A	MAX 7000AE
MAX 3000A	MAX 3000A
MAX 9000	“F”
MAX 9000A	“F”

“F” devices allow programming hardware to program all devices from a set of vectors that are defined before programming begins. This process allows in-circuit testers, such as the HP 3070, to apply programming vectors in the simplest and fastest manner possible. In addition, fixed algorithm devices provide a uniform “beat-rate”. Because all of the devices use the same algorithm and vector set, each device has the same programming time. This scenario is desirable in a production environment where a consistent, predictable beat-rate is necessary for smooth operation of the manufacturing line.

HP 3070 Development Flow

Programming devices with the HP 3070 tester requires several simple steps. See Figure 1.

Figure 1. HP 3070 Development Flow for In-System Programming


Start
Designer
Step 1
Create a Printed Circuit Board (PCB) and Test Fixture
Step 2
Create a Serial Vector Format (.svf) File
Test Engineer
Step 3
Convert the SVF File to Pattern Capture Format (.pcf) File
Step 4
Create Executable Tests from Files
Step 5
Compile Executable Tests
Debug
Step 6
Programming Successful?
No
Yes
Done

Step 1: Create a PCB & Test Fixture

Before starting test development, the first step to successful in-system programming is the proper layout of the board and the subsequent creation of the test fixture.

Creating the PCB

The following recommendations highlight important areas of PCB design issues:

- The TCK signal trace should be treated as carefully as a clock tree. TCK is the clock for the entire Joint Test Action Group (JTAG) chain of devices. These devices are edge-triggered on the TCK signal, so it is imperative that this signal be protected from high-frequency noise and have good signal integrity. Ensure that the signal meets the $\mathbf{t_R}$ and $\mathbf{t_F}$ parameters specified in the device data sheet.
- Add a pull-down resistor to TCK. The TCK signal should be held low through a pull-down resistor in-between PCF downloads (for more information on pattern capture format (PCF) downloads, see "Step 2: Create a Serial Vector Format (SVF) File"). You should hold TCK low because the HP 3070 drivers go into a "high-Z" state in-between tests and briefly drive low as the next PCF is applied. When the TCK line "floats", the programming data stream will be corrupted and the device will not be programmed correctly.
- Provide VCC and GND test access points for the nails of the test fixture. During operation, there should be enough access points to allow quiet PCB operation. Having too few access points results in a noisy system that can disrupt JTAG scans.
- Turn off on-board oscillators. During programming, on-board oscillators should have the ability to be electrically turned off to reduce system noise.
- Add external resistors to pull outputs to a defined logic level during programming.

Output pins are tri-stated during programming and are pulled up by a weak internal resistor (approximately 50 ¾). MAX 9000 and MAX 9000A devices do not have internal pull-ups. However, Altera recommends that signals which require a pre-defined level be externally forced to the appropriate level using an external resistor (approximately 1 ¾).

For more information on board design for ISP, see *Application Note 100 (In-System Programmability Guidelines)*.

Creating the Fixture

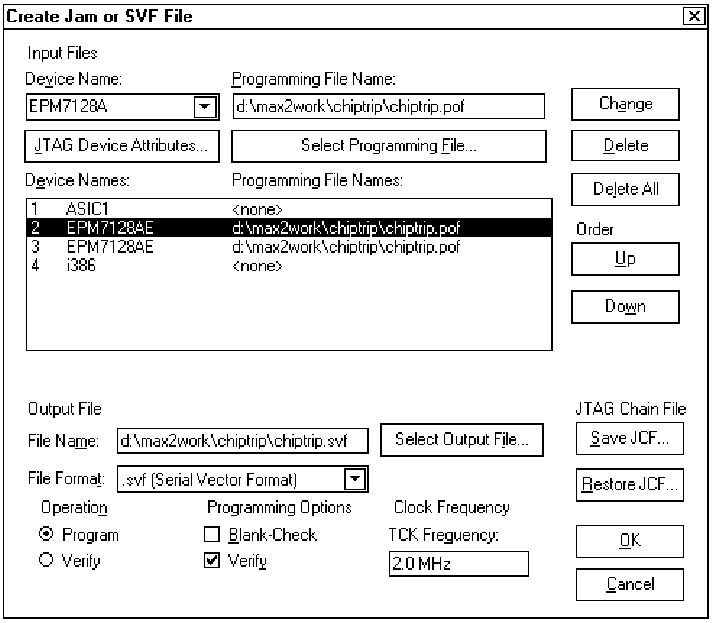
Providing a clean interface between the test fixture and the target board is essential for successful in-system programming. To provide a clean interface, use short wires in the test fixture to improve the TCK connection. Longer wires can introduce inductive noise into the system, which can disrupt programming. The wire connecting TCK should be no longer than 1 inch. Use the HP Fixture Consultant to manage the layout and creation of the test fixture (see the ***HP Series II Board Test Family Manual***).

Step 2: Create a Serial Vector Format (SVF) File

The MAX+PLUS® II software generates SVF Files for programming one or more "F " devices. When targeting multiple devices in the same family, the MAX+PLUS II software automatically generates one SVF File to program the devices concurrently. Therefore, the programming time for all of the devices approaches the programming time for the largest device in the IEEE Std. 1149.1 JTAG chain.

Figure 2 shows the **Create Jam or SVF File** dialog box (File menu), which is used to generate the SVF File.

Figure 2. Create Jam or SVF File Dialog Box



Create Jam or SVF File

Input Files

Device Name: EPM7128A

Programming File Name: d:\max2work\chiptrip\chiptrip.pof

Device Names: / Programming File Names:

1	ASIC1	<none>
2	EPM7128AE	d:\max2work\chiptrip\chiptrip.pof
3	EPM7128AE	d:\max2work\chiptrip\chiptrip.pof
4	i386	<none>

Output File

File Name: d:\max2work\chiptrip\chiptrip.svf

File Format: .svf (Serial Vector Format)

Operation: [✓ Program] Verify

Programming Options: Blank-Check [✓ Verify]

Clock Frequency

TCK Frequency: 2.0 MHz

In Figure 2, the SVF File targets two EPM7128AE devices, where `ASIC1` is the first device in the chain (closest to `TDI`) and `i386` is the last device in the chain (`TDO`). Each Programmer Object File (**.pof**) corresponds to a targeted EPM7128AE device. The value in the *TCK Frequency* box should match the frequency that `TCK` runs at during the test. If you enter a different frequency from the one used in actual testing, programming may fail or you may experience an excessively long programming time.

You can also select whether to perform a program or verify operation and optionally verify or blank-check the device by turning on programming options. Altera recommends generating SVF Files that include verify vectors, which ensure that programming failures are identified and a limited amount of additional programming time is used. You can generate the necessary SVF File based on the scan-chain topology of the board and the Altera devices to be programmed. Once the SVF File is generated, it can be given to test engineers for development.

If a device must be programmed independently, you can generate individual SVF Files for each Altera device in the chain. When creating the SVF File for a single device in the chain, specify the POF for the device and leave the rest of the devices set to `<none>`. These devices are bypassed during programming. Repeat this process until all targeted devices have an SVF File.

Step 3: Convert SVF Files to PCF Files

You must convert the SVF Files to PCF Files for use with the HP 3070 tester with the Altera **svf2pcf** conversion utility. The **svf2pcf** utility can create multiple PCF Files for one device chain; running the utility allows you to specify the number of vectors per file. The amount of memory used by the resulting files varies depending on the data. The HP 3070 digital compiler looks for repeating patterns of vectors and optimizes the directory and sequences RAM on the tester control card to apply the maximum number of vectors before re-loading the files. The number of vectors in a compiled PCF File range from 100,000 to over one million, depending on the size and density of the targeted devices.

To download the **svf2pcf** conversion utility, go to the Altera FTP site at **ftp.altera.com**.

Step 4: Create Executable Tests from Files

Creating digital tests for programming a chain of devices with the HP 3070 tester requires the following steps:

1. Create the library for the target device or scan chain.
2. Run the Test Consultant.
3. Create digital tests.
4. Create the wirelist information for the tests.
5. Modify the test plan.

Create the Library for the Target Device or Scan Chain

The initial program development for the board contains a setup-only node test library for the ISP boundary-scan chain interface. The test library ensures that HP 3070 tester resources are reserved in the test fixture for programming the targeted devices. If only one target device is on the board and it is not part of a boundary-scan chain (isolated), use a pin library; otherwise, use a node library. If using a pin library, you must describe every device pin. Do not include test vectors in a test library.

Figure 3 shows an example of a setup-only node test library.

Figure 3. Code for a Setup-Only Node Test Library

```
!Setup only test for the boundary scan chain
assign  TCK  to  nodes   "TCK"   ! Node name for the TCK pin
assign  TMS  to  nodes   "TMS"   ! Node name for the TMS pin
assign  TDI  to  nodes   "TDI"   ! Node name for the TDI pin
assign  TDO  to  nodes   "TDO"   ! Node name for the TDO pin
inputs  TCK, TMS, TDI
outputs TDO
pcf order is TCK, TMS, TDI, TDO  ! The order is defined by the program that
                                  ! generates the PCF files.
```

Mark the `TCK` and `TMS` boundary-scan nodes as critical in the Board Consultant. This critical marking minimizes the nodes’ wire length in the test fixture.

Run the Test Consultant

Run the Test Consultant to create all of the files for new board development. Once the Test Consultant finishes running with this setup-only test library, it creates an executable test (without vectors) with the correct fixture wiring resource information. Use this file as a template to create the executable test's source code.

Create Digital Tests

Create the digital tests, which are required to program the device(s), by copying the executable template to the desired program names. For example, if **svf2pcf** created four PCF Files, copy the template file to four executable tests (e.g., `prog_a`, `prog_b`, `prog_c`, and `prog_d`) in the digital directory.

Add these test names to your **testorder** file and mark them `permanent` using the following syntax:

```
test digital "prog_a";  permanent
test digital "prog_b";  permanent
test digital "prog_c";  permanent
test digital "prog_d";  permanent
```

Create the Wirelist Information for the Tests

Compile these executable tests to generate object files (see "Modify the Test Plan") for the setup only versions of the tests. Run **Module Pin Assignment** to create the necessary entries in the **wirelist** file.

Next, modify the executable tests so that they contain the vectors to program the target device. An `include` statement can be used in the executable test, or the vectors can be merged into the file. Use the following syntax for the `include` statement, which should be the last statement in the executable test.

```
include "pcf1"
```

Remember that the PCF File must reside in the digital directory and must be a digital file. To ensure that the digital file is in the correct directory, run the following command on the BT-Basic command line:

```
load  digital  "digital/pcf1"  |  re-save
```

You can also use the `chtype` command at a UNIX shell prompt to verify the location of the file:

```
chtype  -n6  digital/pcf1
```

Repeat this step for each PCF File.

Modify the Test Plan

Add the test statements to the test plan using the following syntax:

```
test  "digital/prog_a" ! First program file
test  "digital/prog_b" ! Second program file
test  "digital/prog_c" ! Third program file
test  "digital/prog_d" ! Fourth program file
```

Keep the test execution in the same order in which the SVF File was split. For example, if the SVF File was split into four files (**pcf1**, **pcf2**, **pcf3**, and **pcf4**), the tests must be executed in the order that they split (execute `prog_a` followed by `prog_b` followed by `prog_c` followed by `prog_d`). If the order is not preserved, the device(s) will fail to program correctly.

Step 5: Compile the Executable Tests

Altera recommends batch-driven compilation using either BT-Basic or a UNIX shell. See the following batch file code in BT-Basic (assuming four executable tests to program the target device and generation of debugging object code):

```
compile  "digital/prog_a" ;  debug
compile  "digital/prog_b" ;  debug
compile  "digital/prog_c" ;  debug
compile  "digital/prog_d" ;  debug
```

This file should be saved in the board directory to allow engineering changes to take place at a later date. See the corresponding UNIX shell script (`–D` option generates debugging information):

```
dcomp  -D  digital/prog_a
dcomp  -D  digital/prog_b
dcomp  -D  digital/prog_c
dcomp  -D  digital/prog_d
```

☞ Compile times can be long depending on the number of PCF vectors contained in the source files, the type of controller, and controller loading. Altera recommends using a batch file to automate the compilation of the ISP tests.

If a boundary-scan chain containing Altera devices is defined, only the Altera devices will be programmed when the PCF vectors have been applied to the JTAG interface.

Step 6: Debug the Test

Once the executable tests have been created, the test system can be debugged. The applied vector set ensures that the device is programmed correctly by verifying the contents of the device. The programming algorithm uses the `TDO` pin to check the bitstream coming from the device. If any vector does not match the expected value, the test fails, indicating one of two things:

- The device ID does not match what is expected. This scenario will be evident if the failure occurs at the beginning of the first test. This error is encountered when non-"F" devices are used. Re-apply the vectors using an "F" device. (see "Device Support" on page 1 for more details on "F" versus non-"F" devices).
- Device programming failed.

Because many vectors are verified, it may not be practical to sift through each vector to determine the cause of the failure. Use the following troubleshooting guidelines if the device fails to program:

- Check the pull-down resistor in the test fixture. The design engineer may have placed pull-up resistors on the board for the `TCK` pin. If the pull-down resistor is too large, the `TCK` pin may be above the device's threshold for a logic low. Adjust the value of the resistor accordingly. See the appropriate device family data sheet for the specification on input logic levels.
- If an overpower error on the `TCK` pin occurs, check the value of the resistors because they may be too low for the test system to back-drive for an extended period of time.
- Ensure that the test execution order is correct. If the tests are executed out of order, the programming information will be incorrect. Also, if the same test is executed twice in a row, the target device will be out of sequence and will not receive the correct programming information.
- Ensure that the actual vectors match the expected values for the input pins (`TCK`, `TMS`, and `TDI`). If they are not the same, the tests may need to be recompiled.
- Ensure that the `pcf order` statement in the test matches the order of the PCF code generated in "Step 2: Create a Serial Vector Format (SVF) File" on page 4. If they do not match, the order must be changed and the tests recompiled.
- If possible, verify that the device is programmed correctly by using the MAX+PLUS II software, the ByteBlasterMV™ download cable, and the POF that was used to generate the SVF File. This action is not practical in a production situation, but is useful during test development and debugging.

- If you need to isolate an individual device, you can generate an individual SVF File for each targeted Altera device in the chain. The chain is described with one device getting a POF and everything else in the chain placed in bypass (i.e., `<none>` in the *Programming File Names* box of the **Create Jam or SVF File** dialog box). Then, another SVF File is generated for the next Altera device with all of the other devices in bypass mode. Repeat this process until all targeted devices have an SVF File. This process is useful when a verification error occurs and more than one Altera device is programmed in the chain. This method narrows the verify errors down to the source device.
- If you still have problems, look at the boundary-scan chain definition. Make sure that the number of bits for the instruction register are specified correctly for each device in the chain. If an incorrect number of bits have been defined for any device in the chain, the programming test will fail.

Once the test is running smoothly, the board is ready for production programming. Altera recommends saving the PCF Files and object code for back-up purposes. Use a compression program to minimize the size of the stored binaries and files.

Programming Times

Programming times on the HP 3070 are very consistent. The only variable is the TCK frequency, which affects programming times. The faster the clock, the less time is spent shifting data into the device. *Application Note 85 (In-System Programming Times for MAX Devices)* provides detailed programming time data for in-circuit testers. This application note includes programming times as a function of the TCK frequency, and the number of devices being programmed. The data provided in this application note is based on theoretical calculations. However, as tests have shown, these numbers are accurate to within less than one second of actual programming times on the HP 3070 tester.

The following example provides detailed data points from a specific test case. The data shown in this example provides a general idea of the typical programming times that can be expected, as well as information on file sizes and resources utilized on the host workstation. In this example, four EPM7128SQC160-7F devices are programmed in a chain, using the HP 3070. An SVF File is generated targeting all four EPM7128S devices and the test flow is used to convert the PCF Files into executable vectors. The results are shown in Tables 2 and 3.

Table 2. Data for Programming Four EPM7128SQC160-7F Devices

HP 3070 software revision	B.02.54
Controller type	725/100
Number of PCF Files created	15
Number of vectors per file	about 700,000
Total number of vectors executed	9,925,512
Size of each PCF File	5.4 Mb
Total disc storage for PCF Files	78.7 Mb
Total size of object files (15)	1.5 Mb (about 100,000 bytes each)
Total size of debug objects (15)	430,901 (about 28,800 bytes each)
Total compile time	3 hours, 17 minutes

Table 3. Programming Times—Four EPM7128SQC160-7F Devices

	TCK = 500 kHz	TCK = 2 MHz
Vector cycle time	1,000 ns (1 µs)	250 ns
Test time to program all four devices (first run)	52 seconds	41 seconds
Test time to program all four devices (subsequent runs)	23 seconds	6 seconds

As expected, the programming time is a function of the TCK clock rate. Over this frequency range, the relationship between the frequency and programming time is linear. At higher frequencies, the programming time begins to asymptotically approach the theoretical programming time, reported in *Application Note 85 (In-System Programming Times for MAX Devices)*. Additionally, the example shows that the concurrent algorithm reduces the programming time of the four devices to the programming time of only one EPM7128S device.

Guidelines

While using the HP 3070 tester for programming, use the following guidelines:

- Use caution if a pin library is used to describe the target device in a stand-alone boundary-scan chain. Altera does not recommend describing all of the ISP device's I/O pins as bidirectional. This practice uses a large number of hybrid card channels and potentially causes a fixture overflow error when developing the test.
- Do not include PCF vectors in the test library. Use a setup-only node library. Creating a test library with PCF vectors creates a large library object file and results in a much slower test development time. This delay occurs because the integrated program generator (IPG) looks at the entire vector set of the library object to determine if vectors need to be commented out due to conflicts. Library object compiles are different from executable compiles. Additionally, the IPG may fail due to the large library object file.

- To save time and disk space, generate SVF Files that include a verify in the programming operation. This process integrates verification vectors into one step, minimizing the amount of work in the test development process. This integrated verify accurately captures any programming errors; therefore, it is not necessary to add an additional stand-alone verify in the test sequence.
- While this document describes how to generate a test to apply vectors to the device for programming, a boundary-scan description language (BSDL) file is required to functionally test the device. If you need to perform a boundary-scan test or functional test, generate a BSDL file for the programmed state of the target device that contains the pin configuration information (e.g., which pins are inputs, outputs, or bidirectional pins). Use the HP 3070 boundary-scan software to generate a test.

For more information about Altera's support for boundary-scan testing, see *Application Note 39 (IEEE 1149.1 (JTAG) Boundary-Scan Testing in Altera Devices)*.

Conclusion

Altera provides complete solutions for programming all ISP-capable devices using the HP 3070 test system. Fast programming times are achieved within the existing test flow for the HP 3070 tester. Additionally, Altera offers "F" devices, which provide fast, consistent programming times in a production environment. With both software and device support, the opportunity for cutting costs and increasing manufacturing productivity is available to any HP 3070 user.

Revision History

The information contained in *Application Note 109 (Using the HP 3070 Tester for In-System Programming)* version 1.2 supersedes information published in previous versions. Version 1.2 contains corrected information in Table 1.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
lit_req@altera.com