AN98558

In-System Programming for Cypress SPI Flash on Altera® FPGA Board

AN98558 introduces an alternate method to in-system program the Cypress SPI flash by using Altera's Nios® II tool, which works with all versions of the Quartus II software.

1 Introduction

In an Active Serial (AS) configuration scheme, an SPI flash device can be used to configure the Altera FPGA that acts as the configuration master while the SPI flash acts as a slave.

Altera recommends using their serial configuration devices (EPCS) in the Active Serial scheme, although users may prefer to use Cypress SPI flash instead of EPCS devices. In such cases, users may not be able to use the built-in Quartus Flash Programmer to program a JTAG Indirect File (*.jic) to the Cypress SPI flash, because the FPGA checks the EPCS Device ID before reading the configuration data. Users can disable EPCS ID check during the *.jic file conversion. If such method works, there is no need to use the workaround described in this application note.

However, if the Quartus Programmer cannot properly detect the Cypress device, this application note introduces an alternate method to in-system program the Cypress SPI flash by using Altera's Nios® II tool, which works with all versions of Quartus software.

2 Background Information

The procedure introduced in this application note was verified on the Altera Cyclone IV and Cyclone V Development Board, using Quartus II 13.0sp1 Web Edition software, and updated with Quartus Prime 17.0. In general, the procedure should apply to all other versions of Quartus software.

The document is written assuming the reader is familiar with Altera FPGA development, including the Nios II Flash Programmer User Guide.

3 Procedures

3.1 Create a Flash Programmer Target Design

Creating the flash programming target design is the most time-consuming step in this procedure; however, it only needs to be performed once. After the target design to communicate to the SPI flash has been created, the resulting *.sof file can be used for all future programming.

In this step, use of the Qsys tool (SOPC Builder in earlier versions Quartus software) is required to build a minimum component set design required by the Nios II tool. The Qsys System Design Tutorial provides an introduction to the Qsys tool.

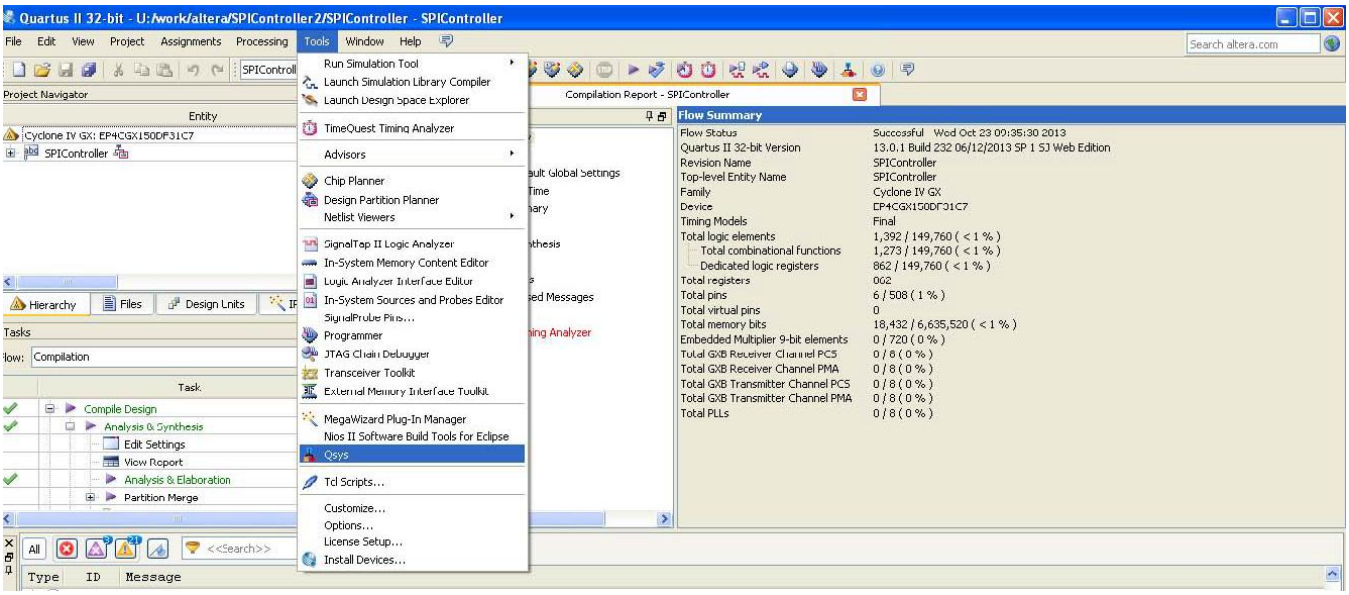
This procedure follows the steps outlined in the "Flash Programmer Target Design" section in Altera's Nios II Flash Programmer User Guide. As stated in the User Guide, two components are required, at a minimum:

- Nios II processor, with JTAG debug module level 1 or greater
- EPCS Serial Flash Controller

Do the following to build the design:

1. Open Quartus II software (screenshot examples are based on rev. 13.0sp1, or stated otherwise).
2. Under **Tools** menu, open **Qsys**.

Figure 1. Launch Qsys



3. Add the Nios II Processor component. Use Nios II/e core, which does not require a license. In the example, this instance is named “cpu”.
   a. Select Nios II/e.
   b. Select **Absolute** for Reset and Exception vector memory to avoid errors.
   c. Select JTAG Level 1 debug.

   Others can be just default values.
4. Add EPCS Serial Flash Controller. In the example, this instance is named “spansion_spi_flash”.
   a. If using Quartus 17 Prime, select “use dedicated active serial interface” to let the tool to automatically complete the pin assignments.
5. Make connections as shown in Figure 2.

Figure 2. Quartus 13.1 sp1 Example

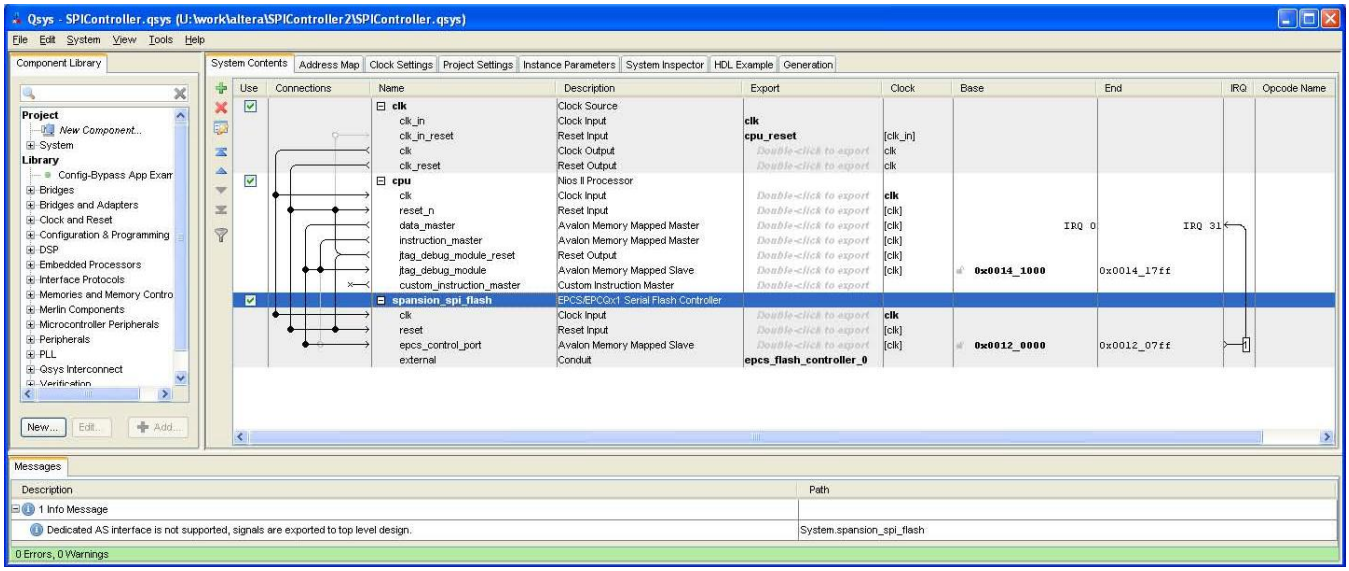
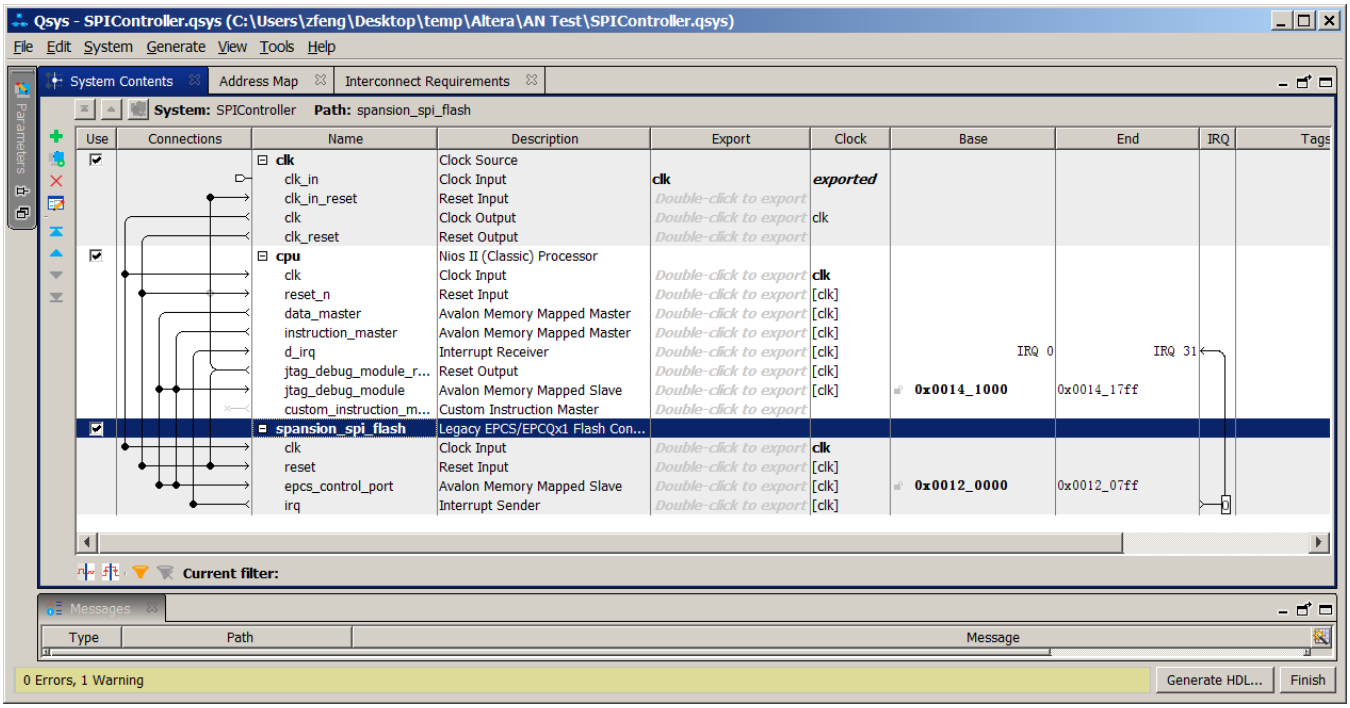
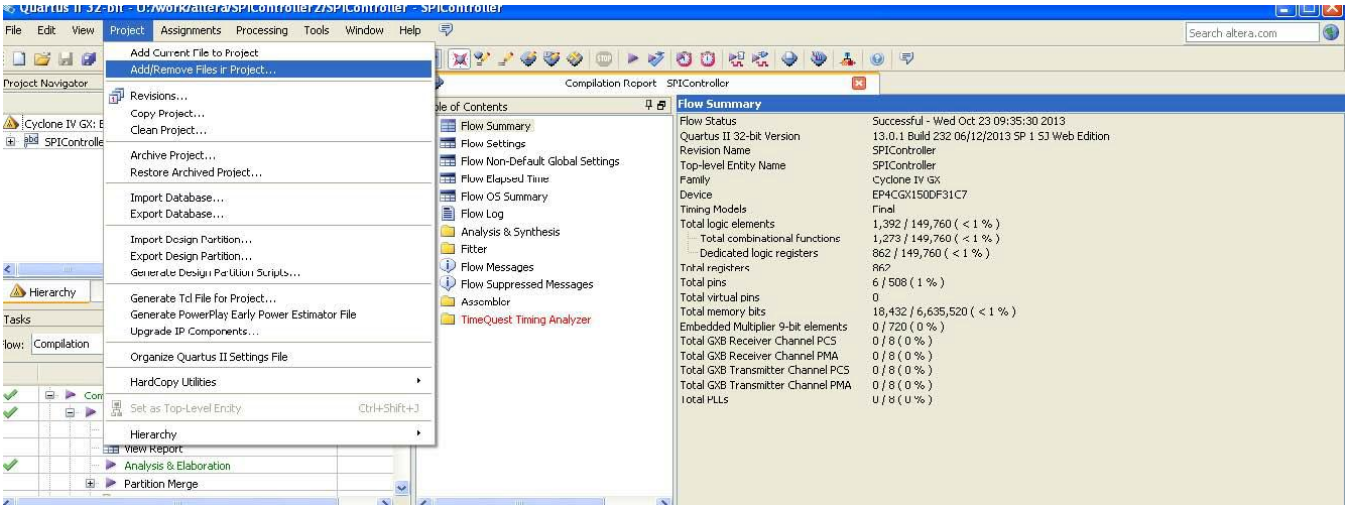


Figure 3. Quartus Prime 17.0 Example



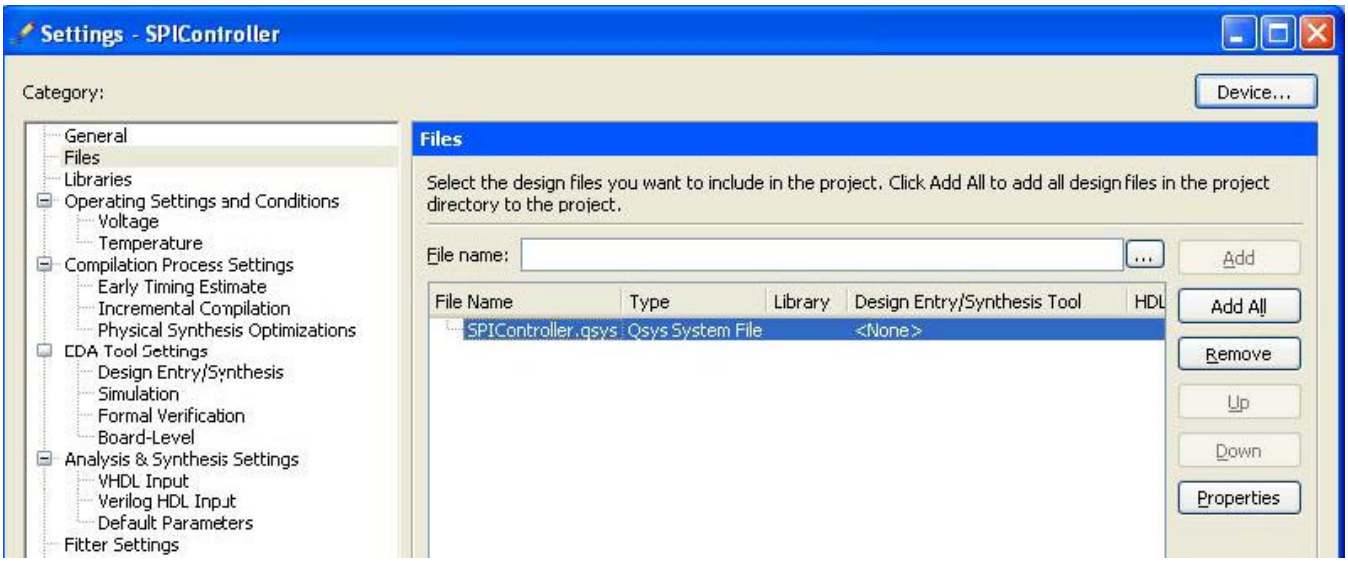
7. Assign the Export pin names as shown in Figure 3.
8. Assign the Base Addresses as shown. Take note of the spansion_spi_flash base address. It is set to 0x120000 in this example. This value will be used later.
9. Save the Qsys design as *SPIController.qsys*.
10. In the Quartus software, open a new project using the New Project Wizard.
11. After the new project is created, go to **Project** > **Add/Remove Files in Project**.

Figure 4. Add Files in Project



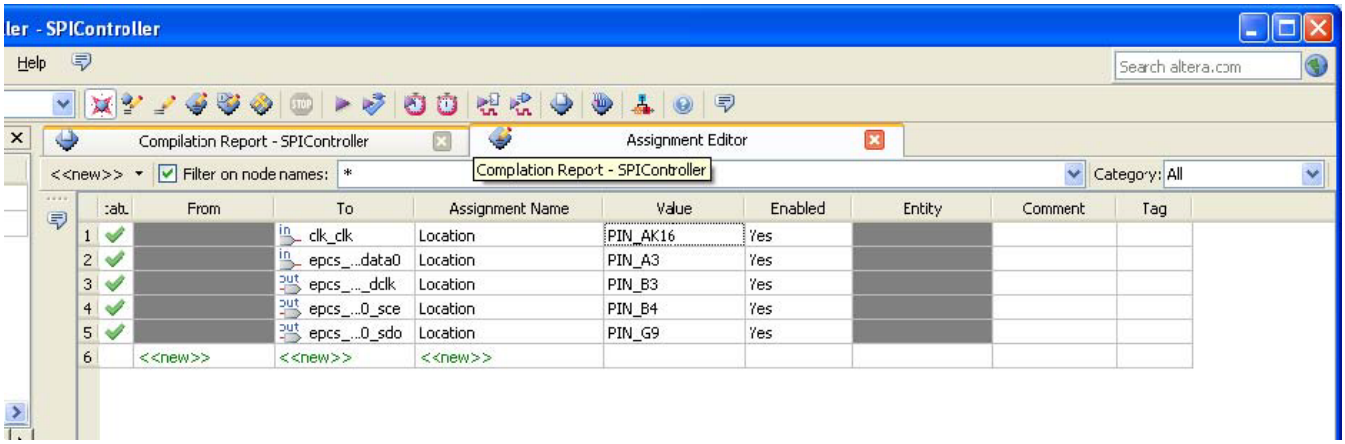
12. Add *SPIController.qsys* to the project. After adding the file, it should look like Figure 5:

Figure 5. Files Added to Project



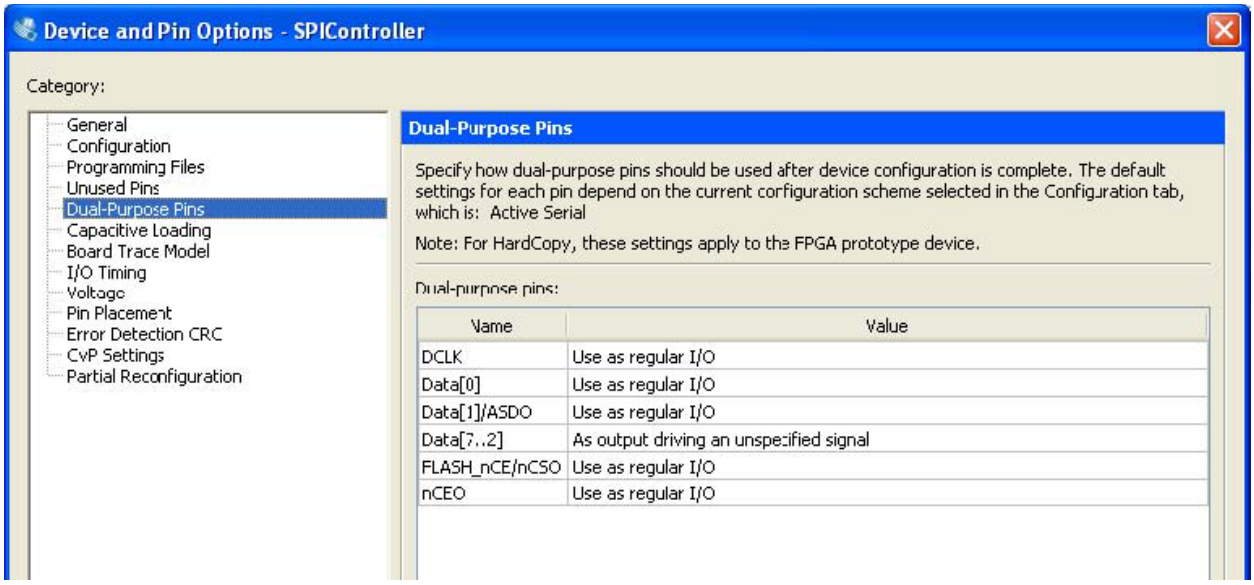
13. Select General: Choose SPIController as the top level entity from the General menu.
14. Use **Assignments** > **Assignment Editor** to assign pin connections. This will depend on your particular FPGA design board. Figure 6 shows the pin assignments made with the Cyclone IV in this example.:

Figure 6. Assigned Pin Connections



15. Make sure the pins are defined as “Use as regular I/O” to prevent compilation errors. This can be done in the **Device and Pin Options** Window as shown in Figure 7.

Figure 7. Device and Pin Options



16. Compile the new project. A *.sof file will be generated. In this example, it is named *SPIController.sof*.

A simple design has now been created to control the SPI flash.

3.2 Create an Override File for Nios II

This step is also done only once. Generate a text file called *nios2-flash-override.txt* with the following contents:

```
[EPCS-012018] # Cypress SPI Flash S25FL128S
sector_size = 65536
sector_count = 128
```

The “012018” suffix corresponds to the first three bytes returned by the RDID (opcode 9Fh) command. They are the “Manufacturer ID”, “Device ID MSB”, and “Device ID LSB” respectively. In this example, the Cypress S25FL128S SPI flash was used. The three byte values are “01h, 20h, 18h”.

These ID numbers, as well as the sector size and count numbers in the text file, may be altered based on the specific flash used. For example, if S25FL256S SPI flash was used, the override file should look like:

```
[EPCS-012019] # Cypress SPI Flash S25FL256S
sector_size = 65536
sector_count = 256
```

Place this text file inside the installed nios II bin directory. For example: *c:\altera\13.0sp1\nios2eds\bin*.

3.3 Verify the SPI Flash Can Be Accessed

Now check to see if the design generated can access the SPI flash.

Power up the FPGA board and connect the USB cable, assuming the built-in USB Blaster is used. Verify by using the Quartus Programmer software that the FPGA can be auto detected.

If using the Cyclone IV Development board as in this example, verify the MAX II Page Select Switch is set to EPCS. This is realized by pressing the PGM SEL button until the EPCS LED (D19) is illuminated.

Open the nios2 shell from the Program > Altera installed package menu. For example: **Program** > **Altera 13.0.1.232 Web Edition** > **Nios II EDS 13.0.1.232** > **Nios II 13.0sp1 Command Shell**.

Navigate to the directory where the *SPIController.sof* was generated during the first step. Execute the following command:

```
nios2-configure-sof SPIController.sof
```

Quartus Prime 17.0 Nios II shell requires a device index for this command, for example:

```
nios2-configure-sof -d 2 SPIController.sof
```

The Quartus II Programmer Running message should be visible. Execute the following command:

```
nios2-flash-programmer --epcs --base_address=0x120000 --debug
```

Or for Quartus Prime 17.0 Nios II shell:

```
nios2-flash-programmer --epcs –base 0x120000 --debug
```

Note: The base address is the one used in SPI Controller design generated as described in the Create a Flash Programmer Target Design section. If a different base address is used, it should be used here. The EPCS Flash Detection message should be visible, similar to below:

```
Reading override file "C:/altera/13.0sp1/nios2eds/bin/nios2-flash-override.txt"
Using cable "USB-Blaster [USB-0]", device 1, instance 0x00
Resetting and pausing target processor: OK
Processor data bus width is 32 bits
Looking for EPCS registers at address 0x00120000 (with 32bit alignment)
Initial values: 0001703A 04C00074 9801483A 9CFFF804 983FFD1E 0000203A
Not here: reserved fields are non-zero
Looking for EPCS registers at address 0x00120100 (with 32bit alignment)
Initial values: 93000237 6300080C 603FFD26 90000335 A8000C26 03010004
Not here: reserved fields are non-zero
Looking for EPCS registers at address 0x00120200 (with 32bit alignment)
Initial values: 4A40100C 483FFD26 92C00135 92400237 4A40200C 483FFD26
Not here: reserved fields are non-zero
Looking for EPCS registers at address 0x00120300 (with 32bit alignment)
Initial values: 00000000 00000000 00000000 00000000 00000000 00000000
Not here: SPI_SLAVE_SEL has 0 valid bits (should be between 1 and 16)
Looking for EPCS registers at address 0x00120400 (with 32bit alignment)
Initial values: 00000000 00000000 00000260 00000000 00000000 00000001
Valid registers found
EPCS signature is 0x17
EPCS identifier is 0x012018
Using EPCS size information from section [EPCS-012018]
Device size is 8MByte (64Mbit)
Erase regions are:
offset 0: 128 x 64K
EPCS status is 0x00
Leaving target processor paused
```

The message shows that it finds the EPCS Flash (in this case, the Cypress SPI flash) at the address 0x120400. Take note of this address as it will be used below.

3.4 Convert Your Own Design File (*.sof) to a Hex File (*.flash)

After building the design configuration file, use the nios2 tool to convert it to a hex file in preparation for programming the SPI flash by executing the following command:

```
sof2flash --epcs --input=<your design file>.sof --output=<your design file>.flash
```

The message indicating the conversion is successful should be visible. The *.flash file is generated.

3.5 Program the .flash File to Cypress Flash

Now, program the configuration file to the Cypress flash by executing the following command:

```
nios2-flash-programmer --epcs --base_address=0x120400 <your design file>.flash
```

The base address used here is the one discovered as described in the Verify the SPI Flash Can Be Accessed section.

For Quartus Prime 17.0 Nios II shell, specify the override file as a command option:

```
nios2-flash-programmer --epcs –base 0x120400 --override nios2-flash-override.txt <your design file>.flash
```

The message indicating Erasing and Programming the flash should be visible.

After this step, flash programming has now been completed. Reboot the FPGA and verify it is properly configured from the SPI flash using the created FPGA design.

4 Conclusion

Although the built-in Quartus II Flash Programmer may not work directly with Cypress SPI flash devices, it is possible to use the Nios II tool to in-system program Cypress SPI flash to configure Altera FPGAs. The complete procedure is documented in this application note.

Contact Cypress Customer Support if any issues are encountered implementing this procedure with Cypress SPI flash.

Document History

Document Title: AN98558 - In-System Programming for Cypress SPI Flash on Altera® FPGA Board

Document Number: 001-98558

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	–	ZHFE	11/25/2013	New Application Note
*A	4960725	SZZX	10/20/2015	Updated in Cypress template
*B	5884969	ZHFE	09/18/2017	Updated logo and Copyright
*C	6111317	ZHFE	03/27/2018	Updated to include procedural changes with Quartus Prime 17.0 Updated template

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2013-2018. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. No computing device can be absolutely secure. Therefore, despite security measures implemented in Cypress hardware or software products, Cypress does not assume any liability arising out of any security breach, such as unauthorized access to or use of a Cypress product. In addition, the products described in these materials may contain design defects or errors known as errata which may cause the product to deviate from published specifications. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.