10555 Scripps Trl, San Diego, CA 92131
858-877-3485 858-408-9550
http://www.pulserain.com

Doc# TRM-0922-3029, Rev 1.0.0

M10 High Speed Configuration

- Dual Boot Solution for Intel® MAX® 10 FPGA

Technical Reference Manual

Nov, 2017

This page is intentionally left blank.

Table of Contents

References

1. PulseRain FP51-1T Microcontroller – Technical Reference Manual, Doc # TRM-0923-00001, Rev 1.0.0, 09/2017, https://github.com/PulseRain/Arduino_M10_IDE/raw/master/docs/PulseRain_8_bit_MCU_TRM.pdf
2. Building Embedded Systems – Programmable Hardware, Changyi Gu, APress Media, July 2016 http://www.apress.com/us/book/9781484219188

1 Introduction

1.1 In Field Upgrade

Intel's MAX® 10 FPGAs can offer a high density of logic elements and block RAMs, plus the tight integration of flash memory. They are widely adopted by industry control, audio/video processing, computer storage, telecommunication, etc. Such popularity inevitably puts a strain on their product maintenance, which consequently brings **in-field upgrade** to the fore.

However, the conventional solutions for in-field upgrade all have their shares of shortcomings, as explained below:

1. Using JTAG to write a new configuration image to flash
   The drawback of this approach is that JTAG connector is bulky. And oftentimes JTAG is not accessible on small form factor products. To program a new image through JTAG, the user has to have a JTAG cable ready, which may not be the case in field.

2. Using single uncompressed image
   In this way, there is only one image stored in the flash memory. And this image has to take on two jobs simultaneously: First, it has to function as intended by its designer. Secondly, it has to take on the side job to figure out the in-field upgrade all by itself. And such in-field upgrade is often implemented ad hoc.

   Since there is only one image on the flash, an erase-write operation on this single image exposes vulnerability during in-field upgrade. If for some reason the upgrade failed mid-stream (such as power loss, user error etc.), the target device will have no valid image left and become brain-dead.

   Also, the side job of in-field upgrade will take precious on-chip resource away from its main job. In other words, such approach will be very intrusive and act more like a distraction to its main job.

3. Using Dual Compress Images for Boot
   This is often the preferred approach as the two images are totally decoupled with each other. And one of the images (named as **main image** for the rest of this literature) can focus on its main function while the other image (named as **boot image** for the rest of this literature) only deals with in-field upgrade. However, because the two images are both compressed, block ram initialization at power-on is no longer available. In conventional approach, the boot image will either do something ad hoc, or occupy the precious UFM (user flash memory) to store its firmware image. And because it stores code on the UFM, upgrading the UFM itself will also expose vulnerability (although the main image might still survive when UFM upgrade fails.)

To challenge the status quo and offer a general solution for in-field upgrade, PulseRain Technology has come up with its own solution: **M10 High Speed Configuration**.

1.2 M10 High Speed Configuration

As illustrated below in Figure 1-1, the M10 High Speed Configuration is composed of two parts: the **boot image** and the **host program**.

The boot image is composed of the following subparts:

- A PulseRain FP51-1T soft-core MCU.
- Dual port on-chip RAM to store code for the processor
- Bootstrapper. Different from the conventional approach mentioned early, M10 High Speed Configuration does not take any user flash memory (**UFM**). Instead, it uses a bootstrapper to get code from the host PC, and initialize the dual port RAM with that code. Based on the code received from the host, the processor (FP51-1T) will carry out the job for In-field upgrade. Since the firmware code of the boot image is stored on host machine (Windows PC) instead of UFM, this solution is future proof!

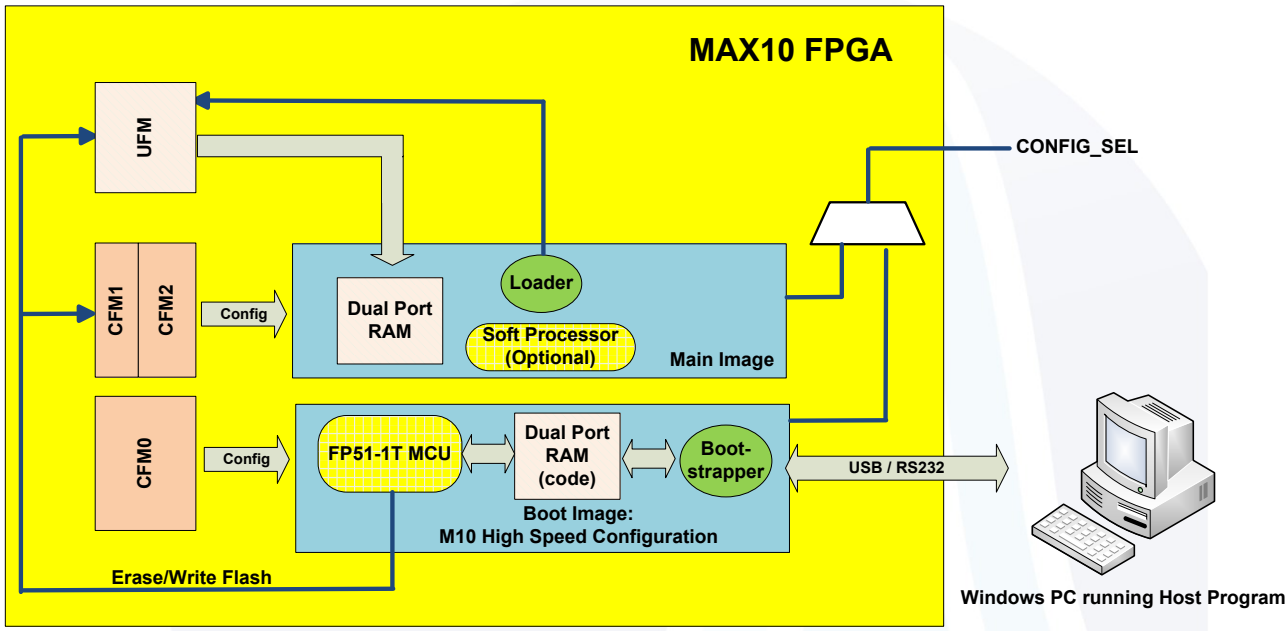

Figure 1-1 The Architecture of M10 High Speed Configuration

The host program, on the other hand, runs on a Windows PC. In M10 High Speed Configuration, it is actually a python script (For those who don't have python installed, an exe file is also provided.). At its startup, the host program will communicate with the bootstrapper and download code to FP51-1T processor in the boot image. After that, it will bring out a GUI to let user write new images to CFM/UFM. But no matter what, the CFM0 (the sector that stores the boot image) will not be changed.

As mentioned early, the main image can now focus on its intended task without worrying about the in-field upgrade, since the boot image and the main image are physically separated in Figure 1-1. Although the main image is not part of the M10 High Speed Configuration, one of its possible implementations is demonstrated

in Figure 1-1. If there is any on-chip RAM that needs to be initialized (such as code for a soft processor, or a lookup table etc.), the RAM content can be saved in UFM. After power-on reset, a simple loader can be used to load UFM content into the RAM. The content of the UFM can be saved as a file of Intel HEX format, and programmed to UFM through the M10 High Speed Configuration Utility.

MAX® 10 FPGA provides a pin called CONFIG_SEL that let user choose the active image at power-on. And this pin can be set through a jumper or switch on the PCB.

2 General Work Flow

2.1 Prepare the main image

As shown in Figure 1-1, the main image is physically separated from the boot image. But to work with M10 High Speed Configuration, the main image has to be configured as "dual compressed images". And the following example assumes 10M08SAE144C8G device is used.

In Quartus Prime Lite Edition 16.0, the configuration can be set as following:

1. Choose the menu "**Assignments/Device...**"
2. In the Device dialogue, choose "**Device and Pin Options...**", as illustrated in Figure 2-1.
3. In the Device and Pin Options dialogue, choose Configuration on the left side, and then set Configuration mode to "**Dual Compressed Image...**", as illustrated in Figure 2-1.
4. Add the IP core called "**altera_dual_boot**" into the design. If the design is already using Nios2 or any other Qsys systems, the altera_dual_boot can be added to the existing Qsys, like the one shown in Figure 2-2. Otherwise, the "**altera_dual_boot**" can be added to the top design by creating a new Qsys that only contains "**altera_dual_boot**". In this regard, a good example can be found under the source repository ***https://github.com/PulseRain/Mustang/tree/master/cores/dual_config***. Users can just copy this example and change its clock frequency to any intended number. And at the top level, this module can be instantiated like the following:

```
dual_config dual_config_i (
    .avmm_rcv_address (3'd0),          // avalon.address
    .avmm_rcv_read (1'b0),             //     .read
    .avmm_rcv_writedata (32'd0),       //     .writedata
    .avmm_rcv_write (1'b0),            //     .write
    .avmm_rcv_readdata(),              //     .readdata
    .clk (clk_24MHz),                  //   clk.clk
    .nreset (1'b1)                     // nreset.reset_n
  );
```

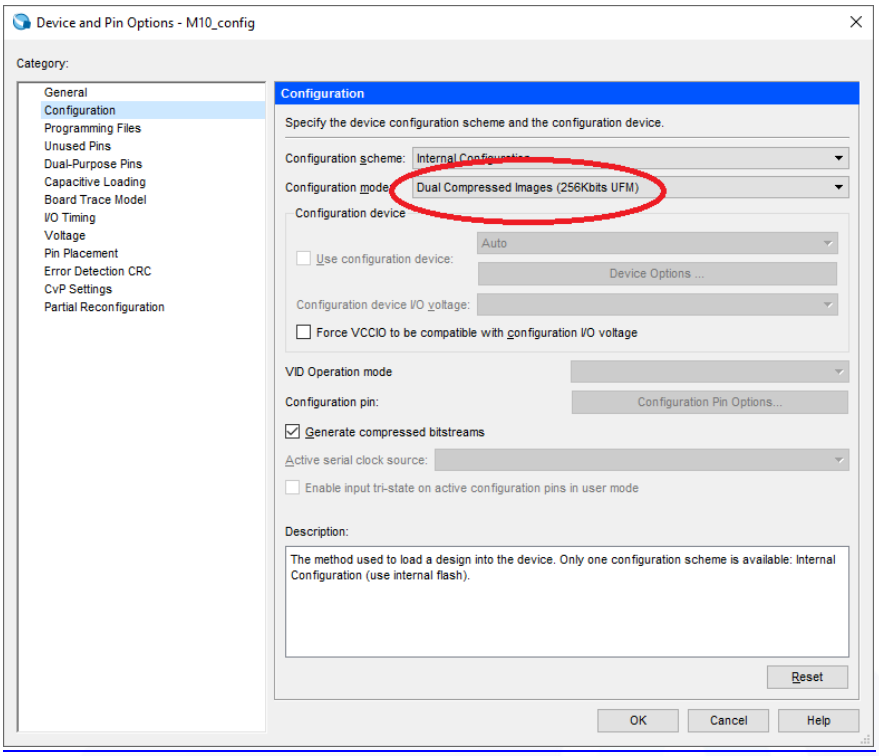

Figure 2-1 Device and Pin Options

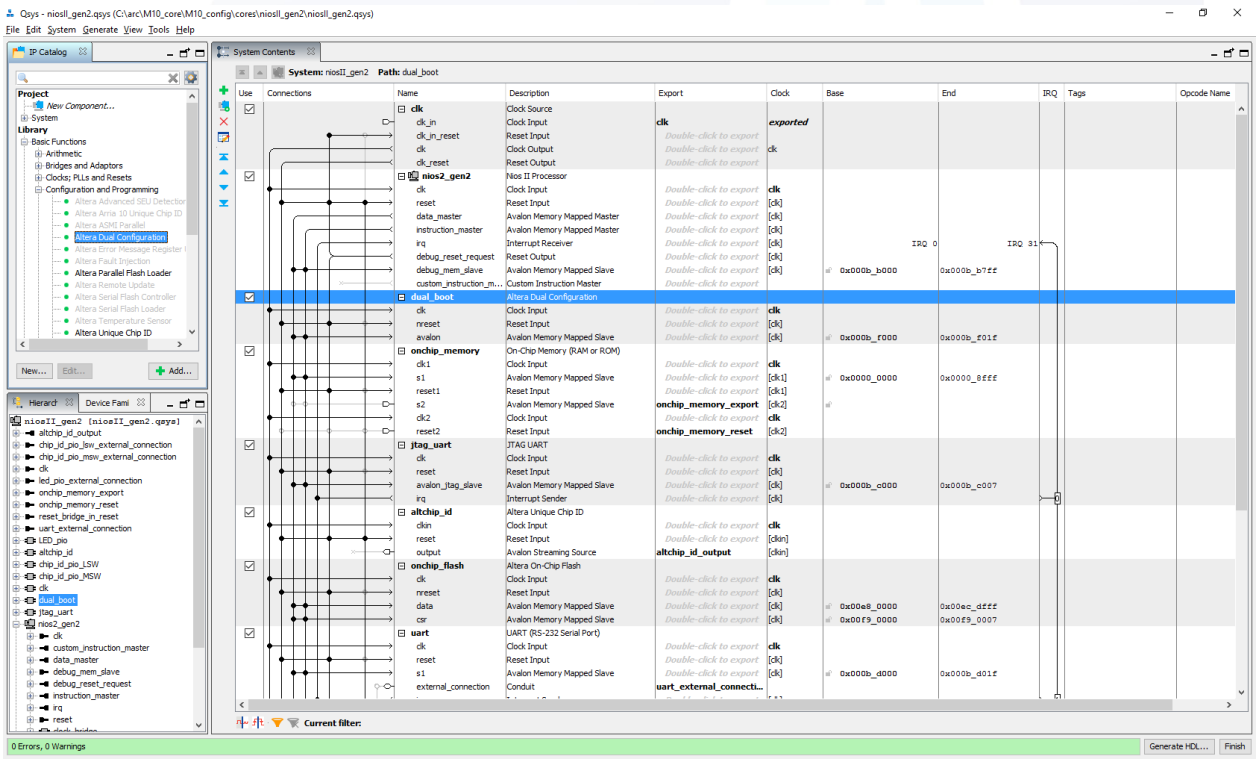

Figure 2-2 An Example Qsys that has Altera Dual Boot

2.2 Prepare pof file and rpd file

For a MAX® 10 FPGA in its out-of-box state, the initial flash images have to be programmed through JTAG. This usually happens during production stage, where the FPGA has to be programmed with a fast pace. And in production, the JTAG can be accessed through a breakout board or an extension cable.

A single pof file that contains everything can be prepared for this. And when this single pof file is generated, two rpd files can also be generated at the same time. These rpd files can be used for in-field upgrade as well.

To produce the single pof file and the rpd files, the following steps can be followed in Quartus:

1. Choose the menu "**File/Convert Programming Files...**"
2. In the "Convert Programming File" dialog,
    a. Set the Mode to be "**Internal Configuration**".
    b. Set the File name to the proper path and name.
    c. Turn on the option for "**Create config data RPD**"
    d. At the bottom of the dialog is the "Input files to convert". Click "Add Files..." to add "**M10_high_speed_config.sof** " (provided with M10 High Speed Configuration) to Page_0. After that, click "Add Sof Page" to add "SOF Data, Page_1". Highlight the "Page_1", and click "Add Files..." again to add the pof file generated by the main image project. (It is the Mustang_fast.sof for the example below in Figure 2-3.

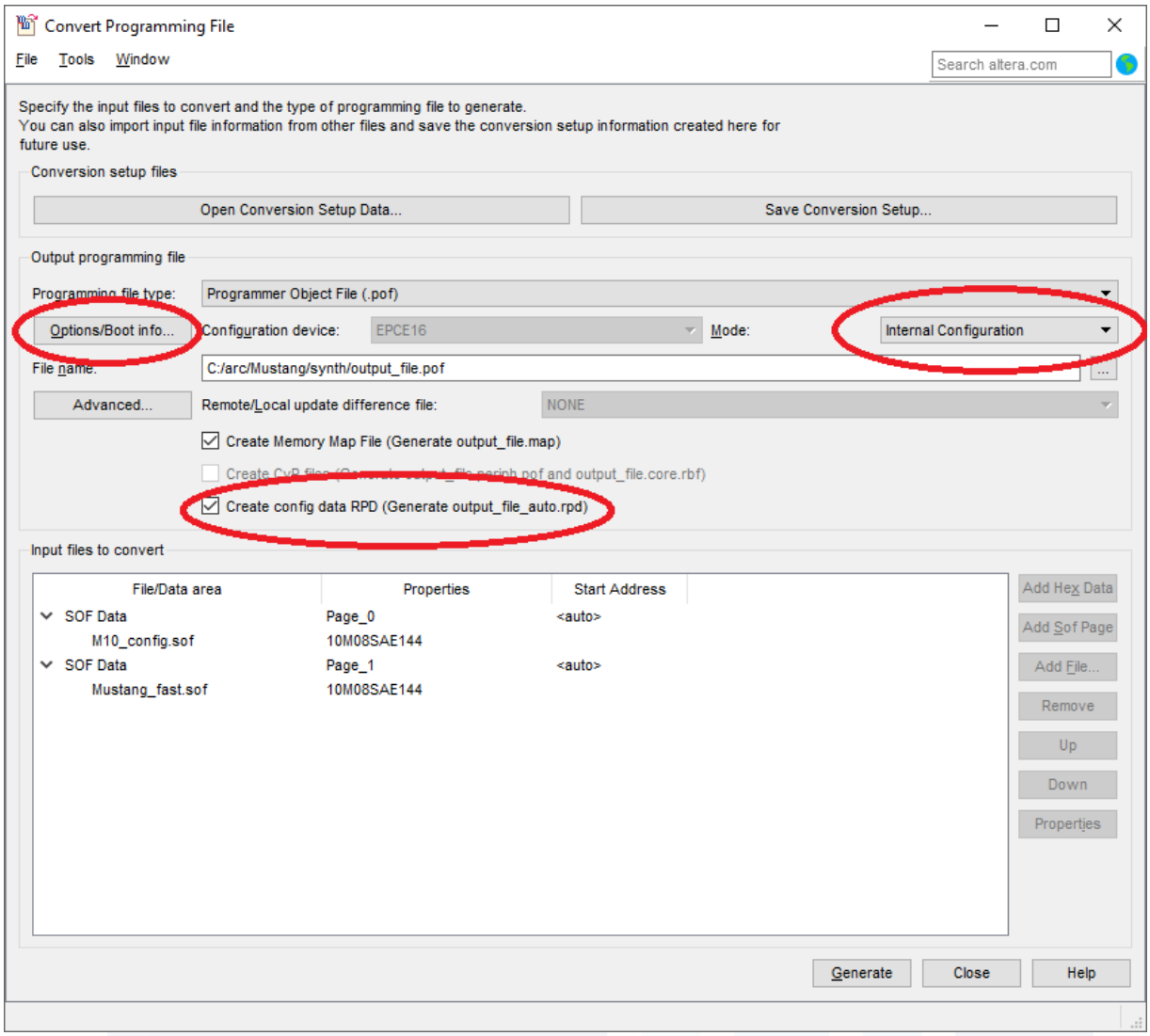

Figure 2-3 Convert Programming File

e. If UFM also needs to be programmed in the same pof file, please click "Options/Boot info" on the left side in Figure 2-3. After that, in the "Max 10 Device Options" dialog, please set "UFM source" as "Load memory file", and point the "File path" to the correspondent Intel hex file, as demonstrated in Figure 2-4. Click "Ok" to close the "Max 10 Device Options" dialogue after everything is configured properly.

f. Click the "Generate" button in Figure 2-3 to produce the single pof file needed for JTAG programming. If everything goes smooth, there will be two prd files generated in the same folder with pof file. The rpd file with the index 1 is for the main image. (Index 0 is for the boot image). And the rpd file of the main image can be used by M10 High Speed Configuration Utility for in-field upgrade.

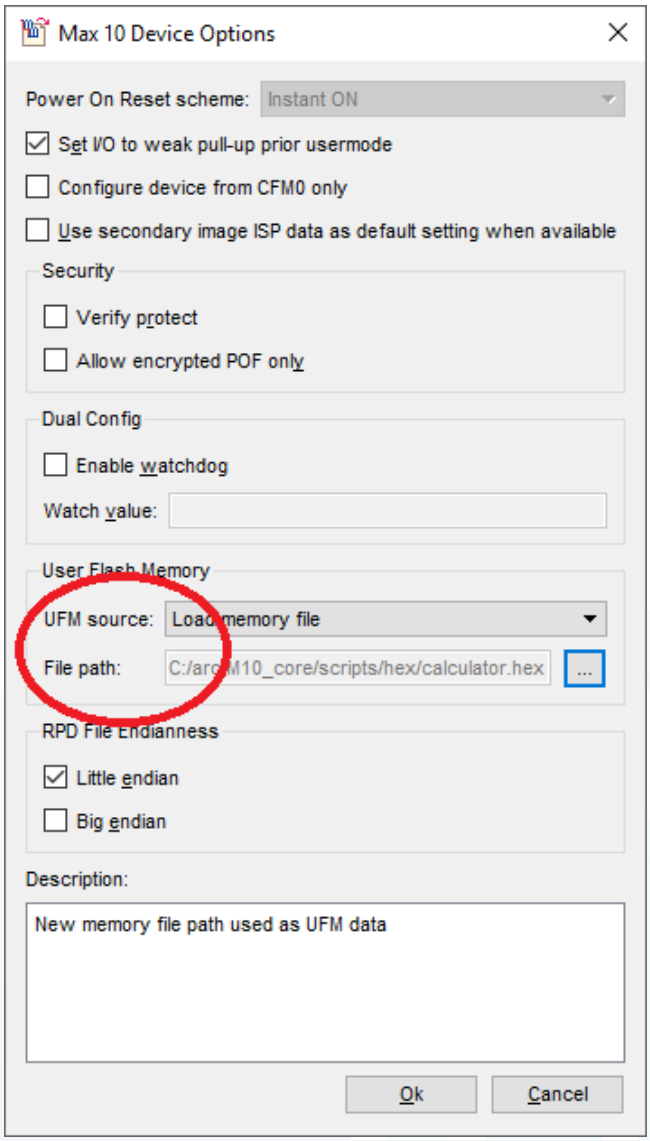

Figure 2-4 Add Intel Hex file to program the UFM

2.3 Program the blank FPGA with pof File

From previous sections, the single pof file produced by "Convert Programming File" can be used to program the FPGA's flash memory through JTAG. As demonstrated in Figure 2-5, the Quartus Prime Programmer is programming CFM0 (boot image), CFM1 (main image) and UFM with a single pof file. This is usually the initial step in production to fill out the blank flash for a new MAX® 10 FPGA device. And in production, the JTAG might be accessed through a breakout board or an extension cable.

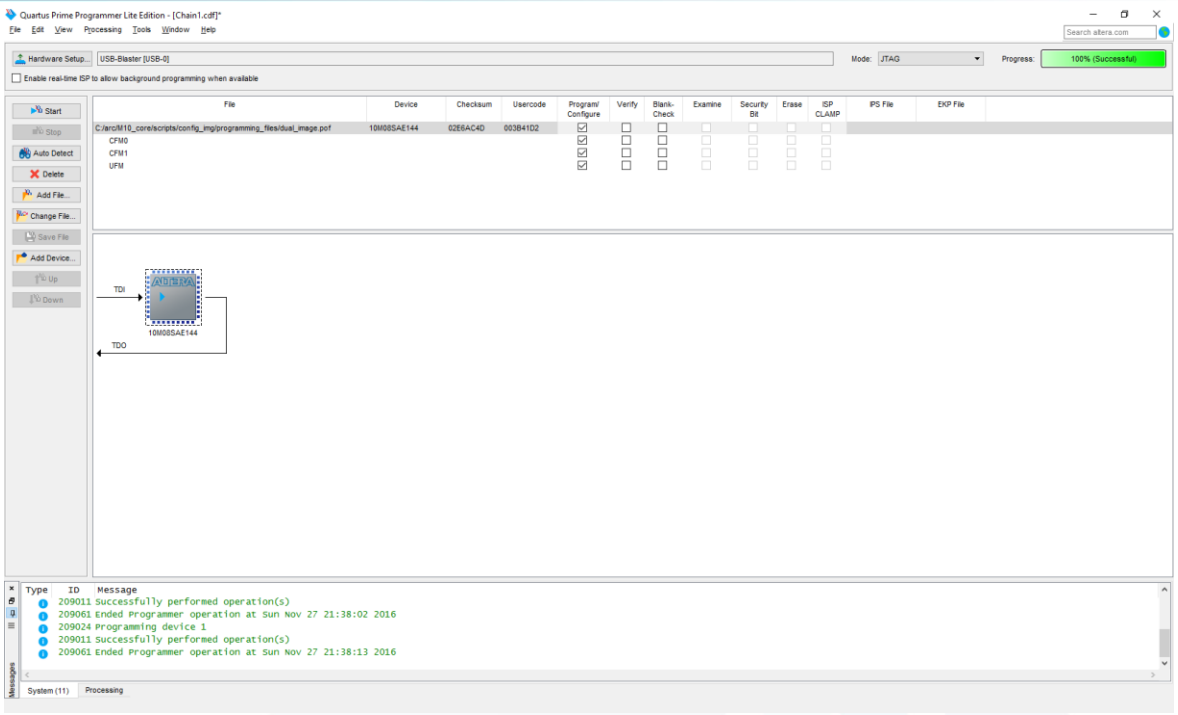

Figure 2-5 Program the FPGA's Flash through JTAG

2.4 In-field Upgrade with M10 Configuration Utility

After the product is deployed to the field, it is expected to operate in normal mode with the main image being active every time. Later if an in-field upgrade is required, the new FPGA/firmware can be downloaded to the MAX® 10 FPGA with the assistance of M10 High Speed Configuration Utility.

To use M10 Configuration Utility, the following steps can be followed:

1. Power off the device. Set the jumper on CONFIG_SEL pin to let CFM0 (the boot image) be the active sector.
2. Connect the USB cable (or RS232 cable, depending on the board design) to PC, and power on the device. Now the boot image becomes active inside the FPGA, and its internal bootstrapper will actively listen to the USB/UART port.
3. Open the utility "M10 High Speed Config Utility" (M10_config_gui) on host PC. The source code of this utility can be found in https://github.com/PulseRain/M10_high_speed_config_software
   And its latest binary can be downloaded from
   https://github.com/PulseRain/M10_high_speed_config_software/raw/master/bin/M10_config_gui.exe

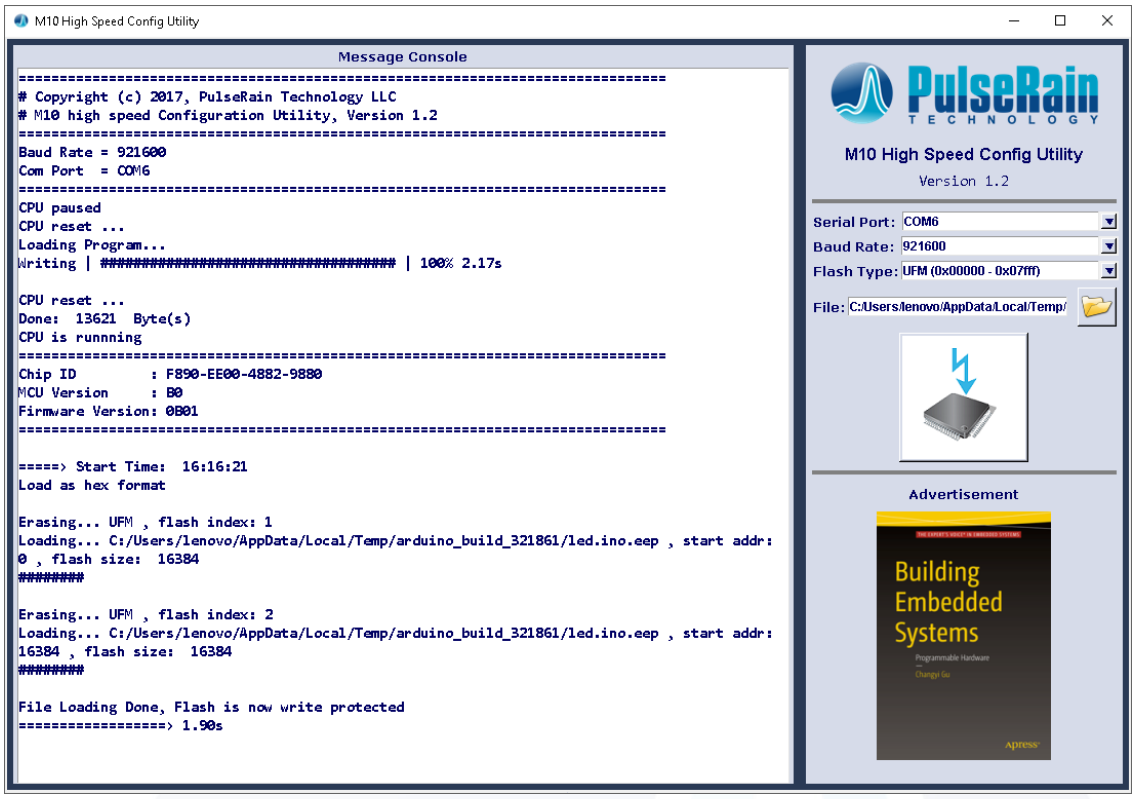

Figure 2-6 M10 High Speed Configuration Utility

4. After the utility is launched, setup the Serial Port and baud rate (921600 bps by default) accordingly. To program a new FPGA image, select CFM in the flash type. To program the UFM (user flash memory), set the flash type as UFM.
5. For CFM, point the File path to the binary file generated by the Quartus Tool. For UFM, point the File path to the hex or binary file produced by the compiler/linker (or other specific tools).
6. Click the big square button on the right-hand side (with the picture of lightning strike). If everything is done right, you will see a message like "***File Loading Done, Flash is now write protected***." at the end of flash programming, as illustrated in Figure 2-6.
7. Power off the device. Set the jumper on CONFIG_SEL pin to let CFM1 (the main image) be the active sector.
8. Power on the device again, And the device is supposed to load the new image at this point.

3 Repository

The RTL code (including the sketch/firmware) of the M10 High Speed Configuration can be found in the following repository on GitHub:

https://github.com/PulseRain/M10_high_speed_config_rtl

The Python script of the M10 High Speed Utility can be found in the following repository on GitHub:

https://github.com/PulseRain/M10_high_speed_config_software