T-Core

USER MANUAL

terasIC
www.terasic.com

CONTENTS

terasIC
www.terasic.com

T-Core
User Manual

1

www.terasic.com
September 24, 2020

Introduction

The T-Core presents a robust hardware design platform built around the Intel MAX 10 FPGA. The MAX 10 FPGA is well equipped to provide cost effective, single-chip solutions in control plane or data path applications and industry-leading programmable logic for ultimate design flexibility. With MAX 10 FPGA, you can get lower power consumption / cost and higher performance. When you need high-volume applications, including protocol bridging, motor control drive, analog to digital conversion, and handheld devices, the T-Core MAX 10 FPGA is your best choice.

The T-Core development board includes hardware such as on-board USB Blaster II, QSPI Flash, ADC-Header, WS2812B RGB LEDs and 2x6 TMD expansion header. By leveraging all of these capabilities, the T-Core is the perfect solution for showcasing, evaluating, and prototyping the true potential of the Intel MAX 10 FPGA.

T-Core also supports RISC-V CPU with on-board JTAG debug, it is an ideal platform for learning RISC-V CPU design or embedded system design.

1.1 Package Contents

Figure 1-1 shows a photograph of the T-Core package.

1. T-Core board
2. Mini USB Cable
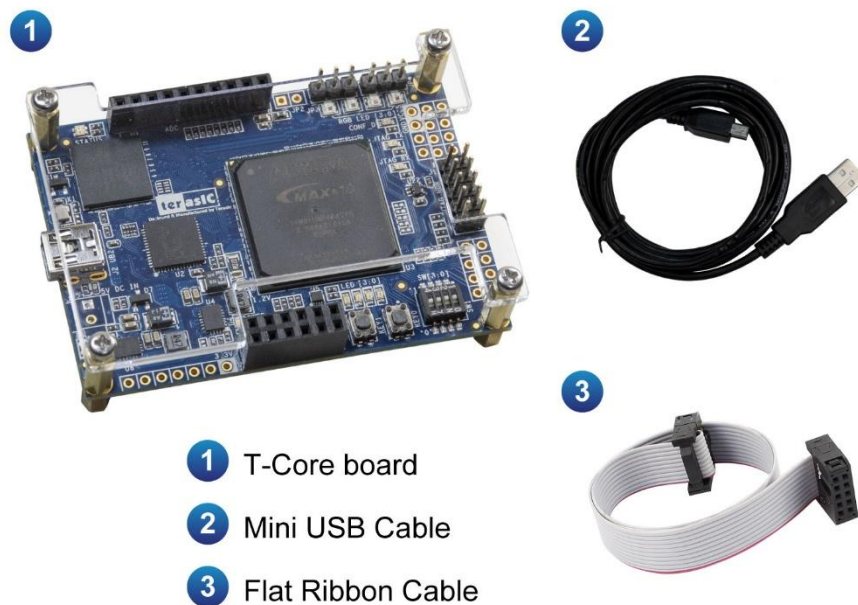3. Flat Ribbon Cable

Figure 1-1 The T-Core package contents

The T-Core package includes:

- The T-Core board

- USB Mini-B Cable

- Flat Ribbon Cable

1.2 T-Core System CD

The T-Core System CD contains the documentation and supporting materials, including the User Manual, reference designs and device datasheets.

User can download this System CD from the web (http://T-Core.terasic.com/cd).

1.3 Layout and Components

This section presents the features and design characteristics of the board.

A photograph of the board is shown in **Figure 1-2** and **Figure 1-3.** It depicts the layout of the board and indicates the location of the connectors and key components.
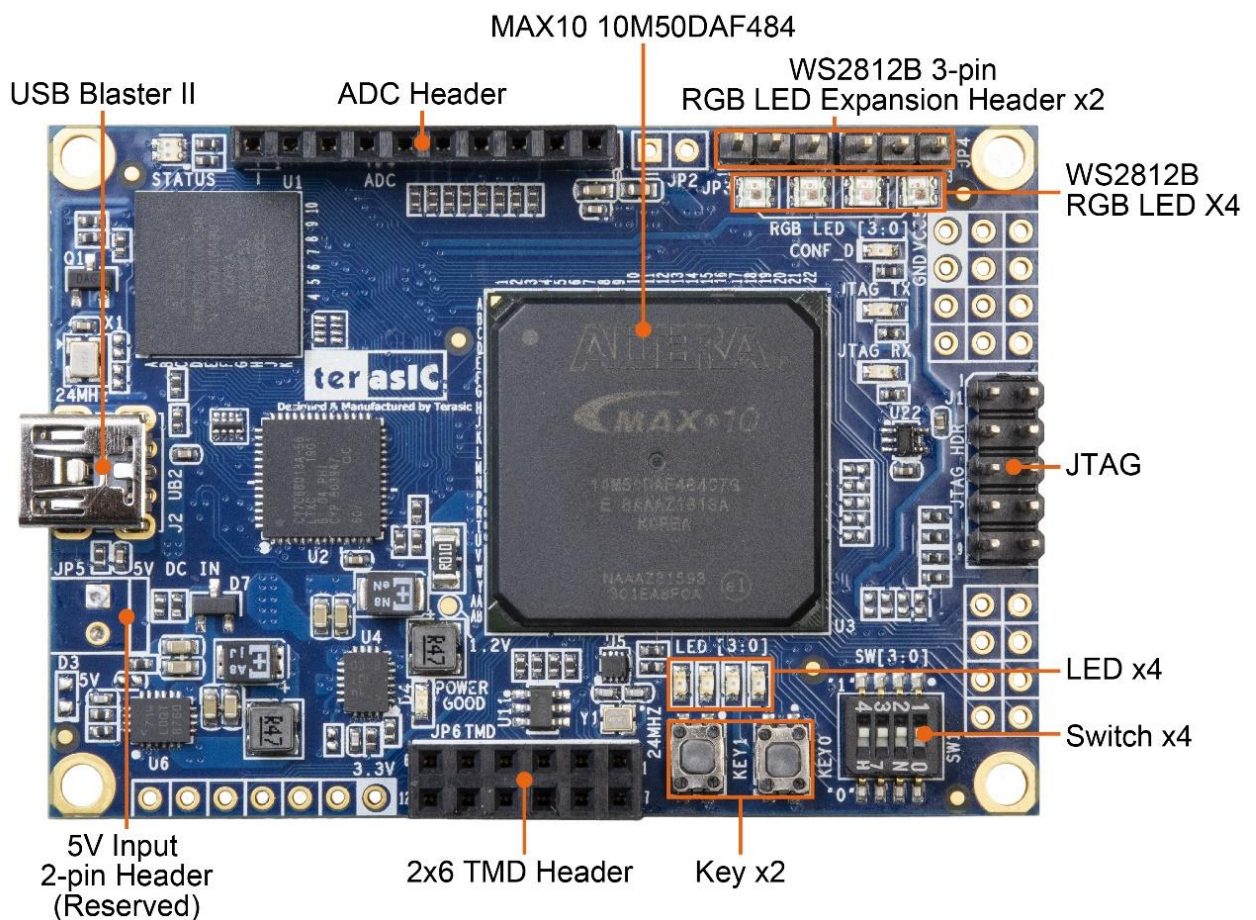

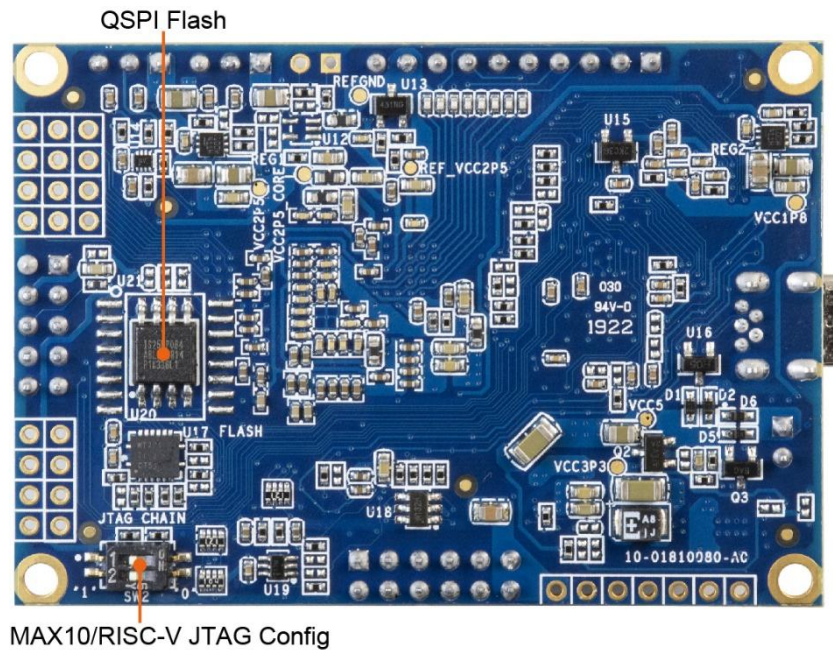**Figure 1-2 Development Board (top view)**

Figure 1-3 Development Board (bottom view)

This board has many features that allow users to implement a wide range of designed circuits, from simple circuits to various multimedia projects.

The following hardware are provided on the board:

FPGA Device

- MAX 10 10M50DAF484C7G Device
- Integrated dual ADCs, each ADC supports 1 dedicated analog input and 8 dual function pins
- 50K programmable logic elements
- 1,638 Kbits M9K Memory
- 5,888 Kbits user flash memory
- 144 18 × 18 Multiplier
- 4 PLLs

Programming and Configuration

- On-Board USB Blaster II (Mini USB connector)

Memory Device

- QSPI Flash (Support RISC-V)

Connectors

- 2x6 TMD GPIO Header

- 1x10 ADC Header

- WS2812B RGB LED Expansion Header

Switches, Buttons and LEDs

- 4 Green LEDs

- 4 Dip Switches

- 2 Push Buttons with Debounced.

- 4 WS2812B RGB LED

Power

- 5V DC input from USB or external power connector.

1.4 Block Diagram of the Board

Figure 1-4 gives the block diagram of the board. To provide maximum flexibility for the user, all connections are made through the MAX 10 FPGA device. Thus, the user can configure the FPGA to implement any system design.
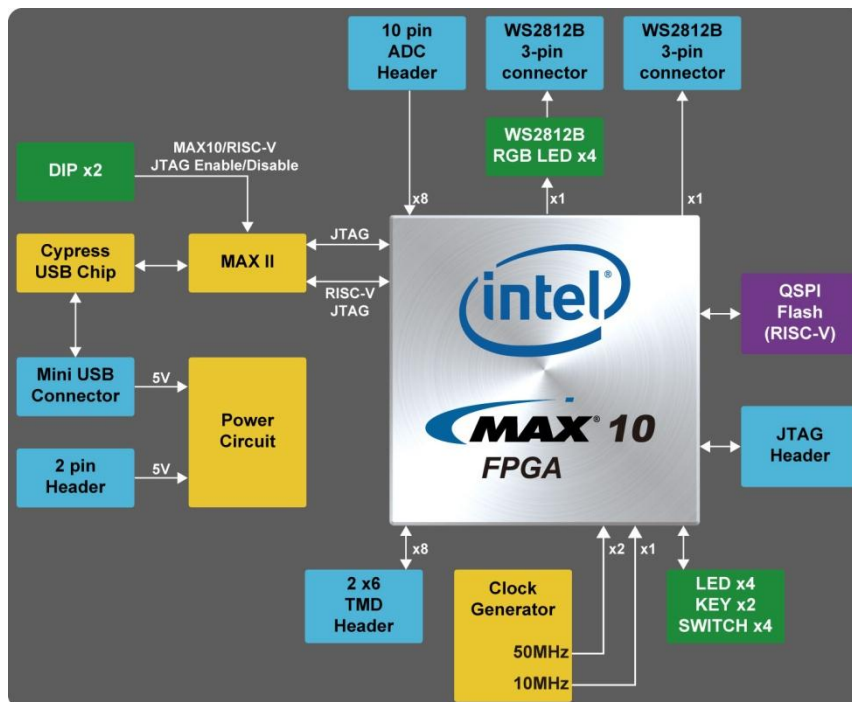

Figure 1-4 Board Block Diagram

1.5 Getting Help

Here are the addresses where you can get help if you encounter any problem:

- Terasic Inc.

- 9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu City, 30070. Taiwan

- Email: support@terasic.com

- Tel.: +886-3-5750-880

- Web: http://T-Core.terasic.com

Using the T-Core

This chapter provides instructions to use the board and the peripherals description.

2.1 Configuration of MAX 10 FPGA on T-Core

There are two types of configuration method supported by T-Core:

1. JTAG configuration: configuration using JTAG port.

JTAG configuration scheme allows you to directly configure the device core through JTAG pins - TDI, TDO, TMS, and TCK pins. The Quartus Prime software automatically generates .sof files that are used for JTAG configuration with a download cable in the Quartus Prime software program.

2. Internal configuration: configuration using internal flash.

Before internal configuration, you need to program the configuration data into the configuration flash memory (CFM) which provides non-volatile storage for the bit stream. The information is retained within CFM even if the T-Core board is turned off. When the board is powered on, the configuration data in the CFM is automatically loaded into the MAX 10 FPGA.

■ **JTAG Chain on T-Core Board**

Figure 2-1 shows the JTAG interface of T-Core board, which uses the USB Blaster II (UB2) circuit to connect to the host PC. The JTAG interface of the T-Core board can be divided into common FPGA configuration and RISC-V applications. Users can configure or debug the MAX10 FPGA on the T-Core through the UB2 interface and Quartus software on the host PC. You can also use the UB2 circuit to download the binary file from the host PC to RISC-V application. The two position switches on the SW2 can control whether the MAX10 FPGA and RISC-V application will appear in the JTAG chain of the T-Core board.
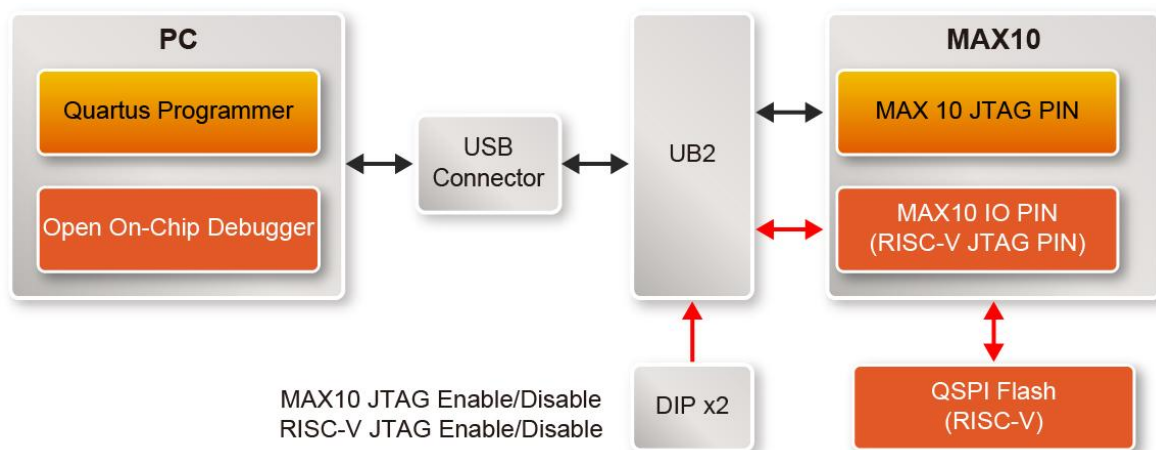
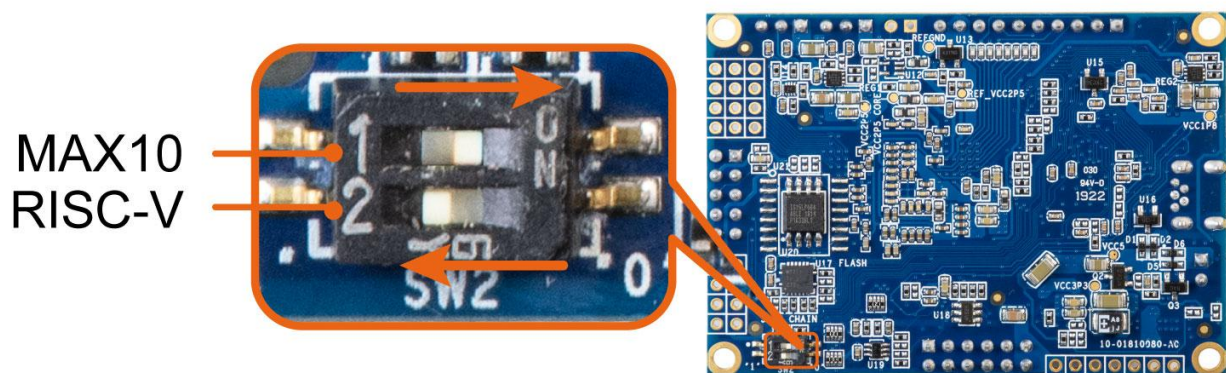**Figure 2-1 The JTAG configuration scheme**

Figure 2-2   The JTAG Switch for MAX10 FPGA and RISC-V

Figure 2-2 shows the factory settings of the SW2. The SW2.1 position is used for enable/disable the MAX 10 FPGA in the JTAG chain. When set to ON, the MAX10 FPGA will appear in the JTAG chain of T-Core. SW2.2 is the JTAG chain switch for RISC-V applications. If it is turned off, the application will not appear in the JTAG chain. For more information about the RISC-V application, please refer to section 3.5 for more information.

Table 2-1 is the setting of SW2 and its function.

Table 2-1 MAX10 and RISC-V JTAG Chain setting for SW2

SW2 Position		Description
MAX10 (SW2.1)	**RISC-V (SW2.2)**	
OFF	OFF	MAX10 and RISC-V off chain
OFF	ON	MAX10 off chain and RISC-V on chain
ON	OFF	MAX10 on chain and RISC-V off chain
ON	ON	MAX10 on chain and RISC-V on chain

■ Configure the FPGA in JTAG Mode

The following shows how the FPGA is programmed in JTAG mode step by step.

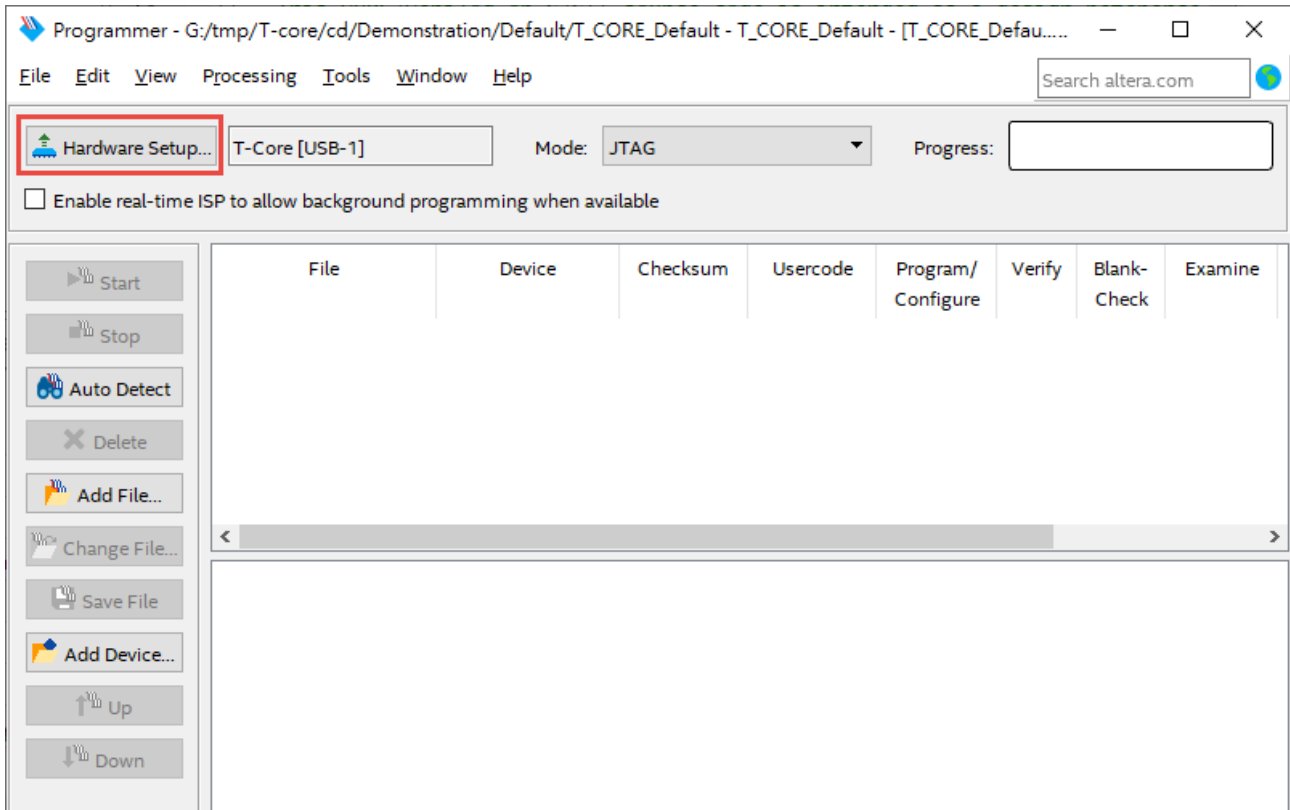1. Open the Quartus Prime programmer, please choose **Tools > Programmer**. The Programmer window opens. See **Figure 2-3**.

Figure 2-3 Programmer Window

2. Click "Hardware Setup", as circled in **Figure 2-3**.

3. If it is not already turned on, turn on the USB-Blaster [USB-0] option under currently selected hardware and click "Close" to close the window. See **Figure 2-4**.
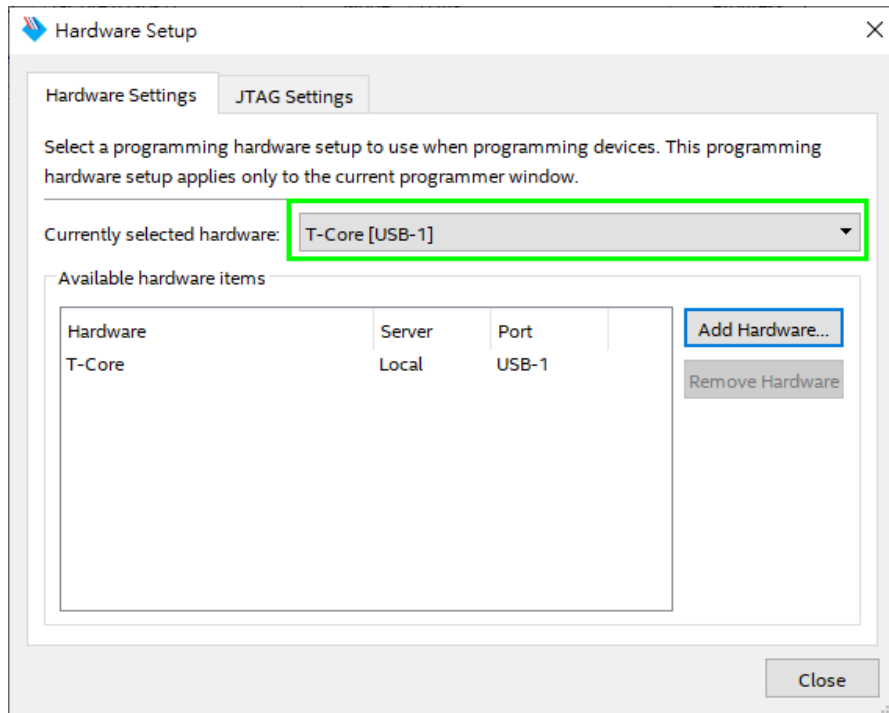
Figure 2-4 Hardware Setting

4.   Click "Auto Detect" to detect all the devices on the JTAG chain, as circled in **Figure 2-5**.
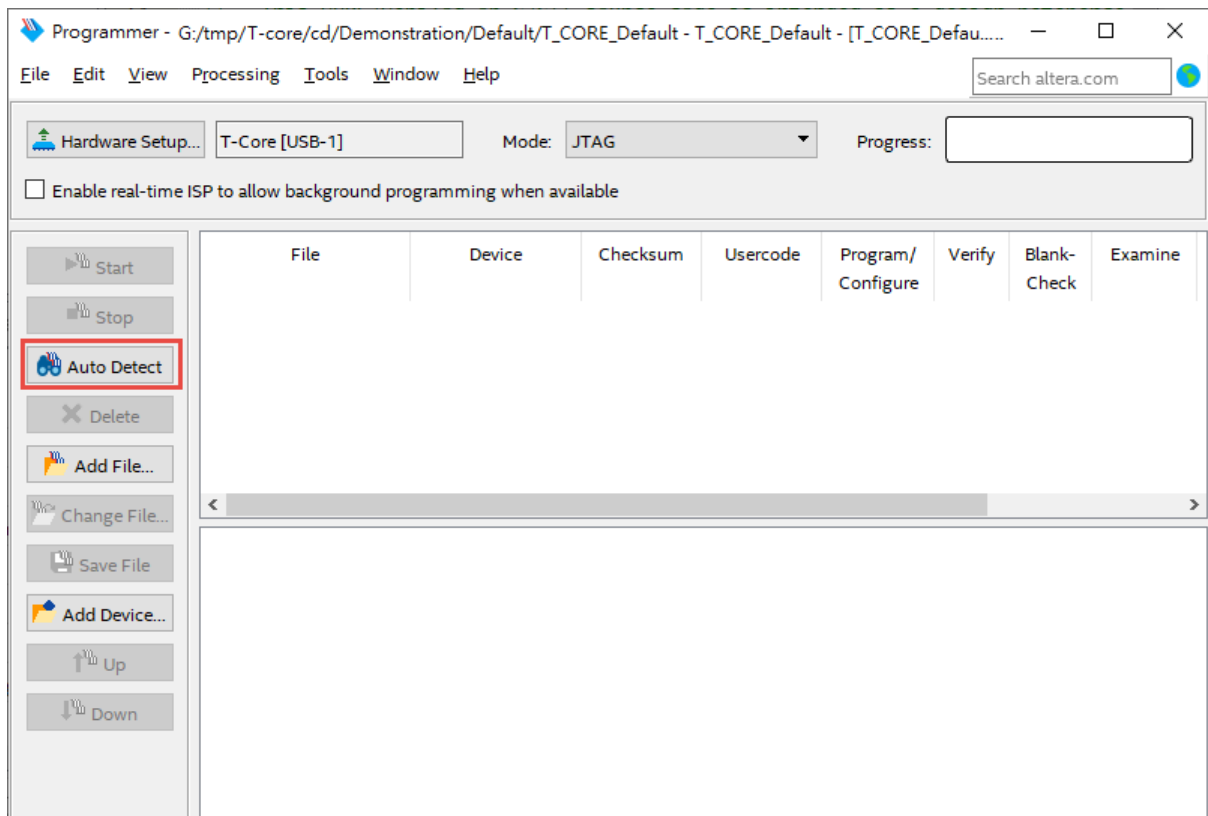
Figure 2-5 Detect FPGA device in JTAG mode

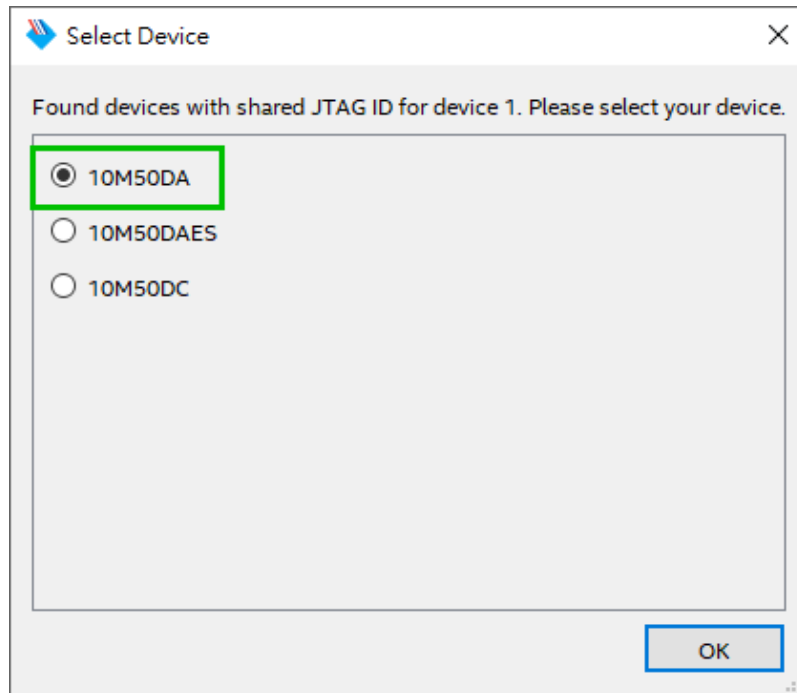5.   Select detected device associated with the board, as circled in **Figure 2-6**.

Figure 2-6 Select 10M50DA device
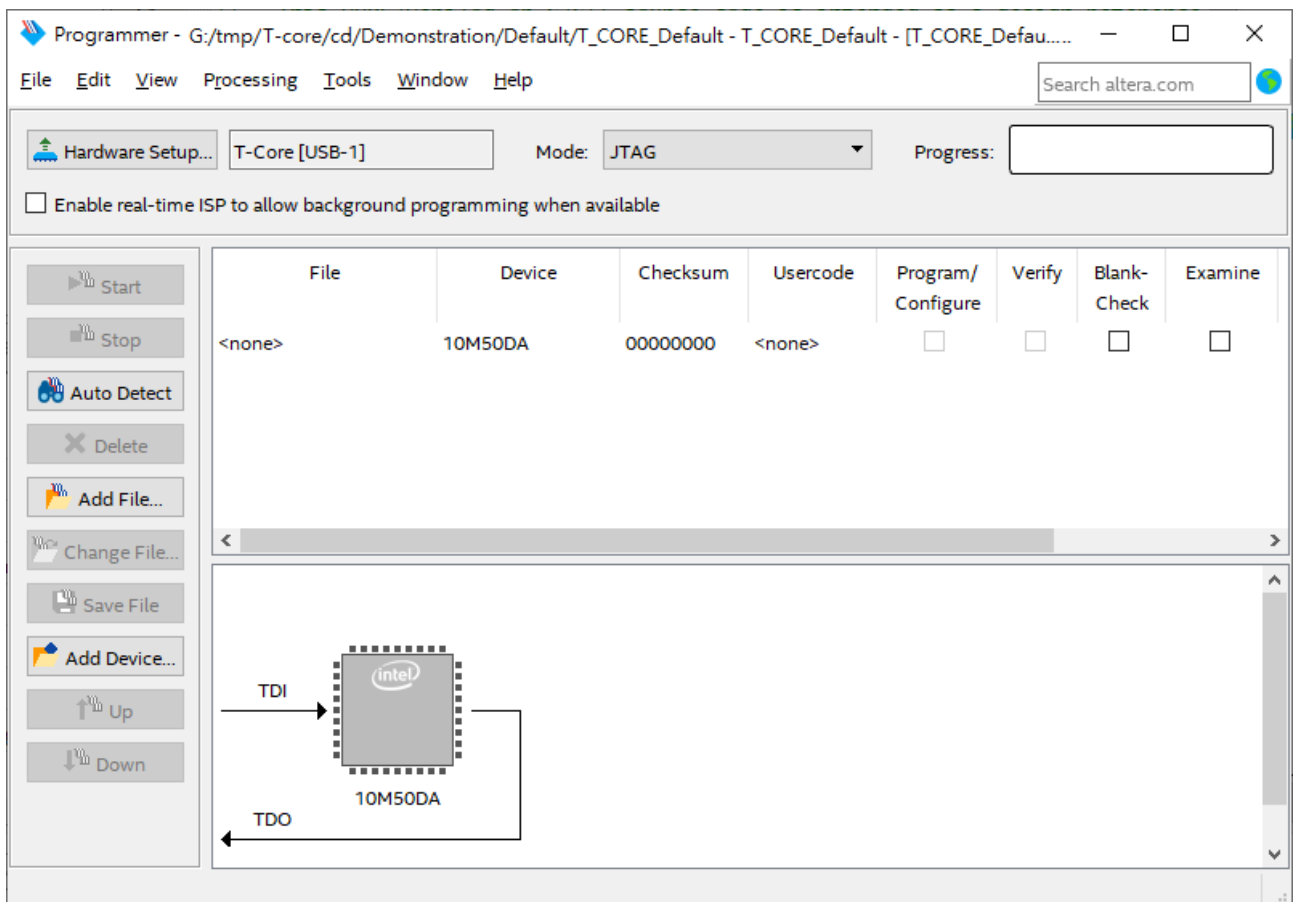
6. FPGA is detected, as shown in **Figure 2-7**.

Figure 2-7 FPGA detected in Quartus Prime programmer

7. Right click on the FPGA device and click "Change File" to open the .sof file to be programmed, as highlighted in **Figure 2-8**.
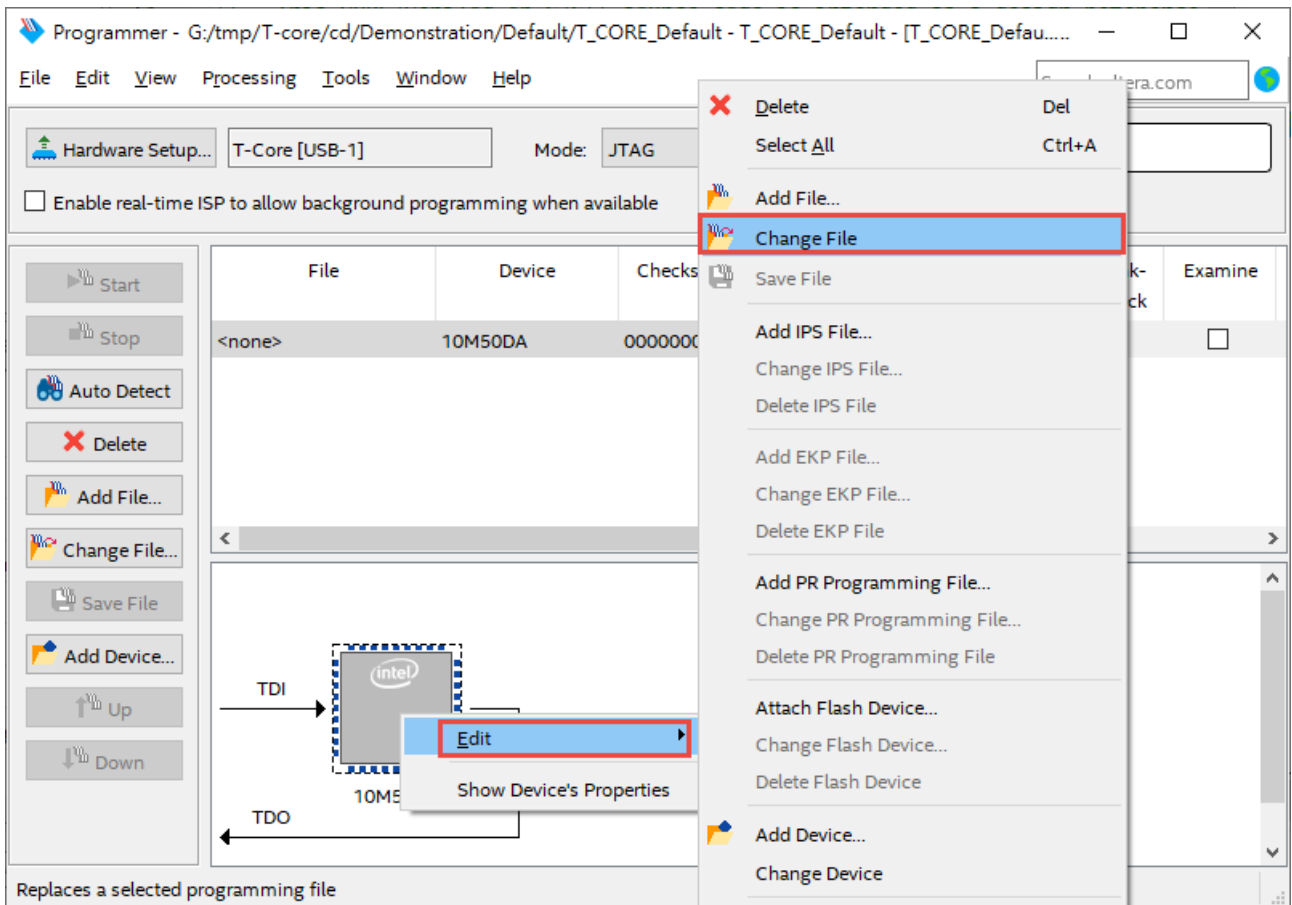

Figure 2-8 Open the .sof file to be programmed into the FPGA device

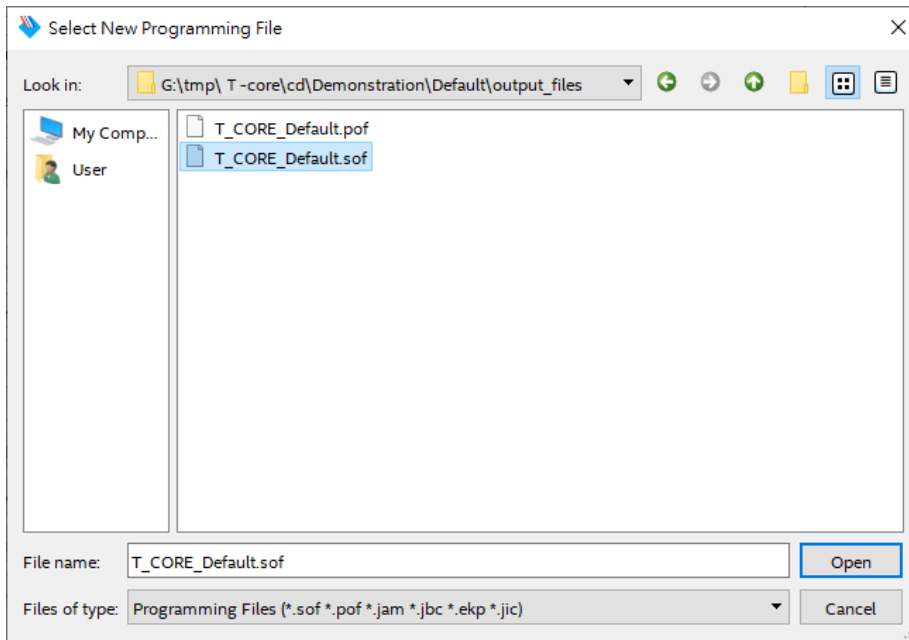**8.** Open the "output_files" folder and select the .sof file to be programmed, as shown in **Figure 2-9**.

Figure 2-9 Select the .sof file to be programmed into the FPGA device

9. Click "Program/Configure" check box and then click "Start" button to download the .sof file into the FPGA device, as shown in **Figure 2-10**.
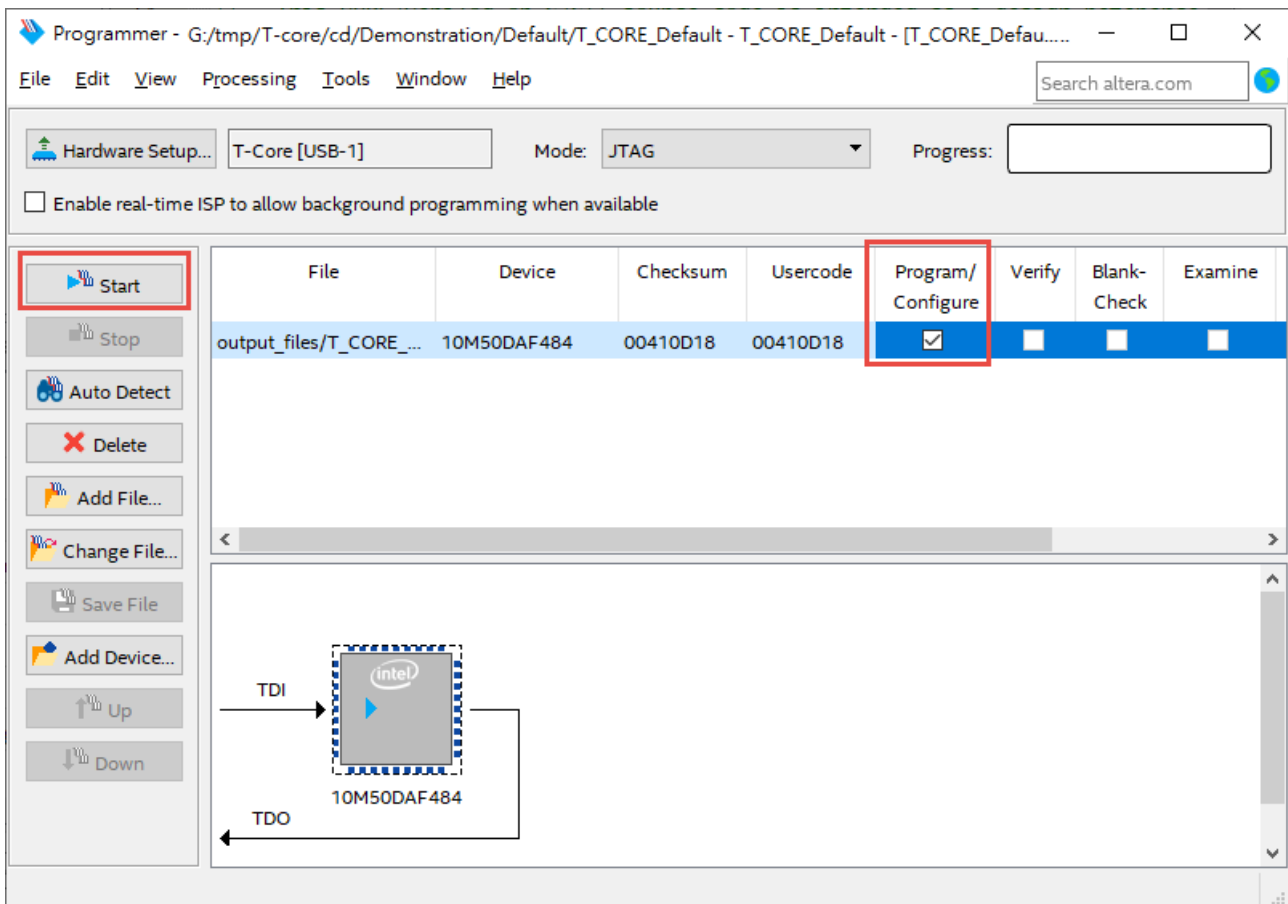

Figure 2-10 Program .sof file into the FPGA device

◼ Internal Configuration

• The configuration data to be written to CFM will be part of the programmer object file (.pof). This configuration data is automatically loaded from the CFM into the MAX 10 devices when the board is powered up.

• Please refer to **Chapter 4: Programming the Configuration Flash Memory (CFM)** for the basic programming instruction on the configuration flash memory (CFM).
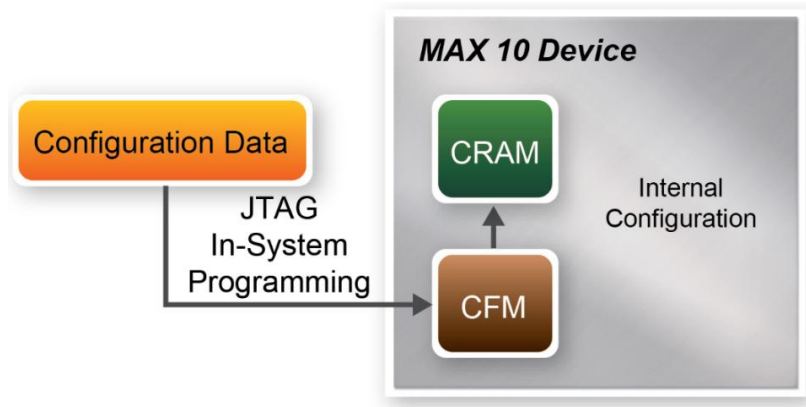

Figure 2-11 High-Level Overview of Internal Configuration for MAX 10 Devices

◼ Status LED

The T-Core development board includes board-specific status LEDs to indicate board status. Please refer to **Table 2-2** for the description of the LED indicator. Please refer to **Figure 2-12** for detailed LED location.
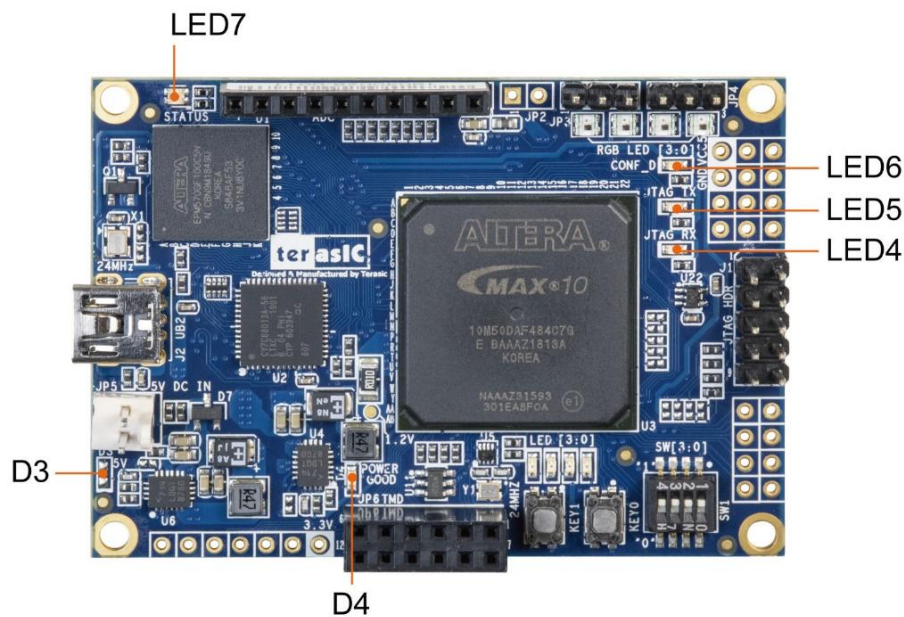

Figure 2-12 Status LED position

Table 2-2 Status LED

Reference	LED Name	Description
LED4	JTAG RX	Illuminate when data is transferred from USB Host to JTAG.
LED5	JTAG TX	Illuminate when data is transferred from JTAG to USB Host.
LED6	CONF_DONE	Illuminates when the FPGA is successfully configured.
LED7	Status	Illuminates Bard status
D3	5V	Illuminates when Input power is active. Not Installed.
D4	Power Good	Illuminates when board power system is OK.

2.2 Clock Circuitry

Figure 2-13 shows the default frequency of all external clocks to the MAX 10 FPGA. A clock generator is used to distribute clock signals with low jitter. The two 50MHz clock signals connected to the FPGA are used as clock sources for user logic. One 24MHz clock signal is connected to the clock inputs of USB microcontroller of USB Blaster. One 10MHz clock signal is connected to the PLL1 and PLL3 of FPGA, the outputs of these two PLLs can drive ADC clock. The associated pin assignment for clock inputs to FPGA I/O pins is listed in **Table 2-3.**

Warning!!

Do not modify the clock generator settings.

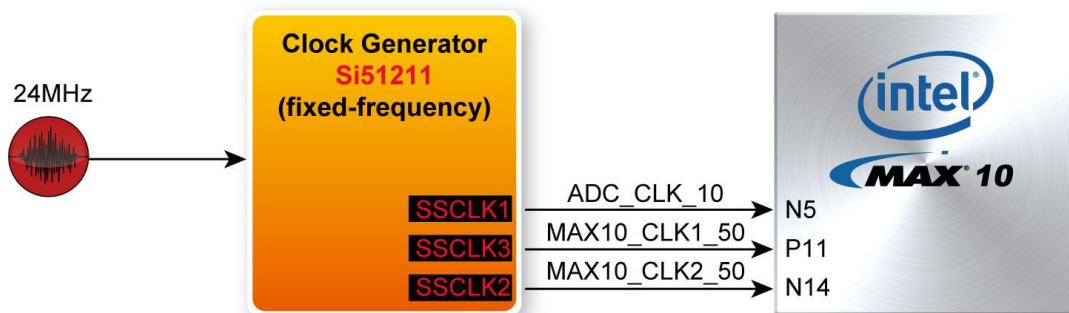*Incorrect setting will cause the system to not work.*

Figure 2-13 Clock circuit of the FPGA Board

Table 2-3 Pin Assignment of Clock Inputs

Signal Name	FPGA Pin No.	Description	I/O Standard
ADC_CLK_10	PIN_N5	10 MHz clock input for ADC (Bank 3B)	3.3-V LVTTL
MAX10_CLK1_50	PIN_P11	50 MHz clock input(Bank 3B)	3.3-V LVTTL
MAX10_CLK2_50	PIN_N14	50 MHz clock input(Bank 3B)	3.3-V LVTTL

2.3 Using the Push-buttons, Switches and LEDs

■ **User-Defined Push-buttons**

The board includes two user defined push-buttons that allow users to interact with the MAX 10 FPGA device. Each of these switches is debounced using a Schmitt Trigger circuit as indicated in **Figure 2-14**. A Schmitt trigger feature introduces hysteresis to the input signal for improved noise immunity, especially for signal with slow edge rate and act as switch debounce in **Figure 2-15** for the push-buttons connected. **Table 2-4** list the pin assignment of user push-buttons.
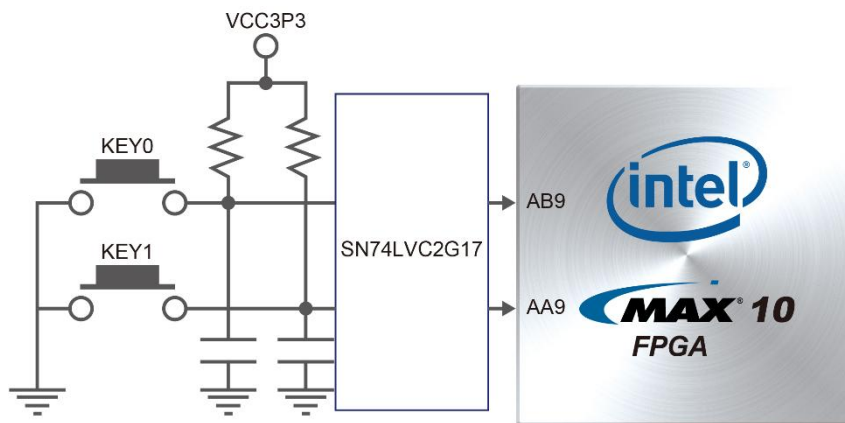
Figure 2-14 Connections between the push-button and MAX 10 FPGA
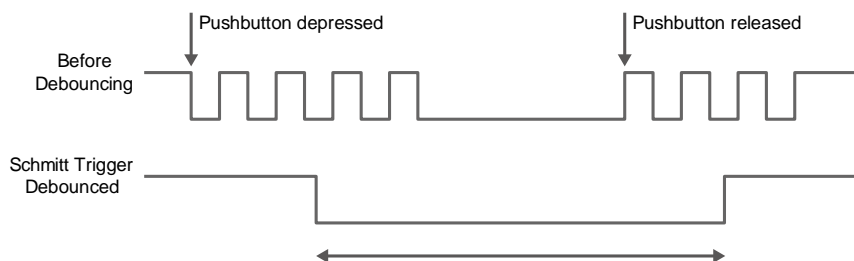
Figure 2-15   Switch debouncing

Table 2-4 Pin Assignment of Push-buttons

Signal Name	FPGA Pin No.	Description	I/O Standard
KEY[0]	PIN_AB9	Push-button[0]	3.3 V SCHMITT TRIGGER"
KEY[1]	PIN_AA9	Push-button[1]	3.3 V SCHMITT TRIGGER"

■ **User-Defined Dip Switch**

There are 4 position dip switches connected to FPGA on the board (See **Figure 2-16**). These switches are used as level-sensitive data inputs to a circuit. Every one of the switches is connected directly and individually to a pin on the MAX 10 FPGA. When the switch is in the ON position

(closest to the edge of the board), it provides a low logic level to the FPGA, and when the switch is in another position, it provides a high logic level. **Table 2-5** list the pin assignments of the user switches.
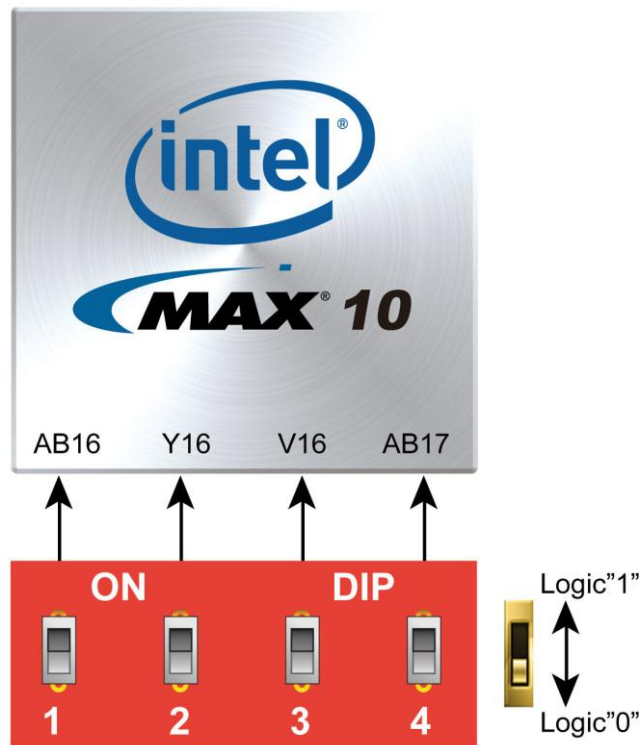

Figure 2-16   Connections between the slide switches and MAX 10 FPGA

Table 2-5 Pin Assignment of Slide Switches

Signal Name	FPGA Pin No.	Description	I/O Standard
SW[0]	PIN_AB16	Dip Switch[0]	3.3-V LVTTL
SW[1]	PIN_Y16	Dip Switch[1]	3.3-V LVTTL
SW[2]	PIN_V16	Dip Switch[2]	3.3-V LVTTL
SW[3]	PIN_AB17	Dip Switch[3]	3.3-V LVTTL

■ **User-Defined LEDs**

There are also four user-controllable LEDs connected to FPGA on the board. Each LED is driven directly and individually by a pin on the MAX 10 FPGA; driving its associated pin to a high logic level turns the LED on, and driving the pin low turns it off. **Figure 2-17** is the connections between LEDs and MAX 10 FPGA. **Table 2-6** list the pin assignment of user LEDs.
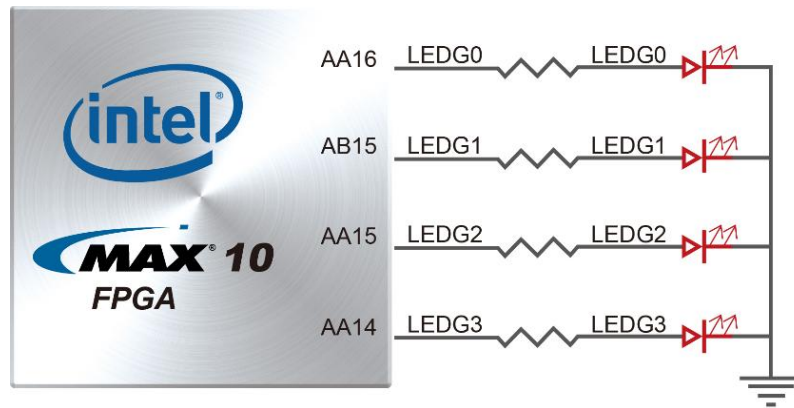
Figure 2-17 Connections between the LEDs and MAX 10 FPGA

Table 2-6 Pin Assignment of LEDs

Signal Name	FPGA Pin No.	Description	I/O Standard
LED[0]	PIN_AA16	LED [0]	3.3-V LVTTL
LED[1]	PIN_AB15	LED [1]	3.3-V LVTTL
LED[2]	PIN_AA15	LED [2]	3.3-V LVTTL
LED[3]	PIN_AA14	LED [3]	3.3-V LVTTL

2.4 Using WS2812B RGB LED

The board has Built-in the four Color LED (WS2812B) with intelligent controller (See **Figure 2-18**) and the four LED are connected in series controlled by one GPIO line. The WS2812B is an intelligent control LED light source that the control circuit and RGB chip are integrated in a package of 5050 components. It internal includes intelligent digital port data latch and signal reshaping amplification drive circuit. Also includes a precision internal oscillator and a 12V voltage programmable constant current control part, effectively ensuring the pixel point light color height consistent.
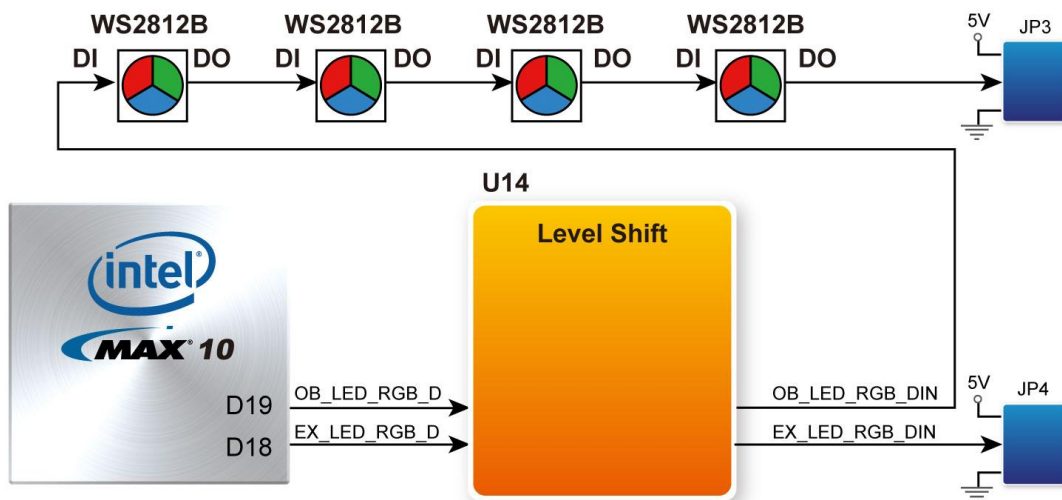
Figure 2-18 Level Shift between the REG LEDs and MAX 10 FPGA

■ Control Method

For the WS281B RGB LED, each R/G/B color can be independently controlled, and each color can be adjusted from level 0 to 255. Therefore, each color needs to use 8bit control, and each LED needs 24bit control. The 24 bits corresponds to the **Figure 2-19**.

Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Note: Follow the order of GRB to sent data and the high bit sent at first.

Figure 2-19 Composition of 24bit data

The logic 1 or 0 of each bit is determined as shown in **Figure 2-20**. It is determined by the length of the voltage level in a period. Detailed specifications can refer to **Table 2-7**.
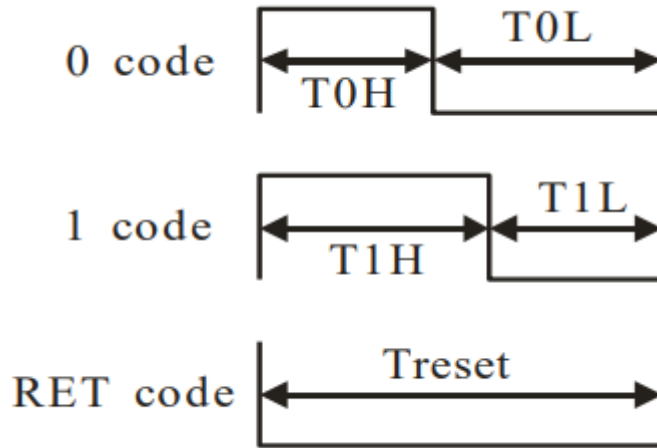
Figure 2-20 Sequence Chart

Table 2-7 Code Timing

Code Status	Description	Timing
T0H	0 code, high voltage time	220ns~380ns
T1H	1 code, high voltage time	580ns~1µs
T0L	0 code, low voltage time	580ns~1µs
T1L	1 code, low voltage time	220ns~420ns
RES	Frame unit, low voltage time	>280µs

The data transfer protocol of the WS281 use single NRZ communication mode. As shown in **Figure 2-21**, after the pixel power-on reset, the DI port receive data from controller, the first pixel collect initial 24bit data then sent to the internal data latch, the other data which reshaping by the internal signal reshaping amplification circuit sent to the next cascade pixel through the DO port. After transmission for each pixel, the signal to reduce 24bit. Pixel adopt auto reshaping transmit technology, making the pixel cascade number is not limited the signal transmission, only depends on the speed of signal transmission.

Cascade method:
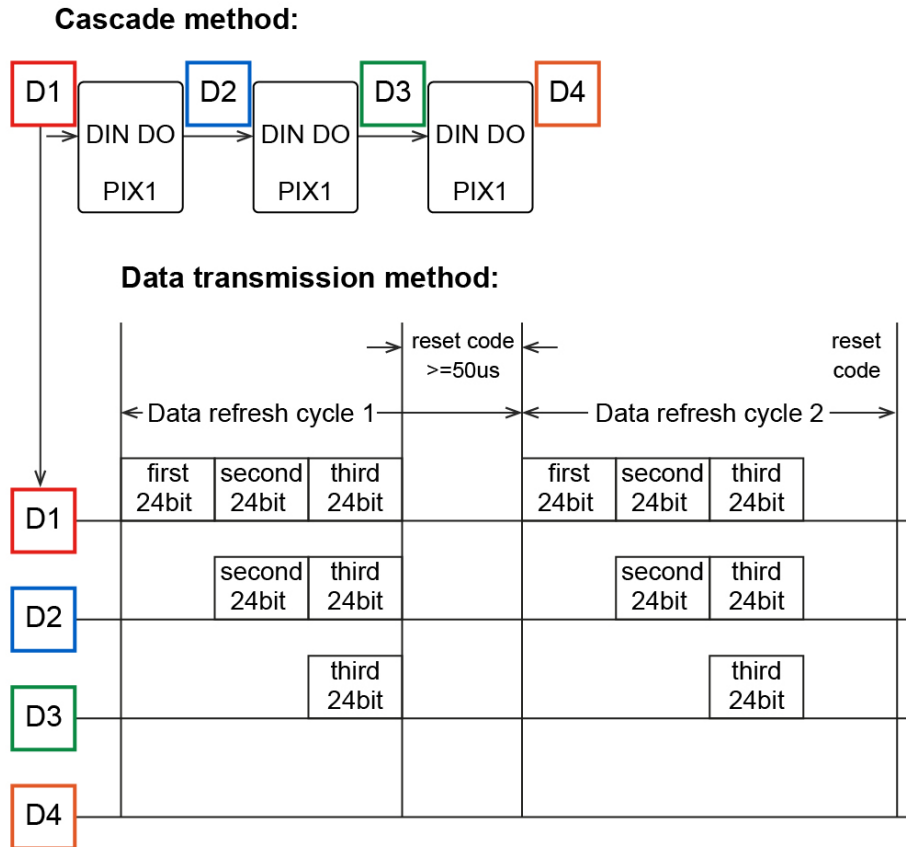
Figure 2-21 Data Transmission Method

The DIN signal level of the WS2812B is 5V, so a Level Shift will be required to raise the control signal voltage to the 5V level. **Table 2-8** shows the pin assignment of RGB LED Control signal.

More information about the RGB Color LED is available in its datasheet. It can be found on manufacturer's website or in the directory \Datasheet\RGB_LED of system CD.

Table 2-8 Pin Assignment for RGB Color LED and Extension port

Signal Name	FPGA Pin No.	Description	I/O Standard
OB_LED_RGB_D	PIN_D19	On-Board RGB LED Data IN	3.3-V LVTTL
EX_LED_RGB_D	PIN_D18	RGB LED Data IN for External Header	3.3-V LVTTL

■ Expansion Port

In addition to the Built-in four RGB LEDs on the T-Core board, two 3-pin headers (JP3 and JP4) are reserved for user to expand more LED. Users can purchase the commercially RGB LED Line Pipe in series to make the application more widely available.

See **Figure 2-22**, if user expands more RGB LEDs on JP3, it must be noted that the control method should take into account that there are four on board LEDs on the front end. If you use JP4 to expand and connect more LEDs, you only need to consider the newly added LEDs for the controlling.

For these 3 pins header, detailed information is described as follows:

Pin 1 of the 3-pin header is a 5V power supply, but may not provide sufficient current for the extended LED strip (when connected a lot). Therefore, if the user needs to connect more than one LED strip, please do not connect the pin.1 of the 3-pin header, please supply it to the LED strip by an external 5V power supply.

Pin 2 of the header is a serial control signal with a voltage level of 5V. The user can define an arbitrary waveform to output.

Pin 3 is grounded. Please note that if the external light strip is attached, the ground of the light strip should be connected to this pin.

For detailed RGB LED and JP3/JP4 position, please refer to **Figure 2-22**, simple connection method, refer to **Figure 2-23.**
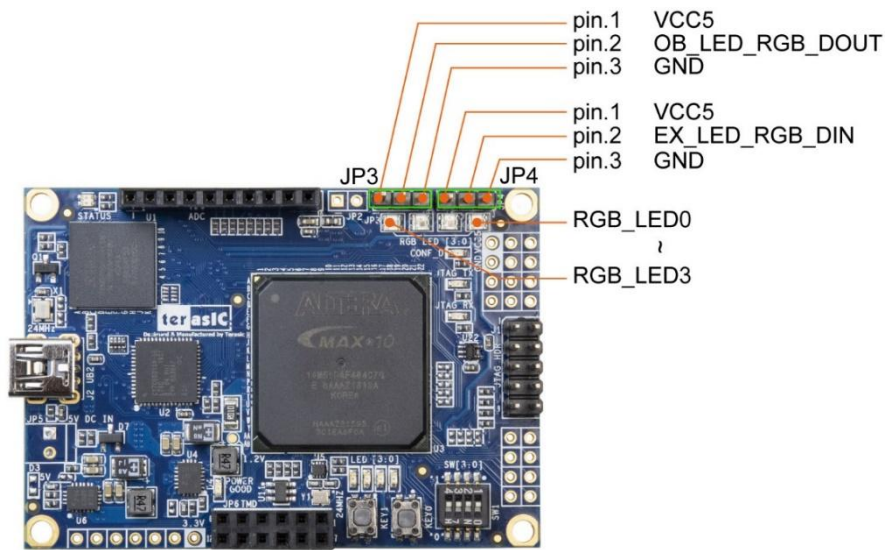
Figure 2-22 JP3 / JP4 and RGB LED Location
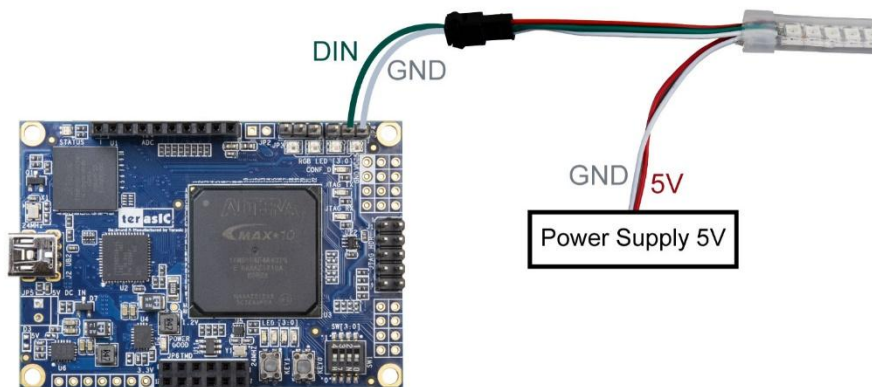
Figure 2-23 An example of a color light bar connection for an external power supply

2.5 Using 2x6 TMD GPIO Expansion Headers

The board has one 2x6 TMD (Terasic Mini Digital) expansion header. The TMD header has 8 digital GPIO user pins connected to the MAX 10 FPGA, two 3.3V power pins and two ground pins.

There are two Transient Voltage Suppressor diode arrays used to implement ESD protection for 8 GPIO user pins.

Figure 2-24 shows the connection between the TMD header and MAX 10 FPGA. **Table 2-9** shows the pin assignment of 2x6 TMD header.
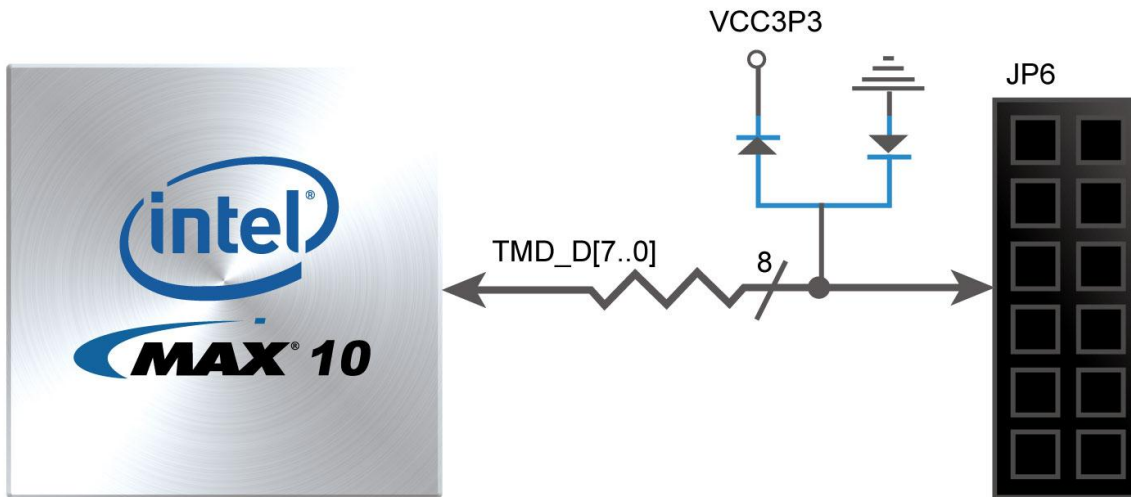
Figure 2-24 Connections between the 2x6 TMD header and MAX 10 FPGA

Table 2-9 Pin Assignment of 2x6 TMD Header.

Signal Name	FPGA Pin No.	Description	I/O Standard
TMD_D[0]	PIN_AB8	GPIO Connection [0]	3.3-V LVTTL
TMD_D[1]	PIN_AA8	GPIO Connection [1]	3.3-V LVTTL
TMD_D[2]	PIN_AA5	GPIO Connection [2]	3.3-V LVTTL
TMD_D[3]	PIN_AB5	GPIO Connection [3]	3.3-V LVTTL
TMD_D[4]	PIN_AB7	GPIO Connection [4]	3.3-V LVTTL
TMD_D[5]	PIN_AA7	GPIO Connection [5]	3.3-V LVTTL
TMD_D[6]	PIN_AB6	GPIO Connection [6]	3.3-V LVTTL
TMD_D[7]	PIN_AA6	GPIO Connection [7]	3.3-V LVTTL

2.6 1x10 A/D Converter and Analog Input

The board has eight analog inputs are connected to MAX 10 FPGA ADC1, through a 1x10 header (JP1) input. Any analog inputs signals sourced through the header JP1 are RC filtered by the Analog Front-End circuit, and the maximum allowable input voltage 2.5V per the MAX 10 FPGA ADC IP block. **Figure 2-25** shows the connections between the header input and FPGA.
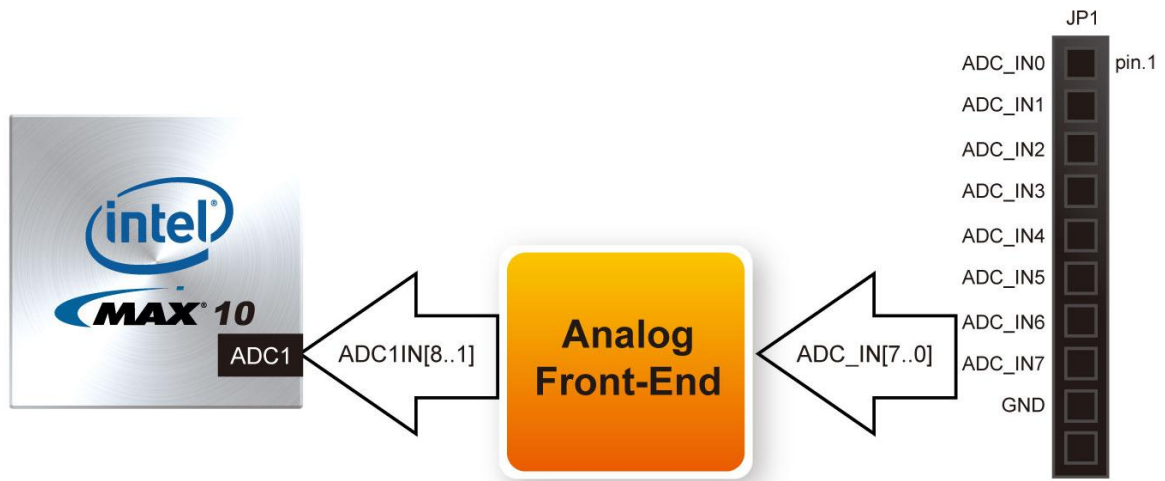
Figure 2-25 Connections between the Arduino Analog input and MAX 10 FPGA

Table 2-10 shows the pin assignment of 1x10 Analog input header.

Table 2-10 Pin Assignment of Analog Input

Signal Name	FPGA Pin No.	Description	I/O Standard
ADC_IN [0]	PIN_F5	Analog Input Data[0]	2. 5-V LVTTL
ADC_IN [1]	PIN_F4	Analog Input Data[1]	2. 5-V LVTTL
ADC_IN [2]	PIN_J8	Analog Input Data[2]	2. 5-V LVTTL
ADC_IN [3]	PIN_J9	Analog Input Data[3]	2. 5-V LVTTL
ADC_IN [4]	PIN_J4	Analog Input Data[4]	2. 5-V LVTTL
ADC_IN [5]	PIN_H3	Analog Input Data[5]	2. 5-V LVTTL
ADC_IN [6]	PIN_K5	Analog Input Data[6]	2. 5-V LVTTL
ADC_IN [7]	PIN_K6	Analog Input Data[7]	2. 5-V LVTTL

2.7 Using QSPI Flash

The board supports a 64M-bit serial NOR flash device for non-volatile storage, user data and program. This device has a 4-bit data interface and uses 3.3V CMOS signaling standard. Connections between MAX 10 FPGA and Flash are shown in **Figure 2-26**. **Table 2-11** shows the QSPI Flash pin assignments
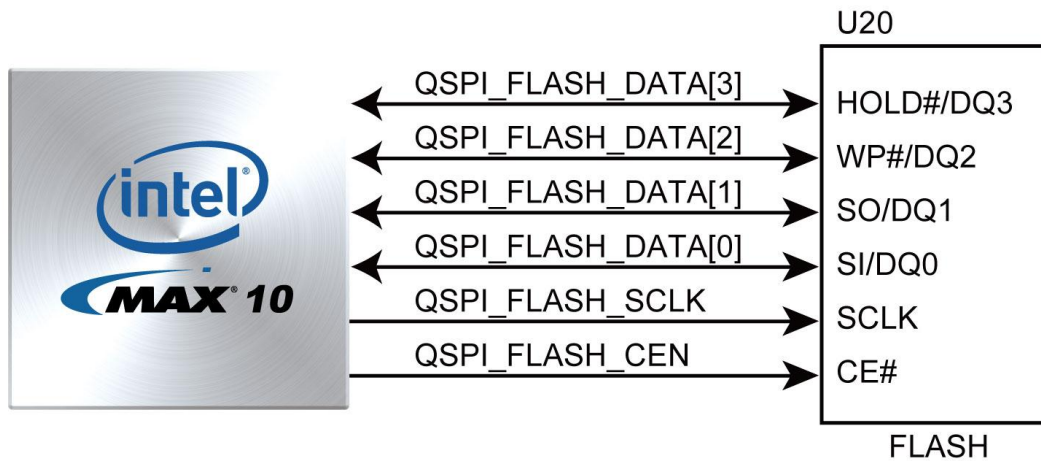
Figure 2-26 Connections between MAX 10 FPGA and QSPI Flash

Table 2-11 Pin Assignment of QSPI Flash

Signal Name	FPGA Pin No.	Description	I/O Standard
QSPI_FLASH_DATA[0]	PIN_P21	FLASH Data[0]	3.3-V LVTTL
QSPI_FLASH_DATA[1]	PIN_P18	FLASH Data[1]	3.3-V LVTTL
QSPI_FLASH_DATA[2]	PIN_R18	FLASH Data[2]	3.3-V LVTTL
QSPI_FLASH_DATA[3]	PIN_P20	FLASH Data[3]	3.3-V LVTTL
QSPI_FLASH_SCLK	PIN_P19	FLASH Data Clock	3.3-V LVTTL
QSPI_FLASH_CEN	PIN_L22	FLASH Chip Enable	3.3-V LVTTL

Example Designs

This chapter provides examples of advanced designs implemented by RTL or Qsys on the T-Core board. These reference designs cover the features of peripherals connected to the FPGA, such as RGB LED. All the associated files can be found in the directory \Demonstrations of T-Core System CD.

■ **Installation of Demonstrations**

To install the demonstrations on your computer:

Copy the folder Demonstrations to a local directory of your choice. It is important to make sure the path to your local directory contains NO space. Otherwise, it will lead to error in Nios II software.

Note: Quartus Prime Standard v18.1 or later is required for all T-Core demonstrations to support MAX 10 FPGA device.

3.1 T-Core Factory Configuration

The T-Core board has a default configuration bit-stream pre-programmed, which demonstrates some of the basic features onboard. The setup required for this demonstration and the location of its files are shown as below.

■ **System Block Diagram**

Figure 3-1 shows the system block diagram of this demonstration. This system can control the four on-board green LEDs to all light on or blink. It can also control the four on-board RGB LEDs to display the Red/Green/Blue/White colors, or the 0~360degree continuous color according to the Hue of the HSL Color Model.
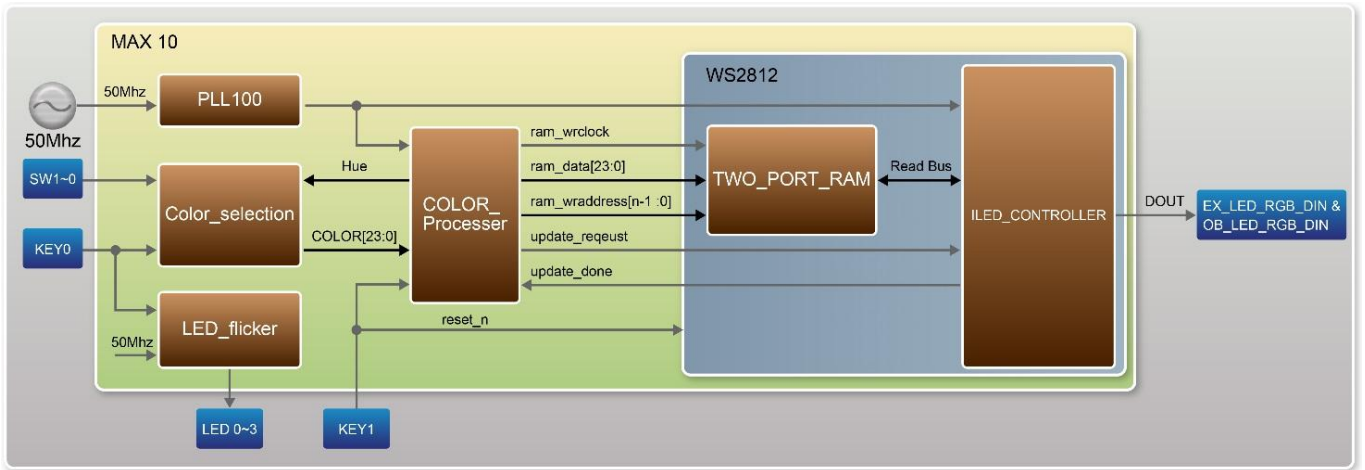
Figure 3-1 Block diagram of the Default demo

Below is the description for the block diagram:

1. **PLL100**: PLL module. Input 50MHz clock and output 100MHz clock.

2. **KEY1**: It is used to reset the system.

3. **Color_selection**: Provides four fixed colors Red/Green/Blue/White or the 0~360 degree continuous color according to the Hue of the HSL Color Model. When the KEY0 is pressed, the color of the four on-board RGB LEDs is depending on the SW[1:0] setting, they will display Red/Green/Blue/White color respectively when the SW[1:0] is set as 00/01/10/11. If the KEY0 is not pressed, the four RGB LEDs will display 0~360 degree continuous color.

4. **LED_flicker**: It controls the four on-board green LEDs to all light on or blink with 1Hz frequency. The four LEDs will all light on when KEY0 is pressed, or will blink with 1Hz if the KEY0 is not pressed.

5. **WS2812**: It is the WS2812 IC control IP, it can control any WS2812 IC, it contains two main parts:

TWO_PORT_RAM: User can write in the R[7:0] G[7:0] B[7:0] data to control any one of RGB LEDs to be light as any colors. For example, user can set ram_wraddress=2, ram_data = {R[7:0], G[7:0], B[7:0]} = {8'h00,8'hff ,8'h00} to make the RGB LED3 light as Green, then set the ram_wrclock to 0->1->0, the data will be written to the RAM.

ILED_CONTROLLER: After the LED RGB data is written successfully, active the update_request as "1" to inform the ILED_CONTROLLER read out the RGB data in sequence through the TWO_PORT_RAM, and input to the corresponding LED. The update_request must return to "0" after a short of delay (>0.02MHz). The DOUT is the bit string data, the "0"/"1" Sequence chart of each bit is generated according to WS2812 datasheet page 4. The sequence chart of each RGB bit is generated according to WS2812 datasheet page 5. After all the data is transferred to WS2812 RGB LED, this module will return update_done ="1" to indicates the data transmission is completed.

6. **COLOR_Processer**: It will extract the color date from Coloe_selection according to WS2812 IP controlling request, then output to the TWO_PORT_RAM in WS2812 IP, when this is completed, it will inform the ILED_CONTROLLER in WS2812 IP to sequence read out the RGB data stored in TWO_PORT_RAM, then the ILED_CONTROLLER will transfer the RGB data to each WS2812 IC and wait for update_done = "1" returned by ILED_CONTROLLER, it indicates the process is completed.

■ Demonstration File Locations

- Project directory: \Default

- Bitstream used: T_CORE_Default.sof, T_CORE_Default.pof

■ Demonstration Setup and Instructions

- Connect the Mini USB connector (J2) of the T-Core board to the host PC with a USB cable and install the USB-Blaster II driver if necessary.

- Execute the demo batch file "test.bat" from the directory \Default\demo_batch\

- If the KEY0 is not pressed, the four on-board green LEDs will blink in 1Hz and the four on-board RGB LEDs will display 0~360 degree continuous color according to the Hue of the HSL Color Model.

- If the KEY0 is pressed on (do not release), the four on-board green LEDs will all light on, the color of the on-board four RGB LEDs is depending on SW[1:0], they will display Red/Green/Blue/White color respectively when the SW[1:0] is set as 00/01/10/11.

- **Figure 3-2** shows the color of the four on-board green LEDs and RGB LEDs display.
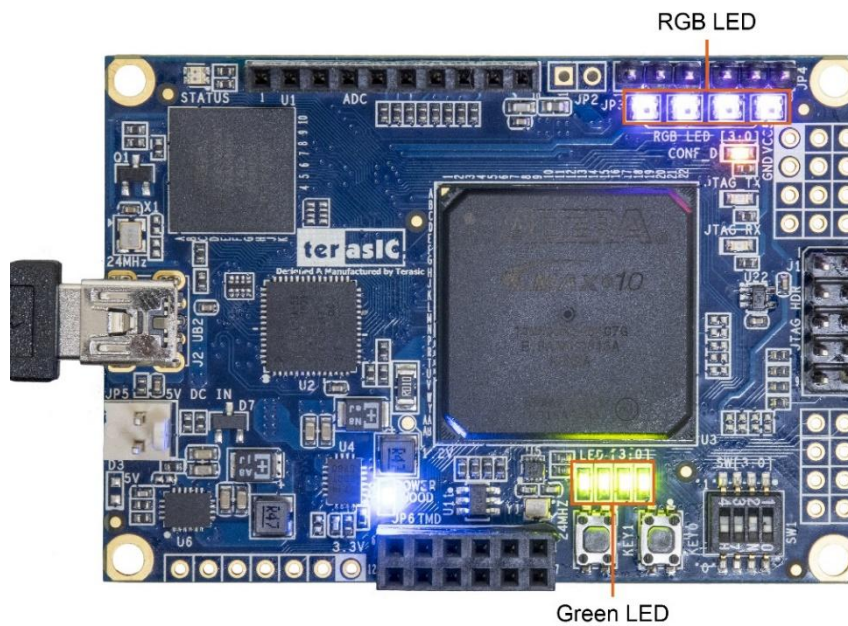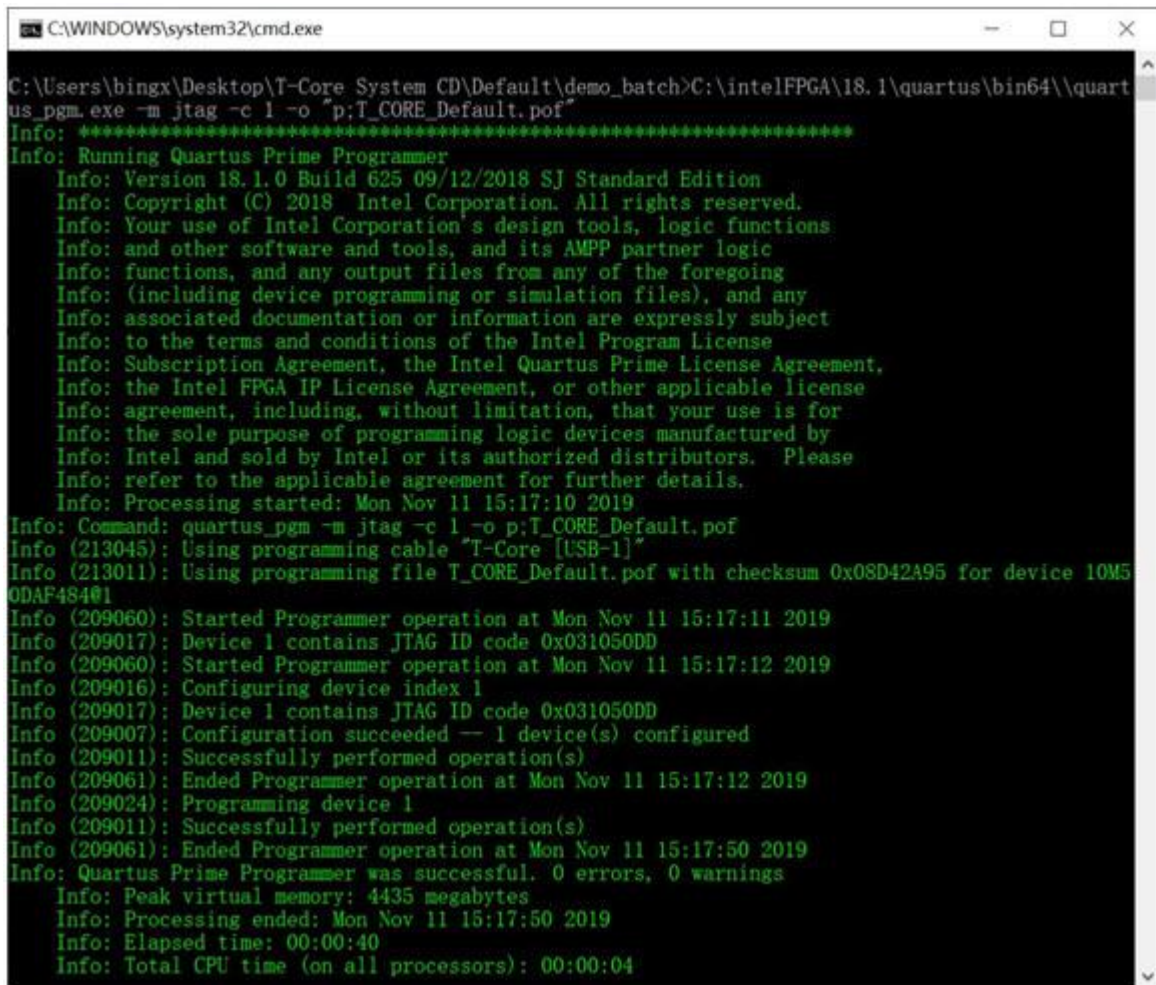
Figure 3-2 On-Board Green LED and RGB LED display

■ **Restore Factory Configuration**

• Connect the mini USB connector (J2) of the T-Core board to the host PC with a USB cable and install the USB-Blaster II driver if necessary.

• Execute the demo batch file "program.bat" from the directory \Default\demo_batch\

• After the program is executed successfully, the program message as shown in **Figure 3-3**. Please press any key to continue and the window will be closed.

Figure 3-3 Quartus Prime Program MAX 10 CFM message

3.2 Control RGB LED in Nios II

Based on this demonstration, user can quickly learn how to develop C language program to control the RGB LEDs, it provides corresponding API function and allows the user to quickly started.

■ **System Block Diagram**

Figure 3-4 shows the system block diagram of this demonstration. Nios II CPU and WS2812 Controller must run in 100MHz, the on-board OSC 50MHz needs to be multiplied to 100MHz

through the PLL, the JTAG is used to print information, the Nios II required a system time, the C language program is stored in On-Chip memory, user can develop C program to call the API WS2812 Controller, then control the RGB LEDs. User can also control the external RGB LED by modifying the parameters in the C program.
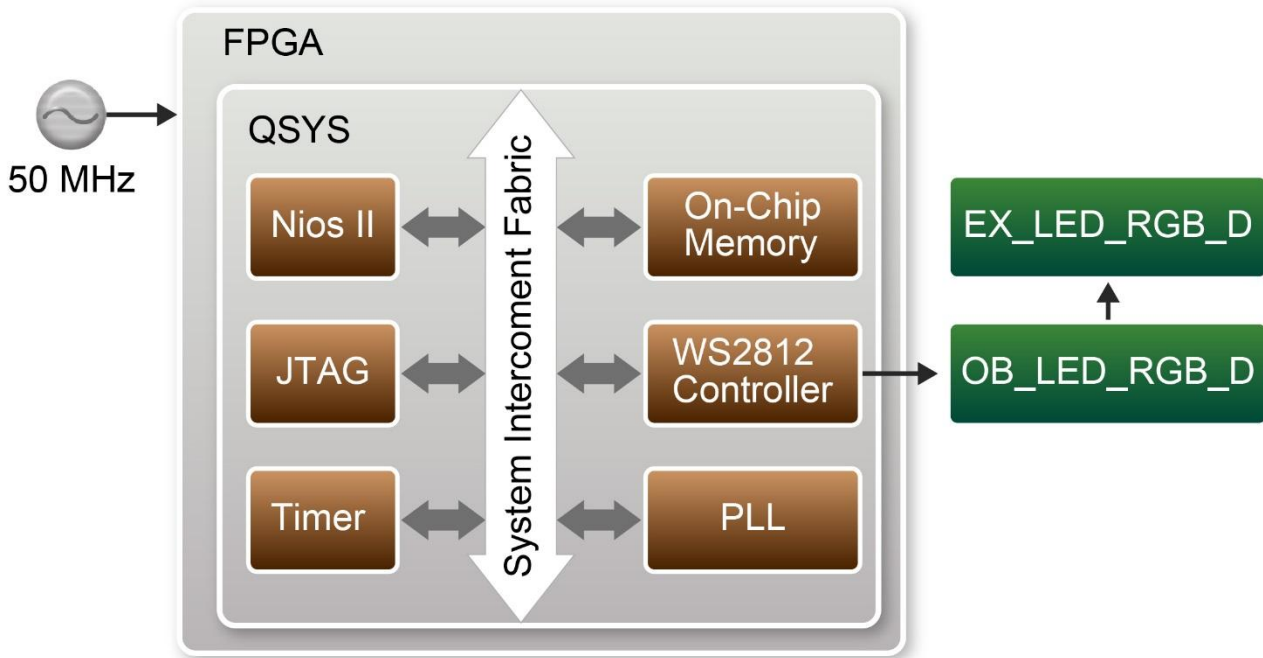
Figure 3-4 Block diagram of the SDRAM test in Nios II

■ **Design Tools**

- Quartus Prime v18.1
- Nios II Eclipse v18.1

■ **Demonstration Source Code**

- Quartus project directory: \RGBLED_NIOS
- Nios II Eclipse directory: \RGBLED_NIOS\Software

■ **Nios II Project Compilation**

- Click "Clean" from the "Project" menu of Nios II Eclipse before compiling the reference design in the Nios II Eclipse.

■ **Demonstration Batch File**

- The files are located in the director: \RGBLED_NIOS\demo_batch
- The folder includes the following files:

- Batch file for USB-Blaster II: test.bat and test.sh

- FPGA configuration file: T_CORE.sof

- Nios II program: WS2812_Test.elf

■ Demonstration Setup

- Quartus Prime v18.1 and Nios II v18.1 must be pre-installed on the host PC.

- Connect the T-Core board to the host PC with a USB cable and install the USB-Blaster driver if necessary.

- Execute the demo batch file "test.bat" from the directory \REGLED_NIOS\demo_batch

- After the program is downloaded and executed successfully, a prompt message will be displayed in nios2-terminal.

- The program will display the test progress and result, as shown in

Figure 3-5.

- As shown in **Figure 3-6**, the RGB LED[3:0] display purple/green/orange/red and red/orange/green/purple respectively in turns.

Figure 3-5 Display the progress and the result for the WS2812 test in Nios II
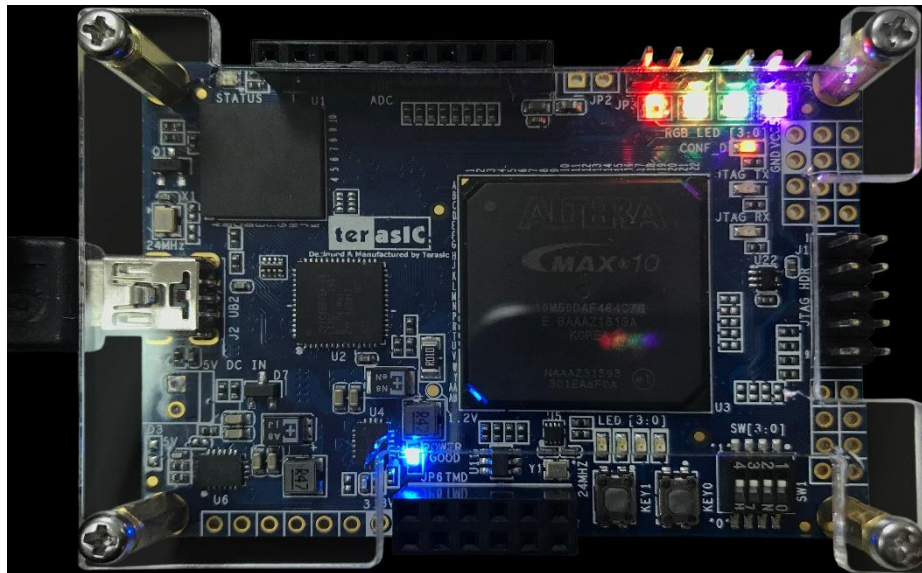
Figure 3-6 The RGB LEDs display after executing the batch file

3.3 Control External RGB LED in Nios II

This demonstration is very similar to the one in section 3.2. The difference is that user need to connect external RGB LEDs. Without the external power, the T-Core can control almost 10~20 external RGB LEDs, the quantities of external RGB LEDs are depending on the user's code which is used to control the LED lightness.

User can refer to below demonstration to modify the C program and control the external RGB LEDs.

```
#define LED_NUM 143 // modify LED number

……

//call function WS2812_LED_SET(alt_u32 SelLED,alt_u8 R_Color, alt_u8 G_Color,   alt_u8 B_Color)

//SelLED: Select LED   ,Red : 8Bit   ,Green: 8Bit   ,Blue: 8Bit

WS2812_LED_SET(100,30,0,0); // set number 100 LED to red color.

WS2812_update();//update status

……
```

■ **Design Tools**

- Quartus Prime v18.1
- Nios II Eclipse v18.1

■ **Demonstration Source Code**

- Quartus project directory: \RGBLED_ NIOS_EXT
- Nios II Eclipse directory: \ RGBLED_ NIOS_EXT\software

■ **Demonstration Batch File**

- The files are located in the director: \ RGBLED_ NIOS_EXT\demo_batch
- The folder includes the following files:
- Batch file for USB-Blaster: test.bat and test.sh
- FPGA configuration file: T_CORE.sof

- Nios II program: WS2812_Test.elf

■ Demonstration Setup

- Quartus Prime v18.1 and Nios II v18.1 must be pre-installed on the host PC.

- As shown in **Figure 3-7**, connect the external RGB LED strip to the JP3 or JP4 port of T-Core board, the pin marked with red box is connected to VCC, the pin marked with green box is connected to the signal, the pin marked with yellow box is connected to GND.

- Connect the T-Core board to the host PC with a USB cable and install the USB-Blaster driver if necessary.

- Execute the demo batch file "test.bat" from the directory \RGBLED_ NIOS_EXT\demo_batch

- After the program is downloaded and executed successfully, a prompt message will be displayed in nios2-terminal.

- The color of external RGB LED strip will change in turn.
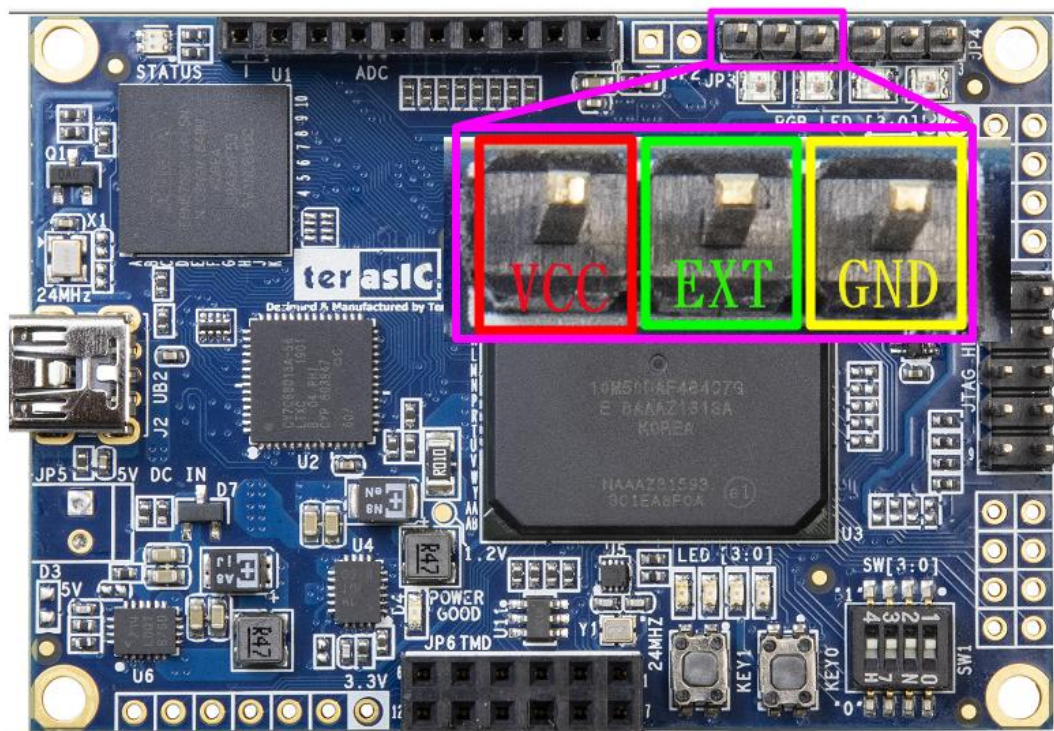
Figure 3-7 External RGB LED connection diagram

3.4 ADC Measurement

This demonstration shows how to use the FPGA on-die ADC to measure the input power voltage from the eight analog input pins among the ADC connector. The measured voltage is displayed on the Signal Tap. **Figure 3-8** shows the block diagram of this demonstration. The main control uses the Intel ADC IP to retrieve the 12-bit digitalized analog value according the channel specified by

the SWITCH on the T-Core board. The input voltage can be calculated based on the 12-bit digital value. Finally, the voltage value will be displayed on the Signal Tap Logic Analyzer.

The 12-bit represents 0~2.5V input voltage to the ADC hardware. The input voltage to the ADC analog input can be calculated with the formula listed below:

Voltage input= (12-bit integer value)/4096 x 2.5
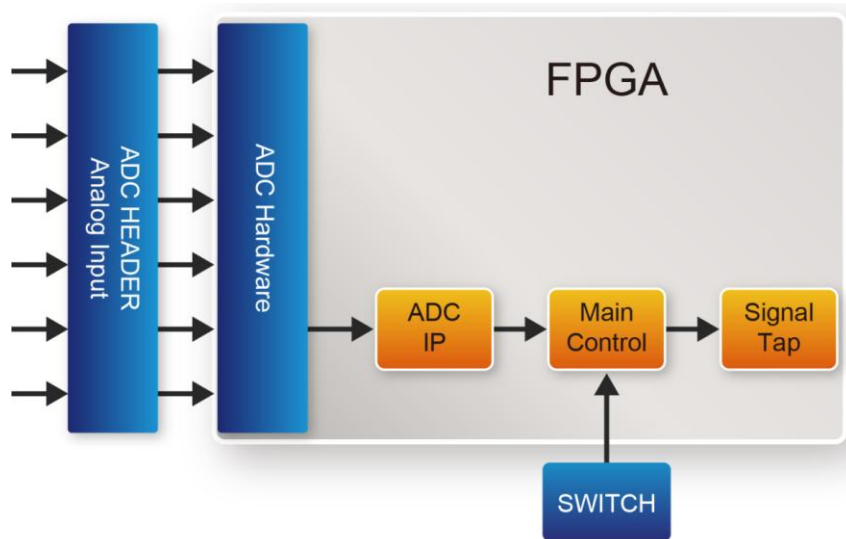
Figure 3-8 Block diagram of the ADC measurement demonstration

Table 3-1 The SW[2:0] Setting

SW(2:0)	ADC_Source
3'b000	ADC_IN0
3'b001	ADC_IN1
3'b010	ADC_IN2
3'b011	ADC_IN3
3'b100	ADC_IN4
3'b101	ADC_IN5
3'b110	ADC_IN6
3'b111	ADC_IN7

In the demonstration, a Signal Tap is provided to display the retrieved 12-bit digitalized value and the calculated voltage for the input power to the analog input as shown in **Figure 3-9**.
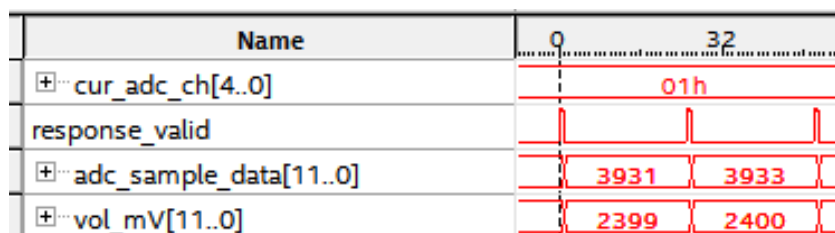
Figure 3-9 Signal Tap of ADC

■ Design Tools

- Quartus Prime 18.1 Standard Edition

■ Demonstration Source Code

- Quartus Project directory: ADC_RTL

- Bitstream used: T_CORE.sof

- Demo Batch File: ADC_RTL\demo_batch\test.bat

■ Demonstration Setup

- Make sure Quartus Prime is installed on your PC

- Connect analog source to the analog input as shown in **Figure 3-10**.
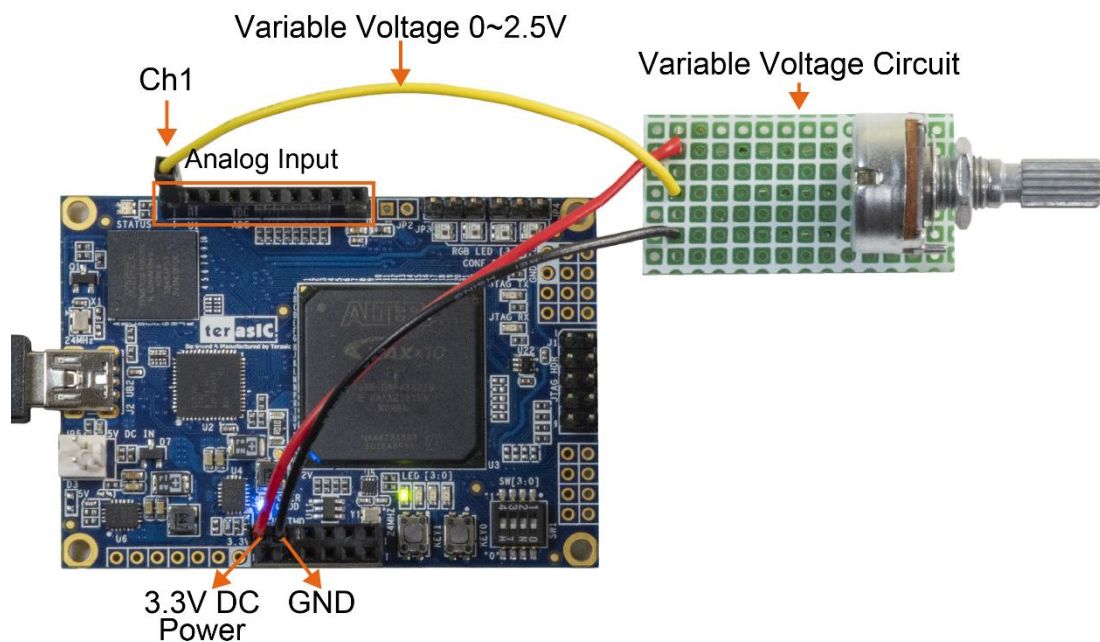
Figure 3-10 Connect analog source to the analog input

- Use the USB Cable to connect your PC and the FPGA board and install USB Blaster II driver if necessary.

- Execute the demo batch file "test.bat" from the directory ADC_RTL\demo_batch.

- Use SW[2:0] to specify measured analog input channel.

- The measured channel and voltage will be displayed on Signal Tap Logic Analyzer as shown in **Figure 3-11**.

In the above steps we use a self-made variable voltage output circuit. It uses the T-Core 3.3V as the power source, the output voltage is adjusted by the divider resistance and the variable resistor of the circuit, which can convert the T-Core 3.3V to a 0~2.5V analog input (to the ADC_IN0 channel).

Note: Users can input their own power source, please make sure the input voltage is 0~2.5V, otherwise there is a risk of damage to the development board.
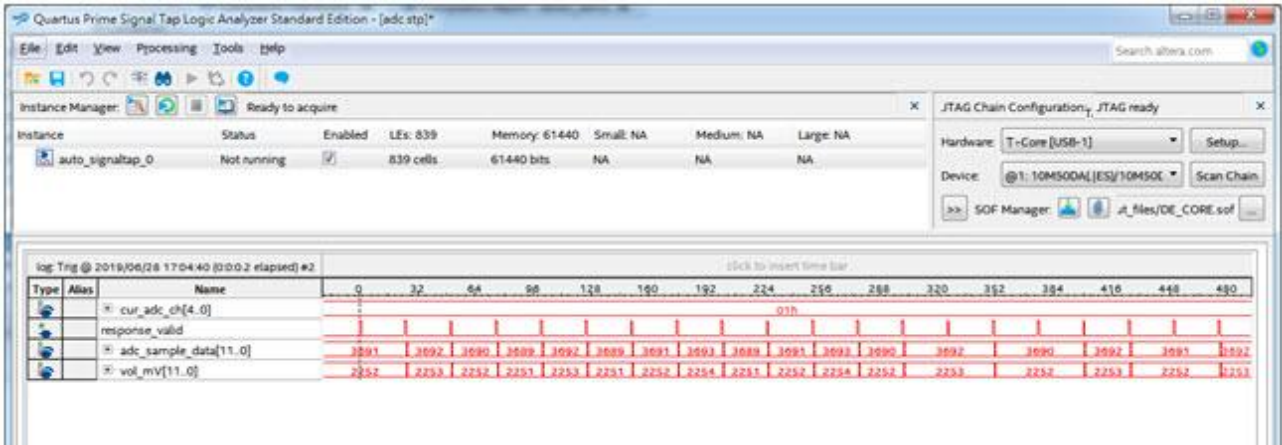
Figure 3-11 Signal Tap Logic Analyzer displays the measured channel and voltage

3.5 RISC-V

■ RISC-V Brief Introduction

RISC-V is a free and open Instruction Set Architecture (ISA) enabling a new era of processor innovation through open standard collaboration. Any organization or individual can design their own processor based on the RISC-V architecture. Freedom E300, a RISC-V core implemented in RTL, is realized on T-Core FPGA Board. For more information about Freedom E300, user can learn more from web (https://github.com/sifive/freedom).

Figure 3-12 shows the top-level block diagram of the Freedom E300 which contains an E31 Coreplex, a selection of flexible I/O peripherals, a dedicated off-chip Quad-SPI flash controller with execute-in-place support, On-chip Memory, clock generation, and an always-on (AON) block. In addition, custom accelerator can be added to provide application-specific processing.
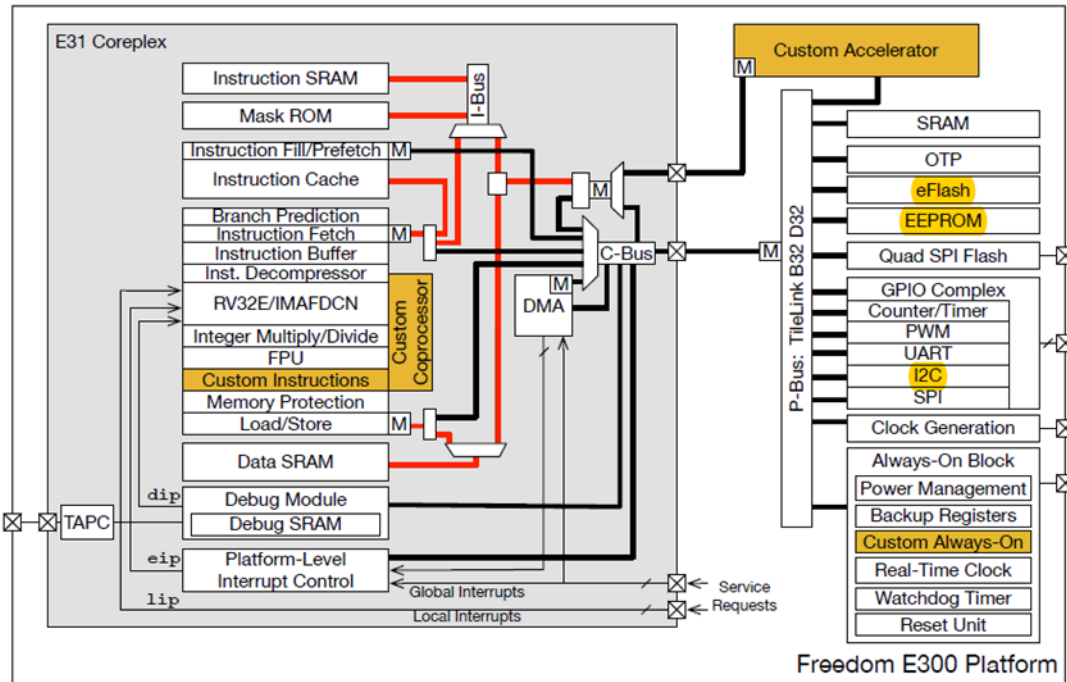
Figure 3-12 Top-Level Block Diagram of the E300 platform

The memory map of Freedom E300 is shown in **Table 3-2**.

Table 3-2 Memory map of Freedom E300

Base	Top	Description	Notes
0x0000_0000	0x0FFF_FFF	refer E3 Coreplex Manual	
0x1000_0000	0x1000_7FFF	Always-On (AON)	E3 Coreplex
0x1000_8000	0x1000_FFFF	Power, Reset, Clock, Interrupt (PRCI)	
0x1001_0000	0x1001_0FFF	On-chip OTP control	
0x1001_1000	0x1001_1FFF	On-chip eFlash control	
0x1001_2000	0x1001_2FFF	GPIO0	
0x1001_3000	0x1001_3FFF	UART0	
0x1001_4000	0x1001_4FFF	QSPI0	On-Chip Peripherals
0x1001_5000	0x1001_5FFF	PWM0	
0x1002_3000	0x1002_3FFF	UART1	
0x1002_4000	0x1002_4FFF	QSPI1	
0x1002_5000	0x1002_5FFF	PWM1	
0x1003_4000	0x1003_4FFF	QSPI2	
0x1003_5000	0x1003_5FFF	PWM2	
0x2000_0000	0x3FFF_FFFF	QSPI0 XIP	Off-Chip Non-Volatile Memory
0x4000_0000	0x7FFF_FFFF	Additional I/O or RAM	
0x8000_0000	0x8001_FFFF	Instruction and Data RAM	On-Chip Volatile Memory
0x8002_0000	0xFFFF_FFFF	Additional RAM	

■ About Quartus Project

The original Freedom E300 was implemented in the Scala and is not suitable for the T-Core FPGA board. We modified the Scala source code so that it can run on T-Core. Since the Verilog code generated by the Scala is hardly readable, we encapsulate it. Place the clock, reset, JTAG interface, QSPI interface, GPIO interface (including various peripherals such as UART, I2C, SPI, etc.) on the top-level port for easy logic operation and pin connection to the FPGA. The interface of the top file is as follows.

```
149  ⊟Freedom_E300_Wrapper E300_SoC_Wrapped(
150   | /*input  wire */        .clock          (clk_32M),
151   | /*input  wire */        .aon_lfextclk   (slowclk);
152   |
153   | /*input  wire */        .aon_erst_n     (~ip_reset_sys_peripheral_reset),
154   | /*input  wire */        .jtag_reset     (fpga_power_on_power_on_reset),
155   | /*output wire */        .io_ndreset     (ip_reset_sys_mb_debug_sys_rst),
156   |
157   | /*input  wire */        .jtag_tck       (RISCV_JTAG_TCK),
158   | /*input  wire */        .jtag_tms       (RISCV_JTAG_TMS),
159   | /*input  wire */        .jtag_tdi       (RISCV_JTAG_TDI);
160   | /*output wire */        .jtag_tdo       (RISCV_JTAG_TDO);
161   |
162   | /*output wire */        .qspi_sck       (QSPI_FLASH_SCLK),
163   | /*output wire */        .qspi_cs_0      (QSPI_FLASH_CE_n),
164   | /*inout  wire [3:0] */  .qspi_dq        (QSPI_FLASH_DATA),
165   |
166   | /*input  wire [31:0] */ .e300_gpio_i_ival (e300_gpio_i_ival),
167   | /*output wire [31:0] */ .e300_gpio_o_ie   (e300_gpio_o_ie),
168   | /*output wire [31:0] */ .e300_gpio_o_oval (e300_gpio_o_oval),
169   | /*output wire [31:0] */ .e300_gpio_o_oe   (e300_gpio_o_oe)
170   |
171   | );
```

Add the PLL, Clock, LED and control logic to the Freedom_E300_Wrapper to generate the complete Quartus Project. The block diagram is shown below. The completed Quartus project is detailed in the demonstration directory.
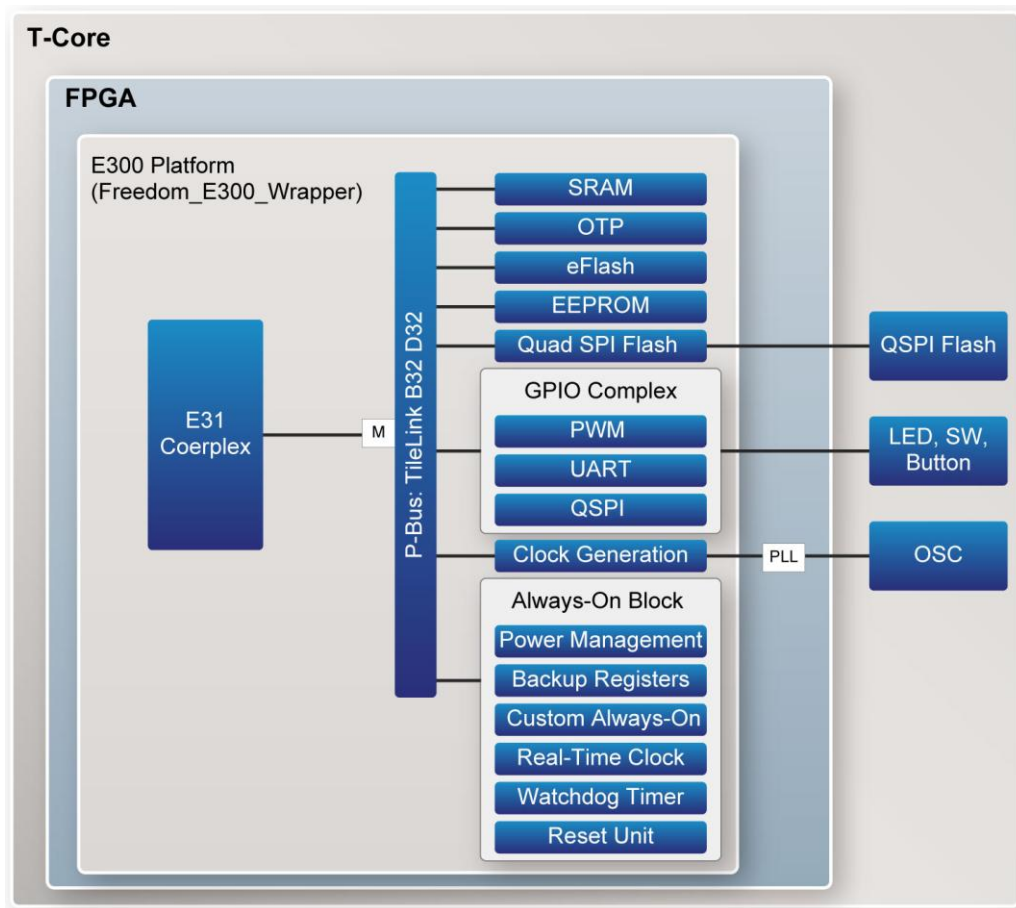
Figure 3-13 Quartus Project System Block

■ **How to Run Demo (Using Ubuntu)**

◆ **Host PC**

A Linux system is required to compile the RISC-V C-code project and download the binary file to the T-Core QSPI Flash. In the following demo, Ubuntu 16.04 is used and recommended. Quartus Prime (or Quartus Programmer), USB Blaster II Driver, RISC-V BSP should be installed on the host PC before running the demo.

◆ **Intel Quartus Prime**

Intel Quartus Prime 18.1 Standard Edition can be download from web site: http://fpgasoftware.intel.com/18.1/?edition=standard&platform=linux&download_manager=dlm3.

For Quartus Prime installation, please make sure the MAX 10 device is included.

Figure 3-14 Download Quartus Prime 18.1 Standard Edition

◆ **USB-Blaster II Driver**

The Intel Quartus Prime software accesses USB-Blaster II through USB file system. After installing the Quartus Prime software, user need to change the port permissions.

1. Create a file named /etc/udev/rules.d/51-usbblaster.rules.

2. Add the following lines.

   SUBSYSTEMS=="usb",   ATTRS{idVendor}=="09fb",   ATTRS{idProduct}=="6010",
   MODE="0666"

   SUBSYSTEMS=="usb",   ATTRS{idVendor}=="09fb",   ATTRS{idProduct}=="6810",
   MODE="0666"

◆ **T-Core RISC-V BSP**

After Quartus Prime and USB-Blaster II driver are installed, please download the T-Core RISC-V BSP (TCORE-RISCV-FreedomE300.tar.gz) from the web (http://T-Core.terasic.com/cd). Then, decompress the file using command: tar -zxvf TCORE-RISCV-FreedomE300.tar.gz. So far, you can see TCORE-RISCV-FreedomE300 file. For more information about T-Core RISC-V BSP, please refer following table.

Table 3-3 T-Core RISC-V BSP

file or folder	Description
Quartus_Project/TCORE_FreedomE 300	a Quartus Prime project, contains a RISC-V Core, Freedom E300.
Quartus_Project/ TCORE_FreedomE300/V/Freedom_ E300_Wrapper.v	the top file of Freedom E300. User can use it to connect to do logic operation, etc.
Toolchains	prebuilt tools for compiling and debugging programs.
freedom-e-sdk/bsp/drivers	contains some driver code, such as the underlying driver code for PLIC modules, PRCI, etc.
freedom-e-sdk/bsp/include	contains header files for register address and parameter of modules, such as SPI, UART, GPIO, etc.
freedom-e-sdk/bsp/libwrap	contains specific implementation of underlying library functions related to the hardware platform.
freedom-e-sdk/bsp/env/freedom-e300 -terasic	contains Terasic T-Core board supported package.
freedom-e-sdk/software	contains demonstration for reference.
freedom-e-sdk/Makefile	the main Makefile for compilation and debug.

◆ **Download a pof file to FPGA**

Follow the procedures below to write RISC-V FPGA configure data into MAX10 internal flash.

o Make sure SW2 is changed to correct location, set SW2.1=0 and SW2.2=1, as shown in **Figure 3-15** to ensure MAX10 is on JTAG chain, and RISC-V is off JTAG chain.
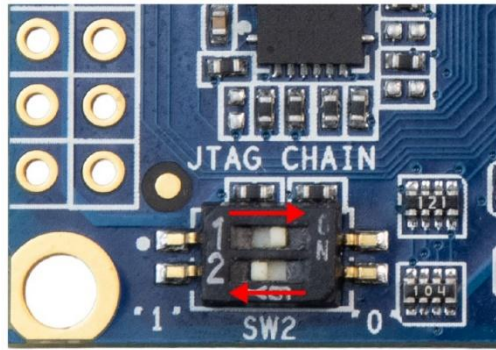
Figure 3-15 SW2 Setting

- o Make sure Quartus Prime 18.1 Standard Edition is installed to the host PC.
- o Connect the T-Core board (J2) to the host PC via the USB cable.
- o Go to TCORE-RISCV-FreedomE300/Quartus_Project/TCORE_FreedomE300/demo_batch directory using command:
  cd TCORE-RISCV-FreedomE300/Quartus_Project/TCORE_FreedomE300/demo_batch.
- o Execute command: sh programming_pof.sh to configure the FPGA.

◆ **Download bitstream to QSPI Flash**

Follow the procedures below to compile a C-code project and download the generated binary file into QSPI Flash via on-board RISC-V JTAG.

- o Remove the USB Blaster Cable from T-Core to power off the board.
- o Set SW2.1=1 and SW2.2=0 as **Figure 3-16** to ensure MAX10 is off JTAG chain, and RISC-V is on JTAG chain.
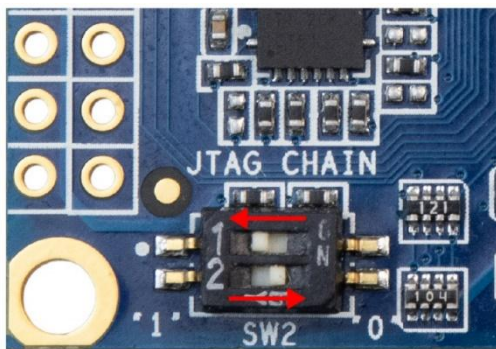
Figure 3-16 SW2 Setting

- o Connect the T-Core board (J2) to the host PC via the USB cable.
- o Change the directory to TCORE-RISCV-FreedomE300/freedom-e-sdk using cd command.
- o Compile software by executing command: make software.
- o Download the binary file by executing command: make upload.

Figure 3-17 Compile and download process

Note: When you programming the QSPI Flash, you may encounter the error like **Figure 3-18**. If so, you can execute the command: sudo apt-get install libfdti1-dev, as shown in **Figure 3-19**, to solve it.

Figure 3-18 The error encountered when programming QSPI Flash

Figure 3-19 Fix the error encountered when programming QSPI Flash

- o Now, you should now be able to observe the green LEDs are blinking.
- o Recycle the power, you can also observe the green LEDs are blinking as the demo program is loaded from QSPI Flash. After the reset operation, the RISC-V core will execute the demo program. You can program new software into board with the SDK, but the FPGA image will not be modified.

Chapter 4

Programming the Configuration

Flash Memory

This tutorial provides comprehensive information that will help you understand how to configure T-Core board using internal configuration mode. The MAX 10 device on the T-Core board supports single and dual image boot. This tutorial explains the details of the internal dual image boot and external image boot. The following sections provide a quick overview of the design flow.

Please note that if you are using the dual image boot function on the T-Core board, you will need to solder the JP2 2-pin header (pitch 0.100" (2.54 mm)) by yourself.

Please refer to the **Figure 4-1** for the JP2 position.

The settings of JP2 are described in Table **4-1**.
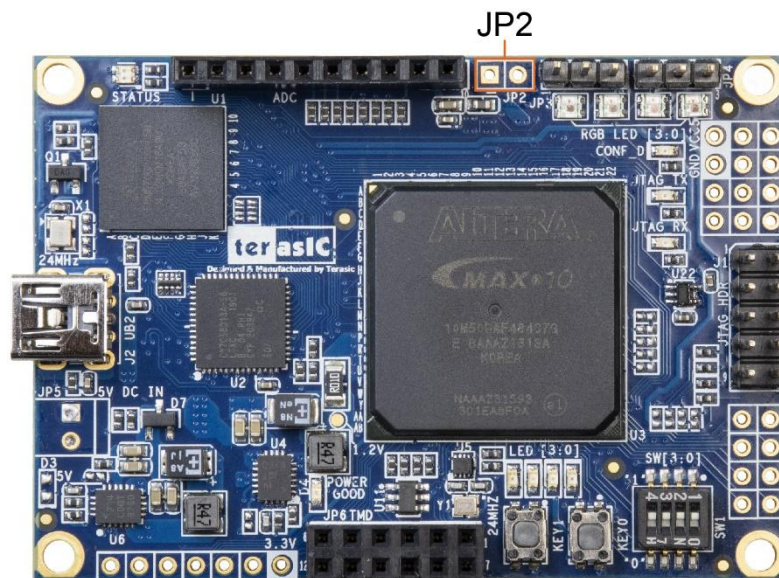

Figure 4-1 JP2 position on the T-Core board

Table 4-1 JP2 Jumper setting instructions

Reference	Jumper Setting	Description
JP2	Open (default)	Boot from image 0
	Close	Boot from image 1

4.1 Internal Configuration

The internal configuration scheme for all MAX 10 devices except for 10M02 device consists of the following mode:

- Dual Compressed Images — configuration image is stored as image0 and image1 in the configuration flash memory (CFM).

- Single Compressed Image.

- Single Compressed Image with Memory Initialization.

- Single Uncompressed Image.

- Single Uncompressed Image with Memory Initialization.

In dual compressed images mode, you can use the BOOT_SEL pin to select the configuration image.

The High-Level Overview of Internal Configuration for MAX 10 Devices as shown in **Figure 4-2.**
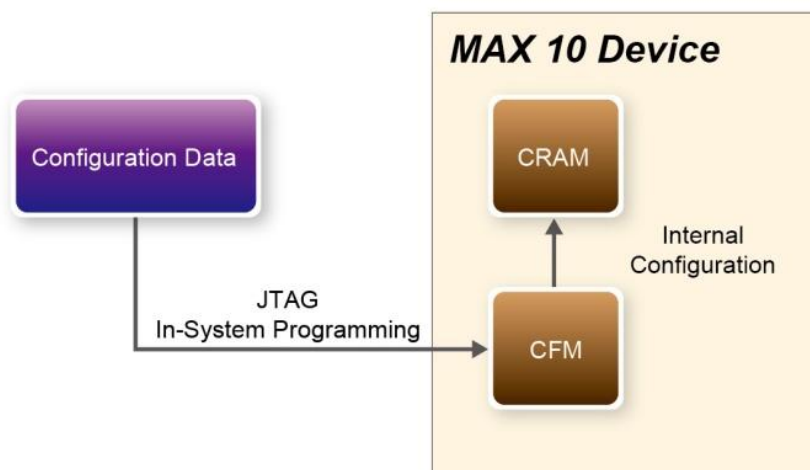
Figure 4-2 High-Level Overview of Internal Configuration for MAX 10 Devices

Before internal configuration, we need to program the configuration data into the configuration flash memory (CFM). The CFM will be part of the programmer object file (.pof) programmed into the internal flash through the JTAG In-System Programming (ISP).

During internal configuration, MAX 10 devices load the configuration RAM (CRAM) with configuration data from the CFM. Both of the application configuration images, image 0 and image 1, are stored in the CFM. The MAX 10 device loads either one of the application configuration image from the CFM. If an error occurs, the device will automatically load the other application configuration image. Remote System Upgrade Flow for MAX 10 Devices is shown in **Figure 4-3**.

Figure 4-3 Remote System Upgrade Flow for MAX 10 Devices

The operation of the remote system upgrade feature detecting errors is as follows:

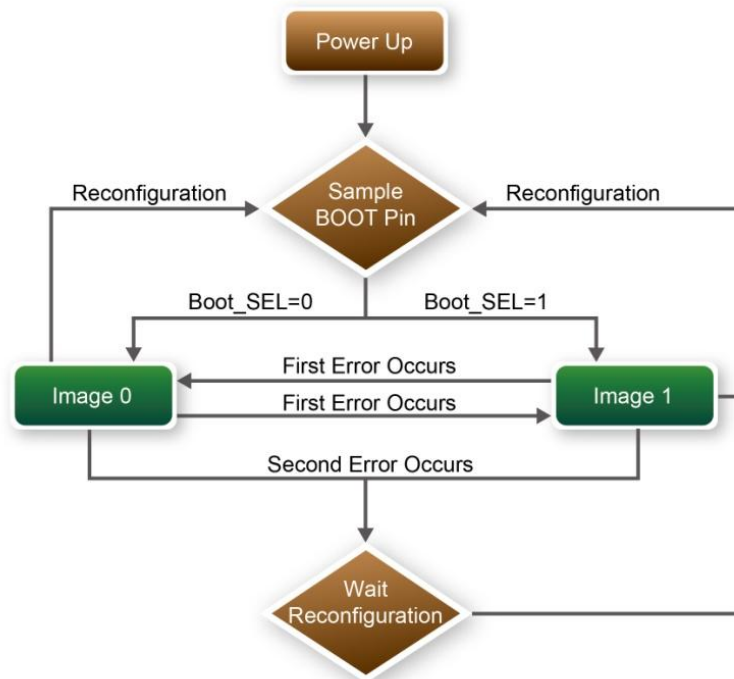1. After powering-up, the device samples the BOOT_SEL pin to determine which application configuration image to boot. The BOOT_SEL pin setting can be overwritten by the input register of the remote system upgrade circuitry for the subsequent reconfiguration.

2. If an error occurs, the remote system upgrade feature reverts by loading the other application configuration image. The following lists the errors that will cause the remote system upgrade feature to load another application configuration image:

   • Internal CRC error

   • User watchdog timer time-out

3. Once the revert configuration completes and the device is in the user mode, you can use the remote system upgrade circuitry to query the cause of error and which application image failed.

4. If the second error occurs, the device waits for a reconfiguration source. If the auto-reconfig is enabled, the device will reconfigure without waiting for any reconfiguration source.

5. Reconfiguration is triggered by the following actions:

   • Driving the nSTATUS low externally

   • Asserting internal or external nCONFIG low

   • Asserting RU_nCONFIG low (Avalon-MM interface signal)

4.2 Using Dual Compressed Images

Before using MAX 10 Dual Compressed Images feature, users must set these two image files' Quartus Prime projects as follows:

- Add dual configuration IP.

- Modify Configuration Mode in device setting.

First of all, a **Dual Configuration IP** should be added in an original project, so that the .pof file can be programmed into CFM through it. Here we use a demonstration named **"LED0_Blink"** as an example to add Intel Dual Configuration IP to the project:

1. Open Quartus project and choose **Tools > Platform Designer** to open Platform Designer GUI as shown in **Figure 4-4**.
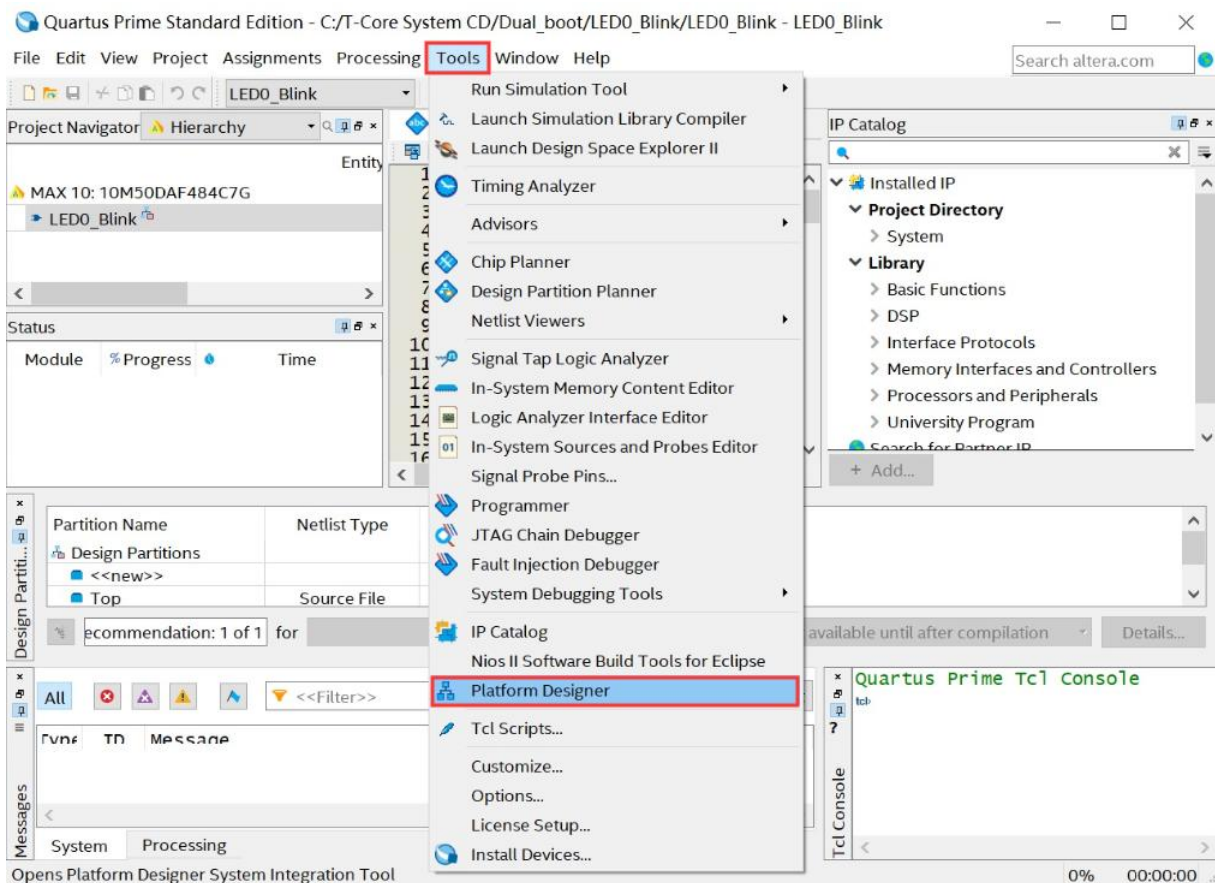

Figure 4-4 Select Qsys menu and click

2. Please choose **Library > Basic Function> Configuration and Programming > Dual Configuration Intel FPGA IP** to open wizard of adding dual boot IP. See **Figure 4-5** and **Figure 4-6**.
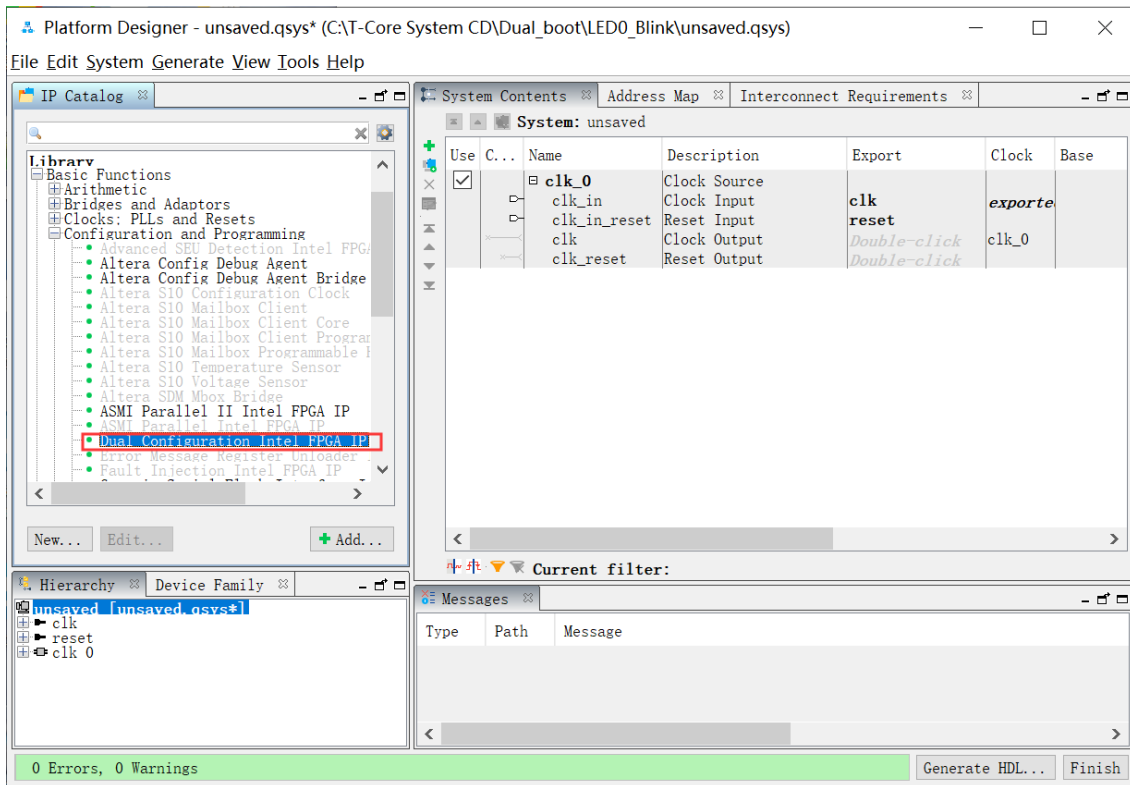
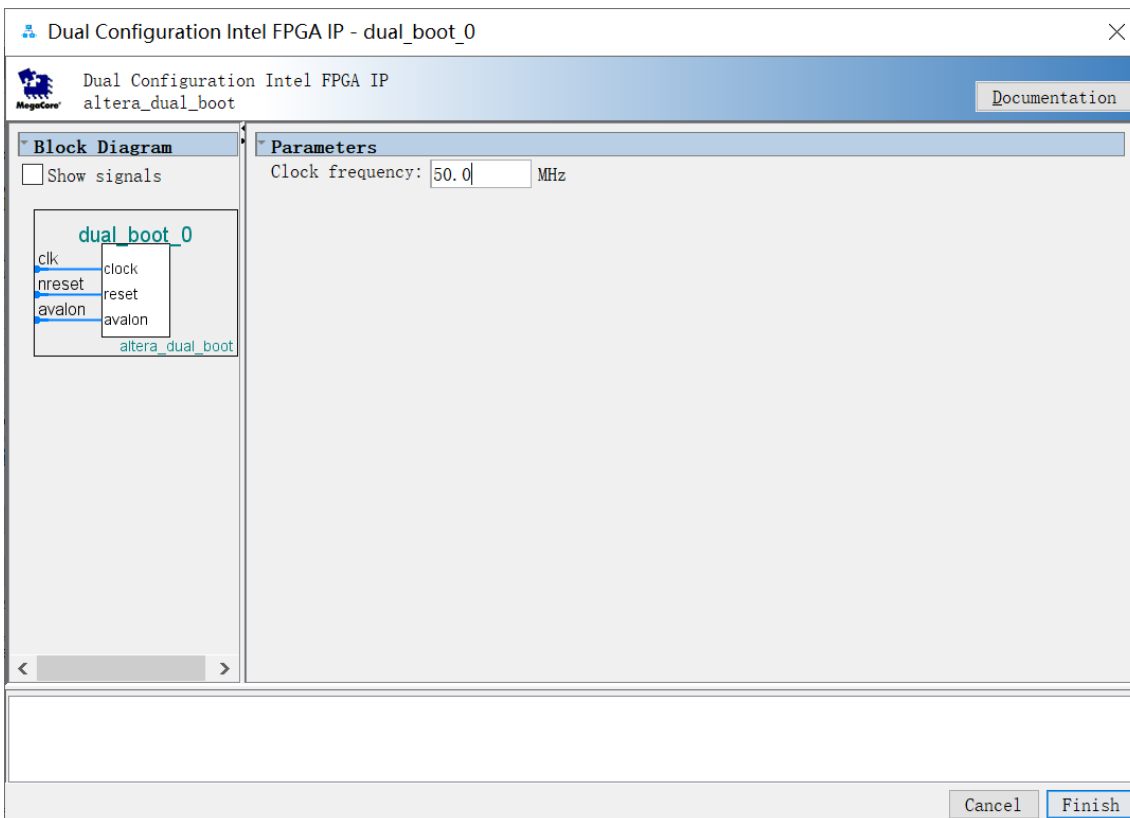**Figure 4-5 Select Dual Configuration Intel FPGA IP and click.**

Figure 4-6 Open wizard and click Finish.

3. Click **Finish** to close the wizard and return to the Platform Designer window. Choose **dual_boot_0** and click right key to select **Rename** it to **dual_boot**, and connect the **clk** and **nreset** to clk_0.clk and clk_0.clk_reset as shown in **Figure 4-7**.
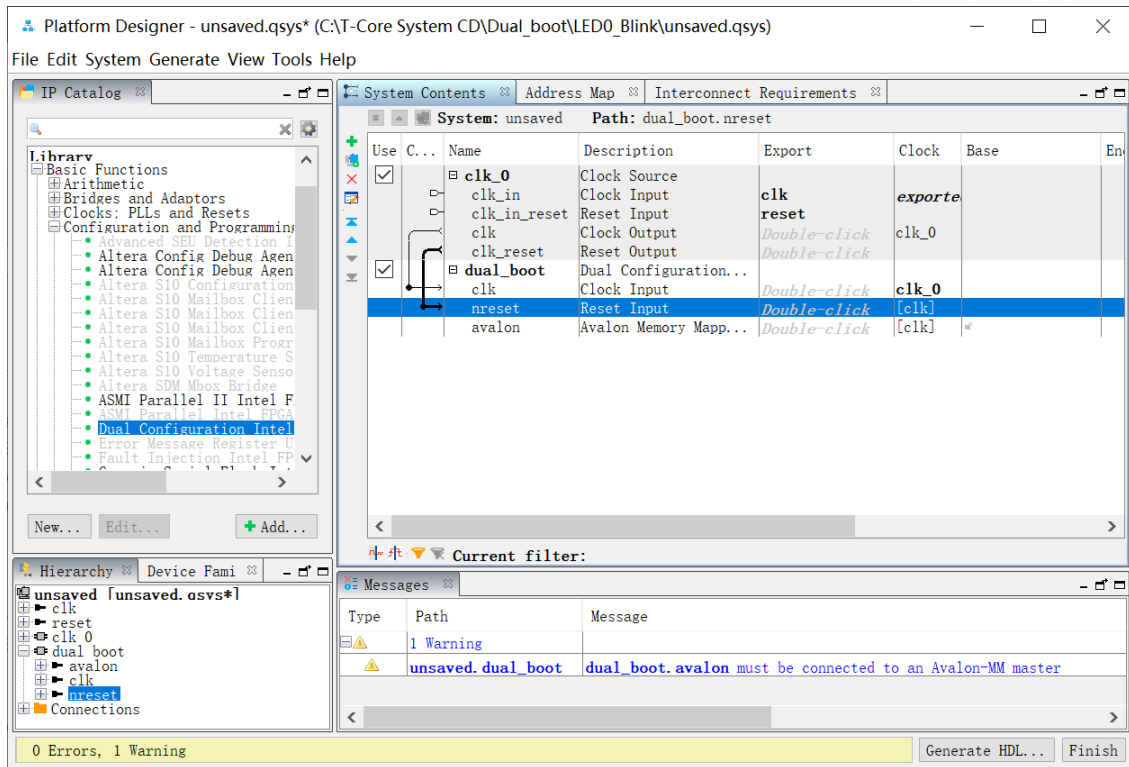

Figure 4-7 Rename and connect dual boot IP OK

4. Click "**Generate HDL..**" button and click **Generate** button when popping a window as shown in **Figure 4-8**. Click **Save** it as dual_boot.qsys and the generation start. If there is no error in the generation, the window will show successful as shown in **Figure 4-9**.
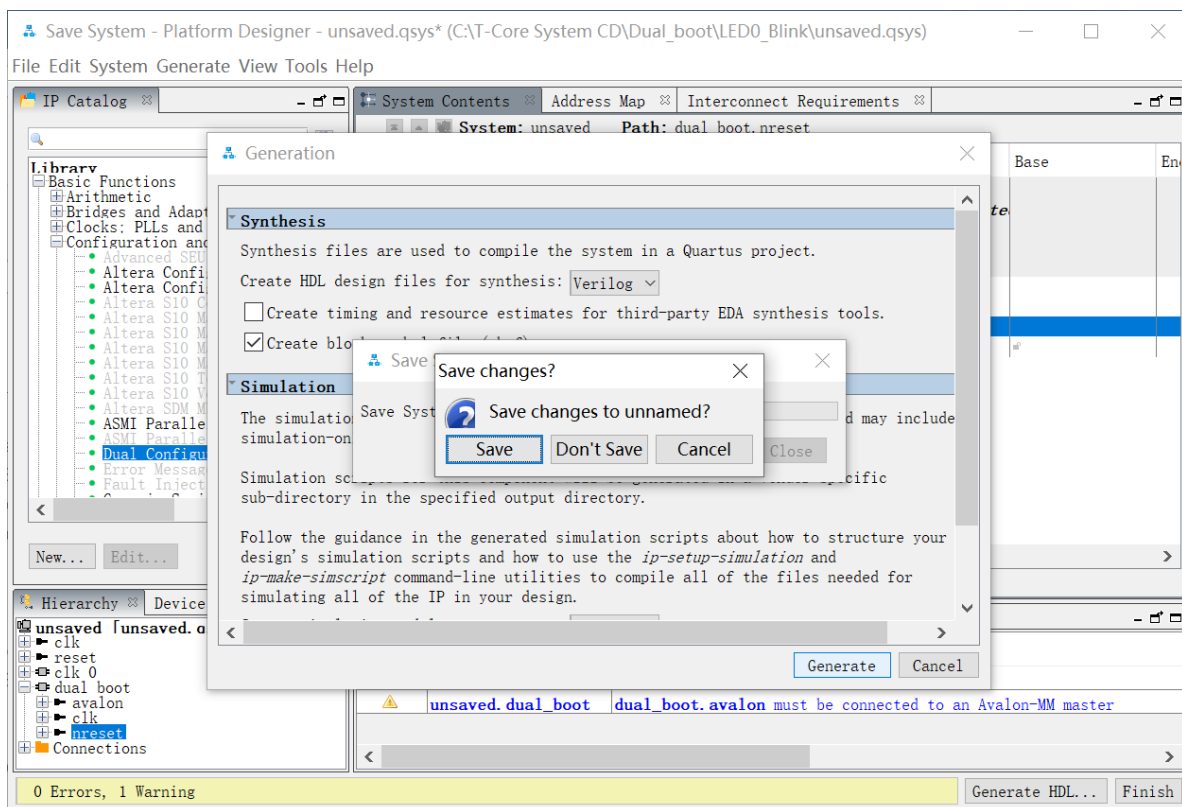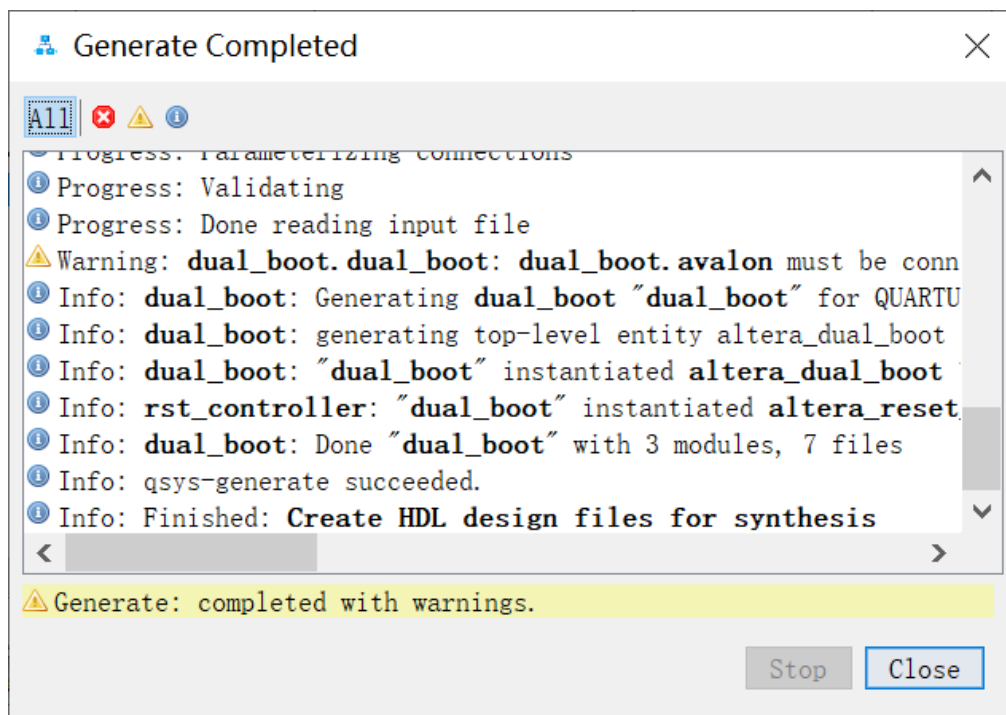
Figure 4-8 Generate and Save Qsys.

Figure 4-9 Generate completed.

5. Click **Close** and **Finish** to return to the Quartus window and add the dual_boot qsys into the top file as shown in **Figure 4-10**, and add the dual_boot.qip file to the project and save.

Figure 4-10 Add the dual_boot module in the top file.

Secondly, the project needs to be set before the compilation. After adding dual_boot IP successfully, please set the project mode as Internal Configuration mode, detailed steps are as follows:

1. Choose **Assignments > Device** to open Device windows shown in **Figure 4-11**.
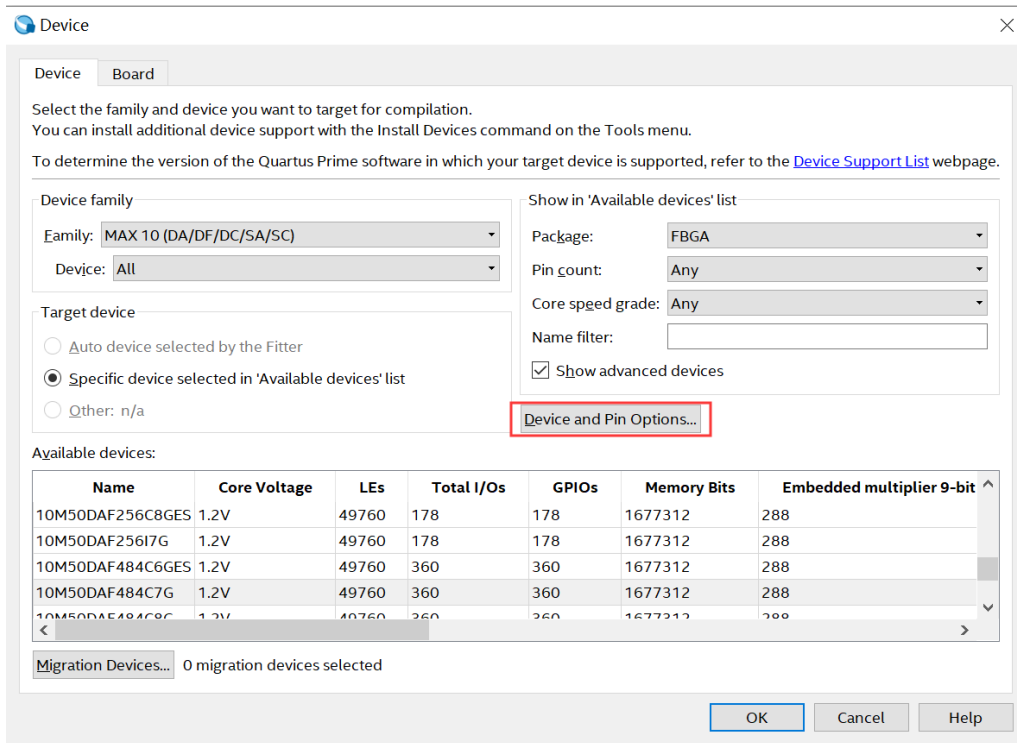

Figure 4-11 Open Device window…

2. Click **Device and Pin Opinions** to open the Device and Pin Opinions windows, and in the **Configuration** tab, Set the **Configuration Scheme** to **Internal Configuration** and the **Configuration Mode** to **Dual Compressed Images**. Check the Option of **Generate compressed bitstreams**. shown in **Figure 4-12**.
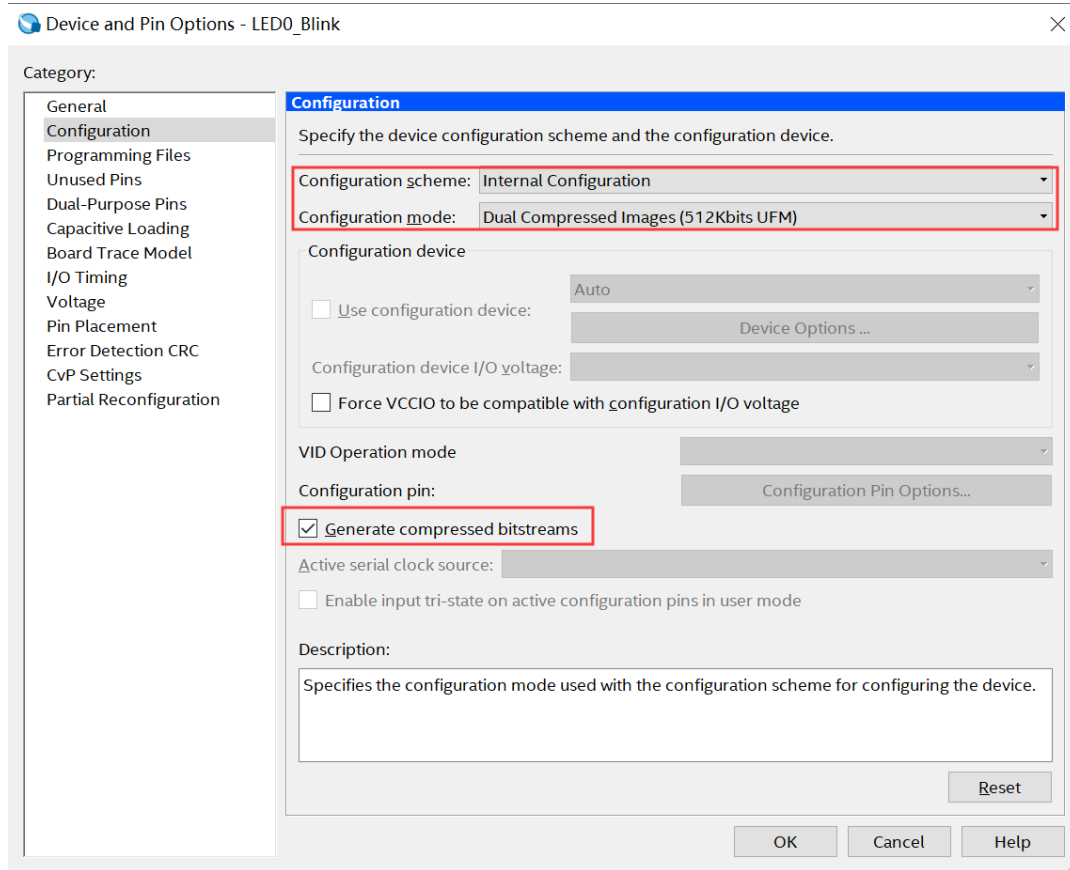
Figure 4-12 Set Dual Configuration Mode

3. Choose **OK** to return to the Quartus window. In the **Processing** menu, choose **Start Compilation** or click the Play button on the toolbar to compile the project, generate the new .sof file.

4. Use the same flow to add the Dual Configuration IP into other project **LED1_Blink** to generate the new .sof file by internal configuration mode.

Finally, so far, we have successfully obtained two image .sof files for dual boot demo according to previous steps. This section describes how to generate .pof from .sof files with the internal configuration mode and program the .pof into configuration flash memory (CFM) through the JTAG interface.

I.   **Convert .SOF File to .POF File**

1. Choose **Convert Programming Files** from the File menu of Quartus Prime to open new window, as shown in **Figure 4-13**.
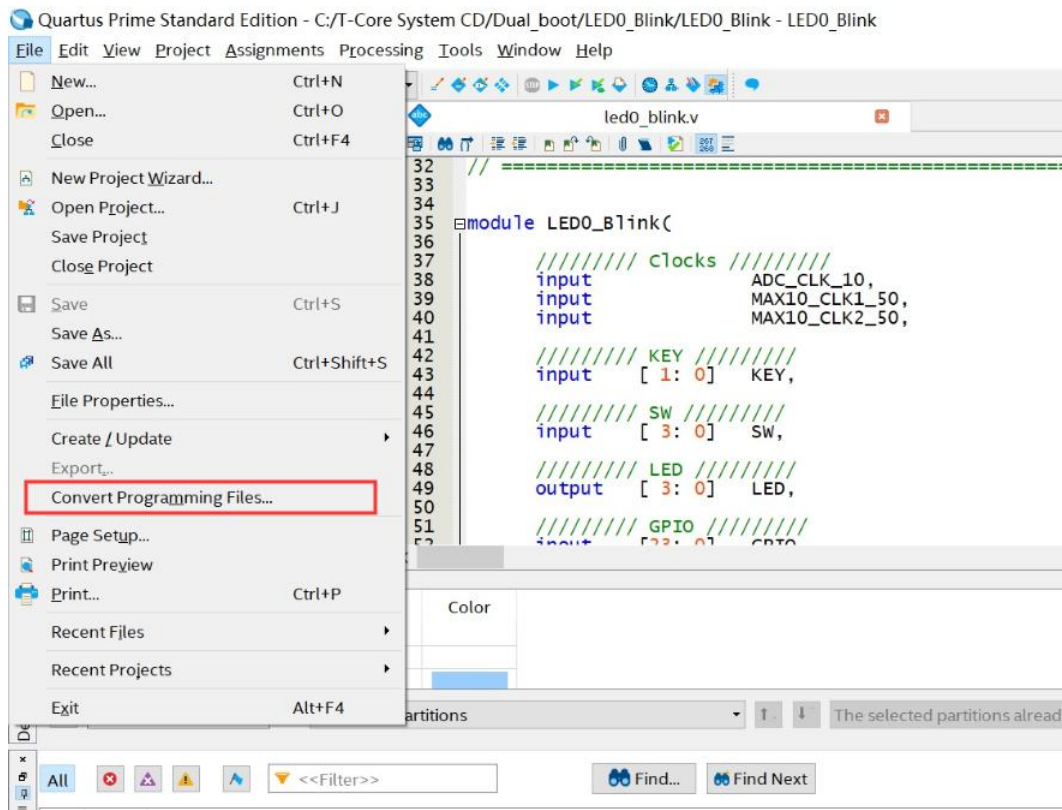
Figure 4-13 Select Convert Programming Files and click

2. Select **Programmer Object File (.pof)** from the **Programming file type** field in the dialog of Convert Programming Files.

3. Choose **Internal Configuration** from the **Mode** filed.

4. Browse to the target directory from the **File name** field and specify the name of output file.

5. Click on the **SOF data** in the section of **Input files to convert**, as shown in **Figure 4-14**.

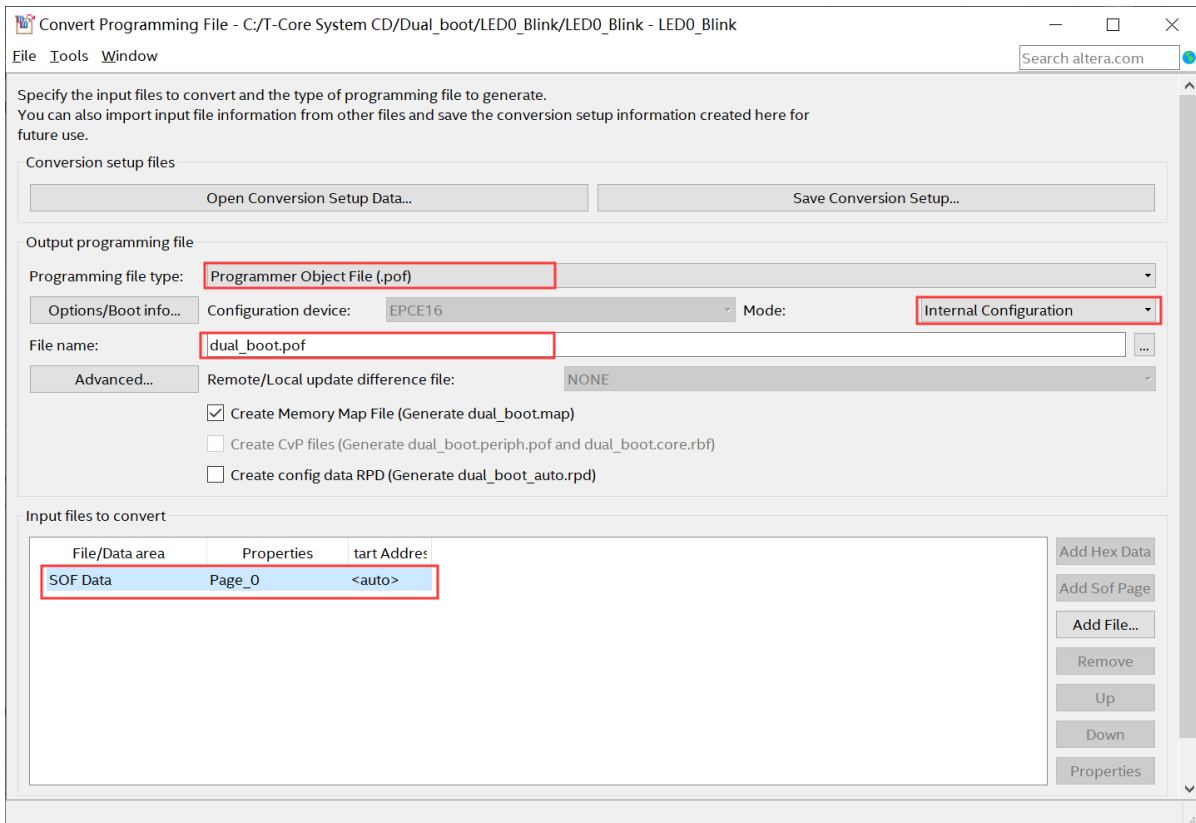**Figure 4-14 Dialog of Convert Programming Files and setting**

6. Click **Add File** and select the LED0_blink.sof to be the sof data of Page_0.

7. Click **Add Sof Page** to add Page_1 and click **Add File**, Select the LED1_blink.sof to be the .sof data of Page_1 as shown in **Figure 4-15**.

8. Click Generate.

These project files can be found in the CD directory: **\Demonstrations\Dual_boot**
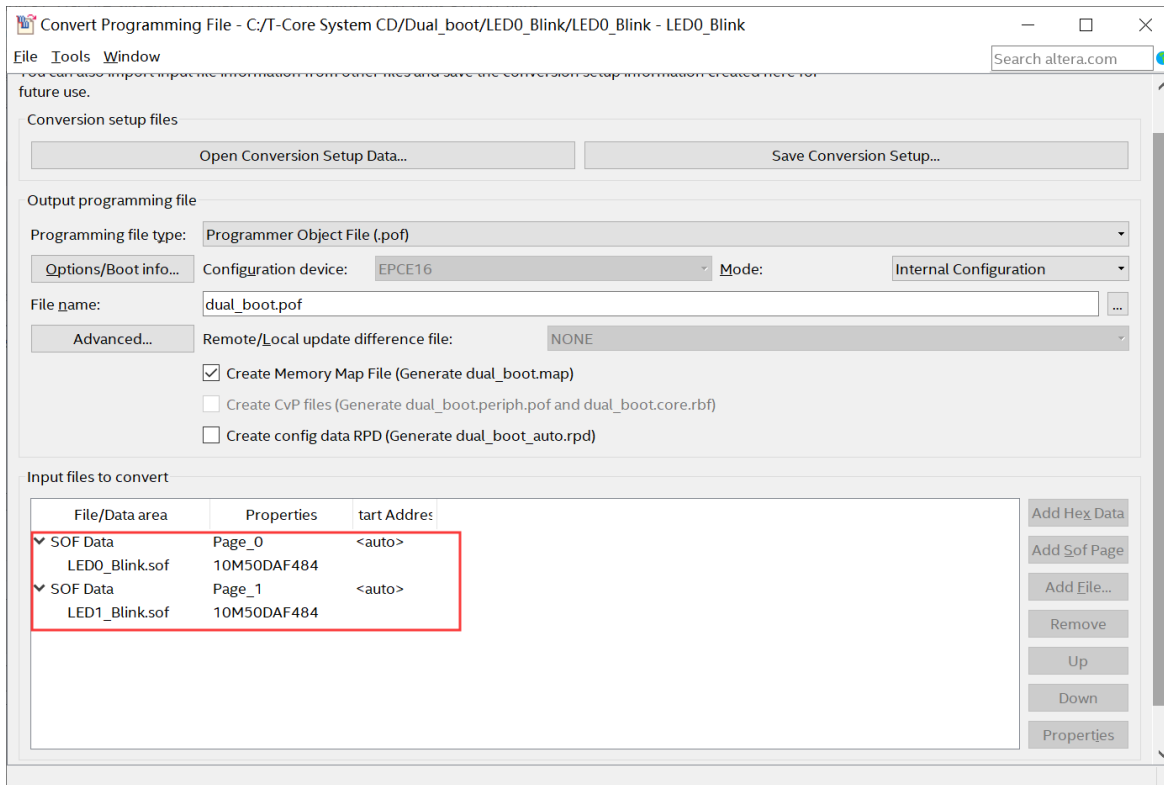
Figure 4-15 Add sof page and sof file

II. Write POF File into the CFM Device

When the conversion of SOF-to-POF file is complete, please follow the steps below to program the MAX 10 device with the .pof file created in Quartus Prime Programmer.

1. Choose **Programmer** from the **Tools** menu and the Chain.cdf window will appear.

2. Click **Hardware Setup** and then select the DECore[USB-*] as shown in **Figure 4-16**.

3. Click **Add File** and then select the dual_boot.pof.

4. Program the **CFM** device by clicking the corresponding **Program/Configure** and **Verify** box, as shown in **Figure 4-17**.

5. Click **Start** to program the **CFM** device.

Now, you can set the BOOT_SEL by JP2, you will find if you open JP2 (BOOT_SEL = 0), the LED0 Blink function would show. Power off the board, insert the jumper to JP2 (BOOT_SEL = 1), then power on, you would find the LED1 Blink function show.
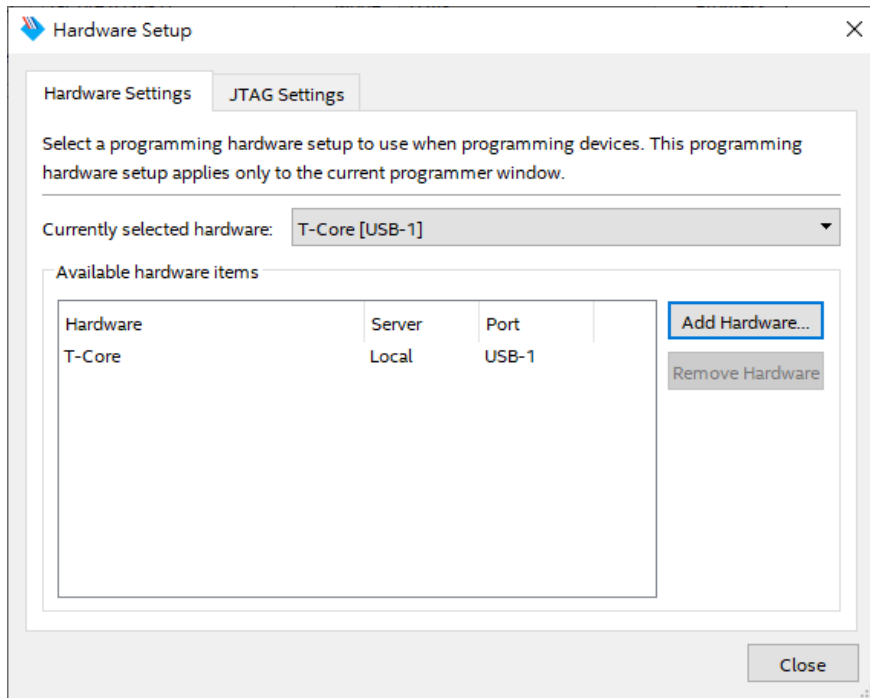
terasic
www.terasic.com

T-Core
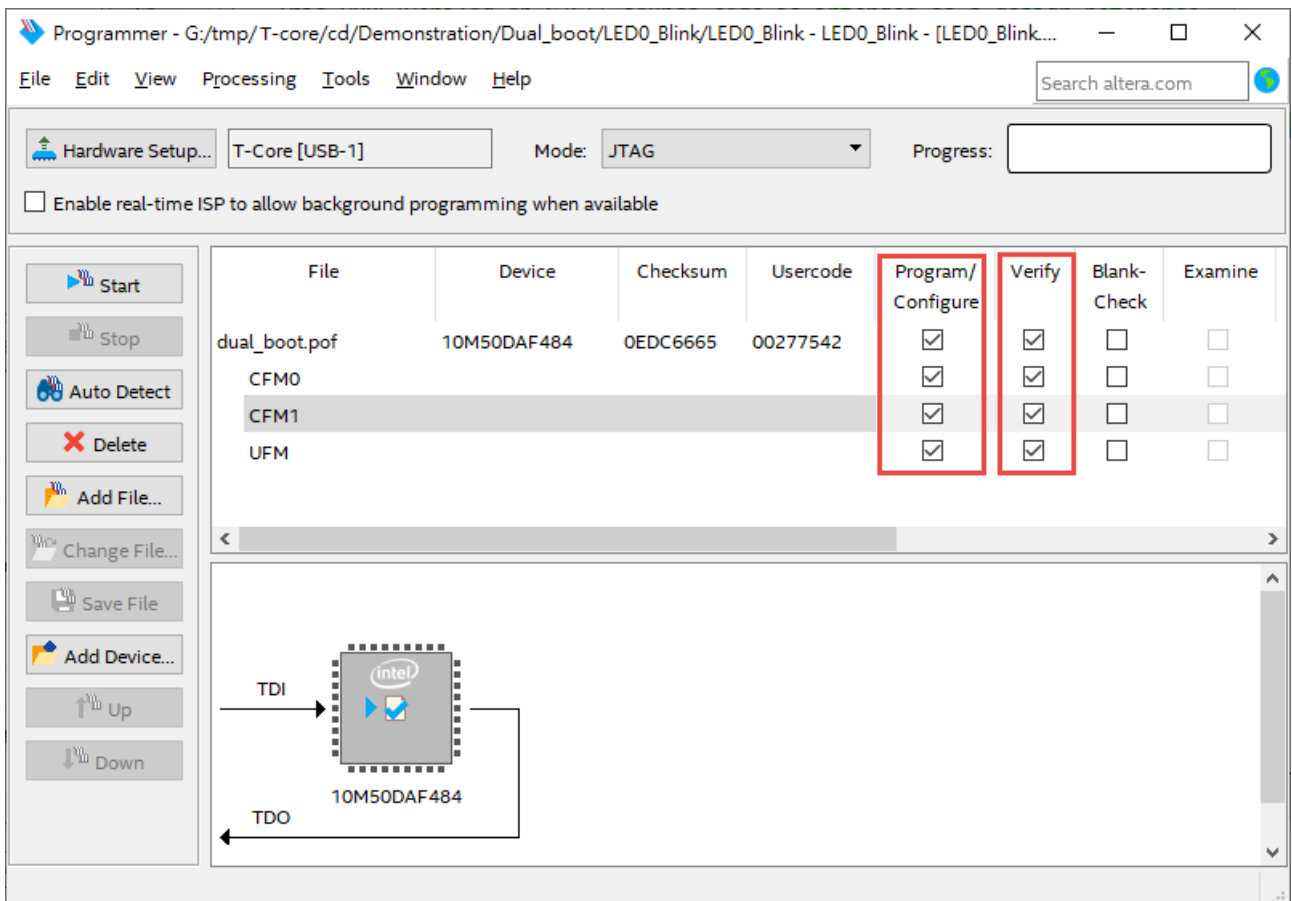User Manual

57

www.terasic.com
September 24, 2020

Figure 4-16 Hardware setup window

Figure 4-17 Programmer window with dual_boot.pof file

Chapter 5

USB Blaster II Cable Feature

The additional feature of T-Core is that it can be used as a USB blaster II to allow users to program other Intel FPGA boards from the host PC. As shown in **Figure 5-1**, users can connect to the external JTAG header (J1) of the T-Core using a flat ribbon cable. Then, use this cable to connect to other FPGA boards with 2x5 pin JTAG header for JTAG programming(See **Figure 5-2**).
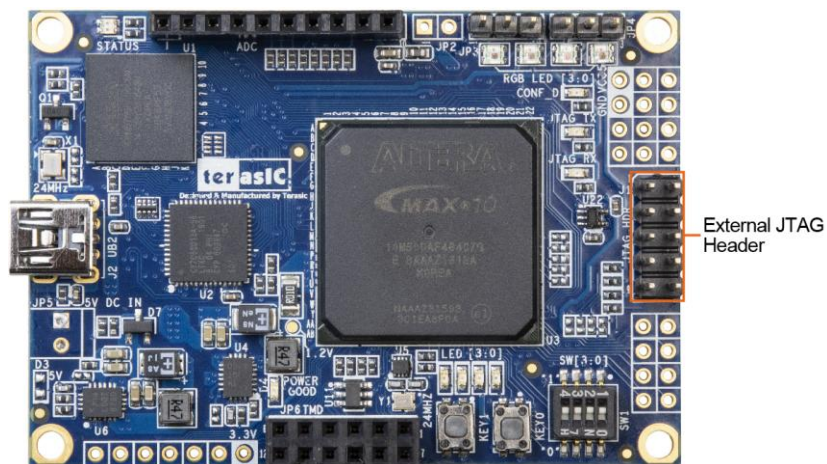

Figure 5-1 The external JTAG header of the T-Core board

Figure 5-2 Using T-Core board as an USB Blaster II cable

As shown in **Figure 5-3**, once the T-Core is connected to the JTAG interface of the external FPGA board, the T-Core board **automatically** detects the connection (detected the power of the target board) and switches the JTAG chain from the Intel MAX10 FPGA to the external JTGA header, and connected to the JTAG interface of the external FPGA board. Implement a USB blaster II function.
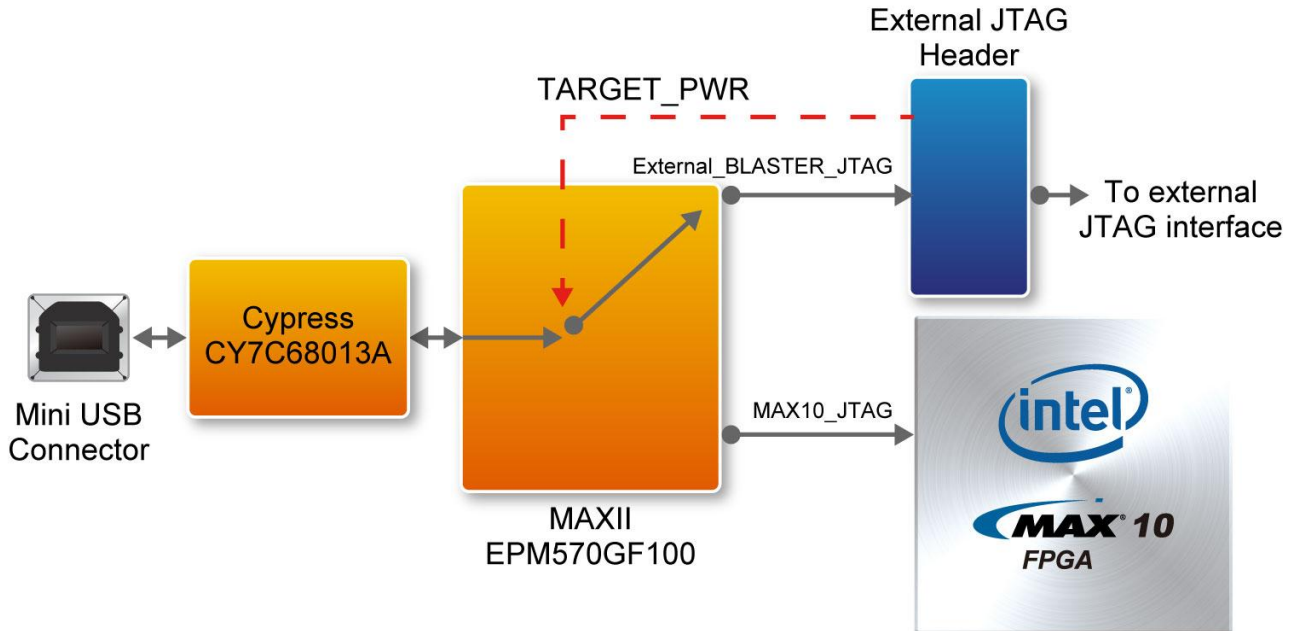

Figure 5-3 The auto-switch JTAG circuit of the T-Core board

In addition, users should note that for externally connected JTAG interface, **the T-Core can support I/O standard voltage range from 1.2V to 3.3V**. If it is out of this range, it may not be able to work properly or cause damage.

The following example demonstrates that use the T-Core to connect the JTAG interface of the Terasic's DE10-Pro Stratix 10 development board and program .sof file into the FPGA via JTAG interface.

1. Connect the flat ribbon cable to the J1 of the T-Core board, as shown in **Figure 5-4**. The red line represents the connection line of pin 1, which needs to be connected to the pin of J1.
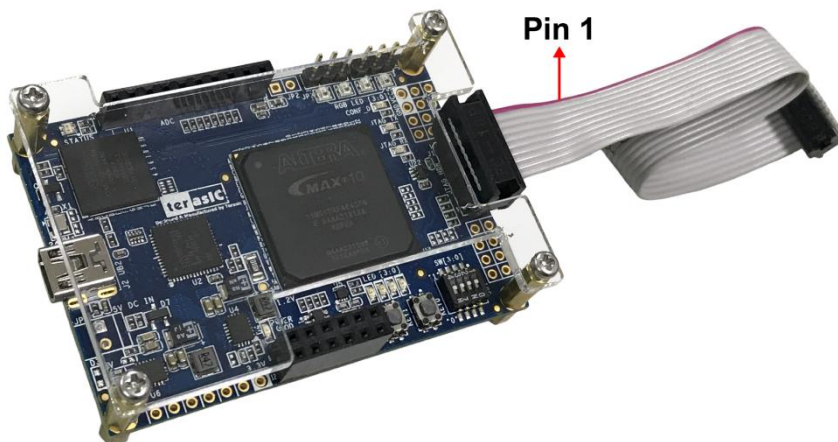
Figure 5-4 The flat ribbon cable connected to the T-Core board

2. Connect the other end of the flat ribbon cable to the 2x5 JTAG header on the DE10-Pro board, see **Figure 5-5**.
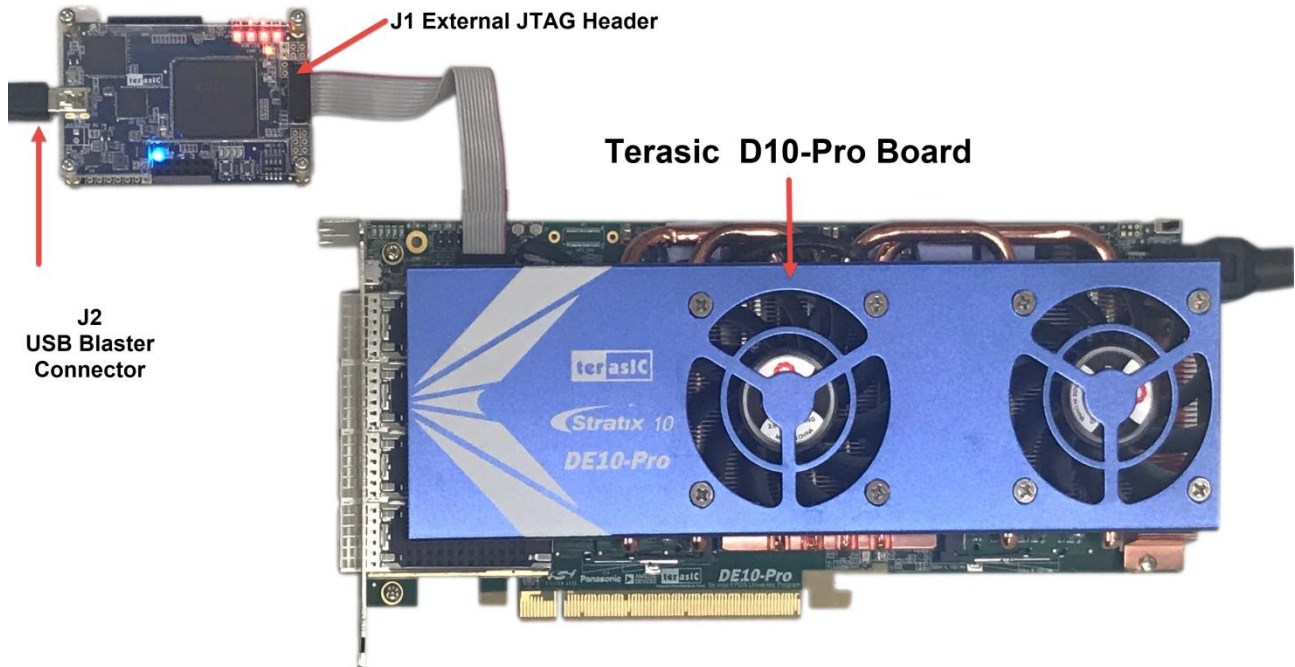

Figure 5-5 The flat ribbon cable connected to the T-Core board

3. Using mini USB cable to connected J2 of the T-Core board and the host PC.

4. Power the DE10-Pro board.

5. Open the Quartus programmer on the host PC, press "Hardware Setup" and select T-Core [USB-*]. Then press "Auto Detect" and the Stratix 10 FPGA device on the DE10-Pro will appear in the window, as shown in **Figure 5-6**.
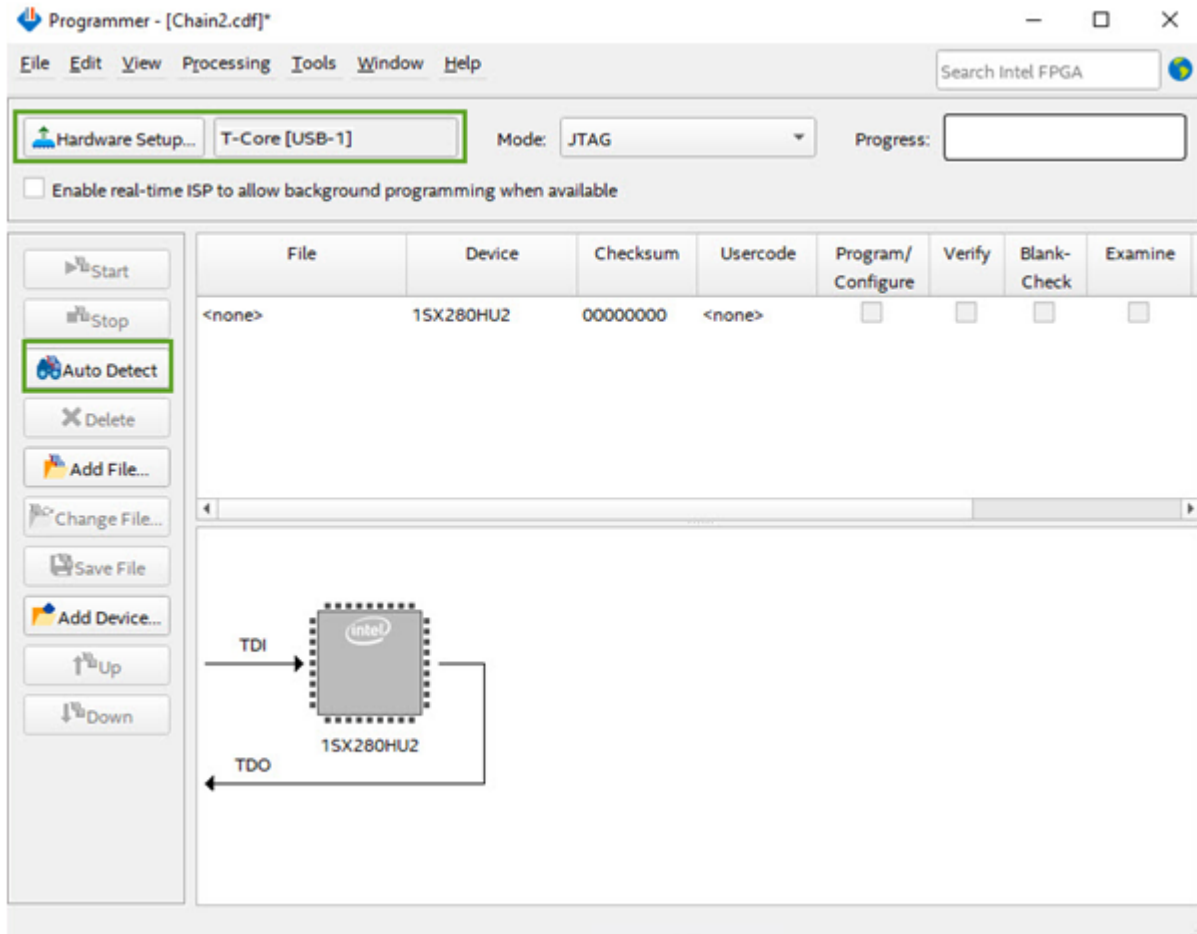
Figure 5-6 Using T-Core board to program the DE10-Pro

6. Select a .sof of the DE10-Pro project and download it. shows the success of the T-Core program Stratix 10 FPGA, as shown in **Figure 5-7**.
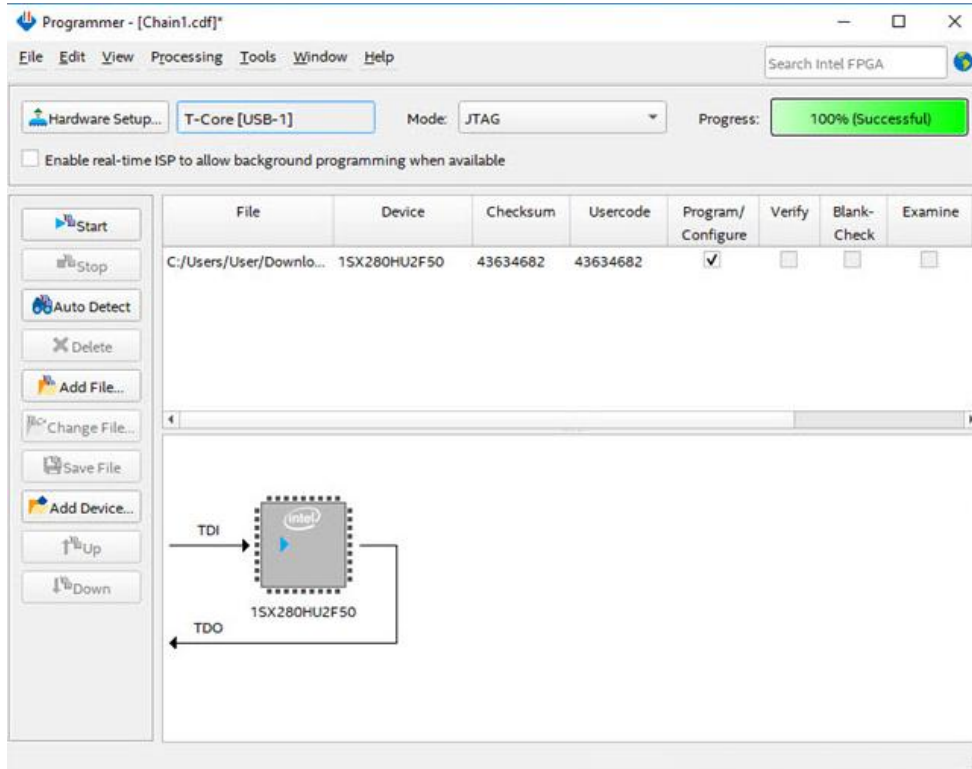
Figure 5-7 Using T-Core board to program the DE10-Pro

Additional Information

Revision History

Date	Version	Change Log
2019.07	V1.0	Initial Version (Preliminary)
2020.01	V1.1	• Modify the SW2 setting description for Figure 3-16. • Change blaster name from DE-Core to T-Core
2020.04	V1.2	Modify Figure 2-25
2020.05	V1.3	Update Figure 2-16 and Figure 1-4
2020.09	V1.4	Modify Figure 5-3

Copyright Statement

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Programmable Logic IC Development Tools *category:*

Click to view products by Terasic Technologies *manufacturer:*

Other Similar products are found below :

DK-DEV-5SGXEA7N SLG4DVKADV 88980182 DEV-17526 DEV-17514 LCMXO3L-SMA-EVN 471-014 80-001005 iCE40UP5K-MDP-EVN ALTHYDRAC5GX ALTNITROC5GX 471-015 Hinj SnoMakrR10 DK-DEV-1SDX-P-A DK-DEV-1SDX-P-0ES DK-DEV-1SGX-L-A DK-DEV-1SMC-H-A DK-DEV-1SMX-H-0ES DK-DEV-1SMX-H-A DK-DEV-4CGX150N DK-DEV-5CGTD9N DK-DEV-5CSXC6N DK-DEV-5M570ZN DK-MAXII-1270N DK-SI-1SGX-H-A DK-SI-1STX-E-0ES DK-SI-1STX-E-A DK-SI-5SGXEA7N ATF15XX-DK3-U SLG46824V-DIP SLG46826V-DIP 240-114-1 6003-410-017 ICE40UP5K-B-EVN DK-SOC-1SSX-L-D ICE5LP4K-WDEV-EVN L-ASC-BRIDGE-EVN LC4256ZE-B-EVN LCMXO2-7000HE-B-EVN LCMXO3D-9400HC-B-EVN LCMXO3L-6900C-S-EVN LF-81AGG-EVN LFE3-MEZZ-EVN LIF-MD6000-ML-EVN LPTM-ASC-B-EVN M2S-HELLO-FPGA-KIT VIDEO-DC-USXGMII 12GSDIFMCCD SFP+X4FMCCD