

ELEVERENCE C

00000000

0000000

# USER MANUAL



Copyright © Terasic Inc. All rights reserved.

# CONTENTS

Chapter 1	Introduction	
1.1 Package Conte	nts	2
1.2 T-Core System	CD	3
1.3 Layout and Con	nponents	3
1.4 Block Diagram	of the Board	5
1.5 Getting Help		6
Chapter 2	Using the T-Core	7
2.1 Configuration	of MAX 10 FPGA on T-Core	7
2.2 Clock Circuitry	r	
2.3 Using the Push	-buttons, Switches and LEDs	
2.4 Using WS2812	B RGB LED	
2.5 Using 2x6 TMI	O GPIO Expansion Headers	21
2.6 1x10 A/D Conv	verter and Analog Input	
2.7 Using QSPI Fla	1sh	
Chapter 3	Example Designs	
3.1 T-Core Factory	Configuration	
3.2 Control RGB L	ED in Nios II	
3.3 Control Extern	al RGB LED in Nios II	
3.4 ADC Measurer	nent	
3.5 RISC-V		
Chapter 4	Programming the Configuration Flash Memory	
4.1 Internal Config	uration	47
4.2 Using Dual Co	mpressed Images	
Chapter 5	USB Blaster II Cable Feature	



# Chapter 1



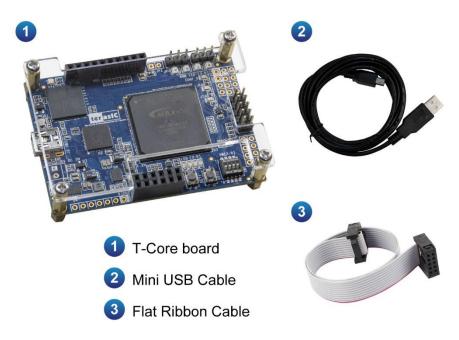
The T-Core presents a robust hardware design platform built around the Intel MAX 10 FPGA. The MAX 10 FPGA is well equipped to provide cost effective, single-chip solutions in control plane or data path applications and industry-leading programmable logic for ultimate design flexibility. With MAX 10 FPGA, 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.

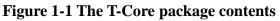
The T-Core development board includes hardware such as on-board USB Blaster II, QSPI Flash, ADC-Header, WS2812B RGB LEDs and 2x6 TMD expansion header. By leveraging all of these capabilities, the T-Core is the perfect solution for showcasing, evaluating, and prototyping the true potential of the Intel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







T-Core User Manual

The T-Core package includes:

- The T-Core board
- USB Mini-B Cable
- Flat Ribbon Cable

# **1.2 T-Core System CD**

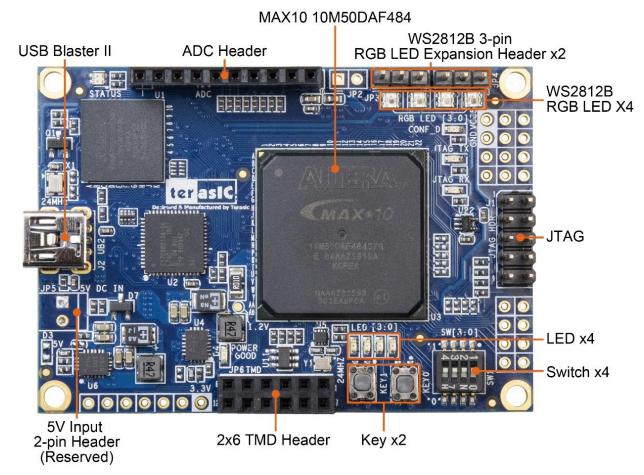
The T-Core System CD contains the documentation and supporting materials, including the User Manual, reference designs and device datasheets.

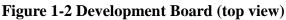
User can download this System CD from the web (<u>http://T-Core.terasic.com/cd</u>).

# **1.3 Layout and Components**

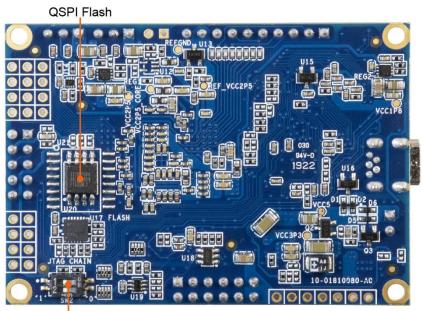
This section presents the features and design characteristics of the board.

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.









MAX10/RISC-V JTAG Config

#### 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  $18 \times 18$  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

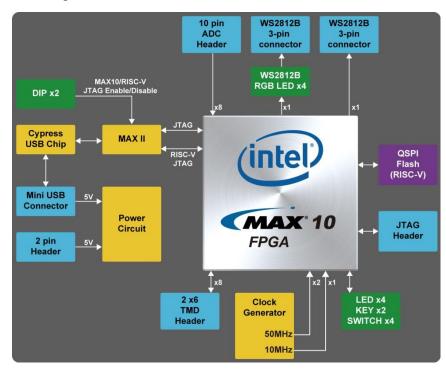
- •4 Green LEDs
- •4 Dip Switches
- •2 Push Buttons with Debounced.
- •4 WS2812B RGB LED

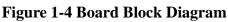
### **Power**

• 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.







# **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: <u>http://T-Core.terasic.com</u>



# Chapter 2



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.



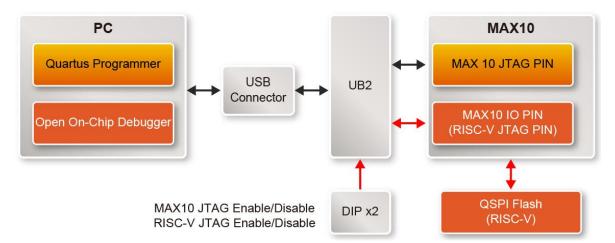


Figure 2-1 The JTAG configuration scheme

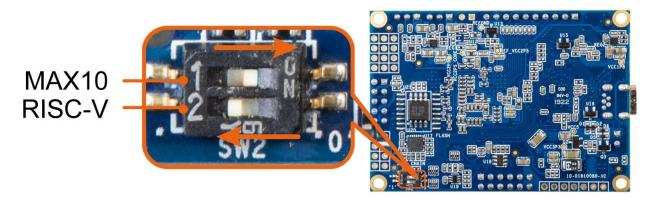


Figure 2-2 The JTAG Switch for MAX10 FPGA and RISC-V

**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.

SW2 Position		Description
MAX10 (SW2.1)	RISC-V (SW2.2)	Description
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

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



### **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**.

	tmp/T-core/cd/Demon rocessing <u>T</u> ools <u>W</u> ir		one_boldar 1		( [[]]00R2]0		rch altera.c	om (
📥 Hardware Setup	T-Core [USB-1] P to allow background pr	Mode:		•	Progress:			
▶ <sup>™</sup> Start ■ <sup>™</sup> Stop	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine
Auto Detect								
Hadd File	<							
Add Device								
ி <sup>ங்</sup> Up <sup>புங்</sup> Down								

#### 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**.



Hardware Setup						×
Hardware Settings	JTAG Settings					
Select a programmin hardware setup appli				evices. T	his programming	
Currently selected ha		[USB-1]			-	
Hardware		Server	Port		Add Hardware	
T-Core		Local	USB-1		Remove Hardware	
					Close	

Figure 2-4 Hardware Setting

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

Programmer - G:/	/tmp/T-core/cd/Demonstr	ation/Default/T_C	DRE_Default - T	_CORE_Defau	lt - [T_CORE_E	efau	_	
<u>F</u> ile <u>E</u> dit <u>V</u> iew P	rocessing <u>T</u> ools <u>W</u> indo	w <u>H</u> elp				Sear	rch altera.c	om 🌀
🚖 Hardware Setup	T-Core [USB-1]	Mode: .	ITAG	•	Progress:			
Enable real-time IS	P to allow background prog	ramming when ava	ilable					
▶ <sup>₩</sup> Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine
Stop								
X Delete								
Change File	<							>
Save File								
Add Device								
1 <sup>10</sup> Down								

#### Figure 2-5 Detect FPGA device in JTAG mode

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



T-Core User Manual

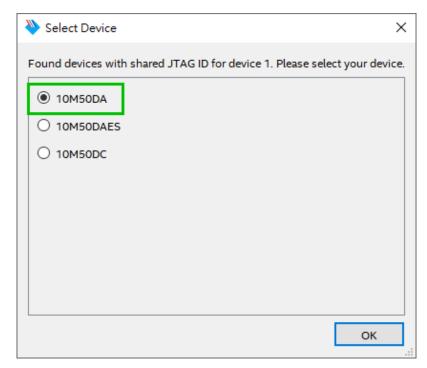
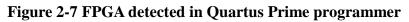


Figure 2-6 Select 10M50DA device

6. FPGA is detected, as shown in **Figure 2-7**.

👋 Programmer - G	:/tmp/T-core/	cd/Demons	tration/Default/T_C	ORE_Default -	T_CORE_Defau	lt - [T_CORE_[	Defau	_	o x
<u>F</u> ile <u>E</u> dit <u>V</u> iew	P <u>r</u> ocessing <u>T</u>	ools <u>W</u> ine	dow <u>H</u> elp				Sea	rch altera.c	om 🌖
🚖 Hardware Setup	T-Core [USI	8-1]	Mode:	JTAG	•	Progress:			
Enable real-time	ISP to allow bac	kground pro	ogramming when ava	ailable					
<sup>▶</sup> <sup>™</sup> Start	File	e	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine
Stop	<none></none>		10M50DA	0000000	<none></none>				
💏 Auto Detect									
🗙 Delete									
🏓 Add File									
Change File	<								>
💾 Save File									^
P Add Device		(intel) 10M50D/							~





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.

Start   Save File   Save File   Add IPS File Change File   Change File   Add Device   Mod Device   TDI   Image File   Show Device's Properties   Show Device's Properties   Save File   Attach Flash Device   Delete Flash Device   Delete Flash Device					×	<u>D</u> elete	Del		
File Device Change File   Stop File Device   Change File Add IPS File   Auto Detect Add File   Change File Add EKP File   Change File Add EKP File   Change File Add EKP File   Change File Add Programming File   Change File Change File   Add Device TDI   TDI Edit   Show Device's Properties Change Flash Device   Delete Flash Device Delete Flash Device	Hardware Setup.	T-Core [USB-1]	Mode:	JTAG		Select <u>A</u> ll	Ctrl+A		
File Device Charge File   Stop Save File   Auto Detect   Auto Detect   Add File   Change File   Add File   Change File   Add Device   TDI   Edit   Show Device's Properties     Change Flash Device   Change Flash Device	Enable real-time I	SP to allow background	programming when a	vailable 🥇	Wn.	Add File			
Start show Device's Properties				L	- A	Change File			
Auto Detect X Delete Add File Change File Change File Save File Add Device TDI Edit 10ME Show Device's Properties Change Flash Device Change Flash Device Change Flash Device Change Flash Device Delete Flash Device Delete Flash Device	▶ <sup>1</sup> Start	File	Device	Checks	4	Save File		k- ck	Examin
Auto Detect   X Delete   Add File   Change File   Save File   Add Device   TDI   Edit   10ME   Show Device's Properties      Delete IPS File Add EKP File Add EKP File Delete IPS File Add EKP File Delete EKP File Add PR Programming File Change PR Programming File Delete PR Programming File Change Flash Device Delete Flash Device Delete Flash Device	Stop	<none></none>	10M50DA	0000000		Add IPS File			
Add File   Change File   Save File   Add Device   TDI   Edit   10Ms   Show Device's Properties						Change IPS File			
Add EKP File   Add File   Change File   Save File   Add Device   TDI   Edit   10Ms   Show Device's Properties     Add EKP File   Change EKP File   Add PR Programming File   Change PR Programming File   Delete PR Programming File   Attach Flash Device   Change Flash Device	Auto Detect					Delete IPS File			
Add Frie     Change File       Save File     Add Device       Multiple     Image Edit       Change File     Change PR Programming File       Change PR Programming File     Delete PR Programming File       Change PR Programming File     Attach Flash Device       Change Flash Device     Change Flash Device	🗙 Delete					Add EKP File			
Change File   Save File   Add Device   TDI   Edit   10Ms   Show Device's Properties	Add File					Change EKP File			
Change File   Save File   Add Device   TDI   Edit   10Ms   Show Device's Properties     Add PR Programming File   Change PR Programming File   Change PR Programming File   Delete PR Programming File   Attach Flash Device   Change Flash Device   Delete Flash Device		<		_		Delete EKP File			
Add Device       TDI       Image: Change PR Programming File         TDI       Edit       Change PR Programming File         Down       Edit       Change PR Programming File         10Ms       Show Device's Properties       Delete Flash Device         Delete Flash Device       Delete Flash Device	Change File					Add PR Programming File		F	
Add Device     Delete PR Programming File       The up     Image: Edit     Attach Flash Device       Image: Delete PR Programming File     Change Flash Device       Image: Delete PR Programming File     Delete PR Programming File	Save File								
TDI     Edit     Attach Flash Device       Down     10ME     Show Device's Properties     Delete Flash Device	Add Davisa	7.000000					-		
Image: Constraint of the second se		TDI				Delete PK Programming File			
10M5 Show Device's Properties Delete Flash Device	1 <sup>ሜ</sup> Up	<b>→</b>	¥			Attach Flash Device			
10M5 Show Device's Properties Delete Flash Device	I'll Davie		<u>E</u> dit	•		Change Flash Device			
	+ - Down		Show Device's P	roperties		Delete Flash Device			
TDO Add Device		TDO		· · · ·	•				
		ogramming file				Change Device			

#### 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**.



Select New Programming File	×
Look in: G:\tmp\ T -core\cd\Demonstration\Default\output_files  G G G My Comp User T_CORE_Default.sof	
File name: T_CORE_Default.sof	Open
Files of type:     Programming Files (*.sof *.pof *.jam *.jbc *.ekp *.jic)	Cancel

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**.

👋 Programmer - G	i:/tmp/T-core/cd/Demon	stration/Default/T_C	ORE_Default -	T_CORE_Defau	ult - [T_CORE_	Defau	_	
<u>F</u> ile <u>E</u> dit <u>V</u> iew	P <u>r</u> ocessing <u>T</u> ools <u>W</u> in	dow <u>H</u> elp				Sea	rch altera.c	om 🌖
🚖 Hardware Setup	T-Core [USB-1]	Mode: .	JTAG	•	Progress:			
Enable real-time	ISP to allow background pr	ogramming when ava	ilable					
▶ <sup>™</sup> Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine
<sup>™</sup> Stop	output_files/T_CORE	10M50DAF484	00410D18	00410D18				
💏 Auto Detect								
🗙 Delete								
Add File								
隆 Change File	<							>
Save File								^
Add Device								v

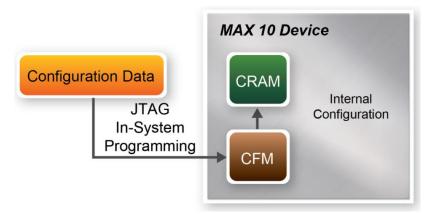
Figure 2-10 Program .sof file into the FPGA device



### 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.

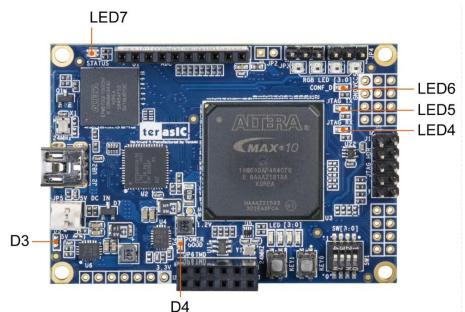
• Please refer to **Chapter 4: Programming the Configuration Flash Memory (CFM)** for the basic programming instruction on the configuration flash memory (CFM).





### 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.



U-

Figure 2-12 Status LED position



T-Core User Manual

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.

#### Table 2-2 Status LED

# **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.

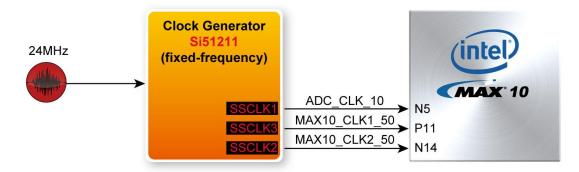


Figure 2-13 Clock circuit of the FPGA Board

		-5 I in 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

#### **Table 2-3 Pin Assignment of Clock Inputs**



# **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.

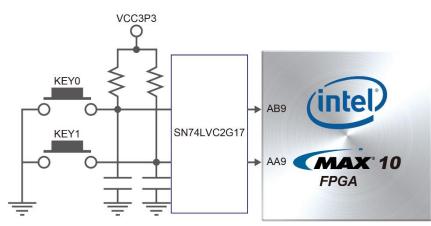


Figure 2-14 Connections between the push-button and MAX 10 FPGA

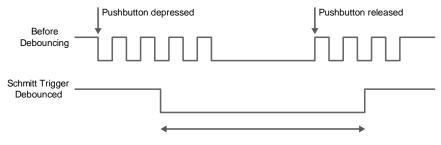


Figure 2-15 Switch debouncing

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.

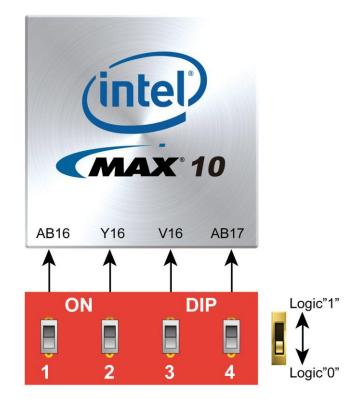


Figure 2-16 Connections between the slide switches and MAX 10 FPGA

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.



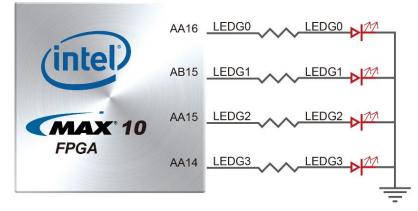


Figure 2-17 Connections between the LEDs and MAX 10 FPGA

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

 Table 2-6 Pin Assignment of LEDs

## 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.

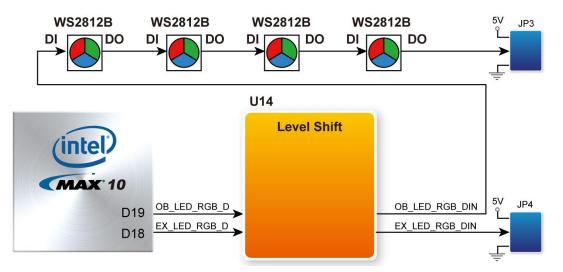


Figure 2-18 Level Shift between the REG LEDs and MAX 10 FPGA



### Control Method

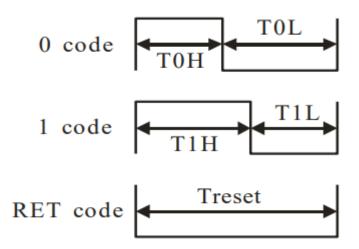
For the WS281B RGB LED, each R/G/B color can be independently controlled, and each color can be adjusted from level 0 to 255. Therefore, each color needs to use 8bit control, and each LED needs 24bit control. The 24 bits corresponds to the **Figure 2-19**.

#### Composition of 24bit data:

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**.



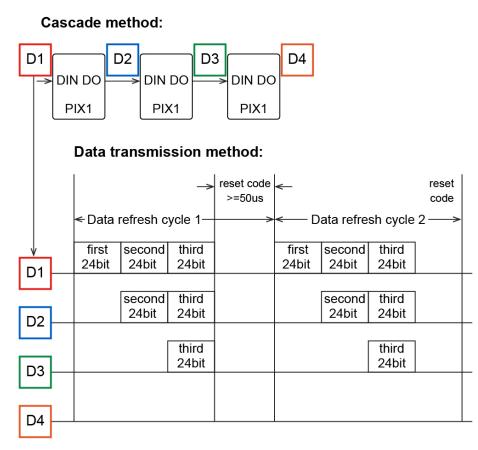
#### Figure 2-20 Sequence Chart

#### **Table 2-7 Code Timing**

Code Status Description		Timing
ТОН	0 code, high voltage time	220ns~380ns
T1H	1 code, high voltage time	580ns~1µs
TOL	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.





#### 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:



T-Core User Manual

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**.

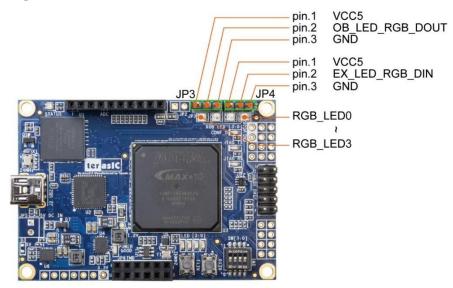


Figure 2-22 JP3 / JP4 and RGB LED Location

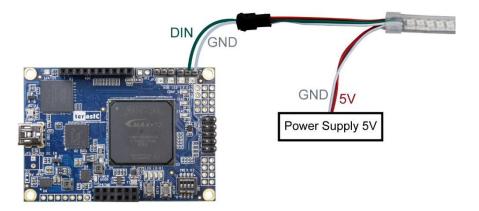


Figure 2-23 An example of a color light bar connection for an external power supply

# **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.



T-Core User Manual

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.

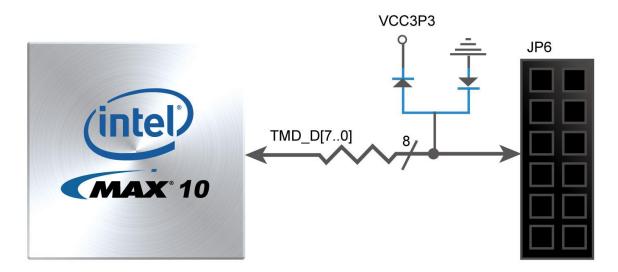


Figure 2-24 Connections between the 2x6 TMD header and MAX 10 FPGA Table 2-9 Pin Assignment of 2x6 TMD Header.

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.



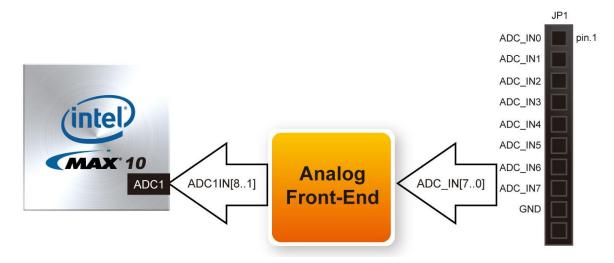


Figure 2-25 Connections between the Arduino Analog input and MAX 10 FPGA

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

Table 2-10 Fill 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

Table 2-1	0 Pin A	Assignment	of A	nalog ]	Input

# 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



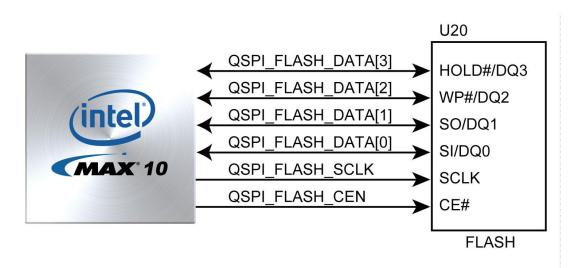


Figure 2-26 Connections between MAX 10 FPGA and QSPI Flash

		8	
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

#### Table 2-11 Pin Assignment of QSPI Flash



# Chapter 3

# 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.



MAX 10 50Mhz PLL 100 50Mhz Color_selection COLOR[23:0] COLOR_Processe	ram_wrclock       ram_data[23:0]       ram_wraddress[n-1:0]       update_reqeust       update_done	ILED_CONTROLLER
LED_flicker reset_n		

#### 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

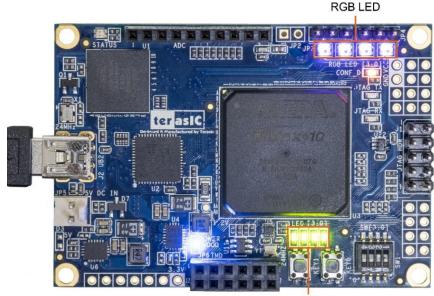
• Connect the Mini USB connector (J2) of the T-Core board to the host PC with a USB cable and install the USB-Blaster II driver if necessary.

 $\bullet$  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.



Green LED

#### Figure 3-2 On-Board Green LED and RGB LED display



T-Core User Manual

### 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.

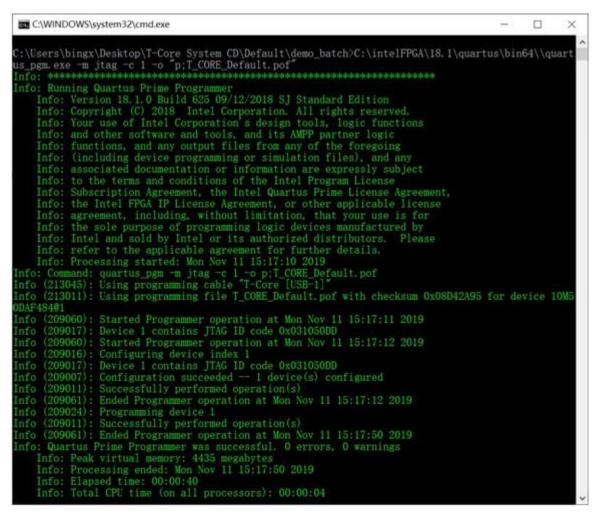


Figure 3-3 Quartus Prime Program MAX 10 CFM message

# **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.

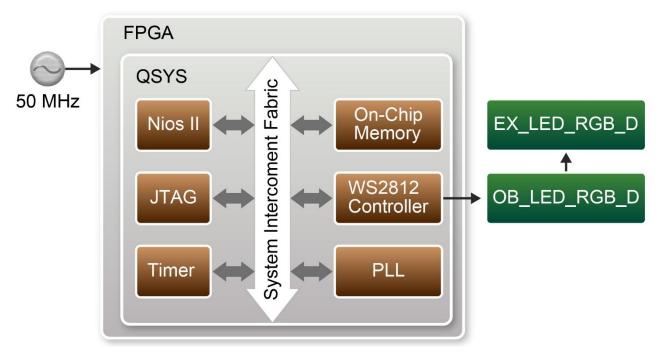


Figure 3-4 Block diagram of the SDRAM test in Nios II

# 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**

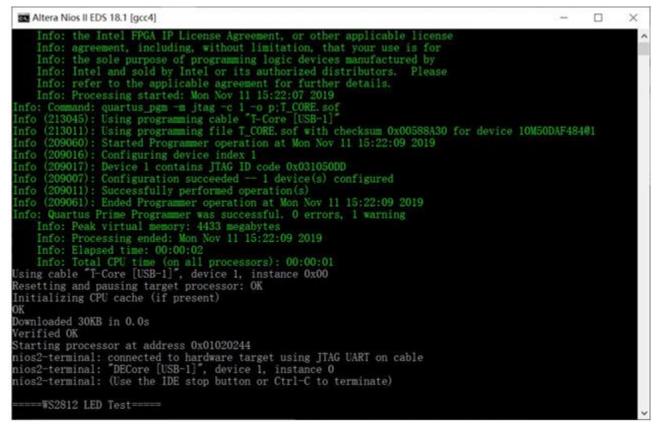
• Quartus Prime v18.1 and Nios II v18.1 must be pre-installed on the host PC.

• 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.



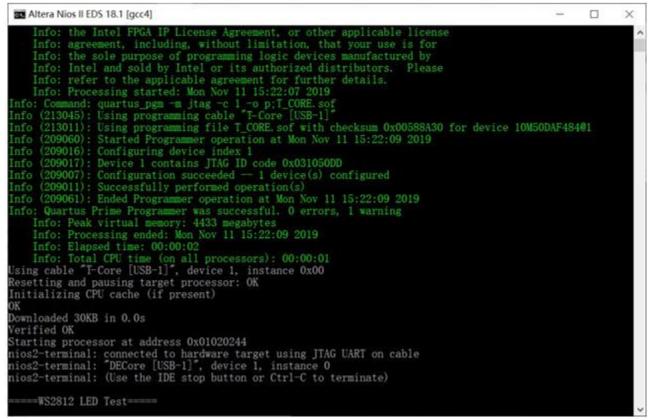


Figure 3-5 Display the progress and the result for the WS2812 test in Nios II



Figure 3-6 The RGB LEDs display after executing the batch file



T-Core User Manual

# **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

- $\bullet$  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.

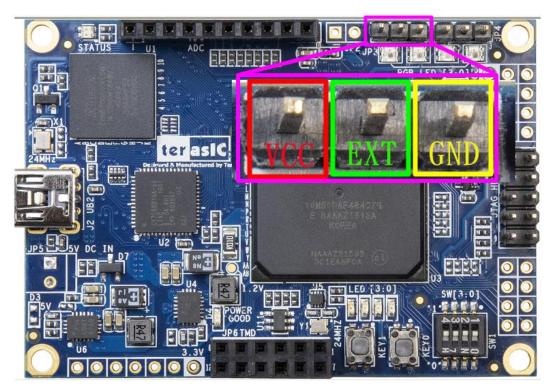


Figure 3-7 External RGB LED connection diagram

# **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



T-Core User Manual

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

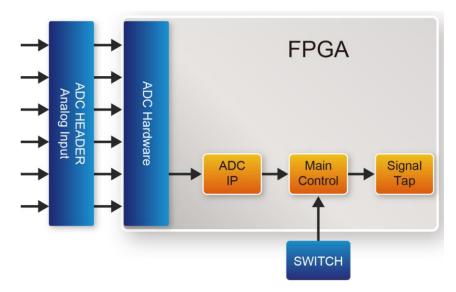


Figure 3-8 Block diagram of the ADC measurement demonstration

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**.

Name	9 <u>32</u>		
	01h		
response_valid			
	3931	3933	
	2399	2400	



#### Figure 3-9 Signal Tap of ADC

### Design Tools

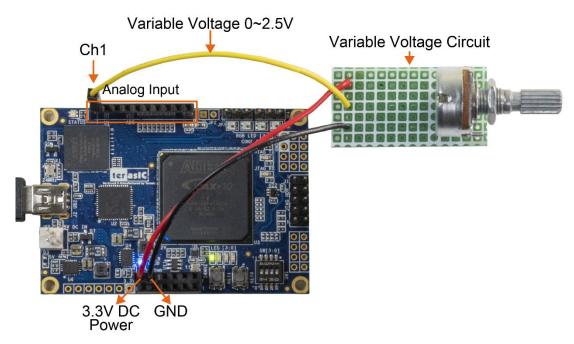
• Quartus Prime 18.1 Standard Edition

## **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.

ile gdt	Yew Processing	I Toola Help										Search altero	com
N	( 主動 )												
nstance Man	uger: 🔁 횐 🛛	🛙 🛄 Ready to act	quire						×	JTAG Cha	n Configuration	T. JTAG ready	
stance		Status	Enabled	LEs: 839	Memory: 61440	Small: NA	Medium: NA	Large: NA		Hardware	T-Core (USB-1	1 -	Setup
auto_s	signaltap 0	Not running	12	839 cells	61440 bits	NA .	NA	NA.					
										Device >> SOF	Construction	LIESI/10MSOC ·	Rectange Lands
los Tra Ø		40 (0:0:0.2 elapsed) #3	2				tick to a	ier line bar		-	Construction		Reconstruction of the
iog Trig @ Type Allas	2019/06/26 17:04	40 (0100.2 elapsed) #: Name	2		.44	28 190	tick to e 192 224			-	Manager 🕌		COREsol
Type Alias	2019/06/26 17:04	Name	_	32	64. <u>96</u> . 1		192 224	set line bar		50F	Manager 🕌	A Mes/OE	CORE sol
Type Alias	2019/08/28 1704 8	Name	_		64		192 224	Lett Serie Laur 256 281		50F	Manager 🕌	A Mes/OE	
Type Alias	2019/06/28 1704 5 1 cur_adc_ch[4	Name _0] data[11_0]	_	L 1 1 2092 L	<u> </u>	28	192 224	Let time tur 256 200 10 1091 1 2093 1 209	1	50F	Manager 🕌	A Mes/OE	490

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.



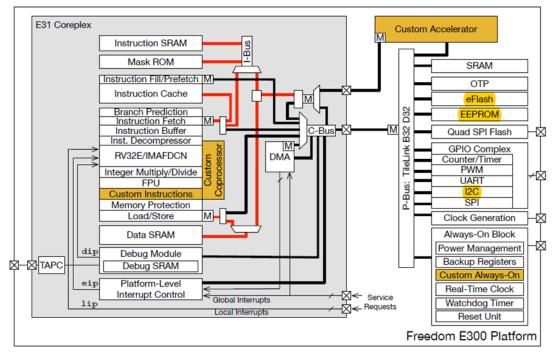


Figure 3-12 Top-Level Block Diagram of the E300 platform

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

	Tuble e 2 h	remory map of Freedom E500	
Base	Тор	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	<b>On-Chip Peripherals</b>
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
0x4000_0000	0x7FFF_FFFF	Additional I/O or RAM	Non-Volatile
			Memory
0x8000_0000	0x8001_FFFF	Instruction and Data RAM	On-Chip Volatile
0x8002_0000	0xFFFF_FFFF	Additional RAM	Memory

### Table 3-2 Memory map of Freedom E300



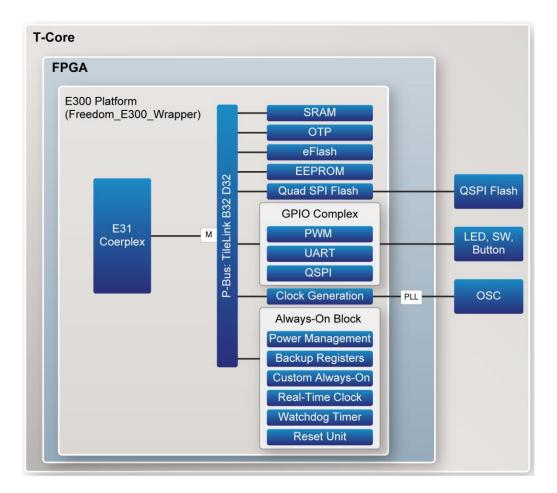
### About Quartus Project

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.

∃Freedom_E300_wrapper E300_ /*input wire */ /*input wire */	SoC_Wrapped( .clock .aon_lfextclk	(clk_32M), (slowclk),
/*input wire */ /*input wire */ /*output wire */	.aon_erst_n .jtag_reset .io_ndreset	<pre>(~ip_reset_sys_peripheral_reset), (fpga_power_on_power_on_reset), (ip_reset_sys_mb_debug_sys_rst),</pre>
/*input wire */	.jtag_tck	(RISCV_JTAG_TCK),
/*input wire */	.jtag_tms	(RISCV_JTAG_TMS),
/*input wire */	.jtag_tdi	(RISCV_JTAG_TDI),
/*output wire */	.jtag_tdo	(RISCV_JTAG_TDO),
/*output wire */	.qspi_sck	(QSPI_FLASH_SCLK),
/*output wire */	.qspi_cs_0	(QSPI_FLASH_CE_n),
/*inout wire [3:0] */	.qspi_dq	(QSPI_FLASH_DATA),
/*input wire [31:0] */	.e300_gpio_i_ival	(e300_gpio_i_ival),
/*output wire [31:0] */	.e300_gpio_o_ie	(e300_gpio_o_ie),
/*output wire [31:0] */	.e300_gpio_o_oval	(e300_gpio_o_oval),
/*output wire [31:0] */	.e300_gpio_o_oe	(e300_gpio_o_oe)
	<pre>/*input wire */ /*input wire */ /*output wire */ /*output wire */ /*output wire [3:0] */ /*input wire [31:0] */ /*output wire [31:0] */ /*output wire [31:0] */</pre>	<pre>/*input wire */ .aon_lfextclk /*input wire */ .itag_reset /*input wire */ .itag_reset /*input wire */ .itag_tck /*input wire */ .itag_tdi /*input wire */ .itag_tdi /*output wire */ .itag_tdo /*output wire */ .itag_tdo /*output wire */ .gspi_sck /*output wire */ .qspi_cs_0 /*inout wire [31:0] */ .e300_gpio_i_ival /*output wire [31:0] */ .e300_gpio_o_ie /*output wire [31:0] */ .e300_gpio_o_oe</pre>

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.







### ■ 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.



<b>Quartus Prime</b>	Standard	Edition
Release date: September.	2018	

Latest Release: v18.1

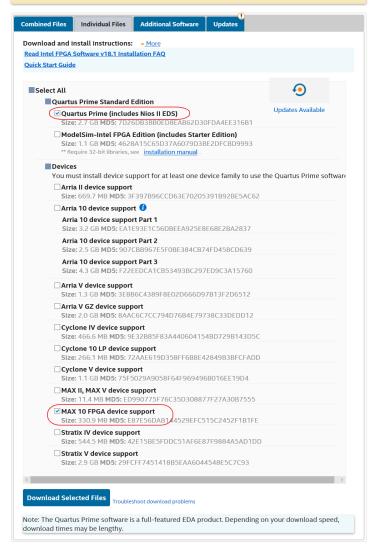
Intel' Quartus' Prime



Download Method 👔 🖲 Akamai DLM3 Download Manager 🍞 ODirect Download

✓ The Quartus Prime Design Software Standard Edition v18.1 Update 1 has been updated with Curl 7.64.0, CygWin 3.0.3, expat 2.2.6, flexnet\_publisher 11.16.2.1, glit 2.56.1, gstreamer 1.14.0, guava 27.0.1, jre (Oracle Java) 1.8.0/202, libicu 61.0, libpng 1.6.36, opensel 1.0.2r, perl 5.28.1, python 3.6.8, qt 5.11.3, ccl\_sqlite3 3.260, sqlite3 (python) 3.260, sqlite3 (pt 3.272, xerces-c++3.2.2, and Zlib 1.2.11, which include functional and security updates. Users should update to the latest version of Quartus Prime design software as soon as possible.

✓ The Quartus Prime Standard software version 18.1 supports the following device families: Stratix IV, Stratix V, Arria II,
 Arria V, Arria V GZ, Arria 10, Cyclone 10 LP, Cyclone IV, Cyclone V, MAX II, MAX V, and MAX 10 FPGA.
 ✓ More



### 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.



www.terasic.com September 24, 2020

- 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 (<u>http://T-Core.terasic.com/cd</u>). 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.

file or folder	Description
Quartus_Project/TCORE_FreedomE	a Quartus Prime project, contains a RISC-V Core,
300	Freedom E300.
Quartus_Project/	the top file of Freedom E300. User can use it to connect
TCORE_FreedomE300/V/Freedom_	to do logic operation, etc.
E300_Wrapper.v	
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	contains Terasic T-Core board supported package.
-terasic	
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.

• 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.



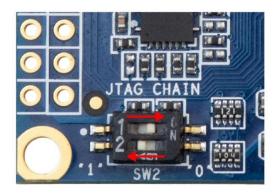


Figure 3-15 SW2 Setting

- Make sure Quartus Prime 18.1 Standard Edition is installed to the host PC.
- Connect the T-Core board (J2) to the host PC via the USB cable.
- Go to TCORE-RISCV-FreedomE300/Quartus\_Project/TCORE\_FreedomE300/demo\_batch directory using command:

 $cd\ TCORE-RISCV-FreedomE300/Quartus\_Project/TCORE\_FreedomE300/demo\_batch.$ 

• 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.

- Remove the USB Blaster Cable from T-Core to power off the board.
- 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

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



T-Core User Manual

🗢 亘 🛛 terasic@terasic: ~/Desktop/TCORE-RISCV-FreedomE300/freedom-e-sdk terasic@terasic:~/Desktop/TCORE-RISCV-FreedomE300/freedom-e-sdk\$ make upload work/build/openocd/prefix/bin/openocd -f bsp/env/freedom-e300-terasic/openocd.cfg & /home/terasic/Desktop/TCORE-RISCV-FreedomE300/freedom-e-sdk/work/build/riscv-gnu-toolc hain/riscv64-unknown-elf/prefix/bin/riscv64-unknown-elf-gdb\_software/demo\_gpio/demo\_gp io --batch -ex "set remotetimeout 240" -ex "target extended-remote localhost:3333" -ex "monitor reset halt" -ex "monitor flash protect 0 64 last off" -ex "load" -ex "monito r resume" -ex "monitor shutdown" -ex "quit" Open On-Chip Debugger 0.10.0+dev-00624-g09016bc (2019-07-16-15:47) Licensed under GNU GPL v2 For bug reports, read http://openocd.org/doc/doxygen/bugs.html Warn : Adapter driver 'usb\_blaster' did not declare which transports it allows; assumi ng legacy JTAG-only Info : only one transport option; autoselect 'jtag' adapter speed: 4000 kHz Info : Altera USB-Blaster II (uninitialized) found Info : Loading firmware.. Info : Waiting for renumerate... Info : Waiting for renumerate... Info : Altera USB-Blaster II found (Firm. rev. = 1.36) Info : This adapter doesn't support configurable speed Info : JTAG tap: riscv.cpu tap/device found: 0x20000913 (mfg: 0x489 (SiFive, Inc.), pa rt: 0x0000, ver: 0x2) Info : datacount=1 progbufsize=16 Info : Disabling abstract command reads from CSRs. Info : Examined RISC-V core; found 1 harts hart 0: XLEN=32, misa=0x40001105 Info : Info : Listening on port 3333 for gdb connections Info : Listening on port 6666 for tcl connections Info : Listening on port 4444 for telnet connections Info : accepting 'gdb' connection on tcp/3333 Info : Found flash device 'issi is25lp064' (ID 0x0017609d) 0x00000000 in ?? () Info : JTAG tap: riscv.cpu tap/device found: 0x20000913 (mfg: 0x489 (SiFive, Inc.), pa rt: 0x0000, ver: 0x2) JTAG tap: riscv.cpu tap/device found: 0x20000913 (mfg: 0x489 (SiFive, Inc.), part: 0x0 000, ver: 0x2) cleared protection for sectors 64 through 127 on flash bank 0 cleared protection for sectors 64 through 127 on flash bank 0 Info : JTAG tap: riscv.cpu tap/device found: 0x20000913 (mfg: 0x489 (SiFive, Inc.), pa rt: 0x0000, ver: 0x2) Loading section .init, size 0x74 lma 0x20400000 Loading section .text, size 0x3c6 lma 0x20400074 Loading section .rodata, size 0x28 lma 0x2040043c Loading section .data, size 0x42c lma 0x20400464 Info : Padding image section 0 at 0x2040043a with 2 bytes Info : Disabling abstract command writes to CSRs. Info : JTAG tap: riscv.cpu tap/device found: 0x20000913 (mfg: 0x489 (SiFive, Inc.), pa rt: 0x0000, ver: 0x2) Start address 0x20400000, load size 2190 Transfer rate: 1 KB/sec, 547 bytes/write. shutdown command invoked shutdown command invoked A debugging session is active. Inferior 1 [Remote target] will be detached. Quit anyway? (y or n) [answered Y; input not from terminal] Remote connection closed terasic@terasic:~/Desktop/TCORE-RISCV-FreedomE300/freedom-e-sdk\$

#### Figure 3-17 Compile and download process



www.terasic.com September 24, 2020 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.

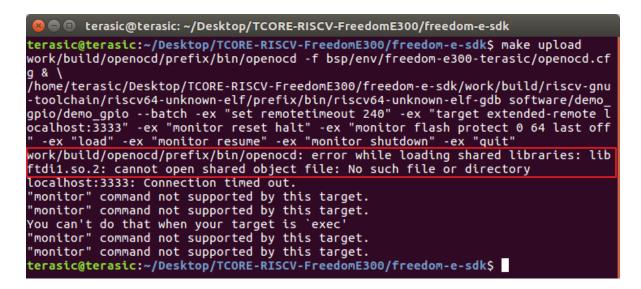


Figure 3-18 The error encountered when programming QSPI Flash

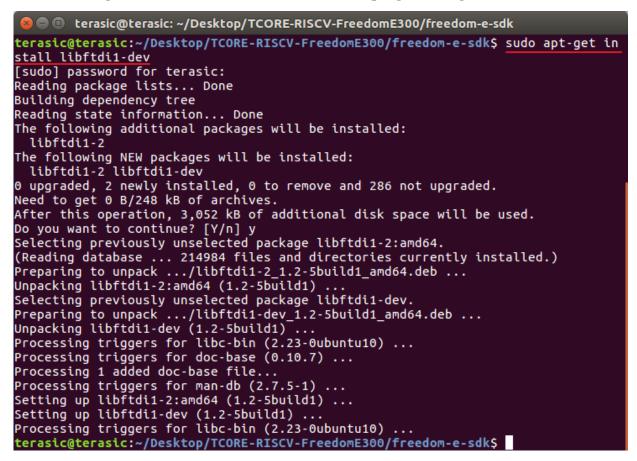


Figure 3-19 Fix the error encountered when programming QSPI Flash



- Now, you should now be able to observe the green LEDs are blinking.
- 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.

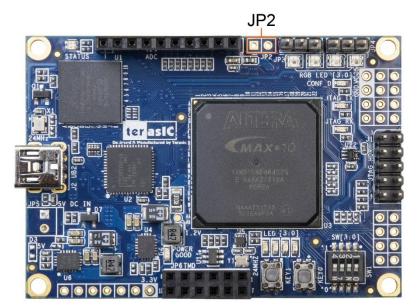


Figure 4-1 JP2 position on the T-Core board

#### Table 4-1 JP2 Jumper setting instructions

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

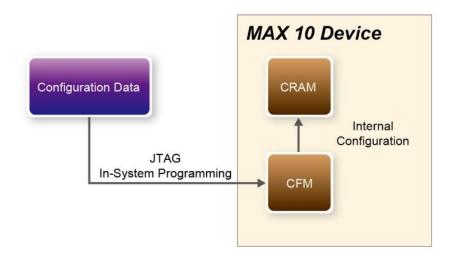


Figure 4-2 High-Level Overview of Internal Configuration for MAX 10 Devices

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**.



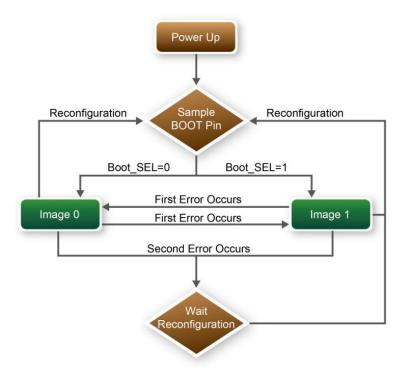


Figure 4-3 Remote System Upgrade Flow for MAX 10 Devices

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**.

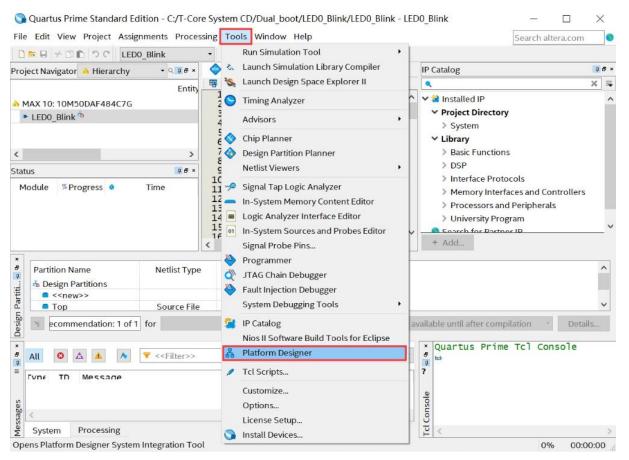


Figure 4-4 Select Qsys menu and click

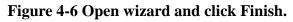
2. Please choose Library > Basic Function> Configuration and Programming > Dual Configuration Intel FPGA IP to open wizard of adding dual boot IP. See Figure 4-5 and Figure 4-6.



Platform Designer - unsaved.qsys* (C:\T-Core S	ystem CD\Dual	_boot\LED0_Blir	nk\unsaved.qsys)	-	- 🗆	×
<u>File E</u> dit <u>S</u> ystem <u>G</u> enerate <u>V</u> iew <u>T</u> ools <u>H</u> elp						
📫 IP Catalog 🛛 🗕 🗖 🗖			s Map ⊠ Interconne	ect Requirements 🛛		- d 🗆
	= A 🖷 Sy	stem: unsaved				
Library	👗 Use C N	ame	Description	Export	Clock	Base
<ul> <li>Basic Functions</li> <li>Basic Functions</li> <li>Basic Action and Programming</li> <li>Clocks: PLLs and Resets</li> <li>Clocks: PLLs and Resets</li> <li>Configuration and Programming</li> <li>Altera Config Debug Agent</li> <li>Altera Config Debug Agent Bridge</li> <li>Altera SIO Configuration Clock</li> <li>Altera SIO Mailbox Client Core</li> <li>Altera SIO Mailbox Client Core</li> <li>Altera SIO Mailbox Client Program</li> <li>Altera SIO Mailbox Programmable I</li> <li>Altera SIO Voltage Sensor</li> <li>Altera SIO Voltage Sensor</li> <li>Altera SIM Whow Bridge</li> <li>ASMI Parallel Intel FPGA IP</li> <li>Dual Configuration Intel FPGA IP</li> <li>Firor Message Resister bnooder</li> <li>Fault Injection Intel FPGA IP</li> </ul>		clk_in clk_in_reset clk	Clock Source Clock Input Reset Input Clock Output Reset Output	clk reset Double-click Double-click	exported clk_0	
New Edit + Add	<					
New Edit + Add		Current filter				>
🤼 Hierarchy 🛛 Device Family 🎘 🗕 🗗 🗖	ŏ≣ Messages ⊗		•			- 6 0
unsaved [unsaved.osvs*] ⊞► clk						
⊫ ► reset = • clk 0	Type Path	Message				
	<					>
0 Errors, 0 Warnings				Generat	e HDL	Finish

Figure 4-5 Select Dual Configuration Intel FPGA IP and click.

Dual Configuration Intel FPGA IP - dual_boot_0	×	
Dual Configuration Intel FPGA IP Megneter altera_dual_boot	Documentation	
<pre> Parameters Clock frequency: 50.0 MHz Clock frequency: 50.0 MHz Clock frequency: 50.0 MHz </pre>		
C	ancel 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**.

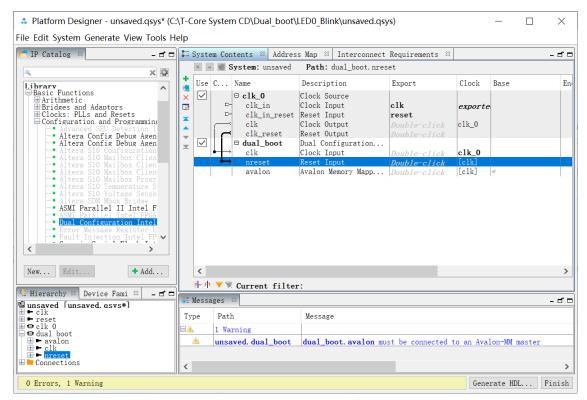


Figure 4-7 Rename and connect dual boot IP OK

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.



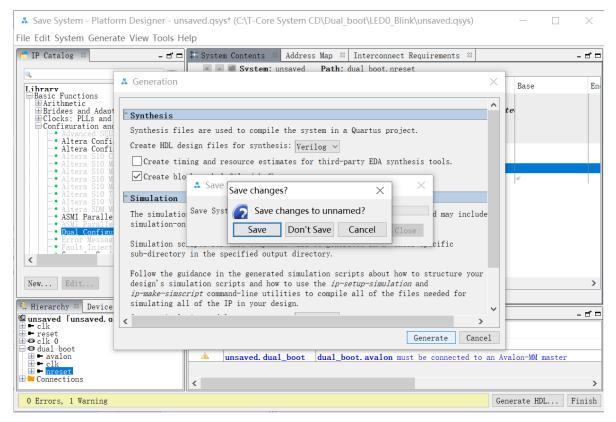


Figure 4-8 Generate and Save Qsys.

🎄 Generate Completed	$\times$
All 🛛 🛆 🕕 Progress: Validating	^
<ul> <li>Progress: Done reading input file</li> <li>Warning: dual_boot. dual_boot: dual_boot. avalon must be conn</li> <li>Info: dual_boot: Generating dual_boot "dual_boot" for QUARTU</li> <li>Info: dual_boot: generating top-level entity altera_dual_boot</li> <li>Info: dual_boot: "dual_boot" instantiated altera_dual_boot</li> <li>Info: rst_controller: "dual_boot" instantiated altera_reset</li> <li>Info: dual_boot: Done "dual_boot" with 3 modules, 7 files</li> </ul>	
<ul> <li>Info: qsys-generate succeeded.</li> <li>Info: Finished: Create HDL design files for synthesis</li> </ul>	~
<pre>&lt;</pre>	
🛆 Generate: completed with warnings.	
Stop Close	

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.

Device						
Device Board						
Select the family and d You can install addition To determine the versi	al device support w	ith the Ins	tall Devices com			r to the <u>Device Support List</u> webpage
Device family				Show in 'Ava	ilable devices' list	
Eamily: MAX 10 (DA)	/DF/DC/SA/SC)		•	Pac <u>k</u> age:	FBGA	•
Dev <u>i</u> ce: All			•	Pin <u>c</u> ount:	Any	•
Target device				Core sp <u>e</u> ed	grade: Any	*
Auto device selec	ted by the Fitter			Name filter:		
_	lected in 'Available	devices' lis	*	✓ S <u>h</u> ow ad	lvanced devices	
O Other: n/a		devices in		<u>D</u> evice and P	in Options	
A <u>v</u> ailable devices:						
Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit ^
10M50DAF256C8GES	1.2V	49760	178	178	1677312	288
10M50DAF256I7G	1.2V	49760	178	178	1677312	288
10M50DAF484C6GES	1.2V	49760	360	360	1677312	288
10M50DAF484C7G	1.2V	49760	360	360	1677312	288
	1 21/	40760	260	260	1677010	200
10M50DAF484C7G	1.2V	49760	360	360	1677312	288

Figure 4-11 Open Device window...

 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.



tegory:									
General	Configuration								
Configuration Programming Files	Specify the device conf	Specify the device configuration scheme and the configuration device.							
Unused Pins	Configuration <u>s</u> cheme: Internal Configuration								
Dual-Purpose Pins	Configuration mode:	Dual Com	pressed Images	s (512Kbits U	FM)				
Capacitive Loading Board Trace Model	Configuration device								
I/O Timing Voltage			Auto				Y		
Pin Placement	<u>U</u> se configuration	device:		Dev	ice Option	S			
Error Detection CRC CvP Settings	Configuration device I/O voltage:						Ţ		
Partial Reconfiguration	Force VCCIO to be	compatibl	e with <u>c</u> onfigur	ation I/O vol	tage				
	VID Operation mode								
	Configuration pin: Configuration Pin Options						ns		
	Generate compress	ed bitstrear	ns						
	Active serial clock source	e:							
	Enable input tri-state on active configuration pins in user mode								
	Description:								
	Specifies the configura	tion mode	used with the c	configuration	scheme fo	or configuring	g the device.		
							<u>R</u> eset		

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**.



-			System CD/Dual_boot/LEDU_Blink/LEDU_Blink - LEDU_Blink	
Eile	<u>E</u> dit <u>V</u> iew <u>P</u> roject <u>A</u> ssign		sing <u>I</u> ools <u>W</u> indow <u>H</u> elp	
	<u>N</u> ew	Ctrl+N	- 2 4 4 4 D F F K 4 8 4 8 5 9	
19	Open	Ctrl+O	led0_blink.v 🛛	
	<u>C</u> lose	Ctrl+F4	1981 66 (7) 課課 10 16 16 10 🔺 😒 🔢 📃	
•	New Project <u>W</u> izard		32 //	
¥.	Open P <u>roj</u> ect	Ctrl+J	34	
	Save Projec <u>t</u>		35 ⊟module LEDO_Blink( 36	
	Clos <u>e</u> Project		37         ////// clocks //////           38         input         ADC CLK 10.	
	Save	Ctrl+S	38inputADC_CLK_10,39inputMAX10_CLK1_50,	
6.4	Save <u>A</u> s		40 input MAX10_CLK2_50,	
ø	Save All	Ctrl+Shift+S	42 ////// KEY ///////	
		out office o	43 input [ 1: 0] KEY,	
	Eile Properties		45 /////// SW ///////	
	Create / Update	٠	46 input [3:0] SW,	
_	Export		48 ////// LED ///////	
L	Convert Programming Files		49 output [ 3: 0] LED, 50	
Ħ	Page Set <u>u</u> p		51 ////// GPIO ///////	
	Print Pre <u>v</u> iew			
0	<u>P</u> rint	Ctrl+P	Color	
	Recent Files	•		
	Recent Projects	•		
ð	Exit	Alt+F4	artitions • 1. I The selected partitions a	alread
× 8	All 🔕 🛕 🔺 🍾	▼ < <filter>&gt;</filter>	💏 Find 🍏 Find Next	

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**.



ම් Convert Programming	File - C:/T-Core Sy	stem CD/Dual_b	boot/LED0_Blink/LE	D0_Blink - LED0_Blink			_		×
ile <u>T</u> ools <u>W</u> indow							Search al	tera.com	
Specify the input files to co You can also import input f uture use.				etup information created here t	for				
Conversion setup files									
	Open Conversion	Setup Data			Save Conversion	Setup			
Output programming file									
Programming file type:	Programmer Obje	ct File (.pof)							•
Options/Boot info	Configuration devi	ce: EPCE16		<ul> <li>Mode:</li> </ul>		Internal Confi	guration		•
File name:	dual_boot.pof								
Advanced	Remote/Local upd	ate difference file	e: NO	ONE					-
	Create Memory	Map File (Genera	ate dual_boot.map)						
	Create CvP files	Generate dual	boot.periph.pof and	dual_boot.core.rbf)					
	Create config d	ata RPD (Generat	e dual_boot_auto.rp	d)					
Input files to convert									
File/Data area	Properties	tart Addres					Add	Hex Data	a
SOF Data	Page_0	<auto></auto>					Add	Sof Page	2
							Ad	dd File	
							R	emove	
								Up	
								Down	

### 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\



conterer rogium	g File - C:/T-Core Syste	em CD/Dual_boot/Li	ED0_Blink/LED0_Blink - LED0_Blink		- []
<u>T</u> ools <u>W</u> indow					Search altera.com
ure use.	incentrol monthe	arer mes and save are	e conversion setup information created ner		
onversion setup files					
	Open Conversion Setu	ıp Data	Sa	ave Conversion Setu	p
output programming file	e				
Programming file type:	Programmer Object F	File (.pof)			•
Options/Boot info	Configuration device:	EPCE16	<ul> <li><u>M</u>ode:</li> </ul>	Inte	rnal Configuration
ile <u>n</u> ame:	dual_boot.pof				
Advanced	Remote/Local update	difference file:	NONE		
	· · · · · · · · · · · · · · · · · · ·	difference men	NONE		
	Create Memory Ma				
	Create Memory Ma	ap File (Generate dual	_boot.map)		
	Create Memory Ma	ap File (Generate dual	_boot.map) riph.pof and dual_boot.core.rbf)		
put files to convert	Create Memory Ma	ap File (Generate dual enerate dual_boot.pe	_boot.map) riph.pof and dual_boot.core.rbf)		
nput files to convert	Create Memory Ma Create CvP files (Ge Create config data	ap File (Generate dual enerate dual_boot.pe RPD (Generate dual_l	_boot.map) riph.pof and dual_boot.core.rbf)		Add Hey Data
File/Data area	Create Memory Ma Create CvP files (Ge Create config data	ap File (Generate dual enerate dual_boot.pe RPD (Generate dual_l tart Addres	_boot.map) riph.pof and dual_boot.core.rbf)		Add He <u>x</u> Data
File/Data area ✓ SOF Data	Create Memory Ma Create CvP files (Ge Create config data	ap File (Generate dual enerate dual_boot.pe RPD (Generate dual_l	_boot.map) riph.pof and dual_boot.core.rbf)		Add He <u>x</u> Data Add <u>S</u> of Page
File/Data area	Create Memory Ma Create CvP files (Ge Create config data Properties Page_0 10M50DAF484	ap File (Generate dual enerate dual_boot.pe RPD (Generate dual_l tart Addres	_boot.map) riph.pof and dual_boot.core.rbf)		
File/Data area ✓ SOF Data LED0_Blink.sof	Create Memory Ma Create CvP files (Ge Create config data Properties Page_0 10M50DAF484	ap File (Generate dual enerate dual_boot.pe RPD (Generate dual_l tart Addres <auto></auto>	_boot.map) riph.pof and dual_boot.core.rbf)		Add <u>S</u> of Page
File/Data area ✓ SOF Data LED0_Blink.sof ✓ SOF Data	Create Memory Ma Create CvP files (Ge Create config data Properties Page_0 10M50DAF484 Page_1	ap File (Generate dual enerate dual_boot.pe RPD (Generate dual_l tart Addres <auto></auto>	_boot.map) riph.pof and dual_boot.core.rbf)		Add <u>S</u> of Page Add <u>F</u> ile
File/Data area ✓ SOF Data LED0_Blink.sof ✓ SOF Data	Create Memory Ma Create CvP files (Ge Create config data Properties Page_0 10M50DAF484 Page_1	ap File (Generate dual enerate dual_boot.pe RPD (Generate dual_l tart Addres <auto></auto>	_boot.map) riph.pof and dual_boot.core.rbf)		Add <u>S</u> of Page Add <u>F</u> ile Remove

### 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.



Hardware Setup						>
Hardware Settings	JTAG Settings					
Select a programmin hardware setup appli				devices. T	his programming	
Currently selected ha		e [USB-1]			-	
Available hardware	items					
Hardware T-Core		Server Local	Port USB-1		Add Hardware Remove Hardware	
					Close	

### Figure 4-16 Hardware setup window

Programmer - G:/tmp/T-core/cd/Demonstration/Dual_boot/LED0_Blink/LED0_Blink - LED0_Blink - [LED0_Blink —										
<u>F</u> ile <u>E</u> dit <u>V</u> iew	Eile Edit View Processing Tools Window Help									
Hardware Setup T-Core [USB-1] Mode: JTAG Progress:										
Enable real-time ISP to allow background programming when available										
► <sup>™</sup> Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine		
Stop	dual_boot.pof	10M50DAF484	0EDC6665	00277542						
💏 Auto Detect	CFM0									
	CFM1									
🗙 Delete	UFM									
🏓 Add File										
Change File	<							>		
Save File								^		
Add Device TDI Down 10M50DAF484 TDO TDI 10M50DAF484 TDO										

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



T-Core User Manual www.terasic.com September 24, 2020 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.

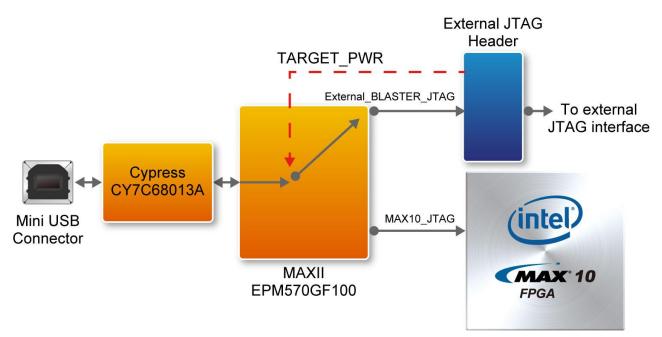
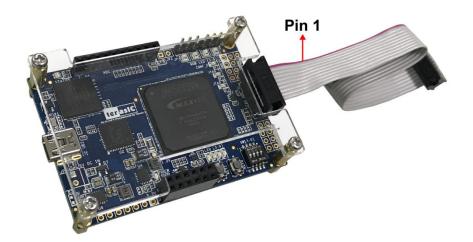


Figure 5-3 The auto-switch JTAG circuit of the T-Core board

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**.



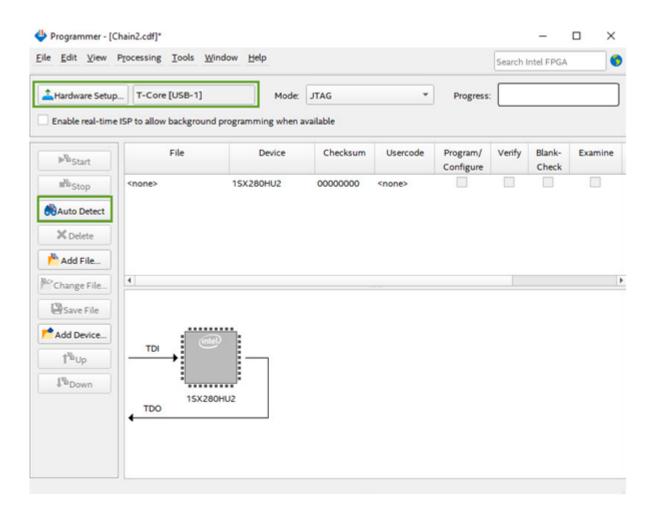


Figure 5-6 Using T-Core board to program the DE10-Pro

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**.



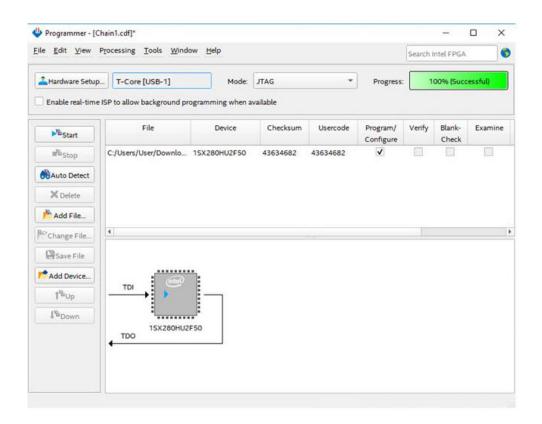


Figure 5-7 Using T-Core board to program the DE10-Pro



# Additional Information

### **Revision History**

Date	Version	Change Log	
2019.07	V1.0	Initial Version (Preliminary)	
2020.01	V1.1	<ul> <li>Modify the SW2 setting description for Figure 3-16.</li> <li>Change blaster name from DE-Core to T-Core</li> </ul>	
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**

Copyright <sup>©</sup> Terasic Inc. All rights reserved.



### **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-ISGX-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