



FRONTGRADE

PROGRAMMING MRAM

UT24CP1008

CertusPro-NX-RT FPGA

4/7/2026

Version #:1.0.0

Introduction

Frontgrade has created example user code for programming external non-volatile storage devices using Lattice Serial Peripheral Interface (SPI) Flash Memory Controller IP. The process is comprised of a set of user modules for interfacing to Lattice SPI Flash IP. The user code interfaces to the IP to download Lattice FPGA image and store it in the external non-volatile memory device.

The IP provides an interface between the user code and an external non-volatile memory device. AHB-Lite interface is used to configure the IP through the IP Graphical User Interface (GUI), and the APB interface is used to issue any flash operations. This IP receives and processes data or control signals from user code, and then translates each flash command into serial form to be communicated to the external non-volatile memory device.

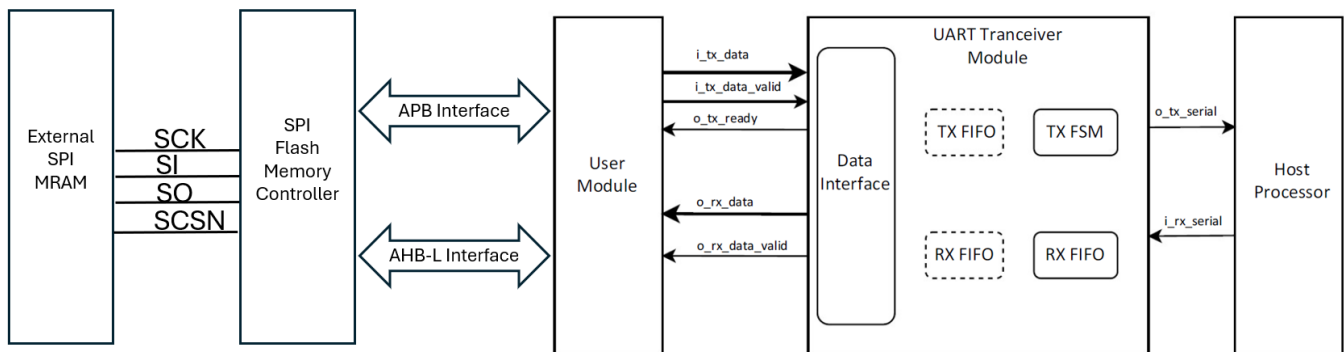
For details on Lattice’s Flash Memory Controller IP, refer to [FPGA-IPUG-02134-2-1-SPI-Flash-Memory-Controller-IP-User Guide.pdf](#).

Note: The hex file was created from the accompanying **ADC app note** and following **Global** settings:

- MASTER_SPI_PORT: SERIAL
- JTAG_PORT: ENABLE
- PROGRAMN_PORT: DISABLE
- MCCLK_FREQ: 3.5

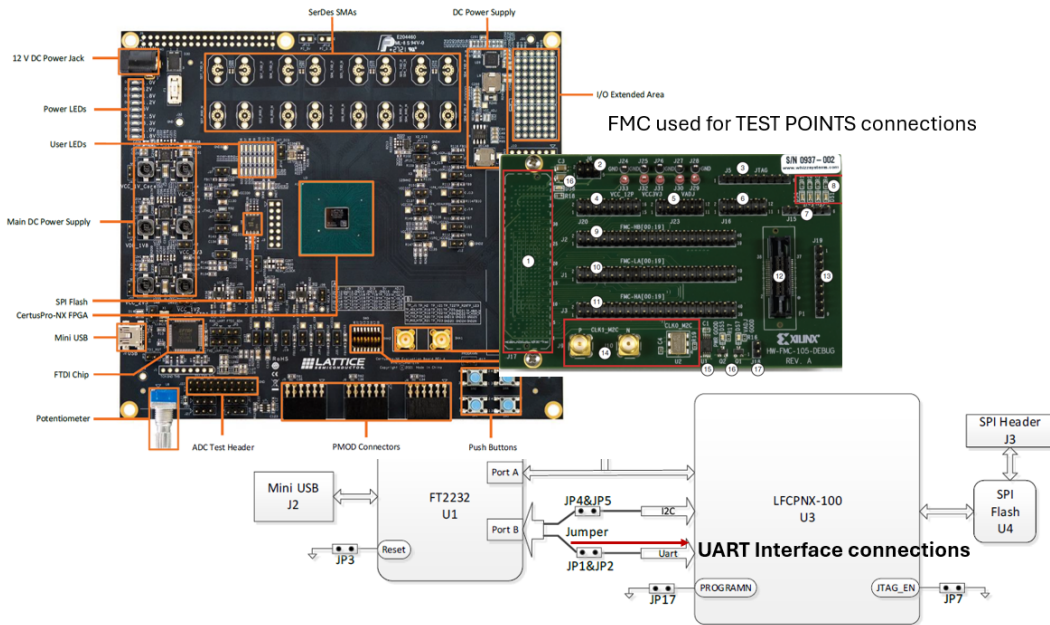
This app note goes through the following steps:

1. Use Lattice Tools to create the hex file.
2. Send intel hex file using <Intel Hex file name>.ttl and Terminal.
Terminal > Control > Macro > <Intel Hex file name>.ttl.
3. User code receives the hex file line by line and assembles one memory page.
4. The SPI Flash Memory Controller sends one memory page at a time to MRAM until the image is complete.



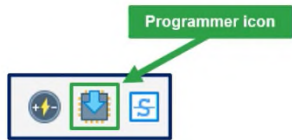
CertusPro-NX Device

CertusPro-NX Evaluation Board is the LFCPNX-100-9LFG672C. The Eval Board can be used in a wide range of applications. For more information on the capabilities of CertusPro-NX device, see CertusPro-NX Family Data Sheet (FPGA-DS-02086).

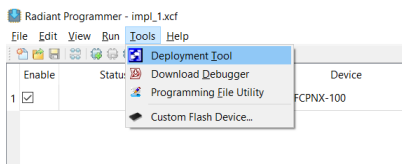


1.1 Preparing MRAM Image from CertusPro-NX-RT Bit File

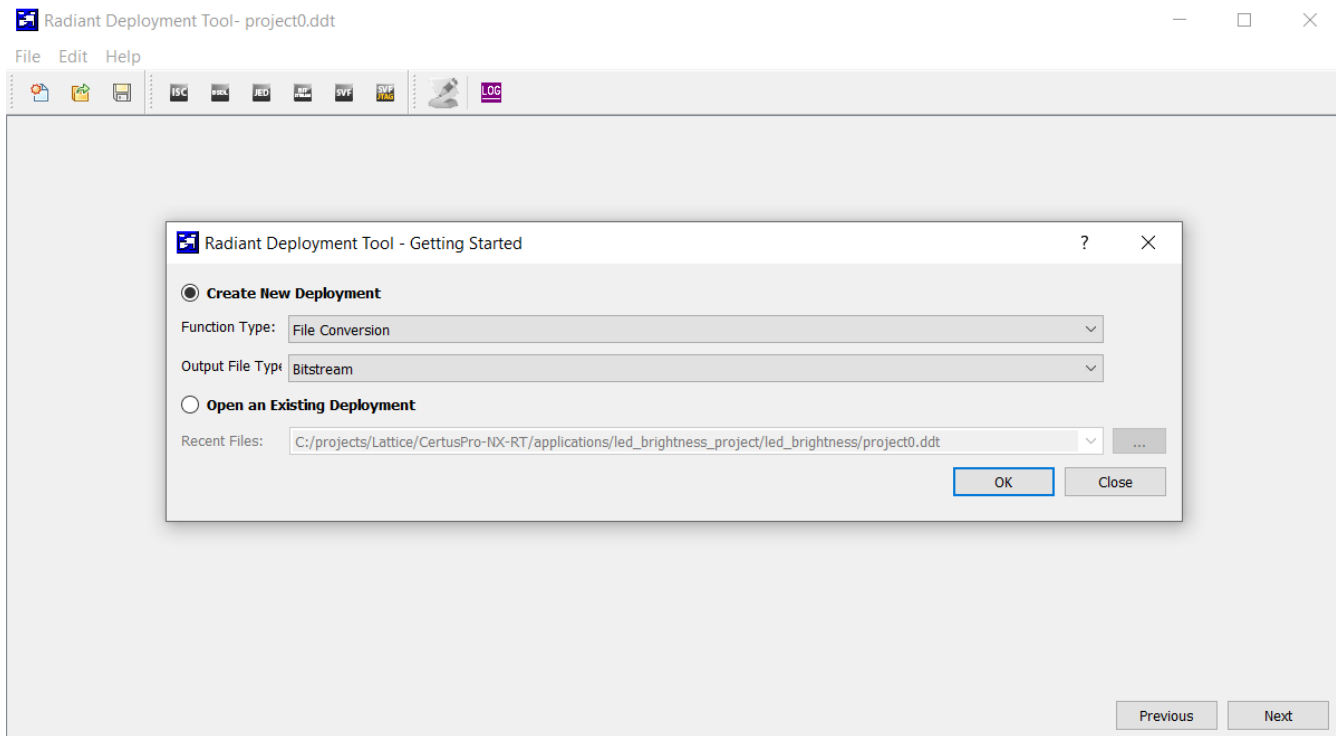
The Programmer tool from Lattice can be used to create the hex file. Launch the programming tool, see picture below.



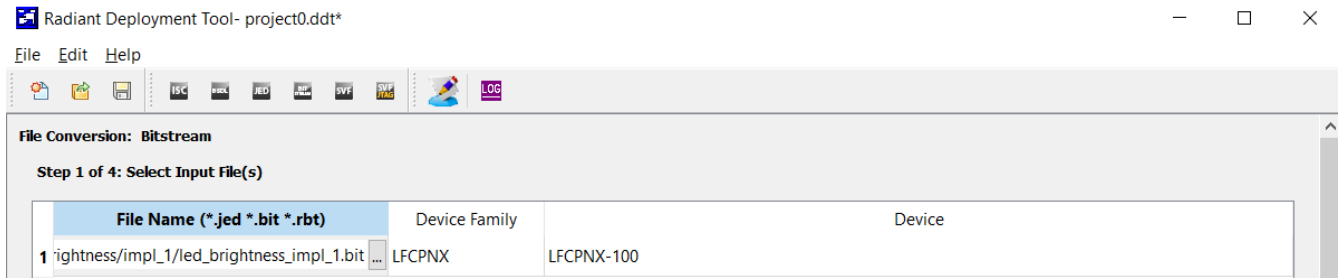
Under Tools, click on Deployment Tool, see picture below.



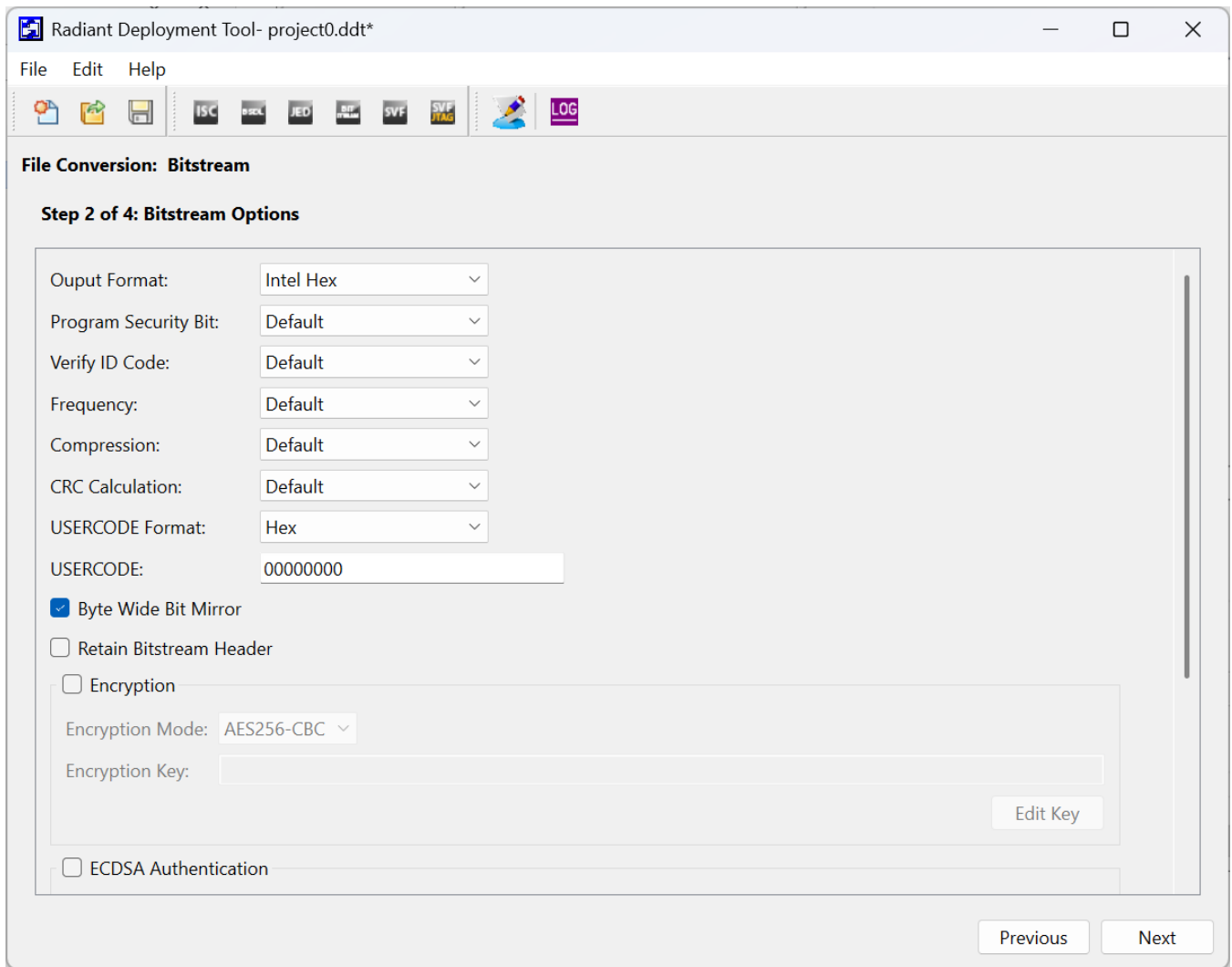
In the Radiant Deployment Tool, under Create New Deployment, choose the Function Type and Output File Type and click OK, see picture below.



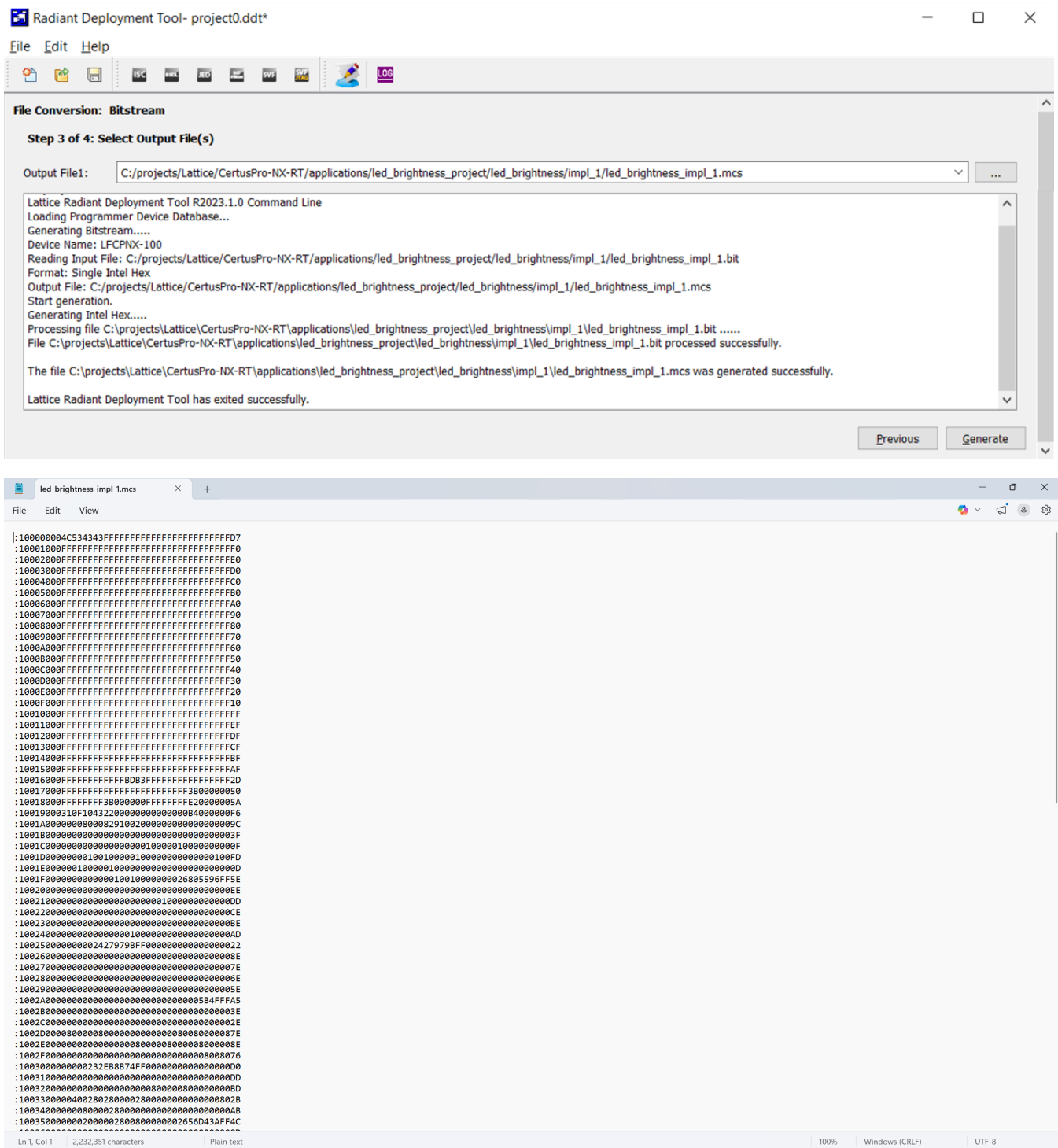
Under File Conversion, choose the bit file and click Next, see picture below.



Under File Conversion, choose the Output Format and click Next, see picture below.



Verify the output file, click Next, and finally generate the hex file, see picture below.



Below is Intel Hex Format for references.

```

:10010000214601360121470136007EFE09D2190140
:100110002146017E17C20001FF5F16002148011928
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:00000001FF
    
```

 Start code
 Byte count
 Address
 Record type
 Data
 Checksum

Hex code	Record type	Description	Example
00	Data	Contains data and a 16-bit starting address for the data. The byte count specifies number of data bytes in the record. The example shown to the right has 0B (eleven) data bytes (61, 64, 64, 72, 65, 73, 73, 20, 67, 61, 70) located at consecutive addresses beginning at address 0010.	:0B0010006164647265737320676170A7
01	End Of File	Must occur exactly once per file in the last line of the file. The data field is empty (thus byte count is 00) and the address field is typically 0000.	:00000001FF
02	Extended Segment Address	The data field contains a 16-bit segment base address (thus byte count is always 02) compatible with 80x86 real mode addressing. The address field (typically 0000) is ignored. The segment address from the most recent 02 record is multiplied by 16 and added to each subsequent data record address to form the physical starting address for the data. This allows addressing up to one megabyte of address space.	:020000021200EA
03	Start Segment Address	For 80x86 processors, specifies the initial content of the CS:IP registers (i.e., the starting execution address). The address field is 0000, the byte count is always 04, the first two data bytes are the CS value, the latter two are the IP value.	:0400000300003800C1
04	Extended Linear Address	Allows for 32 bit addressing (up to 4GiB). The record's address field is ignored (typically 0000) and its byte count is always 02. The two data bytes (big endian) specify the upper 16 bits of the 32 bit absolute address for all subsequent type 00 records; these upper address bits apply until the next 04 record. The absolute address for a type 00 record is formed by combining the upper 16 address bits of the most recent 04 record with the low 16 address bits of the 00 record. If a type 00 record is not preceded by any type 04 records then its upper 16 address bits default to 0000.	:02000004FFFFFFE
05	Start Linear Address	The address field is 0000 (not used) and the byte count is always 04. The four data bytes represent a 32-bit address value (big-endian). In the case of 80386 and higher CPUs, this address is loaded into the EIP register.	:04000005000000CD2A

1.2 Upload Intel Hex File to External Non-volatile Memory using Terminal

Before sending the hex file, Tera Term needs to be set to 115200 Baud rate and the following settings.

The screenshot shows the 'Tera Term: Serial port setup and connection' dialog box. The settings are as follows:

Port:	COM24	New setting
Speed:	115200	
Data:	8 bit	Cancel
Parity:	none	
Stop bits:	1 bit	Help
Flow control:	none	

Transmit delay:
0 msec/char 0 msec/line

Send intel hex file using ut32_can_upload.ttl and Terminal
Terminal > Control > Macro > ut32_can_upload.ttl

```
13 ; Open a file.
14 fopen fhandle 'led_brightness_impl_1.mcs' 0
15
16 while 1
17 ;for i 1 32
18 ; Read a line from the file.
19 filereadln fhandle line
20 if result=1 then
21     break
22 endif
23 ; Display the line.
24 sendln line
25 wait 'z'
26 ;wait 't444'
27 ;wait #13
28 ;break
29 ; Repeat until the end of the file.
30 ;next
31 endwhile
32
33
34 ; Close the file.
35 fileclose fhandle
```

1.3 User code receives the hex file line by line and assembles one memory page.

Data received from the UART is parsed and saved to a buffer. Each line in the Intel Hex Format contains thirty-two characters, which two characters combine to make one byte. The code saves 512 characters to make a page of 256 bytes.

The 256 bytes are saved to a buffer The SPI flash controller takes the 256 bytes and writes them to the external SPI MRAM.

```

else begin
    rcvd_line_cnt    <= 0;
    send_resp_val    <= 2;
    send_resp_type   <= RESP_DATA;
    state            <= SEND_RESP;
end
end
end else if ( (rcvd_line_cnt == 10'd8) & (rx_data[6:0] == 8'h31) ) begin
    eof_done <= 1'b1;
    rcvd_line_cnt <= 10'd41;
end else if ( (rcvd_line_cnt == 10'd8) & ((rx_data[6:0] == 8'h32)|| (rx_data[6:0] == 8'h33)|| (rx_data[6:0] == 8'h34)|| (rx_data[6:0] == 8'h35) ))
    rcvd_line_cnt <= 10'd41;
end else begin
    rcvd_line_cnt    <= rcvd_line_cnt + 1;
    if(rcvd_line_cnt >= 9 && rcvd_line_cnt < 41)begin
        //rx_line_valid <= 1;
        if (char_2_hex == 0)begin
            data_8b_hi <= rx_data[6:0];
        end
        else begin
            image_page[page_ptr*8 +: 8] <= (ascii_to_hex(data_8b_hi),ascii_to_hex(rx_data[6:0]));
            page_ptr    <= page_ptr + 1;
        end
        char_2_hex <= ~char_2_hex;
    end
end
end
end

end

SEND_RESP: begin
    if( new_char_check )begin
        if (send_resp_type == RESP_OK)
            begin
                case (char_cnt) // synthesis full_case
                    0 : tx_data <= " ";
                    2 : tx_data <= "0";
                //apb_wdata_o    <= 32'h100; // PAGE_PROGRAM_SIZE          = 256;
                init_sm_cs    <= ST_INIT_WRITE;
                apb_psel_o    <= 1'b1;
                apb_penable_o<=0;
                apb_wr_rdn_o  <= 1'b1;

                if(flash_wr_rdn) begin
                    apb_offset_o <= PAGE_RD_LEN_0;
                    page_prq_cmds <= 3'b110;
                end
                else begin
                    apb_offset_o <= PAGE_PRG_LEN_0; //Set the page program length
                    page_prq_cmds <= 3'b011;
                end

                apb_wdata_o    <= 32'h100; // PAGE_PROGRAM_SIZE          = 256;
                data_8b_2d_32b <= 0;
                page_ptr    <= 0;

            end
            // PAGE_PRG_BUF; //Generate word data
            3'b011 : begin

                init_sm_cs    <= ST_INIT_WRITE;
                apb_psel_o    <= 1'b1;
                apb_penable_o<=0;
                apb_wr_rdn_o  <= 1'b1;
                apb_offset_o  <= page_ptr*4 + PAGE_PRG_BUF; //Generate word data
                apb_wdata_o    <= image_page[page_ptr*32 +: 32];
                page_ptr    <= page_ptr + 1;

                if(page_ptr>=64)begin
                    page_prq_cmds <= 3'b100;
                    //init_ptr    <= init_ptr + PAGE_PROGRAM_SIZE;
                end
            end
        end
    end
end

```

1.4 SPI Flash Memory Controller Sends One Memory Page to MRAM.

The APB bus is used to directly read from or write to any memory location within the SPI flash memory controller. It is used to define and set how SPI MRAM transactions are executed by the controller. For writing one page to Flash, the following APB interface register accesses were used to perform this transaction:

1. Register write to '0x040' with '32-bit flash address'.
2. Register write to '0x044' to set the Flash Addressing mode to 32-bits.
3. Register write to '0x104' to set the page program length.
4. Register write to '0x200' to set the Page Program Buffer register.
5. Finally, register write to '0x000', Page Program register, to write the page to MRAM.

```

`case (page_prq_cmds[2:0])
// SPI_ADDR_BYTE0; //Set the spi flash address
3'b000: begin
    init_sm_cs    <= ST_INIT_WRITE;
    apb_request_o <= 1'b1;
    apb_wr_rdn_o  <= 1'b1;
    page_prq_cmds <= 3'b001;
    apb_offset_o  <= SPI_ADDR_BYTE0; //Set the spi flash address
    apb_wdata_o   <= init_ptr;//32'h00;
end
// Core Volts on ADC0
3'b001: begin
    init_sm_cs    <= ST_INIT_WRITE;
    apb_request_o <= 1'b1;
    apb_wr_rdn_o  <= 1'b1;
    page_prq_cmds <= 3'b010;
    apb_offset_o  <= SPI_ADDR_MODE; //Set the flash addressing mode to 24-bit or 32-bit mode
    apb_wdata_o   <= 32'h01;
end
// SPI_ADDR_MODE; //Set the flash addressing mode to 24-bit or 32-bit mode
3'b010: begin
    //page_prq_cmds    <= 3'b011;
    //apb_offset_o    <= PAGE_PRG_LEN_0; //Set the page program length
    //apb_wdata_o     <= 32'h100; // PAGE_PROGRAM_SIZE = 256;
    init_sm_cs    <= ST_INIT_WRITE;
    apb_request_o <= 1'b1;
    apb_wr_rdn_o  <= 1'b1;
    page_prq_cmds <= 3'b011;
    apb_offset_o  <= PAGE_PRG_LEN_0; //Set the page program length
    apb_wdata_o   <= 32'h100; // PAGE_PROGRAM_SIZE = 256;
    data_0b_2d_32b <= 0;
    page_ptr      <= 0;
    //tx_ready     <= 1;
end


// PAGE_PRG_BUF; //Generate word data
3'b011 : begin

    init_sm_cs    <= ST_INIT_WRITE;
    apb_psel_o    <= 1'b1;
    apb_penable_o <= 0;
    apb_wr_rdn_o  <= 1'b1;
    apb_offset_o  <= page_ptr*4 + PAGE_PRG_BUF; //Generate word data
    apb_wdata_o   <= image_page[page_ptr*32 +: 32];
    page_ptr      <= page_ptr + 1;

    if (page_ptr >= 64) begin
        page_prq_cmds <= 3'b100;
        //init_ptr    <= init_ptr + PAGE_PROGRAM_SIZE;
    end
end
3'b100 : begin
    init_sm_cs    <= ST_INIT_WRITE;
    apb_request_o <= 1'b1;
    apb_wr_rdn_o  <= 1'b1;
    page_prq_cmds <= 3'b000;
    apb_offset_o  <= PAGE_PRG; //Run the page program in SPI interface
    apb_wdata_o   <= 32'h01; // PAGE_PROGRAM_SIZE = 256;
    init_last     <= 1;
end

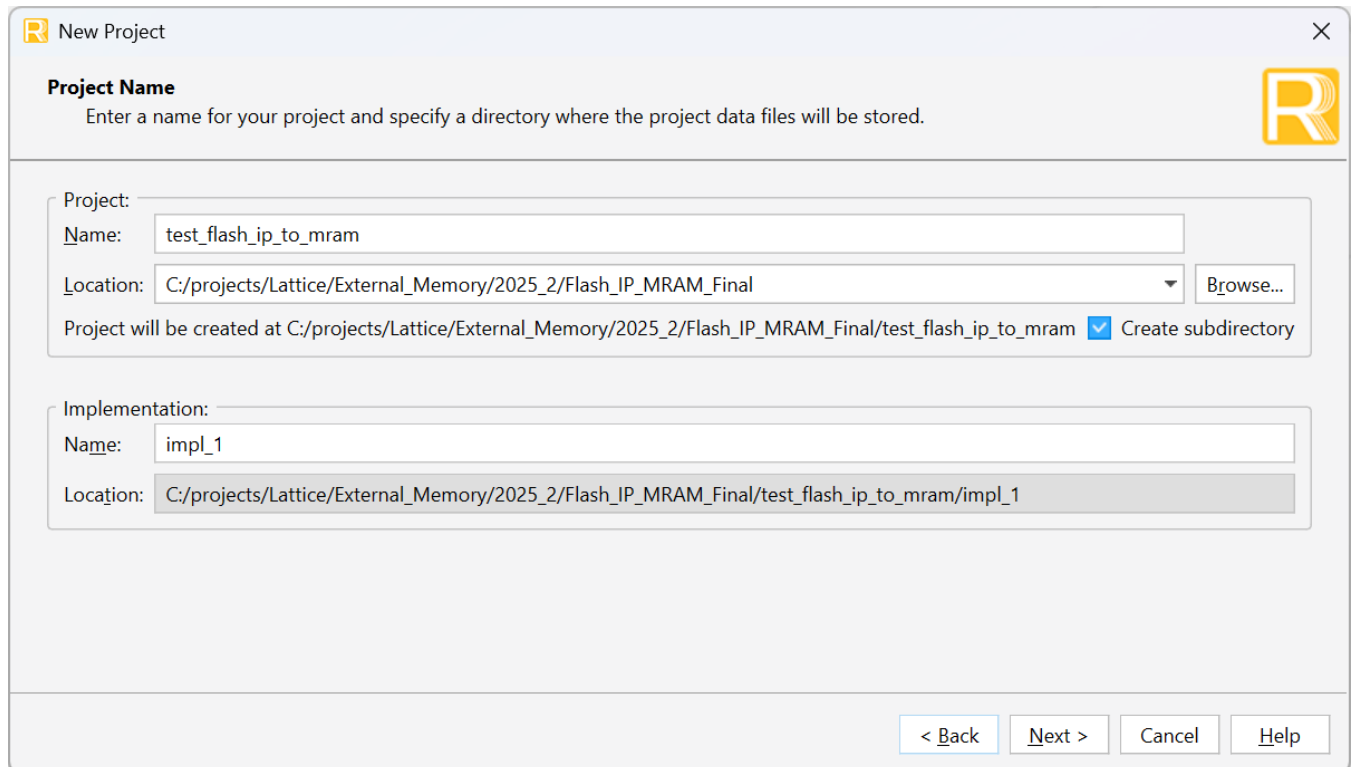
```

Creating User Code Project with Radiant Software

Launch Radiant Software 

From the File menu, select New > Project....

Specify the project name as test_flash_ip_to_mram, navigate to the location of your choice and click Next, see figure below.



New Project [Close]

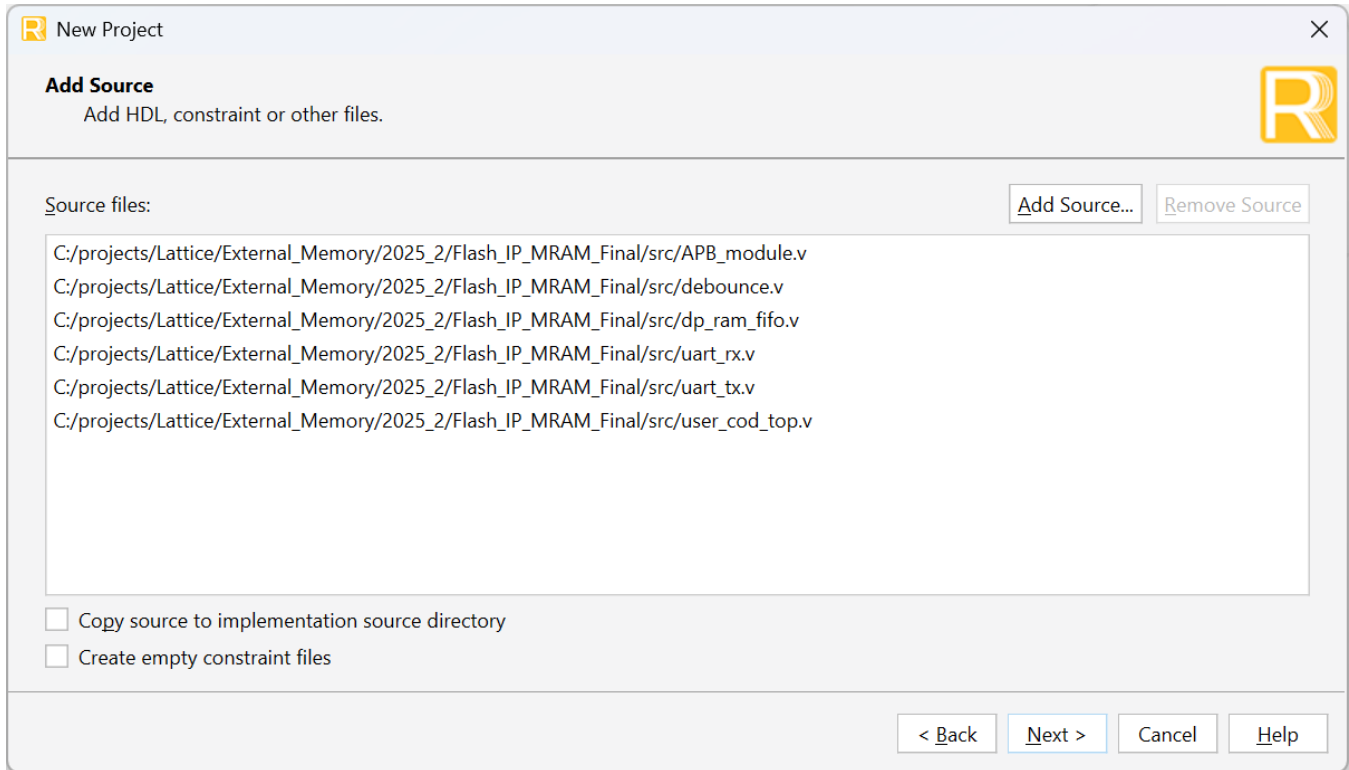
Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project:
Name: test_flash_ip_to_mram
Location: C:/projects/Lattice/External_Memory/2025_2/Flash_IP_MRAM_Final [Browse...]
Project will be created at C:/projects/Lattice/External_Memory/2025_2/Flash_IP_MRAM_Final/test_flash_ip_to_mram Create subdirectory

Implementation:
Name: impl_1
Location: C:/projects/Lattice/External_Memory/2025_2/Flash_IP_MRAM_Final/test_flash_ip_to_mram/impl_1

< Back Next > Cancel Help

Select Add Source... and click Open, see picture below, click Next.



Select the appropriate device, click Next and choose Simplify Pro; click Next and Finish, see picture below.

New Project

Select Device
Specify a target device for the project.

Select Device:

Family:	Device:
iCE40UP (iCE40 UltraPlus)	LFCPNX-50
LAV-AT (Avant)	LFCPNX-100
LFCPNX (CertusPro-NX)	
LFD2NX (Certus-NX)	
LFMXO4 (MachXO4)	
LFMXO5 (MachXO5-NX)	
LIFCL (CrossLink-NX)	
LN2-CT (Certus-N2)	

Operating Condition: Commercial
Package: LFG672

Performance Grade: 9_High-Performance_1.0V

Part Number: LFCPNX-100-9LFG672C

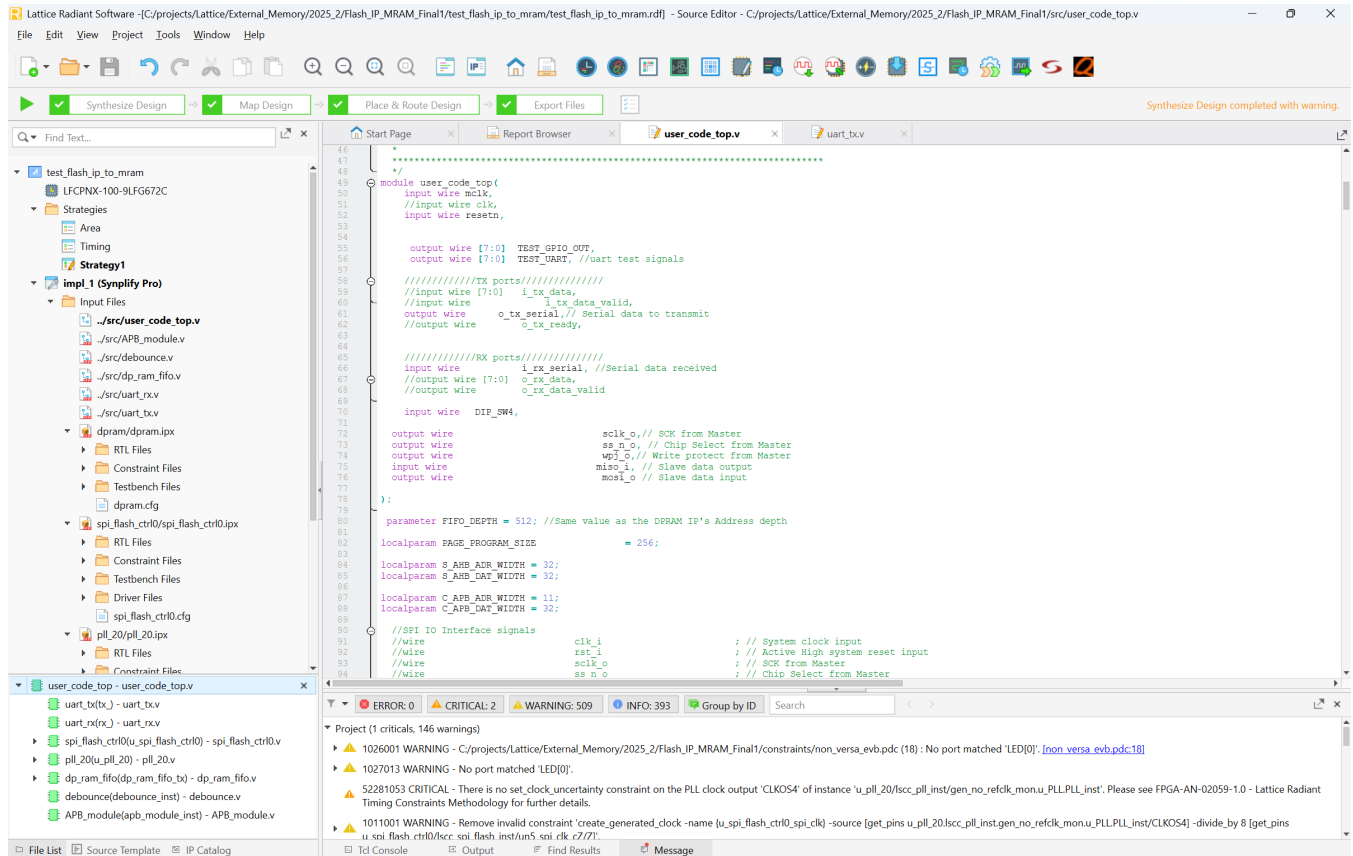
[Online Data Sheet for Device](#)

Device Information:

Core Voltage:	1.00 V
Logic Cells:	96000
LUTs:	79872
Registers:	79872
EBR Blocks:	208
LRAM:	7
DSP (18x18 Multiplier):	156
ADC Blocks:	1
PLLs:	4
DLLs:	2
PCSs:	2
ALUs:	1
DPHYS:	0
PIO Cells:	299
PIO Pins:	299

< Back Next > Cancel Help

The initial project with provided files should look like the figure below.



2.1 SPI Flash Controller IP

From the IP Catalog Architecture Modules, add a SPI Flash Controller IP module with the options shown below and name it spi_flash_ctrl0, see picture below.

Configure Component from IP spi_flash_controller Version 2.0.0
Set the following parameters to configure this component.

Diagram spi_flash_ctrl0

Configure spi_flash_ctrl0:

Property	Value
Control Port Settings	
Enable Page Program Buffer	<input checked="" type="checkbox"/>
Enable Page Read Buffer	<input checked="" type="checkbox"/>
Page Program Buffer Memory Type	EBR
Page Read Buffer Memory Type	EBR
Page Buffer Interface	APB
Control Base Address [0x80000000 - 0xFFFFF800]	32'h80000000
SPI Flash Settings	
Page Size	256
Flash Memory Map Size (KB)	1024
SCLK Rate [0 - 15]	4
SCLK Polarity	0
SCLK Phase	0
First Transmitted Bit	MSB
Data and Control Bus	
Data Port AHB Address Width	32
Data Port AHB Data Width	32
Data Port AHB Data Byte Endianness	little-endian
Control Port APB Address Width	11
Control Port APB Data Width	32
SPI Flash Command Opcodes	
Slow Read	8'h03
Fast Read	8'h0b
Page Program	8'h02
Block Erase Type 1	8'h20
Block Erase Type 2	8'h52
Block Erase Type 3	8'hd8
Chip Erase	8'h60
Write Enable	8'h06
Write Disable	8'h04
Read Status Register	8'h05

Diagram details: spi_flash_ctrl0 block with inputs AHBL_S0, APB_S0, ahbl_hclk_i, apb_pclk_i, miso_i, rst_i and outputs mosi_o, sclk_o, ss_n_o, wpj_o.

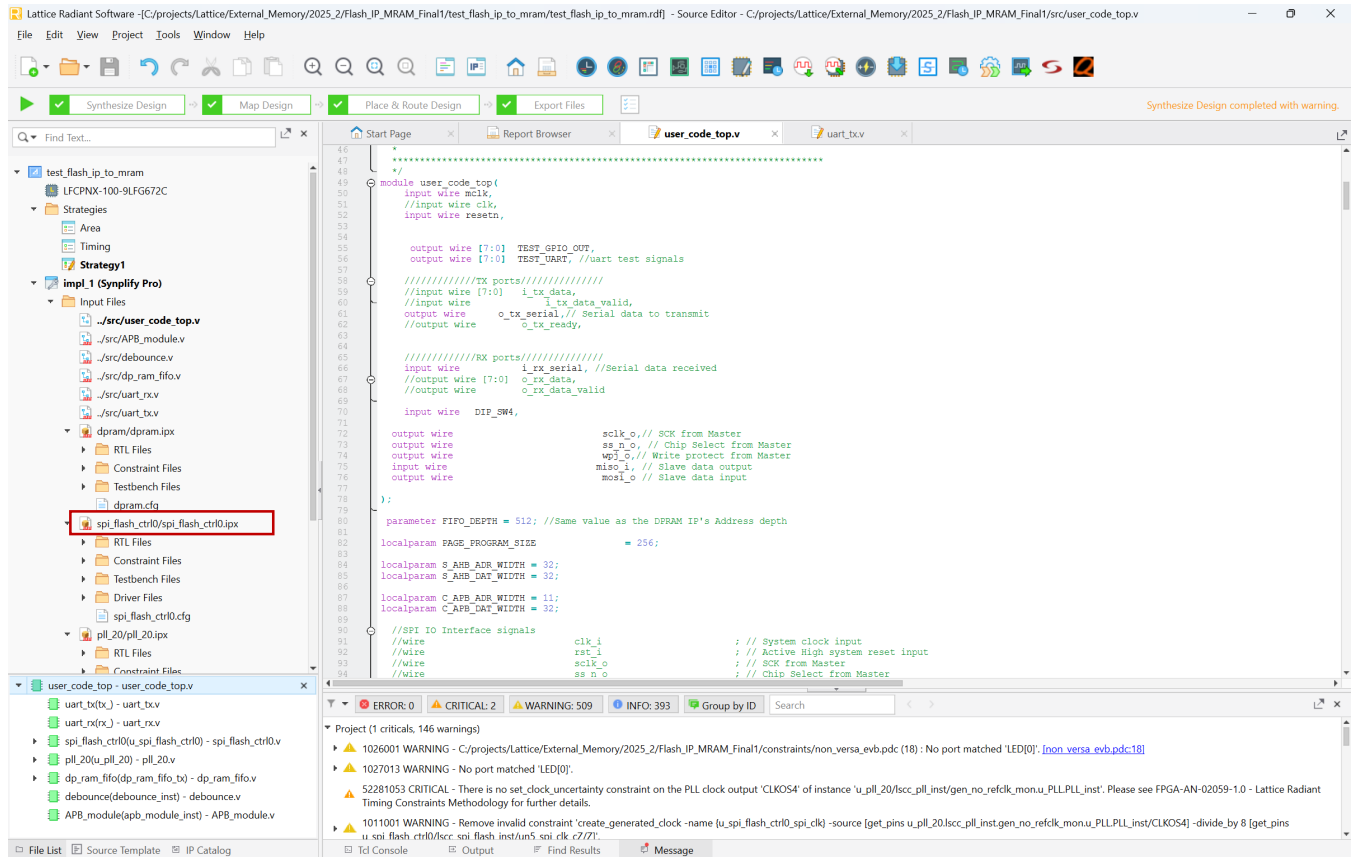
[User Guide](#)

No DRC issues are found.

< Back Generate Cancel

Once the spi_flash_ctrl0 options are entered, click the Generate button to start the creation of the module. On the next screen, click the Finish button to complete the creation of the module.

Once the spi_flash_ctrl0 is created, it is added to the project files, see picture below.



2.2 PLL IP

From the IP Catalog Architecture Modules, add a PLL module with the options shown below and name it pll_20, see picture below.

Module/IP Block Wizard
Configure Component from Module pll Version 1.9.1
Set the following parameters to configure this component.

Diagram pll_20

Configure IP

General		Optional Ports
Property	Value	
CLKI: Frequency (MHz) [18 - 800]	125	
CLKI: Divider Actual Value [1 - 44]	5	
Phase Detector Frequency (MHz) [18 - 500]	25	
Enable Reference Clock Monitor	<input type="checkbox"/>	
Feedback		
CLKFB: Feedback Mode	INTCLKOP	
CLKFB: FBK Divider Actual Value (Integer) [1 - 128]	4	
Primary Clock Output		
CLKOP: Bypass	<input type="checkbox"/>	
CLKOP: Frequency Desired Value (MHz) [10 - 800]	100	
CLKOP: Divider Actual Value [1 - 128]	8	
CLKOP Tolerance (%)	0.0	
CLKOP: ERROR (PPM)	0	
CLKOP: Enable Trim for CLKOP	<input type="checkbox"/>	
Secondary Clock Output		
CLKOS: Enable	<input type="checkbox"/>	
Secondary Clock Output (2)		
CLKOS2: Enable	<input type="checkbox"/>	
Secondary Clock Output (3)		
CLKOS3: Enable	<input type="checkbox"/>	
Secondary Clock Output (4)		
CLKOS4: Enable	<input checked="" type="checkbox"/>	
CLKOS4: Bypass	<input type="checkbox"/>	
CLKOS4: Frequency Desired Value (MHz) [6.25 - 800]	20	
CLKOS4: Divider Actual Value [1 - 128]	40	
CLKOS4 Tolerance (%)	0.0	

Calculate

No DRC issues are found.

Generate Cancel

Once the PLL options are entered, click the Generate button to start the creation of the module.

ON the next screen, click the Finish button to complete the creation of the module.

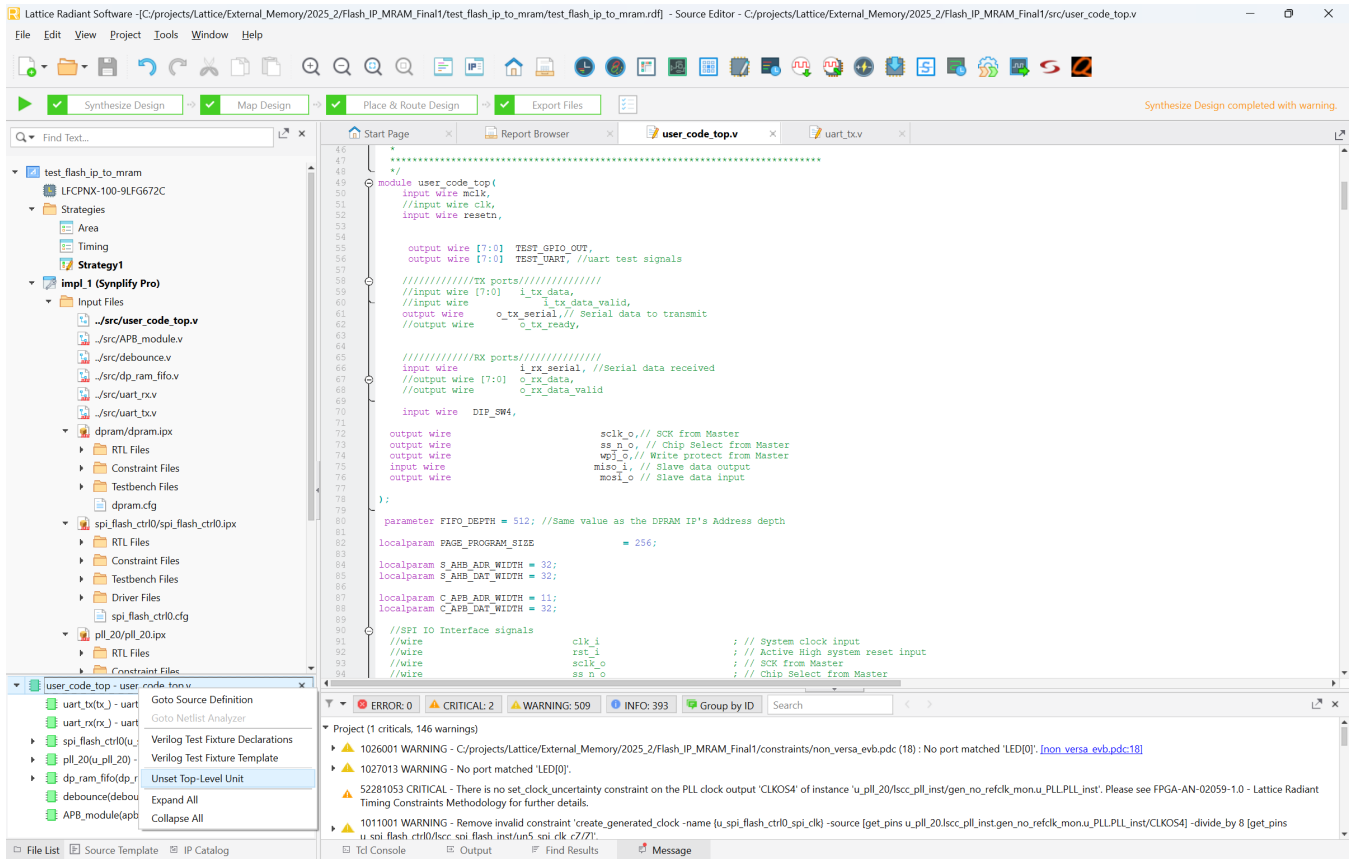
Once the PLL is created, it is added to the project files, see picture below.

The screenshot displays the Lattice Radiant Software interface. The top toolbar shows the synthesis process: Synthesize Design, Map Design, Place & Route Design, and Export Files. The main window is divided into three panes:

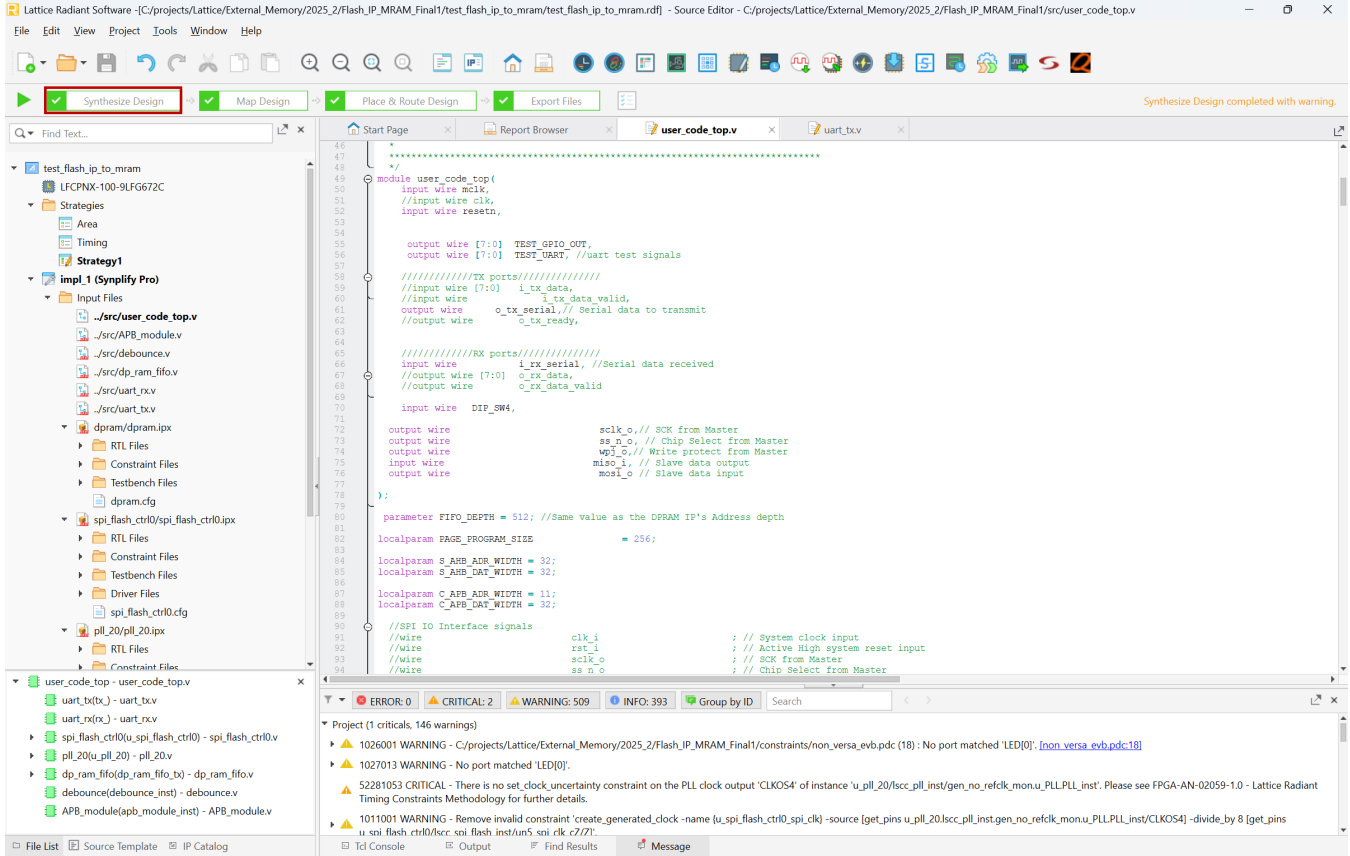
- Left Pane (Project Explorer):** Shows the project hierarchy. Under 'impl_1 (Synplify Pro)' > 'Input Files', the file 'p1l_20/p1l_20.iptx' is highlighted with a red box.
- Center Pane (Code Editor):** Displays the Verilog code for 'user_code_top.v'. The code defines a module with various input and output signals, including clock, reset, and data buses. It also includes local parameters for FIFO depth and program size.
- Bottom Pane (Messages):** Shows the synthesis results. It reports 1 critical error, 2 critical warnings, 509 warnings, and 393 info messages. The messages include warnings about port mismatches and critical errors related to clock uncertainty constraints.

2.3 Compile Design Using Radiant Software

From the Project File List, right click on user_code_top.v and set is as Set Top-Level Unit, see picture below.



Synthesize the design by clicking on the Synthesize Design arrow, see figure below.



2.4 Pin Assignment (Device Constraint Editor)

Add the provided non_versa_evb.pdc constraint files to its respective folder, see picture below.

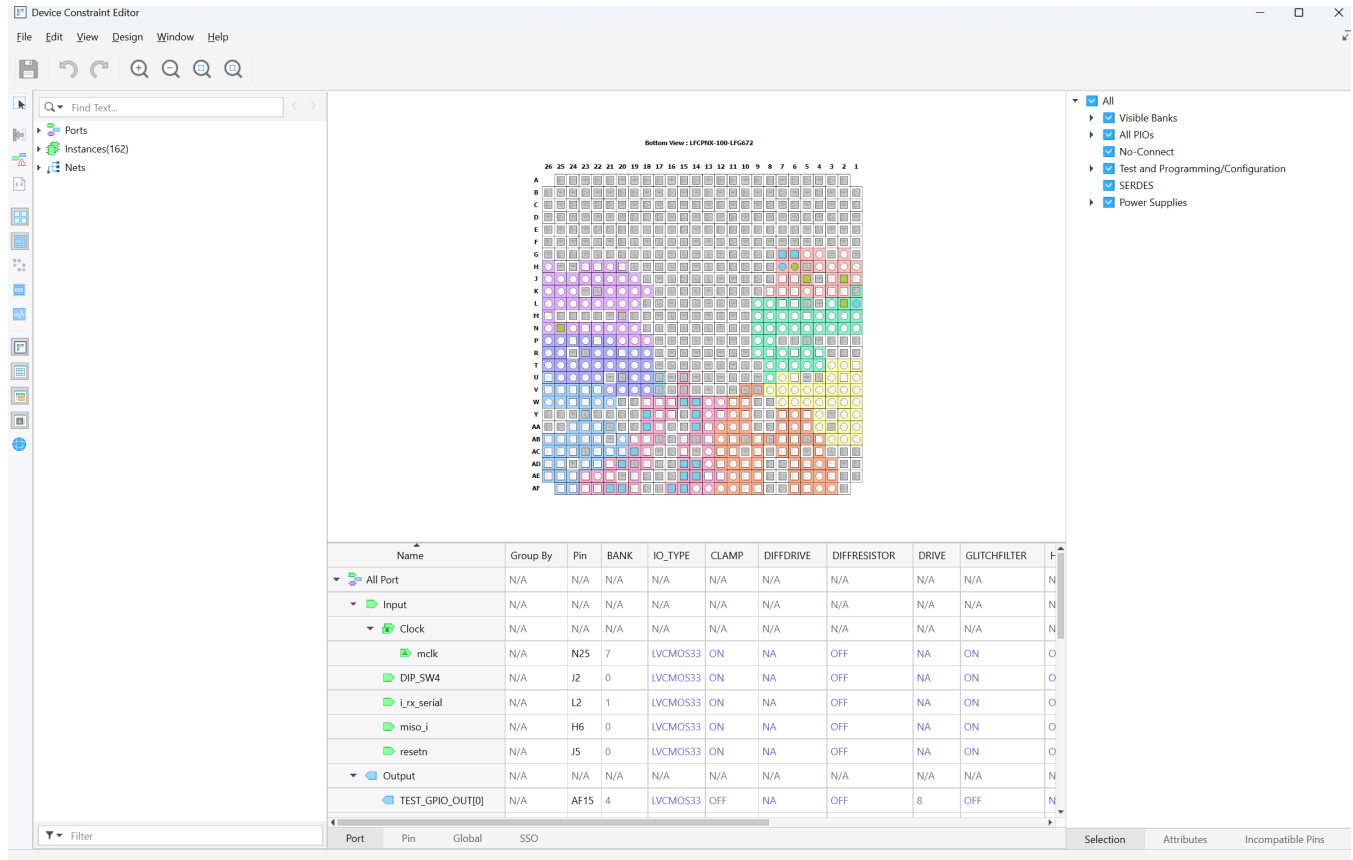
The screenshot displays the Lattice Radiant Software Source Editor interface. The main window shows a Verilog code file named `user_code_top.v`. The code defines a module `user_code_top` with various input and output signals, including `clk`, `resetsn`, `TEST_GPIO_OUT`, `TEST_UART`, `i_tx_data`, `o_tx_data_valid`, `o_tx_serial`, `o_tx_ready`, `i_rx_data`, `o_rx_data`, `o_rx_data_valid`, and `DIP_SW4`. It also includes SPI IO interface signals `clk_i`, `rst_i`, `clk_o`, and `ss_n_o`. The code includes several local parameters for address widths and FIFO depth.

The left sidebar shows a project tree with various files and folders. A red box highlights the file `./constraints/non_versa_evb.pdc` under the `Post-Synthesis Constraint Files` folder.

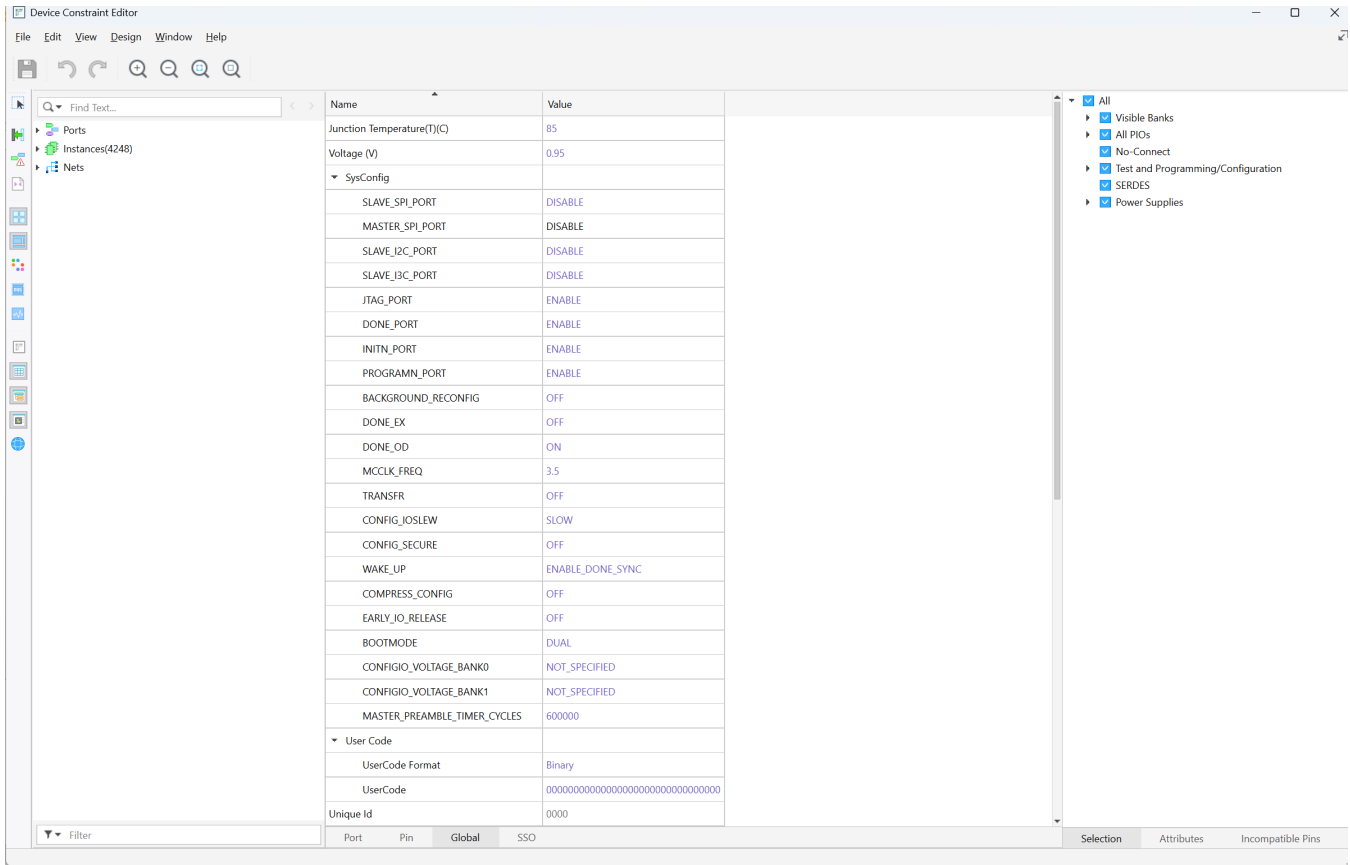
The bottom status bar shows the project summary: 1 critical, 146 warnings, 2 critical, 509 warnings, and 393 info messages. The error log displays the following messages:

- 1026001 WARNING - C:/projects/Lattice/External_Memory/2025_2/Flash_IP_MRAM_Final1/constraints/non_versa_evb.pdc (18): No port matched 'LED[0]'. [non_versa_evb.pdc:18]
- 1027013 WARNING - No port matched 'LED[0]'.
- 52281053 CRITICAL - There is no set_clock_uncertainty constraint on the PLL clock output 'CLKOS4' of instance 'u_pll_20/lsccl_inst/gen_no_refclk_mon_u_PLLPLL_inst'. Please see FPGA-AN-02059-1.0 - Lattice Radiant Timing Constraints Methodology for further details.
- 1011001 WARNING - Remove invalid constraint 'create_generated_clock -name {u_spi_flash_ctrl0_spi_clk} -source {get_pins u_pll_20/lsccl_inst/gen_no_refclk_mon_u_PLLPLL_inst/CLKOS4} -divide_by 8 {get_pins u_spi_flash_ctrl0/lsccl_spi_flash_inst/clk5_spi_rtk_c7?}'.

Once the design has been synthesized, use the Device Constraints Editor option of the Tools drop down menu to confirm the pin numbers to their respective signals, see picture below.



After the particular design has been synthesized, select the following **Global** settings, see picture below.




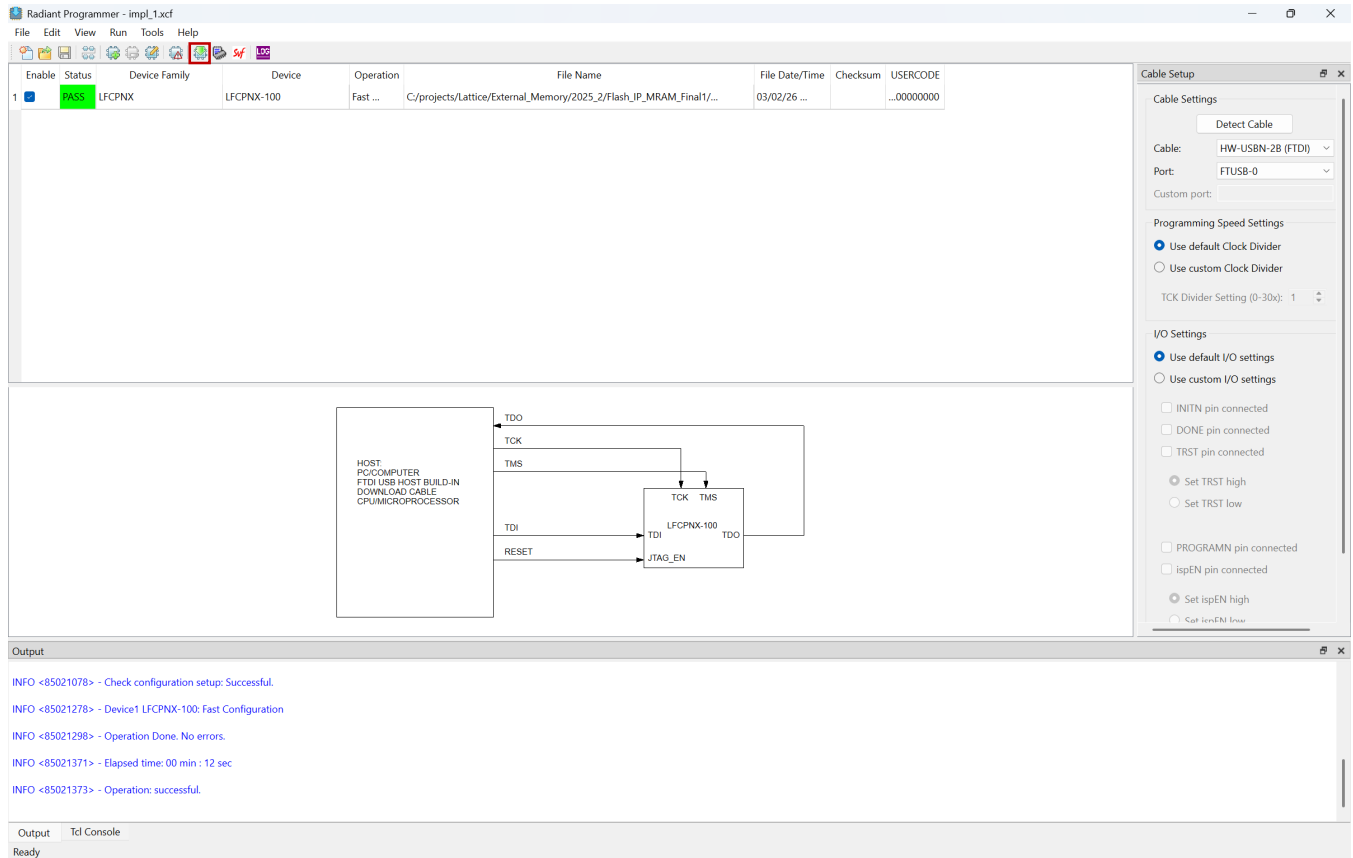
Compile the design by clicking on the Run All arrow, see picture below.

The screenshot displays the Lattice Radiant IDE interface during the synthesis phase. The top status bar indicates 'Synthesize Design completed with warning.' The main window shows the Verilog source code for 'user_code_top.v', which defines a module with various input and output signals, including clock, reset, and data buses. The code includes comments for SPI and UART interfaces and defines several parameters like FIFO_DEPTH and program sizes. The bottom console window shows the following messages:

```
Map (1 criticals, 52 warnings)
> 1026001 WARNING - C:/projects/Lattice/External_Memory/2025_2/Flash_IP_MRAM_Final1/constraints/non_versa_evb.pdc (18): No port matched 'LED[0]'. [non_versa_evb.pdc:18]
> 1027013 WARNING - No port matched 'LED[0]'.
> 1011001 WARNING - Remove invalid constraint 'ldc_set_location -site (N5) [get_ports [LED[0]]]'.
> 52281053 CRITICAL - There is no set_clock_uncertainty constraint on the PLL clock output 'CLKOS4' of instance 'u_pll_20/lscv_pll_inst/gen_no_refclk_mon_u_pll_pll_inst'. Please see FPGA-AN-02059-1.0 - Lattice Radiant Timing Constraints Methodology for further details.
```

2.5 Program the Device with Radiant Programmer

From the **Tools** menu, select **Programmer** and a new window opens, see picture below. Once the cable is detected and settings are set, program the device by clicking the Program Device icon , see picture below.



The screenshot displays the Radiant Programmer interface for an LFCPNX-100 device. The main window contains a table with the following data:

Enable	Status	Device Family	Device	Operation	File Name	File Date/Time	Checksum	USERCODE
1	PASS	LFCPNX	LFCPNX-100	Fast ...	C:/projects/Lattice/External_Memory/2025_2/Flash_IP_MRAM_Final1/...	03/02/2600000000

Below the table is a connection diagram showing a HOST PC/COMPUTER connected to an LFCPNX-100 device. The connections are: TDO, TCK, TMS, TDI, and RESET to JTAG_EN. The LFCPNX-100 device has TCK, TMS, TDI, and TDO pins.

On the right side, the Cable Setup panel is visible, showing settings for Cable (HW-USBN-28 (FTDI)), Port (FTUSB-0), and I/O Settings (Use default I/O settings).

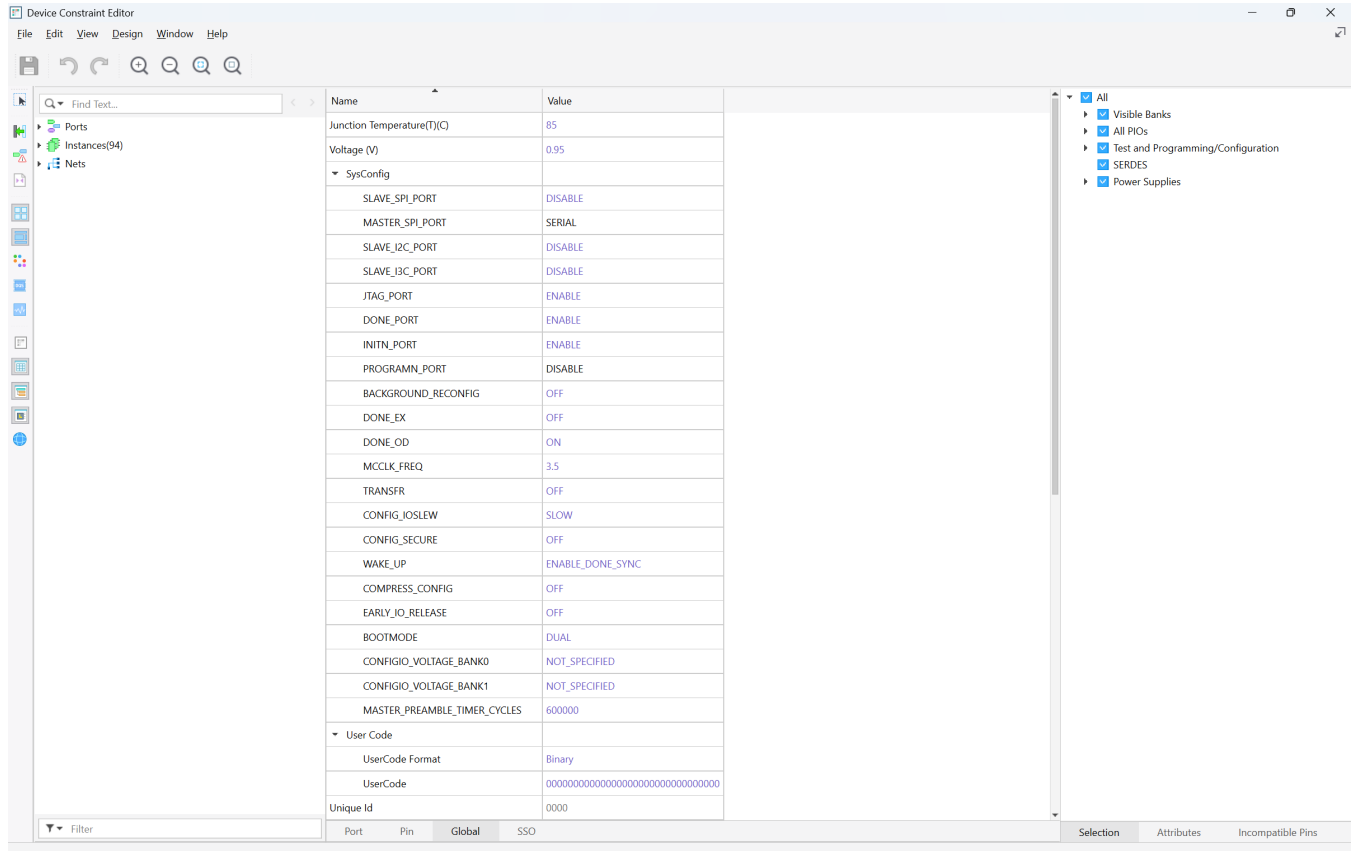
The Output window at the bottom shows the following log messages:

```
INFO <85021078> - Check configuration setup: Successful.
INFO <85021278> - Device1 LFCPNX-100: Fast Configuration
INFO <85021298> - Operation Done. No errors.
INFO <85021371> - Elapsed time: 00 min : 12 sec
INFO <85021373> - Operation: successful.
```

2.6 Verify the Hardware Interface

The project named **led_brightness** provided with the user code app note is provided for testing the hardware. The design uses switches SW[3:0] for controlling the duty cycle to the pwm output, which controls the brightness to LED[0]; LED[7:1] are used as heartbeat of the system.

Note: make sure the image configuring the FPGA has the following Global settings.



References

For more information, refer to the following documents:

- [FPGA-IPUG-02134-2-1-SPI-Flash-Memory-Controller-IP-User Guide](#)
- [FPGA-EB-02046-1-1-CertusPro-NX-Evaluation-Board](#)
- [FPGA-RD-02270-1-8-Universal-Asynchronous-ReceiverTransmitter-Reference-Design](#)
- [FPGA-TN-02099-3-3-sysCONFIG-User-Guide-for-Nexus-Platform](#)
- [CertusPro-NX 100k Pinout File \(FPGA-SC-02022\)](#)
- [fpga-tn-02145-2-1-multiboot-user-guide-for-nexus-platform](#)
- [Avalanche_Serial-Dual-QSPI-P-SRAM-for-Xilinx-Versal-ACAP-VCK-190-Evaluation-Kit-User-Guide-v1-1](#)
- [datasheet_-_ut8mrq2g](#)
- [FPGA-IPUG-02191-1-1-Distributed-Memory-Modules](#)

Revision History

Date	Revision #	Author	Change Description	Page #
4/7/2026	1.0.0	JA	Initial release.	N/A

The following United States (U.S.) Department of Commerce statement shall be applicable if these commodities, technology, or software are exported from the U.S.: These commodities, technology, or software were exported from the United States in accordance with the Export Administration Regulations. Diversion contrary to U.S. law is prohibited.

Frontgrade Colorado Springs LLC (Frontgrade) reserves the right to make changes to any products and services described herein at any time without notice. Consult an authorized sales representative to verify that the information in this data sheet is current before using this product. The company does not assume any responsibility or liability arising out of the application or use of any product or service described herein, except as expressly agreed to in writing; nor does the purchase, lease, or use of a product or service convey a license under any patent rights, copyrights, trademark rights, or any other of the intellectual rights of the company or of third parties.