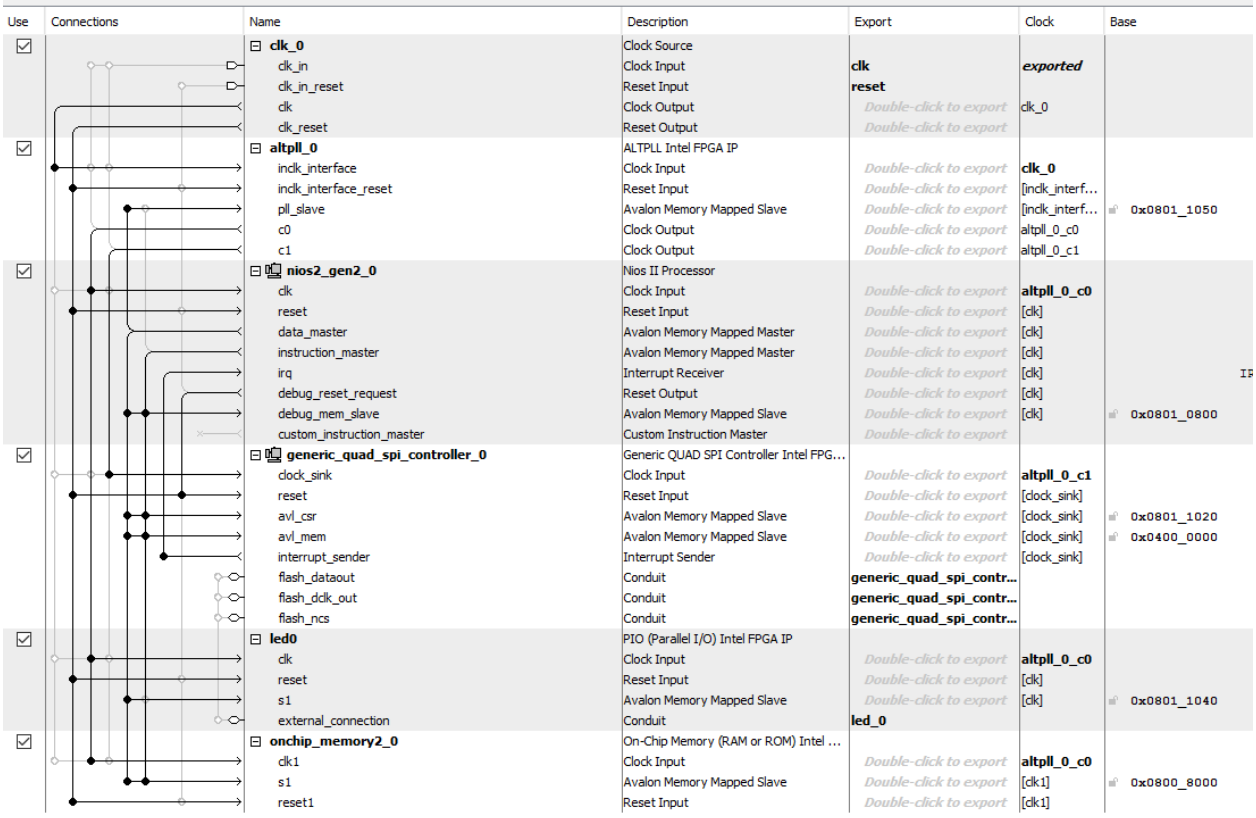
SUBJECT:- NIOS PROCESSOR BOOT USING QSPI FLASH

This document follows all the steps to boot Nios from QSPI FLASH as mentioned in AN 730: Nios II Processor Booting Methods in MAX 10 FPGA Devices (eeweb.com) **(OPTION-5 :** Nios II processor application copied from QSPI flash to RAM using boot copier**)**

Device used:- MAX10 DEVELOPMENT KIT(REV C)

Quartus Software Version:- Quartus Prime 21.1

1- QSYS Design.



Use	Connections	Name	Description	Export	Clock	Base
☑		**clk_0**	Clock Source			
		clk_in	Clock Input	**clk**	*exported*	
		clk_in_reset	Reset Input	**reset**		
		clk	Clock Output	Double-click to export	clk_0	
		clk_reset	Reset Output	Double-click to export		
☑		**altpll_0**	ALTPLL Intel FPGA IP			
		indlk_interface	Clock Input	Double-click to export	**clk_0**	
		indlk_interface_reset	Reset Input	Double-click to export	[indlk_interf…	
		pll_slave	Avalon Memory Mapped Slave	Double-click to export	[indlk_interf…	0x0801_1050
		c0	Clock Output	Double-click to export	altpll_0_c0	
		c1	Clock Output	Double-click to export	altpll_0_c1	
☑		**nios2_gen2_0**	Nios II Processor			
		clk	Clock Input	Double-click to export	**altpll_0_c0**	
		reset	Reset Input	Double-click to export	[clk]	
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]	
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]	
		irq	Interrupt Receiver	Double-click to export	[clk]	
		debug_reset_request	Reset Output	Double-click to export	[clk]	
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0801_0800
		custom_instruction_master	Custom Instruction Master	Double-click to export		
☑		**generic_quad_spi_controller_0**	Generic QUAD SPI Controller Intel FPG…			
		clock_sink	Clock Input	Double-click to export	**altpll_0_c1**	
		reset	Reset Input	Double-click to export	[clock_sink]	
		avl_csr	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]	0x0801_1020
		avl_mem	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]	0x0400_0000
		interrupt_sender	Interrupt Sender	Double-click to export	[clock_sink]	
		flash_dataout	Conduit	**generic_quad_spi_contr…**		
		flash_dclk_out	Conduit	**generic_quad_spi_contr…**		
		flash_ncs	Conduit	**generic_quad_spi_contr…**		
☑		**led0**	PIO (Parallel I/O) Intel FPGA IP			
		clk	Clock Input	Double-click to export	**altpll_0_c0**	
		reset	Reset Input	Double-click to export	[clk]	
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0801_1040
		external_connection	Conduit	**led_0**		
☑		**onchip_memory2_0**	On-Chip Memory (RAM or ROM) Intel …			
		clk1	Clock Input	Double-click to export	**altpll_0_c0**	
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x0800_8000
		reset1	Reset Input	Double-click to export	[clk1]	

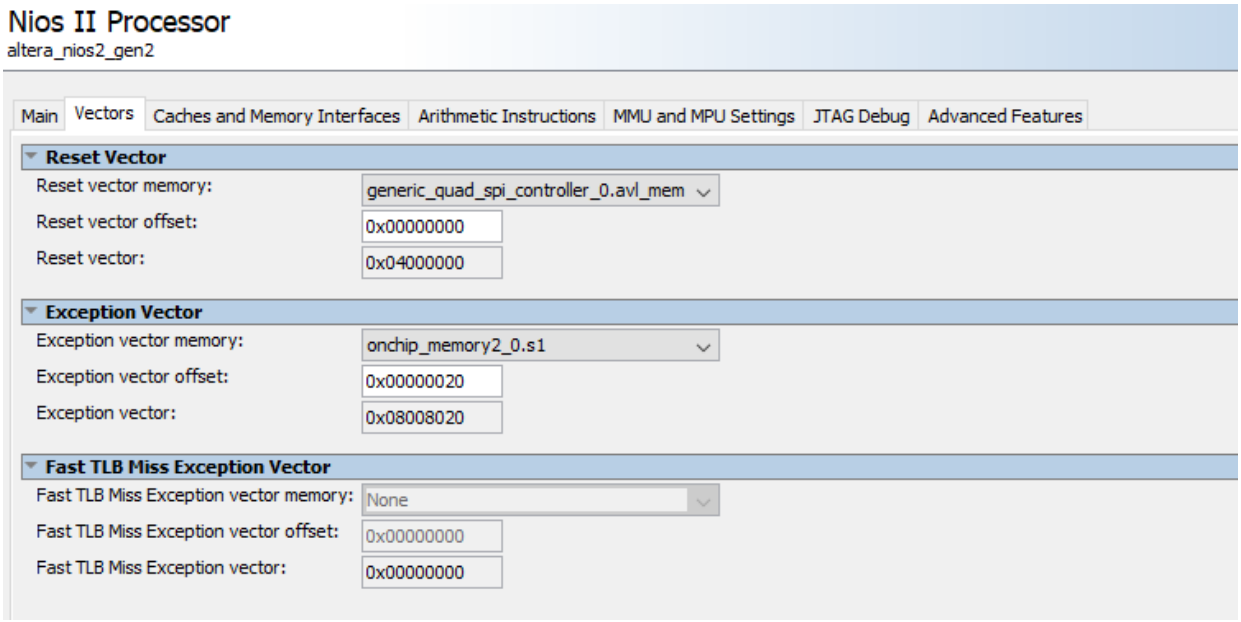
Clock source to NIOS Processor – **50MHz**

Clock source to PLL - **50MHz**

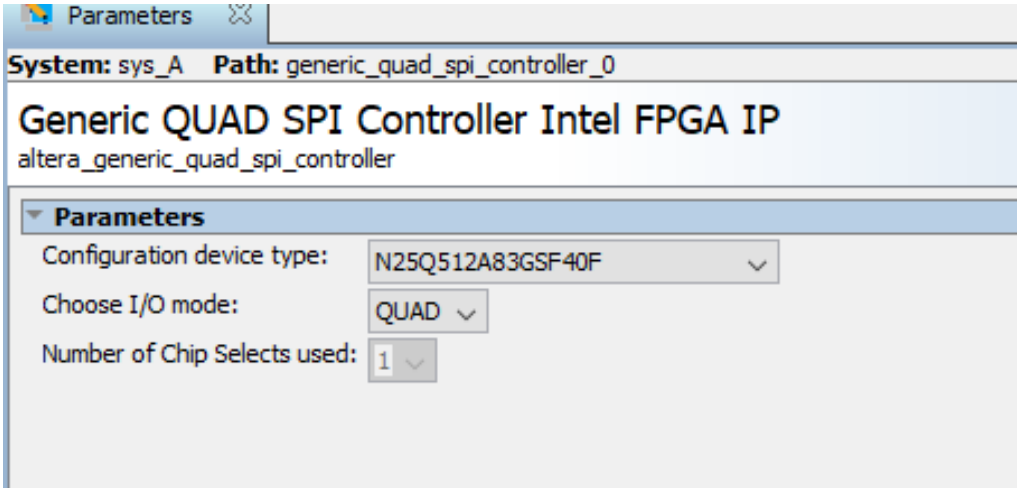
Clock source to Generic Quad SPI Controller through PLL(c1)- **25MHz**

Clock source to On-Chip Memory(c1)- **50MHz**

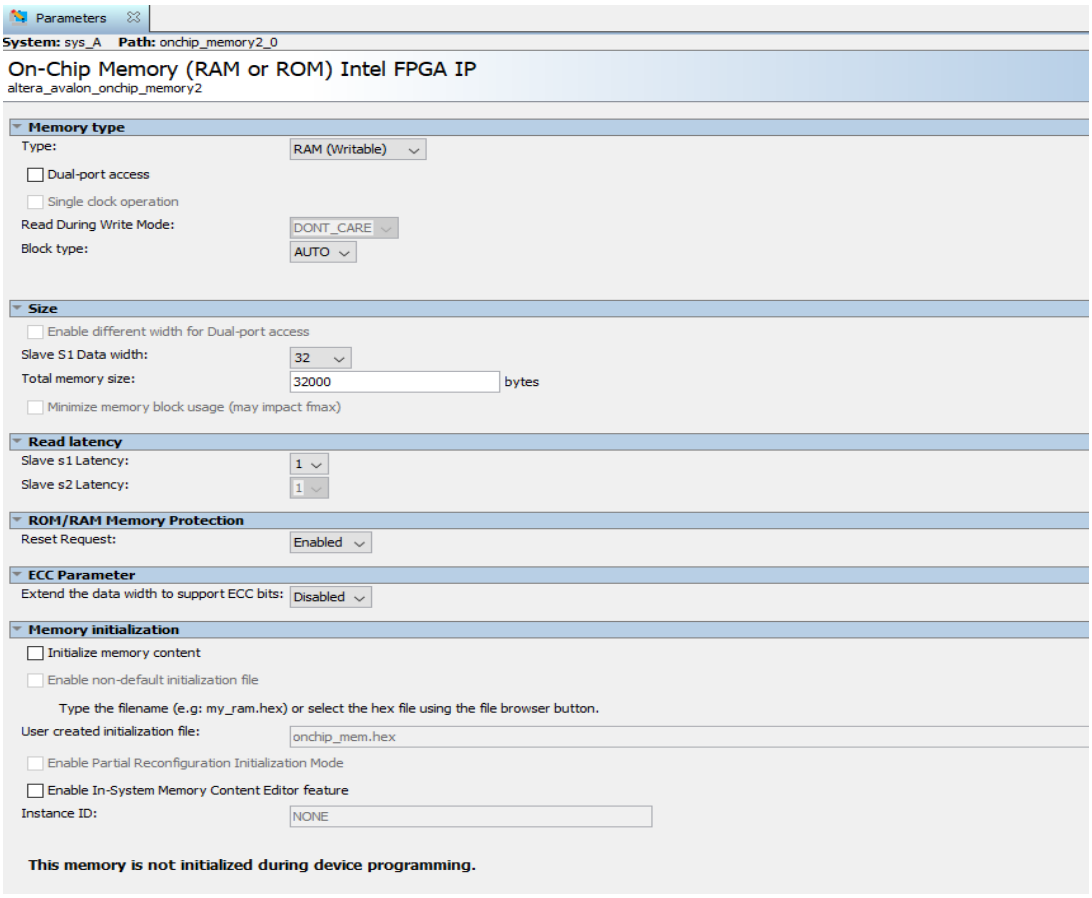
Nios Processor Parameters:-



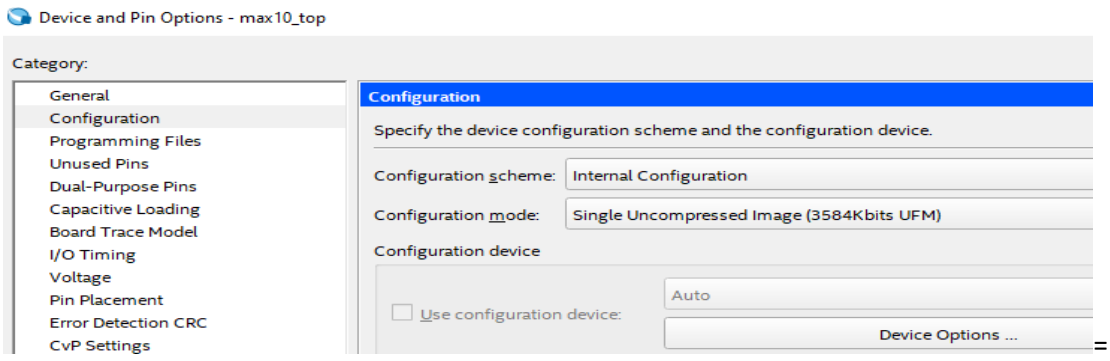
QUAD SPI Controller Parameters:-



On-Chip RAM:-



The Configuration Mode was set to **Single Uncompressed Image(3584 KBits UFM).**



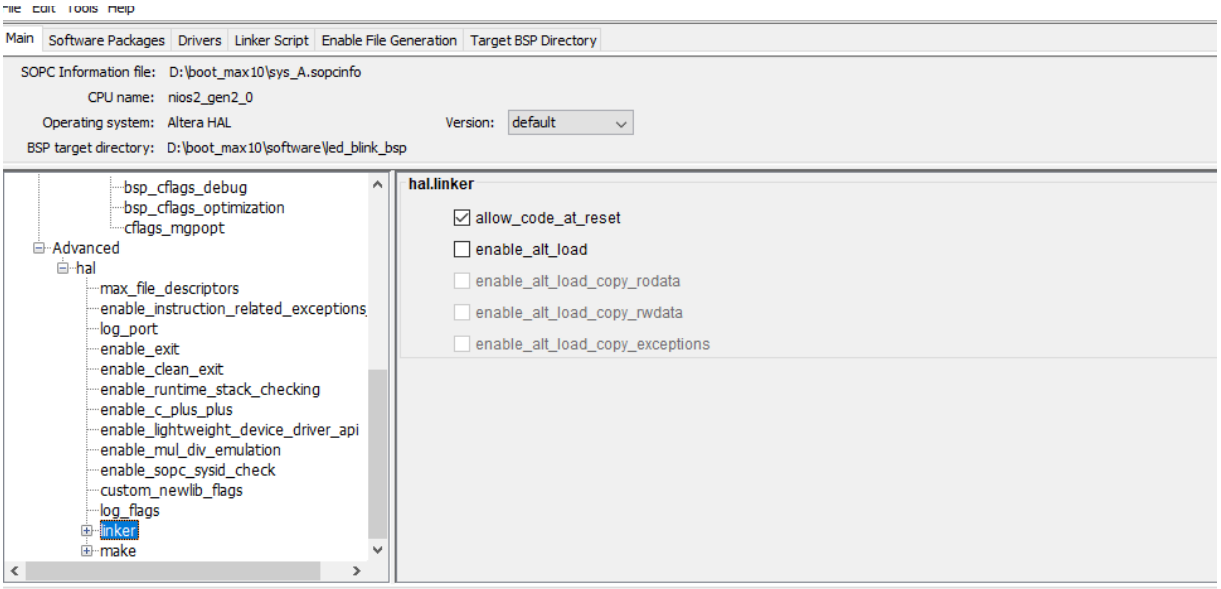
The HDL was generated of this QSYS Design and Full Compilation was done.

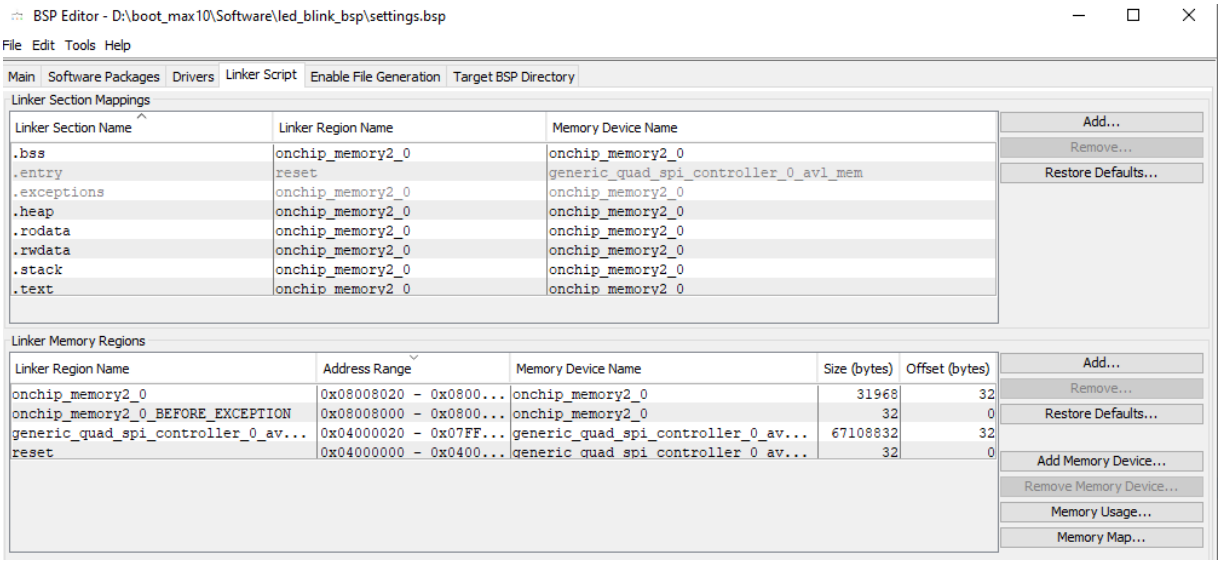
NIOS II Software Build Tool Settings.

The C code written for this is as follow:-

```
 * "Hello World" example.

#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include <unistd.h>

int main()
{



    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x1);
    usleep(1000000);
    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x0);
    usleep(1000000);

    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x1);
    usleep(1000000);
    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x0);
    usleep(1000000);

    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x1);
    usleep(1000000);
    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x0);
    usleep(1000000);

    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x1);
    usleep(1000000);
    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x0);
    usleep(1000000);

    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x1);
    usleep(1000000);
    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x0);
    usleep(1000000);

    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x1);
    usleep(1000000);
    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x0);
    usleep(1000000);

    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x1);
    usleep(1000000);
    IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE,0x0);
    usleep(1000000);
}
```

Advanced.hal.linker Settings in BSP EDITOR





The hex file was then generated for the above settings.

2- Converting the hex file generated into POF File.

The following settings were applied for generating the POF file as mentioned in **AN 730**

1. In Quartus II, click on **Convert Programming Files (.pof)** from the **File** tab.

2. Choose **Programmer Object File** as **Programming file type**.

3. Set **Mode** to **1-bit Passive Serial.**

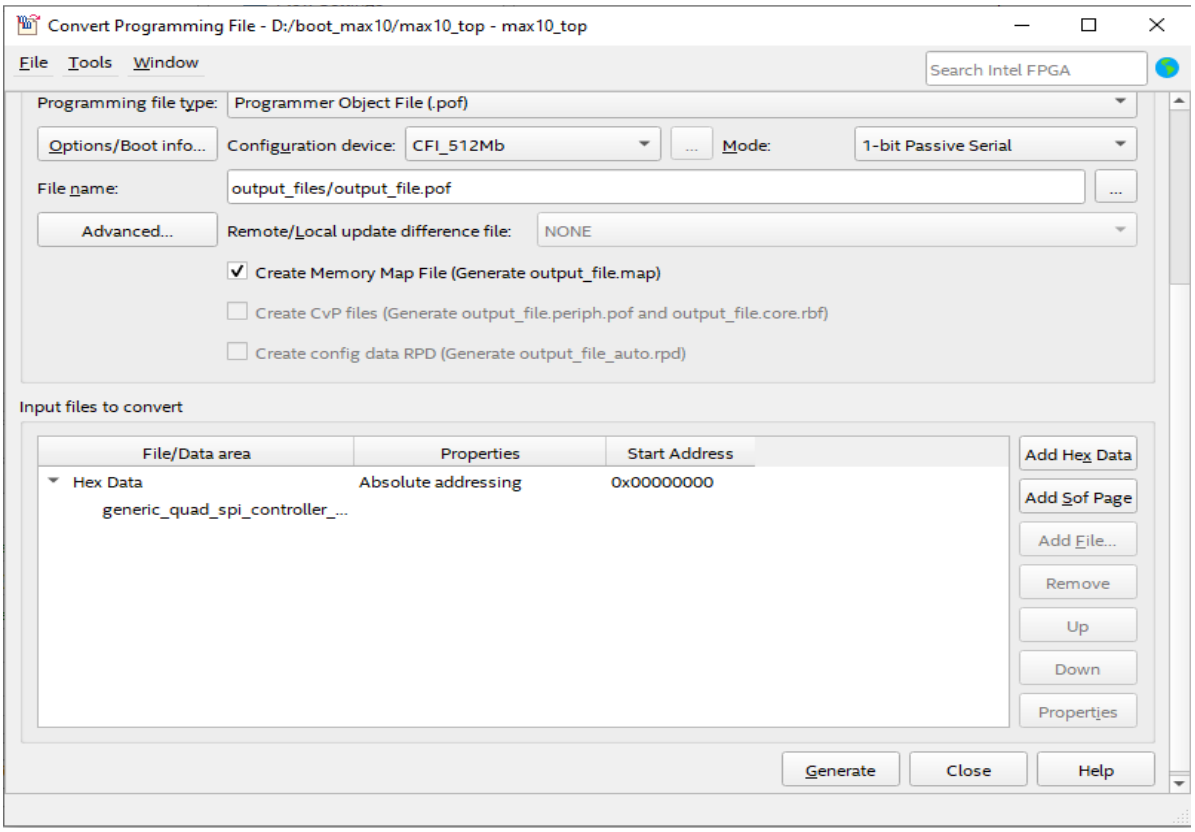
4. Set **Configuration device** to **CFI_512Mb.**

5. Change the **File name** to the desired path and name.

6. Remove the **SOF Page_0**.

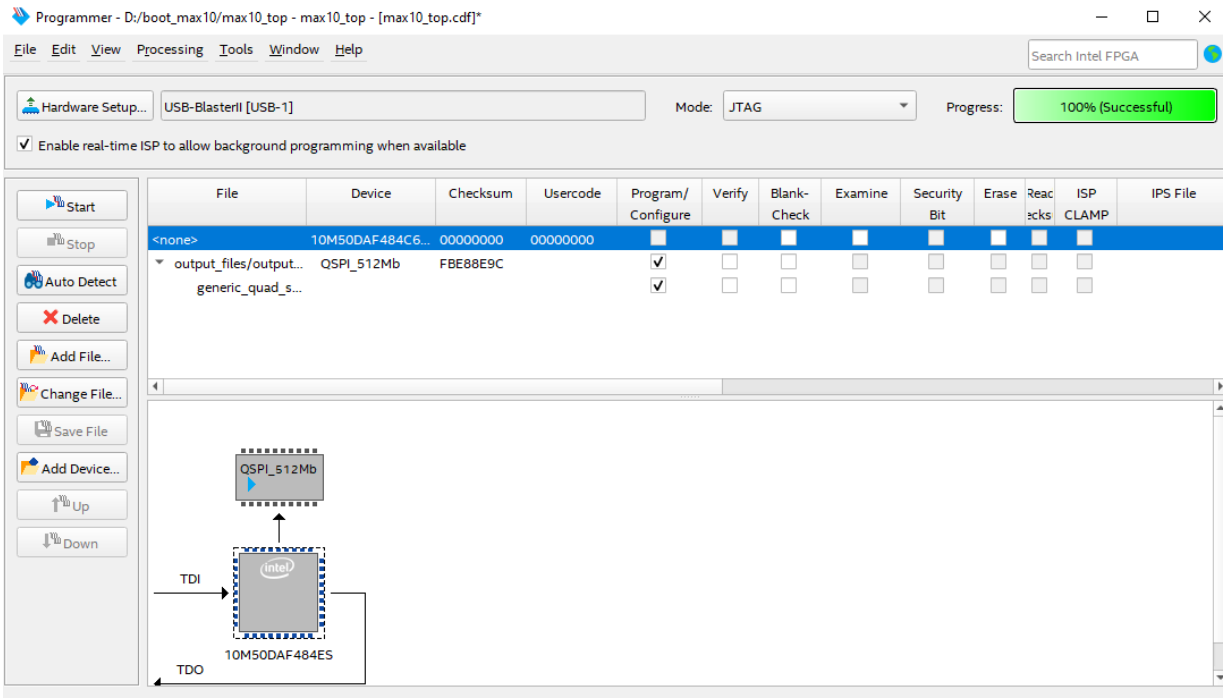
7. Click on **Add HEX Data**, choose the HEX file generated in **HEX Generation section**. Select **Absolute Addressing** and click **OK.**

8. Click **Generate** to create the .pof file.



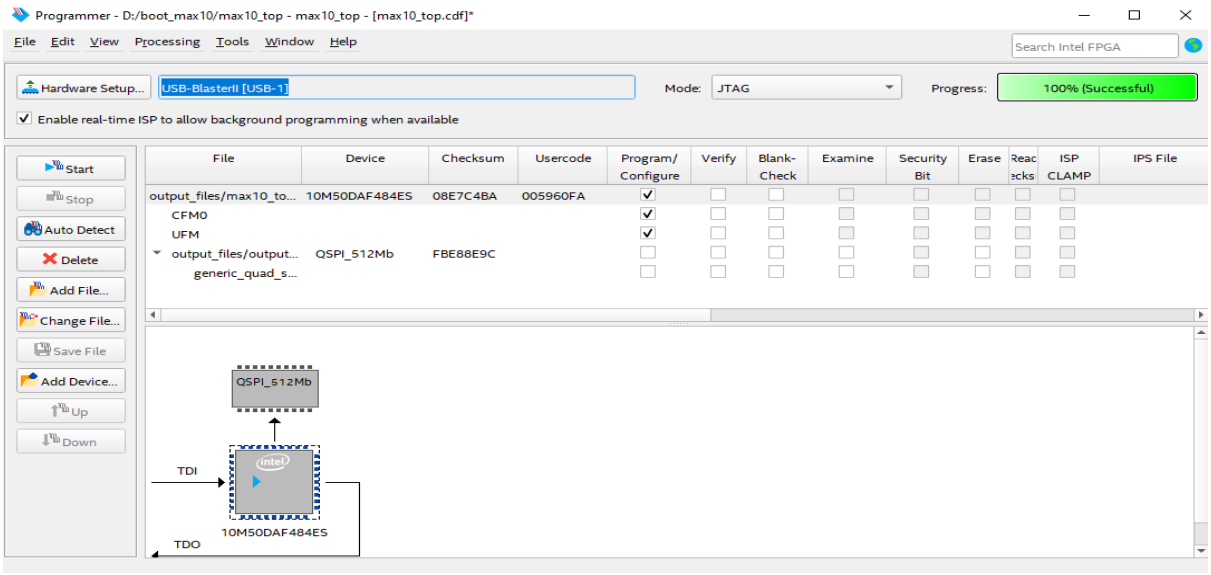
3- Programming the above generated POF into QSPI FLash.

--> To detect the **QSPI Flash** in the Programmer, a **sof** of **Parallel Flash Loader** was uploaded into the MAX 10 device before programming the QSPI flash.(As mentioned in **AN 730**)



4- Programming the POF generated during Compilation into MAX10 Device

After programming the **POF File** (containing the Hex File) successfully, the POF File generated during the **Compilation Process** was programmed into the MAX10 Device.



Result: After completing all the steps, on Power Cycle of MAX10 Kit the LED[0] was not blinking according to the software code. NIOS Processor does not boot using QSPI Flash.