ForgeFPGA Configuration Guide

This document describes how to configure the ForgeFPGA core using three different configuration bitstream sources: External SPI Flash, Internal OTP, MCU as a host. It also details how to debug using the Development Board, Socket Adaptor Board, and the Evaluation Board to debug.

Contents

1. Terms and Definitions

CPOL	Clock Polarity
CPHA	Clock Phase
EVB	Evaluation Board
FPGA	Field-Programmable Gate Array
MCU	Micro Controller Unit
OTP	One Time Programmable on chip NVM
SPI	Serial Peripheral Interface

2. References

For related documents and software, please visit our website:

Download our free ForgeFPGA Workshop software [1] and follow the steps in this user guide. User can reference [2] for the datasheet. Renesas Electronics provides a complete library of application notes [3] featuring design examples as well as explanations of features and blocks within the Renesas IC. Please visit the product page to download the following:

[1] GoConfigure Software Hub, Software Download, Renesas Electronics Corporation
[2] SLG47910 Datasheet, Renesas Electronics Corporation
[3] Application Notes, ForgeFPGA Application Notes & Design Files, Renesas Electronics
[4] ForgeFPGA Advanced Development Board User Manual, Renesas Electronics Corporation
[5] ForgeFPGA Socket Adaptor User Manual, Renesas Electronics Corporation
[6] ForgeFPGA Evaluation Board User Manual, Renesas Electronics Corporation
[7] ForgeFPGA Software User Guide, Renesas Electronics Corporation

3. Introduction

An internal Configuration Circuit is used to configure the ForgeFPGA core. The configuration can be done using three different configuration bitstream sources:

- External SPI Flash
- Internal OTP
- MCU as a host

The ForgeFPGA Designer Software is used to generate bitstreams. The schematic in Figure 1 shows a block diagram of the SLG47910 Configuration Circuit, the external MCU Host, and SPI Flash interface. The four configuration pins are GPIO3 (SPI_CLK), GPIO4 (SPI_SS, Chip Select), GPIO5 (SPI_SI, serial input), and GPIO6 (SPI_SO, serial output). GPIO6 is also used as a Config Done signal. Table 1 shows which modes activate the SPI Master and SPI Slave blocks during configuration.

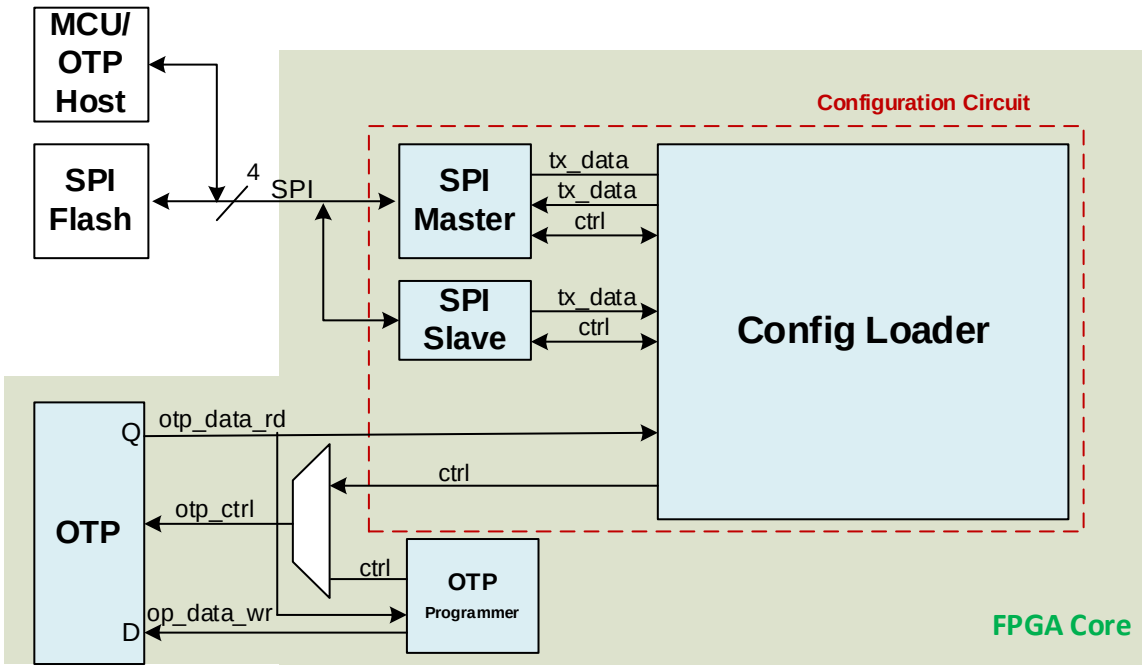


Figure 1: SLG47910 Programming & Configuration Interface

Table 1: Configuration Modes

Configuration Mode	SPI Block Activated	Clock Source	Corresponding Function in SW
SPI	FPGA Master	FPGA	Used to read/write the external Flash Memory
MCU	FPGA Slave	MCU	Testing in Emulation Mode
OTP	None	External FPGA	Programming and reading the OTP memory

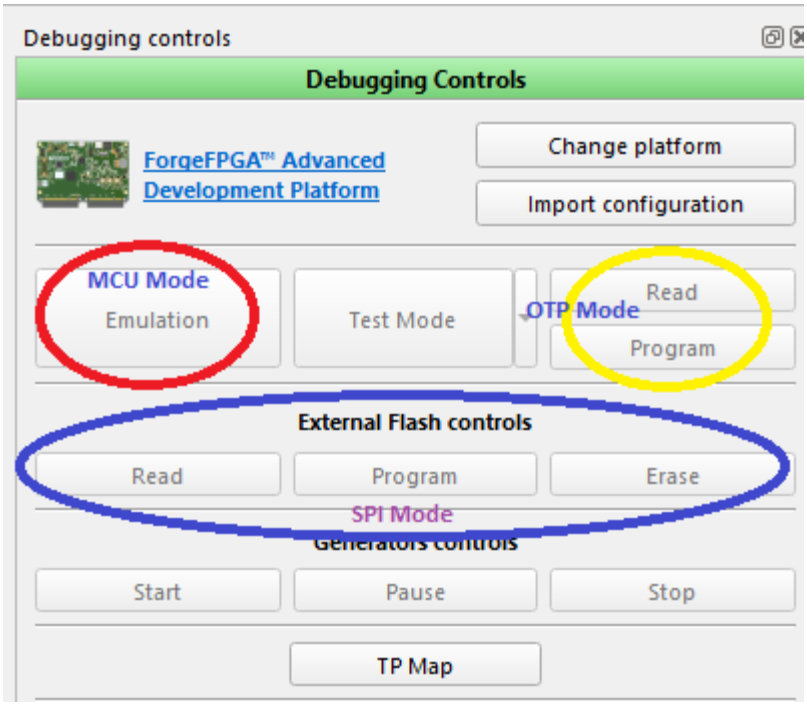


Figure 2 : Configuration Modes in Software

4. General SPI Interface

A 4-wire SPI device has four signals (see Figure 3):

1. **SCLK: Serial Clock** (output from Master). When the master communicates with the slave, the data on the MOSI or MISO pin will be synchronized with the Serial Clock. In the SPI protocol, the master produces the clock, and the slave will only receive the clock, and has no control over the serial clock.
2. **MOSI: Master-Out Slave-In** (data output from master). MOSI is a data pin. This pin is used to transmit data from the master to the slave device. Whenever the master sends data, that data will be collected over the MOSI pin by the slave.
3. **MISO: Master-In Slave-Out** (data output from slave). MISO is a data pin. This pin is used to transmit data from the slave to the master. Whenever the slave sends data, that data will be collected over the MISO pin by the master.
4. **SS: Slave-Select** (often active low, data output from master). Depending on the SPI and slave select setting, the SS pin is used to select an individual slave device for communication. When there is one master and only one slave device, then the SS pin is not required. This slave select pin is only necessary when the master is communicating with multiple different slaves. So, the master can select which slave device to which the master wants to communicate. There is a dedicated SS pin for each slave device connected to the master.

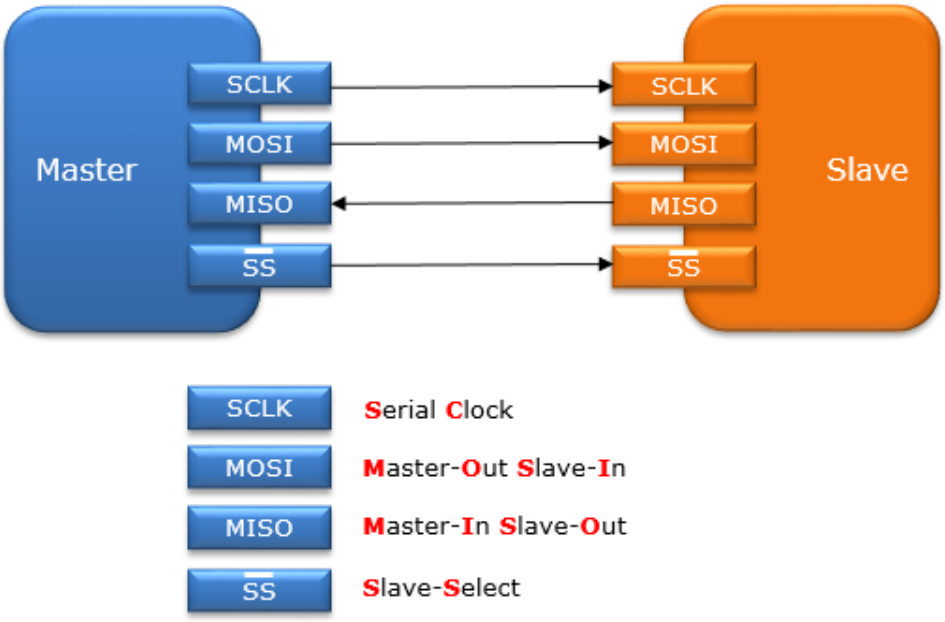


Figure 3: SPI Interface

4.1 SPI Modes with Clock Polarity and Clock Phase

In SPI, the master can select the Clock Polarity (CPOL) and Clock Phase (CPHA). The CPOL bit sets the polarity of the clock signal during the idle state. The idle state is defined as the period when SS is transitioning. The CPHA bit selects the clock phase. Depending on the CPHA bit, the rising or falling clock edge is used to sample and/or shift the data. Depending on the CPOL and CPHA bit selection, four SPI modes are available (see Table 2).

Table 2: SPI Modes

SPI Modes	CPOL	CPHA	Clock Polarity in Idle State	Clock Phase Used to Sample and/or Shift the Data
0	0	0	Logic LOW	Data sampled on the rising edge and shifted out on the falling edge
1	0	1	Logic LOW	Data sampled on the falling edge and shifted out on the rising edge
2	1	1	Logic HIGH	Data sampled on the falling edge and shifted out on the rising edge
3	1	0	Logic HIGH	Data sampled on the rising edge and shifted out on the falling edge

Figure 4 shows the data on the MOSI and MISO line. The green dotted lines show the end and the beginning of the transmission. The data sampling is shown with orange dotted lines which correspond to the rising or falling edge depending on the selected SPI Mode. The shifting edge of the data is depicted using the blue doted lines. Figure 4 depicts SPI Mode 0 with CPOL = 0 and CPHA = 0 with the clk idle state = 0 and hence the data is sampled on the rising edge and shifted on the falling edge according to Table 2.

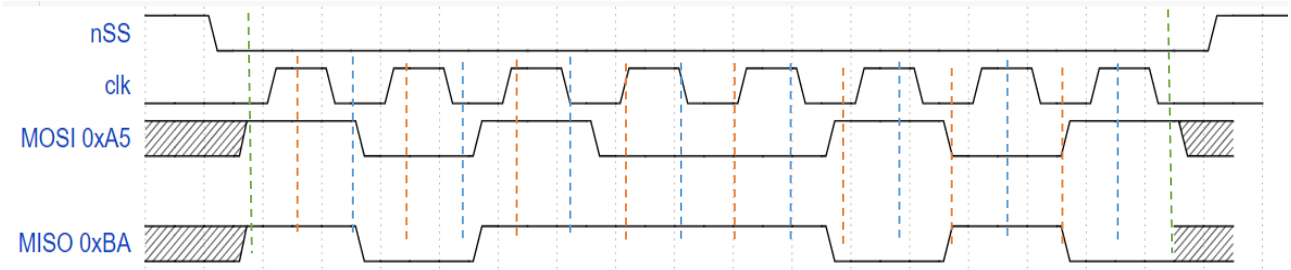


Figure 4: SPI Mode 0

5. Development Board

There are two prerequisite steps that need to be performed before the design is sent to the device and can be further configured with the development board.

a. **RTL Synthesis**: After creating your desired Verilog Code in the HDL Editor Window of the ForgeFPGA Workshop, the next step is to create a Netlist of your design. This can be done with the help of the built-in Synthesis tool that takes the input design and produces a Netlist out of it. While performing RTL synthesis, the input design is analyzed and converted into a gate-level representation. See the ForgeFPGA Software User Guide [7]

b. **Generating the Bitstream**: To prepare your design to be sent to the device you need to perform the Place-and-Route procedure, which takes the elements of the synthesized netlist and maps its primitives to the FPGA physical resources. You can do this after successfully generating the netlist and pressing the "Generate Bitstream" button on the control panel. After completing these two steps, the design will be successfully sent to the device. See the ForgeFPGA Software User Guide [7]

Under the "Debug" button on the toolbar, select the ForgeFPGA Development Board as the platform (see Figure 5) which needs to be configured.

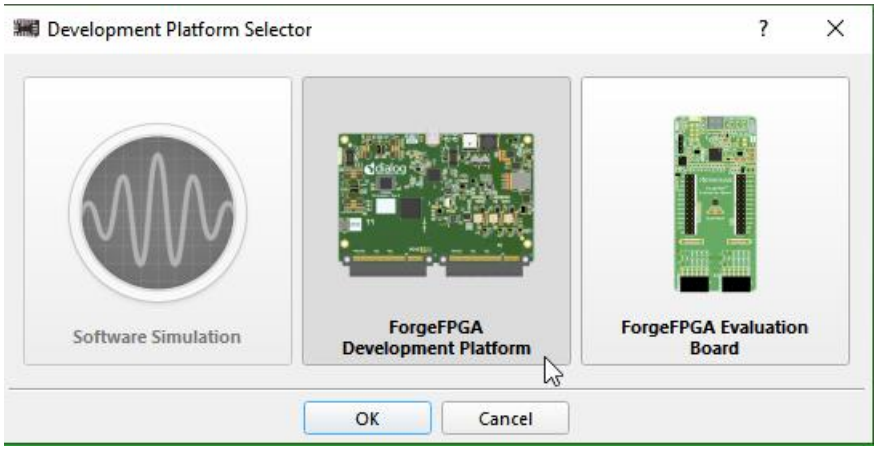


Figure 5: ForgeFPGA Development Platform Selection

The FPGA Development Board is a multi-functional tool that allows the user to develop their FPGA designs easily by providing on-board power sources, digital and analog signal generation, and logic analysis capabilities. The FPGA Development Board can connect to additional external boards called socket adaptors [5]. The function of the socket adapter board is to implement a stable electrical connection between the pins of the chip under test and the FPGA Development Board. To implement this, the FPGA Development Board has a Dual PCIe connector. This connector has 40 differential pairs (80 digital channels), 32 analog pins, service pins, and power pins. This Dual PCIe connector is universal and can be applied to multiple socket adapter boards.

Driven by the free software, the FPGA Development Board can be configured to work as any one of several traditional instruments, which include:

- Logic Analyzer
- Digital pattern generator
- 8-channel analog Arbitrary Waveform Generator (AWG)
- Precision ADC
- Three programmable power supplies (+0.6 V…+3.3 V). The maximum available output current 2 A. The same voltage is supplied to the GPIO, for keeping the logic level compatibility with the circuit under test.

Also, for more information on configuration using the Advanced Development Board, please refer to the Advanced Development Board User Manual [4].

6. Evaluation Board

To enter the debug controls of the Evaluation Board, the correct platform on which we need to configure our device needs to be selected. Under the "Debug" button on the toolbar, select the ForgeFPGA Evaluation Board as the platform (see Figure 6).

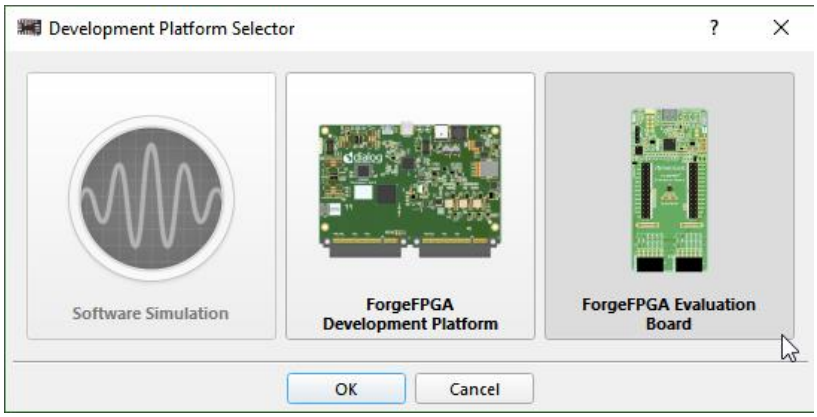


Figure 6: ForgeFPGA Development Platform Selection

The ForgeFPGA Evaluation board provides SLG47910 IC hardware support for design emulation and real time testing. The board consists of 2 main blocks: the Programmer and the SLG47910 IC with external connectors.

For more information on configuration using the EVB, please refer to the Evaluation Board User Manual [6].

7. OTP Read/Write

The Configuration RAM is a volatile memory that stores the FPGA design. The OTP stores the data that can be read and loaded onto the Configuration RAM. The SLG47910 contains three blocks of 4k x 32-bit One-Time Programmable (OTP) Non-Volatile Memory (NVM), which are interfaced via the dedicated SPI Slave circuit block.

The user can read from or write to the OTP block through the SPI interface pins on the SLG47910. If the OTP block of the SLG47910 has been programmed, upon POR, the contents of the NVM will be loaded into the device's internal configuration RAM. User can program the OTP and read the contents of the OTP from the Read & Program button in the software under the Debugging Controls (See Figure 2)

7.1 Writing to the OTP Block

Writing to the OTP is done by using the SPI Slave interface. The OTP write starts with ramp up of the voltage signals V_{DDC} to 1.1 V and V_{DDIO} to 2.75 V which sets the NVM into write program mode. There is an otp_controller that sequences the internal signals to enable the OTP write.

Once the overdrive and program mode inputs are active, the otp_controller will decode the spi_slave data, and then send the data address and control information to the otp_write_controller block to initiate the OTP write. Table 3 shows the OTP Write Packet format. Table 4 shows the Write/Read option bits. The last Write packet is indicated by setting Byte8 bit [6]. Reserve bits are indicated by R and the parity bit by P. After writing to the OTP the write data should be checked using the OTP read command. Once the SLG47910 OTP has been written and after POR (or bringing the PWR pin LOW), the MCU, OTP write, and OTP read will be disabled. The SLG47910 will only load program data from the OTP. This is a design security feature (see Figure 7).

Writing to the OTP has the following steps:

1. Wait for POR and PLL lock time (1300 µs), then send the Signature Bytes through SPI_MOSI (GPIO5) by keeping SPI_SS (GPIO4) LOW.
2. After the verifying that the Signature bytes match, GPIO9 (Config-Sig match) goes HIGH and sends the OTP write command packets after a delay of 80 µs with a delay of 18 µs between the1st and 2nd packet and a delay of 10.11 µs delay for any subsequent packets.
3. The last write packet is indicated by Byte8[6] = 1.
4. When done writing to the OTP bring PWR = LOW which resets the device, then chk_otp_en = 1.

Table 3: OTP Write Packet Format (Incoming)

Bits	0	1	2	3	4	5	6	7
Byte 1	W/Rn (1)	SP/AP (0)	R	R	R	O1_A [15]	O1_A [14]	O1_A [13]
Byte 2	O1_A [12]	O1_A [11]	O1_A [10]	O1_A [8]	O1_A [7]	O1_A [6]	O1_A [5]	O1_A [4]
Byte 3	O1_A [3]	O1_A [2]	O1_A [1]	O1_A [0]	O1_D [3]	O1_D [2]	O1_D [1]	O1_D [0]
Byte 4	O2_A [15]	O2_A [14]	O2_A [13]	O2_A [12]	O2_A [11]	O2_A [10]	O2_A [8]	O2_A [7]
Byte 5	O2_A [6]	O2_A [5]	O2_A [4]	O2_A [3]	O2_A [2]	O2_A [1]	O2_A [0]	O2_D [3]
Byte 6	O2_D [2]	O2_D [1]	O2_D [0]	O3_A [15]	O3_A [14]	O3_A [13]	O3_A [12]	O3_A [11]
Byte 7	O3_A [10]	O3_A [8]	O3_A [7]	O3_A [6]	O3_A [5]	O3_A [4]	O3_A [3]	O3_A [2]
Byte 8	O3_A [1]	O3_A [0]	O3_D [3]	O3_D [2]	O3_D [1]	O3_D [0]	Last	P

Address and Data

OTP1
OTP2
OTP3

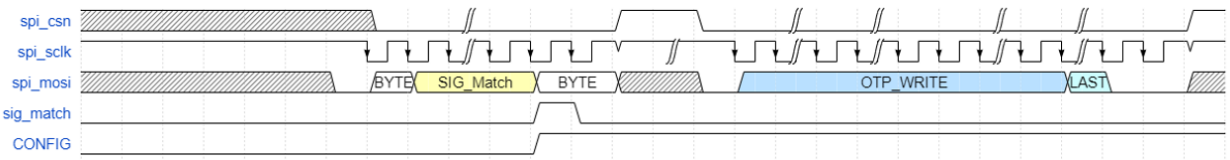


Figure 7: OTP Write Waveforms

Table 4: Write/Read & SP/AP

Byte 1 [0:1] of OTP Packet format	Comments
2'b00: Read mode	Follows format in Table 5
2'b01: Reserved	Not Used
2'b10: Write mode	Follows format in Table 3
2'b11: Return	Exit Read/Write OTP

7.2 Reading from the OTP Block

Reading of the NVM OTP via SPI is done by sending a three-byte command packet, which is met with a five-byte response packet containing the requested NVM data. A diagram of the read command packet structure is given in Figure 8.

Reading from the OTP has the following steps:

1. Wait for POR and the PLL lock time (1300 µs), after that the Signature bytes are sent through SPI_MOSI (GPIO5) by keeping SPI_SS (GPIO4) LOW.
2. After verifying that the Signature bytes match, GPIO9 (Config-Sig match) goes HIGH and then sends the OTP Read command packet on SPI_MOSI (GPIO5) after a delay of 80 µs, OTP read data will arrive at SPI_MISO (GPIO6) by keeping SPI_SS (GPIO4) LOW.

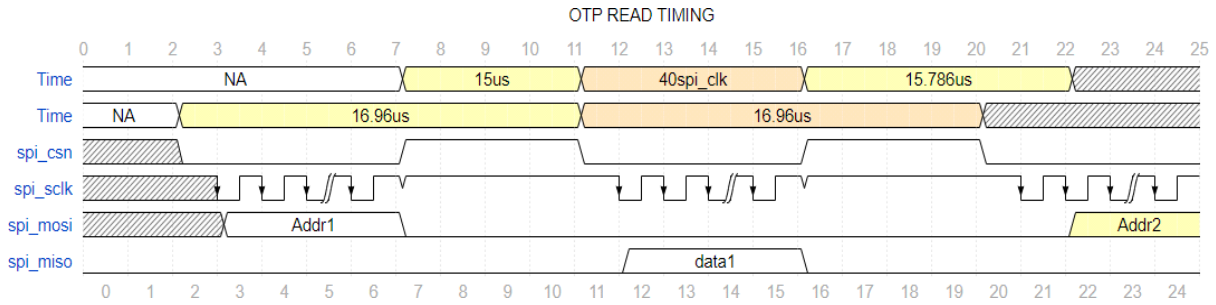


Figure 8: OTP Read Timing

7.3 Read Command Structure

Table 5: NVM OTP Read Command Packet Format

Bits	0	1	2	3	4	5	6	7
Byte 1	nRead (0)	0	R	R	R	R	R	R
Byte 2	A [18]	A [17]	A [16]	A [15]	A [14]	A [13]	A [12]	A [11]
Byte 3	A [10]	A [4]	A [3]	A [2]	A [1]	A [0]	0	P

Byte 1 of the packet will remain the same across all read operations and contains only the read command bit (Byte1[0]). Byte1 Bits [7:2] are reserved (R) bits. A read address is then provided across Bytes 2 and 3, before a parity bit (P) is provided at the MSB of Byte 3. This read address is separated into two sections. The first section is comprised of bits A [18:17] and determines which of the three 4 k x 32 NVM blocks to read from (named OTP1, OTP2, and OTP3). The second section is comprised of bits A [16:10] and bits A [4:0] and determines the address of the specified NVM block to read. The parity bit is calculated by performing an AND operation for all incoming bytes, excluding the parity bit = (^Byte1) ^(^Byte2) ^(^[Byte3[0:6]]).

Table 6: NVM Block Selection

NVM Block Selection		Address to Read											
A [18]	A [17]	A [16]	A [15]	A [14]	A [13]	A [12]	A [11]	A [10]	A [4]	A [3]	A [2]	A [1]	A [0]

8. SPI Programming (Master Mode)

In SPI Master mode, the communication starts by driving SPI_SS LOW, and then sends a “Release from Power-down” command (0xAB) to the SPI Flash. The timing diagram below provides an example waveform. This initial command wakes up the SPI Flash if it is already in Deep Power-down Mode (see Figure 9). The SPI Master transmits data on the SPI_SO output, on the falling edge of the SPI_CLK output. Currently, no data is sent on SPI_SI. After sending the last command bit, the SPI_SS is de-asserted HIGH, completing the command. A minimum buffer of 300 µs is given before sending the next SPI Flash command. Master SPI Frequency is 12.5 MHz with MCU in slave mode.

Table 7 represents the pins of the SLG47910 and their corresponding SPI functions.

Table 7: Pin numbers and SPI Pin functions

QFN Pin No	Pin Name	SPI Mode
16	GPIO3	SPI_CLK
17	GPIO4	SPI_SS
18	GPIO5	SPI_SI (MISO)
19	GPIO6	SPI_SO (MOSI)

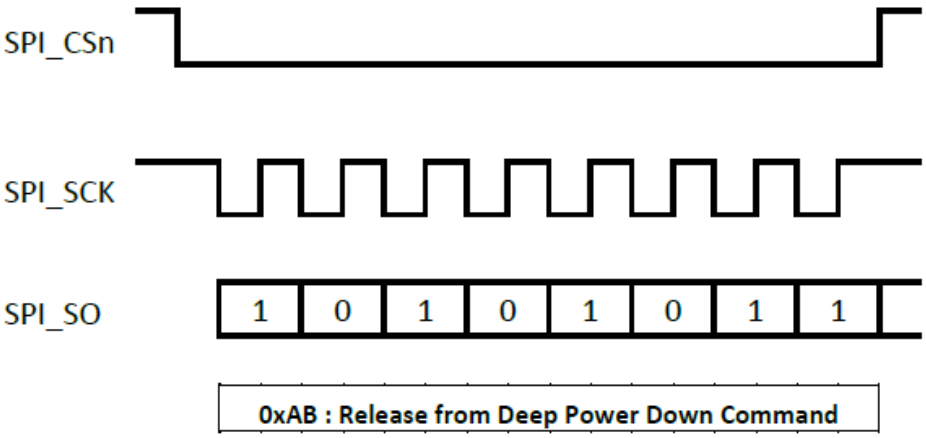


Figure 9: SPI Release from Deep Power Down Command

After the 300 µs buffer has passed, the SPI Master sends a Fast Read Command with a 24-Address on the SPI_SO pin and receives the data on the SPI_SI port as shown (see Figure 10). The first data byte (Data Byte 0) read from the SPI Flash is the lowest byte of a 32-bit data word.

After transferring the required number of configuration data bits, the SLG47910 ends the Fast Read command by de-asserting its SPI_SS select output. To conserve power, the SLG47910 then issues a final "Deep Power-down" command (0xB9).

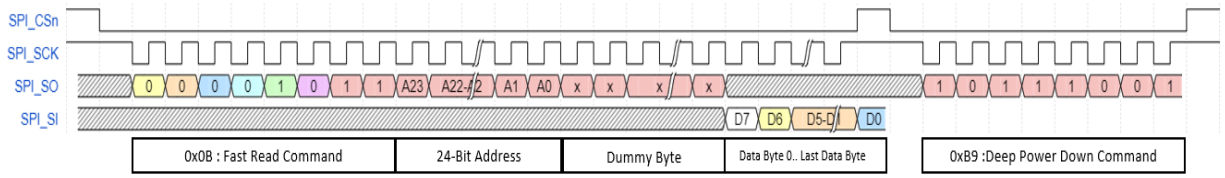


Figure 10: SPI Read Fast Command & Deep Power-Down Command

The SLG47910 device is configured using a single data pin (SPI_SI).

The procedure to enable SPI Configuration is given below:

1. Hold GPIO4 (SPI_SS) HIGH for a minimum delay of 1055 µs to enable SPI mode.
2. After entering SPI mode, internal logic will release GPIO4 (SPI_SS) from driving through the GPIO.
3. The internal Config-Circuit which acts as the master will control the SPI interface (GPIO3, GPIO4, GPIO5 & GPIO6) throughout the FPGA configuration.
4. The Config-Circuit sends a wake-up command (AB) to the SPI flash device first and then it sends a fast read command (QB). After the configuration is completed, it sends a sleep command (B9).
5. The Config-Circuit will generate an internal Config done = 1 signal.

8.1 Configuring the ForgeFPGA from External Flash Memory

The ForgeFPGA Socket Adapter Board has an onboard 4 Mbit (2M x 2) SPI serial flash memory (external). It is used for uploading a bitstream into the ForgeFPGA externally. The Go Configure Software Hub software allows the user to program and debug the external flash memory on ForgeFPGA Socket Adapter.

For more information on how to configure ForgeFPGA from External Flash Memory, refer Section 1.4 in the Socket Adaptor User Manual [5]. User can use the external flash memory functions in the Debugging Controls section of the software to program and read the external flash memory (see Figure 2).

9. MCU Programming (Slave Mode)

In this mode, SPI_SS should be held LOW initially. After POR, the CONFIG/SPI_MOSI (GPIO6) signal will go LOW signaling the host (MCU) that it can start sending data to the device. After the reset is de-asserted, the host should send 320-word SPI_SCK cycles (PREAMBLE) while holding SPI_SSn LOW and set SPI_SI data to 0. PREAMBLE is used to flush the SLG47910 FPGA Core before configuring the array. After the PREAMBLE, the MCU sends a sync word (32-bits) followed by 288-bits of data to configure other SoC registers. After the 288-bits are sent, the MCU will then send configuration bits to the FPGA Core. Once the configuration bitstream is sent, the host keeps SPI_SS LOW and continue to send SPI_SCKs (POSTAMBLE) until it sees the CONFIG signal go HIGH. Figure 11 shows the SPI slave timing.

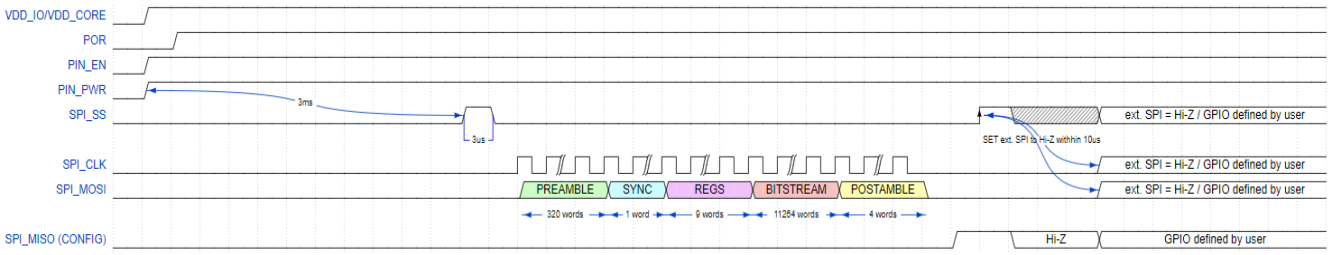


Figure 11: SPI MCU Mode Timing

The SLG47910 will then check this synchronization word to determine if the transfer is targeting this device. If the synchronization word does not match this device, then the configuration bitstream will be discarded. This synchronization word is checked on both the SPI slave and the SPI master. Slave SPI Clock Frequency is 16 MHz with MCU in slave mode. The MCU Mode corresponds to the Emulation mode in the software (see Figure 2). The Emulation mode allows the user to test the functionality of the design without having to program the part.

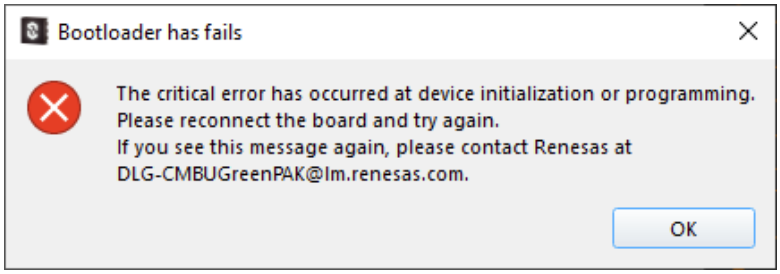
The following steps are used to program the SLG47910 in MCU mode.

1. Power On, set SPI_SS pin to "0"and PWR and EN pins to "1"
2. Wait 3 ms
3. Set SPI_SS pin to "1"
4. Wait 3 µs
5. Set SPI_SS pin to 0
6. Send PREAMBLE (320 words, MOSI = 0)
7. Send SYNC (1-word LSB 0x11FF22AA)
8. Send REGS (9 words)
9. Send BITSTREAM (11264 words)
10. Send POSTAMBLE (4 words, MOSI)
11. Set SPI_SS pin to "1" when CONFIG is HIGH (MISO line)
12. Change SPI pins into Hi-Z state within 10 µs after rising edge of SS

10. Troubleshooting

Issue:

When connecting the Evaluation Board to the software, it displays an error message as it is unable to initialize the board. You will see that the blue LED flashes briefly and then it disappears. This means there is an initialization error. The message displayed resembles this



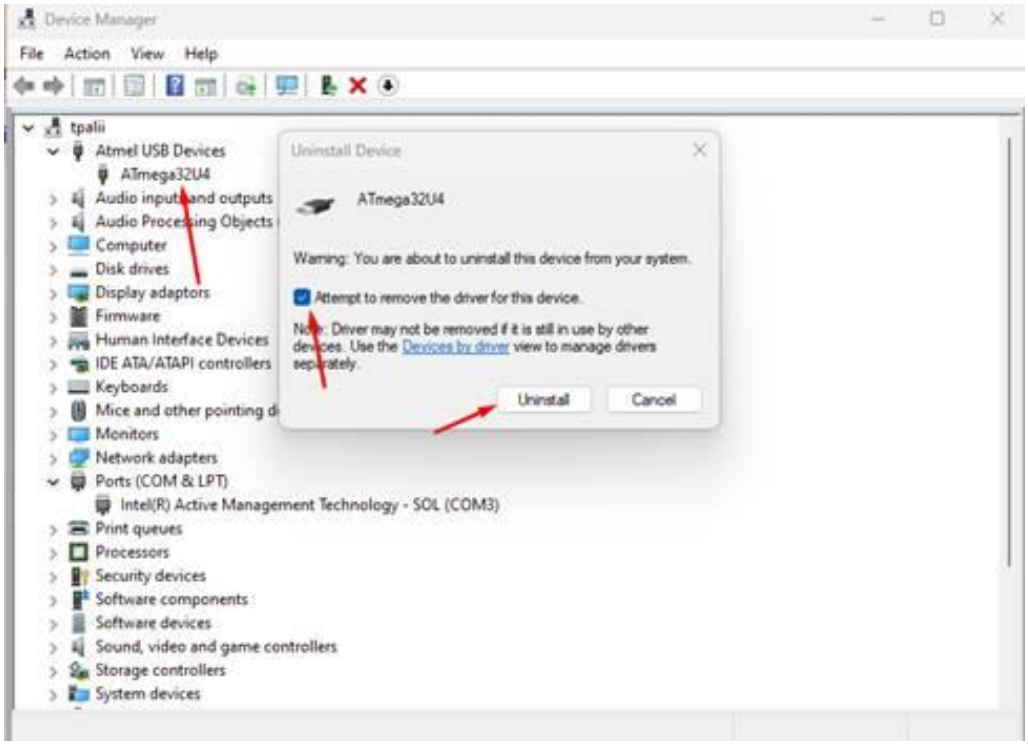
Solution:

This is due to a missing or incorrect device driver. We have found that installing the latest Atmel driver for different devices would cause a conflict.

Typically, during the installation, the software avoids forcefully replacing any existing drivers in the user's system. If a different driver version had been installed before, our devices might be inoperable.

To fix the issue, a user must uninstall the conflicting driver and rerun our driver installation package.

1. Open Go Configure software -> press Debug -> select ForgeFPGA Evaluation Board.
2. Hold the 'RST' button on the board pressed for 5 seconds.
3. Open the Device Manager, find the 'ATm32U4DFU' or 'ATmega32U4' device, and uninstall the driver. If multiple driver versions were installed previously, this action might need to be done several times. Make sure the driver is removed.



4. To install the drivers, a user must do one of the following:
    - Run 'ForceInstallation.bat' as Administrator from the driver installer. Usually, it is provided with the software at 'C:\Program Files\Renesas Electronics\Go Configure Software Hub\driver'.
    - Alternatively, a user can reinstall the software, and the driver will be installed normally, provided there are no more conflicts. Repairing the software should also do the job.

11. Conclusion

This configuration manual focuses on three configuration options: OTP, MCU and SPI Flash. The document describes how if the device is configured using OTP, then the other configuration options will be disabled. It also explains the design and the operation of the Development Board and the Evaluation Board to test and debug designs. If interested, please contact the ForgeFPGA Business Support Team.

12. Revision History

Revision	Date	Description
1.10	Jul-23-2024	Added troubleshooting steps for EVB
1.00	Jun-04-2024	Release Version

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.