ForgeFPGA

This document describes how to configure the ForgeFPGA core using three different configuration bitstream sources: External SPI Flash, Internal OTP, and MCU as a host for SLG47910/12 and SLG47920/21. It also details how to perform debugging using the hardware boards.

Contents

Figures

Tables

1. References

Download our free ForgeFPGA Designer software [1] and follow the steps in this user guide [7]. Users can reference [2] ,[8] and [11] for the datasheets. 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 ForgeFPGA 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 Deluxe Development Board User Manual, Renesas Electronics Corporation
[5] ForgeFPGA Socket Adapter User Manual, Renesas Electronics Corporation
[6] ForgeFPGA Evaluation Board User Manual, Renesas Electronics Corporation
[7] ForgeFPGA Workshop User Guide, Renesas Electronics Corporation
[8] SLG47920/21 Datasheet, Renesas Electronics Corporation
[9] GoConfigure Development Board User Manual, Renesas Electronics Corporation
[10] ForgeFPGA Socket Card User Manual, Renesas Electronics Corporation
[11] SLG47912 Datasheet, Renesas Electronics Corporation

2. 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 GoConfigure Software is used to generate bitstreams. The schematic in Figure 1 shows a block diagram of the Configuration Circuit, the external MCU Host, and SPI Flash interface. The four SLG47910 configuration pins are GPIO3 (SPI_SCK), GPIO4 (SPI_CS, Chip Select), GPIO5 (SPI_SI, serial input), and GPIO6 (SPI_SO, serial output). GPIO6 is also used as a Config Done signal in SLG47910.

On SLG47912/20/21 the four configuration pins are GPIO0 (SPI_SCK), GPIO1 (SPI_CS, Chip Select), GPIO2 (SPI_SI, serial input), and GPIO3 (SPI_SO, serial output). Table 1 shows which modes activate the SPI Controller (Master) and SPI Target (Slave) blocks during configuration.

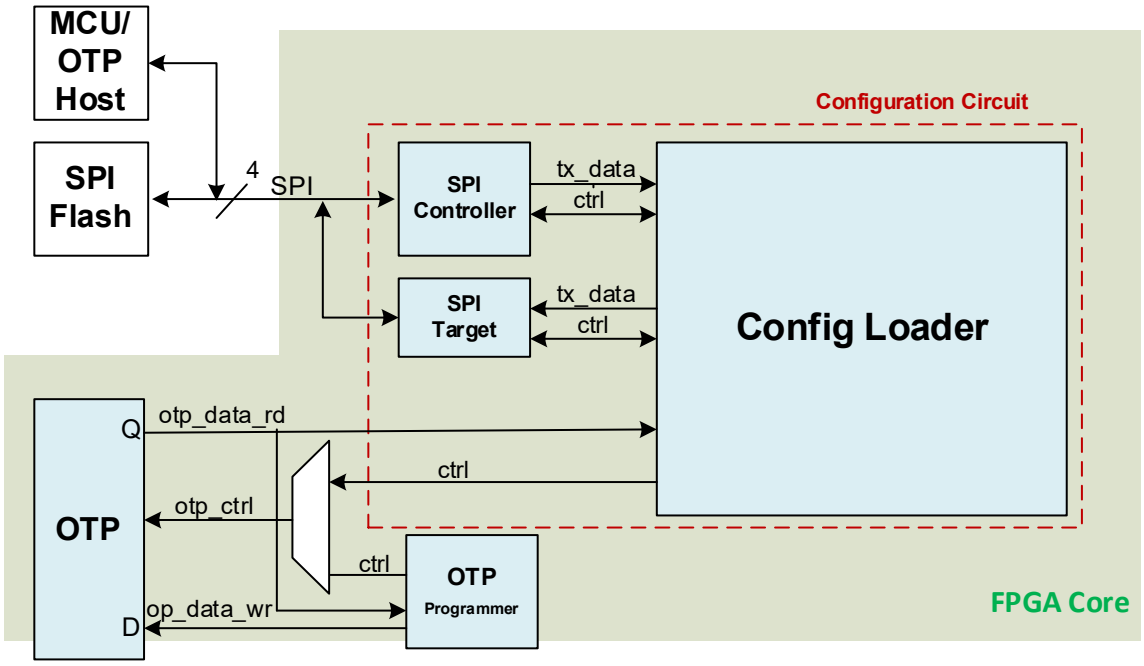

Figure 1. Programming and Configuration Interface

Table 1. Configuration Modes

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

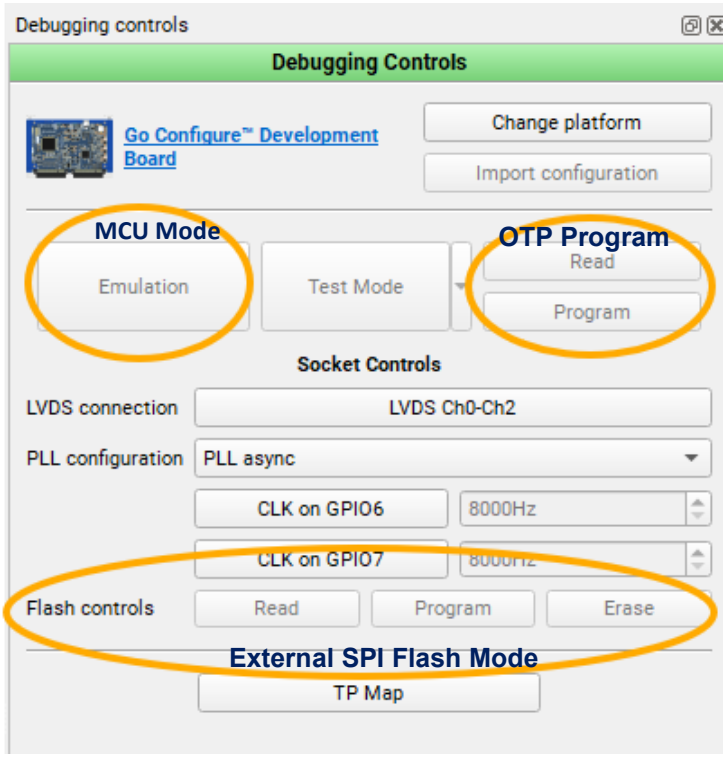

Figure 2. Configuration Modes in Software

3. General SPI Interface

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

SCK: Serial Clock (output from Controller). When the controller communicates with the target, the data on the MOSI or MISO pin will be synchronized with the Serial Clock. In the SPI protocol, the controller produces the clock, and the target will only receive the clock and has no control over the serial clock.

MOSI: Master-Out Slave-In (data output from controller). MOSI is a data pin. This pin is used to transmit data from the controller to the target device. Whenever the controller sends data, that data will be collected over the MOSI pin by the target.

MISO: Master-In Slave-Out (data output from target). MISO is a data pin. This pin is used to transmit data from the target to the controller. Whenever the target sends data, that data will be collected over the MISO pin by the controller.

CS: Chip-Select (often active low, data output from controller). Depending on the SPI and chip select setting, the CS pin is used to select an individual target device for communication. When there is one controller and only one target device, then the CS pin is not required. This target select pin is only necessary when the controller is communicating with multiple different targets. So, the controller can select which target device to which the controller wants to communicate. There is a dedicated CS pin for each target device connected to the controller.

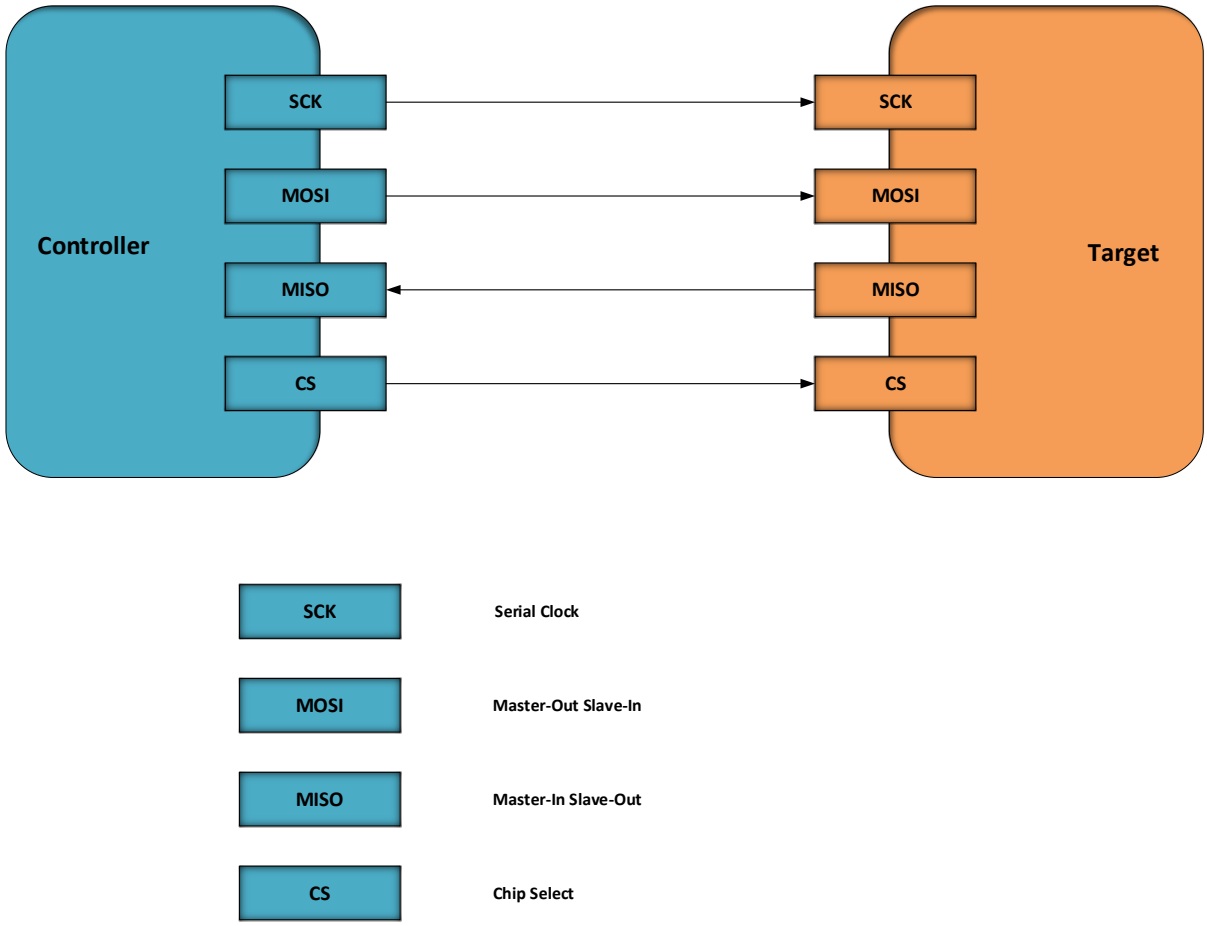

Figure 3. SPI Interface

3.1 SPI Modes with Clock Polarity and Clock Phase

In SPI, the controller 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 CS 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 3 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 the orange dotted lines corresponding to the rising or falling edge depending on the selected SPI Mode and the blue dotted lines corresponding to the shifting edge of the data.

Table 2 depicts SPI Mode 0 with CPOL = 0 and CPHA = 0 with the clk idle state = 0. The data is sampled on the rising edge and shifted on the falling edge according to Table 2.

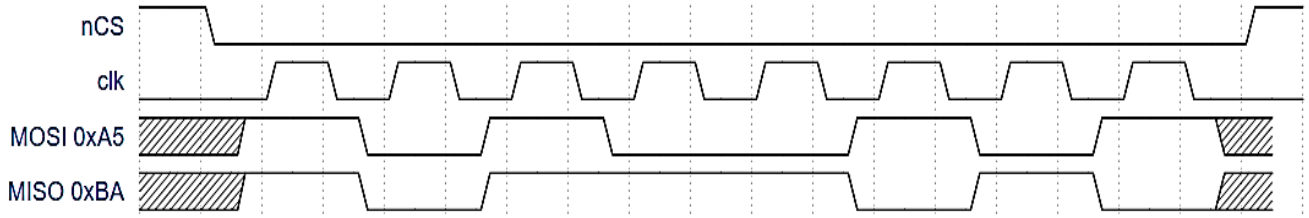

Figure 4. SPI Mode 0 with CPOL = 0 and CPHA = 0

4. ForgeFPGA Development Platform

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.

1. **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*.
2. **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*.

4.1 ForgeFPGA Deluxe Development Platform

Under the "Debug" button on the toolbar, select the ForgeFPGA Deluxe Development Platform as the platform (see Figure 5) to begin Configuration.

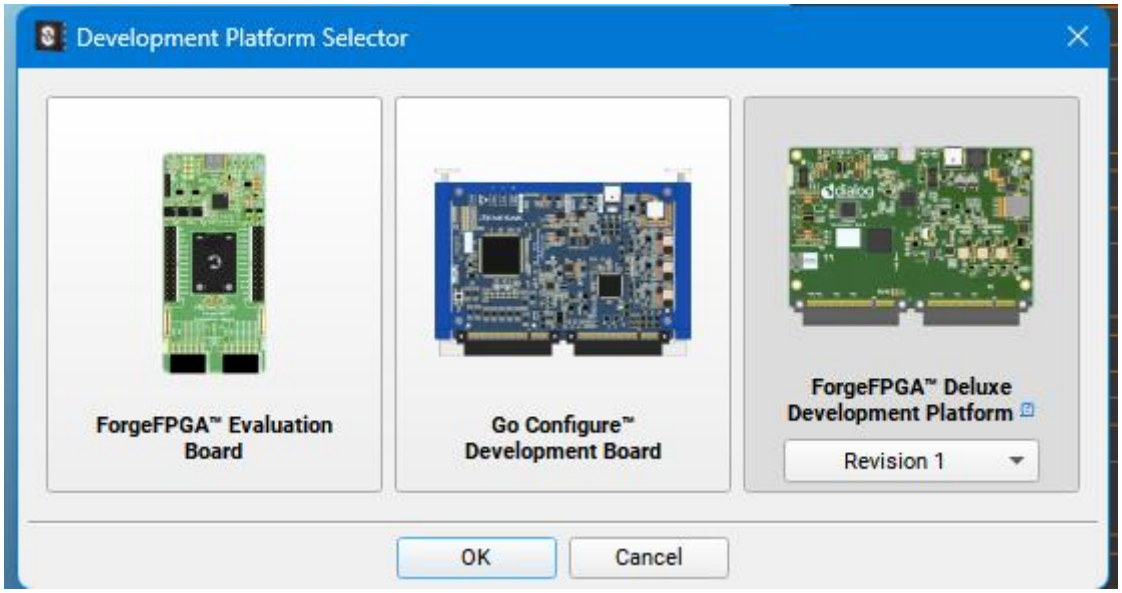

Figure 5. Development Platform Selector

The FPGA Deluxe Development Board is a multi-functional tool that makes it possible to develop FPGA designs easily by providing on-board power sources, digital and analog signal generation, and logic analysis capabilities.

The FPGA Deluxe Development Board can connect to additional external boards called *Socket Adapters*. 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 Deluxe Development Board. To implement this, the FPGA Deluxe 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 open-source software, the FPGA Deluxe Development Board can be configured to work as any one of several traditional instruments including the following:

- Logic Analyzer (LA) up to 800 MS/s
- Digital pattern generator (PG) up to 200 MS/s
- 8-channel Analog Arbitrary Waveform Generator (AWG)
- Configurable voltage sources
- Waveform power supply
- 80 Test Points (TP) with LA and PG
- Real-time Test Point Control
- Three programmable power supplies (+0.6 V…+3.465 V) and a maximum available output current of 2 A. The same voltage is supplied to the GPIO, for keeping the logic level compatibility with the circuit under test.

Using the Deluxe Development Platform, users can configure the device via MCU Mode, OTP Mode, or through external SPI Mode.

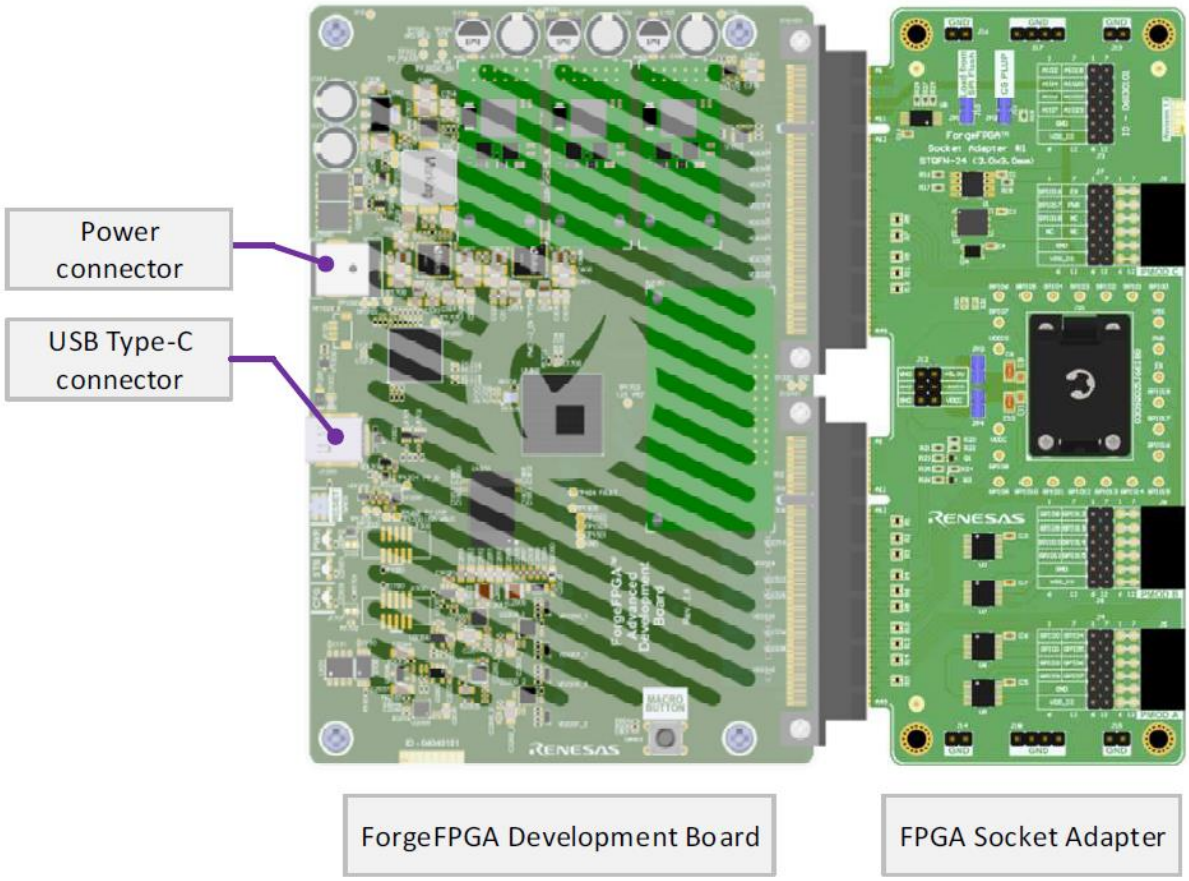

Figure 6. Connection between ForgeFPGA Deluxe Development Board and Socket Adapter

For more information on configuration with the Deluxe Development Boards, see the References section.

4.2 ForgeFPGA Evaluation Board

To use the Evaluation Board's debug controls, select the, ForgeFPGA Evaluation Board under the "Debug" button on the toolbar (see Figure 7).

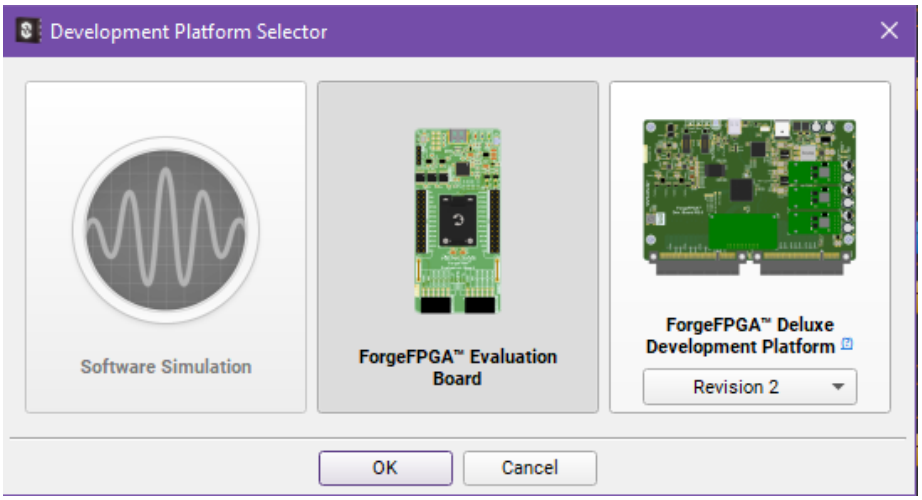

Figure 7. ForgeFPGA Evaluation Board Selection

Driven by the Go Configure Software Hub, the ForgeFPGA Evaluation Board is configured to work with basic FPGA designs and comes with several features:

- Configurable V_{DDC} and V_{DDIO} power sources
- Zero-Force 24-pin Socket Connector
- Emulation and programming options
- PMOD connectors
- UART Terminal Interface

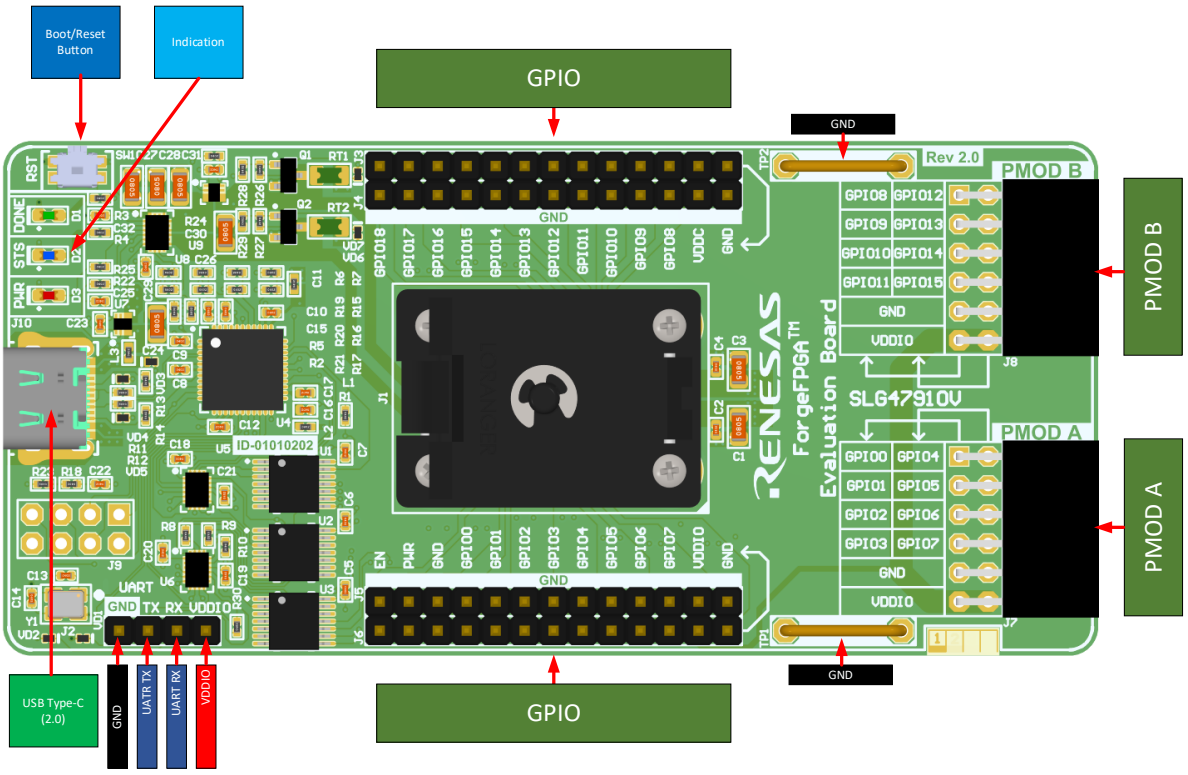

Figure 8. ForgeFPGA Evaluation Board R2.0 Overview

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

The ForgeFPGA Evaluation Board R2.0 (Figure 8) has following connectors:

- USB 2.0 Type-C connector
- 2 PMOD connectors
- 2 GPIO access connectors

It includes three LEDs for Power, Status, and Done, and has one Reset Button.

For more information on configuration using the EVB, see [6] ForgeFPGA Evaluation Board User Manual, Renesas Electronics Corporation in the References section.

4.3 ForgeFPGA GoConfigure Development Board

The GoConfigure Development Board (see Figure 9) is an updated platform from Renesas, designed for development using products, such as GreenPAK family mixed signal ICs and ForgeFPGA programmable arrays. The Board works in combination with the Go Configure Software Hub and provides rich functionality, such as:

- Flexible power management with software-configurable protection (OVP, OCP)
- Three Programmable and three Fixed power sources
- Digital generators
- Analog generators (AWG)
- Onboard sequencer for Reading/Programming/Emulating Renesas products
- Up to 80 multi-functional configurable I/O lines (availability depending on connected target board)

Working with target devices is done through interaction connectors and corresponding mating boards such as ForgeFPGA Socket Cards. All onboard signals, communication interfaces and power lines are exposed to connectors.

Development of ForgeFPGA products is possible through specialized Socket Cards. Cards are built for specific parts and packages and provide minimum required set of external components for proper IC configuration and work, as well as some extra features like power selector, clocks, I/Os exposed to PMOD connectors and others.

For more information on the GoConfigure Development Board and for specific Socket Cards see the References section.

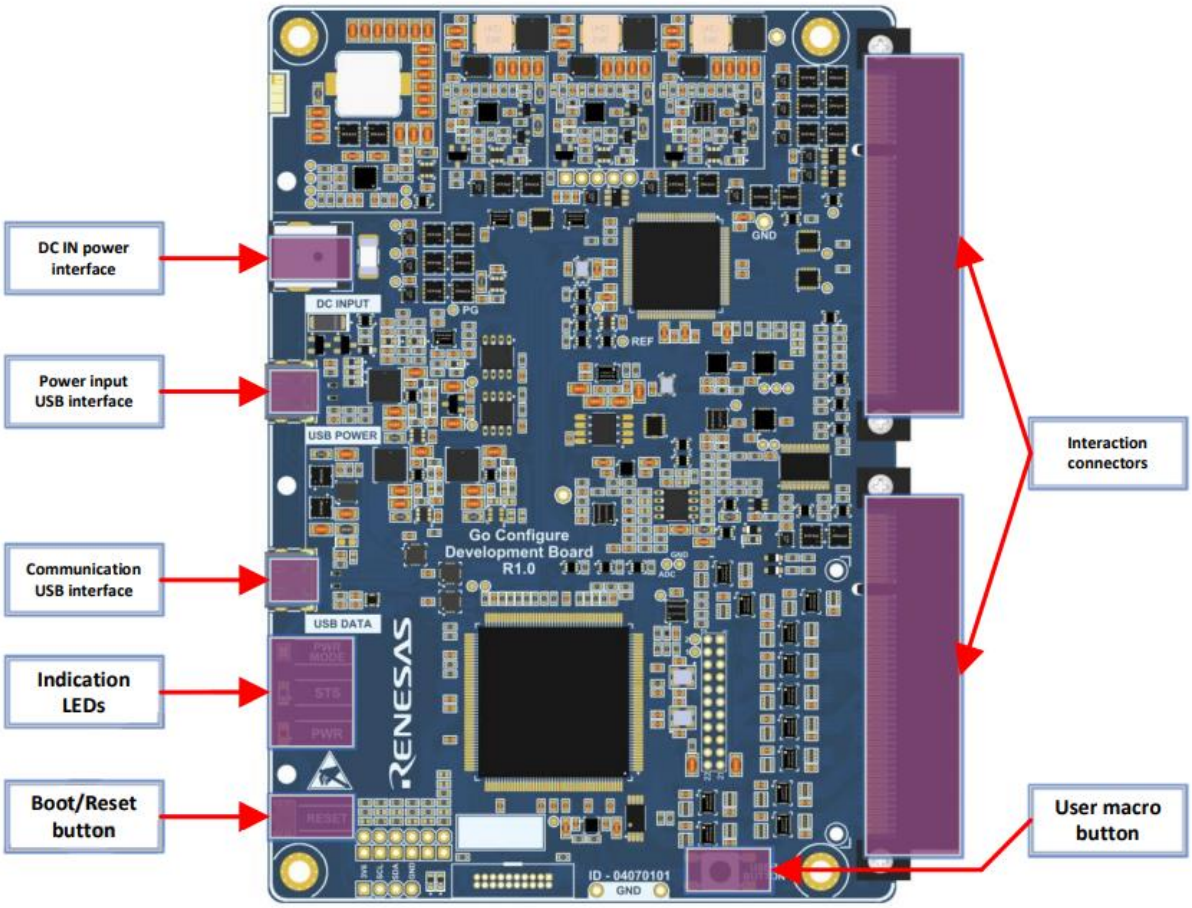

Figure 9. GoConfigure Development Board

5. SLG47910 OTP Write

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

Writing to the OTP block is achieved through the SPI interface pins. If the OTP block of the SLG47910 has already been programmed, upon power up, the contents of the NVM will be loaded into the device's internal configuration RAM. If force load from OTP is on, or the SLG47910 fails six times to load from the SPI Flash or MCU, it will default to OTP mode.

5.1 Boot Config Project Settings

There are project settings that need to be configured before writing to the OTP (see Figure 10).

5.1.1 Force Load from OTP

Disable/Enable is used to select "Force Load from OTP". if *Enable* is selected before program OTP, ForgeFPGA will only boot from OTP NVM and there will be no way to load from external Flash or MCU after writing the OTP. If D*isable* is selected before programming the OTP, then loading from an External Flash or MCU after writing the OTP is an option.

5.1.2 NVM Lock Status

There are four selectable Lock Status options, "Unlocked/ Read lock/Write lock/Lock read and write". Be careful to select this before writing the NVM.

If *Unlocked* is selected before writing to NVM, users can only Read after writing to the NVM, if *Read lock* is selected before writing to the NVM, users cannot read out NVM data after writing it. If *Write lock* is selected before writing to the NVM, users can not write again (for MTP only), if *Lock read and write* is selected before writing to the NVM, users will lose any access to NVM after writing it. Users cannot read from the NVM and will be shown notification as well about this. This is also known as a measure to ensure user data security.

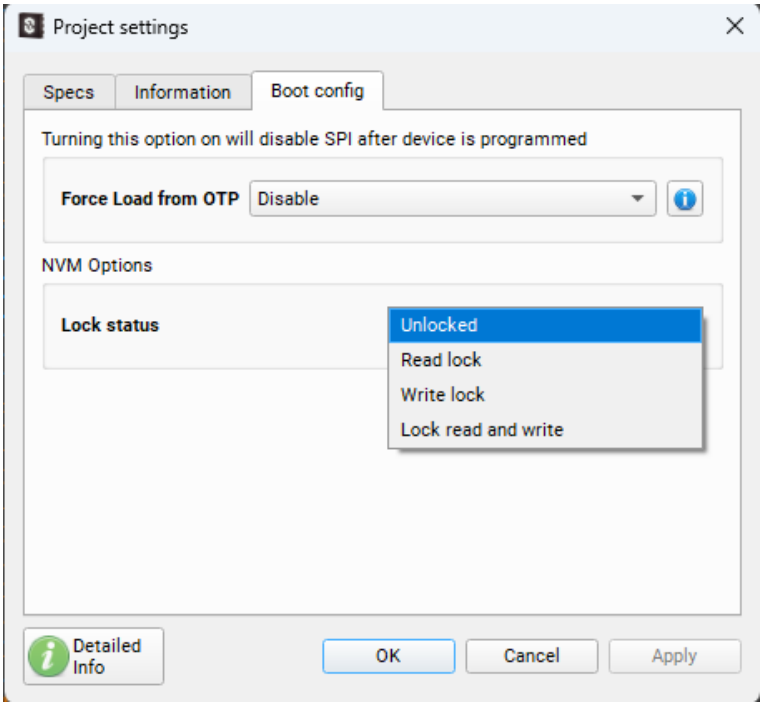

Figure 10. Boot Config Project Settings

5.2 SLG47910 Writing to the OTP Block

Writing to the OTP is done using the SPI Target interface. The OTP write operation starts with ramping up the V_{DDC} voltage to 1.1 V and the V_{DDIO} voltage to 2.75 V which will set the NVM to operate in “write program” mode. There is an OTP_programmer (see Figure 1) that sequences the internal signals to enable the OTP write operation.

Once the overdrive and program mode inputs are active, the OTP_programmer decodes the spi_target data, and the data address. Then control information is sent to the OTP_programmer 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 Byte 8, 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 are disabled. At this point, the SLG47910 will only load program data from the OTP.

To write to the OTP:

Wait for POR and PLL lock time (1300 µs), then send the Signature Bytes through SPI_SO (GPIO5) by keeping SPI_CS (GPIO4) LOW.

After 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 an additional delay of 18 µs between the 1st and 2nd packet and a delay of 10.11 µs for any subsequent packets.

The last write packet is indicated by Byte 8, Bit 6 = 1.

After writing to the OTP, bring PWR = LOW (which resets the device), then chk_otp_en = 1.

Table 3. OTP Write Packet format

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

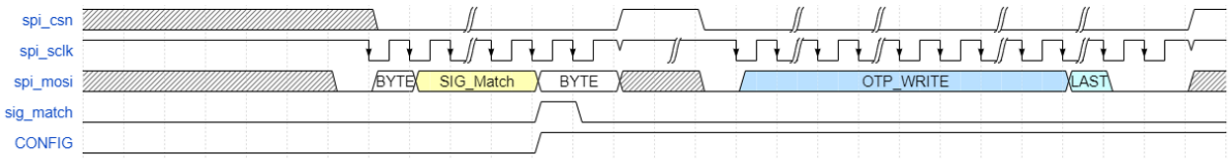

Figure 11. OTP Write Waveforms

Table 4. Write/Read & SP/AP

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

6. SLG47912/20/21 OTP Write

For the SLG47912/20/21, reading and writing to the OTP block is achieved through the SPI interface pins. If the OTP block has already been programmed, upon POR, the contents of the OTP will be loaded into the device's internal configuration RAM and configuration registers.

The OTP can be programmed, and its contents can be read via the Read and Program buttons respectively, in the software under "Debugging Controls" (Figure 2).

6.1 Boot Config Project Settings

There are project settings that need to be configured before writing to the OTP, see Figure 12.

6.1.1 Boot Priority

There are four selections available for Boot priority, "*SPI_CS line state*" determines whether the FPGA will boot according to the SPI_CS pin level when the chip configuration process starts. A HIGH level on the SPI_CS pin means the FPGA will boot from the external SPI flash, while a LOW level on the SPI_CS pin means that the FPGA will boot from the MCU. If the FPGA needs to boot from OTP, the otp_en register should be HIGH when the chip configuration process starts.

"*Force boot from MCU*" means the FPGA will only boot from MCU after writing to the OTP.

"*Force boot from OTP*" means the FPGA will only boot from OTP after writing to the OTP.

"*Force SPI boot*" means the FPGA will only boot from external SPI flash after writing to the OTP.

NVM Options are the same as SLG47910V, see section 5.1.2 for Lock Status options.

There is an option to choose what will be a part of the OTP configuration file. The two selections are either the "*Full FPGA Configuration*" information or to include the "*Boot Configuration information along with OTP user data".*

6.1.2 SPI Boot Control

SPI mode offers two options, *Single mode* and *Dual mode*. *Single mode* means there is only one bitstream loaded in the external Flash. Dual mode means there are two bitstreams loaded in the external Flash: Golden image and Primary image. Golden image is a stable version of the design; it is stored at a safe address, typically the starting address of the flash (0x000000). The primary image is the main, updated application design; it's stored at a different, user-defined address (e.g., 0x100000).

By configuring these settings, the FPGA has a fallback function:

1. The FPGA is set up to first try to boot from the primary image address.
2. If this boot is successful, the system runs normally.
3. If the boot fails (e.g., due to data corruption or glitch during the initial load), the FPGA triggers its fallback mechanism.

4. It automatically resets and falls back and loads the “Golden Image” from the safe address (0x000000).

6.1.3 QSPI Boot Control Logic

QSPI boot control logic offers two options: *Disable* and *Enable*

If *Disabled*, the FPGA will boot from the first address of the External Flash.

If *Enabled*, the QSPI boot control code selection is enabled. The user can change the default boot address if needed, even if boot control logic is disabled.

In the BIT (bitstream) row, when the users click the DIS (disable) button for any of the numbered columns (#3 /#2 /#1 /#0), it *Enables* the corresponding bitstream, changing the displayed text from DIS (disabled) to EN (enabled). This selection enables the address Control code bit and lets the user choose a source for the bit control bit (either GPIO or IOB). When all four BIT’s (corresponding to GPIO12, GPIO13, GPIO14 and GPIO15) are in EN mode, users can address up to 16 different bitstreams from external Flash. The QSPI boot addresses need to be defined when enabling those boot control selection bits (see Figure 12). Each control code is assigned to the address of the Flash memory from which the bitstream reading should start. Different control codes can point to the same Flash memory address.

To change the bitstream, the user should reset the chip via the nRST pin and then choose the bitstream using the selected control code source.

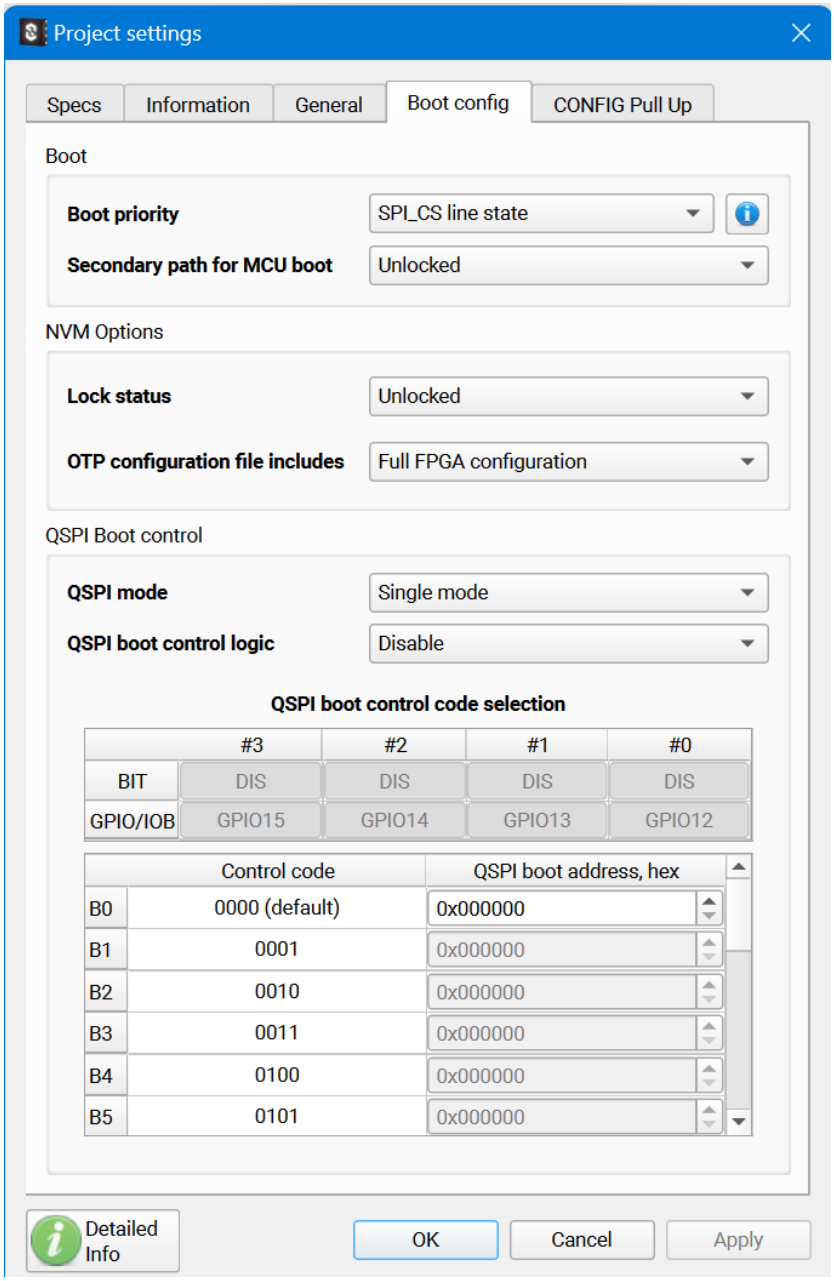

Figure 12. Boot Config Settings in the Project Settings

7. SPI Programming (Controller Mode)

In Controller mode, communication starts by driving SPI_CS LOW and then sending a “Release from Deep Power-down” command to the SPI Flash, 0xAB. An example waveform is given in Figure 13.

This initial command wakes up the SPI Flash if it is in Deep Power-down Mode. The SPI Controller transmits data on the SPI_SO output, on the falling edge of the SPI_SCK output. No data is sent on SPI_SI during this time. After sending the last command bit, SPI_CS is de-asserted High, completing the command. A buffer of 10 µs (minimum) is given before sending the next SPI Flash command.

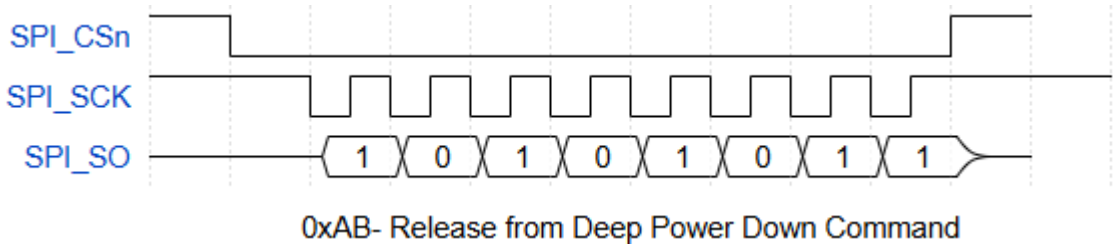

Figure 13. SPI Release from Deep Power Down Command

After 10 µs has passed, the SPI Controller sends a Fast Read Command with a 24-bit Address on the SPI_SO pin and receives the data on the SPI_SI port as shown in Figure 14. The first data byte (Data Byte 0) read from the QSPI flash is the lowest byte of a 32-bit data word.

After transferring the required number of configuration data bits, the wrapper ends the Fast Read command by de-asserting SPI_CS. To conserve power, the wrapper then issues a final Deep Power-down command, 0xB9.

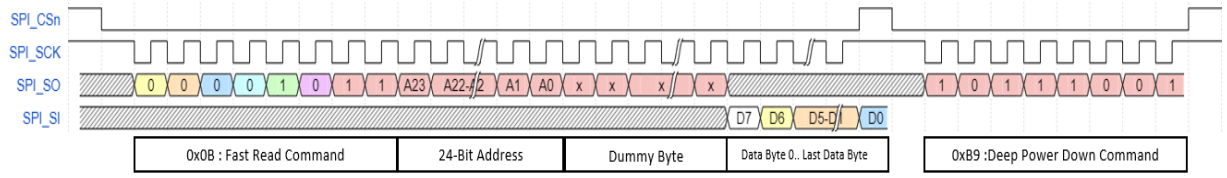

Figure 14. SPI Read Fast Command & Deep Power-Down Command.

7.1 Controller Mode SPI Programming for SLG47910

Table 5 lists the pins of the SLG47910 and their corresponding SPI functions.

Table 5. Pin numbers and SPI Pin functions for SLG47910

SLG47910V	SLG47910C	Pin Name	SPI Mode
16	B3	GPIO3	SPI_SCK
17	C4	GPIO4	SPI_CS
18	A5	GPIO5	SPI_SI (MISO)
19	B4	GPIO6	SPI_SO (MOSI)

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

To enable SPI Configuration:

Hold GPIO4 (SPI_CS) HIGH for a minimum delay of 1055 µs to enable SPI mode.

After entering SPI mode, internal logic will release GPIO4 (SPI_CS) from driving through the GPIO.

The internal Config-Circuit which acts as the controller will control the SPI interface (GPIO3, GPIO4, GPIO5, and GPIO6) throughout the FPGA's configuration.

The Config-Circuit sends a wake-up command (0xAB) to the SPI flash device first and then it sends a fast read command (0x0B).

The Config-Circuit will generate an internal Config signal, Done = 1.

7.2 Controller Mode SPI Programming for SLG47920 and SLG47912/21

Table 6 lists the pins of the SLG47920 and SLG47912/21 and their corresponding SPI functions.

Table 6. Pin numbers and SPI Pin functions for SLG47920 and SLG47912/21

SLG47920V	SLG47912/21V	Pin Name	SPI Mode
1	2	GPIO0	SPI_CS
2	3	GPIO1	SPI_SCK
3	4	GPIO2	SPI_SI (MISO)
4	5	GPIO3	SPI_SO (MOSI)

The SLG47912/20/21 device is configured using a single data pin (SPI_SI).

The enable SPI Configuration:

Hold GPIO0 (SPI_CS) HIGH for a minimum delay of 1055 µs to enable SPI mode.

After entering SPI mode, internal logic will release GPIO0 (SPI_CS) from driving through the GPIO.

The internal Config-Circuit which acts as the controller will control the SPI interface (GPIO0, GPIO1, GPIO2, and GPIO3) throughout the FPGA's configuration.

The Config-Circuit sends a wake-up command (0xAB) to the SPI flash device first and then it sends a fast read command (0x0B). After configuration is completed, it sends a sleep command (0xB9).

The Config-Circuit will generate an internal Config signal, Done = 1.

7.3 Configuring the ForgeFPGA from External Flash Memory

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

The Socket Adapter Board Onboard Flash is an Adesto AT25FF041A Flash memory, there are lots of Flashes supported by ForgeFPGA, below list is tested Flash part numbers.

Table 7: Compatible Flash Parts

Flash Brand	Flash Part Number
Winbond	W25X40
Winbond	W25Q64
Winbond	W25Q128
Atmel	AT25DF041A
MXIC	MX25L25645G
GigeDevice	GD25Q16CSIG
Adesto	25SF641
Adesto	25DF081A
Adesto	25EU0021A
Renesas	25EU0081A
Renesas	25EU0161A

7.3.1 Programming the External Flash Memory

There are two ways to program the external flash memory, using either the Go Configure Software Hub or using a third-party programmer.

Method 1: Using the Go Configure Software Hub

After generating a bitstream for a design, the design can be programmed to the external flash memory by pressing the “Program” button in the External Flash control section of the Debugging control panel in the main window (Figure 15). A notification will be given once the bitstream has been successfully uploaded to the external flash memory.

Figure 15. Debugging Controls Panel: Program & Read

The uploaded bitstream data can be read back by clicking the “Read” button (Figure 16).

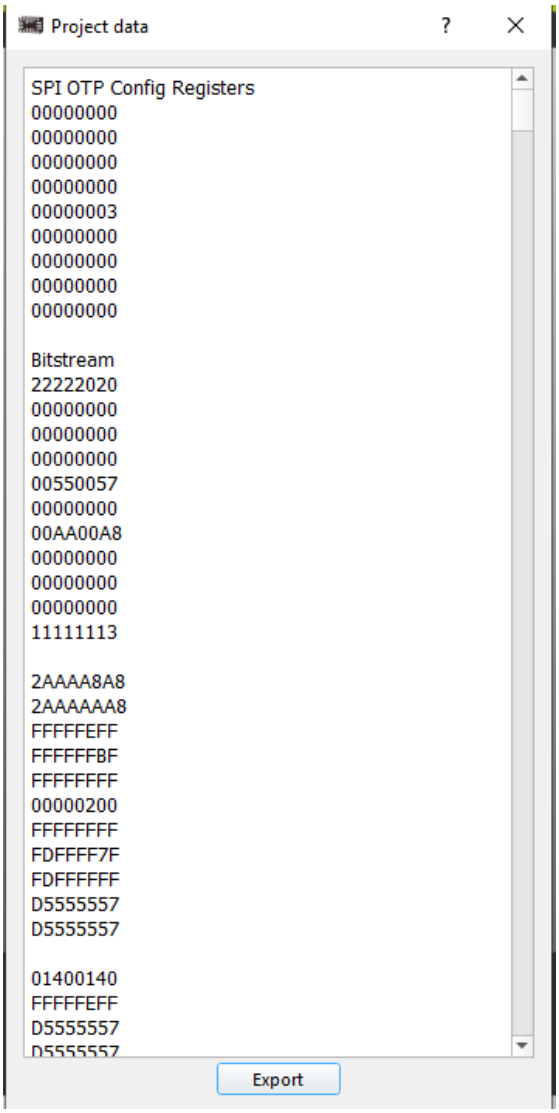

Figure 16. Reading from External Flash

Method 2: Using a Third-party programmer

The ForgeFPGA Socket Adapter Boards utilize an Adesto AT25FF041A Flash memory. Therefore, any third-party programmer compatible with the AT25FF041A can also be used to program the on-board external flash memory. Figure 17 illustrates an example of a compatible Flash programmer, provided for reference only.

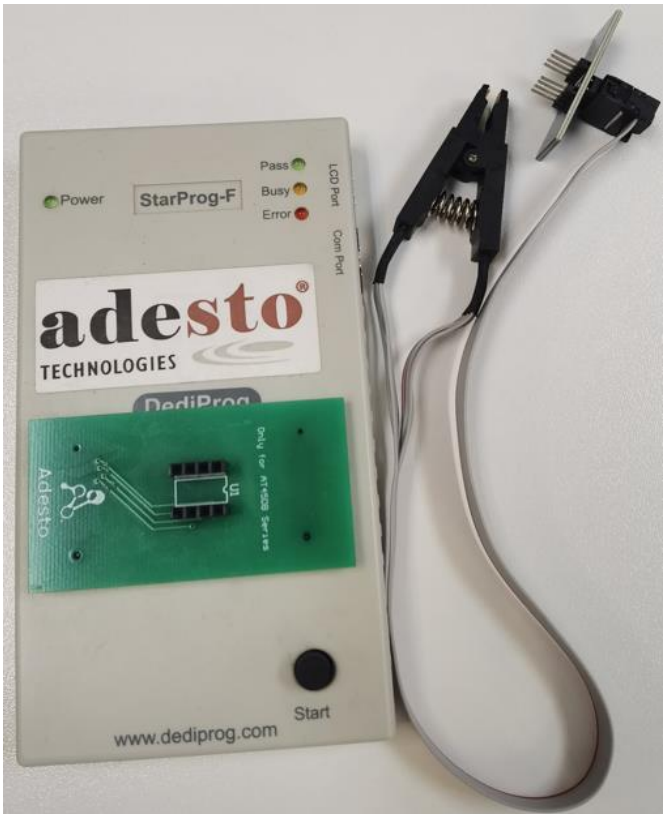

Figure 17. Adesto AT25FF041A Compatible Flash Programmer

To program the external flash:

Ensure that the ForgeFPGA Development Board is powered off. No external power is required during programming.

Disconnect the ForgeFPGA Socket Adapter from the development board.

Remove the jumper labeled “Onboard Flash connect” on the adapter board (see Figure 18).

Note: This method can be used with both ForgeFPGA Socket Adapter board #1 and #3. If working with ForgeFPGA Socket Adapter board #1, there is no need to remove any jumpers, and you can proceed to Step 4.

Connect the Flash Programmer

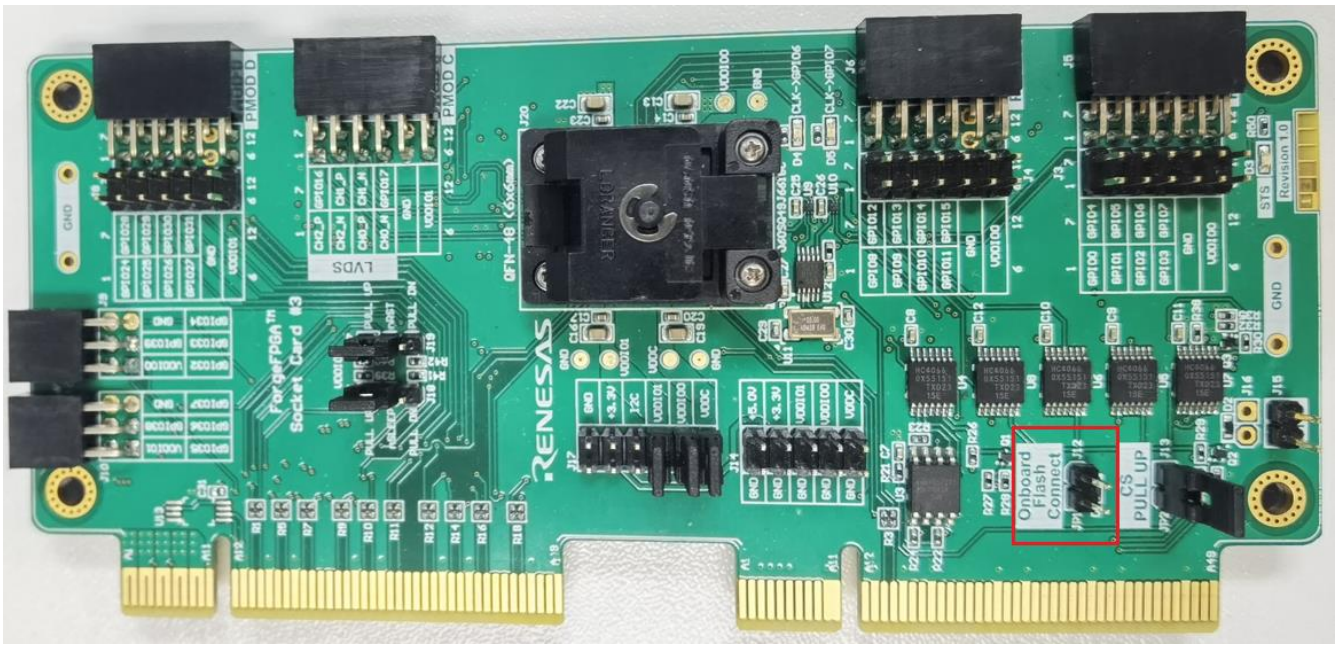

Figure 18. Disconnect Adapter from Development Board and Remove the Jumper

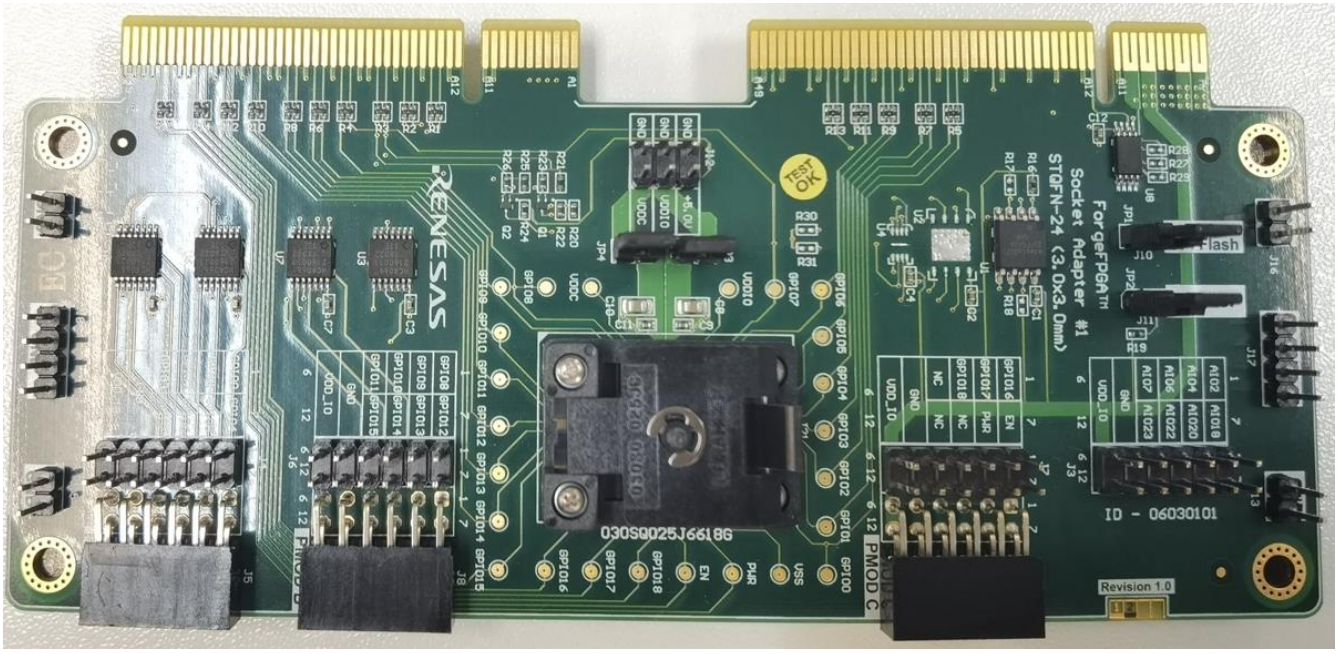

Figure 19. Disconnect Adapter #1 from Development Board

Hardware Connection:

1. Use a clamp to securely connect the Flash programmer to the onboard Flash memory. Ensure all contacts are properly aligned, and the connection is stable (Figure 20).
2. Note that the red wire in this clamp indicates Flash chip Pin 1, Figure 20 shows the connection with Socket Adapter Card #3, and the connection for Socket Adapter Card #1 is identical.

Software Setup:

1. Connect the Flash programmer to a computer via USB.
2. Verify that the computer has the dedicated programming application “DediWare General” installed and running.

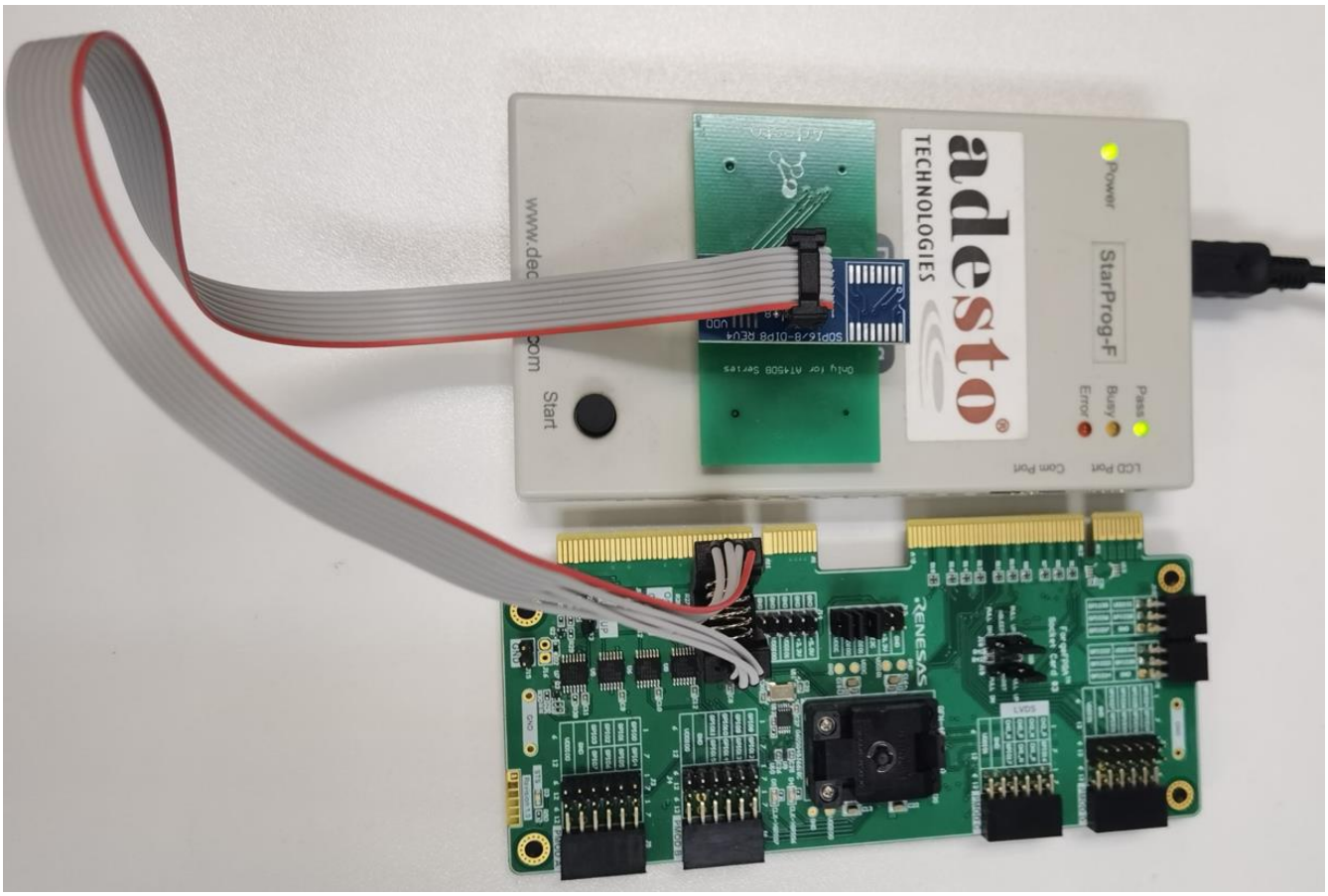

Figure 20. Connect the Flash Programmer to the Onboard Flash

Configure the Programming Software (Figure 21)

- Launch the Application:
  a. Double-click "DediWare General" to start the programming software.
- Select Engineering Mode:
  a. Click the "Select" button under Engineering Mode.
- Choose the Flash Memory:
  a. In the "Select Chip" window:
    - Select "Adesto Tech" from the manufacturer list.
    - Enter "AT25FF041A" in the search bar.
    - Choose "AT25FF041A [SOP8 208mil]-Adesto Tech" from the results.
  b. Click "OK" to confirm.

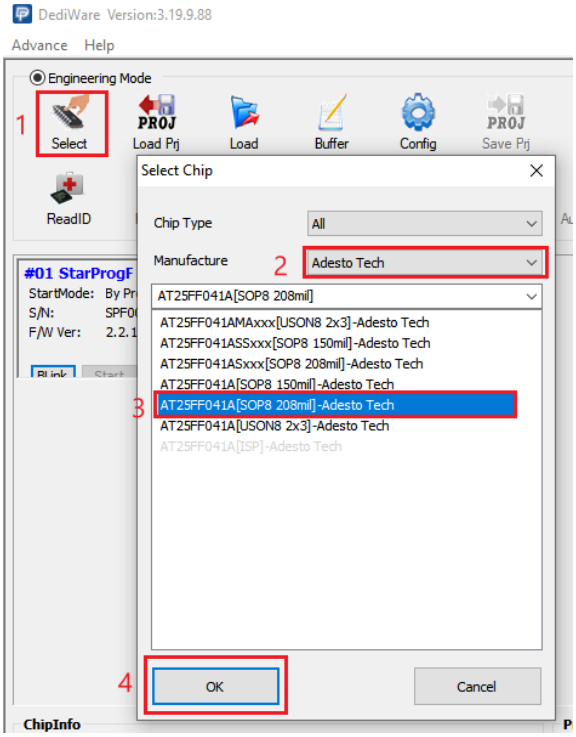

Figure 21. Configuring the Programming Software (Step 5)

Load the Bitstream File (Figure 22)

- Open File Selection dialog:
  a. Click the "Load" button under Engineering Mode.
  b. A "Select File" window will appear.
- Locate the Bitstream File:
- Click the "View" button next to FilePath.
  a. Navigate to your ForgeFPGA project directory: <ForgeFPGA_Project>/ffpga/build/bitstream
  b. Select the file "FPGA_bitstream_Flash_MEM.bin".
- Confirm Selection:
  a. Click "OK" to load the file

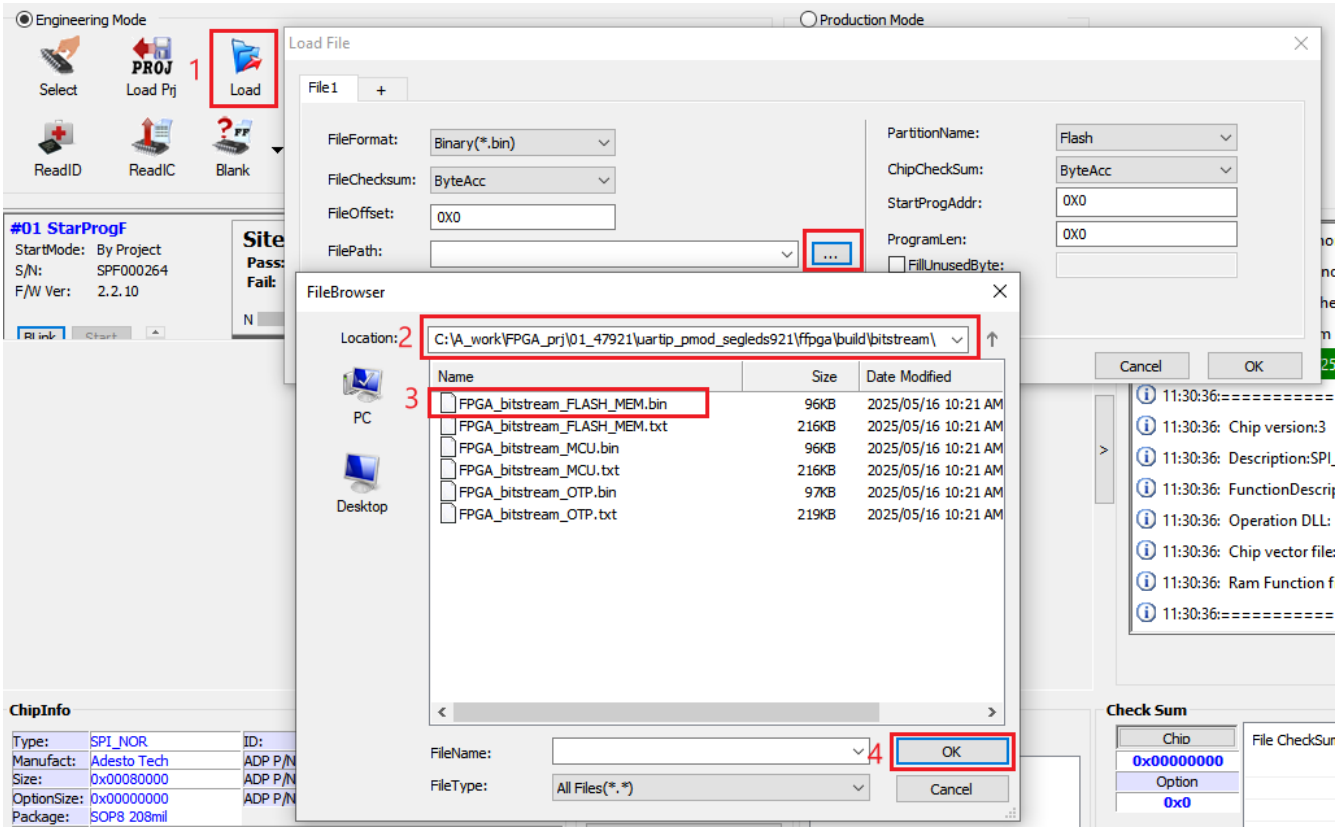

Figure 22. Loading the Bitstream (.bin) File (Step 6)

Program the Flash Memory (Figure 23)

- Initiate the Programming Process:

a. Click the "Program" button located under Engineering Mode.

- Select the Programming Target:

a. In the pop-up window, click "Flash" to begin programming.

- Monitor Progress:

a. Wait for the progress bar to complete (100%).

b. Do not interrupt the process until the program is completed.

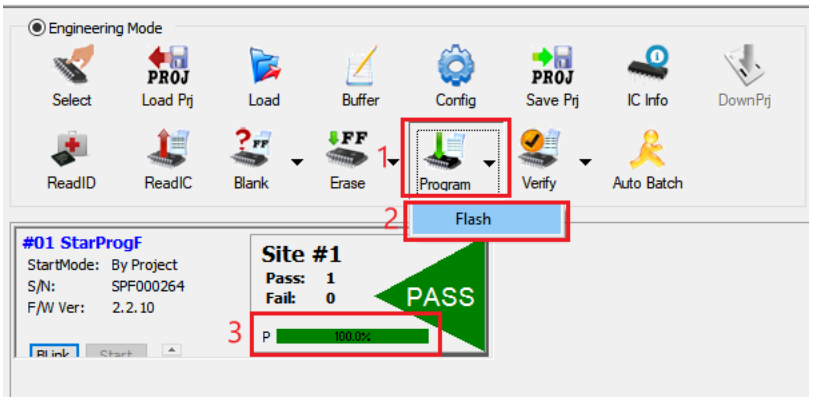

Figure 23. Programming the Flash Memory (Step 7)

7.3.2 Uploading into ForgeFPGA

Once the bitstream has been programmed into the external flash memory, it can be uploaded to the FPGA chipset. To do this, the jumper “FPGA<>Flash” (marked with red circle in Figure 24) needs to be disconnected in order to connect to the onboard external flash memory (for Socket Adapter Card #1 revision 1.0). There is no need to remove jumpers for Socket Adapter Card #1 revision 1.1 or Socket Adapter Card #3.

Method 1:

The ForgeFPGA Socket Adapter first needs to be disconnected from the ForgeFPGA Development Board. To power the Socket Adapter, use an external power source and connect it to the connection headers (purple circle in Figure 24) V_{DDIO} and V_{DDC}. Check the pin configuration of these headers specified on the back side of the Socket Adapter Board.

Note: Socket Adapter Card #3 has V_{DDIO0} and V_{DDIO1}, provide power supply to both, voltage should be 1.8 V ~ 3.3 V.

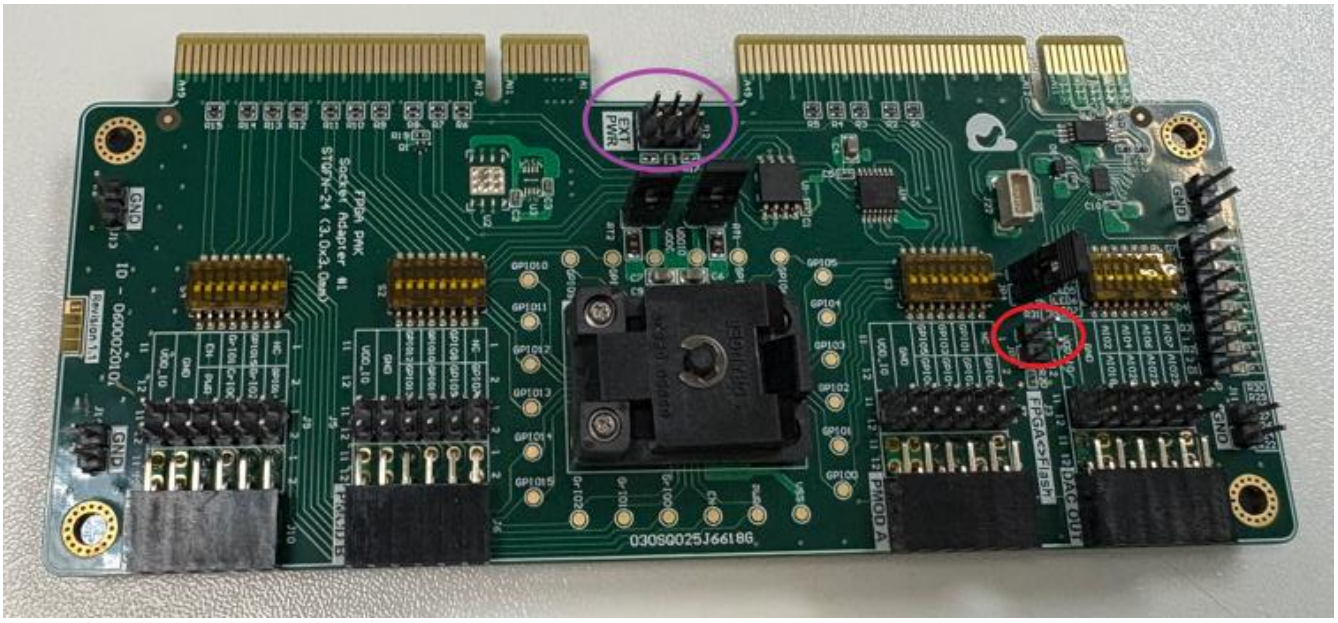

Figure 24. Jumpers and External Power Connection Points on the Board

Method 2:

From the Debugging Controls panel in the software, choose the “Test Mode” drop down menu. Next, choose the Test Mode (*) option to upload the bitstream from the external flash memory to the ForgeFPGA platform (Figure 25).

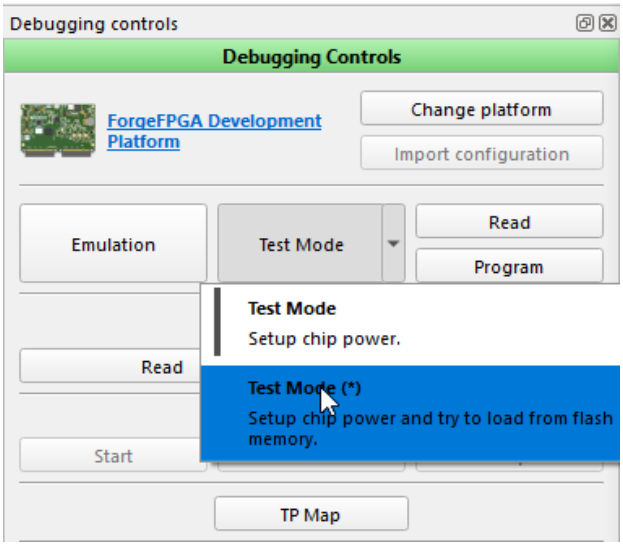

Figure 25. Choosing the Test Mode (*) option (Method 2)

8. MCU Programming (Target Mode)

MCU Mode corresponds to the *Emulation mode* in the ForgeFPGA software (see Figure 2). Emulation mode allows the user to test the functionality of the design without needing to program the part OTP.

8.1 MCU Programming in the SLG47910

In this mode, SPI_CS should be held LOW initially. After POR, the CONFIG/SPI_SO signal will go LOW, signaling to the host (MCU) that it can start sending data to the device. After the reset is de-asserted, the host should send FPGA_bitstream_MCU.bin while holding SPI_CS LOW and set SPI_SI data to 0. See Table 5 for list of GPIOs used for configuration.

Program the SLG47910 in MCU mode by completing the following steps:

Power On, set SPI_CS pin to "0" and set nRST (PWR) and nSLEEP (EN) pins to "1"

Wait 3 ms

Set SPI_CS pin to "1"

Wait 3 µs

Set SPI_CS pin to 0

Send the BITSTREAM file (FPGA_bitstream_MCU.bin) from the Build folder at the following path ./your_project/fpga/build/bitstream/.

Set SPI_CS pin to "1" when CONFIG is HIGH (SPI_SI line)

Switch the SPI pins to Hi-Z state within 10 µs after rising edge of SPI_CS

The user can connect our evaluation board or deluxe development board to computer with a USB type-C cable, click “Debug” on ForgeFPGA workshop to lunch the debugging Controls, Click Emulation will load the current project bitstream to FPGA via MCU programming mode.

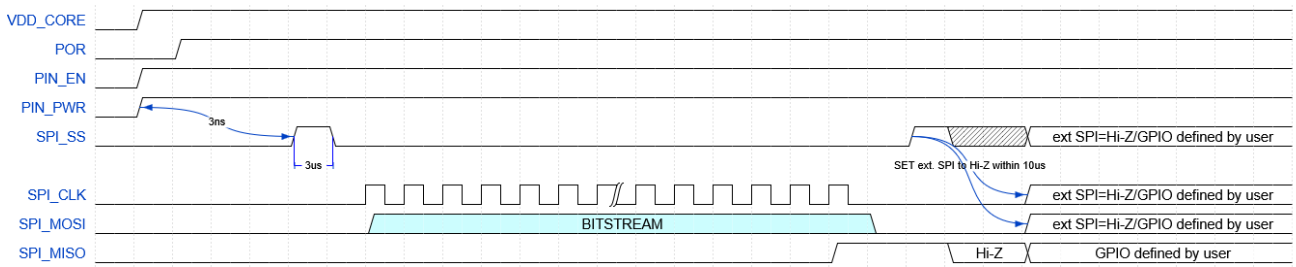

Figure 26. MCU programming waveform for SLG47910

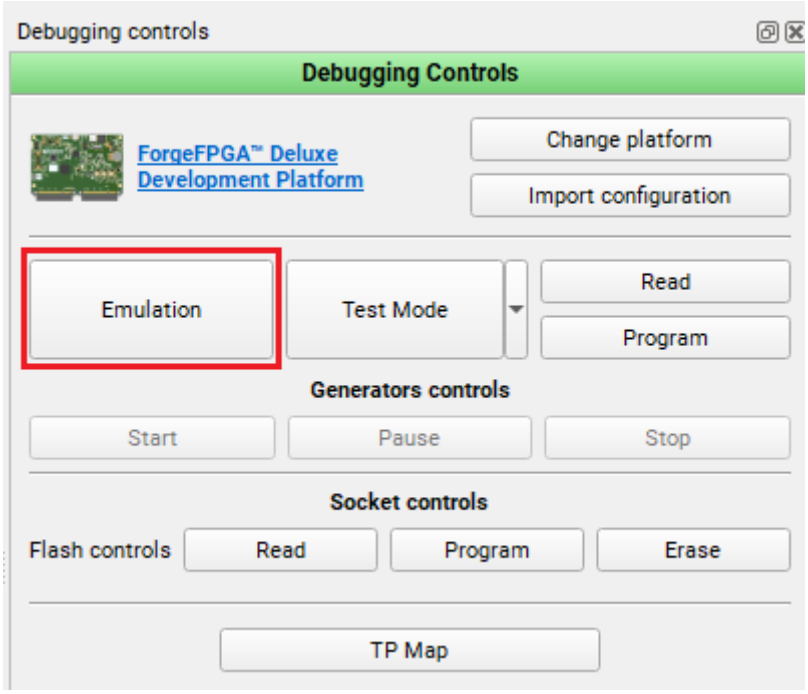

Figure 27. Emulation Design with Debugging Controls Tool

8.2 MCU Programming in the SLG47912/20/21

See Table 6 for a list of GPIOs used for configuration. The configuration process uses the following sequence of steps:

During power-up of the device, the following signals should be in the following states: nSLEEP = HIGH, nRST = LOW, SPI_CS (GPIO0) = LOW.

Wait at least 2 ms for the device to power up and initialize.

Set nRST = HIGH and keep SPI_CS (GPIO0) = LOW.

Check if the CONFIG (SPI_SI, GPIO3) signal is set LOW, which indicates that the SLG47920/21 and SLG47912 is ready to receive configuration data. Once the device is ready, proceed to step 5.

SPI_CS is held LOW by the MCU and begins to send SPI_SCKs and data.

Note: The Send BITSTREAM file (FPGA_bitstream_MCU.bin) can be used from the Build folder at the following path: **./your_project/fpga/build/bitstream/**

The MCU will continue zeroes to send until CONFIG (SPI_SI, GPIO3) is set HIGH which indicates the successful completion of the configuration process.

After all data is received, SPI_CS is set HIGH by the MCU.

For SPI Target (MCU) Configuration mode, the completion of the configuration process is indicated by default on GPIO3 (as described in step 6). The nSLEEP pin shall be kept HIGH during the configuration process. When the MCU detects that GPIO3 is set HIGH, this indicates that the configuration process has completed successfully.

If, after a predefined timeout limit, GPIO3 is still LOW, this indicates that the configuration process has failed.

The SLG47920/21/12 configuration process is finished when the MCU sets SPI_CS = HIGH and this triggers a transition to Functional Mode or to Fail-safe OFF (depending on configuration process result).

Emulation user design with the debugging control tools is the same as SLG47910, see section 8.1 and Figure 27. Emulation is done with MCU programming mode. See Figure 28 for MCU programming waveform.

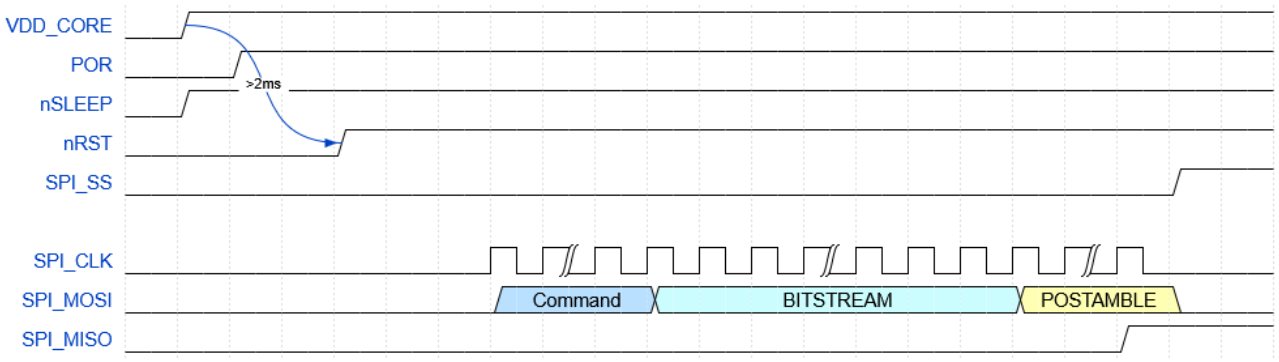

Figure 28. MCU programming waveforms for SLG47912/20/21

9. Go Configure Driver Tool

If the driver required for detecting the development platform is missing, or a conflicting driver is present in the system, the board will not be connected successfully. The Go Configure Driver Tool is designed to resolve this situation (for Windows operating systems). Its main functions are as follows:

- Detect drivers required for successful development platform connection
- Install missing drivers
- Remove drivers
- Resolve driver conflicts (remove conflicting drivers and install missing ones if required)

You can launch the tool manually via the main menu, under *Tools → Go Configure Driver Tool*. It displays the list of the development platforms and the driver statuses (whether the driver is present, missing, or conflicting).

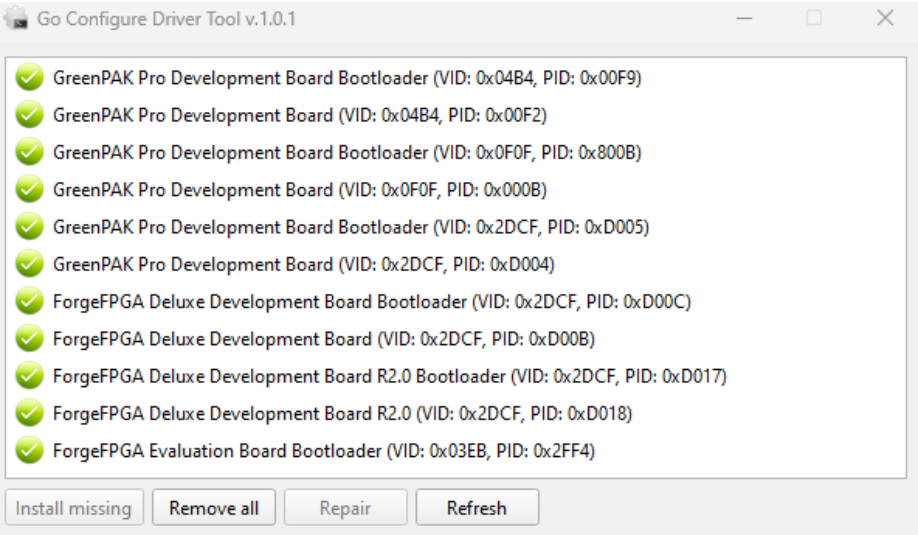

Figure 29. Go Configure Driver Tool

When the platform is selected in the *Debugging Controls panel*, a highlighted banner will appear at the top which will show any driver issues that have been detected. Clicking the Resolve button will launch the Go Configure Driver Tool where the issue can be resolved. In this case, the tool lists the drivers relevant to the selected platform along with their statuses.

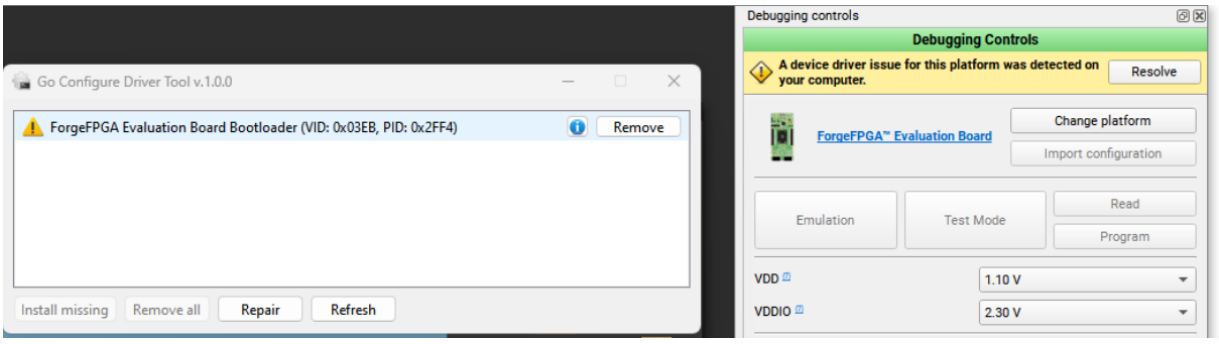

Figure 30. Driver Issue Detection

Click the info icon to find out more information about the conflicting driver that is preventing successful board detection.

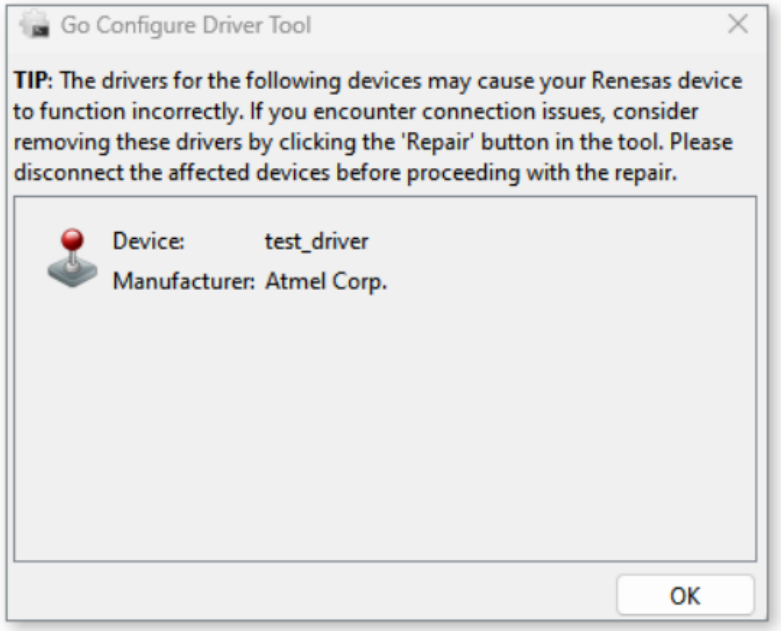

Figure 31. Conflicting Driver info pop-up

Once all driver issues have been resolved, a pop-up window will appear asking to reconnect the platform.

10. Conclusion

This configuration guide focuses on three configuration options: OTP, MCU, and SPI Flash for SLG47910 and for SLG47912/20/21.

The document describes how if the device is configured using OTP, then the other configuration options will be disabled and explains the design and the operation of the ForgeFPGA Development Platform hardware to test and debug designs. If interested, please contact the ForgeFPGA Business Support Team for more information.

11. 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
NVM	Non-Volatile Memory
GPIO	General Purpose Input/Output
QSPI	Quad Serial Peripheral Interface
OCP	Open Core Protocol
PMOD	Peripheral Module
UART	Universal Asynchronous Receiver/Transmitter
POR	Power On-Reset
PLL	Phase Locked Loop
RTL	Register-Transfer Level
USB	Universal Serial Bus

12. Revision History

Revision	Date	Description
2.5	Jan 28, 2026	Added configuration data for SLG47912 Added MCU programming waveform for SLG47920/21/12 Added reference links to website Added more Terms and Defintions
2.4	Dec 12,2025	Added information on: ▪ Program External Flash with a third-party programmer ▪ GoConfigure Development Board ▪ GoConfigure Driver Tool Troubleshooting steps ▪ Project settings figures and descriptions ▪ Boot config Project settings for SLG47910 and SLG47920/21 ▪ List of compatible Flash Devices Updated information on: ▪ document session, third-party programmer part moved to session 8.1 ▪ Deluxe Development Platform ▪ MCU Programming and added SLG47920 programming steps ▪ OTP Write Programming section and added programming steps for SLG47920 Deleted OTP Read instructions Changed SPI Master/Slave Naming to SPI Controller/Target
2.3	Jan 2, 2024	Updated document as per BB revision. Changed the block architecture and removed the power sequence
2.2	Jul 26, 2023	Added updated versions of the socket adapter board and the EVB
2.1	Oct 17, 2022	Added section for how to configure from external flash memory
2.00	Oct 3, 2022	Added contents related to Evaluation Board Added OTP Parameters Table
1.00	Mar 10, 2022	Initial 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.