

ForgeFPGA Configuration

Introduction

This document provides an overview of ForgeFPGA™ configuration methods using External SPI Flash, Internal OTP, and MCU host modes for SLG47910, SLG47912, and SLG47920/21. It covers the required hardware setup, software options, SPI interface behavior, and programming flow for supported ForgeFPGA devices. The guide also includes procedures for debugging and validating configuration using Renesas development platforms.

Contents

1. General Description	3
2. General SPI Interface	5
2.1 SPI Modes with Clock Polarity and Clock Phase	5
3. ForgeFPGA Development Platform	7
3.1 ForgeFPGA Go Configure Development Board (SLG4DVKGOCONF)	7
3.2 ForgeFPGA Deluxe Development Platform (SLG7DVKFORGE)	9
3.3 ForgeFPGA Evaluation Board (SLG7EVBFORGE)	11
4. SLG47910 OTP Write	13
4.1 Boot Config Project Settings	13
4.1.1 Force Load from OTP	13
4.1.2 NVM Lock Status	13
4.2 SLG47910 Writing to the OTP Block	14
5. SLG47912 and SLG47920/21 OTP Write	16
5.1 Boot Config Project Settings	16
5.1.1 Boot	16
5.1.2 NVM Options	16
5.1.3 QSPI Boot Control	16
6. SPI Programming (Controller Mode)	18
6.1 Controller Mode SPI Programming	18
6.2 Configuring the ForgeFPGA from External Flash Memory	19
6.2.1 Programming the External Flash Memory	19
6.2.2 Loading Data into ForgeFPGA	27
7. MCU Configuration Mode (Target Mode)	29
7.1 MCU Configuration in SLG47910	29
7.2 MCU Configuration Mode in SLG47912 and SLG47920/21	30
8. Go Configure Driver Tool	32
9. Conclusion	33
10. Terms and Definitions	34
11. References	35
12. Revision History	36

Figures

Figure 1. Programming and Configuration Interface	3
Figure 2. Configuration Modes in Software	4
Figure 3. SPI Interface.....	5
Figure 4. SPI Mode 0 with CPOL = 0 and CPHA = 0.....	6
Figure 5. Development Platform Selector.....	7
Figure 6. Go Configure Development Board.....	8
Figure 7. Connection between Go Configure Development Board and Socket Card #3	9
Figure 8. Connection between ForgeFPGA Deluxe Development Board and Socket Adapter.....	10
Figure 9. ForgeFPGA Evaluation Board Selection	11
Figure 10. ForgeFPGA Evaluation Board R2.0 Overview.....	11
Figure 11. Boot Config Project Settings.....	13
Figure 12. OTP Write Waveforms.....	14
Figure 13. Boot Config Settings in the Project Settings	17
Figure 14. SPI Release from Deep Power-Down Command	18
Figure 15. SPI Read Fast Command and Deep Power-Down Command	18
Figure 16. Debugging Controls Panel: Program and Read	20
Figure 17. Reading from External Flash	21
Figure 18. Adesto AT25FF041A Compatible Flash Programmer	22
Figure 19. Disconnect Socket Card from Development Board and Remove the Jumper	23
Figure 20. Connect the Flash Programmer to the Onboard Flash (Step 4)	24
Figure 21. Configuring the Programming Software (Step 5).....	25
Figure 22. Loading the Bitstream (.bin) File (Step 6).....	26
Figure 23. Programming the Flash Memory (Step 7).....	27
Figure 24. Jumpers and External Power Connection Points on the Board	27
Figure 25. Choosing the Test Mode (*) Option (Method 2).....	28
Figure 26. MCU Configuration Waveform for SLG47910	29
Figure 27. Emulation Design with Debugging Controls Tool	30
Figure 28. MCU Configuration Waveforms for SLG47912, SLG47920/21	31
Figure 29. Go Configure Driver Tool.....	32
Figure 30. Driver Issue Detection	32
Figure 31. Conflicting Driver Info Pop-up.....	33

Tables

Table 1. Configuration Modes.....	3
Table 2. SPI Modes	6
Table 3. OTP Write Packet Format.....	14
Table 4. Write/Read and SP/AP	15
Table 5. Pin Numbers and SPI Pin Functions for SLG47910	18
Table 6. Pin Numbers and SPI Pin Functions for SLG47920, SLG47912, and SLG47921	18
Table 7. Compatible Flash Parts	19

1. General Description

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 Go Configure™ 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, Serial Clock), 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 the SLG47910.

In the SLG47912 and SLG47920/21, the four configuration pins are GPIO0 (SPI_SCK, Serial Clock), 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.

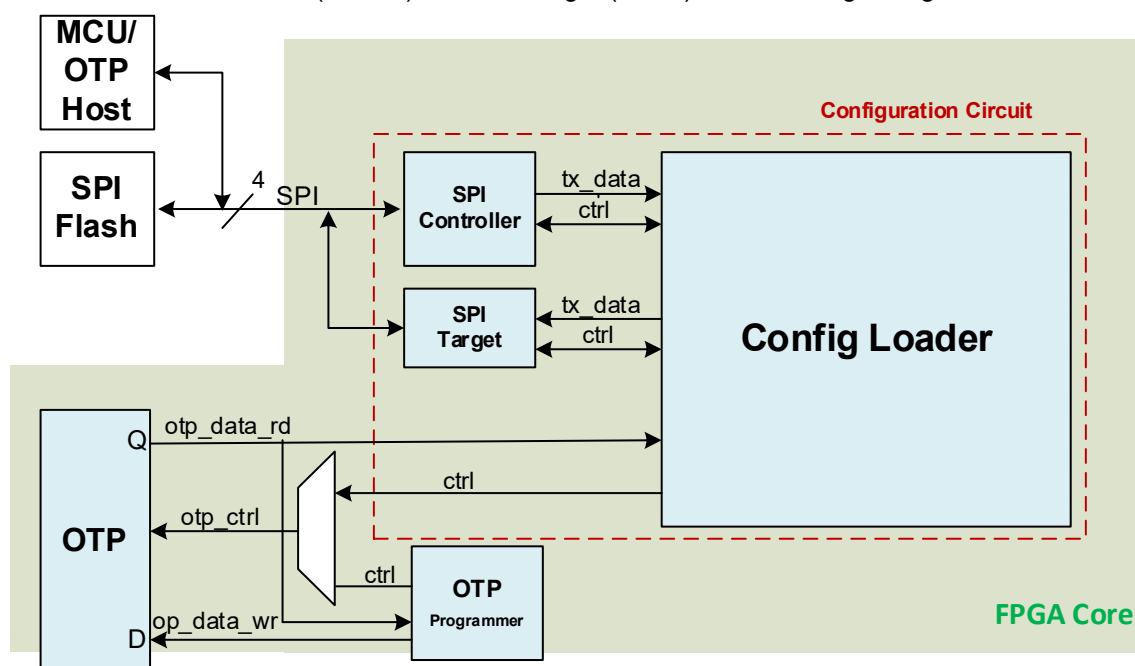


Figure 1. Programming and Configuration Interface

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

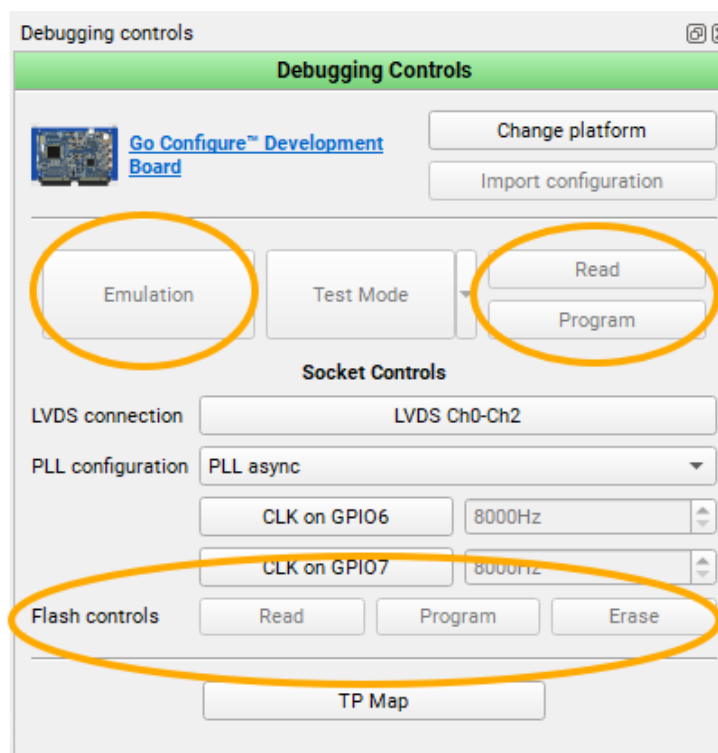


Figure 2. Configuration Modes in Software

2. 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 with the Controller having 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 received at 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 received at the MISO pin by the Controller.
- **CS: Chip Select** (often active low, data output from Controller). Depending on the SPI and Chip Select settings, the CS pin is used to select an individual Target device for communication. When there is one Controller and only a single Target device, then the CS pin is not used. The CS pin is only necessary when the Controller needs to communicate with multiple different Target devices. The CS pin is used to select which Target device to which the Controller needs to communicate. A dedicated CS pin is used for each Target device connected to the Controller.

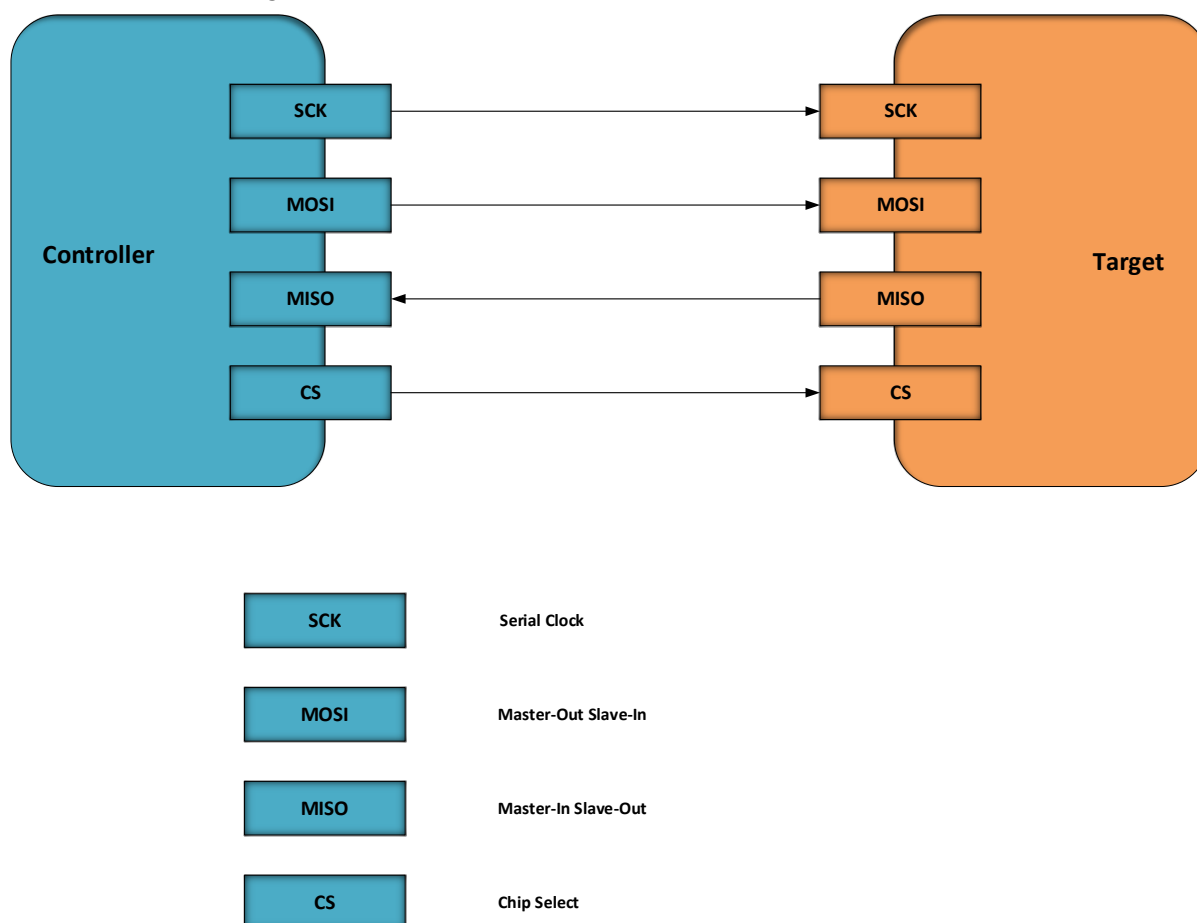


Figure 3. SPI Interface

2.1 SPI Modes with Clock Polarity and Clock Phase

In the SPI interface, the Controller selects the Clock Polarity (CPOL) and Clock Phase (CPHA). The CPOL bit sets the polarity of the clock signal during idle state. The idle state is defined as the period when CS is transitioning. The CPHA bit selects the clock phase. A rising or falling clock edge is used to sample and/or shift the data depending on the CPHA bit selection. Four SPI modes are available, and the mode is selected depending on the CPOL and CPHA bit selection (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 lines.

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

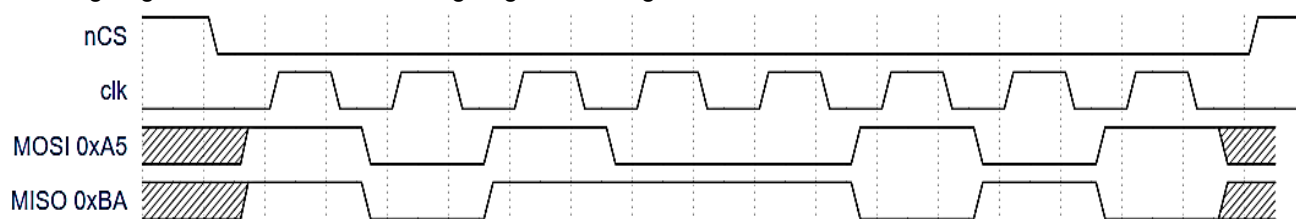


Figure 4. SPI Mode 0 with CPOL = 0 and CPHA = 0

3. ForgeFPGA Development Platform

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

1. **RTL Synthesis:** after creating the desired Verilog Code in the HDL Editor Window of the ForgeFPGA Workshop, the next step is to create a Netlist of the 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. For more information, see the *ForgeFPGA Workshop User Guide*, [1].
2. **Generating the Bitstream:** to prepare the design for programming, the Place-and-Route procedure must be performed. This process maps the primitives in the synthesized Netlist to the FPGA's physical resources. After the Netlist is generated successfully, click the **Generate Bitstream** button on the control panel. After these two steps are completed, the design is ready to be sent to the device. For more information, see the *ForgeFPGA Workshop User Guide*, [1].

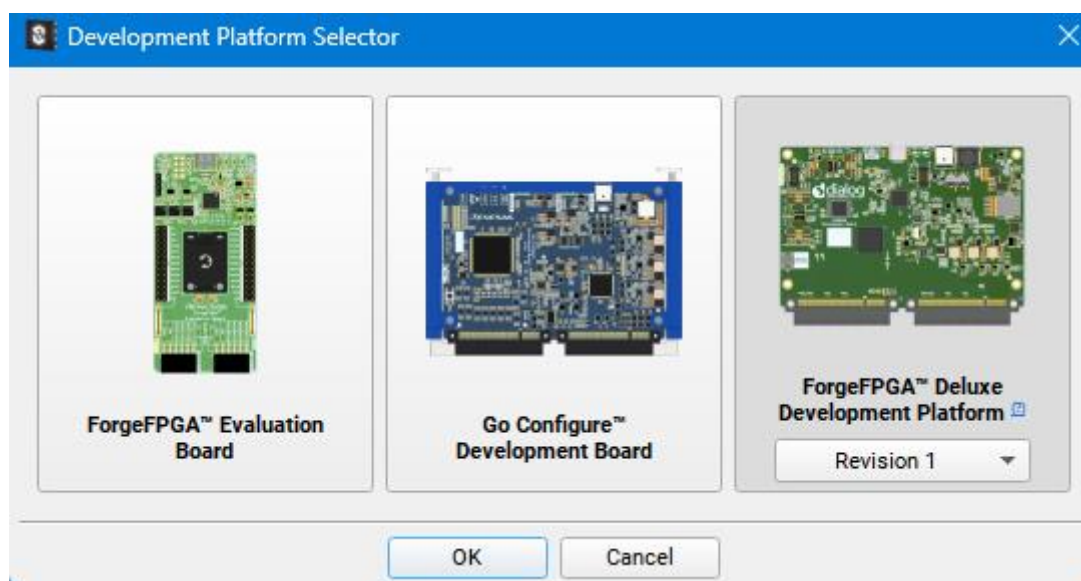


Figure 5. Development Platform Selector

3.1 ForgeFPGA Go Configure Development Board (SLG4DVKGOCNF)

The Go Configure Development Board (see [Figure 6](#)) is an updated platform from Renesas, designed for development using products such as the GreenPAK™ family of mixed-signal ICs and ForgeFPGA programmable arrays. The board works in combination with the Go Configure Software Hub and provides rich functionality, such as:

- Programming and Emulation
- 80 multi-function configurable I/O lines Test Points (TPs)
 - 32 channels Digital Pattern Generator
 - 16 channels Analog Waveform Generator (AWG)
- Dual PCIe connector
- Flexible power management with software-configurable protection (OVP, OCP)
- USB 2.0 interface for communication (USB-C connector)

Operation 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. These cards are built for specific devices and packages and come with a minimum required set of external components for proper IC configuration and operation, as well as extra features such as a power selector, clocks, I/Os exposed to PMOD

connectors, among others. Check documentation for specific socket card for more information about pin mapping and available onboard resources.

For more information on the Go Configure Development Board and for specific socket cards, see section [11 References](#).

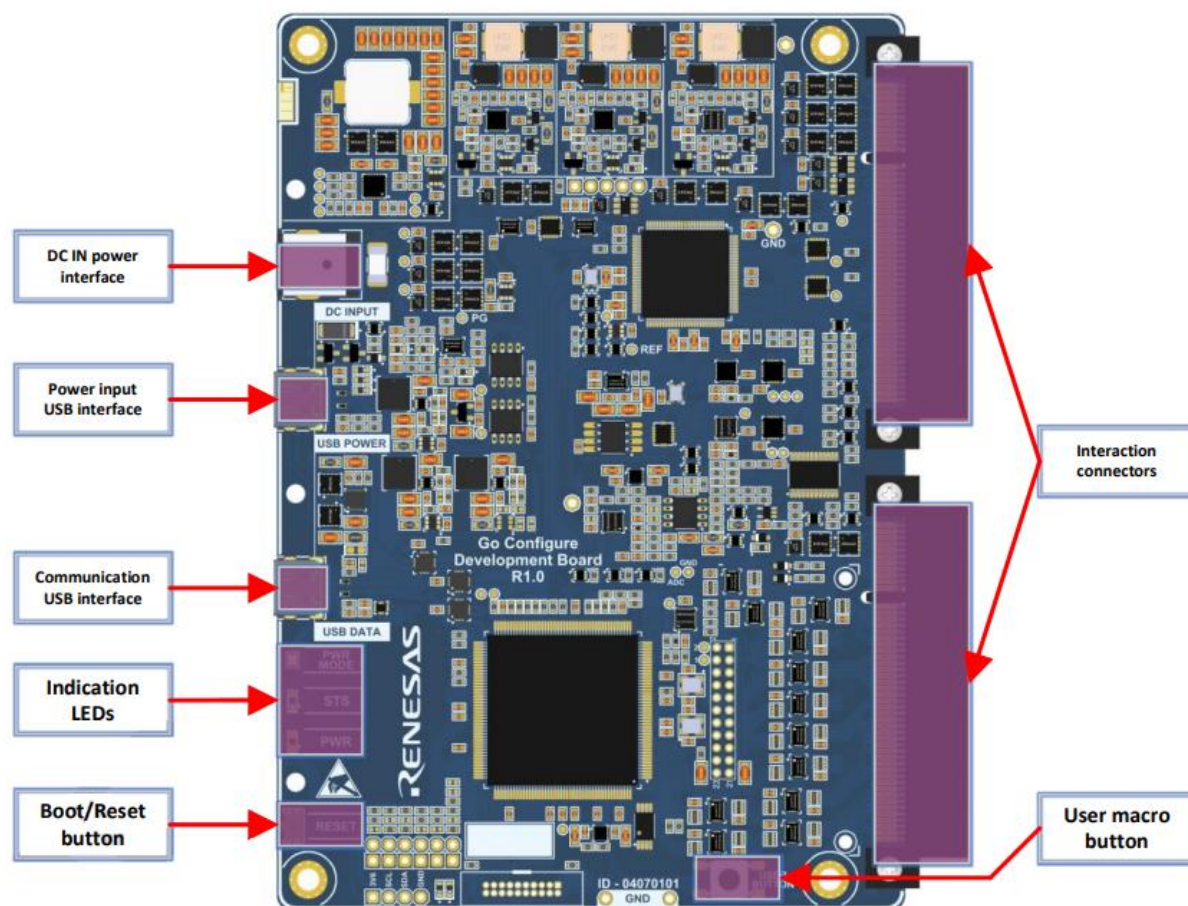


Figure 6. Go Configure Development Board

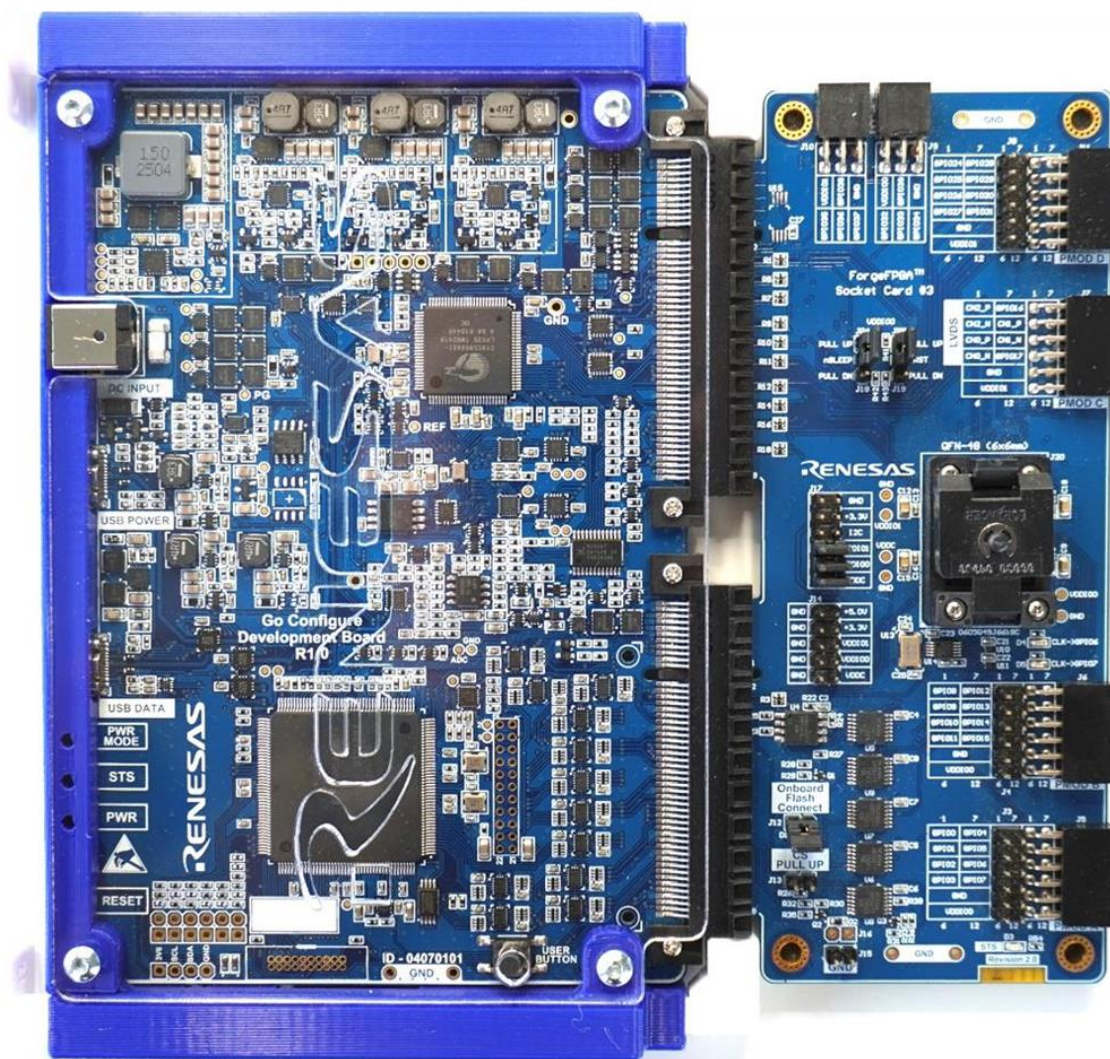


Figure 7. Connection between Go Configure Development Board and Socket Card #3

3.2 ForgeFPGA Deluxe Development Platform (SLG7DVKFORGE)

Under the **Debug** button on the toolbar, select the ForgeFPGA Deluxe Development Platform as the platform (see [Figure 5](#)) to begin configuration. 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 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 adapters.

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 to +3.465 V) and a maximum available output current of 2A. The same voltage is supplied to the GPIO, for keeping the logic level compatibility with the circuit under test.

Using the Deluxe Development Platform the device can be configured via MCU mode, OTP mode, or External SPI mode.

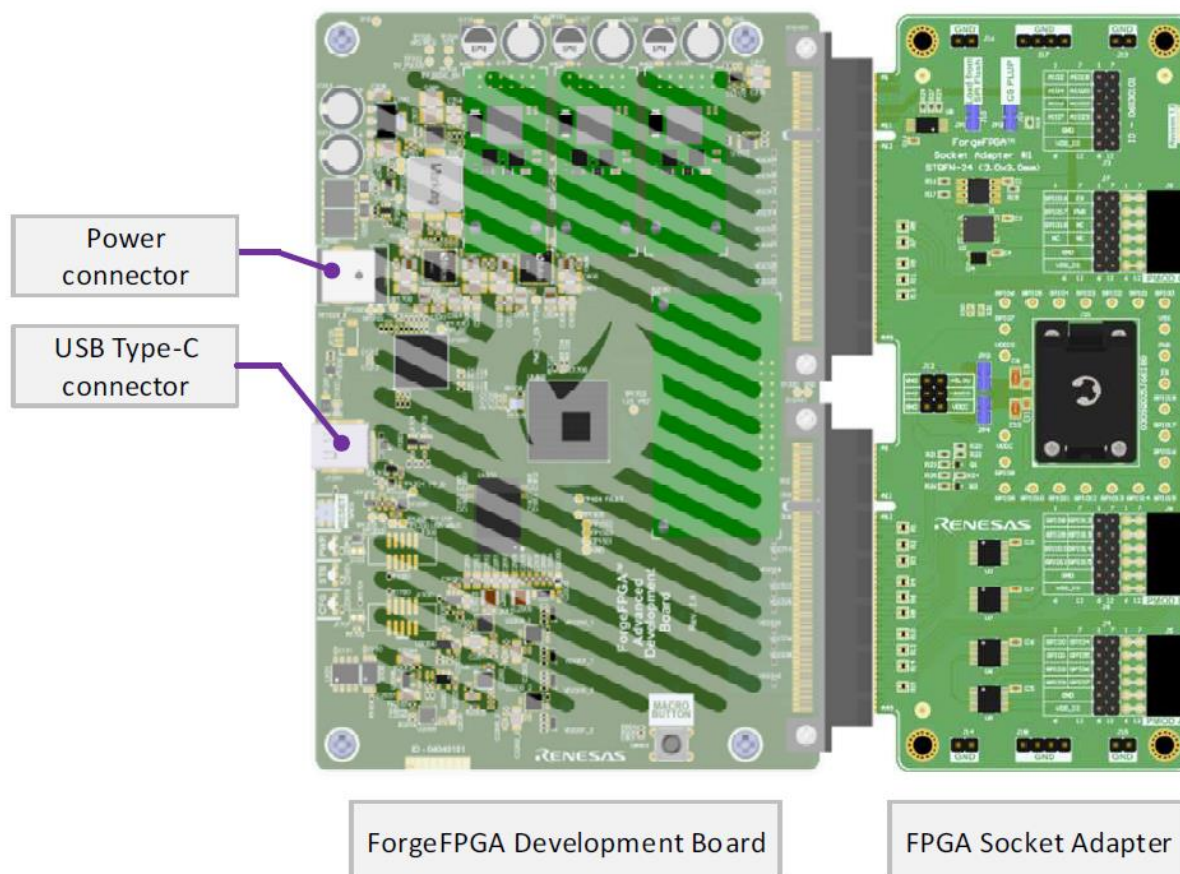


Figure 8. Connection between ForgeFPGA Deluxe Development Board and Socket Adapter

For more information on configuration with the Deluxe Development Boards, see section [11 References](#).

Select the ForgeFPGA Evaluation Board under the **Debug** button on the toolbar (see [Figure 9](#)) to see the debug controls.



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



The ForgeFPGA Evaluation Board provides the support for design Emulation, OTP programming, and real-time testing for the SLG47910 IC hardware. The EVB is controlled by the Go Configure Software Hub Emulation, V_{DD}/V_{DDC} regulation, IC programming, and internal UART Terminal options.

The ForgeFPGA Evaluation Board R2.0 (see [Figure 10](#)) has the following connectors:

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

It includes three LEDs for Power, Status, and Done, and has one **Boot/Reset** button.

For more information on using the EVB, see the *ForgeFPGA Evaluation Board R2.0 User Manual*, [\[5\]](#).

4. 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.

4.1 Boot Config Project Settings

Boot Config can be configured before writing to the OTP in the **Project Settings** GUI. The **Project Settings** GUI can be launched from the toolbar of the main software window (see [Figure 11](#)).

4.1.1 Force Load from OTP

Disable/Enable is used to select **Force Load from OTP**. If **Enable** is selected before programming the 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 **Disable** is selected before programming the OTP, then loading from an External Flash or MCU after writing the OTP is an option.

4.1.2 NVM Lock Status

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

If **Unlocked** is selected before writing to the NVM, the NVM data can be read after programming. If **Read lock** is selected, the NVM data cannot be read after programming. If **Write lock** is selected, the NVM cannot be written again after programming (MTP only). If **Lock read and write** is selected, all read and write access to the NVM is disabled after programming. A notification indicates when the NVM cannot be read. These lock options help protect stored data.

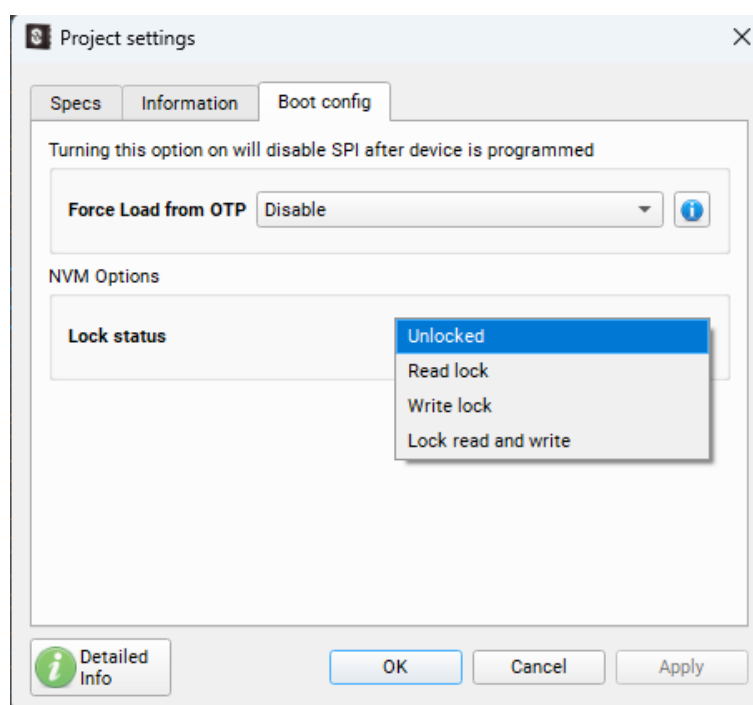


Figure 11. Boot Config Project Settings

4.2 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. The OTP_programmer (see [Figure 1](#)) sequences the internal signals to enable the OTP Write operation.

When 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 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:

1. Wait for POR and PLL lock time (1300 μ s), then send the Signature Bytes through SPI_SO (GPIO5) by keeping SPI_CS (GPIO4) LOW.
2. 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 μ s with an additional delay of 18 μ s between the 1st and 2nd packet and a delay of 10.11 μ s for any subsequent packets.
3. The last Write packet is indicated by Byte 8, Bit 6 = 1.
4. 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

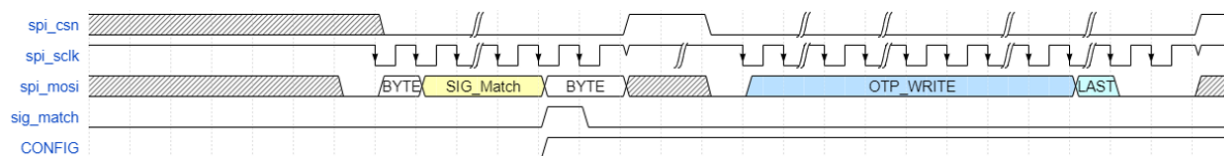


Figure 12. OTP Write Waveforms

Table 4. Write/Read and 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

5. SLG47912 and SLG47920/21 OTP Write

For the SLG47912 and SLG47920/21, 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** (see [Figure 2](#)).

5.1 Boot Config Project Settings

Boot Config can be configured before writing to the OTP in the **Project Settings** GUI. The **Project Settings** GUI can be launched from the toolbar of the main software window (see [Figure 13](#)).

5.1.1 Boot

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.

5.1.2 NVM Options

The available **NVM Options** are identical to the SLG47910V, see section [4.1.2 NVM Lock Status](#) for **Lock status** options.

There is an option to choose what will be included in the OTP configuration file. The two selections are either to include **Full FPGA Configuration** information or to include **Boot config + OTP user data only**.

5.1.3 QSPI Boot Control

External Flash **QSPI mode** controls QSPI data transfer modes. **Single mode** is standard SPI mode with one I/O interface. **Dual mode** is SPI mode with a dual I/O interface that can double the transfer rate compared to **Single mode**.

5.1.3.1 QSPI Boot Control Logic

This option unlocks multi-bitstream controls. **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. If needed, change the default boot address, even if boot control logic is disabled.

In the **BIT** (Bitstream) row, clicking the **DIS** (disable) button for any of the numbered columns #3, #2, #1, #0 enables the corresponding Bitstream, changing the displayed text from **DIS** (disabled) to **EN** (enabled). This selection enables the address control code bit and allows to choose a source for the control bit (either **GPIO** or **IOB**). When all four BITs (corresponding to GPIO12, GPIO13, GPIO14, and GPIO15) are in EN mode, 16 different Bitstreams from External Flash can be addressed. The **QSPI boot addresses** must be defined when enabling those boot control selection bits (see [Figure 13](#)). 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, reset the chip via the nRST pin and then choose the Bitstream using the selected control code source.

Project settings

Specs

Information

General

Boot config

CONFIG Pull Up

Boot

Boot priority

SPI_CS line state

Secondary path for MCU boot

Unlocked

NVM Options

Lock status

Unlocked

OTP configuration file includes

Full FPGA configuration

QSPI Boot control

QSPI mode

Single mode

QSPI boot control logic

Disable

QSPI boot control code selection

	#3	#2	#1	#0
BIT	DIS	DIS	DIS	DIS
GPIO/IOB	GPIO15	GPIO14	GPIO13	GPIO12

	Control code	QSPI boot address, hex
B0	0000 (default)	0x000000
B1	0001	0x000000
B2	0010	0x000000
B3	0011	0x000000
B4	0100	0x000000
B5	0101	0x000000

Detailed Info

OK

Cancel

Apply

Figure 13. Boot Config Settings in the Project Settings

R19US0005EU0251 Rev.2.51
Sep 10, 2026

RENESAS

Page 17

6. 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 shown in Figure 14.

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 to 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.

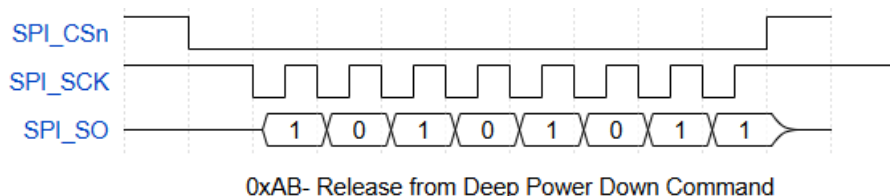


Figure 14. SPI Release from Deep Power-Down Command

After 10 µs has passed, the SPI Controller sends a Fast Read command with a 24-bit address to the SPI_SO pin and receives the data on the SPI_SI port, as shown in Figure 15. 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.

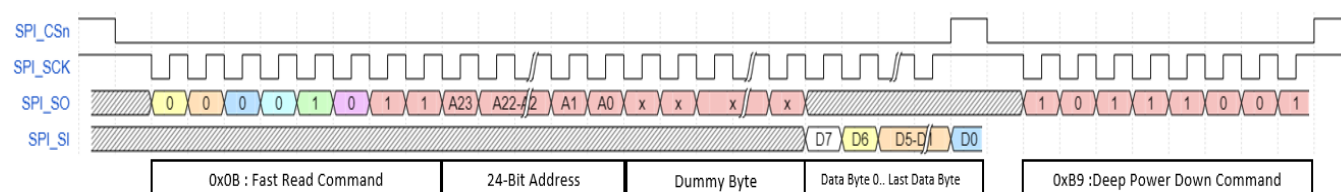


Figure 15. SPI Read Fast Command and Deep Power-Down Command

6.1 Controller Mode SPI Programming

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)

Table 6 lists the pins of the SLG47920, SLG47912, and SLG47921 and their corresponding SPI functions.

Table 6. Pin Numbers and SPI Pin Functions for SLG47920, SLG47912, and SLG47921

SLG47920V	SLG47912V, SLG47921V	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 ForgeFPGA device is configured using a single data pin (SPI_SI).

To enable SPI configuration:

1. Hold SPI_CS HIGH for a minimum delay of 1055 μ s to enable SPI mode.
2. After entering SPI mode, internal logic will release SPI_CS from driving through the GPIO.
3. The internal Config-Circuit, which acts as the Controller, will control the SPI signals (SPI_CS, SPI_SCK, SPI_SI, and SPI_SO) throughout the FPGA's configuration.
4. 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).
5. The Config-Circuit will generate an internal Config signal, Done = 1.

6.2 Configuring the ForgeFPGA from External Flash Memory

The ForgeFPGA socket adapter and all the socket cards have an Adesto AT25FF041A 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 makes it possible to program and debug the External Flash Memory on the ForgeFPGA socket adapter.

Table 7 shows a list of compatible Flash parts supported by ForgeFPGA.

Table 7. Compatible Flash Parts

Flash Brand	Flash Part Number
Adesto	25FF041A
Adesto	25SF641
Adesto	25DF081A
Renesas	25EU0021A
Renesas	25EU0081A
Renesas	25EU0161A
Winbond	W25X40
Winbond	W25Q64
Winbond	W25Q128
Atmel	AT25DF041A
MXIC	MX25L25645G
GigaDevice	GD25Q16CSIG

6.2.1 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 the 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 (see Figure 16). A notification will be given once the Bitstream has been successfully uploaded to the External Flash Memory.

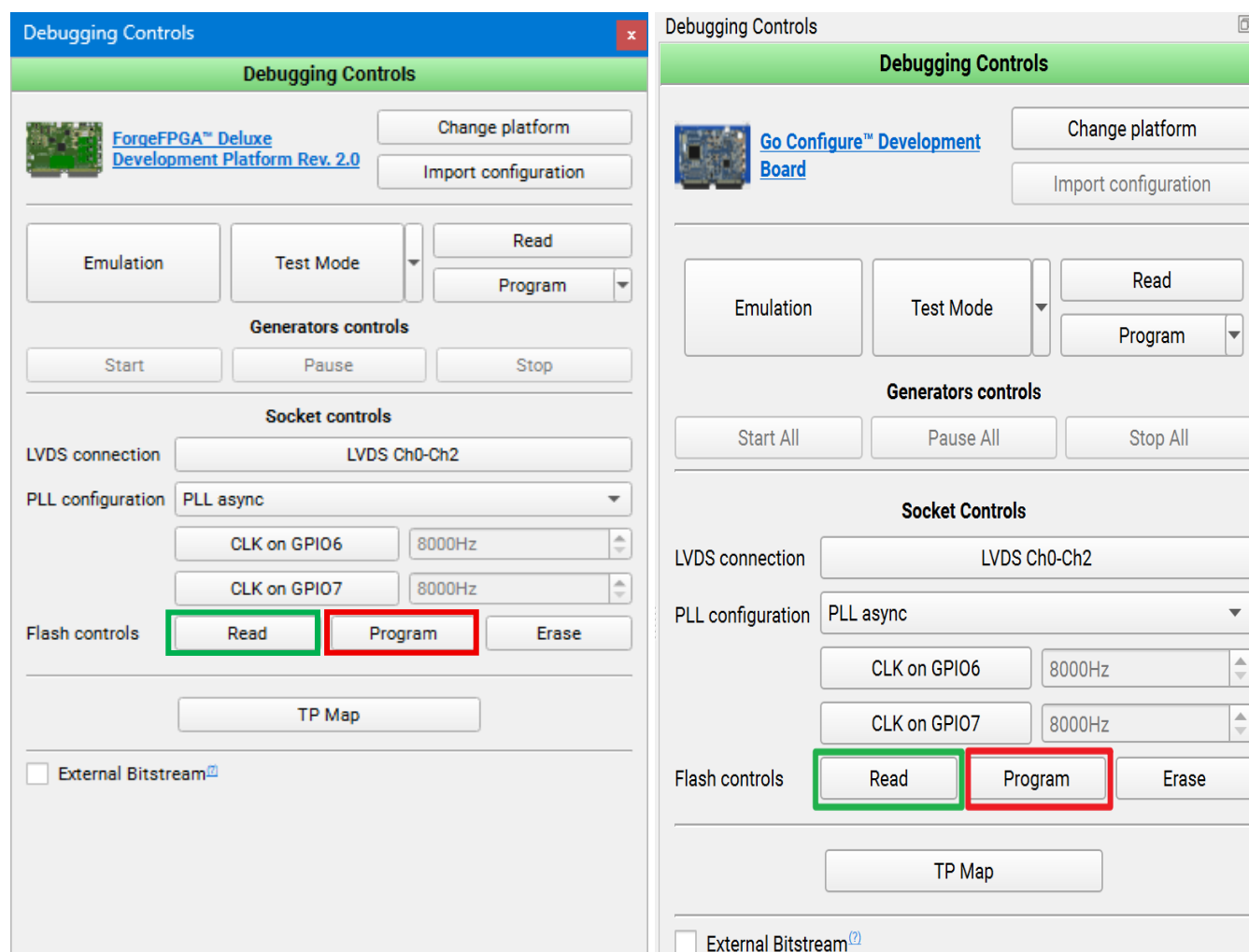


Figure 16. Debugging Controls Panel: Program and Read

The uploaded Bitstream data can be read back by clicking the **Read** button (see [Figure 17](#)).

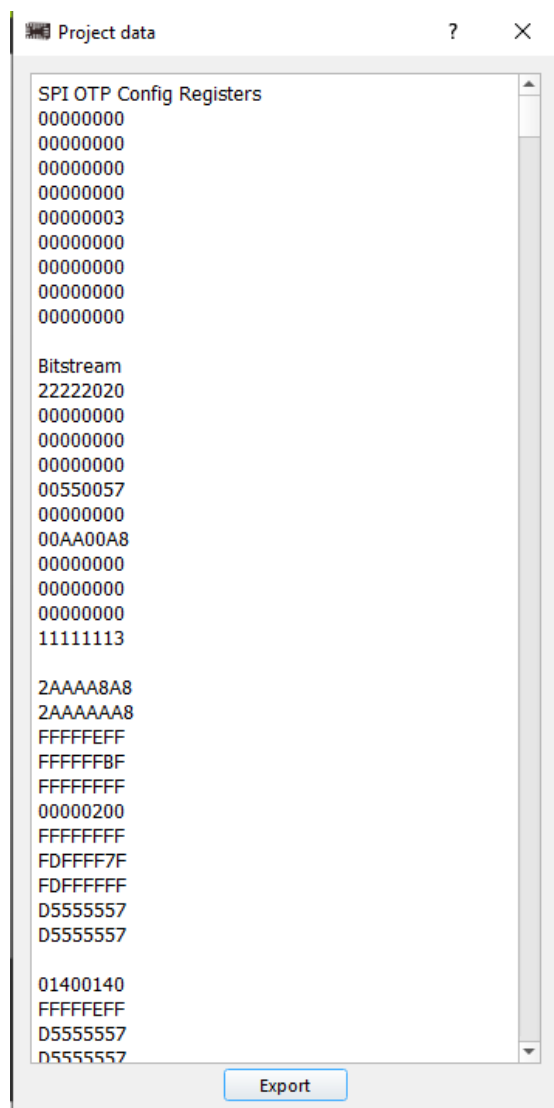


Figure 17. Reading from External Flash

Method 2: Using a Third-party Programmer

The ForgeFPGA socket adapters and socket cards use 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 18](#) illustrates an example of a compatible Flash programmer, provided for reference only.



Figure 18. Adesto AT25FF041A Compatible Flash Programmer

To program the External Flash:

1. Ensure that the ForgeFPGA Development Board is powered off. No external power is required during programming.
2. Disconnect the ForgeFPGA Socket from the Development Board.
3. Remove the jumper labeled “Onboard Flash connect” on the socket card (see [Figure 19](#)).

Note: This method can be used with both the ForgeFPGA Socket Adapter #1 and #3, and all ForgeFPGA socket cards. If working with the ForgeFPGA Socket Adapter #1, proceed with step 4 without removing any jumpers.

4. Connect the Flash Programmer.



Figure 19. Disconnect Socket Card from Development Board and Remove the Jumper

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 (see [Figure 20](#)).

Note: The red wire in this clamp indicates Flash chip Pin 1. [Figure 20](#) shows the connection with Socket Card #3. The connection for Socket Card #1 and all ForgeFPGA socket cards 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.

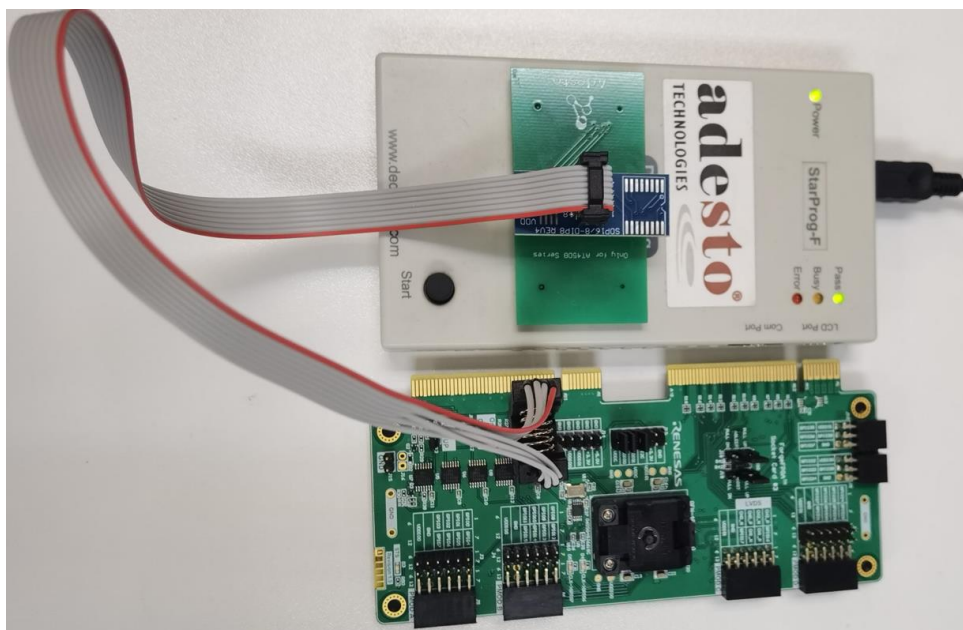


Figure 20. Connect the Flash Programmer to the Onboard Flash (Step 4)

Configure the DediWare programming software (see [Figure 21](#)):

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

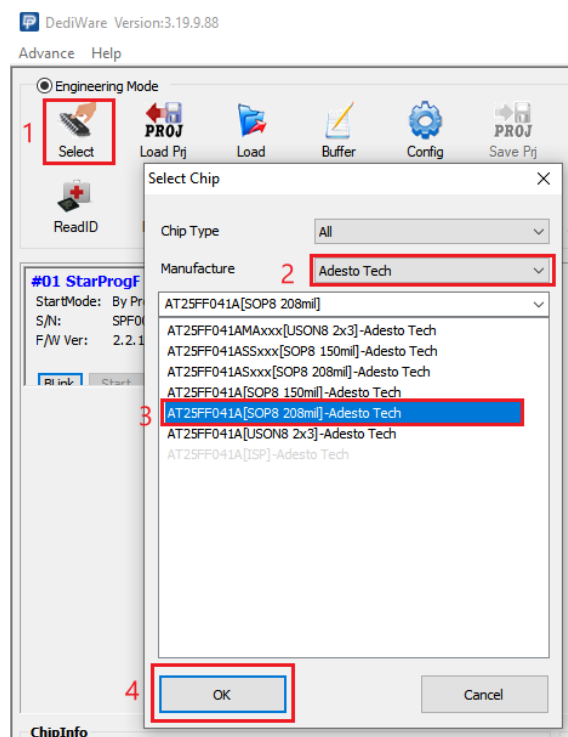


Figure 21. Configuring the Programming Software (Step 5)

Load the Bitstream file (see [Figure 22](#)):

1. Open File Selection dialog.
 - a. Click the **Load** button under Engineering Mode.
 - b. A Select File window will appear.
2. Locate the Bitstream file.
3. Click the **View** button next to FilePath.
 - a. Navigate to the ForgeFPGA project directory: `/project name/ffpga/build/bitstream`.
 - b. Select the file `FPGA_bitstream_Flash_MEM.bin`.
4. Confirm selection.
 - a. Click **OK** to load the file.

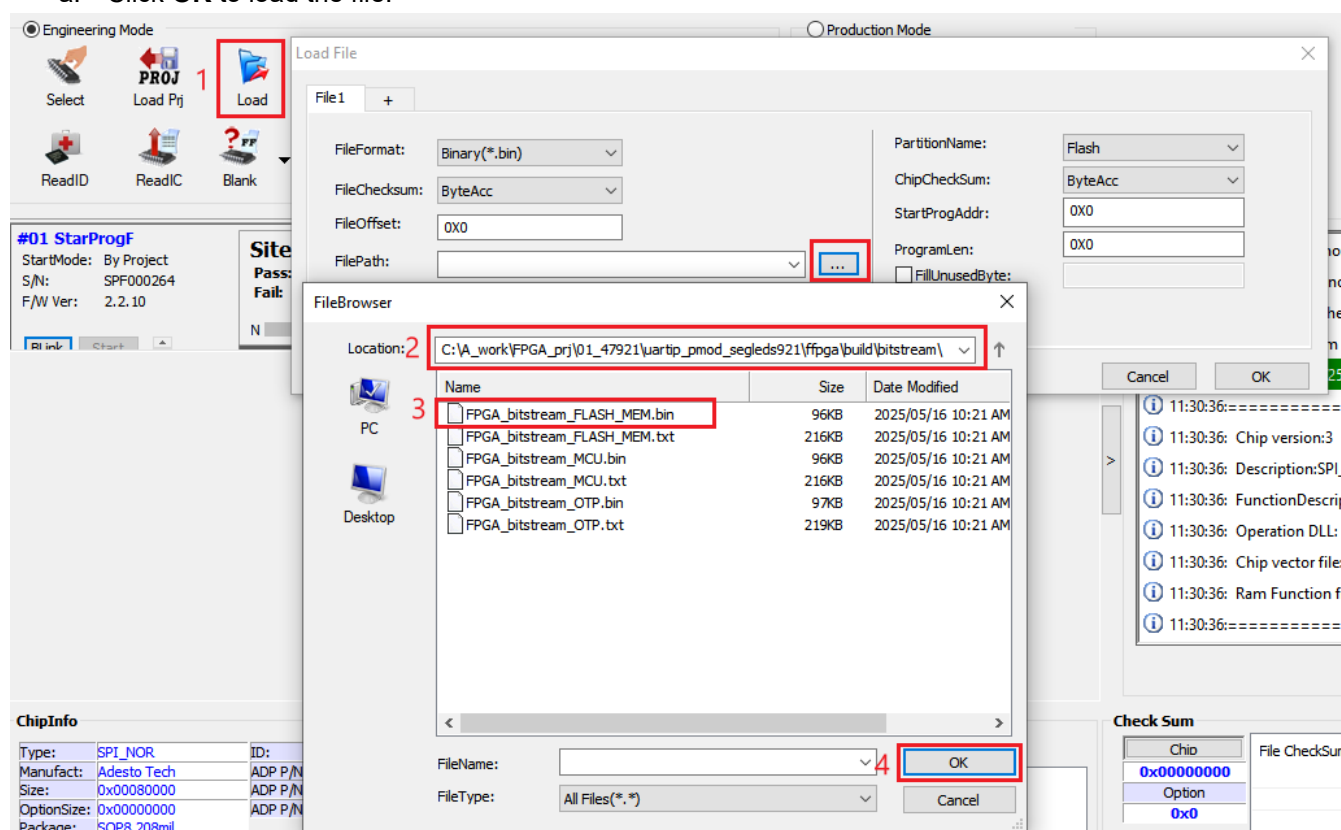


Figure 22. Loading the Bitstream (.bin) File (Step 6)

Program the Flash Memory (see [Figure 23](#)):

1. Initiate the Programming Process.
 - a. Click the **Program** button located under Engineering Mode.
2. Select the Programming Target.
 - a. In the pop-up window, click **Flash** to begin programming.
3. Monitor Progress.
 - a. Wait for the progress bar to complete (100%).
 - b. Do not interrupt the process until the program is completed.

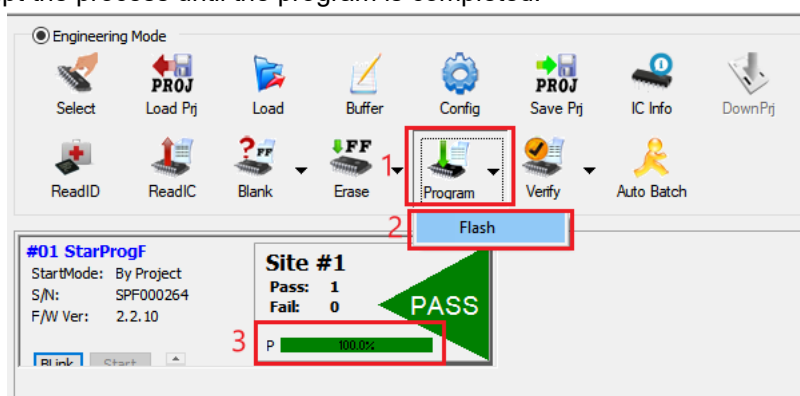


Figure 23. Programming the Flash Memory (Step 7)

6.2.2 Loading Data into ForgeFPGA

When the Bitstream has been programmed into the External Flash Memory, it can be loaded to the FPGA chipset. To do this, the jumper **FPGA<>Flash** (see [Figure 24](#)) must be disconnected in order to connect to the onboard External Flash Memory (for the Socket Adapter Card #1, revision 1.0) and the jumper **Onboard Flash Connect** must be disconnected in order to connect to the onboard External Flash Memory for other socket cards (there is no need to remove jumpers for the Socket Adapter Card #1, revision 1.1).

Method 1:

The ForgeFPGA socket card first must be disconnected from the Go Configure Development Board. To power the socket card, use an external power source and connect it to the connection headers (purple rectangles in [Figure 24](#)) V_{DDIO} and V_{DDC} . Check the pin configuration of these headers, specified on the bottom of the socket card.



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

Method 2:

From the Debugging Controls panel in the Go Configure Software Hub, choose the **Test Mode** from the drop-down menu. Next, choose the **Test Mode (*)** option to upload the Bitstream from the External Flash Memory to the ForgeFPGA platform (see Figure 25).

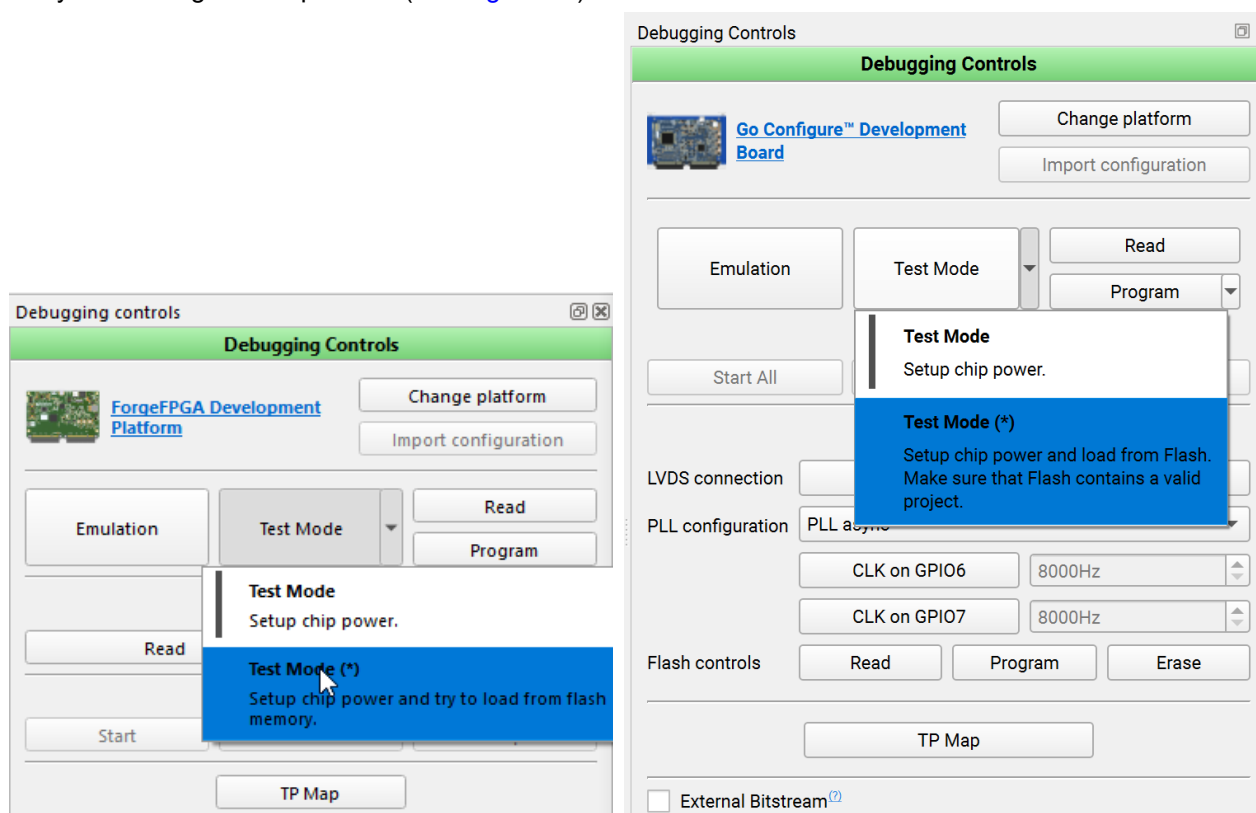


Figure 25. Choosing the Test Mode (*) Option (Method 2)

7. MCU Configuration Mode (Target Mode)

MCU Configuration Mode corresponds to the Emulation mode in the ForgeFPGA software (see [Figure 2](#)). Emulation mode allows to test the functionality of the design without a need to program the part OTP.

7.1 MCU Configuration in 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 as follows:

1. Power On, set SPI_CS pin to '0' and set nRST (PWR) and nSLEEP (EN) pins to '1'.
2. Wait 3 ms.
3. Set SPI_CS pin to '1'.
4. Wait 3 μ s.
5. Set SPI_CS pin to '0'.

Note: Send the Bitstream file (FPGA_bitstream_MCU.bin) from the Build folder at the following path: `/project name/ffpga/build/bitstream/`.

6. Set SPI_CS pin to '1' when CONFIG is HIGH (SPI_SI line).
7. Switch the SPI pins to Hi-Z state within 10 μ s after rising edge of SPI_CS.

Connect the ForgeFPGA Evaluation Board or ForgeFPGA Deluxe Development Board to the computer with a USB Type-C cable, then launch the Debugging Controls by clicking the **Debug** button on the ForgeFPGA Workshop toolbar. Clicking **Emulation** will load the current project Bitstream to FPGA via MCU programming mode.

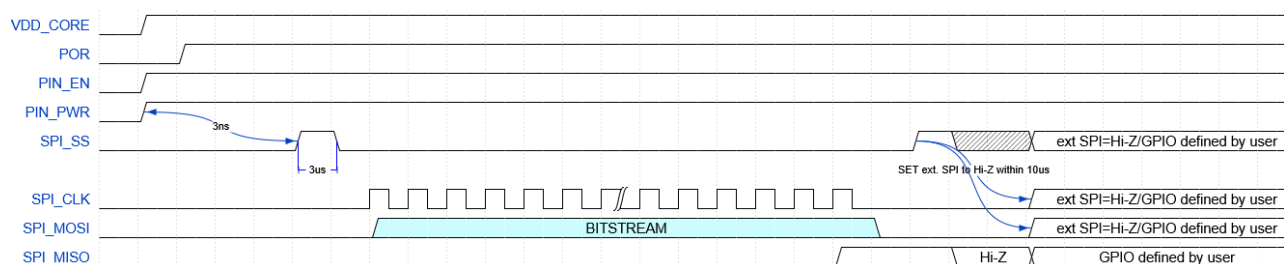


Figure 26. MCU Configuration Waveform for SLG47910

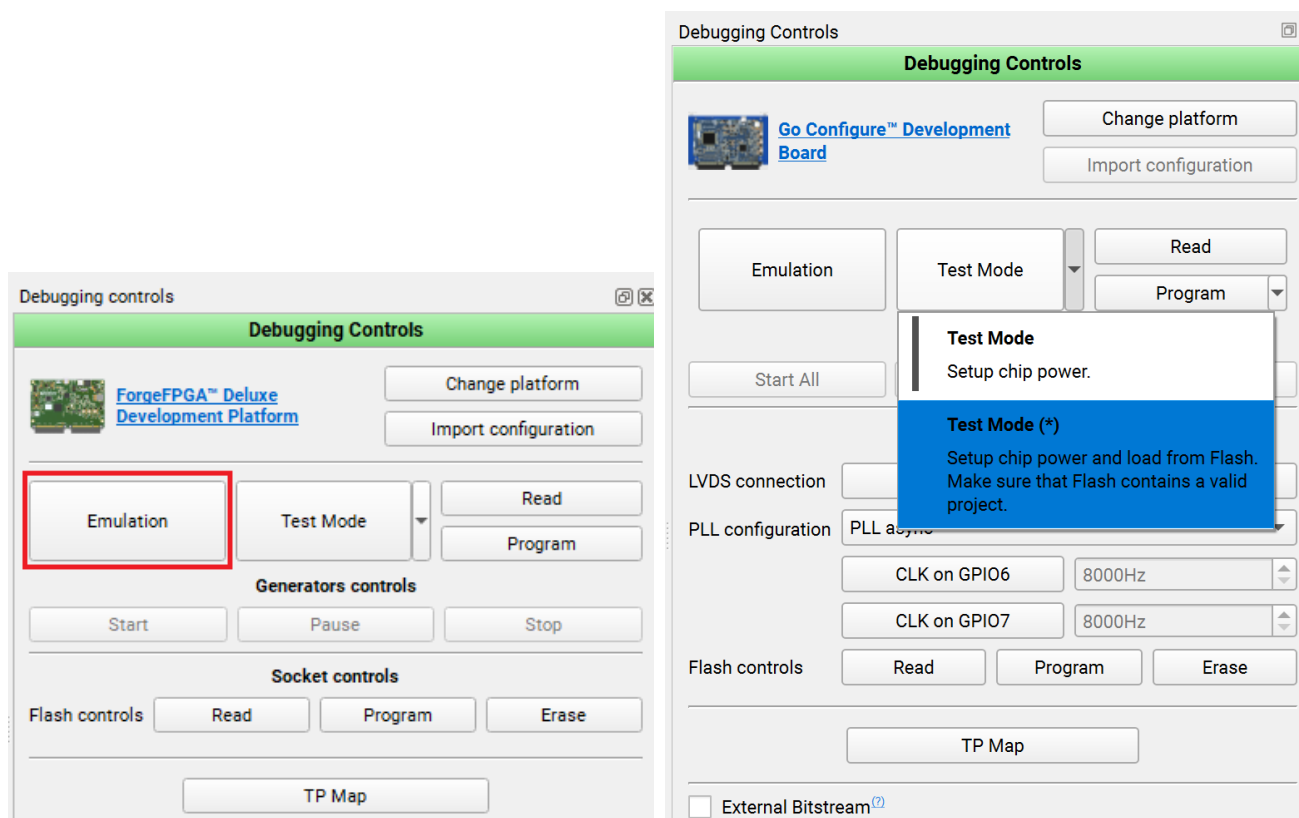


Figure 27. Emulation Design with Debugging Controls Tool

7.2 MCU Configuration Mode in SLG47912 and SLG47920/21

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

1. During power-up of the device, the following signals should be in the following states: nSLEEP = 1, nRST = 0, SPI_CS (GPIO0) = 0.
2. Wait at least 2 ms for the device to power up and initialize.
3. Set nRST to '1' and keep SPI_CS (GPIO0) = 0.
4. Check if the CONFIG (SPI_MISO, GPIO3) signal is set to '0', which indicates that the SLG47920/21 and SLG47912 is ready to receive configuration data. When the device is ready, proceed to step 5.
5. SPI_CS is held '0' by the MCU and begins to send SPI_SCKs and data. The data is expected to be received in the following order:
 - a. CMD – 1 byte to define SPI mode: 02h – single
 - b. SyncWord (0x22FF22AA)
 - c. 1024-bits of SPI/OTP Registers data
 - d. BRAM memory initialization data
 - e. Bitstream

Note: The Bitstream file (FPGA_bitstream_MCU.bin or FPGA_bitstream_MCU.txt) can be used from the Build folder at the following path: /project name/ffpga/build/bitstream/.

6. 1 byte command 0x02 must be sent before the Bitstream.
7. The MCU will continue to send SPI_SCKs (POSTAMBLE) until CONFIG (SPI_MISO, GPIO3) is set to '1', which indicates the successful completion of the configuration process.
8. After all data is received, SPI_CS is set to '1' 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 should be kept HIGH during the configuration process. When the MCU detects that GPIO3 is set to '1', 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 and SLG47912 configuration process is finished when the MCU sets SPI_CS = 1 and this triggers a transition to Functional Mode or to Fail-safe OFF (depending on configuration process result).

Emulation of the design with the debugging control tools is the same as for the SLG47910, see section [7.1 MCU Configuration in SLG47910](#) and [Figure 27](#). Emulation is done with MCU Configuration mode. See [Figure 28](#) for MCU Configuration waveform.

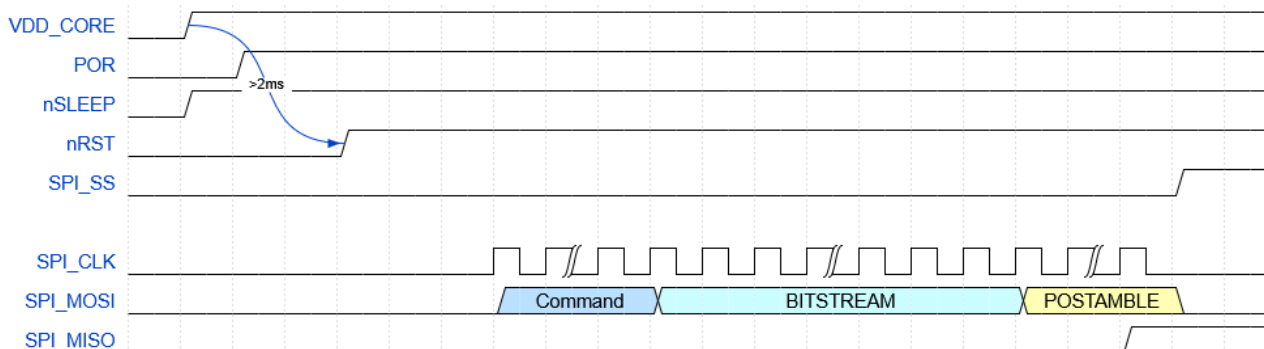


Figure 28. MCU Configuration Waveforms for SLG47912, SLG47920/21

8. 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)

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).

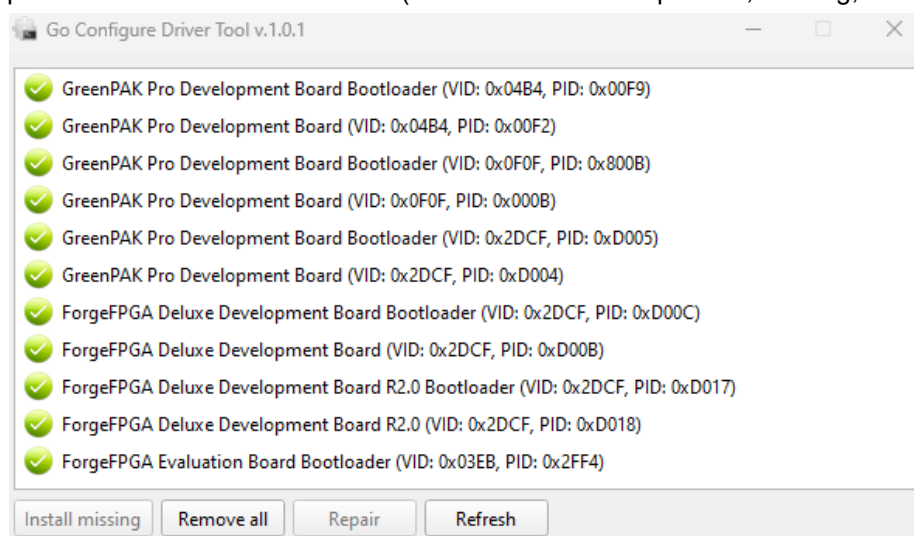


Figure 29. Go Configure Driver Tool

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.

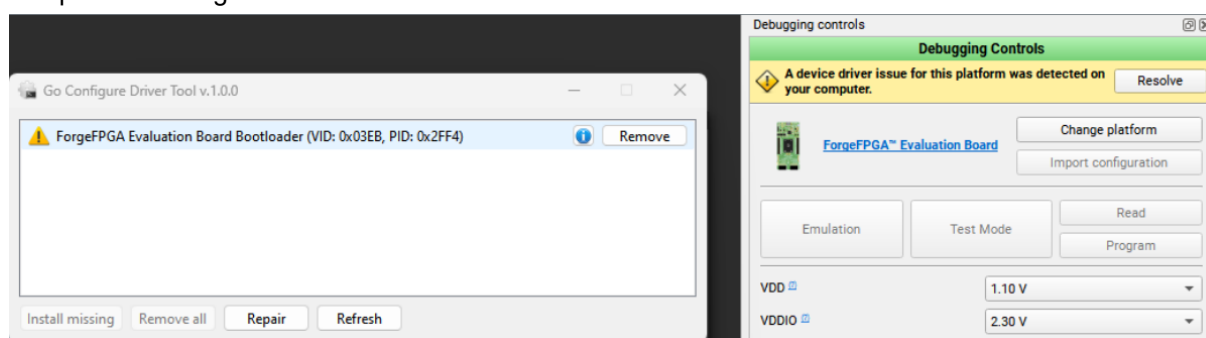


Figure 30. Driver Issue Detection

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

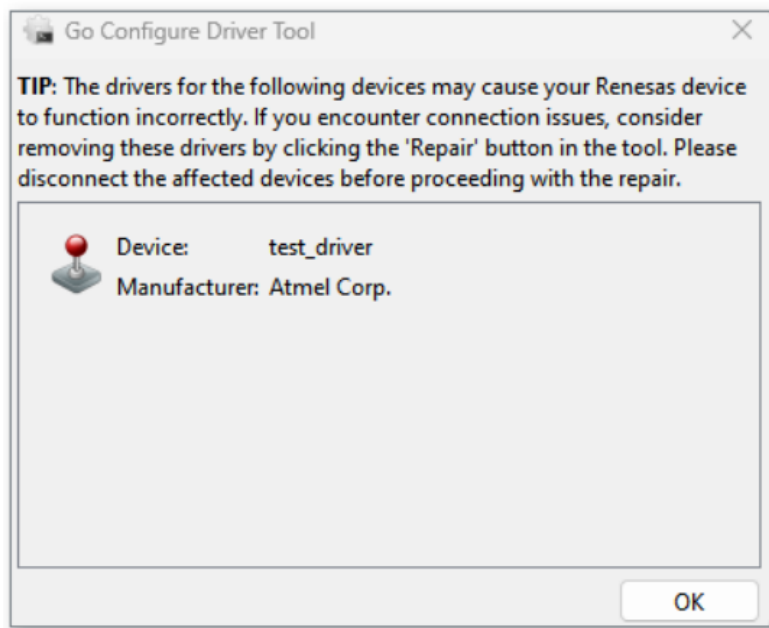


Figure 31. Conflicting Driver Info Pop-up

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

9. Conclusion

This guide summarizes the available methods for configuring the ForgeFPGA device, including External SPI Flash, Internal OTP, and MCU host configuration modes. It also explains the required hardware setup, software settings, SPI interface behavior, programming flow, and debugging options across the supported ForgeFPGA devices and development platforms. The procedures described in this document allow to select the appropriate configuration method, generate and load the required Bitstream, verify programming results, and troubleshoot common setup or driver-related issues.

Proper configuration of boot settings, memory options, and board connections is essential to ensure reliable operation of ForgeFPGA in development, Emulation, and final programmed use cases. For more information, contact the [ForgeFPGA Business Support Team](#).

10. Terms and Definitions

BRAM	Block Random Access Memory
CPHA	Clock Phase
CPOL	Clock Polarity
EVB	Evaluation Board
FPGA	Field-Programmable Gate Array
GPIO	General Purpose Input/Output
HDL	Hardware Definition Language
IOB	Input/Output Block
MCU	Micro Controller Unit
MISO	Master-In Slave Out
MOSI	Master-out Slave in
nRST	Reset, active low
nSLEEP	Sleep, active low
NVM	Non-Volatile Memory
OCF	Open Core Protocol
OTP	One-Time Programmable on chip NVM
PLL	Phase Locked Loop
PMOD	Peripheral Module
POR	Power-On Reset
QSPI	Quad Serial Peripheral Interface
RTL	Register-Transfer Level
SPI	Serial Peripheral Interface
SCK	Serial Clock
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VDD	Supply Voltage
VDDC	Core Supply Voltage
VDDIO	Input/Output Supply Voltage

11. References

Download our free ForgeFPGA Designer software and follow the steps in the *ForgeFPGA Workshop User Guide* [1]. Refer to [2] for the datasheets, user manuals and guides. Renesas provides a complete library of application notes [2] featuring design examples, as well as explanations of features and blocks within the Renesas IC. Visit the [Renesas ForgeFPGA product page](#) to download the following:

- [1] [Go Configure Software Hub](#) (for the software download, go to the **Download** tab; for the *ForgeFPGA Workshop User Guide*, go to the **Documentation** tab)
- [2] [Product Selector: ForgeFPGA Low-Density FPGAs](#) (select the product; for datasheets, user manuals and guides, and application notes, go to the **Documentation** tab; for individual socket card user manuals and the *Go Configure Development Board R1.0 Manual*, go to the **Boards & Kits** tab)
- [3] [ForgeFPGA Advanced Development Board R1.2 User Manual \(ForgeFPGA Deluxe Development Board User Manual\)](#)
- [4] [ForgeFPGA Socket Adapter #1 User Manual](#)
- [5] [ForgeFPGA Evaluation Board R2.0 User Manual](#)

12. Revision History

Revision	Date	Description
2.51	Sep 10, 2026	- Added SPI mode command information to section 7.2 MCU Configuration Mode in SLG47912 and SLG47920/21 - Updated information on ForgeFPGA socket cards - Added more terms to section 10 Terms and Definitions - Updated section 11 References - Rearranged subsections in section 3 ForgeFPGA Development Platform - Updated QSPI Boot control information in section 5.1.3 QSPI Boot Control
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 Definitions
2.4	Dec 12, 2025	Added information on: - Program External Flash with a third-party programmer - Go Configure Development Board - Go Configure 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 section 7.1 - 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 02, 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 03, 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.