

AN-SL2001-07

Programming Guide for SL2001 & SL2002

Introduction

This Application Note provides instructions for programming the Non-Volatile Memory (NVM) and Volatile Memory (Registers) for SL2001 and SL2002 ICs. SL2001 and SL2002 both have NVM, supporting up to 3 times programmability as well as internal registers which can be used to program various parameters of the ICs. This is achieved by using the standard I2C interface of SL2001 and SL2002 and allows real time programmability during their operation.

This application note provides instructions for reading and writing to the volatile and non-volatile sections of the memory blocks using both the Silanna Semiconductor's Graphical User Interface (GUI) and I2C dongle [1], as well as using any other standard I2C interface.

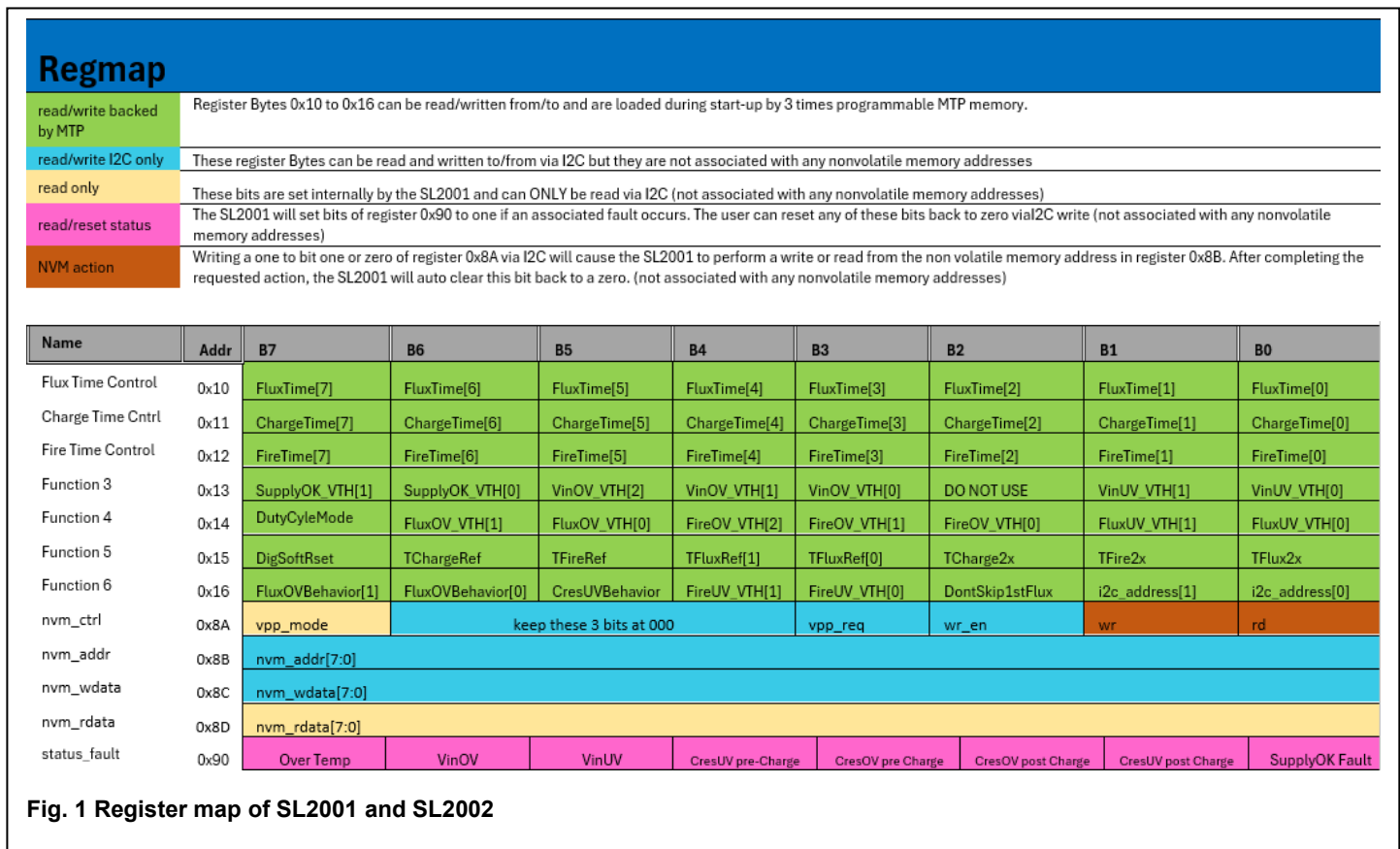
Contents

Introduction.....	1
High Level Register Map	1
NVM Programming	2

NVM Programming using GUI	2
Example 1:	3
Example 2:	4
Example 3:	4
NVM Programming using I2C	4
How to Read from the MTP Memory	5
How to Program MTP	5
Volatile Memory (Registers)	6
I2C Protocol Specifications.....	7
I2C Protocol for Write	7
I2C Protocol for Read	7
Summary and Conclusions.....	7
References.....	7

High Level Register Map

The high-level register map of SL2001 and SL2002 ICs are shown below. For detailed description of various registers,



please use the Silanna GUI (Multi Comm tab) and follow the instructions in the GUI application note [2].

NVM Programming

Registers 0x10 through 0x16 (Fig. 1) can be set via SL2001 and SL2002's three-time programmable (MTP-Multiple Time Programmable) ROM, which will be loaded to those corresponding registers (volatile memory) when the part starts-up. SL2001 and SL2002 also contain 219 additional One Time Programmable (OTP) Bytes in memory addresses (0x17-0x5F and 0x67-0xAF and 0xB7- 0xFF) that users can use as they wish. These additional Bytes of memory have no direct impact on product operation and can be loaded with any information, for example board or laser specific information that a microcontroller could use to modify register settings during operation due to temperature or other variations.

One example for usage of the 219 Bytes of OTP would be to store board specific information about the optimal Flux Time Control (Register 0x10) codes for various laser temperatures and/or other environmental variants in order to achieve one or several desired Light power profiles. During operation, the temperature could then be sensed near to the laser, and the code could then be read back via I2C for the memory address associated with that temperature, such that the code in that memory address could then be written via I2C to register 0x10 to modify the Flux timing.

The Graphical User Interface (GUI) [2] is a computer program that allows customers to configure SL2001 and SL2002 ICs through I2C programming for their target applications. GUI.exe is an executable file that opens I2C Communicator program that enables changing various parameters of the SL2001 and SL2002 ICs and when they are executed, the chip is programmed through the I2C dongle connected to the USB port of the computer [1]. For more details, please review the GUI application note [2] which introduces various features and functionalities of the GUI.

There are several key sections in this Application note. The first section provides step-by-step instructions on reading and writing to the NVM memory of SL2001 and SL2002 using the GUI. The second section provides instructions on how to do the same without using the GUI (directly interfacing via I2C), including instructions for writing and reading from the volatile memories, also known as the registers.

NVM Programming using GUI

This section provides step-by-step instructions on programming the NVM of SL2001/ SL2002 using the GUI.

Note: Vin must be set to 7.75V for programming the NVM. VDD needs to be separated from VIN and biased to 5V.

1. After turning on the board and starting the GUI (see the Quick Start Guide [1] for details), enter the customer password (1933537937) and press the Startup Button (red box in Fig.2).



Fig. 2 GUI's Setup screen showing the Startup button in a red box

2. Click on the NVM Comm tab of the GUI. Type in the Start and Stop hex addresses (e.g. 0x10) for the OTP Read Range as shown in the red box of Fig.3.

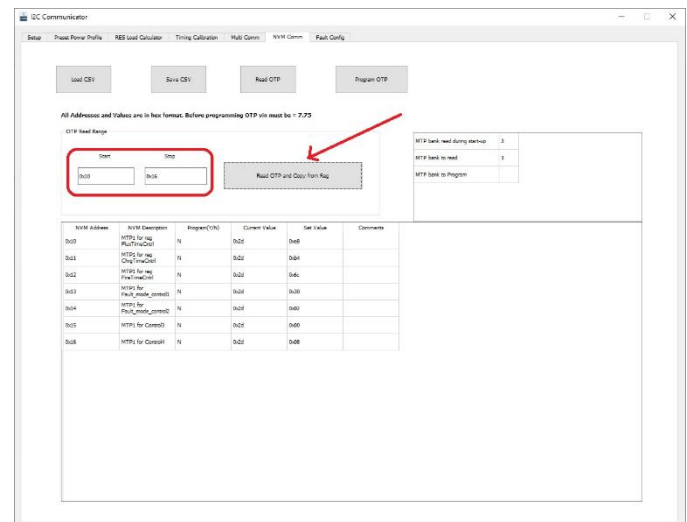


Fig. 3 GUI's NVM Comm screen showing how to read a range of values from the OTP memory

3. Press the button labelled "Read OTP and Copy from Reg" (shown with a red arrow in Fig.3), which reads and displays the existing values of the selected NVM addresses in Current Value column and the

corresponding register values in Set Value column. If you get all the values shown as 0xff, you may need to re-type in the password (i.e. **1933537937**) in the Setup page and press the Startup button again, or you may have selected a register which the user doesn't have the permission to read or write to.

4. If you can successfully read the values from the selected range of the OTP memory, you should see a table of NVM memory addresses with their corresponding values as shown with a red box in Fig.4.

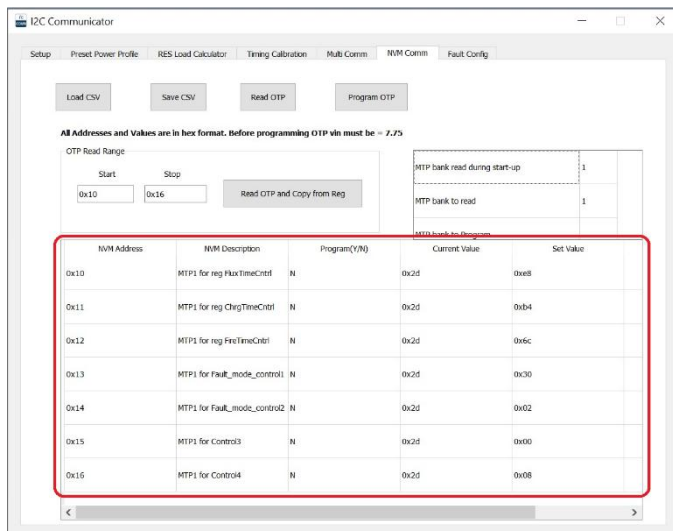


Fig. 4 GUI's NVM Comm screen showing the selected range of values from the OTP memory

5. There are three fields of information shown in the right side of the NVM Comm page, as marked with red box in Fig.5 below. **MTP bank read during start-up** reads from the OTP directly and tells the status of MTP usage. Each time the NVM Comm tab is selected, this field will automatically reflect the latest information on which MTP bank is being read during the start-up of this IC. When a new MTP bank is programmed, this field will also be auto updated to show that (read only).

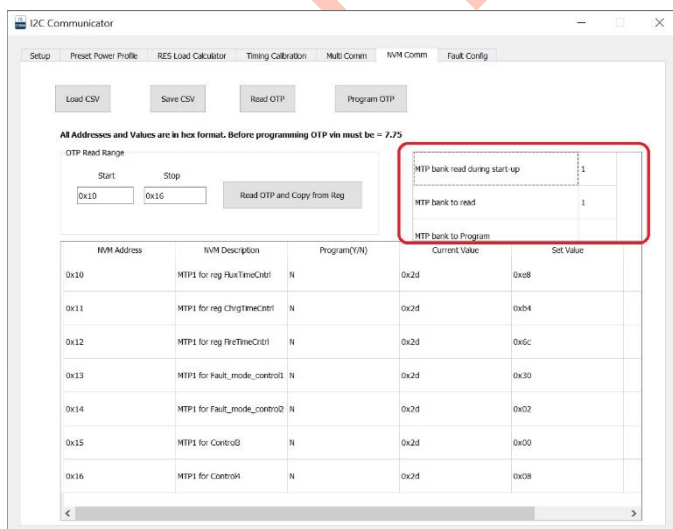


Fig. 5 GUI's NVM Comm screen showing MTP Bank information

6. The next field is **MTP bank to read**. This is a read and write field which specifies which MTP bank to read to display the values from the NVM memory addresses as demonstrated in step 4. The user can change this field to read from different MTP banks.
7. The last field is **MTP bank to Program**, which is a write field. This is the field where the user will need to pay extra attention. By default, when **MTP bank to read** field is set to n, **MTP bank to Program** will be automatically set to be n+1 (GUI assumes user is done with the current MTP bank and ready to move to the next MTP bank for programming). In cases where the user wishes to program only several bytes of the same MTP bank at a time, the user needs to manually alter this field back to the same MTP bank number where they want to keep programming to different memory addresses. The GUI will show a pop-up window to confirm this, giving the user a 2nd chance if they want to change their mind.

In the next section of this application note, three examples of reading or writing to the NVM MTP (3 times) are provided.

Example 1:

This example provides step-by-step instructions on how to program only 1 byte of data to MTP.

1. Please follow step 1 to 4 in the previous section and specify the OTP Read Range to include the NVM address where the user would like to program 1 byte of data.
2. In this example we will make changes to the NVM address byte 0x11 which is used for the duration of Charge Time Control (see Fig.1).
3. Once you select the OTP read range to include the memory byte of your choice and press "Read OTP and Copy from Reg" button, a table showing those memory block values should be displayed as previously shown in Fig. 4 (also in Fig.6 below).

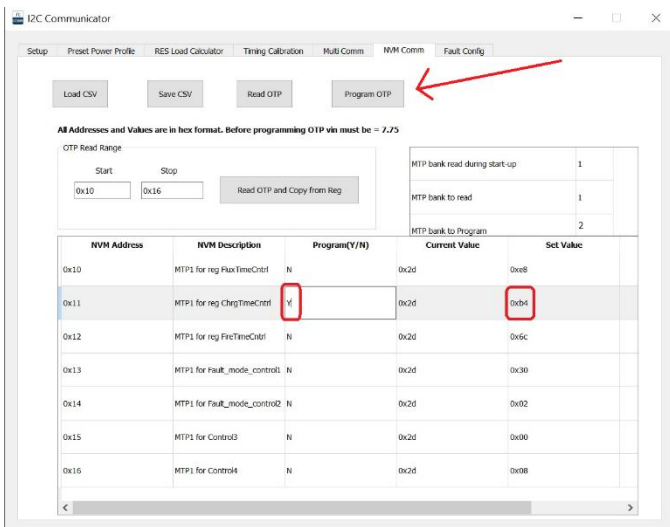


Fig. 6 Example showing how to program 1 byte of NVM Bank 2

4. Then change the “Program (Y/N)” column from “N” to “Y” and type in the new value that you want to set in “Set Value” column of the desired memory byte as shown with red boxes in Fig.6.

Before moving to the next step, please ensure all addresses and values are correct and are in hex format. Also, make sure you are programming this 1 byte of data in the right MTP bank (MTP bank to Program field).

Please Note: For programming the NVM, Vin must be set to 7.75V. VDD needs to be separated from VIN and biased to 5V.

5. After that press the “Program OTP” button (shown with a red arrow in Fig.6), and that should change the current value to the set value in the selected memory byte (0x11 in this case).

6. To confirm the information you just programmed in this example, you will also need to change the field “MTP bank to read” to “2” and then click on “Read OTP”.

Example 2:

This example walkthroughs the steps needed to migrate one entire MTP bank of information to another MTP bank.

1. Please follow step 1 in the previous section and specify the OTP Read Range from 0x10 to 0x16 to include the target NVM address bytes of the current MTP bank (MTP bank to read field).

2. Press “Read OTP and Copy from Reg” button, which will populate the lines in the table. The Current Value column displays the current values in the MTP corresponding to register 0x10 to 0x16. The Set Value column displays the current register values, which reflects the present behavior of the IC. These values will be later programmed into the next MTP bank.

3. In order to migrate the current values of the entire MTP bank to another, users need to modify the entire Program (Y/N) column to Y (as shown in the previous example).

4. Make sure the MTP bank to Program field is updated to the right destination MTP bank (commonly n+1 where n is the current MTP bank).

5. Finally press the Program OTP button and confirm your selection. This should copy over all the values from the current MTP bank to the new MTP bank.

6. To confirm the change, press “Read OTP and Copy from Reg” button (make sure MTP bank to read is set to the new MTP bank).

Example 3:

SL2001 contains 219 additional One Time Programmable (OTP) Bytes in memory addresses (0x17-0x5F and 0x67-0xAF and 0xB7- 0xFF) that users can use as they wish. This example shows how to program these additional OTP bytes that SL2001 or SL2002 provide for customer use.

There are two ways to program these additional OTP bytes. First is to follow Example 1 above and program each byte at a time with the desired information. Please make sure the NVM address and Set Values are entered correctly before programming the OTP. Second is to use a .CSV file with right information for the corresponding bytes and load the CSV file into the GUI NVM Comm tab by pressing the Load CSV button on the top left corner. Please ensure the password 1933537937 is entered in the setup tab to unlock the NVM programming as mentioned before. The method of using a .CSV file can also be used when making a large amount of data changes in the existing or a new MTP. The CSV file should have the right format / column reference. An example is shown below in Fig. 7.

	A	B	C	D	E	F
	NVM Address	NVM Description	Program(Y/N)	Current Value	Set Value	Comments
1	0x10	MTP for reg FluxTimeCtrl	Y		0xe4	Will go to target MTP set in GUI
2	0x11	MTP for reg ChrgTimeCtrl	Y		0xe4	Will go to target MTP set in GUI
3	0x12	MTP for reg FrcTimeCtrl	Y		0xe4	Will go to target MTP set in GUI
4	0x13	MTP for Fault_mode_control1	Y		0x00	Will go to target MTP set in GUI
5	0x14	MTP for Fault_mode_control2	Y		0x00	Will go to target MTP set in GUI
6	0x15	MTP for Control3	Y		0x00	Will go to target MTP set in GUI
7	0x16	MTP for Control4	Y		0x66	Will go to target MTP set in GUI
8	0x17	OTP for customer usage	Y		0x01	
9	0x18	OTP for customer usage	Y		0x10	
10	0x19	OTP for customer usage	Y		0xF6	

Fig. 7 Example of an Excel .CSV file for programming the existing or new MTP bank

NVM Programming using I2C

This section describes how to read or write to SL2001 and SL2002 NVM without using the Silanna GUI. The first subsection explains how to read from the MTP

memory addresses, and the second subsection provides details on how to write to the MTP memory addresses.

How to Read from the MTP Memory

This section describes how to read from the MTP addresses which are part of the NVM.

The initial conditions are set with VDD between 4.5V to 5.5V. Vin is set at VDD or could be left floating. I2C permission status needs to be unlocked with the password sequence provided in the table below.

Please note: The user must write the correct sequence of data in order to unlock the password. Writing any other sequence causes the password to lock. If the password is locked, then all writes will fail (except the password itself) and all reads will result in a value of 0xAA.

Step	I2C cmd	Device Address	Register Address	Data	Notes
1	Write	0x09	0x81	0xF7	
2	Write	0x09	0x81	0x49	
3	Write	0x09	0x81	0xC1	Unlocked

The following example reads Flux DAC code from MTP1 (address 0x10) with data 0xB5

Step	I2C cmd	Device Address	Register Address	Data	Notes
1	Write	0x09	0x8B (NVM Address)	0x01	Set address to fault_en
2	Write	0x09	0x8A (NVM Control)	0x01	Set read = 1
3	read	0x09	0x10 (NVM Read Data)	0xB5	NVM read data=0xB5

As shown in the above table, in order to read from the NVM, the user needs to know the following NVM register addresses:

Register Name	Address
I2C Password	0x81
NVM Control	0x8A
NVM Address	0x8B
NVM Read Data	0x8D

How to Program MTP

The following section describes how to send data over I2C and program the MTP NVM memory of SL2001 and SL2002.

The initial conditions are set with VDD between 4.5V to 5.5V. Vin is set at VDD or could be left floating. I2C permission status needs to be unlocked with the password sequence provided in the table below.

Please note: The user must write the correct sequence of data in order to unlock the password. Writing any other sequence causes the password to lock. User will need to start from the first password byte again. Before the correct password combination is detected, any readback of Register address 0x81, will be 0xFF. Only when the correct password is detected, its readback value will be changed to 0xAA. Once the password is correctly written (readback value of register address 0x81 equals 0xAA), the password will be unlocked and not need to be re-written until it is shutdown. Alternatively, to shut the part down, the user could enter any data to register address 0x81 to lock it again.

Step	I2C cmd	Device Address	Register Address	Data	Notes
1	Write	0x09	0x81	0xF7	
2	Write	0x09	0x81	0x49	
3	Write	0x09	0x81	0xC1	Unlocked

Programming MTP requires writing to the MTP shadow registers (registers address 0x10 to 0x16) as a first step. Then using the str_all bits, the values from the shadow registers are transferred to the MTP bank as shown in the table below. Users can load all the data into register address 0x10-0x16 and program it into MTP altogether. Users can also take steps to program them one or several bytes at a time. Please keep in mind, for the MTP register bytes or bits user doesn't want to program, user need to keep them as 0.

Step	I2C cmd	Device Address	Register Address	Data	Notes
1	Write	0x09	0x10 (Flux Time)	0xE8 (example)	Target data to Target REG address 0x10-0x16 (keep uncertain registers value to 0, to prevent double programming)
2	Write	0x09	0x11 (Charge Time)	0xB4 (example)	
3	...	...	...	...	Continue writing all the

					data in the shadow registers
4					Set Vin = 7.75V
5	Read	0x09	0x88	0xxx	Confirm which MTP is already occupied by reading the register 0x88 (bit 7 to 5)
6	Write	0x09	0x88 (str_all)	0xxx	0x02, 0x04 or 0x08 for writing to the 1 st , 2 nd or 3 rd MTP bank, respectively. This is a self-clearing bit.

Please note, it is not recommended to use byte-by-byte approach for programming the MTP addresses directly in the NVM memory banks as it will require updating several additional register bits to properly initiate the startup memory bank location. Flipping the bits in str_all register (0x88 bit 3 to 1) automatically takes care of those updates.

Also note, NVM addresses 0x10 to 0x16 are used for MTP1, 0x60 to 0x66 are used for MTP2 and 0xB0 to 0xB6 are used for MTP3. Those are the 3 banks of MTPs that sharing shadow register 0x10 to 0x16. Please do not program any data that is not relevant to MTP into these NVM addresses. As shown above, please use str_all register for programming instead for writing directly to NVM addresses.

Volatile Memory (Registers)

Reading or Writing to the commonly used volatile memory (e.g. 0x10 to 0x16) doesn't need any password. Please note, there are registers which are internal to Silanna use and are not accessible to the customers/ users outside Silanna. Silanna application note AN-SL2001-01 (GUI Application Note) provides details on how to use the GUI's Multi Comm tab to read or write to the registers. That section is copied below for the customer convenience.

In the Multi Comm tab of the GUI (Fig. 8) the 'Read from I2C' button allows reading from the accessible registers identified between the 'Start' and 'Stop' text boxes.

These two boxes expect Hex values of the target registers. After the registers are read and present in the Multi Comm table, there are 2 methods to edit the register values. If any register is selected in the table, the 'Edit' button pops up a

Single Register Edit Page which provides detailed information about the bit options in that register. Users can change any bit values and click on "write" to write to the IC. Rather than use the "Edit" button to pop up the Single Register Edit Page, users can also just modify values in the Multi Comm table for any registers, then click "Execute" to perform all read and write functions in the table. Register contents can also be edited by double clicking value on the 'Set Value' column but the user needs to change 'Read/Write' column to W to be able to do so.

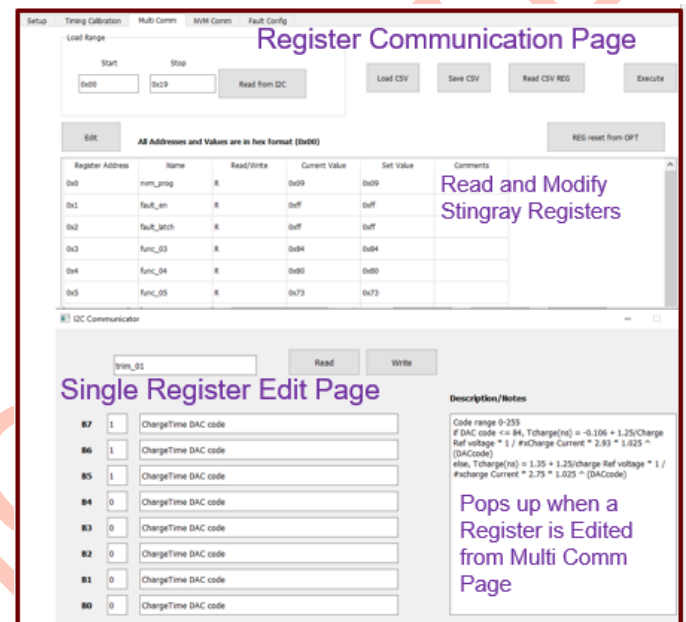


Fig.8 The Multi Comm Tab from the GUI

As an alternative to using the "Read from I2C" button to load and read a range of registers, the user can also use a CSV file to load a series of register reads and writes. Once the Multi Comm Table is present, the user can click the "Save CSV" button to save all of the rows to a csv file. Then, the user can edit that csv file using a text editor or Excel and load the modified register settings back into the GUI by pressing the "Load CSV" button. by pressing the 'Load CSV' button. Fig. 9 shows an example .CSV file for loading in the GUI's multi comm tab.

C12		X ✓ fX					
A	B	C	D	E	F	G	H
1	Register A Name	Read/Write	Current Value	Set Value	Comments		
2	0x10 Flux Time Control	R	0xce	0xce			
3	0x11 Charge Time Cntrl	R	0xe0	0xe0			
4	0x12 Fire Time Control	R	0x85	0x85			
5	0x13 Function 3	R	0x09	0x09			
6	0x14 Function 4	R	0x09	0x09			
7	0x15 Function 5	R	0x10	0x10			
8	0x16 Function 6	R	0x2e	0x2e			
9							
10							

Fig.9 Example of a CSV file to Load from the GUI

Once the target register values are modified, the user needs to click on the 'Execute' button to write to the registers

through the I2C connection. Please note, the GUI automatically reads the registers when transferring from one tab to another.

The “Read CSV Reg” button provides the ability to read all of the register values for registers present in the Multi Comm table without performing any writes.

The “REG reset from OTP” function will read the values from the non-volatile memory into the Multi Comm table (rather than the values in the volatile registers).

I2C Protocol Specifications

This section describes I2C protocol specification and the byte structure for reading and writing. This gives an idea on how fast the data can be read or written to SL2001 and SL2002 ICs using I2C protocol. The SL2001 and SL2002 ICs’ standard I2C protocol allow maximum SCL (Clock) speed of 1 MHz. It was designed to support 100kHz, 400kHz, and 1Mhz SCL speed. I2C is a 2-bit protocol. SDA is both an input and an output. One key advantage of I2C communication protocol is it is very low pin count, making the design and layout of the board quite simple and efficient.

I2C Protocol for Write

I2C writes receive binary data from the pins and stores them into bytes. The first byte is the devaddr (device address) and if the byte matches this device, then the message is for this device; otherwise, the message is discarded. The second received byte is the write data address. This is the address where the first data byte is stored. The next byte is the data byte which is stored at the write address. The host will send a Stop bit to end the transaction.

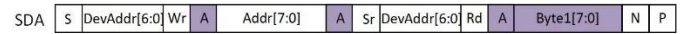
Each byte needs an acknowledgement (ACK/NAK) bit. Therefore, write takes 3*9 bits + S (stop) bit + P (end) bit + idle time (29 bits + idle time). Assuming the SCL is Running at 1Mhz, this 1 byte write operation takes $29 * 1\mu s + \text{idle time} = 29\mu s + \text{idle time}$. Typically, the idle time can be as little as 2-bit equivalent time, making the total write time of $31\mu s$ for 1 byte of data.



I2C Protocol for Read

I2C reads data from the internal registers or NVM and transmits the I2C data to the I2C pins. The first byte is the devaddr (device address) and if the byte matches this device, then the message is for this device; otherwise, the message is discarded. The second received byte is the

read data address. This is the address where the first data byte is read from. The next byte is the devaddr byte followed by 1 data read bytes. After the data byte has been read, the host will transmit a NACK and a Stop bit to signal the end of the I2C read transaction.



One byte of read requires $4*9 + S + P + \text{idle time}$ or a total of $40\mu s$.

To summarize, the three Write transaction bytes are devaddr/write, address and data to write.

The four Read transaction bytes are devaddr/write, address, devaddr/read and read data.

Summary and Conclusions

In summary, SL2001 and SL2002 provide standard I2C interface to program the volatile and non-volatile memories built into them. This application note provides details on how to program the memory section using the GUI or any other standard I2C interface for changing various features and functionalities of these ICs. For more details, please contact Silanna's Application team to schedule a training.

References

1. Quick Start Guide – provides step-by-step instructions on how to set up the GUI software and initialize the evaluation board hardware
2. GUI Application Note – provides the details of the GUI software and its various features and functionalities

Revision History

Revision	Date	Author	
1.0	12/17/2025	AZ	