How to use the L9962 PC software GUI for L9962 based evaluation

Introduction

This document covers various aspects of the L9962 PC software GUI (Graphical User Interface) and the features that can be used to interact with the evaluation board.

The L9962 GUI utilizes the PC serial communication port to interface with the STM32 microcontroller. The STM32 then communicates directly with the L9962 via I2C.

For technical details of the L9962 BMS device, the L9962 datasheet should be referenced.

1 L9962 software package

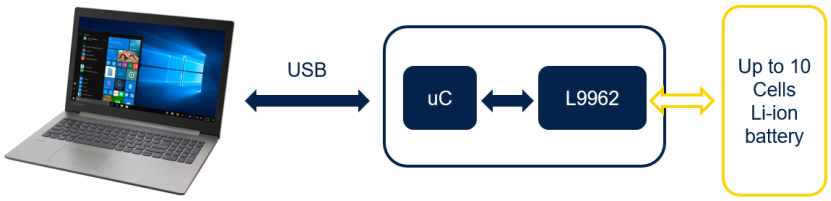
1.1 Package content

The L9962 software package includes the L9962 GUI .exe installer, the application sample scripts to enable key parameters such as cell voltage, current and battery temperature acquisitions, and the binary code to flash the NUCLEO-G071RB STM32 Nucleo-64 development board. This board is used to control the EVL9962 device via I2C and GPIO and also to interface to the PC via the USB connection.

The software GUI exploits the features of the L9962 BMS IC.

Note: Please refer to the UM2951 on how to flash the NUCLEO-G071RBMCU board.

Figure 1. Board connection block diagram



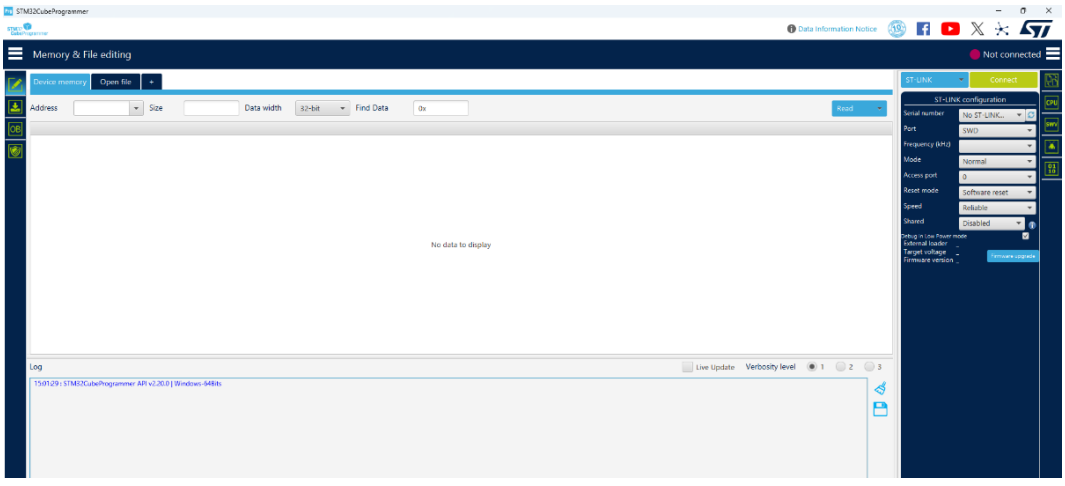
1.2 NUCLEO board firmware programming with STM32CubeProgrammer

1.2.1 STM32CubeProgrammer installation and connection

Install STM32CubeProgrammer from

STM32CubeProg | Software - STMicroelectronics

Figure 2. . STM32CubeProgrammer



1.2.2 Connect the NUCLEO board to the PC

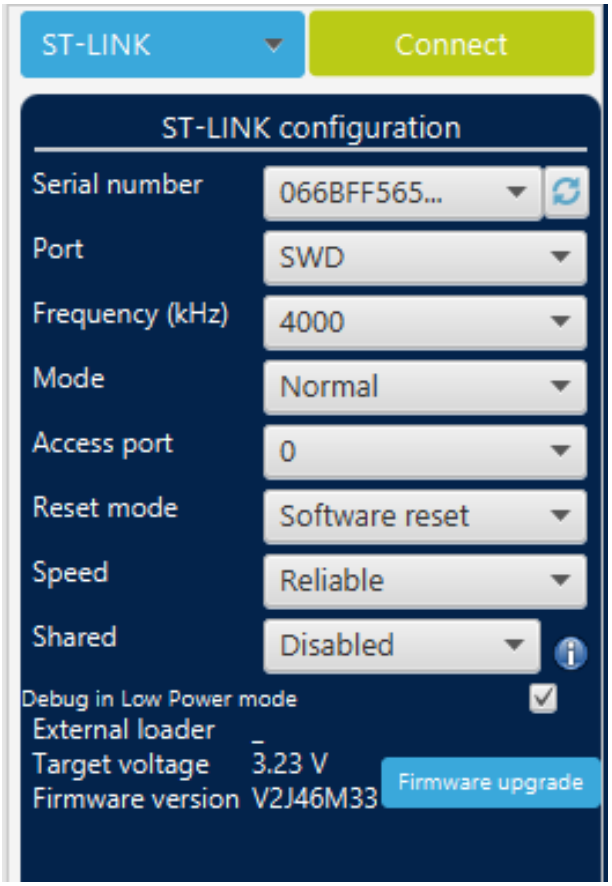
- Use the USB connector linked to the on-board **ST-LINK** (usually labeled **USB ST-LINK**).
- Make sure that:
  - The board is powered (**LD3 / Power LED ON**).
  - No other tool (IDE/debugger) is currently connected to the ST-LINK

1.2.3 ST-LINK configuration and connection

In the right panel of STM32CubeProgrammer, configure ST-LINK as follows (these are good defaults for most NUCLEO boards):

- Serial number: select your ST-LINK (e.g. 066BFF...).
- Port: SWD
- Frequency (kHz): 4000 (4 MHz). If you see connection issues, try 1000 or 500.
- Mode: Normal
- Access port: 0
- Reset mode: Software
- Speed: Reliable
- Shared: Disabled

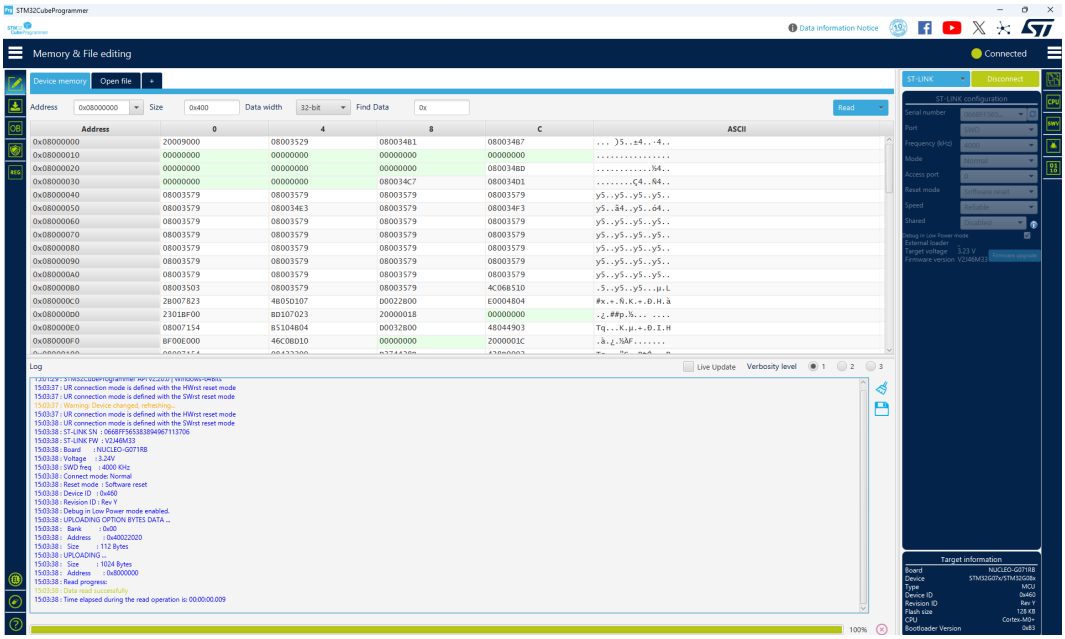
Figure 3. ST-LINK configuration



Click on ‘Connect’ and check on the right panel it switches from 

to 

Figure 4. **Connect flag**



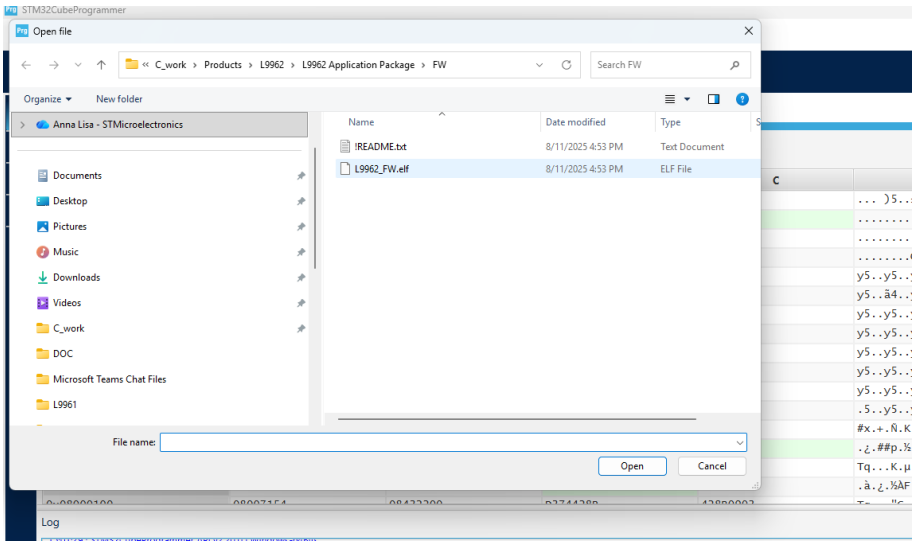
1.2.4 Loading the firmware file (.elf)

In STM32CubeProgrammer, click **Open file** in the top-left of the main window (or the small folder icon in the left toolbar).

A standard file browser window appears (see figure).

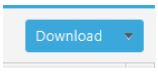
Navigate to the folder that contains the firmware delivered with the application package.

Figure 5. **load .elf file**



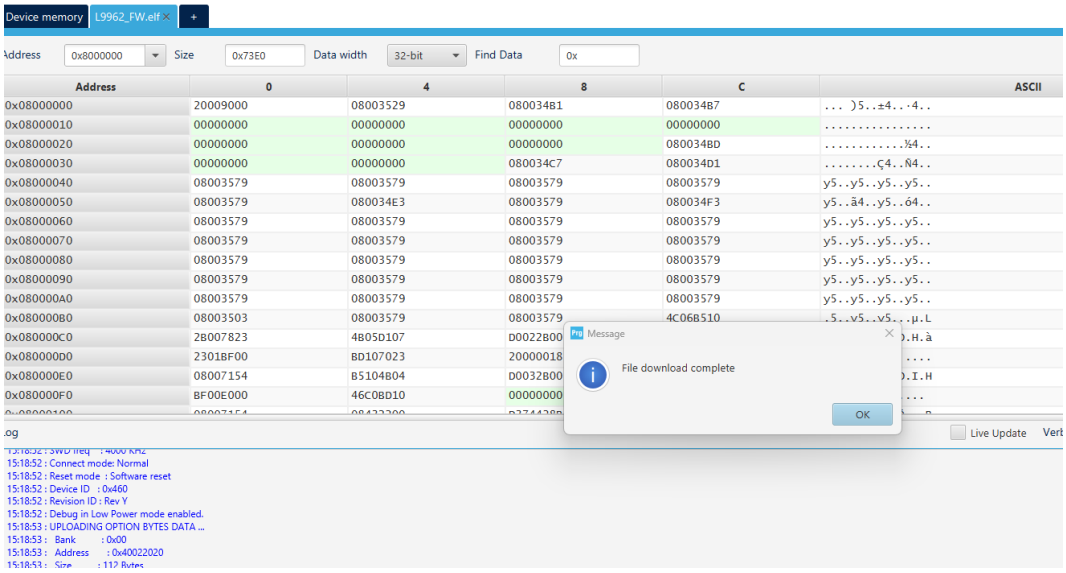
1. Select the firmware file **L9962_FW.elf** in the list.
2. Click **Open**.

1.2.5 Programming the microcontroller

You can then proceed clicking on 

After pressing **Download** / **Start Programming**, STM32CubeProgrammer programs the flash.
When the operation is finished, a pop-up message appears:

Figure 6. Check elf file has been loaded

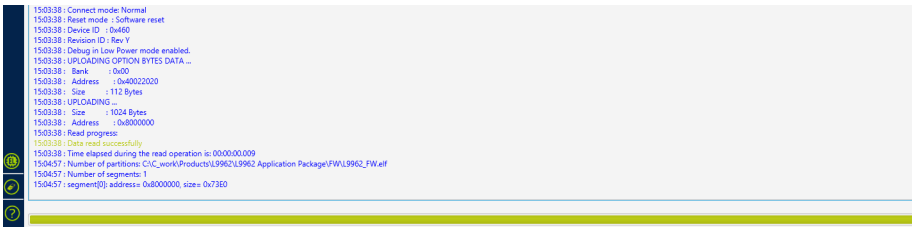


Click **OK** to close the message.

1.2.6 Programming the microcontroller

Check the ELF file has been loaded (the memory view is filled as in the figure).

Figure 7. Check elf file has been loaded

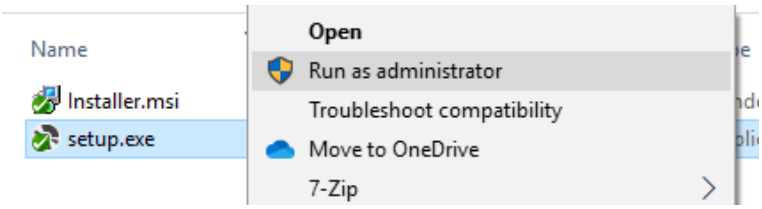


2 GUI installation

The GUI software comes packaged as a standard Windows installer. The program must be installed as an administrator. Therefore, the user must have administrative rights to properly install this application. This can be verified by your local IT department.

To initiate installation as administrator, navigate to the setup.exe file, right-click, and select "Run as administrator" (see Figure 8).

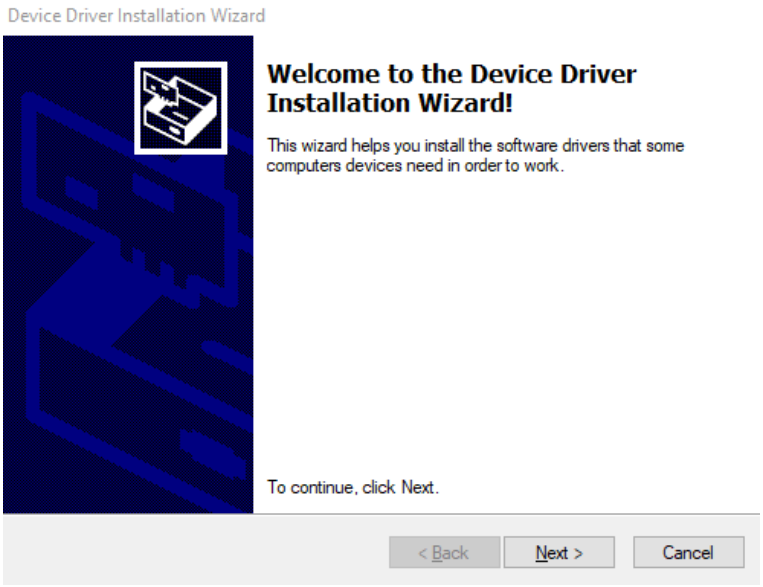
Figure 8. Run GUI installation as administrator



The GUI installer will then go through a series of dialog boxes. Unless there is a need to change one of the default settings, the default values can be used.

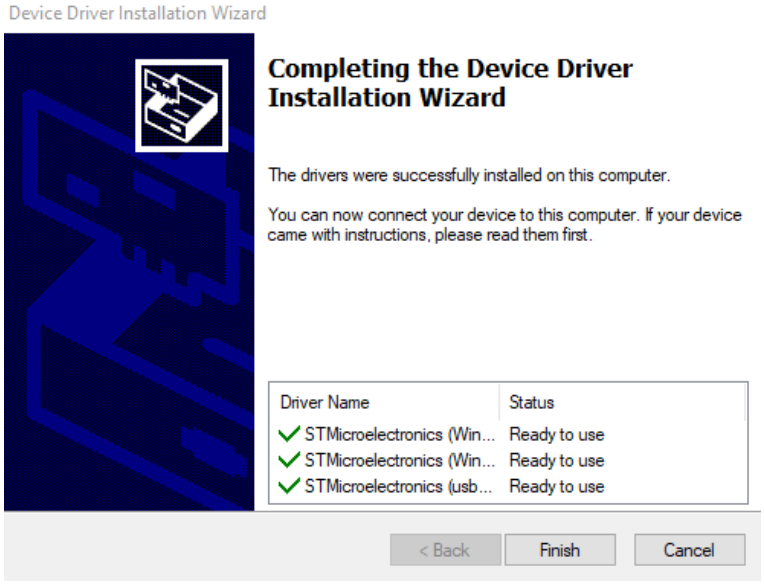
Following the installation of the GUI, the installer for the necessary device drivers will be launched, and the following window will appear (see Figure 9):

Figure 9. Device driver installation wizard



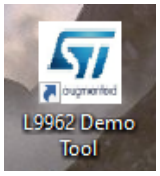
As with the GUI installer, select the default values to complete the installation, and then press the "Finish" button (see Figure 10):

Figure 10. **Successful installation of mandatory device drivers**



Once installation is complete, an L9962 demo tool shortcut icon appears on the desktop (see Figure 11):

Figure 11. **L9962 GUI desktop shortcut**



Additionally, the GUI can be located using the Windows start menu (see Figure 12):

Figure 12. **L9962 GUI Windows start menu location**

3 GUI operation

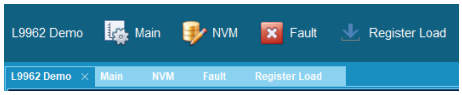
3.1 Overview

Figure 13. **L9962 evaluation GUI**

The L9962 GUI consists of five tabs that can be selected at the top left of the GUI (see Figure 14). These tabs are reviewed in more detail later in this document. The tabs are as follows:

- L9962 demo
- Main
- NVM
- Fault
- Register load

Figure 14. **L9962 GUI tab selections**



The tabs can be arranged in any order the user would like by grabbing a tab (left-click and hold) and dragging it to the desired window location (see Figure 15). It is possible to display all five tabs at one time (see Figure 16).

Figure 15. **Dragging a tab to the top window location**

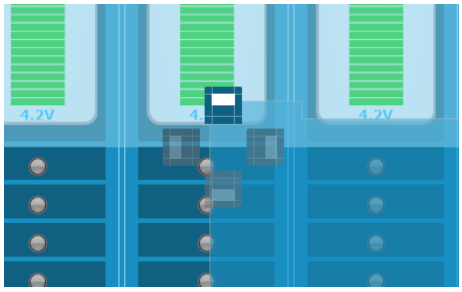


Figure 16. **All five tabs are displayed at once**

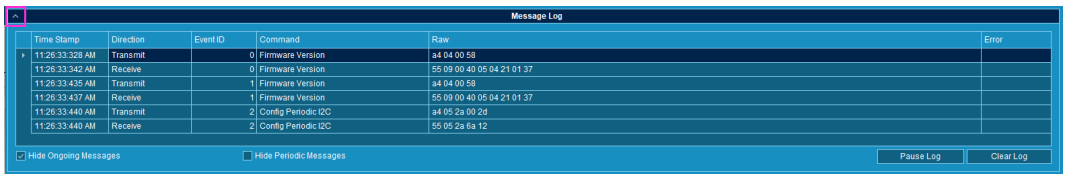
There are three sections of the GUI that can remain visible across all six tabs (highlighted above in Figure 13):

- Message log - can be expanded/collapsed using the caret.
- Board connection status - always visible.
- I^2C read/write array - can be expanded/collapsed using the caret.

3.1.1 Message log

The message log provides a history of the command communication between the STM32G071 MCU and the PC. It can be expanded or collapsed by pressing the caret shown in Figure 17:

Figure 17. **L9962 GUI message log**



The fields in the message log are as follows:

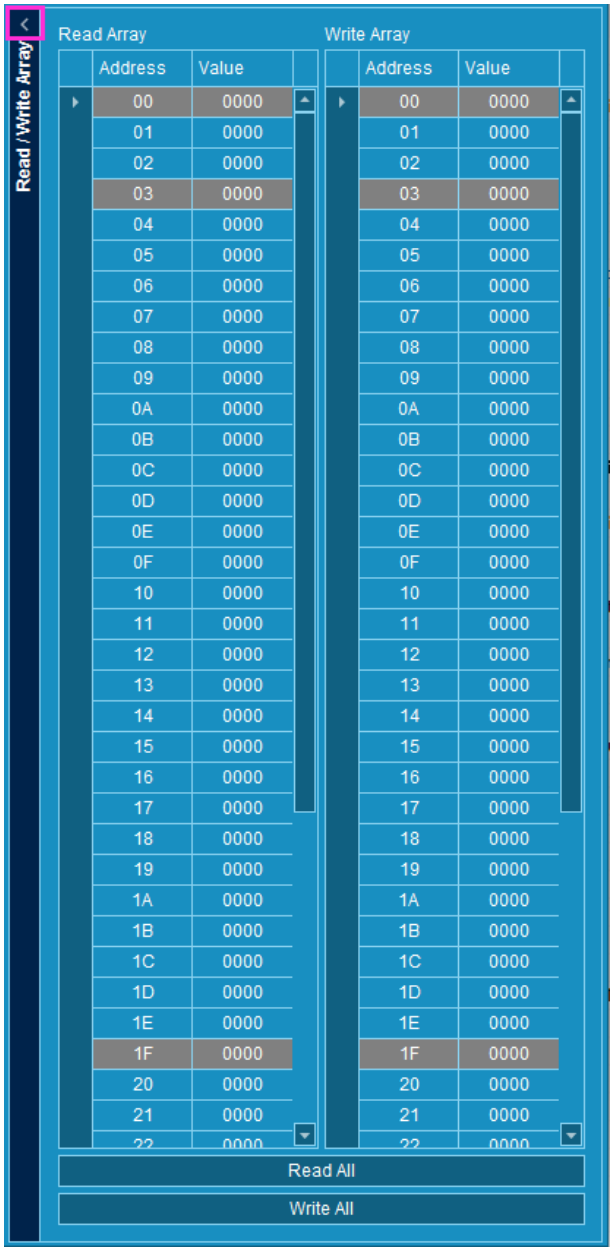
- Time stamp - The time that the transaction occurred
- Direction - Whether the message was transmitted or received
- Event ID - An incremental number indicating the transaction number. For each transmit ID, there should be a matching receive ID
- Command - The command that was sent or is being responded to
- Raw - The raw communication between the STM32/L9962 and the PC

Once the message log is expanded, the options of pausing and clearing of the log are available. Additionally, the ability to enable/disable the logging of ongoing or periodic messages is available.

3.1.2 I²C read/write array log

Similarly, the I²C read/write array log can be collapsed or expanded by pressing the caret shown in Figure 18:

Figure 18. I²C read/write array log



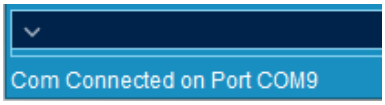
The user can also send read/write commands to all of the registers in the array log by pressing "Read All" or "Write All". The write data (value) field can be edited by double-clicking the text field (see Figure 19):

Figure 19. Editing the data field of register 0x0D in the write array

3.2 Connecting to the Nucleo board

As soon as the GUI is opened, the PC attempts to connect to the STM32 Nucleo board. If a valid COM port is found, and a connection is made successfully, the lower left corner of the GUI will show that the connection has been established (see Figure 20):

Figure 20. GUI connected to Nucleo board successfully



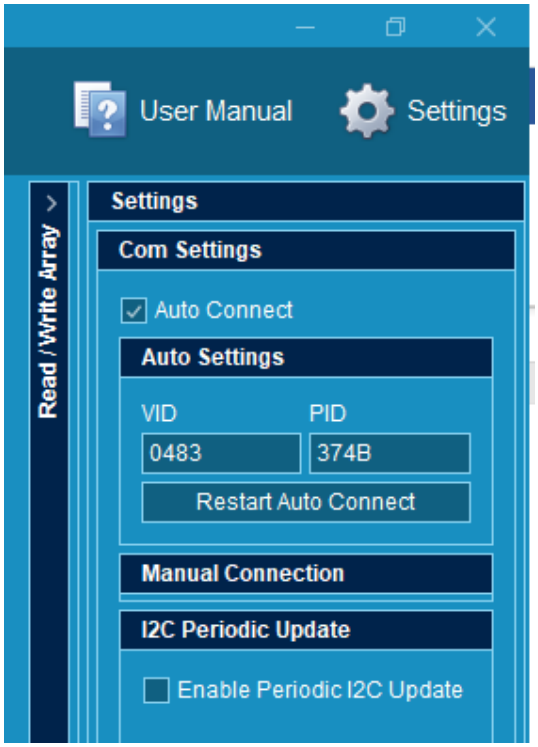
If a connection between the PC and the evaluation board is not established, the GUI will actively look for the device and display the following message (see Figure 21):

Figure 21. COM failure

In the event that the GUI cannot detect the board (and it has been confirmed it is powered on, plugged in, and all device drivers are installed), the user can adjust the COM setting manually.

Navigate to the settings menu, located in the upper right corner of the GUI (see Figure 22):

Figure 22. L9962 GUI settings window

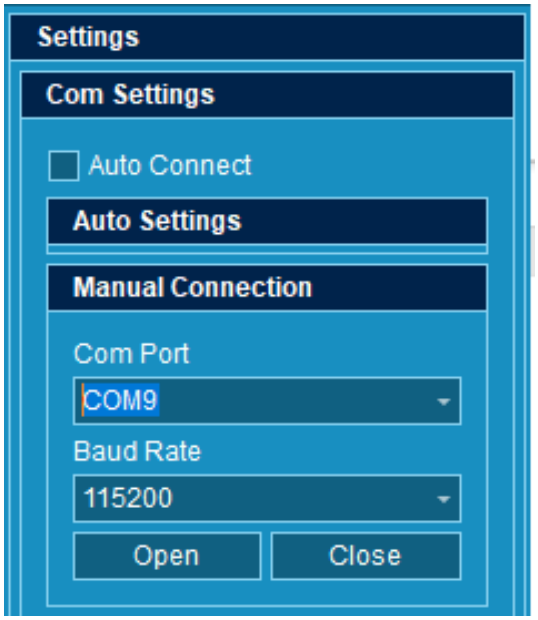


When the auto connect checkbox is enabled, the application is able to automatically detect when a unit is plugged in or removed. If the auto connect is not enabled, it is necessary to press the restart auto connect button.

The VID/PID values are the default values for the USB to serial interface on the STM32/L9962 evaluation board. There should be no need to change these values, but the option is available should the need arise.

If reconfiguration of the VID/PID values does not fix the communication problem, the port can be manually selected by turning off the auto connect feature (see Figure 23).

Figure 23. **Manual COM port selection**

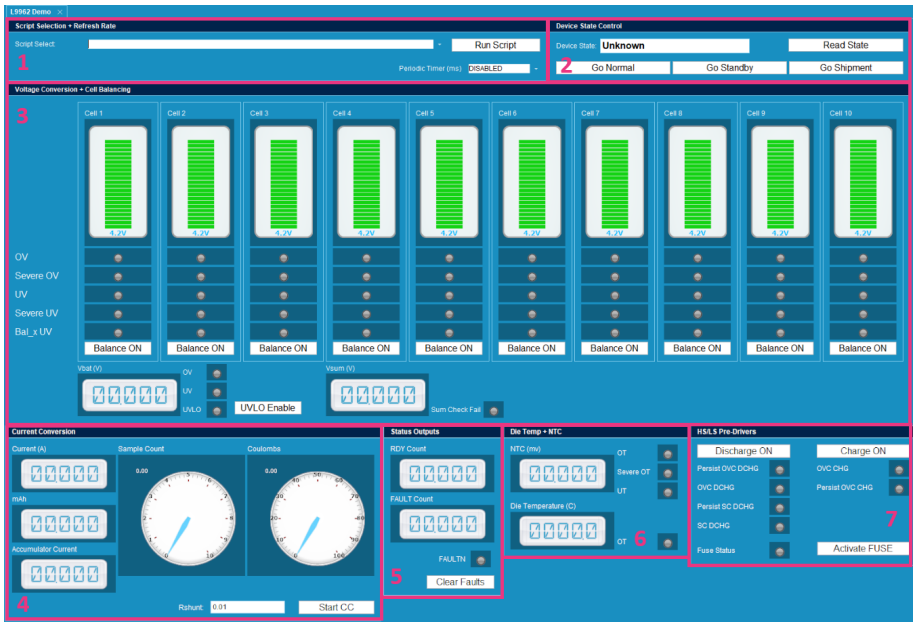


3.3 GUI tab overview

3.3.1 L9962 demo tab

When the GUI is launched, this tab is displayed by default (see Figure 24):

Figure 24. **L9962 demo tab display**



The L9962 demo tab is used to work with the device in an easy-to-visualize way. All aspects of the device are laid out in this tab, which consists of seven sections:

1. Script selection + demo tab refresh rate
2. Device state control
3. Voltage conversion and cell balancing
4. Current conversion
5. Status outputs

6. Die temp and NTC
7. HS/LS predrivers

3.3.1.1 *Script selection*

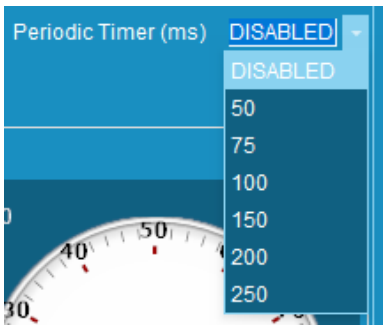
Figure 25. Script select drop-down box

The L9962 GUI is provided with a set of scripts that can be used to get the user up and running quickly. However, if the user wishes to create their own script, then Section 3.3.5 Register load tab should be referenced.

To access a script, press the belonging to the "script select" box (see Figure 25), and click on the desired script.

Next, change the periodic timer (see Figure 26) to "200 ms" for any of the sample scripts. This can be less for custom scripts depending on the timings selected.

Figure 26. L9962 demo tab periodic update rate selection



Following the selection of the periodic update rate, the selected script runs. Once the script has completed, the components on the tab begin to update per the script's instructions.

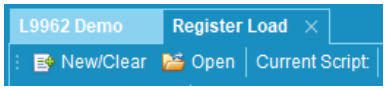
Example:

If the user selects the script titled "voltage acquisition Init - 10-Cell + VB + NTC.csv" and selects a periodic update of 200 ms, the components for cells, Vbat, Vsum, NTC, and die temperature (along with their associated fault LEDs) will all display their respective data at a 200 ms update rate.

Note: To have the scripts appear in the "script select" drop-down box, the file extension must be configured properly in the register load tab. This only needs to be done for first-time users - the settings will be retained for future.

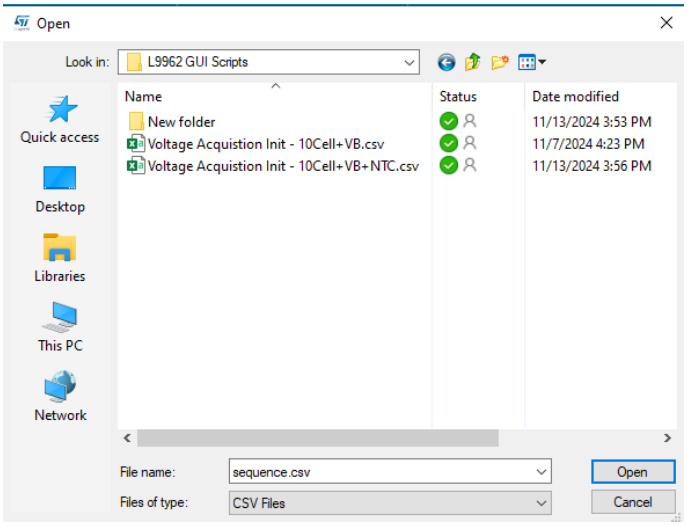
To do this, navigate to the register load tab and click "Open" (see Figure 27).

Figure 27. Register load file open



Next, navigate to the folder where the sample scripts (or custom scripts) are located, click on one of the scripts, and press "open" (see Figure 28).

Figure 28. **Script folder selection**



Now, the "current script" should show the file extension of the selected script.

Navigating back to the L9962 demo tab and pressing the for the script select box, it should now show the scripts located in the folder selected in the register load tab.

3.3.1.2 *Voltage conversion and cell balancing*

Figure 29. **Voltage conversion and cell balancing section**

The voltage conversion and cell balancing section of the L9962 demo tab (see Figure 29) allows the user to easily read the cell voltages and associated fault statuses. Additionally, the Vbat and Vsum voltages and associated fault statuses are also available.

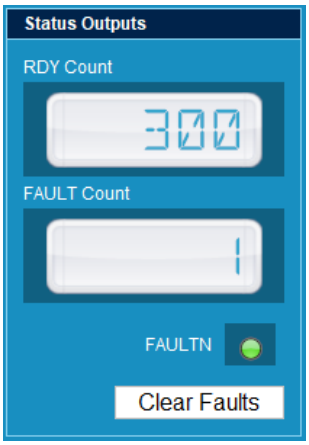
The fault statuses are indicated by LEDs. Each LED is associated with a specific I2C register bit. When no faults are present, all LEDs are illuminated in green (see Figure 30). In the event that one or more of the LEDs changes to red, this indicates that a fault is present (see Figure 31).

Figure 30. **No fault present**

Figure 31. **Fault present**

To clear the fault, first resolve the condition causing the fault, then press the "Clear Faults" button located in the "status outputs" section of the tab (see Figure 32).

Figure 32. Clear faults button location



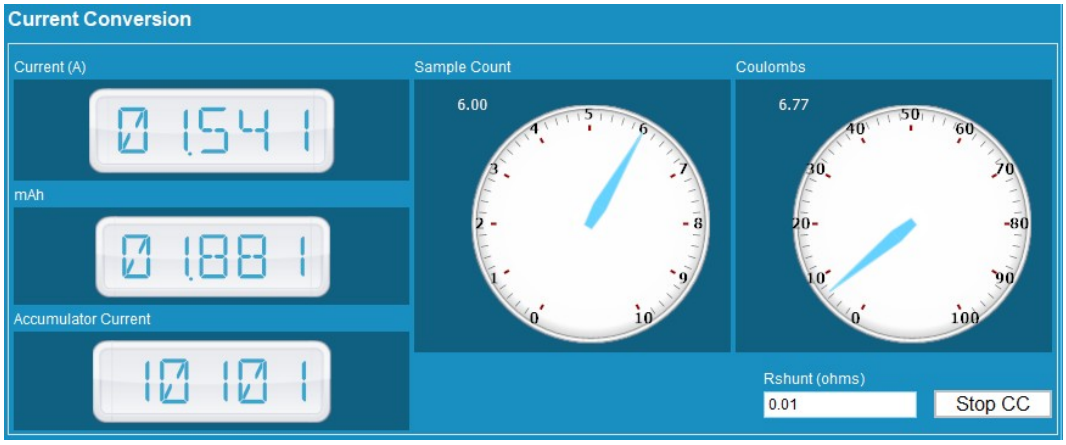
Note: For this section to properly update the visual components, the selected script must properly configure the device for voltage acquisition. Details on properly configuring the device are outlined in the L9962 datasheet. To quickly access this functionality of the device, the user can select from one of the sample scripts. This section of the L9962 demo tab also allows for the enabling and disabling of the balancing current on cells 1-10. This is done by pressing the Balance ON/OFF buttons belonging to each cell (see Figure 33).

Figure 33. Balance ON, Balance OFF buttons

Note: These buttons change their text based on the current state of the BALx_ON I2C bits. If the bit is"0" (default/ OFF), the button reads "Balance ON" to turn balancing on. If the bit is "1" (ON), the button reads "Balance OFF" to turn balancing off.

3.3.1.3 *Current conversion*

Figure 34. Current conversion section



The current conversion section of the L9962 demo tab (see Figure 34) is used to clearly display the data acquired by the current conversion routine of the L9962.

When the current conversion routine is commanded through the script, the accumulator current, sample count, and current values are collected from the device.

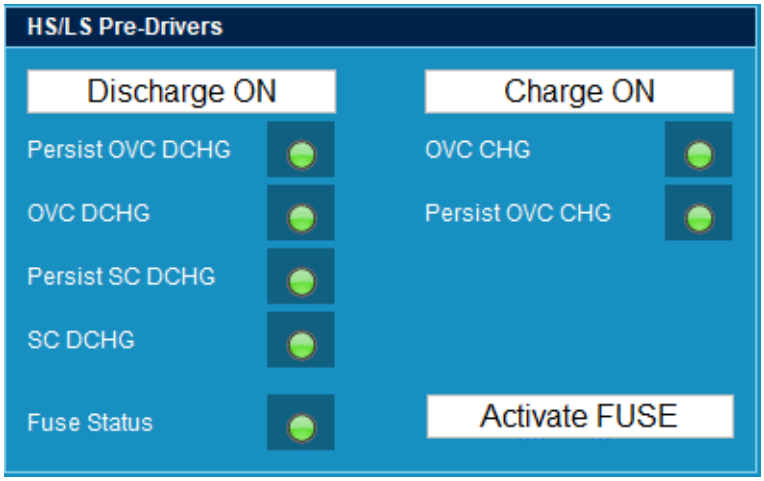
To enable the coulomb counting routine and obtain the mAh and coulomb count, the user must enter the correct Rshunt value (default value matches demo board value) and press "Start CC" (see Figure 35).

Figure 35. Start coulomb counting routine

Note: For this section to properly update the visual components, the selected script must adequately configure the device for current conversion acquisition. Details on appropriately configuring the device are outlined in the L9962 datasheet. To quickly access this functionality of the device, the user can select from one of the sample scripts.

3.3.1.4 HS/LS predrivers

Figure 36. HS/LS predrivers section

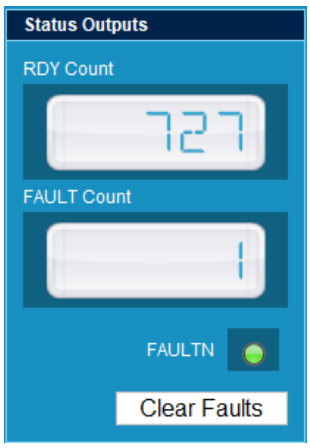


The HS/LS predrivers section of the demo tab (see Figure 36) is used to easily control the three pre-driver stages of the L9962 - external charge, discharge, and fuse switches. The relevant fault statuses are indicated by the LEDs. The behavior of the LEDs is described in detail in Section 3.3.1.2 Voltage conversion and cell balancing.

Note: For this section to properly update the visual components, the selected script must adequately configure the device by programming the pre-driver stages. Details on appropriately configuring the device are outlined in the L9962 datasheet.

3.3.1.5 Status outputs

Figure 37. Status outputs section

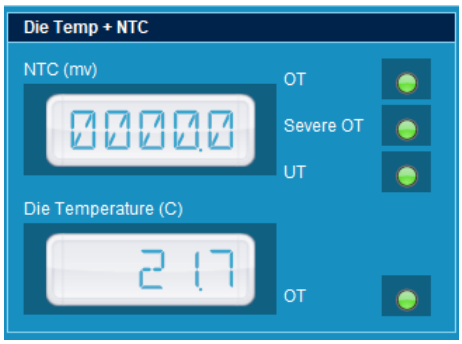


The status output section of the L9962 demo tab (see Figure 37) is used to display the count value for the data-ready interrupt pin (RDY) and the fault interrupt pin (FAULTN). Additionally, this section of the tab is used to clear all faults and displays the status of the FAULTN pin with an LED.

To clear all existing faults, press "Clear Faults". Note that this will only clear the faults that have not saturated their counters. In the case that a fault will not clear, the fault counter has saturated, and the user must follow the steps outlined in the L9962 datasheet.

3.3.1.6 *Die temp and NTC*

Figure 38. Die temp and NTC section



The die temp + NTC section of the L9962 demo tab (see Figure 38) is used to display the NTC voltage and the die temperature of the device. The relevant fault statuses are indicated by the LEDs. The behavior of the LEDs is described in detail in Section 3.3.1.2 Voltage conversion and cell balancing.

Note: For this section to properly update the visual components, the selected script must adequately configure the device for NTC voltage acquisition. Details on appropriately configuring the device are outlined in the L9962 datasheet. To quickly access this functionality of the device, the user can select from one of the sample scripts.

3.3.1.7 *Device state*

Figure 39. Device state section

The device state section of the L9962 demo tab is used to read and select the state of the device (see Figure 39).

The L9962 has three states: shipment, standby, and normal (for details on the different states, refer to the datasheet section "device functional states"). To read the current state of the device, press the "Read Mode" button. To change the state of the device, press one of the state buttons (see Figure 40):

Figure 40. Device state selection

Note that any externally driven state changes cannot be tracked by the GUI and may result in the incorrect state being depicted in the GUI. To avoid this, the state should be controlled in the GUI using the "GO" and "READ" buttons.

3.3.2 Main tab

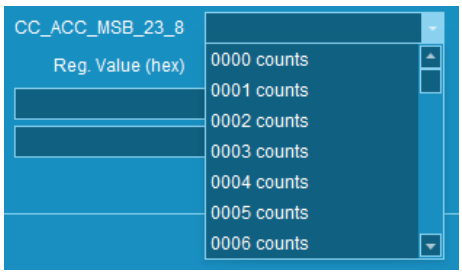
Figure 41. **Main tab display**

The main tab provides access to all of the registers not part of the NVM group or fault group. All register values are presented in hexadecimal format. For more information on I^2C registers, refer to the L9962 datasheet. Grayed out fields are read-only and cannot be modified by the user (see Figure 42).

Figure 42. **Read only register bit**

Fields that are writable are indicated by white drop-down boxes. The selection lists only valid entries (see Figure 43).

Figure 43. **Selectable bit write options**



For each register section, a button to read is available. Registers that allow write operations display both write and read buttons (see Figure 44).

Figure 44. **Read and write register buttons**

Following a read or write command, the register data is displayed in the corresponding "Reg. Value (hex)" field in hexadecimal format (see Figure 45).

Figure 45. **Register value field**

The option to read or write all registers within the main tab is also available. In the lower right corner of the tab, the "Read All" and "Write All" buttons can be used to perform this operation (see Figure 46).

Figure 46. Read All and Write All buttons

Note that the "Write All" button only applies to registers on the tab that support write operations.

3.3.3 NVM tab

Figure 47. NVM tab display

The NVM tab provides access to all of the registers that are part of the NVM group. All register values are presented in hexadecimal format. For more information on I^2C registers, refer to the L9962 datasheet.

The NVM page consists of five tabs (see Figure 48):

- Threshold
- Mask
- Manufacturer/Device
- Filter/Enable/CSA Gain
- Write NVM

Figure 48. NVM tab selections

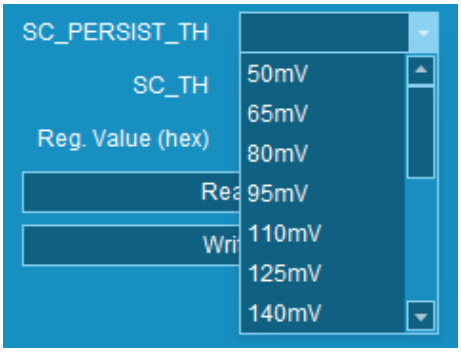
3.3.3.1 *Threshold, Mask, Mfg/Device, Filter/Enable/CSA Gain tabs*

As mentioned previously, grayed out fields are read-only and cannot be modified by the user (see Figure 49).

Figure 49. Read-only register bit

Fields that are writable are indicated by white drop-down boxes. The selection lists only valid entries (see Figure 50).

Figure 50. **Selectable bit write options**



For each register section, a button to read is available. Registers that allow write operations display both write and read buttons (see Figure 51).

Figure 51. **Read and write register buttons**

Following a read or write command, the register data is displayed in the corresponding "Reg. Value (hex)" field in hexadecimal format (see Figure 52).

Figure 52. **Register value field**

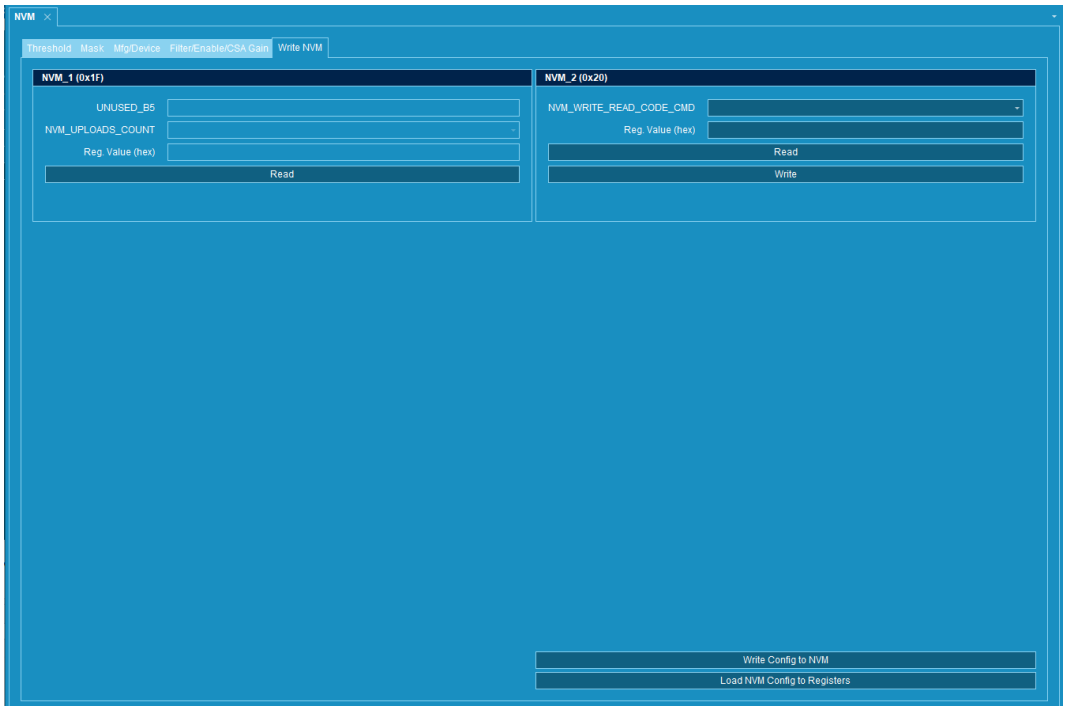
The option to read or write all registers within the selected NVM tab is also available. In the lower right corner of the tab, the "Read All" and "Write All" buttons can be used to perform this operation (see Figure 53).

Figure 53. **Read All and Write All buttons**

Note that the "Write All" button only applies to registers on the tab that support write operations.

3.3.3.2 *Write NVM tab*

Figure 54. Write NVM tab display



Note: This tab should only be used by those users that are well-versed with the L9962 device. The L9962 NVM is limited to a total of 31 write cycles. The user can read the number of NVM uploads using the NVM_1 "Read" button. If the limit is exceeded, data retention is not guaranteed! See the L9962 datasheet for details on writing to the NVM.

3.3.4 Fault tab

Figure 55. Fault tab display

The fault tab provides access to all of the registers that are part of the fault group. All register values are presented in hexadecimal format. For more information on I^2C registers, refer to the L9962 datasheet.

The user can press "Read" to retrieve the current fault status, for any of the three fault registers, at any time. All of the faults are displayed as "Fault" and highlighted in red (see Figure 56). All fields with no faults present are displayed as "No Fault" (see Figure 57). Additionally, the returned register data is shown in the "Reg. Value (hex)" field in hexadecimal format (see Figure 58).

Figure 56. Example of fault bit flagged

Figure 57. Example of no fault present

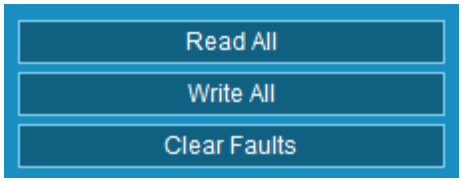
Figure 58. Register data

To clear faults, press "Write" to clear the flagged fault bits. Note that faults can only be cleared if the fault counters have not saturated, and the fault is no longer present. See the L9962 datasheet for details. As mentioned previously, grayed out fields are read-only and cannot be modified by the user (see Figure 59).

Figure 59. Read-only register bit

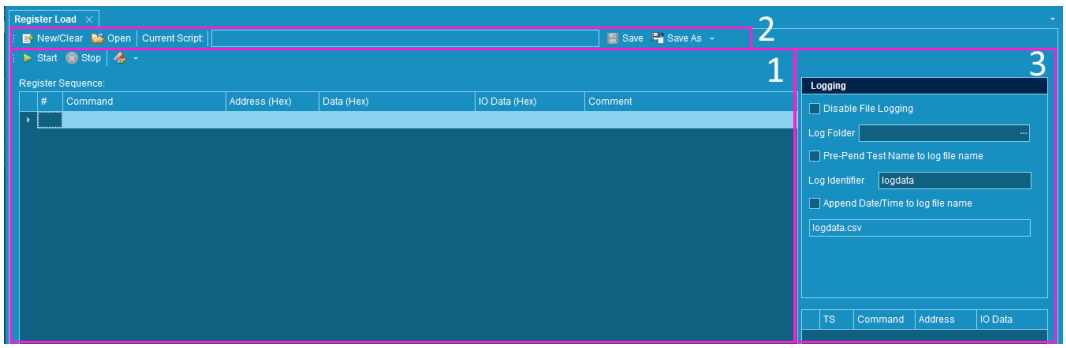
The option to read all, write all, and clear faults within the fault tab is also available. In the lower right corner of the tab, the "Read All", "Write All", and "Clear Faults" buttons can be used to perform this operation (see Figure 60).

Figure 60. Read All, Write All, Clear Faults buttons



3.3.5 Register load tab

Figure 61. Register load tab display



This tab allows the user to create a sequence of commands or script that can be run on the device (see Figure 61. Register load tab display - 1). The user can save and recall previously created register sequences/ scripts (see Figure 61. Register load tab display - 2). Additionally, when the command sequence is running, the user has the option of logging the data/results (see Figure 61. Register load tab display - 3).

Figure 62. **Register sequence display**

The user can create scripts by entering commands into the register sequence grid (see Figure 62). Each line consists of the following six columns:

1. **#**- The line number of the command.
   – This is available for reference and is auto generated by the GUI.
2. **Command** - The command to be issued.
   – This is selected by the user. The available commands are outlined in more detail in Section 3.3.5.1 Commands of this document.
3. **Address (Hex)** - The address of the register to be used in the command.
   – Not all commands require the address field and are outlined in Section 3.3.5.1 Commands.
4. **Data (Hex)** - The data for the command to be used.
   – Not all commands require the data field and are outlined in Section 3.3.5.1 Commands.
5. **IO Data (Hex)** - Commands that generate data display the results here.
6. **Comment** - If needed, a comment can be added to each line.

3.3.5.1 *Commands*

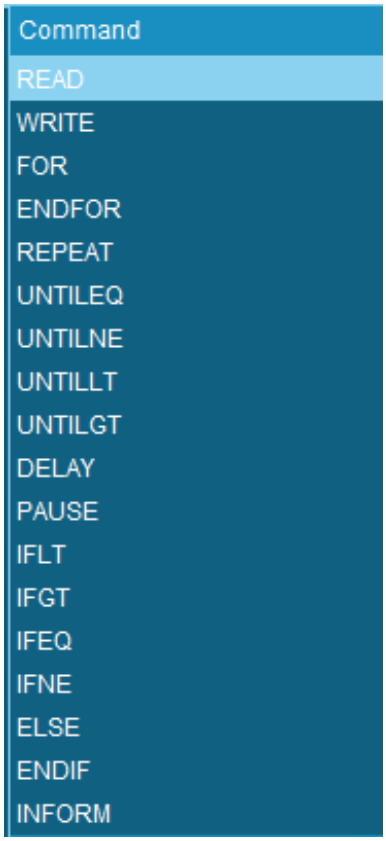
The section of the document outlines and explains the available commands that can be used in the register sequence.

The available commands are as follows:

1. **READ** - Read a value from a register.
2. **WRITE** - Write a value to a register.
3. **FOR** - Begin a FOR loop that starts at 1 and goes to set value "n".
4. **ENDFOR** - Close of FOR loop.
5. **REPEAT** - Begin of until-equal-to loop.
6. **UNTILEQ** - Close until-equal-to loop when result equals "n".
7. **UNTILNE** - Close until-equal-to loop when result does not equal "n".
8. **UNTILLT** - Close until-equal-to loop when result is less than"n".
9. **UNTILGT** - Close until-equal-to loop when result is greater than "n".
10. **IFLT** - Perform proceeding lines of commands if result is less than"n".
11. **IFGT** - Perform proceeding lines of commands if result is greater than"n".
12. **IFNE** - Perform proceeding lines of commands if result is not equal to"n".
13. **IFEQ** - Perform proceeding lines of commands if result is equal to"n".
14. **ELSE** - Perform proceeding lines of commands in case prior IF statement is not met.
15. **ENDIF** - Close IF statement.
16. **DELAY** - Delay for a specified number of milliseconds.
17. **INFORM** - Output user supplied information.
18. **PAUSE** - Allows for pausing of the script with a prompt to continue.

These commands can be selected by clicking in the "Command" field of a selected row:

Figure 63. **Command selection in register sequence**



Once the command is selected, the address or data may need to be filled in.

Note: The values for data and address are assumed to be hexadecimal numbers. There is one exception to this and that is when using the INFORM command. In this case, the data field can be any alphanumeric character.

In the following sections, commands requiring text for the address and/or data fields will have the following descriptor after the command label:

- <Address>
- <Data>
  - Example: The READ command requires the user to enter a valid hexadecimal address. The section for the READ command reads as follows: READ <Address>.

If either of these fields are not required, the above descriptor(s) will not be present.

3.3.5.1.1 READ <Address>

The READ command is used to request a read of a specified address. The address must be entered in hexadecimal format to be acknowledged as a valid command. The returned data are available in the "IO Data (Hex)" column on the same line.

Figure 64. **Valid READ command**

3.3.5.1.2 WRITE <Address> <Data>

The WRITE command is used to command a write of data to a specified address. The address and data must be entered in hexadecimal format to be acknowledged as a valid command. The returned data is available in the "IO Data (Hex)" column on the same line.

Figure 65. Valid WRITE command

#	Command	Address (Hex)	Data (Hex)	IO Data (Hex)	Comment
0	WRITE	6	AFF		N=10, Cell OV = 4.9976V

3.3.5.1.3 DELAY <Data>

The DELAY command is used to set a delay for a set number of milliseconds. The time of the delay is entered in the "Data (Hex)" field.

Figure 66. Valid DELAY command

#	Command	Address (Hex)	Data (Hex)	IO Data (Hex)	Comment
0	DELAY		A		

Note: Depending on the PC hardware and software the evaluation software is running on, there is potential for the delay to exhibit some variances from PC to PC.

3.3.5.1.4 FOR <Data>

The FOR command is used to build a FOR loop within the register sequence. The value entered in the "Data (Hex)" field specifies the number of times the loop will be executed (must be entered in hexadecimal format). An ENDFOR command must follow a FOR command to close the loop. Failure to do so will result in errors.

Figure 67. Valid FOR loop structure

#	Command	Address (Hex)	Data (Hex)	IO Data (Hex)	Comment
0	FOR		5		
1	READ	A			
2	WRITE	E	AF		
3	ENDFOR				

Note: It is not possible to interleave FOR loops with other REPEAT/IF/FOR statements. To repeat a block containing a FOR loop, the REPEAT command would have to be inserted before (or after) the FOR loop command (see Figure 68):

Figure 68. Valid FOR loop with repeat until equal statement

#	Command	Address (Hex)	Data (Hex)	IO Data (Hex)	Comment
0	REPEAT				
1	FOR		5		
2	READ	A			
3	WRITE	E	AF		
4	ENDFOR				
5	UNTILEQ	A	5		

3.3.5.1.5 ENDFOR

The ENDFOR command is used to close a FOR loop within the register sequence. A FOR command must proceed an ENDFOR command to complete the FOR loop. Failure to do so will result in errors.

Figure 69. Valid ENDFOR command structure

#	Command	Address (Hex)	Data (Hex)	IO Data (Hex)	Comment
0	FOR		5		
1	READ	A			
2	WRITE	E	AF		
3	ENDFOR				

3.3.5.1.6 REPEAT

The REPEAT command is used to create a repeat-until-equal conditional statement within the register sequence. A REPEAT command must be followed by a UNTILEQ/UNTILNE/UNTILLT/UNTILGT command to complete the repeat-until-equal conditional statement. Failure to do so will result in errors.

Figure 70. Valid REPEAT command structure

#	Command	Address (Hex)	Data (Hex)	IO Data (Hex)	Comment
0	REPEAT				
1	READ	A			
2	UNTILEQ	A	5		

3.3.5.1.7 UNTILEQ/UNTILNE/UNTILLT/UNTILGT <Address><Data>

The UNTILEQ/UNTILNE/UNTILLT/UNTILGT commands are used to close a repeat-until-equal conditional statement. A REPEAT command must proceed an UNTILxx command to complete the conditional statement. Failure to do so will result in errors.

The UNTILxx command definitions are as follows:

- UNTILEQ = Until address result is equal to"n"
- UNTILNE = Until address result is not equal to"n"
- UNTILLT = Until address result is less than "n"
- UNTILGT = Until address result is greater than"n"

The value entered in the "Address (Hex)" field specifies the register to perform the repeat-until-equal conditional statement on. The value entered in the "Data (Hex)" field specifies the comparison value "n".

Note: The address to be evaluated must be READ prior to an UNTILxx command.

Figure 71. Valid UNTILxx command structure

#	Command	Address (Hex)	Data (Hex)	IO Data (Hex)	Comment
0	REPEAT				
1	READ	A			
2	UNTILEQ	A	5		

3.3.5.1.8 IFLT/IFGT/IFEQ <Address> <Data>

The IFLT/IFGT/IFEQ commands are used to create a conditional IF statement within the register sequence. An IFxx command must be followed by an ENDIF or ELSE command to complete the conditional IF statement.

Failure to do so will result in errors.

The IFxx command definitions are as follows:

- IFLT = If address result is less than "n"
- IFGT = If address result is greater than "n"
- IFEQ = If address result is equal to "n"
- IFNE = If address result is not equal to "n"

The value entered in the "Address (Hex)" field specifies the register to perform the conditional IF statement. The value entered in the "Data (Hex)" field specifies the comparison value "n".

If the IFxx condition is met, then the next series of commands are run.

Note: The address to be evaluated must be READ prior to an IFxxcommand.

Figure 72. Valid IFxx command structure

#	Command	Address (Hex)	Data (Hex)	IO Data (Hex)	Comment
0	READ	A			
1	IFGT	A	5		
2	READ	D			
3	READ	B			
4	ENDIF				

3.3.5.1.9 ELSE

The ELSE command is used to create a branch from a previous conditional IF statement if the IFxx conditions are not met. An IFxx command (and its series of commands) must precede an ELSE command, and an ENDIF command must follow an ELSE command. Failure to do so will result in errors.

Figure 73. Valid ELSE command structure

#	Command	Address (Hex)	Data (Hex)	IO Data (Hex)	Comment
0	READ	A			
1	IFGT	A	5		
2	READ	D			
3	READ	B			
4	ELSE				
5	WRITE	A	5		
6	ENDIF				

3.3.5.1.10 ENDIF

The ENDIF command is used to close a conditional IF statement within the register sequence. An IFxx or ELSE command must precede an ENDIF command to complete the IF statement. Failure to do so will result in errors.

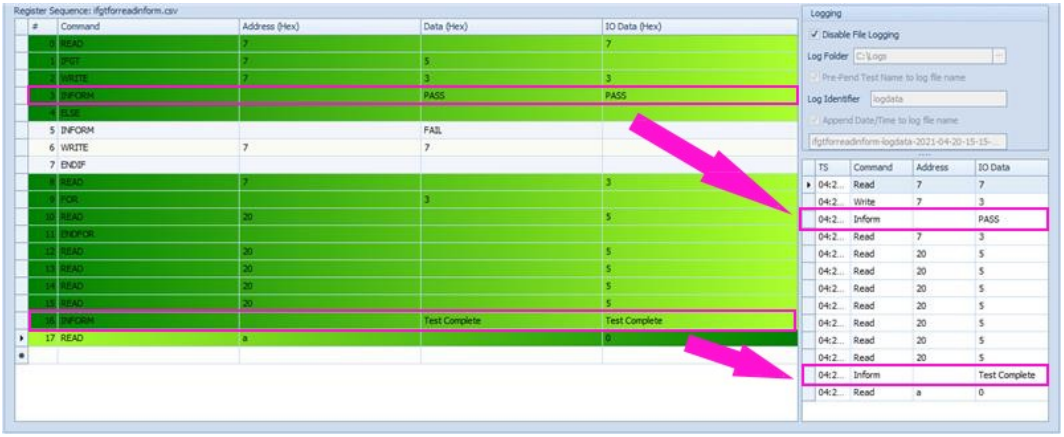
Figure 74. Valid ENDIF command structure

#	Command	Address (Hex)	Data (Hex)	IO Data (Hex)	Comment
0	READ	A			
1	IFGT	A	5		
2	READ	D			
3	READ	B			
4	ENDIF				

3.3.5.1.11 INFORM <Data>

The INFORM command is used to pass text to the log file. The text entered in the "Data (Hex)" field will be passed to the "IO Data (Hex)" field in the event that the script reaches the line. This instruction does not require data to be in hexadecimal format.

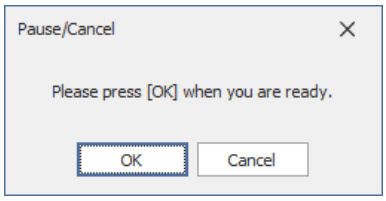
Figure 75. Valid INFORM command



3.3.5.1.12 PAUSE <Data>

The PAUSE command is used to pause the execution of the script. Once the script reaches this line, it pauses and prompts the user to continue or cancel (see Figure 76). This can be useful in instances where a hardware adjustment needs to be made before proceeding to the next line in the script.

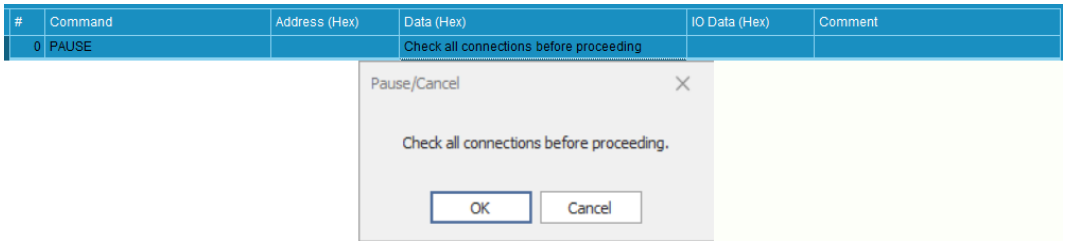
Figure 76. Pause prompt window



To proceed to the next line in the script, select "OK". Otherwise, if the user wishes to exit the script, press "Cancel."

The user also has the ability to change the text shown in the pop-up window. To change from the default, enter the desired text in the data column. For example, if the user needs to check hardware connection before proceeding, the text "Check all connections before proceeding" can be entered in the data column. This will then appear in the pop-up window with the script executes (see Figure 77).

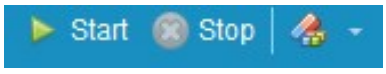
Figure 77. Customizing the text of a pause pop-up window



3.3.5.2 *Run/Abort register sequence*

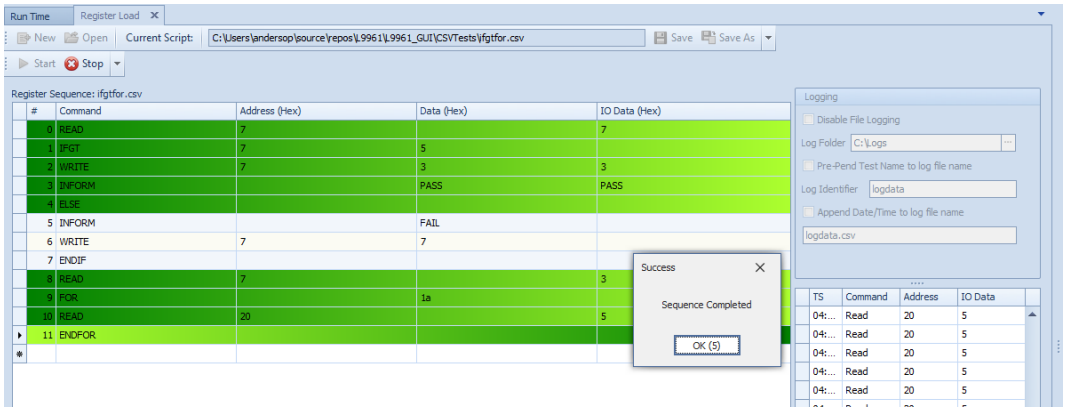
Once the user has completed the register sequence, the script can be run using the ‘Start’ button. Should it become necessary to abort a register sequence while it is running, the "Stop" button can be used (see Figure 78).

Figure 78. Script "Start" button



After completing a sequence, a pop-up window with a six-second timer will appear, indicating that the script ran successfully. Additionally, the lines that executed successfully will be indicated by turning green (see Figure 79).

Figure 79. Completion of sequence



3.3.5.3 *Clear, save, and load sequence*

Figure 80. Clear, save, and load

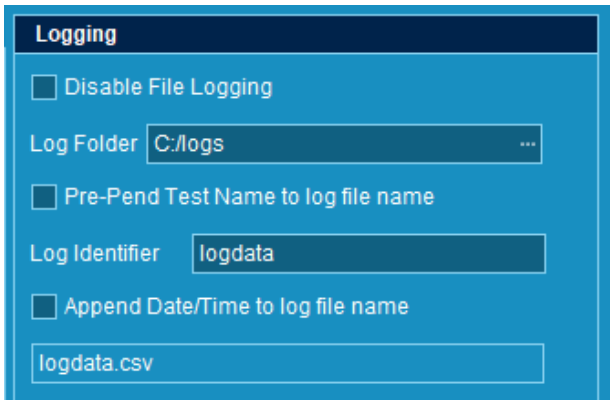
To clear the register sequence table and create a new sequence, press the "New/Clear" button.

Sequences can be saved via the “Save” button. A file dialog box pops up if no file name has been previously selected. The file is saved as a comma-separated file (csv).

To load a pre-existing sequence file, press the "Open" button and select a valid csv file through the file dialog window. Once the file has been selected, the register sequence is loaded into the register sequence table.

3.3.5.4 *Logging sequence*

Figure 81. Sequence logging



The user has the option to log the results from a sequence execution.

Sequence logging can be enabled or disabled on demand through the "Disable File Logging" check box. By default, the check box is cleared, logging data to the "Log Folder" location. If the check box is highlighted, the file logging is disabled.

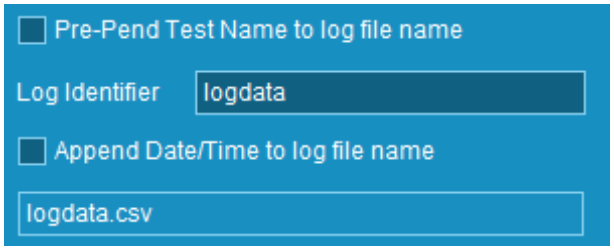
To configure the "Log Folder" location, press the three dots next to the text box, and navigate to the desired save location.

A valid log file name can contain the following:

- Current script name
  Log identifier
  Date/time(YYYY-MM-DD-HH-MM-SS)

The fields above can be added/removed by using the following fields (see Figure 82):

Figure 82. Log file name fields



Pre-pending the test script name to the identifier can be turned on or off via the check box.

If "Append Data/Time to log file name" is checked, the data time is appended to the file name.

If neither the pre-pend nor append check boxes are selected, the log identifier is used as the file name. If both check boxes are selected, the file name has the following format:

TestName-LogIdentifier-DataTime.csv.

Opening the log file from the specified save location yields the following (see Figure 83):

Figure 83. Register Load tab data log file

TimeStamp	Command	Address	Data
05:05:21:10:05:53:118	Write	6	aff
05:05:21:10:05:53:141	Write	7	a00
05:05:21:10:05:53:155	Write	8	0
05:05:21:10:05:53:171	Write	9	a00
05:05:21:10:05:53:186	Write	a	aff
05:05:21:10:05:53:205	Write	b	a00
05:05:21:10:05:53:218	Write	c	ff
05:05:21:10:05:53:240	Write	d	a000
05:05:21:10:05:53:251	Write	e	afff
05:05:21:10:05:53:266	Write	f	1
05:05:21:10:05:53:282	Write	10	0
05:05:21:10:05:53:304	Write	11	0
05:05:21:10:05:53:314	Write	12	0
05:05:21:10:05:53:340	Write	4	383f
05:05:21:10:05:54:752	Write	2	7ff

The header in the log grid contains the time stamp (TS), the command (Command), the address (Address), and the IO data (IO Data) (see Figure 84). The grid and the log only provide IO Data for READ, WRITE, and INFORM functions.

Figure 84. Sequence log grid

TS	Command	Address	IO Data
04:...	Read	20	5
04:...	Read	20	5
04:...	Read	20	5

Revision history

Table 1. **Document revision history**

Date	Version	Changes
19-Jan-2026	1	Initial release.
06-Jul-2026	2	Inserted new chapter, see NUCLEO board firmware programming with STM32CubeProgrammer

Contents

List of tables

List of figures

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers' market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.