



Getting started with the X-CUBE-EEPRMA1 software expansion for STM32Cube

Introduction

The X-CUBE-EEPRMA1 software expansion for STM32Cube provides an evaluation software example for M24256E-F I²C, M24M01E-F I²C, M24XX I²C, M95XX SPI standard EEPROM or M95P32 SPI page EEPROM.

The package is built on STM32Cube software technology to ease portability across different STM32 microcontrollers.

The software comes with sample implementations of the drivers running on the X-NUCLEO-EEICA1, X-NUCLEO-EEPRMA2, and X-NUCLEO-PGEEZ1 expansion boards connected to the featured development boards.

Related links

Visit the STM32Cube ecosystem web page on www.st.com for further information

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
BSP	Board support package
CMSIS	Cortex® microcontroller software interface standard
HAL	Hardware abstraction layer
I2C	Inter integrated circuit
SPI	Serial peripheral interface
CDA	Configurable device address
SWP	Software write protect
DTI	Device type identifier
ID page	Identification page

2 X-CUBE-EEPRMA1 software expansion for STM32Cube

2.1 Overview

The X-CUBE-EEPRMA1 software package key features are:

- Complete software to build applications using M24256E-F I²C, M24M01E-F I²C, M24XX I²C, M95XX SPI standard EEPROM or M95P32 SPI page EEPROM
- Sample implementation available on the X-NUCLEO-EEICA1, X-NUCLEO-EEPRMA2, and X-NUCLEO-PGEEZ1 expansion boards plugged to a NUCLEO-F401RE, NUCLEO-G474RE, NUCLEO-L053R8, or NUCLEO-H753ZI (only for X-NUCLEO-EEICA1 and X-NUCLEO-PGEEZ1) development board
- Easy portability across different MCU families thanks to STM32Cube
- Free user-friendly license terms
- Package compatible with STM32CubeMX, can be downloaded from and directly installed into STM32CubeMX

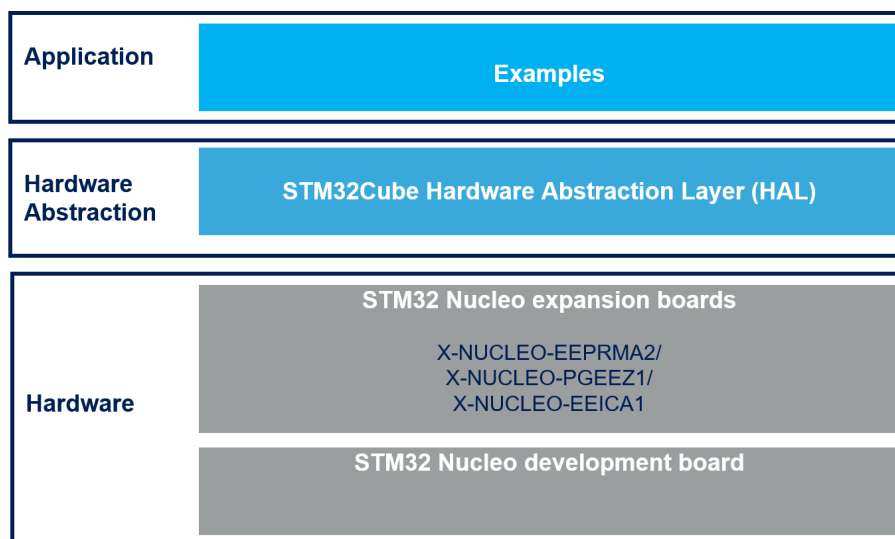
2.2 Architecture

The software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller and extends STM32Cube by providing a board support package (BSP) for the X-NUCLEO-EEICA1, X-NUCLEO-EEPRMA2, and X-NUCLEO-PGEEZ1 expansion boards.

The software layers used by the application software to access and use the board are:

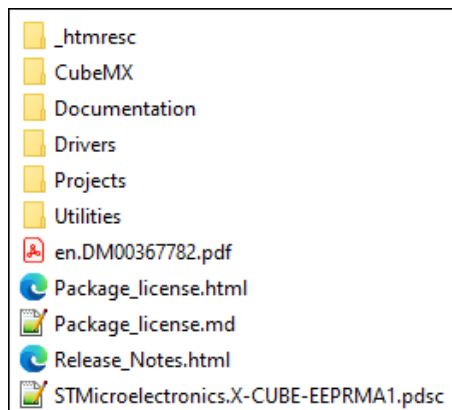
- **STM32Cube HAL layer:** provides a generic, multi-instance set of simple APIs (application programming interfaces) to interact with the upper layers (application, libraries and stacks). It is composed of generic and extension APIs.
- **Board support package (BSP) layer:** supports the peripherals on the STM32 Nucleo board, except for the MCU. This limited set of APIs provides a programming interface for certain board specific peripherals (like the user button, the reset button, etc.) and helps in identifying the specific board version.

Figure 1. X-CUBE-EEPRMA1 software architecture



2.3 Folder structure

Figure 2. X-CUBE-EEPRMA1 package folder structure



The software is packaged in the following main folders:

- **Documentation:** with a compiled HTML file generated from the source code that details the software components and APIs.
- **Drivers:** contains the HAL drivers, the board specific drivers for each supported board or hardware platform, including the on-board components ones and the CMSIS layer which is a vendor-independent hardware abstraction layer for the Cortex-M processor series.
- **Projects:** contains sample application used for the [NUCLEO-L053R8](#), [NUCLEO-G474RE](#), [NUCLEO-H753ZI](#), and the [NUCLEO-F401RE](#) platforms with three development environments: IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit (MDK-ARM), and [STM32CubeIDE](#).

2.4 APIs

Detailed function and parameter descriptions for the user-APIs are compiled in an HTML file in the software package Documentation folder.

3 System setup guide

3.1 Hardware description

3.1.1 STM32 Nucleo

STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

The Arduino connectivity support and ST morpho connectors make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide range of specialized expansion boards to choose from.

The STM32 Nucleo board does not require separate probes as it integrates the ST-LINK/V2-1 debugger/programmer.

The STM32 Nucleo board comes with the comprehensive STM32 software HAL library together with various packaged software examples for different IDEs (IAR EWARM, Keil MDK-ARM, STM32CubeIDE, mbed and GCC/LLVM).

All STM32 Nucleo users have free access to the mbed online resources (compiler, C/C++ SDK and developer community) at www.mbed.org to easily build complete applications.

Figure 3. STM32 Nucleo board



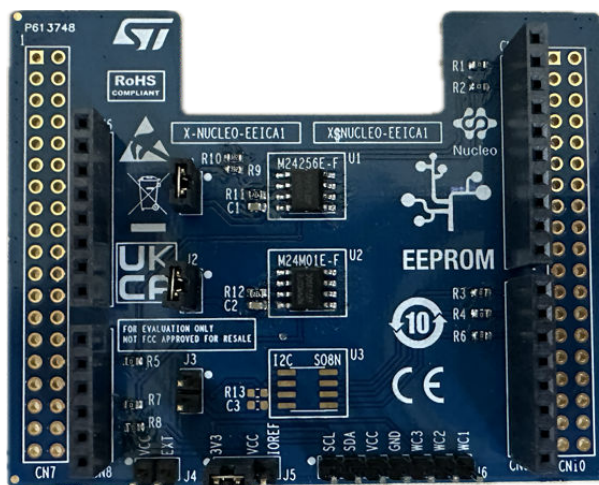
3.1.2 X-NUCLEO-EEICA1 expansion board

The X-NUCLEO-EEICA1 expansion board is designed for M24256E-F and M24M01E-F series I2C EEPROM for data reading and writing.

The expansion board acts as an external storage device that can be used to store data such as manufacturing traceability, calibration, user setting, error flags, data log and monitoring data to make applications more flexible and accurate.

The X-NUCLEO-EEICA1 expansion board is compatible with the Arduino UNO R3 connector pin assignment and can be easily plugged to any STM32 Nucleo development board. You can mount the ST morpho connectors if required.

Figure 4. X-NUCLEO-EEICA1 expansion board



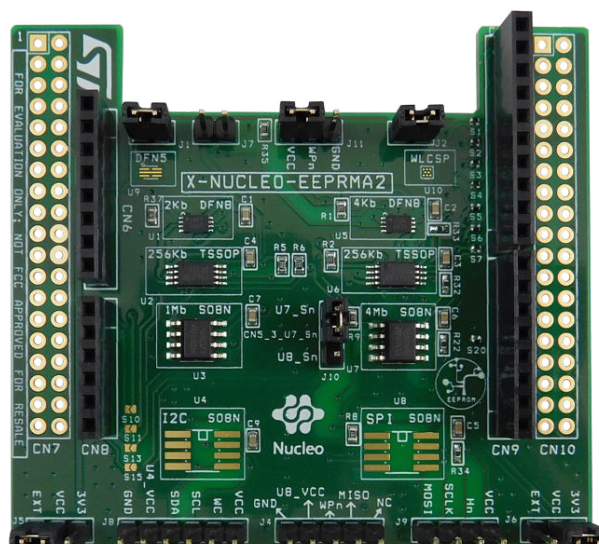
3.1.3 X-NUCLEO-EEPRMA2 expansion board

The X-NUCLEO-EEPRMA2 expansion board is designed for M24xx series I²C and M95xx series SPI EEPROM for data reading and writing.

The expansion board acts as an external storage device that can be used to store data such as manufacturing traceability, calibration, user settings, error flags, data logs and monitoring data to make applications more flexible and accurate.

The X-NUCLEO-EEPRMA2 expansion board is compatible with the Arduino UNO R3 connector pin assignment and can be easily plugged to any STM32 Nucleo development board. You can mount the ST morpho connectors if required.

Figure 5. X-NUCLEO-EEPRMA2 expansion board



3.1.4 X-NUCLEO-PGEEZ1 expansion board

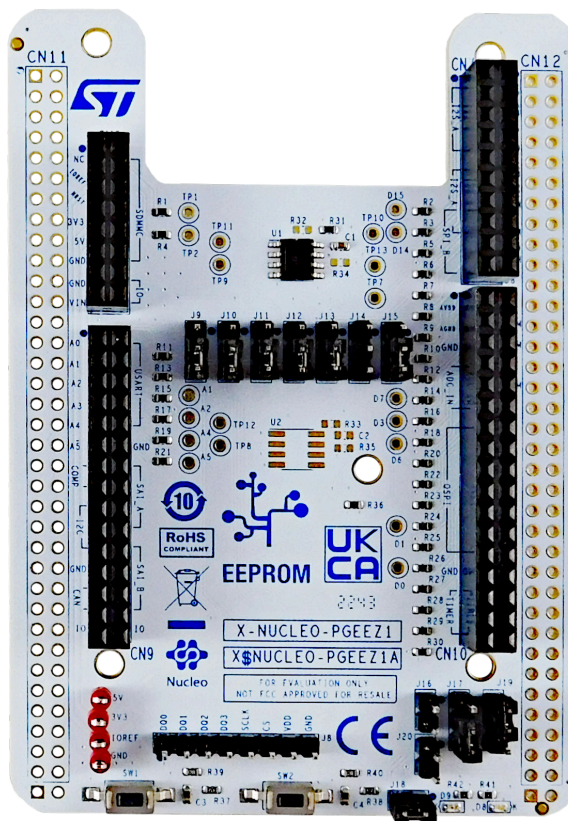
The X-NUCLEO-PGEEZ1 expansion board is designed for the M95P32 series SPI page EEPROM for data reading and writing.

This expansion board allows developers to evaluate the new memory page EEPROM through a single/dual/quad SPI interface.

It acts as an external storage device that can be used to store data, such as manufacturing traceability, calibration, user settings, error flags, data logs, and monitoring data to build more flexible and accurate applications.

The X-NUCLEO-PGEEZ1 expansion board can be plugged on top of an STM32 Nucleo-144 (via ZIO connectors) or STM32 Nucleo-64 (via Arduino® UNO R3 connectors).

Figure 6. X-NUCLEO-PGEEZ1 expansion board



3.2 Hardware setup

The following hardware components are needed:

1. One STM32 Nucleo development platform (NUCLEO-F401RE, NUCLEO-G474RE, NUCLEO-L053R8, or NUCLEO-H753ZI (only for X-NUCLEO-EEICA1 and X-NUCLEO-PGEEZ1)
2. One X-NUCLEO-EEICA1, X-NUCLEO-EEPRMA2, or X-NUCLEO-PGEEZ1 expansion board
3. One USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC

3.2.1 STM32 Nucleo and X-NUCLEO-EEICA1/X-NUCLEO-EEPRMA2/X-NUCLEO-PGEEZ1 expansion board setup

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer.

The developer can freely download the ST-LINK/V2-1 from www.st.com.

The X-NUCLEO-EEICA1, X-NUCLEO-EEPRMA2, or X-NUCLEO-PGEEZ1 expansion board can be easily connected to the STM32 Nucleo through the Arduino® UNO R3 extension connector as shown below.

Figure 7. X-NUCLEO-EEPRMA2 expansion board plugged to an STM32 Nucleo board

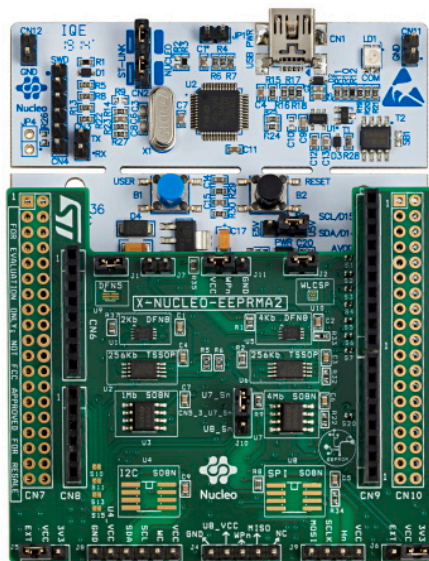


Figure 8. X-NUCLEO-PGEEZ1 expansion board plugged to an STM32 Nucleo board

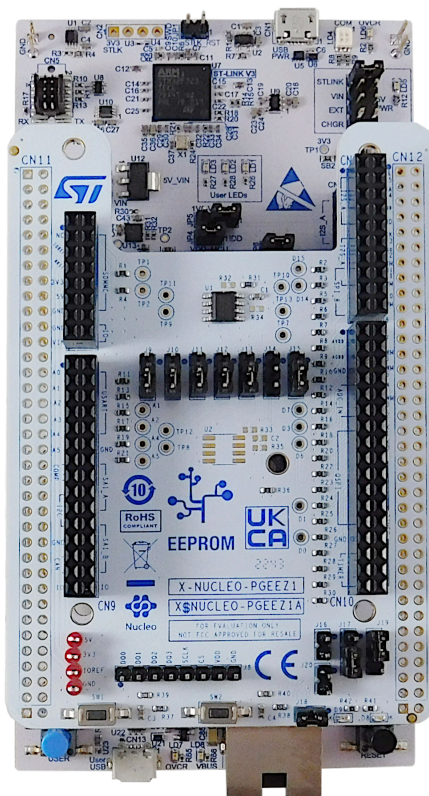


Table 2. X-NUCLEO-PGEEZ1 jumper settings

Jumper	Name	Configuration (for NUCLEO-H753ZI board)	Configuration (for NUCLEO-L053R8 board)
J09	M95_MOSI	2-3	1-2
J10	M95_MISO	2-3	1-2
J11	M95_WPn	2-3	1-2
J12	M95_Hn	2-3	1-2

Jumper	Name	Configuration (for NUCLEO-H753ZI board)	Configuration (for NUCLEO-L053R8 board)
J13	U1_CS	2-3	1-2
J15	M95_SCLK	2-3	1-2
J17	M95_VDD	2-3	2-3
J19	USR_VDD	1-2	1-2

3.3 Software setup

The following software components are needed for a suitable development environment for creating applications for the **STM32 Nucleo** equipped with the **X-NUCLEO-EEICA1**, **X-NUCLEO-EEPRMA2**, or **X-NUCLEO-PGEEZ1** expansion board:

- **X-CUBE-EEPRMA1** expansion for **STM32Cube** that supports **X-NUCLEO-EEICA1**, **X-NUCLEO-EEPRMA2** and **X-NUCLEO-PGEEZ1** expansion boards.
- One of the following development tool-chain and compilers:
 - Keil RealView Microcontroller Development Kit (MDK-ARM) + **ST-LINK**
 - IAR Embedded Workbench for ARM (EWARM) + **ST-LINK**
 - Integrated Development Environment for STM32 (**STM32CubeIDE**) + **ST-LINK**

The package includes the following firmware examples for the developer to start experimenting with the code:

1. a standard initialization of the I²C and SPI EEPROM
2. a firmware example containing:
 - block protect, which reads the status after "write enable" and "write disable" for the SPI EEPROM
 - write protection feature for the I²C EEPROM
3. a firmware example that writes the complete memory and reads the information received
4. a firmware example that demonstrates Configurable Device Address, Software Write Protect, Device Type Identifier features

Figure 9. Console log for X-NUCLEO-EEPRMA2

```

Docklight V2.4
File Edit Run Tools Help Stop Communication (F6)
Communication port open Colors&Fonts Mode COM9 115200, None, 8, 1
Communication
Send Sequences
--SPI EEPROM READ STATUS--
target: M95256|rx: 00
target: M95256|rx: 02
target: M95040|rx: f0
target: M95040|rx: f2
target: M95M04|rx: 00
target: M95M04|rx: 02

-- I2C/SPI EEPROM SINGLE BYTE--
TestByte|target: M24C02|address: 0|TX: k| RX: k|result: passed
TestByte|target: M24256|address: 200|TX: k| RX: k|result: passed
TestByte|target: M24M01|address: 5|TX: k| RX: k|result: passed
TestByte|target: M95040|address: 1|TX: b| RX: b|result: passed
TestByte|target: M95M04|address: 1|TX: b| RX: b|result: passed

-- I2C/SPI EEPROM DATA --
TestData|target: M24C02|address: 0|TX: abcdefghE-EEPROM-Expansion Firmware library EEPROM driver example : This firmware
provides a basic example of how to use the X-Nucleo-eXpansion firmware library. This block of data is specially written to
test the data write function of EEPROM (SPI/I2C) | RX: abcdefghE-EEPROM-Expansion Firmware library EEPROM driver example
: This firmware provides a basic example of how to use the X-Nucleo-eXpansion firmware library. This block of data is
specially written to test the data write function of EEPROM (SPI/I2C) |data size: 255|result: passed
TestData|target: M95040|address: 0|TX: abcdefghE-EEPROM-Expansion Firmware library EEPROM driver example : This firmware
provides a basic example of how to use the X-Nucleo-eXpansion firmware library. This block of data is specially written to
test the data write function of EEPROM (SPI/I2C) | RX: abcdefghE-EEPROM-Expansion Firmware library EEPROM driver example
: This firmware provides a basic example of how to use the X-Nucleo-eXpansion firmware library. This block of data is
specially written to test the data write function of EEPROM (SPI/I2C) |data size: 255|result: passed

--I2C EEPROM PAGE--
TestPage|target: M24C02|address: 0|data size: 16|result: passed

--SPI EEPROM PAGE--
TestPage|target: M95M04|address: 0|data size: 511|result: passed

--SPI M95M04 EEPROM READ ID PAGE--
Read ID Page from address : 000000x:
55 55 55 55 55 55 55 55 55 55 00 00 ff ff ff
Writing Identification Page...
Read ID Page from address : 000000x:
55 55 55 55 55 55 55 55 55 55 00 00 ff ff ff
Lock not active
  
```

Figure 10. Console log for X-NUCLEO-PGEEZ1

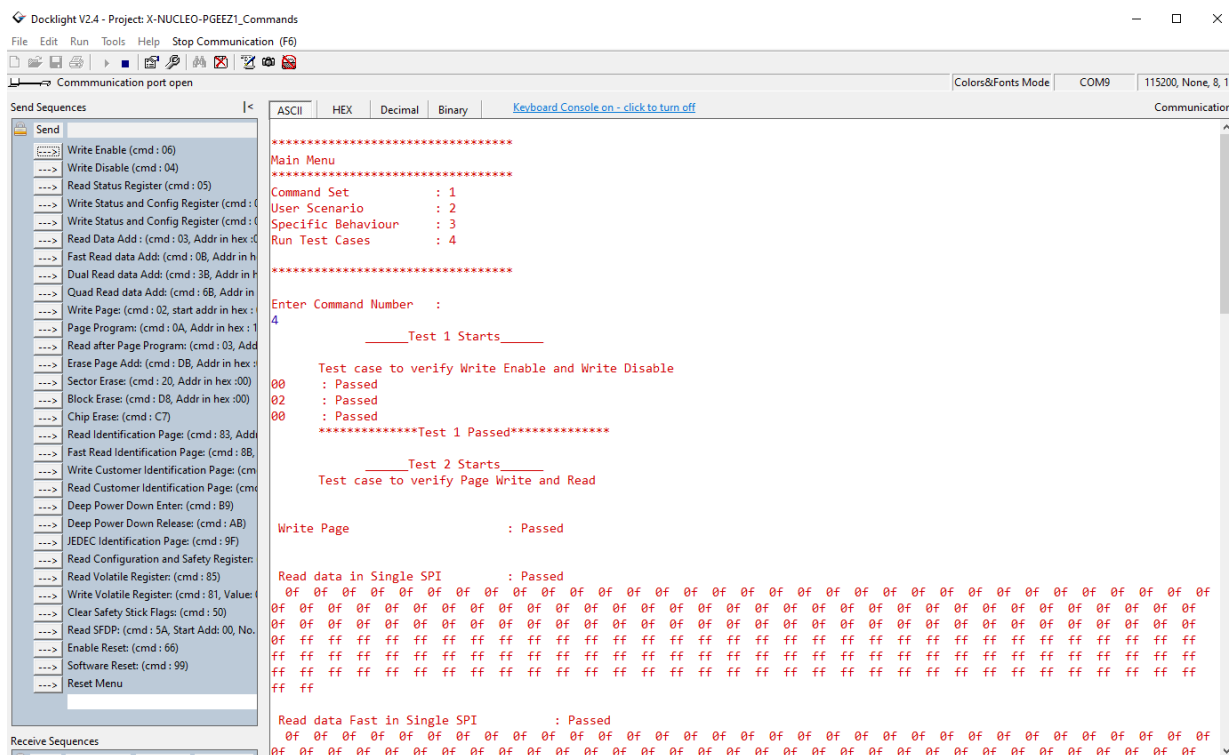
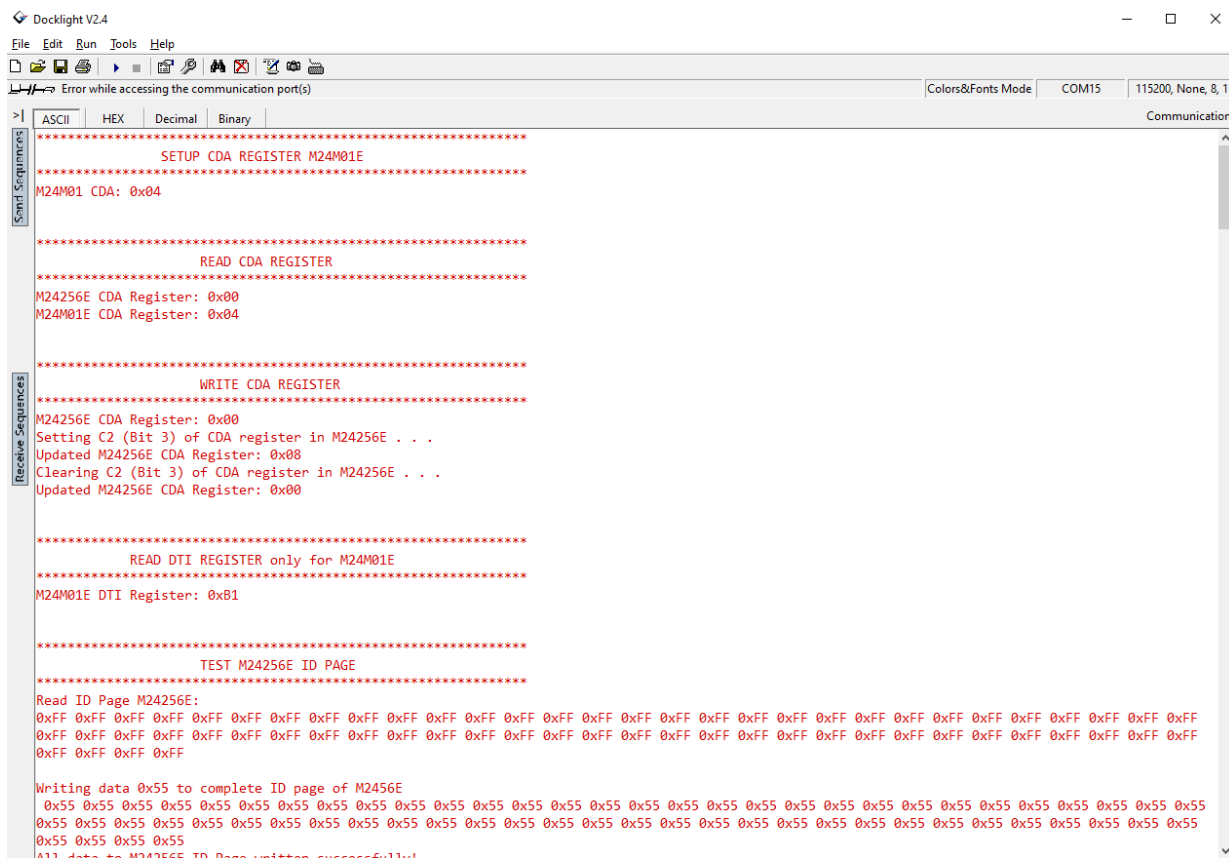


Figure 11. Console log for X-NUCLEO-EEICA1



4 Test setup

All the expansion boards supported in the package i.e. [X-NUCLEO-EEICA1](#), [X-NUCLEO-EEPRMA2](#) and [X-NUCLEO-PGEEZ1](#) have been tested with different types of test cases. The test environment setup is same for all the tests.

4.1 Firmware binary

The firmware binary used for the tests are EEICA1_EEPROMRW.bin, EEPRMA2_EEPROMRW.bin and PGEEZ1_EEPROMRW.bin with [X-NUCLEO-EEICA1](#), [X-NUCLEO-EEPRMA2](#) and [X-NUCLEO-PGEEZ1](#) respectively for their supported NUCLEO development boards.

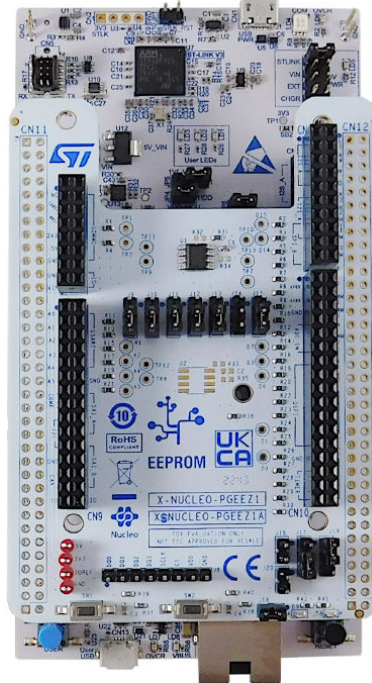
4.2 Test platforms (boards)

- [NUCLEO-H753ZI](#) + ([X-NUCLEO-EEICA1](#) or [X-NUCLEO-PGEEZ1](#))
- [NUCLEO-L053R8](#) + ([X-NUCLEO-EEICA1](#) or [X-NUCLEO-EEPRMA2](#) or [X-NUCLEO-PGEEZ1](#))
- [NUCLEO-F401RE](#) + ([X-NUCLEO-EEICA1](#) or [X-NUCLEO-EEPRMA2](#) or [X-NUCLEO-PGEEZ1](#))
- [NUCLEO-G474RE](#) + ([X-NUCLEO-EEICA1](#) or [X-NUCLEO-EEPRMA2](#) or [X-NUCLEO-PGEEZ1](#))

4.3 Precondition

- ST VCOM port drivers must be installed

- Keep the jumper settings as shown in the figure below (when using the X-NUCLEO-PGEEZ1 expansion board with NUCLEO-H753ZI development board).

Figure 12. Jumper settings (for X-NUCLEO-PGEEZ1)

Table 3. X-NUCLEO-PGEEZ1 jumper settings

Jumper	Name	Configuration (for the NUCLEO-H753ZI development board)	Configuration (for the other development boards)
J09	M95_MOSI	2-3	1-2
J10	M95_MISO	2-3	1-2
J11	M95_WPn	2-3	1-2
J12	M95_Hn	2-3	1-2
J13	U1_CS	2-3	1-2
J15	M95_SCLK	2-3	1-2
J17	M95_VDD	2-3	2-3
J19	USR_VDD	1-2	1-2

- For other expansion boards, jumper settings can be kept as it is. Refer respective user manual for further information.

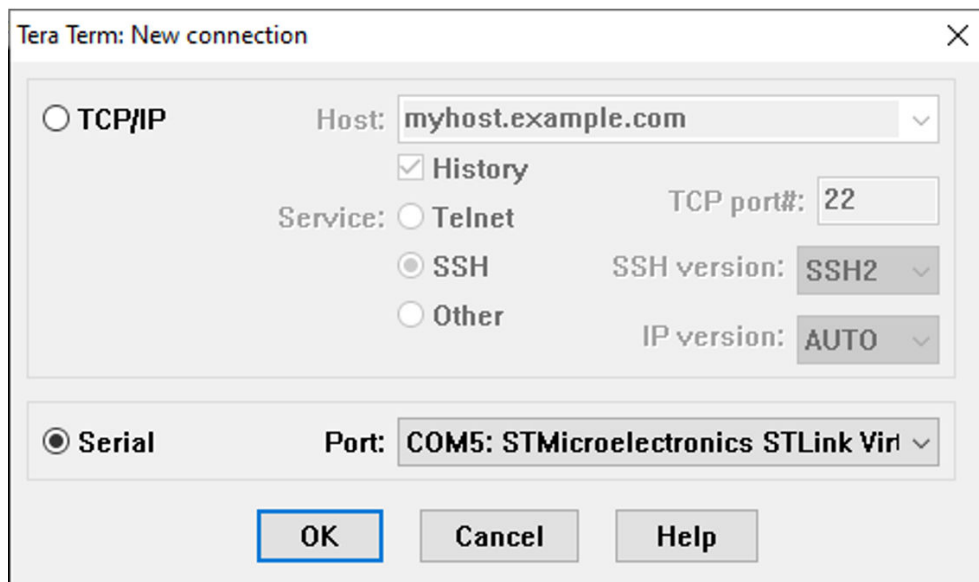
4.4 Test execution

- Step 1.** Connect the expansion board on top of the NUCLEO development board via Zio connector..
- Step 2.** Connect the NUCLEO development board to the PC using a USB cable to power it on.
- Step 3.** Drag and drop the required binary file to the connected NUCLEO development board through PC.

Note: Perform this step only once. You can use the same STM32 Nucleo development board to test similar expansion boards.

- Step 4.** Open Tera Term with the below connection settings.
Different COM ports number could appear.

Figure 13. Tera term settings – COM port



Tera Term: New connection

☐ TCP/IP

Host: myhost.example.com

☒ History

Service: ☐ Telnet

TCP port#: 22

☒ SSH

SSH version: SSH2

☐ Other

IP version: AUTO

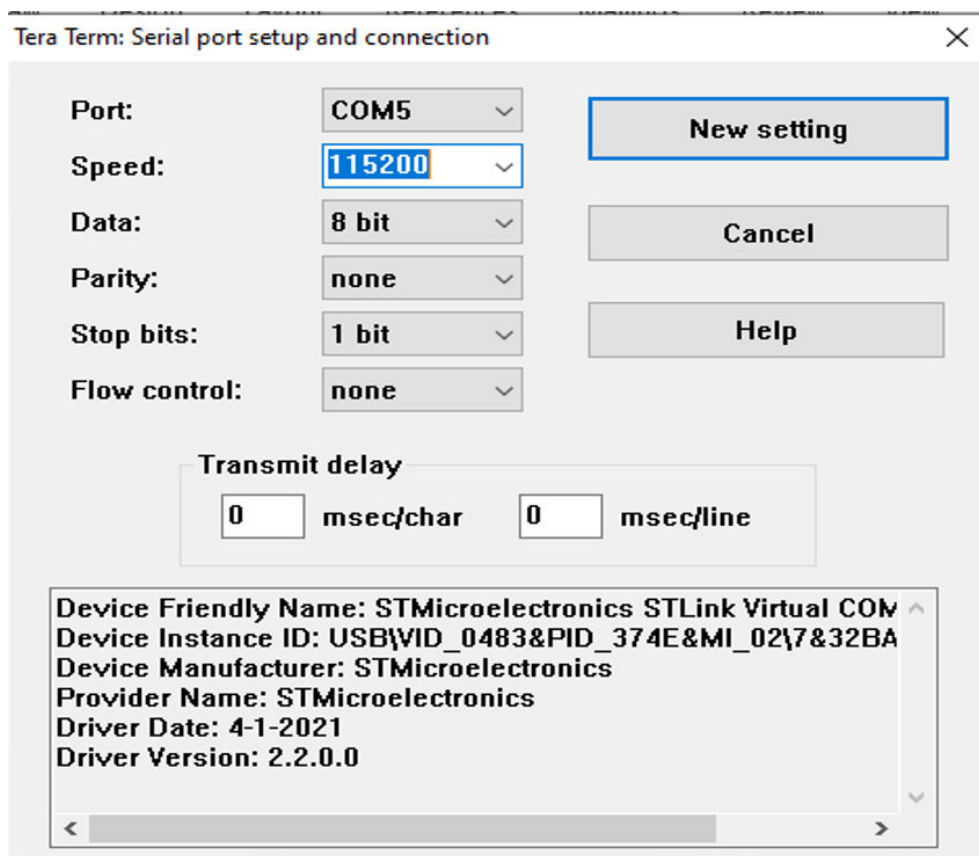
☒ Serial

Port: COM5: STMicroelectronics STLink Virtual COM Port

OK Cancel Help

- a. Serial port setup and connection:
- b. COM port setting:
 - Baud rate: 115200
 - Data: 8 bit
 - Parity: none
 - Stop bits: 1 bit

Figure 14. Tera term settings – baud rate



Tera Term: Serial port setup and connection

Port: COM5

Speed: 115200

Data: 8 bit

Parity: none

Stop bits: 1 bit

Flow control: none

New setting

Cancel

Help

Transmit delay

0 msec/char 0 msec/line

Device Friendly Name: STMicroelectronics STLink Virtual COM
 Device Instance ID: USB\VID_0483&PID_374E&MI_02\7&32BA
 Device Manufacturer: STMicroelectronics
 Provider Name: STMicroelectronics
 Driver Date: 4-1-2021
 Driver Version: 2.2.0.0

- a. Terminal setup:
- b. Receiver: CR+LF
- c. Transmit: CR+LF
- d. Local Echo: enable
- e. Auto switch: enable

Figure 15. Tera term settings – terminal setup

- Step 5.** Press the black reset button (B2).
Logs appear on the Tera Term and show if the test is passed or failed.

4.5 Test results

After flashing the binary to the STM32 Nucleo development board, apart from the test case specific results, check also the green LED on the [X-NUCLEO-PGEEZ1](#) EEPROM expansion board:

- if the LED is on and is stable, the board is correctly working;
- if the LED starts toggling, the board is not working properly.

Note:

- For [X-NUCLEO-EEPRMA2](#), refer terminal logs for test results
- For [X-NUCLEO-EEICA1](#), check the LED on the NUCLEO development board

5 Test cases

5.1 X-NUCLEO-EEICA1

5.1.1 Test case 1

With default factory setting, the device address in CDA register of both M24256E and M24M01E is 0x00. The Bit1 in CDA register of M24256E is C0 but for M24M01E it is "Dont Care". This property can be leveraged to select M24M01E EEPROM initially and change its CDA register value to 0x04 to differentiate between M24256E and M24M01E EEPROM.

5.1.1.1 **SETUP CDA REGISTER**

- Step 1.** Initially, attempt to read CDA register value of M24M01E with device select code as 0xB2. Only M24M01E device on the bus should respond.
- Step 2.** Update the CDA register value of M24M01E to 0x04.
- Step 3.** If the CDA register value of M24M01E has been set already, check the CDA address for M24M01E with device select code as 0xB4.

5.1.1.2 **Test acceptance criteria**

- The CDA register value of M24M01E should be updated to 0x04

Figure 16. Docklight screen showing SETUP CDA REGISTER

```
*****
          SETUP CDA REGISTER M24M01E
*****
__Writing CDA Val_M24M01 <C1 Bit Set>
Updated M24M01 CDA: 0x04
```

5.1.2 Test case 2

5.1.2.1 **READ CDA REGISTER**

Read CDA register values of M24256E-F and M24M01E-F EEPROM

- Step 1.** Read CDA register of M24256E
- Step 2.** Print CDA register value read for M24256E in above step
- Step 3.** Read CDA register of M24M01E
- Step 4.** Print CDA register value read for M24M01E in above step

5.1.2.2 **Test acceptance criteria**

- CDA register value for M24256E should be 0x00
- CDA register value for M24M01E should be 0x04

Figure 17. Docklight screen showing READ CDA REGISTER

```
*****
          READ CDA REGISTER
*****
M24256E CDA Register: 0x00
M24M01E CDA Register: 0x04
```

5.1.3 Test case 3

5.1.3.1 **WRITE CDA REGISTER**

Write CDA register value of M24256E-F

- Step 1.** Read CDA register of M24256E
- Step 2.** Print CDA register value read for M24256E in above step
- Step 3.** Write CDA register of M24256E with value 0x08
- Step 4.** Clear CDA register value of M24256E to 0x00

5.1.3.2 **Test acceptance criteria**

- Initial, CDA register value for M24256E should be 0x00
- CDA register value for M24256E should be updated to 0x08
- CDA register value for M24256E should reset to 0x00

Figure 18. Docklight screen showing WRITE CDA REGISTER

```
*****
                        WRITE CDA REGISTER
*****
M24256E CDA Register: 0x00
Setting C2 (Bit 3) of CDA register in M24256E . . .
Updated M24256E CDA Register: 0x08
Clearing C2 (Bit 3) of CDA register in M24256E . . .
Updated M24256E CDA Register: 0x00
```

5.1.4 **Test case 4**

5.1.4.1 **READ DTI REGISTER only for M24M01E**

Read DTI register of M24M01E

- Step 1.** Read DTI register of M24M01E
- Step 2.** Print DTI register value read for M24M01E in above step

5.1.4.2 **Test acceptance criteria**

- DTI register value for M24M01E read is 0xB1.

Figure 19. Docklight screen showing READ DTI REGISTER for M24M01E

```
*****
                        READ DTI REGISTER only for M24M01E
*****
M24M01E DTI Register: 0xB1
```

5.1.5 **Test case 5**

5.1.5.1 **TEST M24256E ID PAGE**

Test ID Page of M24256E

- Step 1.** Read and print ID Page of M24256E
- Step 2.** Write 0x55 to complete ID Page and print ID Page of M24256E
- Step 3.** Compare if data written successfully and print the result
- Step 4.** Clear ID Page with default data 0xFF

5.1.5.2 Test acceptance criteria

- Initially, the ID Page data should be 0xFF
- After ID Page is updated, the check should confirm data is written successfully
- Clear ID Page and next time when ID Page is read, it should be 0xFF

Figure 20. Docklight screen showing TEST M24256E ID PAGE

```

*****
TEST M24256E ID PAGE
*****
Read ID Page M24256E:
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF

Writing data 0x55 to complete ID page of M2456E
0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55
0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55
0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55 0x55
0x55 0x55 0x55 0x55

All data to M24256E ID Page written successfully!
Cleared ID page of M24256E to 0xFF
  
```

5.1.6 Test case 6

5.1.6.1 TEST M24M01E ID PAGE

Test ID Page of M24M01E

- Step 1.** Read and print ID Page of M24M01E
- Step 2.** Write 0xAA to complete ID Page and print ID Page of M24M01E
- Step 3.** Compare if data written successfully and print the result
- Step 4.** Clear ID Page with default data 0xFF

5.1.6.2

- Initially, the ID Page data should be 0xFF
- After ID Page is updated, the check should confirm data is written successfully
- Clear ID Page and next time when ID Page is read, it should be 0xFF

Figure 21. Docklight screen showing TEST M24M01E ID PAGE

[illegible]

5.1.7

5.1.7.1

- Step 1.** Read SWP register of M24M01E
- Step 2.** Modify SWP register to 0x0E to lock complete memory
- Step 3.** Attempt to write memory
- Step 4.** Read memory to check data is not written
- Step 5.** Modify SWP register again to 0x00
- Step 6.** Read SWP register

5.1.7.2 **Test acceptance criteria**

- Initially, the SWP register should read 0x00
- After SWP register is updated to 0x0E, write to complete memory must be disabled
- This is confirmed when attempt to write data to memory is not successful
- SWP register should read 0x00 when modified

Figure 22. Docklight screen showing TEST SWP REGISTER only for M24M01E

```

*****
      TEST SWP REGISTER only for M24M01E
*****
M24M01E SWP Register: 0x00
Modified M24M01E SWP Register to: 0x0E. Complete memory locked.
Attempt to write to memory from Address 0x00 with 10 bytes of data. . .
Write failed as expected
Checked, data not written to memory as expected
Updated M24M01E SWP Register: 0x00
  
```

5.1.8 **Test case 8**

5.1.8.1 **TEST M24256E MEMORY**

Write 1024 bytes to memory from Address 0x65

- Step 1.** Read 20 pages in memory
- Step 2.** Write 1024 bytes to memory starting from address 0x65
- Step 3.** Read 1024 bytes from memory starting from address 0x65
- Step 4.** Check memory contents are same as test data
- Step 5.** Clear memory contents to 0xFF

5.1.8.2 Test acceptance criteria

- The data should be written correctly over pages
- The memory pages should be cleared to 0xFF

Figure 23. Docklight screen showing TEST M24256E MEMORY

```

ASCI  HEX  Decal  Binary  Communicati
*****
TEST M24256E MEMORY
*****

Write 1024 bytes to memory from Address 0x65

Write Success
Read M24256E 20 Memory Pages:

Page 1
  Address 0x00 to 0x3F:
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

Page 2
  Address 0x40 to 0x7F:
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85

Page 3
  Address 0x80 to 0xBF:
0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85
0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85
0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85

Page 4
  Address 0xC0 to 0xFF:
0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85
0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85
0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85 0x85

```

5.1.9 Test case 9

5.1.9.1 TEST M24M01E MEMORY

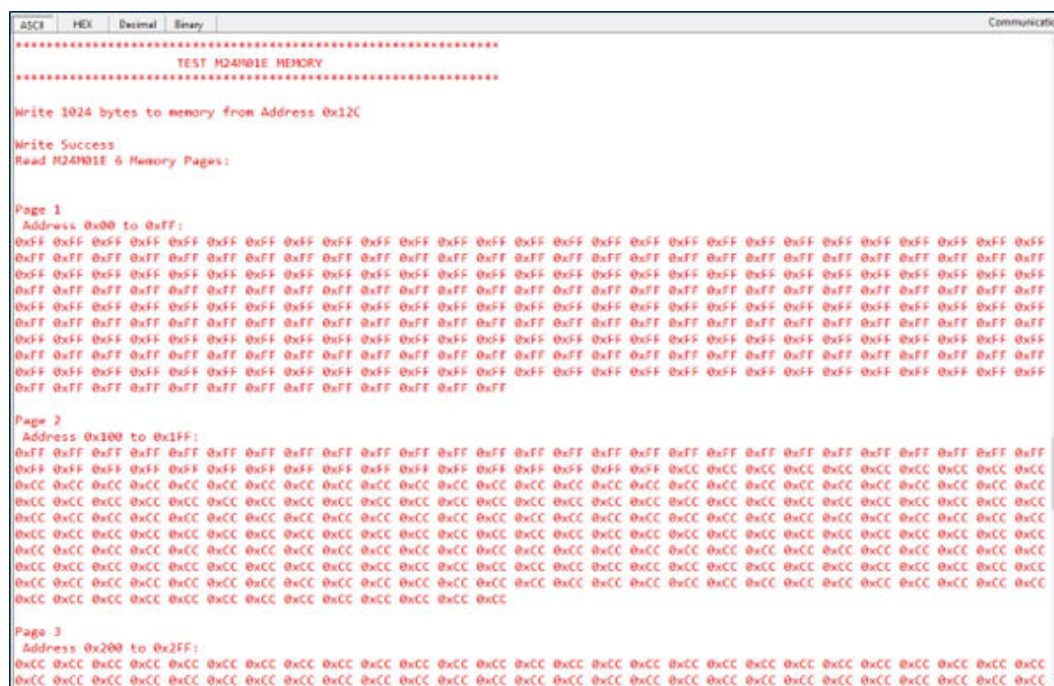
Write 1024 bytes to memory from Address 0x12C

- Step 1.** Read 6 pages in memory
- Step 2.** Write 1024 bytes to memory starting from address 0x12C
- Step 3.** Read 1024 bytes from memory starting from address 0x12C
- Step 4.** Check memory contents are same as test data
- Step 5.** Clear memory contents to 0xFF

5.1.9.2 Test acceptance criteria

- The data should be written correctly over pages
- The memory pages should be cleared to 0xFF

Figure 24. Docklight screen showing TEST M24M01E MEMORY

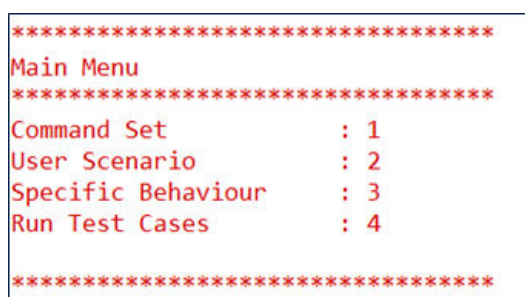


5.2 X-NUCLEO-PGEEZ1

The example application presents interactive menu-based options that can be used to explore the features of X-NUCLEO-PGEEZ1.

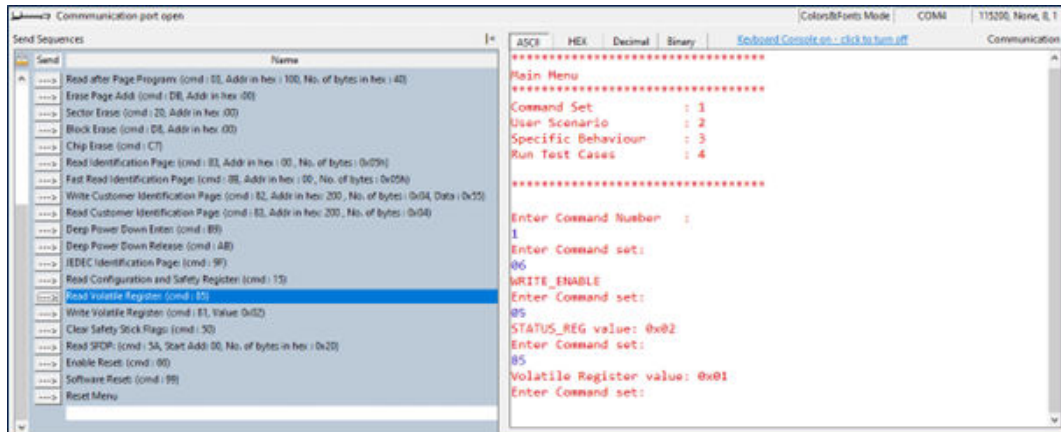
Below figure shows the Main Menu that appears when the example application is run. Users can input their numerical choices and execute the test cases.

Figure 25. Docklight screen showing Main Menu in example application for X-NUCLEO-PGEEZ1



5.2.1 Command set

The user can execute the “Command set” option from “Main menu” by selecting the first option. “Command set” enables the use of AT commands to be executed. Users can use the docklight application provided in the utilities folder of the **X-CUBE-EEPRMA1** package to run the basic functionalities supported by **X-NUCLEO-PGEEZ1**.

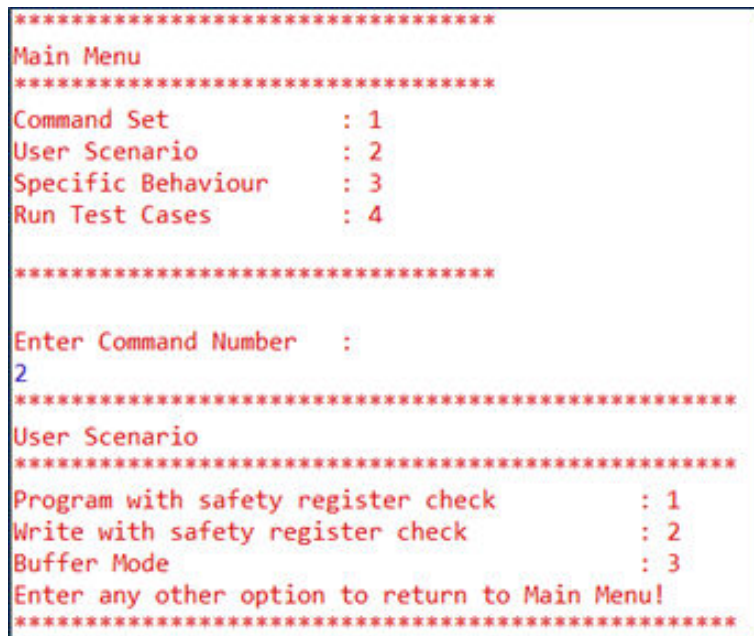
Figure 26. AT command support in Docklight


To return to a main menu again, the user can send a “Reset menu” AT command sequence from docklight. The “Reset menu” option is only to be used to return from “Command set” to “Main menu”.

5.2.2 User Scenario

User can execute “User Scenario” option from “Main menu” by selecting second option. “User scenario” provides the following features:

1. Program with safety register check
2. Write with safety register check
3. Buffer Mode

Figure 27. Docklight screen showing User Scenario menu options


5.2.2.1 Program with safety register check

The safety register in M95P32 EEPROM provides information on the device's status. This test checks the safety register when performing page program.

- Step 1.** Poll on Status register to check Write In Progress (WIP) bit is not set
- Step 2.** Poll on Safety register to check Power Up Flag (PUF) bit is not set
- Step 3.** Write Enable

- Step 4.** Erase Page
- Step 5.** Poll on Status Register until WIP bit is cleared
- Step 6.** Check Safety register that Erase Flag (ERF) bit is not set
- Step 7.** Check page was erased successfully by reading page data
- Step 8.** Read Safety register to ensure no Error Correction Code (ECC) issue
- Step 9.** Write Enable
- Step 10.** Perform page program operation
- Step 11.** Check Safety register that Program Flag (PRF) bit is not set
- Step 12.** Check page was programmed successfully by reading page data
- Step 13.** Read Safety register to ensure no Error Correction Code (ECC) issue

Figure 28. Docklight screen showing Page Program with Safety Register Check

```

*****
User Scenario
*****
Program with safety register check      : 1
Write with safety register check       : 2
Buffer Mode                           : 3
Enter any other option to return to Main Menu!
*****

Enter Command Number :
1
Program with Safety Register Check:
Power UP Done.
Power UP OK.
Page Erase Done.
Page Erase Done without issues.
Page Erased Correctly.
NO ECC detected during Erase Check.
Page Program Done without issues.
Page Programmed Correctly.
NO ECC detected during Page Program Check.
  
```

5.2.2.2 *Write with safety register check*

The safety register in M95P32 EEPROM provides information on the device's status. This test checks the safety register when performing page write.

- Step 1.** Poll on Status register to check WIP bit is not set
- Step 2.** Poll on Safety register to check PUF bit is not set
- Step 3.** Write Enable
- Step 4.** Perform page write operation
- Step 5.** Check Safety register that ERF and PRF flags are not set
- Step 6.** Check page write was done successfully by reading page data
- Step 7.** Read Safety register to ensure no Error Correction Code (ECC) issue
- Step 8.** Again, perform page write operation with some different data
- Step 9.** Check Safety register that ERF and PRF flags are not set
- Step 10.** Check page write was done successfully by reading page data

Step 11. Read Safety register to ensure no Error Correction Code (ECC) issue

Figure 29. Docklight screen showing Page Write with Safety Register Check

```

*****
User Scenario
*****
Program with safety register check      : 1
Write with safety register check       : 2
Buffer Mode                           : 3
Enter any other option to return to Main Menu!
*****

Enter Command Number :
2
Write with Safety Register Check:
Power UP Done.
Power UP OK.
Page Write Done without issues.
Page Written Correctly.
NO ECC detected during Page Write Check.
Page Write Done without issues.
Page Written Correctly.
NO ECC detected during Page Write Check.

```

5.2.2.3 **Buffer Mode**

- Step 1.** Poll on Status register to check WIP bit is not set
- Step 2.** Poll on Safety register to check PUF bit is not set
- Step 3.** Write Enable
- Step 4.** Perform erase sector operation
- Step 5.** Read safety register to ensure no issue

Step 6. Use specific buffer mode API (*PageProgram_with_BufferLoad*) to program pages with buffer load operation

Note: Refer Technical note [TN1397](#) for further information.

Figure 30. Docklight screen showing Page Program with Buffer Load operation

```

*****
User Scenario
*****
Program with safety register check      : 1
Write with safety register check       : 2
Buffer Mode                           : 3
Enter any other option to return to Main Menu!
*****

Enter Command Number :
3
Buffer Mode:
Power UP Done.
Power UP OK.
Safety Register status OK.
Data is in erased state. OK.
Data is in erased state. OK.
NO ECC correction detected. OK.
Last Page program is managed outside of an array.
Safety Register status OK.
BUFFEN bit is 0. OK.
Data programmed correctly.
Data programmed correctly.
NO ECC correction detected. OK.
  
```

5.2.3 Specific behaviour

User can execute “Specific Behaviour” option from “Main Menu” by selecting 3rd option. “Specific Behaviour” provides the following features:

- 16 word align
- Write in protected area

Figure 31. Docklight screen showing Specific Behaviour menu options

```

*****
Main Menu
*****
Command Set           : 1
User Scenario         : 2
Specific Behaviour    : 3
Run Test Cases       : 4
*****

Enter Command Number  :
3
*****
Specific Behaviour
*****
16 word align         : 1
Write in protected area : 2
Enter any other option to return to Main Menu!
*****

Enter Command Number :

```

5.2.3.1 16 word align

M95P32 Page EEPROM is a memory device with a page architecture. This means that pages are the smallest erasable blocks of memory. Each page is made up of words and each word is composed of 16 bytes. Refer AN5747 for further information.

“16 word align” test case shows this property of M95P32 page EEPROM. When a byte of a word is programmed, the operations are word aligned. If the user tries to program another byte in the same word again, relevant error bits will be set in status register.

- Step 1.** Poll on Status register to check WIP bit is not set
- Step 2.** Poll on Safety register to check PUF bit is not set
- Step 3.** Write Enable
- Step 4.** Perform page erase operation
- Step 5.** Read safety register to ensure no issue
- Step 6.** Check page erase was done successfully by reading page data
- Step 7.** Read Safety register to ensure no Error Correction Code (ECC) issue
- Step 8.** Write Enable
- Step 9.** Page program 1 byte on first page address
- Step 10.** Poll on Status register until WIP bit is cleared
- Step 11.** Read safety register to ensure no issue
- Step 12.** Read 1 byte data programmed in above step to confirm data was written successfully
- Step 13.** Read Safety register to ensure no Error Correction Code (ECC) issue
- Step 14.** Write Enable
- Step 15.** Page program 1 byte on 16th byte of first page
- Step 16.** Poll on Status register until WIP bit is cleared

- Step 17.** Read safety register to ensure no issue
- Step 18.** Read 1 byte data programmed in above step to confirm data was written successfully
- Step 19.** Read Safety register to ensure no Error Correction Code (ECC) issue
- Step 20.** Write Enable
- Step 21.** Page program 1 byte on 17th byte of first page
- Step 22.** Poll on Status register until WIP bit is cleared
- Step 23.** Read safety register, Program Flag (PRF) bit should be set
- Step 24.** Clear safety register
- Step 25.** Read 1 byte data programmed in above step to confirm data was not written
- Step 26.** Read Safety register to ensure no Error Correction Code (ECC) issue

Figure 32. Docklight screen showing 16 word align concept

```

*****
Specific Behaviour
*****
16 word align      : 1
Write in protected area : 2
Enter any other option to return to Main Menu!
*****

Enter Command Number :
1
16 word align:
Power UP Done.
Power UP OK.
Page Erased from address 0x00. Safety Register Value is: 0x0
PAMAF and ERF bit not set. OK.
Erased page data is in erased state <0xFF>. OK.
Safety Register Value is: 0x0
No ECC flag set. OK.

Programmed 1 byte from address 0x00 with value 0x55. Safety Register Value is: 0x0
No flag set. OK.
Read 1 byte from address 0x00. Safety Register Value is: 0x0
No ECC flag set. OK.

Programmed 1 byte from address 0x100 with value 0xA5. Safety Register Value is: 0x0
No Flag set. OK.
Data at address 0x100 is: 0xA5
Page Program successful
Read 1 byte from address 0x100. Safety Register Value is: 0x0
No ECC flag set. OK.

Programmed 1 byte from address 0x101 with value 0x55. Safety Register Value is: 0x10
PRF Flag set as expected!
Data at address 0x101 is: 0xFF
Page Program NOT successful
Safety Register Value is: 0x0
No ECC flag set. OK.
  
```

5.2.3.2 *Write in protected area*

Test case to check that write operations to memory are disabled if the chip is write protected.

- Step 1.** Poll on Status register to check WIP bit is not set
- Step 2.** Poll on Safety register to check PUF bit is not set

- Step 3.** Write Enable
- Step 4.** Write status register to 0x1C to write protect full EEPROM memory
- Step 5.** Read status register to confirm the configuration
- Step 6.** Read random data
- Step 7.** Write Enable
- Step 8.** Write data at the same random location
- Step 9.** Call Write Disable explicitly
- Step 10.** Check safety register, Protected array modify attempt flag (PAMAF) bit should set
- Step 11.** Clear safety register
- Step 12.** Read data at above location, data should be same as in Step 6 above
- Step 13.** Read Safety register to ensure no Error Correction Code (ECC) issue
- Step 14.** Write Enable
- Step 15.** Clear status register to 0x00
- Step 16.** Read status register to confirm configuration

Figure 33. Docklight screen showing Write in protected area operation

```

*****
Specific Behaviour
*****
16 word align          : 1
Write in protected area : 2
Enter any other option to return to Main Menu!
*****

Enter Command Number :
2
Write in protected area
Power UP Done.
Power UP OK.
Status Register value is: 0x1C
Chip fully protected.
Data stored at address 0x600 is 0xFF
Attempt to write 0x87 at address 0x600. Safety Register Value is: 0xB0
PAMAF bit set, Modify operation to a protected memory area has been attempted.
Data stored at address 0x600 is 0xFF
NO ECC correction detected. OK.
Status Register value : 0x0
  
```

5.2.4 Run test cases

5.2.4.1 Test case 1

Perform this test to verify the value of the status register of the M95P32.

- Step 1.** Read the default value of the status register.
- Step 2.** Write Enable.
- Step 3.** Read the status register value after the Write Enable operation.
- Step 4.** Write Disable.
- Step 5.** Read the status register value after the Write Disable operation.

5.2.4.1.1 Test acceptance criteria

- The default status register value should be 0x00 (0xFF is failed)
- The status register value after the Write Enable operation should be 0x02
- The status register value after the Write Disable operation should be 0x00

If the status register value after each step is the same as the accepted one, the test is passed.

5.2.4.1.2 Test results

Table 4. Test 1 results

Board	Acceptance criteria	Failed criteria	Ideal output
X-NUCLEO-PGEEZ1	• Status register value: 00 • Status register value after Write Enable: 02 • Status register value after Write Disable: 00	• Status register value: 0xff • Status register value after Write Enable is different from 0x02 • Status register value after Write Disable is different from 0x02	• Status register value: 00 • Status register value: 02 • Status register value: 00

Once the status register value is the one described in [Section 5.2.4.1.1: Test acceptance criteria](#), the test ends successfully for the X-NUCLEO-PGEEZ1 board.

Figure 34. Tera Term screen showing test case 1 results

```

_____Test 1 Starts_____

      Test case to verify Write Enable and Write Disable
00      : Passed
02      : Passed
00      : Passed

*****Test 1 Passed*****
  
```

5.2.4.2 Test case 2

Perform this test to verify the page Write and Read operations.

- Step 1.** Write Enable.
- Step 2.** Write the first 100 bytes of a page to 0x0f at address: 0x00 (page number 1).
- Step 3.** Verify if the page bytes are updated to 0x0f or not.
- Step 4.** Read 200 bytes of the page at address: 0x00 using the single SPI.
- Step 5.** Read 200 bytes of the page at address: 0x00 using the fast read single SPI.
- Step 6.** Read 200 bytes of the page at address: 0x00 using the fast read dual SPI.
- Step 7.** Read 200 bytes of the page at address: 0x00 using the fast read quad SPI.

5.2.4.2.1 Test acceptance criteria

After the Page Write operation and reading the data using read operations (single SPI, fast single SPI, fast dual SPI, fast quad SPI):

- the first 100 bytes should have the value 0x0f;
- after 100 bytes, all bytes should have the default value 0xff.

If the above two conditions are met, the test is passed.

5.2.4.2.2 Test results

Table 5. Test 2 results

Board	Acceptance criteria	Failed criteria	Ideal output
X-NUCLEO-PGEEZ1	The first 100 bytes of data read should have the value of 0x0f and the next 100 bytes of data should have the default value of 0xff	If any of the first 100 bytes have a value different from 0x0f If any of the next 100 bytes have a value different from 0xff	The figure below showing the data read after using Write and Read operations

Once the data have been read as described in [Section 5.2.4.2.1: Test acceptance criteria](#), the test ends successfully for the X-NUCLEO-PGEEZ1 board.

Figure 35. Tera Term screen showing test case 2 results

```

_____Test 2 Starts_____
Test case to verify Page Write and Read

Write Page                               : Passed

Read data in Single SPI                  : Passed
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

Read data Fast in Single SPI              : Passed
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

Read data Fast in Dual SPI                : Passed
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

Read data Fast in Quad SPI                : Passed
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f 0f
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

*****Test 2 Passed*****

```

5.2.4.3 Test case 3

Perform this test to verify the page erase operation (erasing 512 bytes).

Step 1. Write Enable.

Step 2. Erase the page (512 bytes) at address 0x00.

Step 3. Verify that, after the page erase, all the bytes are at their default value of 0xff.

5.2.4.3.1 Test acceptance criteria

After the page erase operation and reading data using the read operation:

- all 512 bytes from the starting address of the page should have the default value of 0xff

If the above condition is met, the test is passed.

5.2.4.3.2 Test results

Table 6. Test 3 results

Board	Acceptance criteria	Failed criteria	Ideal output
X-NUCLEO-PGEEZ1	All 512 bytes should have the default value of 0xff	If any of the 512 bytes from the starting address 0x00 has a value different from 0xff	The figure below showing the data read after using the page erase operation

Once the data have been read as described in [Section 5.2.4.3.1: Test acceptance criteria](#), the test ends successfully for the X-NUCLEO-PGEEZ1 board.

Figure 36. Tera Term screen showing test case 3 results

```

_____Test 3 Starts_____

      Test Case to verify Page Erase

Data after Page Erase

ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

Page Erase                               : Passed

*****Test 3 Passed*****
  
```

5.2.4.4 Test case 4

Perform this test to verify the sector erase operation (erasing 1 sector = 8 pages).

Step 1. Write Enable.

Step 2. Writing the first 100 bytes of a page to 0x0f at address: 0x400 (page number: 3).

Step 3. Write Enable.

Step 4. Erase a sector (8 pages) at address 0x00.

Step 5. Verify that, after the sector erase, all bytes of the first 8 pages have a value of 0xff.

Note: In this step, we have read just the bytes of the third page (starting address = 0x400) after the sector erase operation for verification.

5.2.4.4.1 Test acceptance criteria

After the sector erase operation and reading data using the read operation:

- all 512 bytes of the first 8 pages from the starting address of sector 0x00 should have the default value of 0xff.

Note: We have verified just the bytes of the page at the address = 0x400.

If above condition is met, the test is passed.

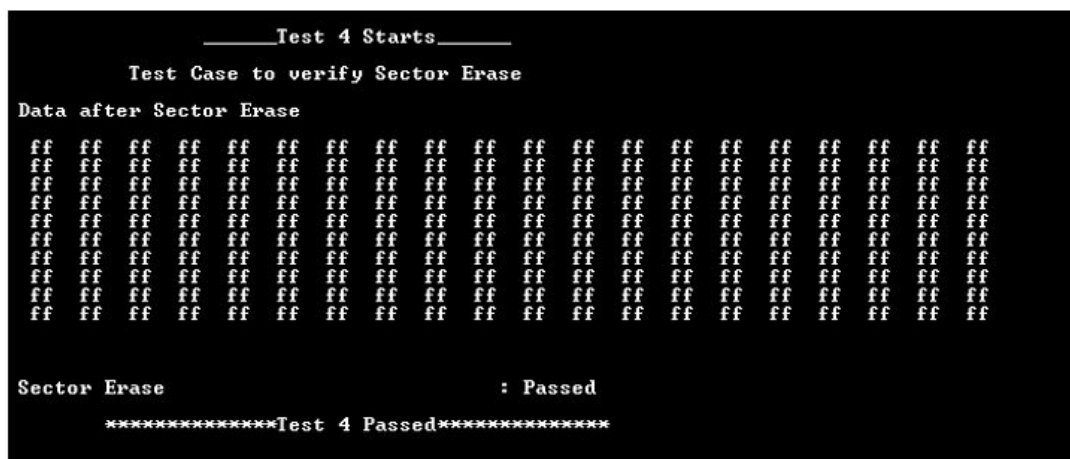
5.2.4.4.2 Test results

Table 7. Test 4 results

Board	Acceptance criteria	Failed criteria	Ideal output
X-NUCLEO-PGEEZ1	All 512 bytes of the first 8 pages should have the default value of 0xff	If any of the 8 pages starting from the address 0x00 has a value different from 0xff	The figure below showing the data read after using the sector erase operation

Once the data have been read as described in [Section 5.2.4.4.1: Test acceptance criteria](#), the test ends successfully for the **X-NUCLEO-PGEEZ1** board.

Figure 37. Tera Term screen showing test case 4 results



5.2.4.5 Test case 5

Perform this test to verify the block erase operation (erasing 1 block = 128 pages).

- Step 1.** Write Enable.
- Step 2.** Write the first 100 bytes of a page to 0x0f at address: 0x1800 (page number: 10).
- Step 3.** Write Enable.
- Step 4.** Erase a block (128 pages) at address 0x00.
- Step 5.** Verify that, after the sector erase, all bytes of the first 128 pages have a value of 0xff.

5.2.4.5.1 Test acceptance criteria

After the sector erase operation and reading data using the read operation:

- all 512 bytes of the first 128 pages from the starting address of block 0x00 should have the default value of 0xff.

Note: We have verified just the bytes of the page at the address = 0x1800.

If the above condition is met, the test is passed.

5.2.4.5.2 Test results

Table 8. Test 5 results

Board	Acceptance criteria	Failed criteria	Ideal output
X-NUCLEO-PGEEZ1	All 512 bytes of the 128 pages should have the default value of 0xff	If any byte of the 128 pages starting from the address 0x00 has a value different from 0xff	The figure below showing the data read after using the block erase operation

Once the data have been read as described in [Section 5.2.4.5.1: Test acceptance criteria](#), the test ends successfully for the X-NUCLEO-PGEEZ1 board.

Figure 38. Tera Term screen showing test case 5 results

```

_____ Test 4 Starts _____
      Test Case to verify Sector Erase
Data after Sector Erase
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff

Sector Erase                      : Passed
*****Test 4 Passed*****
  
```

5.2.4.6 Test case 6

Perform this test to verify the page program with the buffer mode operation.

- Step 1.** Write Enable.
- Step 2.** Write the volatile register to enable the buffer mode (0x81 02).
- Step 3.** Write Enable.
- Step 4.** Program 1024 bytes of a page to 0x09 starting from the address: 0x00.
- Step 5.** Write the volatile register to exit the buffer mode.
- Step 6.** Read 1024 bytes from the starting address 0x00.
- Step 7.** Verify that, after the page program in the buffer mode, the 1024 bytes from the address 0x00 have the value 0x09.

5.2.4.6.1 Test acceptance criteria

After the page program with the buffer load operation and reading data using the read operation:

- all 1024 bytes from the starting address of 0x00 should have the value: 0x09

If the above condition is met, the test is passed.

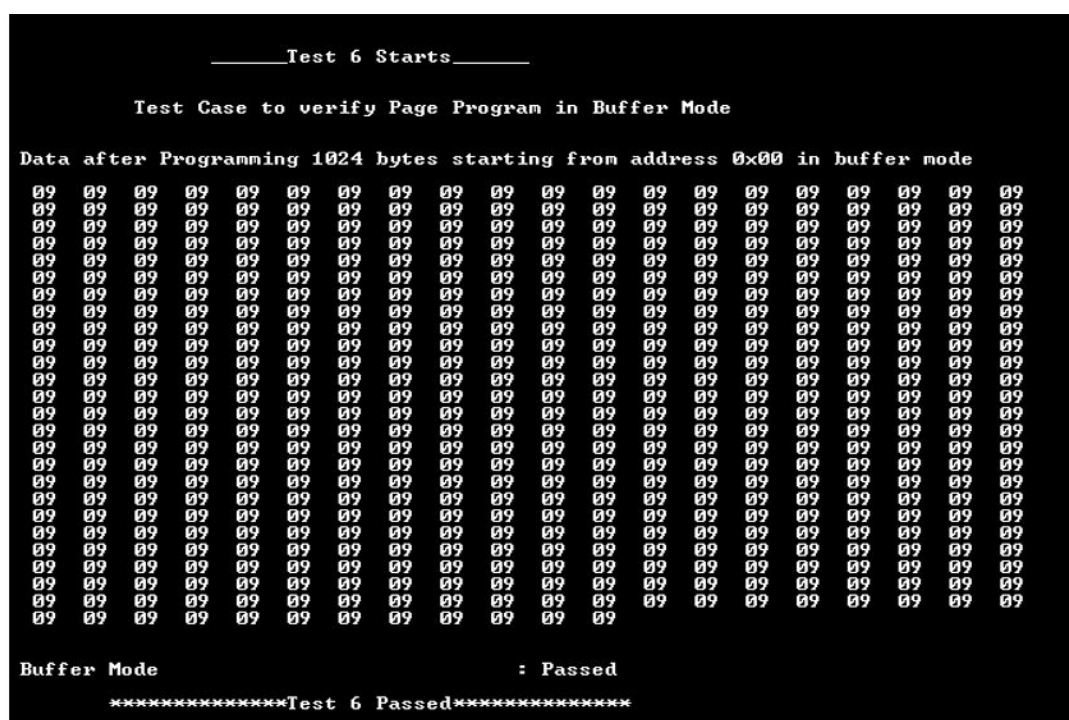
5.2.4.6.2 Test results

Table 9. Test 6 results

Board	Acceptance criteria	Failed criteria	Ideal output
X-NUCLEO-PGEEZ1	All 1024 bytes starting from the address 0x00 must have a value equal to 0x09	If any of the next 1024 bytes from the starting address of 0x00 has a value different from 0x09, the test is failed	The figure below showing the data read after using page program with the buffer operation

Once the data have been read as described in [Section 5.2.4.6.1: Test acceptance criteria](#), the test ends successfully for the **X-NUCLEO-PGEEZ1** board.

Figure 39. Tera Term screen showing test case 6 results



6 Test completion

Remove the USB cable from the STM32 Nucleo development board and proceed with the test on [X-NUCLEO-EEICA1](#), [X-NUCLEO-EEPRMA2](#) or [X-NUCLEO-PGEEZ1](#) expansion boards.

You can use the featured STM32 Nucleo development boards without flashing the binaries on them again.

Revision history

Table 10. Document revision history

Date	Version	Changes
04-Oct-2018	1	Initial release.
20-Apr-2020	2	Updated Introduction, Section 2.1 Overview, Figure 1. X-CUBE-EEPRMA1 software architecture, Section 3.2 Hardware setup, Section 3.2.1 STM32 Nucleo and X-NUCLEO-EEPRMA1/X-NUCLEOEEPRMA2 expansion board setup and Section 3.3 Software setup. Added Section 3.1.3 X-NUCLEO-EEPRMA2 expansion board. Removed Section 2. What is STM32Cube and replaced it by a link in the Introduction.
13-Oct-2020	3	Updated Section 2.1 Overview and Section 3.3 Software setup.
16-Nov-2022	4	Updated introduction, Section 2.1: Overview, Section 2.2: Architecture, Section 2.3: Folder structure, Section 3.2: Hardware setup, Section 3.2.1: STM32 Nucleo and X-NUCLEO-EEICA1/X-NUCLEOEEPRMA2/X-NUCLEO-PGEEZ1 expansion board setup, and Section 3.3: Software setup. Added Section 3.1.4: X-NUCLEO-PGEEZ1, Section 4: Test setup, Section 4.1: Firmware binary, Section 4.2: Test platforms (boards), Section 4.3: Precondition, Section 4.4: Test execution, Section 4.5: Test results, Section 5: Test cases, Section 5.2.4.1: Test case 1, Section 5.2.4.1.1: Test acceptance criteria, Section 5.2.4.1.2: Test results, Section 5.2.4.2: Test case 2, Section 5.2.4.2.1: Test acceptance criteria, Section 5.2.4.2.2: Test results, Section 5.2.4.3: Test case 3, Section 5.2.4.3.1: Test acceptance criteria, Section 5.2.4.4: Test case 4, Section 5.2.4.4.1: Test acceptance criteria, Section 5.2.4.4.2: Test results, Section 5.2.4.5: Test case 5, Section 5.2.4.5.1: Test acceptance criteria, Section 5.2.4.5.2: Test results, Section 5.2.4.6: Test case 6, Section 5.2.4.6.1: Test acceptance criteria, Section 5.2.4.6.2: Test results, Section 6: Test completion, and AT commands.
19-Dec-2023	5	Updated Section Introduction, Section 1: Acronyms and abbreviations, Section 1: Acronyms and abbreviations, Section 3: System setup guide, Section 4: Test setup, Section 5: Test cases and Section 6: Test completion.
24-Jun-2025	6	Updated Section 2.1: Overview, Section 2.3: Folder structure, Section 3.2: Hardware setup, Section 3.2.1: STM32 Nucleo and X-NUCLEO-EEICA1/X-NUCLEO-EEPRMA2/X-NUCLEO-PGEEZ1 expansion board setup, Section 4.2: Test platforms (boards), Section 4.3: Precondition. Replaced NUCLEO-H743ZI with NUCLEO-H753ZI.

Contents

1	Acronyms and abbreviations	2
2	X-CUBE-EEPRMA1 software expansion for STM32Cube	3
2.1	Overview	3
2.2	Architecture	3
2.3	Folder structure	4
2.4	APIs	4
3	System setup guide	5
3.1	Hardware description	5
3.1.1	STM32 Nucleo	5
3.1.2	X-NUCLEO-EEICA1 expansion board	5
3.1.3	X-NUCLEO-EEPRMA2 expansion board	6
3.1.4	X-NUCLEO-PGEEZ1 expansion board	6
3.2	Hardware setup	7
3.2.1	STM32 Nucleo and X-NUCLEO-EEICA1/X-NUCLEO-EEPRMA2/X-NUCLEO-PGEEZ1 expansion board setup	7
3.3	Software setup	9
4	Test setup	12
4.1	Firmware binary	12
4.2	Test platforms (boards)	12
4.3	Precondition	12
4.4	Test execution	13
4.5	Test results	16
5	Test cases	17
5.1	X-NUCLEO-EEICA1	17
5.1.1	Test case 1	17
5.1.2	Test case 2	17
5.1.3	Test case 3	17
5.1.4	Test case 4	18
5.1.5	Test case 5	18
5.1.6	Test case 6	19
5.1.7	Test case 7	20
5.1.8	Test case 8	21
5.1.9	Test case 9	22
5.2	X-NUCLEO-PGEEZ1	23
5.2.1	Command set	23

5.2.2	User Scenario	24
5.2.3	Specific behaviour.	27
5.2.4	Run test cases	30
6	Test completion	37
	Revision history	38

List of tables

Table 1.	List of acronyms	2
Table 2.	X-NUCLEO-PGEEZ1 jumper settings	8
Table 3.	X-NUCLEO-PGEEZ1 jumper settings	13
Table 4.	Test 1 results.	31
Table 5.	Test 2 results.	32
Table 6.	Test 3 results.	33
Table 7.	Test 4 results.	34
Table 8.	Test 5 results.	35
Table 9.	Test 6 results.	36
Table 10.	Document revision history	38

List of figures

Figure 1.	X-CUBE-EEPRMA1 software architecture	3
Figure 2.	X-CUBE-EEPRMA1 package folder structure	4
Figure 3.	STM32 Nucleo board	5
Figure 4.	X-NUCLEO-EEICA1 expansion board	6
Figure 5.	X-NUCLEO-EEPRMA2 expansion board	6
Figure 6.	X-NUCLEO-PGEEZ1 expansion board	7
Figure 7.	X-NUCLEO-EEPRMA2 expansion board plugged to an STM32 Nucleo board	8
Figure 8.	X-NUCLEO-PGEEZ1 expansion board plugged to an STM32 Nucleo board	8
Figure 9.	Console log for X-NUCLEO-EEPRMA2	10
Figure 10.	Console log for X-NUCLEO-PGEEZ1	11
Figure 11.	Console log for X-NUCLEO-EEICA1	11
Figure 12.	Jumper settings (for X-NUCLEO-PGEEZ1).	13
Figure 13.	Tera term settings – COM port	14
Figure 14.	Tera term settings – baud rate	15
Figure 15.	Tera term settings – terminal setup	16
Figure 16.	Docklight screen showing SETUP CDA REGISTER	17
Figure 17.	Docklight screen showing READ CDA REGISTER	17
Figure 18.	Docklight screen showing WRITE CDA REGISTER.	18
Figure 19.	Docklight screen showing READ DTI REGISTER for M24M01E	18
Figure 20.	Docklight screen showing TEST M24256E ID PAGE	19
Figure 21.	Docklight screen showing TEST M24M01E ID PAGE.	20
Figure 22.	Docklight screen showing TEST SWP REGISTER only for M24M01E	21
Figure 23.	Docklight screen showing TEST M24256E MEMORY	22
Figure 24.	Docklight screen showing TEST M24M01E MEMORY	23
Figure 25.	Docklight screen showing Main Menu in example application for X-NUCLEO-PGEEZ1	23
Figure 26.	AT command support in Docklight	24
Figure 27.	Docklight screen showing User Scenario menu options	24
Figure 28.	Docklight screen showing Page Program with Safety Register Check	25
Figure 29.	Docklight screen showing Page Write with Safety Register Check	26
Figure 30.	Docklight screen showing Page Program with Buffer Load operation	27
Figure 31.	Docklight screen showing Specific Behaviour menu options	28
Figure 32.	Docklight screen showing 16 word align concept.	29
Figure 33.	Docklight screen showing Write in protected area operation	30
Figure 34.	Tera Term screen showing test case 1 results.	31
Figure 35.	Tera Term screen showing test case 2 results.	32
Figure 36.	Tera Term screen showing test case 3 results.	33
Figure 37.	Tera Term screen showing test case 4 results.	34
Figure 38.	Tera Term screen showing test case 5 results.	35
Figure 39.	Tera Term screen showing test case 6 results.	36

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. 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.

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 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.

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.

© 2025 STMicroelectronics – All rights reserved