PM0047
Programming manual

STM8 Flash programming

Introduction

This manual describes how to program Flash memory of an STM8 microcontroller.

It is intended to provide information to programming tool manufacturers and to customers who want to implement programming by themselves on their production line.

For implementation details such as register or stack top address, refer to the product datasheet.

The **In-Circuit Programming (ICP)** method is used to update the entire contents of the Flash memory (including option bytes) while the user application is not running. It uses the **Single Wire interface Module (SWIM)** to communicate between the programming tool and the device.

In contrast to the ICP method, **In-Application Programming (IAP)** uses any communication interface supported by the microcontroller (I/Os, SPI, SCI, USB, CAN...). IAP has been implemented for users who want their application software to update itself by re-programming the Flash memory during execution. The main advantage of IAP is its ability to re-program the Flash memory when the chip has already been soldered on the application board and while the user application is running. Nevertheless, part of the Flash memory has to be previously programmed using ICP.

The STM8 family also contains a bootloader embedded in a ROM memory. Through this firmware the device memory can be re-programmed using a standard communication interface. This programming method is not described in this document.

Related document:

- STM8 SWIM communication protocol and debug module (UM0470)

Contents

GLOSSARY

This chapter gives a brief definition of acronyms and names used in this document:

- **Block:** A block is a set of bytes that can be programmed or erased in one single programming operation.
  The size of a block is 128 bytes. Operations that are available on a block are fast programming, erase, standard programming (which includes erase).
- **Boot area:** The boot area is a part of the Flash memory which is write protected in user mode (that is protected against modification by the user software). The boot area contents can be modified using the ICP (SWIM) mode. The boot area size is programmable by option byte by incrementing one page.
- **Bootloader:** The bootloader is an IAP application embedded in the system memory of the device. It is used to erase and program the device using a standard serial communication port. The bootloader is not present on small devices and is not described in this document.
- **Driver:** A driver is a control program defined by the application developer. It is used to manage the allocation of system resources to start application programs. In this document two drivers are described: ICP and IAP drivers.
- **IAP (In-Application Programming):** IAP is the ability to re-program the Flash memory of a microcontroller while the device is already plugged-in to the application and the application is running.
- **ICP (In-Circuit Programming):** ICP is the ability to program the Flash memory of a microcontroller using the SWIM protocol while the device is plugged-in to the application.
- **ICD (In-Circuit Debugging):** ICD is the ability to debug the user application using the SWIM protocol. The user has the ability to connect the device to a debugger and place breakpoints in his firmware. Debugging may be intrusive (application patched to allow debugging) or non intrusive (using a debug module).
- **MASS Keys:** The Memory Access Security System (MASS) consists of a memory write protection lock designed to prevent unwanted memory modifications due to EMS or program counter loss. To unlock the memory protection, one or more keys must be written in a dedicated register and in a specific order. When the operation (write or erase) is completed, the MASS must be activated again to provide good memory security.
- **RWW (Read-While-Write):** The RWW feature provides the ability for the user to perform write operation on data EEPROM while reading and executing the program memory. Execution time is therefore optimized. The opposite operation is not allowed: the user cannot read data memory while writing program memory.
- **Pages:** A page is a set of blocks. The page size is 4 blocks , that is 512 bytes. Refer to *Table 1: Memory partition on page 8*. The boot area size, which is programmable by option byte, can be changed by incrementing one page.
- **SWIM (Single Wire Interface Module):** The SWIM is a communication protocol managed by hardware in the STM8 microcontrollers. The SWIM main purpose is to provide non intrusive debug capability. It can also be used to download programs into RAM and execute them. It can also write (registers or RAM) or read any part of the memory space and jump to any memory address. The SWIM protocol is used for ICP. It is accessed by providing a specific sequence on the SWIM pin either during the reset phase or when the device is running (if allowed by the application).
- **System memory**: The STM8 system memory is a small ROM accessible when the user application is executed. It contains the bootloader.

- **User mode**: The user mode is the standard user application running mode in the STM8. It is entered by means of the $\overline{\text{RESET}}$ pin and without any specific reset sequence.
- **Word**: A word is a set of 4 bytes and corresponds to the memory granularity.

1 Memory protection strategy

The STM8 Flash protection strategy is based on 3 different mechanisms:

- **Read-Out Protection:** The user is able to protect the program and data stored in the Flash memory from being read. To do so, AAh must be written in the ROP option byte using the ICP method.
- **User Boot Code area protection:** In order to guaranty the capability to recover from an interrupted or erroneous IAP programming, STM8 devices provide a write protected area called Boot (or user's Boot) area. The size of this area is programmable by option byte. It can be defined from 0 to the full memory size.
- **Unwanted memory access protection:** The unwanted memory access protection purpose is to prevent from unexpected memory modification by firmware write access (for example due to a firmware bug or EMC disturbance).
  This protection consists of authorizing write access to the memory only by a specific software sequence unlikely to happen randomly or by mistake. It includes writing a MASS (Memory Access Security System) key into a key register to enable access to the Flash and EEPROM memory.

1.1 Read-out protection

The read out protection can be set by writing AAh in the first option byte (ROP). This must be done by ICP or by software. It can be removed only by ICP.

When the read-out protection is removed, the entire Flash memory is erased. Readout protection, when selected, provides a protection against program memory content extraction and against write access to Flash memory. When available, the data EEPROM memory is also protected against read and write access through ICP.

Even though no protection can be considered as totally unbreakable, the feature provides a very high level of protection for a general purpose microcontroller. Of course, a user application that permits the user to dump the Flash memory contents renders this read-out protection useless. *Table 3* describes the capability of the Flash memory sectors versus the different modes and read-out protection settings.

Caution: When the read-out protection is set, STMicroelectronics is not able to disable this option without erasing the Flash content.

1.2 User Boot Code area protection

Whatever the memory contents, it is always possible to restart an ICP session after a critical error by applying a reset and restarting the SWIM communication.

On the contrary, during IAP sessions, the programming software driver must always be write protected to be able to recover from any critical failure that could happen during programming (such as power failure for example).

The pages where the IAP driver is implemented must be in the write protected boot area. This boot area size is defined in the User Boot Code (UBC) area option byte. This option byte is slightly different from one product to another.

The application reset and interrupt vectors and the reset routine as well as the IAP driver must be in the write protected/boot area. These conditions enable the user application to manage the recovery from potential critical failure by applying a reset and restarting the IAP routine from the protected boot area.

1.3 Unwanted memory access protection

The unwanted memory access protection consists of writing two 8-bit keys in the right order into the dedicated MASS key register.

Writing the correct sequence of keys in the program memory MASS key register (FLASH_PUKR) enables the programming of the program memory area excluding the boot section. If wrong keys are inserted, a reset needs to be generated to be able to reinsert the correct key sequence. Once write memory protection is removed, it is possible to reactivate the protection of the area by resetting the PUL bit in FLASH_IAPSR.

To enable write access to the data area and the option byte area, another specific MASS key register (FLASH_DUKR) and a different key sequence must be used. Once the data/option memory is unlocked, it is possible to reactivate the protection of the area by resetting the DUL bit in FLASH_IAPSR.

In order to be as effective as possible, the application software need to lock again the unwanted memory access protection as soon as the programming is completed. Otherwise, the protection level of the MASS is significantly reduced. To activate the MASS protection again, the user must reset the corresponding bit in the FLASH_IAPSR register (bit 3 = DUL bit for data or bit 1 = PUL bit for program memory).

2 Programming STM8 Flash MCUs

2.1 Introduction

This section describes how to program the STM8 Single-Voltage Flash microcontrollers (MCUs).

2.2 Flash organization

The Flash program memory of the STM8 devices is composed of 128 Kbytes of program memory organized in 256 pages of 4 blocks of 128 bytes. It also contains 2K of data EEPROM organized in 16 blocks of 128 bytes.

A memory accelerator takes advantage of the parallel 4-byte storage, which corresponds to a word.

The Flash is mapped in the upper part of the STM8 addressing space and includes the reset and interrupt vector area.

The Flash memory can be erased and programmed either byte per byte, by word or by block.

In word programming mode, 4 bytes can be programmed or erased during the same cycle, and in block programming mode, 128 bytes at the same cycle.

An additional area of Flash memory is used to store the option bytes of the device. These bytes can only be modified using either SWIM, IAP or the bootloader.

Table 1. Memory partition

Area	Page number	Block number	Address
Data EEPROM	0	0	4000h-407Fh
		1	4080h-40FFh
		2	4100h-417Fh
		3	4180h-41FFh
	1	4 to 7	4200h-43FFh
	2	8 to 11	4400h-45FFh
	3	12 to 15	4600h-47FFh
Option bytes	-	0 (one block only)	4800h-487Fh
Program memory	0	0	8000h-807Fh
		1	8080h-80FFh
		2	8100h-817Fh
		3	8180h-81FFh
	1	4 to 7	8200h-83FFh
	2	8 to 11	8400h-85FFh
		...	...
	255		
		1021	27E80h-27EFFh
		1022	27F00h-27F7Fh
		1023	27F80h-27FFFh

2.3 Flash programming

2.3.1 Unlocking the Memory Access Security System (MASS)

Before any erase or write operation, the memory must be unlocked.

Follow the procedure described in *Chapter 1.3: Unwanted memory access protection on page 6*. The register addresses and key values are summarized in *Table 2*.

Table 2. MASS

Memory		STM8
Data EEPROM	Unlock	Write AEh then 56h in FLASH_DUKR (00 4841h)
	Lock	Reset bit 3 (DUL) in FLASH_IAPSR (00 505Fh)
Option bytes	Unlock	Write AEh then 56h in FLASH_DUKR (00 5064h)
	Lock	Reset bit 3 (DUL) in FLASH_IAPSR (00 505Fh)
Program memory	Unlock	Write 56h then AEh in FLASH_PUKR (00 5062h)
	Lock	Reset bit 1 (PUL) in FLASH_IAPSR (00 505Fh)

2.3.2 Byte programming

Byte programming is done by executing any write instruction (ld, mov...) to a Flash memory address when the memory is unlocked. The write instruction initiates the erase/programming cycle and any core access to the memory is blocked until the cycle is over. This means that program execution from Flash is stopped until the end of the erase/programming cycle.

At the end of the programming, the EOP bit in the FLASH_IAPSR is set and the program execution restarts from the instruction following the write/erase instruction.

Contrary to byte programming of the Flash memory, the byte programming of Data EEPROM with RWW feature (when available) does not stop program execution. Only read or write access to the Data EEPROM is blocked until the end of programming. The EOP bit can then be used in order to know if the previous operation is completed. This bit is automatically reset when reading FLASH_IAPSR but is not reset when a new program/erase operation is started.

The erase/programming cycle lasts longer if the location is not yet empty because in this case an erase operation is performed automatically. If the location is empty, the erase operation is not performed. However, if a defined programming time is wanted, the FIX bit in the FLASH_CR1 register forces the programming operation to always erase first whatever the contents of the memory. Therefore a fixed programming time is guaranteed (erase time + write time).

To erase a location, just write '0' to this location.

2.3.3 Block/word programming

A block operation combines the erase/programming cycles of all bytes composing a block or a word.

There are three possible block operations: erase, write only (also called fast programming) and combined erase/write cycle (also called standard block programming).

For any of these operations, the memory must be unlocked first.

In user mode, the execution of a block operation cannot start from the program memory. It must start from RAM:

- If the block operation is performed to the Flash program memory, the program must continue execution in the RAM.
- If the block operation is not performed on the Flash program memory, the user can go back to the Flash program memory (after he has checked the HVOFF bit in the FLASH_IAPSR register) but code execution from the Flash program memory is frozen till the block operation ends.

The programming can also be performed directly through the SWIM.

If the SWIM is used, in order to prevent the core from accessing the Flash memory during the block programming or erase operation, it is recommended to STALL it. This can be done by setting the STALL bit in the DM_CSR2 debug module register. Refer to the STM8 SWIM communication protocol and debug module (UM0470) for more information.

Caution: FLASH_CR2 and FLASH_NCR2 must be written consecutively to be taken into account. If only one register is set, both are forced to their reset values. This would result in byte operations.

Standard block programming operation

The following steps must be followed to perform a block erase and program operation (standard block programing):

- Unlock the memory if not already done
- Check in the option bytes if the block you want to program is not in the boot area
  If necessary, reprogram the option bytes to allow programming and erasing of the targeted block
- Write 0x01 in FLASH_CR2 and 0xFE in FLASH_NCR2 (PRG/NPRG bits active)
- Write all the data bytes of the block you want to program starting with the very first address of the block. No read or write access to the program memory is allowed during these load operations as they could corrupt the values to program
- The programming cycle starts automatically when the 128 data have been written
- Optional: check the WR_PG_DIS bit in FLASH_IAPSR to verify if the block you want to program is not write protected
- Poll the EOP bit in FLASH_IAPSR register for the end of operations. When the bit goes high, the memory word has been erased and reprogrammed.

Caution: FLASH_CR2 and FLASH_NCR2 must be written consecutively to be taken into account. If only one register is set, both are forced to their reset values. This would result in byte operations.

Fast block programming operation

These are the steps to follow to perform a fast block programming (write with no erase) operation:

- Unlock the memory if not already done
- Check in the option bytes if the block you want to erase is not in the boot area.
  If necessary, reprogram the option bytes to allow programming and erasing of the targeted block.
- Write 0x10 in FLASH_CR2 and 0xEF in FLASH_NCR2 (FPRG/NFPRG bits active)
- Write all the data bytes of the block you want to program starting with the very first address of the block. No read or write access to the program memory is allowed during these load operations as they could corrupt the values to program.
- The programming cycle starts automatically when the complete word has been written
- Optional: check the WR_PG_DIS bit in FLASH_IAPSR to verify if the block you want to erase is not write protected
- Poll the EOP bit in FLASH_IAPSR register for the end of operations. When the bit goes high, the memory block has been programmed.

Caution: The memory block must be empty when the 'fast block programming' operation is performed.

Caution: FLASH_CR2 and FLASH_NCR2 must be written consecutively to be taken into account. If only one register is set, both are forced to their reset values. This would result in byte operations.

Block erase operation

The following steps must be followed to perform a block erase operation:

- Unlock the memory if not already done
- Check in the option bytes if the block you want to erase is not in the boot area.
  If necessary, reprogram the option bytes to allow programming and erasing of the targeted block.
- Write 0x20 in FLASH_CR2 and 0xDF in FLASH_NCR2 (ERASE/NERASE bits active)
- Write '0' to the 4 bytes of the very first word of the memory block to erase, that is to the first 4 addresses of the block to erase
- Optional: check the WR_PG_DIS bit in FLASH_IAPSR to verify if the block you tried to erase is not write protected
- Poll the EOP bit in FLASH_IAPSR register for the end of operations. When the bit goes high, the memory block is erased

Caution: To start the ERASE cycle, only the very first word must be written to '0', that is, the first 4 addresses for the STM8. If more memory locations are written, the additional ones are handled as redundant byte erase. If less than one word is cleared, the erase process does not start and the CPU may stall when accessing the memory.

Caution: FLASH_CR2 and FLASH_NCR2 must be written consecutively to be taken into account. If only one register is set, both are forced to their reset values. This would result in byte operations.

Word programming

Word operations can also be done in the same way as block operations writing the following codes:

- Unlock the memory if not already done
- Check in the option bytes if the block you want to erase is not in the boot area.
  If necessary, reprogram the option bytes to allow programming and erasing of the targeted block
- Write 0x40 in FLASH_CR2 and 0xBF in FLASH_NCR2 (WP/NWP bits active)
- Write the 4 data bytes to write to the memory starting with the very first address of the word to program
- The programming cycle starts automatically when the 4 data have been written
- Optional: check the WR_PG_DIS bit in FLASH_IAPSR to verify if the block you want to erase is not write protected
- Poll the EOP bit in FLASH_IAPSR register for the end of operations. When the bit goes to 1, the memory block has been erased and reprogrammed.

Caution: FLASH_CR2 and FLASH_NCR2 must be written consecutively to be taken into account. If only one register is set, both are forced to their reset values. This would result in byte operations.

Note: 1 *Word programming does not require to be executed from RAM.*

2 *If a programming cycle is interrupted (by software or a reset action), the integrity of the data in memory is not guaranteed.*

2.3.4 Option byte programming

Option bytes contain configurations for hardware features of the device as well as the memory protection of the device. They are stored in a dedicated row of the memory. For safety reasons, each option has to be stored twice, once in a regular form and once in a complemented one. If both values are not complemented, the default value is forced, ensuring a safe configuration.

They can also be modified in user mode, except the read-out protection that can only be removed in ICP (using the SWIM).

In ICP mode, the FLASH_CR2 and FLASH_NCR2 registers have an additional configuration bit. To modify the option bytes, these two bits (OPT and NOPT) must be set to their active state (OPT = 1 and NOPT = 0).

Note: *Refer to the device datasheet for more information on Flash registers.*

2.3.5 Memory dedicated option bytes

The Flash memory includes two option bytes dedicated to memory protection. The first one is used to protect the Flash content from being read while the second one is used to write protect the User Boot Code area of the Flash memory of the device.

Read-out protection: as soon as this option is set, it is not possible to read-out the Flash content as shown in *Table 3*. If this option bit is erased in order to re-program the Flash that has been read-out protected, the whole Flash memory is erased first. Note that even though no protection can be considered as totally unbreakable, the feature provides a very high level of protection for a general purpose microcontroller.

User Boot Code area protection: the UBC option byte allows to program the size of the write protected boot area. The boot area always includes the reset and interrupt vectors and can include up to the full program memory size. The boot area size granularity is one page.

Caution: When the read-out protection is set, STMicroelectronics is not able to disable this option without erasing the entire Flash memory.

Table 3. **Memory access table**

MODE	ROP	Memory Area	STM8
User Mode and IAP Bootloader (if available)	Read out protection ENABLED	Boot area	R/E
		Flash memory	R/W/E [(1)]
		Data area	R/W/E [(2)]
		Option bytes	R
	Read Out Protection DISABLED	Boot area	R/E
		Flash memory	R/W/E [(1)]
		Data area	R/W/E [(2)]
		Option bytes	R/W [(2)]
ICP and SWIM	Read out protection ENABLED	Boot area	P
		Flash memory	P
		Data area	P
		Option bytes	P/W_{ROP} [(3)]
	Read Out Protection DISABLED	Boot area	R/E
		Flash memory	R/W/E [(1)]
		Data area	R/W/E [(2)]
		Option bytes	R/W [(2)]

1. The Flash memory is write protected (locked) until the correct MASS key is written in the FLASH_PUKR. It is possible to lock the memory again by resetting the PUL bit in the IAPSR register. Unlocking can only be done once between two resets.
2. The data memory is write protected (locked) until the correct MASS key is written in the FLASH_DUKR. It is possible to lock the memory again by resetting the DUL bit in the IAPSR register.
3. When ROP is removed, the whole memory is erased, including option bytes.

Note: *The following abbreviations are used in the previous table:*
NA: memory range not available
P: Protected, Read Write and Execute is impossible
R: Read possible, Write or Execute impossible
R/E: Read or Execute is possible, Write is impossible
R/W/E: Read, Write and Execute is possible
R/W_{ROP}: Read Possible, only the ROP option can be written [(3)].
P/W_{ROP}: Protected, only the ROP option can be written [(3)].

2.3.6 ICP method

The programming interface for STM8 devices is the SWIM (Single Wire Interface Module). It is used to communicate with an external programming device connected via a cable.

See “STM8 SWIM communication protocol and debug module” (UM0470) for more details on the SWIM mode entry and SWIM protocol.

ICP using the SWIM protocol

When using the SWIM protocol, two methods can be used:

- **First method**

1. Apply a RESET
2. Activate the SWIM by sending the entry sequence on the SWIM pin
3. Activate the SWIM_CSR register by writing 1 to the DM bit in SWIM_CSR
4. Disable interrupts by setting the SAFE_MASK bit in SWIM_CSR
5. Release RESET
6. Verify the DeviceID by reading it using ROTF command
7. Unlock the memory (by writing the MASS key)
8. Send the SWIM SRST command
9. Download the ICP driver software into the device RAM using the SWIM WOTF command
10. Execute the ICP driver using the SWIM monitor:
    a) Modify the CPU registers (new PC, X, Y, CC...) using the WOTF commands
    b) set the FLUSH bit in the DM_CSR2 register
    c) clear the STALL bit in the DM_CSR2 register

- **Second method**

It consists of writing directly into the Flash registers and memory locations through the write memory command of the SWIM protocol. It uses the same operation sequence (point 1 to 9), but instead of firmware download, the tool can directly access the registers of the microcontroller. During block Flash programming, in order to make sure the core is not accessing the memory, it must be STALLED by setting the STALL bit in the DM_CSR2 debug module register.

2.3.7 IAP method

This method will be described in a future revision of this document.

3 Revision history

Table 4. Document revision history

Date	Revision	Changes
17-Jan-2008	1	Initial release