PM0212
Programming manual

How to program the STM8TL5xxx Flash program memory and data EEPROM

Introduction

This manual describes how to program Flash program memory and data EEPROM on STM8TL5xxx microcontrollers. It applies to STM8TL5xxx devices. It is intended to provide information to the programming tool manufacturers and to the customers who want to implement programming by themselves on their production line.

The in-circuit programming (ICP) method is used to update the content of Flash program memory and data EEPROM while the user software 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) can use any communication interface supported by the microcontroller (I/Os, SPI, USART, I^2C, USB, CAN...). IAP has been implemented for users who want their application software to update itself by re-programming the Flash program memory during program execution. The main advantage of IAP is its ability to re-program Flash program memory and data EEPROM when the chip has already been soldered on the application board and while the user software is running. Nevertheless, part of the Flash program memory has to be previously programmed using ICP.

For details on memory implementation and features, registers or stack top addresses, refer to the product datasheet.

Related documents

- STM8TL5xxx microcontroller family reference manual (RM0312)
- STM8TL5xxx datasheet
- STM8 SWIM communication protocol and debug module (UM0470)
- Basic in-application programming example using the STM8 I^2C and SPI peripherals AN2737)

Contents

List of tables

1 Glossary

This section gives a brief definition of acronyms and terms used in this document:

- **Block**

  A block is a set of bytes that can be programmed or erased in one single programming operation. Operations that are available on a block are fast programming, erase only, and standard programming (which includes an erase operation). Refer to *Section 2: Memory organization* for details on block size according to the device.

- **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.

- **In-application programming (IAP)**

  IAP is the ability to re-program the Flash program memory and data EEPROM (DATA) of a microcontroller while the device is already plugged-in to the application and the application is running.

- **In-circuit programming (ICP)**

  ICP is the ability to program the Flash program memory and data EEPROM of a microcontroller using the SWIM protocol while the device is plugged-in to the application.

- **In-circuit debugging (ICD)**

  ICD is the ability to debug the user software using the SWIM protocol. The user has the ability to connect the device to a debugger and insert breakpoints in his firmware. Debugging may be intrusive (application patched to allow debugging) or non intrusive (using a debug module).

- **Memory access security system (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.

- **Page**

  A page is a set of blocks. The number of blocks in a page may differ from one device to another. Refer to *Section 2: Memory organization* for details on page size according to the device.

  Dedicated option bytes can be used to configure by increments of one page the size of the user boot code, the proprietary code, and the data EEPROM.

- **Proprietary code area (PCODE)**

  The proprietary code area (PCODE) can be used to protect proprietary software libraries used to drive peripherals. Refer to *Section 3.2: Proprietary code area protection* for details.

- **Single wire interface module (SWIM)**

  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).

- **User boot code area (UBC)**

  The user boot code area is a write-protected area which contains reset vector, interrupt vectors, and IAP routine for the device to be able to recover from interrupted or erroneous IAP programming.

- **User mode**

  The user mode is the standard user software running mode in the STM8. It is entered either by performing a power-on-reset on the device or by issuing the SWIM SRST command from a development tool.

- **Word**

  A word is a set of 4 bytes and corresponds to the memory granularity.

2 Memory organization

STM8TL5xxx microcontrollers feature up to 16 Kbytes of Flash program memory including up to 2 Kbytes of data EEPROM.

A memory accelerator takes advantage of the parallel 4-byte storage, which corresponds to a word. The Flash program memory and data EEPROM can be erased and programmed at byte level, word level or block level. In word programming mode, 4 bytes are programmed/erased during the same cycle, while in block programming mode, a whole block is programmed/erase during the same cycle.

2.1 Memory features

The STM8TL5xxx memory features are as follows:

- Up to 16 Kbytes of Flash program memory including up to 2 Kbytes of data EEPROM. The whole memory array is divided into 256 pages of one block (64 bytes) each.

  The Flash program memory is divided into 3 areas:

  – The user boot code area (UBC)
  – A configurable data EEPROM area (DATA)
  – The main program area

  The DATA and main program areas can be write protected independently by using the memory access security mechanism (MASS).

  The size of UBC, and DATA areas can be configured through option bytes.
- One block (64 bytes) of option bytes of which 5 bytes are already used to configure device hardware features. The option bytes can be programmed only in ICP/SWIM mode.

Refer to *Table 1: STM8TL5xxx memory partitions* for a detailed description of STM8TL5xxx memory partitioning.

Table 1. STM8TL5xxx memory partitions(1)

Area			Page/block number (1 page = 1 block)	Address
Option byte			0	0x00 4800-0x00 483F
Flash program memory	UBC, PCODE and main program memory		0	0x00 8000-0x00 803F
			1	0x00 8040-0x00 807F
			2	0x00 8080-0x00 80BF
			3	0x00 80C0-0x00 80FF
			…	…
			95	…
		Configurable data EEPROM(2)	224	0x00 B800 0x00 B83F
			-	…
				0x00BF80-0x00BFBF
	Main program memory		Up to 255	0x00BFC0-0x00BFFF

1. The memory mapping is given for the devices featuring 16 Kbytes of Flash program memory including up to 2 Kbytes of data EEPROM.
2. The size of the data EEPROM area is configurable from 0 to 32 pages starting from the last page of the Flash program memory down to page 224.

3 Memory protection strategy

The STM8 devices feature several mechanisms allowing to protect the content of the Flash program and data EEPROM areas:

- **Readout protection**

  The software can prevent application code and data stored in the Flash program memory and data EEPROM from being read and modified in ICP/SWIM mode. The readout protection is enabled and disabled by programming an option byte in ICP/SWIM mode. Refer to *Section 3.1: Readout protection* for details.

- **Proprietary code area (PCODE)**

  To protect proprietary peripheral software driver libraries, some STM8 devices features a permanently readout protected area, the proprietary code area (PCODE). This area is part of the Flash program memory. Its content cannot be modified and can only be read/executed in user privileged mode.

  The size of the PCODE area can be configured in ICP/SWIM mode through an option byte by increments of one page.

  Refer to *Section 3.2* for details on PCODE area.

- **User boot code area (UBC)**

  In order to guaranty the capability to recover from an interrupted or erroneous IAP programming, all STM8 devices provide a write-protected area called user boot code (UBC). This area is a part of the Flash program memory which cannot be modified in user mode (that is protected against modification by the user software). The content of the UBC area can be modified only in ICP/SWIM mode after clearing the UBC option byte.

  The size of the user boot code area can be configured through an option byte by increments of one page.

  Refer to *Section 3.3: User Boot Code area protection* for details on user boot code area.

- **Unwanted memory access protection**

  All STM8 devices offer unwanted memory access protection, which purpose is to prevent unintentional modification of program memory and data EEPROM (for example due to a firmware bug or EMC disturbance).

  This protection consists of authorizing write access to the memory only through a specific software sequence which is unlikely to happen randomly or by mistake. Access to Flash program and data EEPROM areas is enabled by writing MASS keys into key registers.

  Refer to *Section 3.4: Unwanted memory access protection* for details on unwanted memory access protection.

3.1 Readout protection

On STM8TL5xxx microcontrollers, the readout protection is set by writing any value except for 0xAA in the ROP option byte. It is disabled by reprogramming the ROP option byte with 0xAA, and resetting the device. To unprotect the device, the ROP must be written 2 times: the first writing, with any value, launches a global erase which includes the option byte; the second writing, AA, unprotects the device, and then the AA value remains in the ROP byte. Please note the EOP should be checked each time.

The readout protection can only be disabled in ICP/SWIM mode.

When the readout protection is selected, reading or modifying the Flash program memory in ICP mode (using the SWIM interface) is forbidden. The data EEPROM memory is also protected against read and write access through ICP.

Erasing the ROP option byte to disable the readout protection causes the Flash program memory, the DATA area and the option bytes to be erased.

Even though no protection can be considered as totally unbreakable, the readout protection feature provides a very high level of protection for general purpose microcontrollers. Of course, a software that allows the user to dump the Flash program memory content make this readout protection useless. *Table 5: Memory access versus programming method* describes possible accesses to each memory areas versus the different modes and readout protection settings.

3.2 Proprietary code area protection

The memory pages containing peripherals software libraries must be located in the proprietary code area (PCODE).

The size of the PCODE area can be configured through the PCODE option byte (PCODESIZE) in ICP/SWIM mode. Refer to *Table 2* for details on PCODE minimum and maximum size. Once programmed, the PCODE option byte cannot be erased, the size of the PCODE area remains fixed and its content protected from read and write operations whatever the mode.

Table 2. PCODE size

PCODESIZE[0:7]	Interrupt vector table	Protected page	Protected addresses
0	All interrupt vectors can be written	0	None
3	Only TRAP is write-protected	1	0x8080-0x80BF
4	Only TRAP is write-protected	2	0x8080-0x80FF
...			
0xFF	Only TRAP is write-protected	253	0x8080-0xBFBF

Note: Values 1 and 2 do not protect any address, but the PCODE option byte is write-protected.

Except for the interrupt vectors which can be read directly, the PCODE area can be read only through TRAP interrupt in ICP/SWIM (with readout protection disabled), user and IAP mode.

When the PCODE option byte is set, the PCODE area and the TRAP vector are write protected to prevent a malicious user/program from inserting a “dump” routine inside the protected code.

When accessing the PCODE area, no other program can be executed (all interrupt and debug access are disabled). The programmer must consequently ensure that the protected code embedded in the PCODE area gives back control to the main/end user software at reasonable periods of time.

3.3 User Boot Code area protection

Whatever the memory content, 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 might happen during programming (such as power failure).

The pages where the IAP driver is implemented must be located in the write-protected boot code area (UBC). The application reset and interrupt vectors and the reset routine must also be stored in the UBC. These conditions allow the user software to manage the recovery from potential critical failure by applying a reset and restarting the IAP routine from the protected boot area.

The UBC size is defined by the user boot code (UBC) area option byte. See the following table for the minimum and maximum size of the UBC area.

Table 3. Recommended minimum and maximum sizes of the UBC area

Recommended minimum size of the UBC area	Maximum size of the UBC area
3 pages = 192 bytes	255 pages

3.4 Unwanted memory access protection

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

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 UBC and the PCODE area. If wrong keys are provided, a reset must to be generated to be able to reprogram the right keys.

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

If wrong keys have been provided to the FLASH_PUKR register, the device must be reset before performing a new key program sequence. However, when wrong keys are provided to

the FLASH_DUKR register, new keys can be entered without the device being previously reset.

The size of the DATA area can be configured through the DATASIZE option byte.

In order to be as effective as possible, the application software must 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 bits in the FLASH_IAPSR register (DUL bit for data EEPROM or PUL bit for Flash program memory).

Note: 1 The mechanism to lock and unlock unwanted memory access protection is identical for option bytes and data EEPROM (see Table 4: MASS).

2 Before starting programming program memory or data EEPROM, the software must verify that the area is not write protected by checking that the PUL or DUL bit is effectively set.

4 Programming STM8 Flash microcontrollers

This section describes how to program STM8 single-voltage Flash microcontrollers.

4.1 Unlocking the Memory Access Security System (MASS)

The memory must be unlocked before attempting to perform any erase or write operation. To unlock it, follow the procedure described in *Section 3.4: Unwanted memory access protection*, and *Table 4: MASS*.

The software must poll the PUL and DUL bit, before attempting to write to program memory and data EEPROM, respectively.

Table 4. MASS

Microcontroller family	Data EEPROM and option bytes		Program memory	
	Unlock	Lock	Unlock	Lock
STM8TL5xxx	Write 0xAE then 56h in FLASH_DUKR (0x00 5053)(1)(2)	Reset bit 3 (DUL) in FLASH_IAPSR (0x00 5054)	Write 0x56 then 0xAE in FLASH_PUKR (0x00 5052)(3)	Reset bit 1 (PUL) in FLASH_IAPSR (0x00 5054)

1. In STM8TL5xxx devices, the option bytes are not accessible in user/IAP mode.
2. The OPT bit of the FLASH_CR2 register must be set/cleared to enable access to the option bytes.
3. If wrong keys have been entered, a reset must to be generated to be able to reprogram the right keys.

4.2 Block programming

Block write operations allow to program an entire block in one shot, thus minimizing the programming time.

There are three possible block programming modes: erase, write only (also called fast programming) and combined erase/write cycle (also called standard block programming). The programming mode is selected through FLASH_CR2 register.

The memory must be unlocked before performing any of these operations.

Block program operations can be performed both to main program memory and DATA area:.

- Programming a block of main program memory:

  The block program operation has to be executed totally from RAM.

  The program execution continues from RAM. If the program goes back to main program memory, it is stalled until the block program operation is complete. The DMA controller can be programmed to perform a block transfer to Flash program memory, and put the CPU in Wait mode.

- Programming a block of data EEPROM:

  The block program operation must be executed totally from RAM.

The programming can also be performed directly through the SWIM interface. In this case, it is recommended to stall the device in order to prevent the core from accessing the Flash program memory during the block program or erase operation. 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: During a block program or erase operation, it is recommended to avoid executing instructions performing a read access to program memory.

Caution: If the number of written memory locations is higher than what is required in the block program/erase sequence, the additional locations are handled as redundant byte write operations.

If the number of written memory locations is lower than what is specified in the block program/erase sequence, the block program/erase process does not start and the CPU stalls waiting for the remaining operations to be performed.

Caution: EOP and WR_PG_DIS bits of FLASH_IAPSR register are automatically cleared when a program/erase operation starts.

Caution: If a block program or erase sequence is interrupted by a reset, the data programmed in the memory may be corrupted.

Standard block programming

The following sequence is required to perform a standard block program sequence (block erased and programmed):

1. Unlock the memory if not already done.

   The UBC option byte can be read to check if the block to program is not in the UBC area. If necessary, reprogram it to allow erasing and programming the targeted block.
2. Write 0x01 in FLASH_CR2 (PRG bit active).
3. 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 might corrupt the values to be programmed.

   The programming cycle starts automatically when all the data in the block have been written.
4. Check the WR_PG_DIS bit in FLASH_IAPSR to verify if the block you attempted to program was not write protected (optional)
5. To check if the program operation is complete, poll the EOP bit in FLASH_IAPSR from program memory. EOP is set to ‘1’ when the standard block program operation has completed. To avoid polling the EOP bit, an interrupt can be generated when EOP is set.

Note: It is mandatory to execute steps 2 to 4 from RAM.

Caution: EOP and WR_PG_DIS bits are cleared by reading the FLASH_IAPSR register. It is consequently strongly recommended to perform one single read operation to the FLASH_IAPSR register to check the values of these bits.

Fast block programming operation

The following sequence is required to perform a fast block program sequence (block programmed without previous erase):

1. Unlock the memory if not already done.

   The UBC option byte can be read to check if the block to program is not in the UBC area. If necessary, reprogram it to allow programming the targeted block.
2. Write 0x10 in FLASH_CR2 (FPRG bit active).
3. 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 might corrupt the values to be programmed.

   The programming cycle starts automatically when the complete block has been written.
4. Check the WR_PG_DIS bit in FLASH_IAPSR to verify if the block you attempted to program was not write protected (optional).
5. To check if the program operation is complete, poll the EOP bit in FLASH_IAPSR from program memory. EOP is set to ‘1’ when the block program operation has completed. To avoid polling the EOP bit, an interrupt can be generated when EOP is set.

Note: It is mandatory to execute steps 2 to 4 from RAM.

Caution: EOP and WR_PG_DIS bits are cleared by reading the FLASH_IAPSR register. It is consequently strongly recommended to perform one single read operation to the FLASH_IAPSR register to check the values of these bits.

Caution: The memory block must be empty when performing a fast block programming operation.

Block erase operation

The following sequence is required to perform a block erase sequence:

1. Unlock the memory if not already done.

   The UBC option byte can be read to check if the block to erase is not in the UBC area. If necessary, reprogram it to allow erasing the targeted block.
2. Write 0x20 in FLASH_CR2 (ERASE bit active).
3. Write '0x00 00 00 00' to any word inside the block to be erased using a LOAD instruction.
4. Check the WR_PG_DIS bit in FLASH_IAPSR to verify if the block you attempted to erase was not write protected (optional).
5. To check if the erase operation is complete, poll the EOP bit in FLASH_IAPSR from program memory. EOP is set to ‘1’ when the block erase operation has completed. To avoid polling the EOP bit, an interrupt can be generated when EOP is set.

Note: It is mandatory to execute steps 2 to 4 from RAM.

Caution: EOP and WR_PG_DIS bits are cleared by reading the FLASH_IAPSR register. It is consequently strongly recommended to perform one single read operation to the FLASH_IAPSR register to check the values of these bits.

4.3 Word programming

Both main program memory and data EEPROM can be programmed and erased at word level. Word operations are performed in the same way as block operations. They can be executed either from program memory or from RAM.

When a new word program operation starts, EOP and WR_PG_DIS bits of FLASH_IAPSR register are automatically cleared.

The EOP bit can then be used to know if the previous operation has completed. This bit is automatically reset when reading FLASH_IAPSR.

The following sequence is required to perform a word program operation:

1. Unlock the memory if not already done.

   The UBC option byte can be read to check if the word you want to program is not in the UBC area. If necessary, reprogram it to allow programming the targeted word.
2. Write 0x40 in FLASH_CR2 (WP bit active).
3. Write the 4 data bytes to the memory starting with the very first address of the word to be programmed.

   The programming cycle starts automatically when the 4 bytes have been written.
4. Check the WR_PG_DIS bit in FLASH_IAPSR to verify if the word you attempted to program was not write-protected (optional).
5. To check if the program operation is complete, poll the EOP bit in FLASH_IAPSR register for the end of operations. EOP is set to ‘1’ when the word program operation has completed. To avoid polling the EOP bit, an interrupt can be generated when EOP is set.

Caution: EOP and WR_PG_DIS bits are cleared by reading the FLASH_IAPSR register. It is consequently strongly recommended to perform one single read operation to the FLASH_IAPSR register to check the values of these bits.

Caution: If a word program operation is interrupted by a reset, the data programmed in the memory may be corrupted.

4.4 Byte programming

Both main program memory and data EEPROM can be programmed and erased at byte level.

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

When a new byte program operation starts, EOP and WR_PG_DIS bits of FLASH_IAPSR register are automatically cleared. At the end of the program operation, the EOP bit in the FLASH_IAPSR register is set and the program execution restarts from the instruction following the write/erase instruction.

The EOP bit can then be used in order to know if the previous operation has completed. To avoid polling the EOP bit, an interrupt can be generated when EOP is set. This bit is automatically reset when reading FLASH_IAPSR.

The erase/program cycle lasts longer if the whole word containing the byte to be programmed is not empty because in this case an erase operation is performed automatically. If the word 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 byte location, just write '0x00' to the byte location.

Caution: A byte programming operation performs a word (4-byte) access to the Flash program memory. If a byte program operation is interrupted by a reset, the 4 bytes programmed in the memory may be corrupted.

4.5 Programming the option bytes

Option bytes are used to configure the device hardware features as well as the memory protection. They are stored in a dedicated memory block.

4.5.1 Summary of memory dedicated option bytes

The Flash program memory includes several option bytes dedicated to memory protection:

- ROP

  The ROP option byte is used to prevent the Flash program memory from being read and modified in ICP/SWIM mode. Refer to *Section 3.1: Readout protection* for a detailed description of readout protection.

- PCODESIZE

  The PCODESIZE option byte is used to configure the size of the proprietary code area (PCODE) which can be used to store proprietary software libraries. The minimum size of the proprietary code area is of 1 page (64 bytes) and the maximum size of 253 pages. The PCODESIZE option byte can be modified only in ICP/SWIM mode.

  Refer to *Section 3.2: Proprietary code area protection* for a detailed description of the PCODE area and PCODESIZE option byte.

- UBC

  The UBC option byte is used to program the size of the write protected user boot code area. The boot area always includes the reset and interrupt vectors and can go up to the full program memory size. The boot area size granularity is of one page.

  Refer to *Section 3.3: User Boot Code area protection* for a detailed description of the UBC area.

- DATASIZE

  The DATASIZE option byte is used to configure the size of the data EEPROM area.

  This option byte specifies the number of pages starting from the end of the memory. The maximum size of the data EEPROM area is of 2 Kbytes.

  Refer to STM8TL5xxx datasheet for details on DATASIZE value programming.

4.5.2 How to program the option bytes

The option bytes are stored only once.

On STM8TL5xxx devices, the option bytes can be modified only in ICP/SWIM mode with OPT bit of the FLASH_CR2 register set to '1' (refer to the description of FLASH_CR2 register in the reference manual).

4.6 Memory access versus programming method

Table 5 gives a description of possible accesses from the core to memory areas according to the programming method.

Table 5. Memory access versus programming method

Mode	ROP	Memory Area	Access from core(1)
User, IAP	Readout protection enabled	Interrupt vectors except for TRAP	R/W(2)/E
		TRAP	R/W/E
		Proprietary code area (PCODE)	R/E
		User boot code area (UBC)	R/E
		Main program	R/W/E
		Data EEPROM area (DATA)	R/W(3)
		Option bytes	R
	Readout protection disabled	Interrupt vectors except for TRAP	R/W(2)/E
		TRAP	R/W/E
		Proprietary code area (PCODE)	R/E(4)
		User boot code area (UBC)	R/E(5)
		Main program	R/W/E(7)
		Data EEPROM area (DATA)	R/W(3)
		Option bytes	R

Table 5. Memory access versus programming method (continued)

Mode	ROP	Memory Area	Access from core(1)
SWIM active (ICP mode)	Readout protection enabled	Interrupt vectors except for TRAP	P
		TRAP	P
		Proprietary code area (PCODE)	P(4)
		User boot code area (UBC)	P
		Main program	P
		Data EEPROM area (DATA)	P
		Option bytes	P/W_{ROP}(6)
	Readout protection disabled	Interrupt vectors except for TRAP	R/W/E
		TRAP	R/W/E
		Proprietary code area (PCODE)	R/E(4)
		User boot code area (UBC)	R/E(5)
		Main program	R/W/E(7)
		Data EEPROM area (DATA)	R/W(3)
		Option bytes	R/W

1. R/W/E = Read; Write and Execute;
   R/E = Read and Execute (write operation forbidden);
   R = Read (write and execute operations forbidden);
   P = the area cannot be accessed (read, execute and write operations forbidden);
   P/W_{ROP} = Protected, write forbidden except for ROP option byte.
2. When no UBC area has been defined, the interrupt vectors, except for TRAP, can be modified in user /IAP mode.
3. 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. If wrong keys are provided, another key program sequence can be performed without resetting the device.
4. The PCODE area can be read and executed only in privileged mode through the TRAP vector. The PCODE cannot be directly accessed through the SWIM.
5. To program the UBC area the application must first clear the UBC option byte.
6. When ROP is removed, the whole memory is erased, including option bytes.
7. The Flash program 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 FLASH_IAPSR register. If wrong keys are provided, the device must be reset and new keys programmed.

4.6.1 ICP methods

The in-circuit programming (ICP) method is used to update the content of Flash program memory and data EEPROM.

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 user manual (UM0470) for more details on the SWIM mode entry and SWIM protocol.

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

First method

The first method consists of writing directly into the Flash registers and memory locations through the write memory command of the SWIM protocol. To make sure that the CPU is not accessing the memory during block Flash programming, the core must be stalled by setting the STALL bit in the DM_CSR2 debug module register.

The following sequence is required:

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. Send the SWIM SRST command
8. Unlock the memory by writing the MASS keys
9. Program the Flash program memory using the SWIM WOTF command

Second method

The second method uses the same sequence of operations as the first method except that the ICP driver firmware must be downloaded in RAM before being launched:

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. Send the SWIM SRST command
8. Unlock the memory by writing the MASS keys
9. Download the ICP driver firmware into the device RAM using the SWIM WOTF command
10. Execute the ICP driver:
    a) Modify the CPU registers (new PC, X, Y, CC...) using the WOTF commands
    b) Set the FLASH bit in the DM_CSR2 register
    c) Clear the STALL bit in the DM_CSR2 register

4.6.2 IAP method

Refer to application note AN2737- Basic in-application programming example using the STM8 I^2C and SPI peripherals.

5 Flash program memory and data EEPROM summary

Table 6. STM8TL5xxx summary

Feature	STM8TL5xxx
Memory	
Block size	64 bytes
Page size	1 block (64 bytes)
Flash program memory	Up to 16 Kbytes
Data EEPROM	Up to 2 Kbytes included in Flash program memory Size configurable by option byte
Proprietary code area (PCODE)	YES
User boot code (UBC)	YES - size configurable by option byte
Option bytes	Programmable in ICP/SWIM 3 memory dedicated option bytes (ROP, UBC, DATASIZE)
Programming/erasing features	
Block programming (fast and standard)(1) Word programming(1) Byte programming(1) Block Erase(1)(2)	YES
Flash control registers	FLASH_CR2

1. Block program/erase sequence must be executed from RAM.
2. Any word in the block programmed to 0.

6 Revision history

Table 7. Document revision history

Date	Revision	Changes
20-Oct-2011	1	Initial release.
10-Apr-2012	2	Replaced “STM8TL53xx” with “STM8TL5xxx”. Added PCODE information throughout the document. Removed all references to TLI. Updated *Table 2: PCODE size*, *Section 4.2: Block programming*, *Section 4.3: Word programming*, *Section 4.4: Byte programming*.